|  |  |
| --- | --- |
|  | Thomas More Campus de Nayer  Elektronica-Embedded Hardware  Jan Pieter de Nayerlaan 5  2860 Sint-Katelijne-Waver |
|  | |
| **Spectrum analyzer** | |
|  | |
| Praktisch Enterprise van: | Stijn Hoeymans |
|  | 2EE |
|  | Schooljaar 2020 - 2021 |

**Voorwoord:**

Ik ben Stijn Hoeymans en ik zit in het 2de jaar Elektronica-Embedded Hardware te Campus de Nayer, en hebben zoals iedereen in deze richting de opdracht om een Praktisch Enterprise (PE) op te stellen.

Dit PE is een belangrijk onderdeel van onze opleiding. Het leert ons om zelf een praktisch en theoretisch project uit te werken op zelfstandige basis. Met dit PE wil ik aantonen dat ik voldoende kennis heb vergaard om zelfstandig projecten aan te pakken in de toekomst. Ik heb dit project gekozen omwille van mijn fascinatie naar audio.

In de eerste plaats wil ik alle leerkrachten bedanken voor hun medewerking en begeleiding bij het maken van de PE.

Tot slot gaat mijn dank voor hun aanmoediging en ondersteuning naar mijn ouders, grootvader en Bart Huyskens

[1 Inleiding 5](#_Toc72617724)

[2 Lastenboek 6](#_Toc72617725)

[2.1 Voeding 6](#_Toc72617726)

[2.2 ADC-filter 6](#_Toc72617727)

[2.3 Signaalverwerking 6](#_Toc72617728)

[2.4 Ledmatrix 6](#_Toc72617729)

[2.5 Equalizer 6](#_Toc72617730)

[2.6 Touch LCD 6](#_Toc72617731)

[2.7 Versterker 6](#_Toc72617732)

[3 Gebruikte principes 7](#_Toc72617733)

[3.1 Microcontroller 7](#_Toc72617734)

[3.2 Het voedingsgedeelte 7](#_Toc72617735)

[3.2.1 De emittervolger met zenerdiode 8](#_Toc72617736)

[3.2.2 De lm317 8](#_Toc72617737)

[3.2.3 De lm337 9](#_Toc72617738)

[3.3 Inlezen van Audio 9](#_Toc72617739)

[3.3.1 De offset 9](#_Toc72617740)

[3.3.2 De verschilversterker 10](#_Toc72617741)

[3.4 Signaalverwerking 10](#_Toc72617742)

[3.4.1 De DFT 10](#_Toc72617743)

[3.4.2 De FFT 10](#_Toc72617744)

[3.4.3 Extra 11](#_Toc72617745)

[3.5 De Ledmatrix 11](#_Toc72617746)

[3.5.1 De rijen 11](#_Toc72617747)

[3.5.2 De kolommen 12](#_Toc72617748)

[3.6 De equalizer 12](#_Toc72617749)

[3.6.1 De digitale potentiometer 12](#_Toc72617750)

[3.7 Touch LCD 13](#_Toc72617751)

[3.8 Volumeregeling 13](#_Toc72617752)

[3.9 25W volumeversterker 14](#_Toc72617753)

[3.9.1 Klasse-D 14](#_Toc72617754)

[4 Risicoanalyse 16](#_Toc72617755)

[4.1 Oververhitten 16](#_Toc72617756)

[4.1.1 Berekening diodebrug 16](#_Toc72617757)

[4.1.2 Berekening koelvin emittervolger (13V) 16](#_Toc72617758)

[4.1.3 Berekening koelvin LM317 & LM337 (13,75V) 16](#_Toc72617759)

[4.1.4 Berekening koelvin LM317 (5V) 16](#_Toc72617760)

[5 Slotwoord 18](#_Toc72617761)

[6 Programma 19](#_Toc72617762)

[7 Bijlage 45](#_Toc72617763)

[7.1.1 Aansluitgegevens Ringkerntransformator 45](#_Toc72617764)

[7.1.2 Schema voeding 45](#_Toc72617766)

[7.1.3 PCB voeding 46](#_Toc72617767)

[7.1.4 Schema ledmatrix 47](#_Toc72617768)

[7.1.5 PCB ledmatrix 47](#_Toc72617769)

[7.1.6 Schema equalizer 47](#_Toc72617770)

[7.1.7 PCB equalizer 48](#_Toc72617771)

[7.1.8 Schema verschilversterker 49](#_Toc72617772)

[7.1.9 PCB verschilversterker 50](#_Toc72617773)

# Inleiding

Ik heb als project gekozen voor het maken van een spectrum analyzer. Audio komt binnen via een AUX-poort die verbonden is met een equalizer en een filter. Vanuit de filter wordt er 1 kanaal, via de ADC, ingelezen door de microcontroller. Die waardes of ook wel samples genoemd, worden dan door de microcontroller verwerkt naar bruikbare waardes. De bruikbare waardes worden vervolgens getoond op een Ledmatrix. De audio dat naar de equalizer gaat, kan daar aangepast worden op zijn bass en treble. Vervolgens gaat het aangepaste signaal naar een versterker waarvan het volume geregeld kan worden.

# Lastenboek

## Voeding

* Voeding die op het net aangesloten kan worden
* Geen schakelende onderdelen
* Transformator zonder middenaftakking
* 13,75VDC,13VDC, 5VDC, -13,75VDC

## ADC-filter

* Geen negatieve waardes
* Verzwakte signalen na 16kHz

## Signaalverwerking

* DFT of FFT

## Ledmatrix

* Duidelijke weergaven
* Minimum 30 frequenties

## Equalizer

* Bass en treble aanpasbaar
* Stereo uitvoering

## Touch LCD

* Duidelijke weergaven

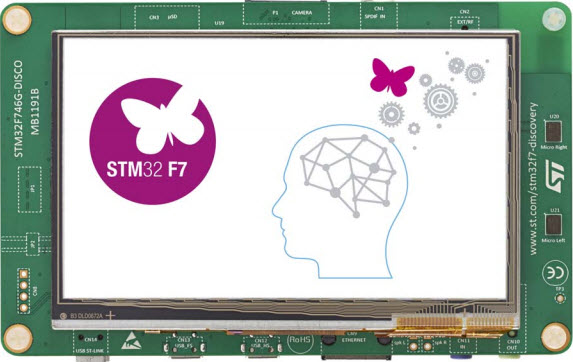
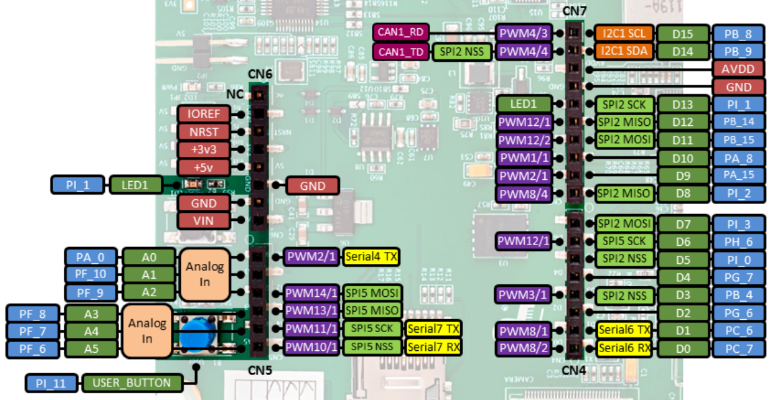
## Versterker

* Stereo versterker
* Klasse D

# Gebruikte principes

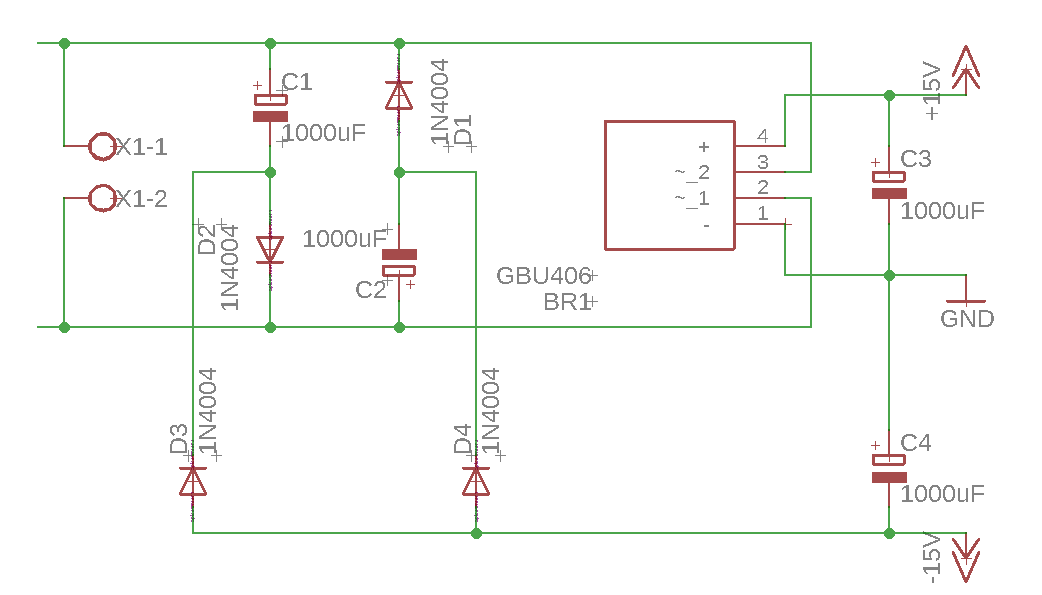
## Microcontroller

In het project wordt er met het stm32f746 bordje gewerkt. Op dit bordje zijn pinnen naar buiten gebracht die compatibel zijn met een arduino shield. Ook is er een display voorzien met touch en externe ram. De externe ram wordt gebruikt voor het verwerken van audio en het maken van een beeld frame voor de display.

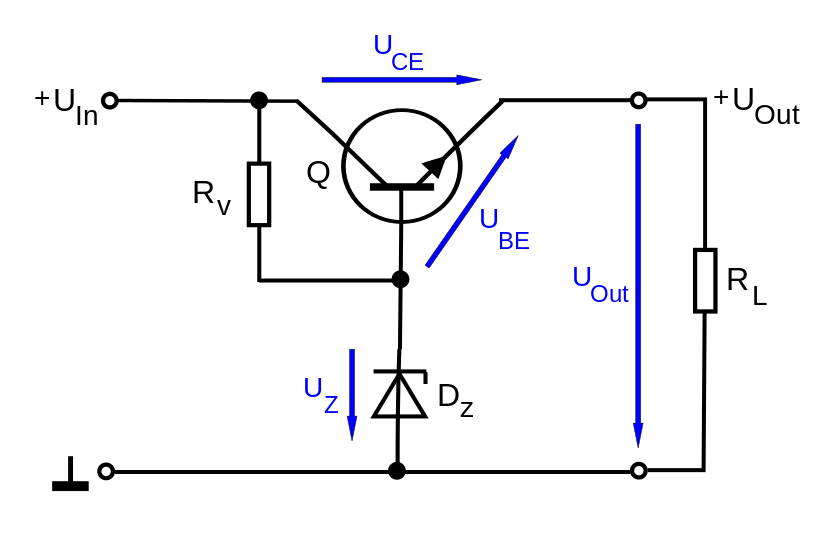


## Het voedingsgedeelte

Het voedingsgedeelte van het geheel bestaat uit de vertrekkende 230VAC. De 230VAC wordt getransformeerd naar 1x12VAC door een ringkerntransformator. Om een positieve gelijkspanning te verkrijgen wordt er een bruggelijkrichter in combinatie met een afvlakcondensator gebruikt. Voor de negatieve gelijkspanning wordt er gebruik gemaakt van diodes en condensators. Wanneer de spanning stijgt laadt de condensator op. De negatieve kant van die condensator zal proberen mee te stijgen. Als de spanning boven de 0,7V gaat dan begint de diode te geleiden. Hierdoor ontstaat er potentieel verschil tussen de positieve en de negatieve kant van de condensator. Wanneer de spanning daalt aan de positieve kant, dan gaat de negatieve kant proberen het potentiaal te behouden. Op de negatieve kant van de condensator zal dan een negatieve spanning aanwezig zijn. Door de negatieve spanning op deze manier te verkrijgen is er geen transformator met middenaftakking nodig. Bij iedere ic, transistor en diodebrug wordt er een koelvin voorzien om de opgewekte warmte te beperken.

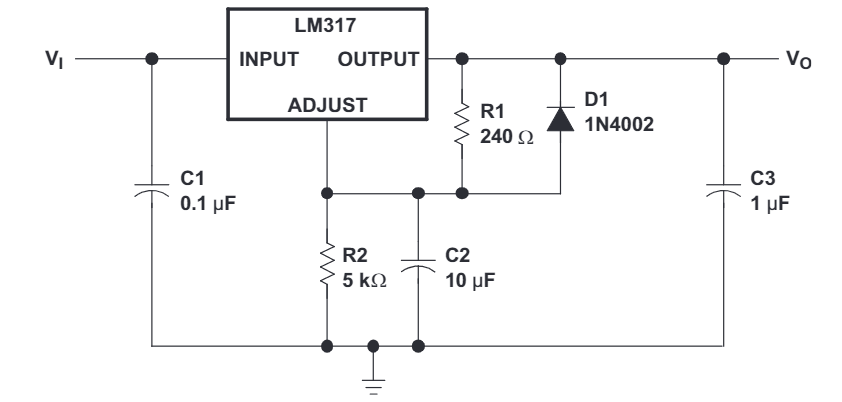


### De emittervolger met zenerdiode

Voor de versterker moet er een spanning van 13VDC aanwezig zijn. Dit wordt verwezenlijkt door een transistor schakeling te maken met een zenderdiode. De zenerdiode bepaalt de spanning over de versterker en de transistor de stroom. Om een minimumstroom te hebben van 1 ampère wordt er gebruik gemaakt van een vermogen transistor waarvan de basisstroom bepaald wordt door een basisweerstand. De 2N3055 is een NPN vermogentransitor die een maximumstroom kan schakelen van 15 ampère.

### De lm317

Voor de 13,75VDC en de 5VDC wordt de lm317 gebruikt. Het is een lineaire spanningsregelaar waarvan de uitgangspanning aanpasbaar is tussen 1,25V en 30V

Om de spanning te bepalen wordt er gebruik gemaakt van de spanningsdeler R2/R1. Om de storing op de uitgangsspanning te verminderen, wordt de condensator C2 voorzien. De diode D1 is om C2 te ontladen wanneer de uitgang naar massa wordt kortgesloten.

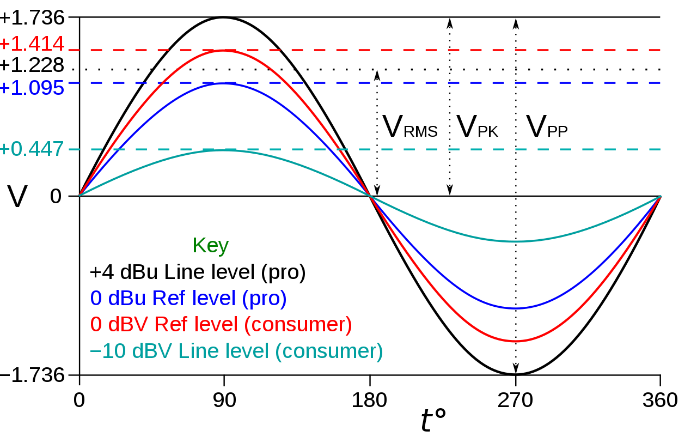
### De lm337

Voor de negatieve spanning, -13,75VDC, wordt de lm337 gebruikt. Deze werkt net zoals de lm317 met de spanningsdeler R2/R1. Bij de lm337 zijn de condensators en de diode tegengesteld gepolariseerd.

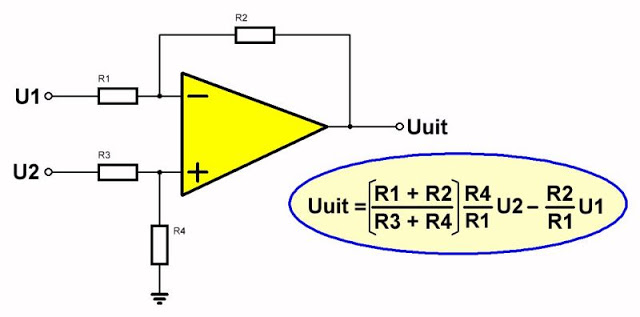
## Inlezen van Audio

Audio is zowel positief als negatief. Dit vormt een probleem wanneer we dit willen inlezen met de microcontroller. Dit kunnen we oplossen door het audiosignaal te voorzien van een offset.

### De offset

Om audio positief te krijgen moet de offset hoger zijn dan 1,414 volt. Voor het project wordt er een spanningsdeler gebruikt die 3,3V deelt door twee (1,65V). De spanning uit de spanningsdeler en die van de audio, gaan vervolgens door een verschilversterker.

### De verschilversterker

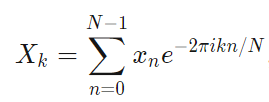
Door gebruik te maken van de verschilversterker wordt de audio (U1), afgetrokken van de spanning uit de spanningsdeler (U2). Door alle weerstanden gelijk te stellen aan elkaar is enkel de offset veranderd maar niet de sterkte van het signaal. Het uitgangsignaal gaat dan door een laagdoorlaatfilter die daalt rond de 16kHz. Het nut van de laagdoorlaatfilter wordt verder in het boek nog besproken.

## Signaalverwerking

Audio is een samenvoeging van verschillende frequenties. Wilt men een frequentie apart met zijn amplitude, dan gaat men een wiskundige algoritme toepassen. Dit kan via een DFT of een FFT. Zowel de DFT als de FFT gebruiken samples. Een sample is vergelijkbaar met een moment foto. Als men in een bepaalde tijdspan een x aantal foto’s neemt, dan kunnen we een beeld maken van het signaal. Het aantal samples wordt bepaald door de hoogste frequentie die men wilt meten.

### De DFT

Om de FFT te begrijpen moeten we beginnen met de DFT. DFT staat voor Discrete Fourier transform, wat betekent eenvoudige fouriertransformatie. De naam ‘fourier’ komt van de persoon (Joseph Fourier) die de wiskunde hierachter heeft uitgevonden.

Een DFT werkt door iedere sample te vermenigvuldigen met een factor. Die uitkomsten worden opgeteld. De uitkomst van de totale wiskundige formule geeft de amplitude voor een frequentie.

N = aantal samples

n = getal tussen 0 en max sample

Xn= nde sample

k = gewenste frequentie

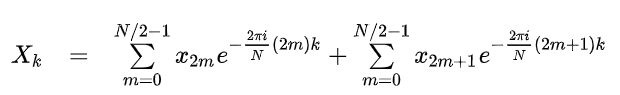
-i = imaginair

e = euler

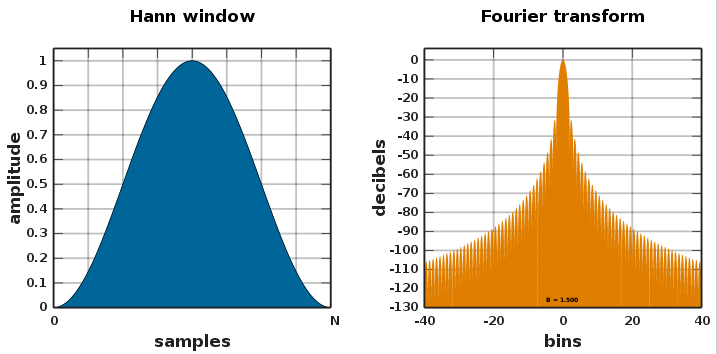
∑ = symbool voor een sommatie (optelling)

### De FFT

De FFT, fast Fourier transform, is een snelle manier om een DFT te berekenen. Dit gebeurt door de formule te splitsen.

In de nieuwe formule is duidelijk zichtbaar dat het aantal optellingen al met de helft gehalveerd is (N/2). Wat ook opvalt is dat in het 1ste deel de even geïndexeerde samples worden gebruikt (2m) en in het andere deel de oneven geïndexeerde samples (2m+1). Als we nu deze splitsing blijven toepassen, dan zal het aantal optellingen ook dalen.

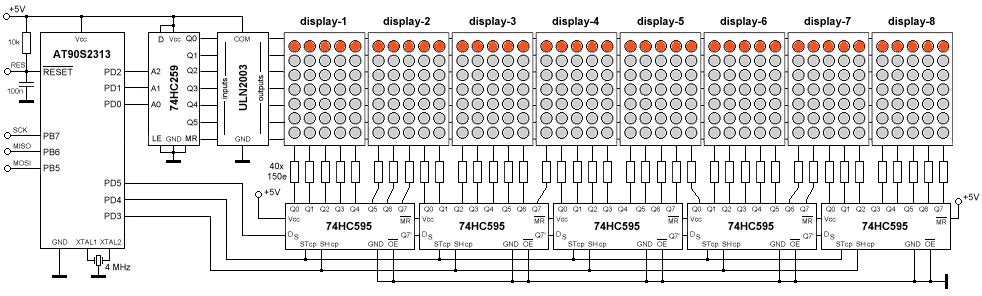
### Extra

Om betere metingen te doen heb ik een low pas filter voorzien aan de ingang van de ADC. Het voordeel hiervan is dat er geen spiegelfrequenties aanwezig zijn in de samples. Ook is er in de code een Hann window voorzien. Een Hann window is een venster dat geplaatst wordt over een signaal om het verschuiven van de frequentie amplitude te minderen

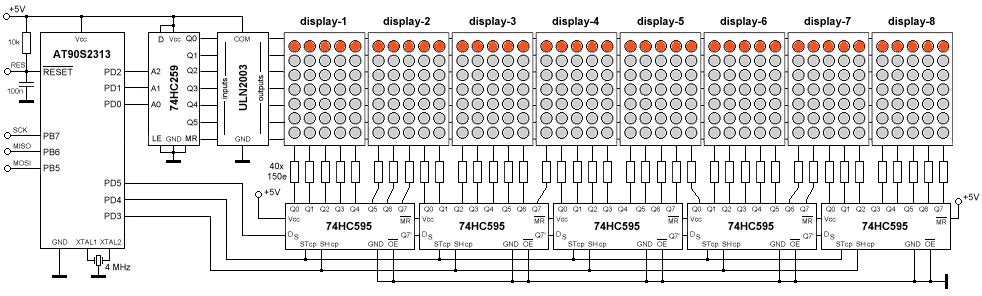
## De Ledmatrix

Omdat ik 30 frequenties wil laten zien, maak ik gebruik van een 32x8 Ledmatrix. De rijen zullen de amplitude weergeven en de kollommen de frequentie. De 30 frequenties zijn bepaald door van ieder octaaf zijn tertsen te nemen. Een octaaf is een bepaalde toon/frequentie en ligt dubbel zo hoog als zijn vorige octaaf.

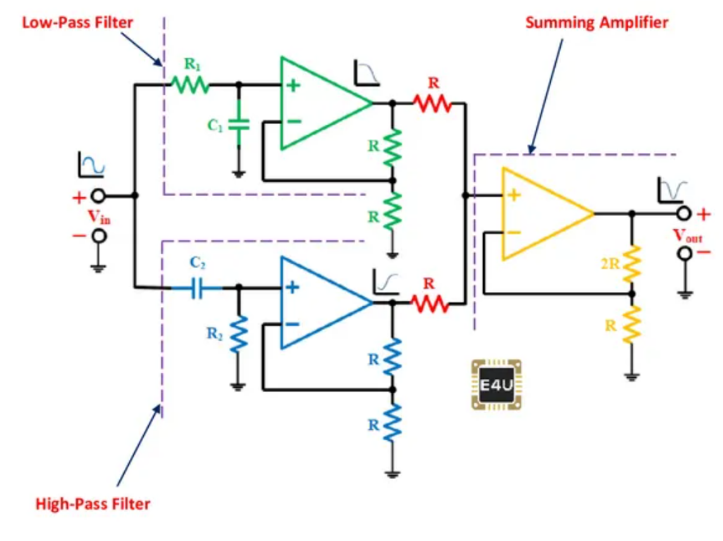
### De rijen

De rijen worden aangestuurd via een ULN2003. De ULN2003 is een ic waar 8 transistors in een behuizing zitten. Het voordeel hiervan is dat de stroom niet uit de microcontroller komt maar vanuit de voeding.

### De kolommen

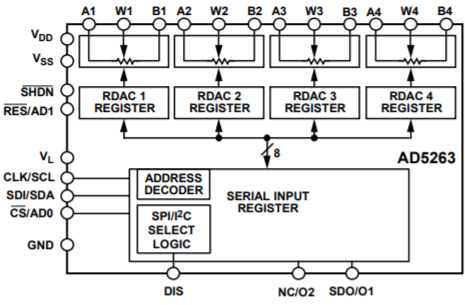
Voor de kollommen maak ik gebruik van een serieel in parallel uit schuifregister. In dit project is er gekozen voor de tpic6c595. Het heeft dezelfde werking als de bekende 74hc595 met als grote verschil dat het meer stroom aankan. Omdat de LED’s maar in een volledige cyclus 1/32 van de tijd oplichten worden de LED’s overstuurd. Hierdoor zijn ze feller en dus duidelijker.

## De equalizer

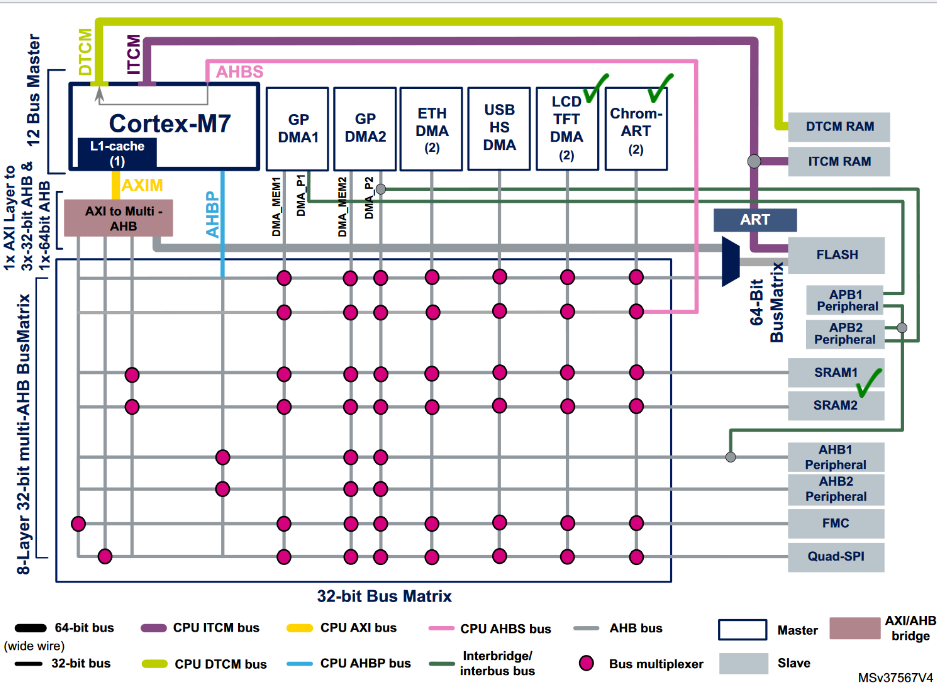
De Audio die binnenkomt gaat ook door een equalizer. Zo kan de bass en de treble aangepast worden op de lcd. De bassen worden aangepast door een laagdoorlaatfilter en de treble door een hoogdoorlaatfilter. De uitgangen van die filters worden dan samen opgeteld door een sommeerversterker. Die uitgang gaat dan naar een potentiometer. De gain van de laag- en hoogdoorlaatfilter worden aangepast door een digitale potentiometer.

### De digitale potentiometer

Om de gain aan te passen moeten we gebruik maken van potentiometers. In het project wordt er gebruik gemaakt van een lcd om de instellingen te doen van de equalizer. Het gevolg hiervan is dat het volledig digitaal geregeld moet worden. De potentiometer die gebruikt wordt, is van analog devices. De AD5263 is een digitale potentiometer die zowel I2C als SPI ondersteunt. In het ic zitten 4 potentiometers die apart geregeld kunnen worden.



## Touch LCD

Op de stm32f746 is een grafische display aanwezig met touch. In de externe ram wordt een fram gemaakt. Dat frame wordt dan door een dma gestuurt naar de TFT-controller. De display is 272x480 pixels groot, wat zeker voldoende is om een duidelijke afbeelding te tonen. Er kan op 2 lagen gewerkt maar 1 laag voldoet voor dit project. De touch werkt op een capacitieve manier. Doordat je vinger het scherm aanraakt, wordt er op die moment een capacitieve verbinding gemaakt en wordt het elektrisch veld op het scherm onderbroken. Dit wordt dan gedetecteerd door een ic en naar de microcontroller gestuurd.

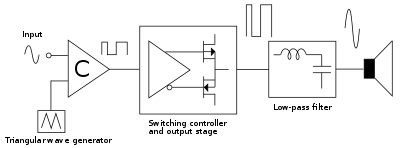
## Volumeregeling

Het volume gaan we regelen met een mechanische potentiometer. We gebruiken een 50k ohm potentiometer die rechtstreeks verbonden is met de input van de versterker. De potentiometer is dubbel uitgevoerd zodat het linker en rechter kanaal apart regelbaar zijn.

## 25W volumeversterker

De TDA7492P is een stereo klasse-D versterker. Dit IC, van ST, heeft een uitgangsimpedantie van 8ohm en een vermogen van 25W per kanaal. Het is een voorgemaakte versterker waar men nog enkele randcomponenten moet voorzien voor een optimale werking. Ik gebruik bij deze versterker de shutdownpin en de mute pin om de versterker op of af te zetten. De versterker heeft een zeer hoge efficiëntie van 90% waardoor er een koelvin kan vermeden worden

### Klasse-D

De klasse-D versterk is een schakelende versterker. Onder schakelend zien we dat het audiosignaal geschakeld wordt op de uitgang. De werking begint met het mixen van het audiosignaal en een driehoekspanning in een comparator. Een comparator is een component dat 2 ingangen vergelijkt. Als het signaal van ingang 1 groter is als die van 2 dan wordt de uitgang hoog. Wanneer het signaal van ingang 2 groter is, dan zal de uitgang laag zijn. De uitgang van de comparator wordt geschakeld op een hogere spanning. Zo wordt het “zwakke” signaal van de comparator versterkt. Het versterkte signaal gaat door een laagdoorlaatfilter om een bruikbaar audiosignaal te krijgen.

# Risicoanalyse

## Oververhitten

De complete schakeling vraagt wel wat vermogen. Om te voorkomen dat de hitte voor problemen zorgt, worden er koelvinnen voorzien.

### Berekening diodebrug

Geg: Ud = 2,2V I = 4 A

Rthj-c = 4 °C/W Rthc-h = 0,5 °C/W

Rthh-a = 6,5 °C/W Ta = 25 °C

Gev: Tj

Opl: P = 𝑈d∗𝐼 = 2,2 ∗ 4 = 8,8𝑊

Rthj-a = Rthj−c+ Rthc−h+Rthh−a = 4 + 0,5 + 6,5 = 11°C/W

Tj = (𝑃𝐷∗ Rthj−a)+𝑇𝑎 = (8,8∗11)+25 = 121,8°C

### Berekening koelvin emittervolger (13V)

Geg: Uin = 15,9 V Ic = 1,5 A Uz = 13V

Ube = 0,7V Rthj-c = 1,52 °C/W Rthc-h = 0,5 °C/W

Rthh-a = 6 °C/W Ta = 25 °C

Gev: Tj

Opl: Uce = 𝑈𝑖𝑛 − 𝑈𝑧 – 𝑈𝑏𝑒 = 15,9 − 13 − 0,7 = 2,2V

P = 𝑈𝑐𝑒∗𝐼𝑐 = 2,2 ∗ 1,5 = 3,3𝑊

Rthj-a = Rthj−c+ Rthc−h+Rthh−a = 1,52 + 0,5 + 6 = 8,2°C/W

Tj = (𝑃𝐷∗ Rthj−a)+𝑇𝑎 = (8,2∗3,3)+25 = 52 °C

### Berekening koelvin LM317 & LM337 (13,75V)

Geg: Uin = 15,9 V I = 0,5 A Uout = 13,75V

Rthj-c = 5 °C/W Rthc-h = 0,5 °C/W

Rthh-a = 17°C/W Ta = 25 °C

Gev: Tj

Opl: Up = 𝑈𝑖𝑛 − 𝑈𝑜𝑢𝑡 = 15,9 – 13,75 = 2,15V

P = 𝑈𝑝∗𝐼=2,15∗0,5= 1,075𝑊

Rthj-a = Rthj−c+ Rthc−h+Rthh−a= 5 + 0,5 + 17 = 22,5°C/W

Tj = (𝑃𝐷∗ Rthj−a)+𝑇𝑎 = (1,075∗22,5)+25 = 49,19°C

### Berekening koelvin LM317 (5V)

Geg: Uin = 15,9 V I = 1 A Uout = 5V

Rthj-c = 5 °C/W Rthc-h = 0,5 °C/W

Rthh-a = 3,6°C/W Ta = 25 °C

Gev: Tj

Opl: Up = 𝑈𝑖𝑛 − 𝑈𝑜𝑢𝑡 = 15,9 – 5 = 10,9V

P = 𝑈𝑝∗𝐼=10,9∗1= 10,9𝑊

Rthj-a = Rthj−c+ Rthc−h+Rthh−a= 5 + 0,5 + 3,6 = 9,1°C/W

Tj = (𝑃𝐷∗ Rthj−a)+𝑇𝑎 = (10,9∗9,1)+25 = 124,19°C

# Slotwoord

Het PE was voor mij een persoonlijke uitdaging. Ik kon een project verwezenlijken dat ik voor ogen had. Bij het uitwerken van de equalizer waren er een paar struikelblokken. Ik geleerd om te werken met een dma, programmeren van externe ram en gebruik te maken van een FFT. Wat ik wel anders zou doen, is het ontwerpen van de equalizer. Hier komt veel meer bij zien en vraagt enige aandacht bij het ontwerpen.

# Programma

/\* USER CODE BEGIN Header \*/

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file : main.c

\* @brief : Main program body

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.

\* All rights reserved.</center></h2>

\*

\* This software component is licensed by ST under Ultimate Liberty license

\* SLA0044, the "License"; You may not use this file except in compliance with

\* the License. You may obtain a copy of the License at:

\* www.st.com/SLA0044

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* USER CODE END Header \*/

/\* Includes ------------------------------------------------------------------\*/

#include "main.h"

#include "lwip.h"

/\* Private includes ----------------------------------------------------------\*/

/\* USER CODE BEGIN Includes \*/

#include <string.h>

#include "stm32746g\_discovery.h"

#include "stm32746g\_discovery\_lcd.h"

#include "stm32746g\_discovery\_sdram.h"

#include "stm32746g\_discovery\_ts.h"

#include "math.h"

#include "achtergrond\_data.h"

#include "-100\_data.h"

#include "-75\_data.h"

#include "-50\_data.h"

#include "-25\_data.h"

#include "0\_data.h"

#include "25\_data.h"

#include "50\_data.h"

#include "75\_data.h"

#include "100\_data.h"

#define SCREENSAVER\_DELAY 10000

/\* USER CODE END Includes \*/

/\* Private typedef -----------------------------------------------------------\*/

/\* USER CODE BEGIN PTD \*/

/\* USER CODE END PTD \*/

/\* Private define ------------------------------------------------------------\*/

/\* USER CODE BEGIN PD \*/

#define aantal\_samples 65536

#define aantal\_samples\_d2 aantal\_samples/2

/\* USER CODE END PD \*/

/\* Private macro -------------------------------------------------------------\*/

/\* USER CODE BEGIN PM \*/

/\* USER CODE END PM \*/

/\* Private variables ---------------------------------------------------------\*/

ADC\_HandleTypeDef hadc3;

DMA\_HandleTypeDef hdma\_adc3;

DMA2D\_HandleTypeDef hdma2d;

I2C\_HandleTypeDef hi2c1;

LTDC\_HandleTypeDef hltdc;

TIM\_HandleTypeDef htim2;

TIM\_HandleTypeDef htim8;

UART\_HandleTypeDef huart1;

SDRAM\_HandleTypeDef hsdram1;

/\* USER CODE BEGIN PV \*/

/\* Private variables ---------------------------------------------------------\*/

uint32\_t N = aantal\_samples;

float Cte = 0;

uint32\_t i = 0;

uint8\_t j = 0;

uint16\_t k = 0;

uint16\_t terts[]={25,31,40,50,63,80,100,125,160,200,250,315,400,500,630,800,1000,1250,1600,2000,2500,3150,4000,5000,6300,8000,10000,12500,16000,20000};

//uint16\_t terts[]={63,16456};

uint8\_t \* pointer = (uint8\_t \*)0xC0000000;

float \* sample = (float \*)0xC0411000;

float \* even = (float \*)0xC0101000;

float \* oneven = (float \*)0xC0201000;

float \* han = (float \*)0xC0301000;

uint8\_t m = 32;

uint16\_t LedState = 0;

uint8\_t val[32];

TS\_StateTypeDef TS\_State;

volatile uint8\_t Bass = 4;

volatile uint8\_t Treble = 4;

volatile uint8\_t change = 0;

volatile uint8\_t AD5236\_ADDR = 0x58; // Use 8-bit address

volatile uint8\_t REG\_INS1 = 0b00000000;

volatile uint8\_t REG\_INS2 = 0b00100000;

volatile uint8\_t REG\_INS3 = 0b01000000;

volatile uint8\_t REG\_INS4 = 0b01100000;

uint8\_t REG\_DATAb = 128;

uint8\_t REG\_DATAt = 128;

//uint8\_t pointer[50];

/\* USER CODE END PV \*/

/\* Private function prototypes -----------------------------------------------\*/

void SystemClock\_Config(void);

static void MX\_GPIO\_Init(void);

static void MX\_DMA\_Init(void);

static void MX\_LTDC\_Init(void);

static void MX\_USART1\_UART\_Init(void);

static void MX\_DMA2D\_Init(void);

static void MX\_FMC\_Init(void);

static void MX\_ADC3\_Init(void);

static void MX\_TIM8\_Init(void);

static void MX\_I2C1\_Init(void);

static void MX\_TIM2\_Init(void);

/\* USER CODE BEGIN PFP \*/

/\* Private function prototypes -----------------------------------------------\*/

/\* USER CODE END PFP \*/

/\* Private user code ---------------------------------------------------------\*/

/\* USER CODE BEGIN 0 \*/

#include <errno.h>

#include <sys/stat.h>

#include <sys/times.h>

#include <sys/unistd.h>

int \_write(int file, char \*ptr, int len) {

HAL\_StatusTypeDef xStatus;

switch (file) {

case STDOUT\_FILENO: /\*stdout\*/

xStatus = HAL\_UART\_Transmit(&huart1, (uint8\_t\*)ptr, len, HAL\_MAX\_DELAY);

if (xStatus != HAL\_OK) {

errno = EIO;

return -1;

}

break;

case STDERR\_FILENO: /\* stderr \*/

xStatus = HAL\_UART\_Transmit(&huart1, (uint8\_t\*)ptr, len, HAL\_MAX\_DELAY);

if (xStatus != HAL\_OK) {

errno = EIO;

return -1;

}

break;

default:

errno = EBADF;

return -1;

}

return len;

}

volatile uint8\_t ready = 0;

void HAL\_ADC\_ConvCpltCallback(ADC\_HandleTypeDef\* hadc)

{

// Conversion Complete & DMA Transfer Complete As Well

// So The AD\_RES Is Now Updated & Let's Move IT To The PWM CCR1

// Update The PWM Duty Cycle With Latest ADC Conversion Result

HAL\_ADC\_Stop\_DMA(&hadc3);

ready++;

if(ready == 1){

HAL\_ADC\_Start\_DMA(&hadc3, (uint32\_t\*)&pointer[aantal\_samples\_d2], aantal\_samples\_d2);

}

// for(int i = 0; i<20; i++){

// printf("int%d: %d \n\r",i, pointer[i]);

// }

// memset(pointer,'\0',20);

// HAL\_ADC\_Start\_DMA(&hadc3, (uint32\_t\*)&pointer[0], 20);

}

void SysTickDelayCount(unsigned long ulCount){

CoreDebug->DEMCR |= CoreDebug\_DEMCR\_TRCENA\_Msk;

DWT->LAR = 0xC5ACCE55;

DWT->CYCCNT = 0;

DWT->CTRL |= DWT\_CTRL\_CYCCNTENA\_Msk;

while(DWT->CYCCNT < ulCount);

}

HAL\_TIM\_PeriodElapsedCallback(TIM\_HandleTypeDef \*htim){

uint8\_t buf[2];

BSP\_TS\_GetState(&TS\_State);

if(TS\_State.touchDetected){ //zien of er een aanraking was

//bepalen waar de vinger is

uint16\_t Y = TS\_State.touchY[0];

uint16\_t X = TS\_State.touchX[0];

if (X <= 254 && X >= 220) { //- knop bass

printf("Bass: %d \n\r", Bass );

if(Y >= 30 && Y <= 81){

if(Bass != 0){

Bass--;

change = 1;

printf("Bass: %d \n\r", Bass );

}

}

if(Y >= 190 && Y <= 241){ //-knop treble

if(Treble != 0){

Treble--;

change = 1;

printf("Treble: %d \n\r", Treble );

}

}

}

if (X >= 420 && X <= 450) {

printf("Bass: %d \n\r", Bass );

if(Y >= 30 && Y <= 81){ //+knop bass

if(Bass != 8){

Bass++;

change = 1;

printf("Bass: %d \n\r", Bass );

}

}

if(Y >= 190 && Y <= 241){ //+knop treble

if(Treble != 8){

Treble++;

change = 1;

printf("Treble: %d \n\r", Treble );

}

}

}

if(change == 1){

change = 0;

//bepalen welke figuur er op komt

if(Bass == 0){

WDA\_LCD\_DrawBitmap(n100\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 255;

}

if(Bass == 1){

WDA\_LCD\_DrawBitmap(n75\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 223;

}

if(Bass == 2){

WDA\_LCD\_DrawBitmap(n50\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 191;

}

if(Bass == 3){

WDA\_LCD\_DrawBitmap(n25\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 159;

}

if(Bass == 4){

WDA\_LCD\_DrawBitmap(\_0\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 128;

}

if(Bass == 5){

WDA\_LCD\_DrawBitmap(\_25\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 112;

}

if(Bass == 6){

WDA\_LCD\_DrawBitmap(\_50\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 96;

}

if(Bass == 7){

WDA\_LCD\_DrawBitmap(\_75\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 80;

}

if(Bass == 8){

WDA\_LCD\_DrawBitmap(\_100\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAb = 64;

}

if(Treble == 0){

WDA\_LCD\_DrawBitmap(n100\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 64;

}

if(Treble == 1){

WDA\_LCD\_DrawBitmap(n75\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 80;

}

if(Treble == 2){

WDA\_LCD\_DrawBitmap(n50\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 96;

}

if(Treble == 3){

WDA\_LCD\_DrawBitmap(n25\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 112;

}

if(Treble == 4){

WDA\_LCD\_DrawBitmap(\_0\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 128;

}

if(Treble == 5){

WDA\_LCD\_DrawBitmap(\_25\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 159;

}

if(Treble == 6){

WDA\_LCD\_DrawBitmap(\_50\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 191;

}

if(Treble == 7){

WDA\_LCD\_DrawBitmap(\_75\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 223;

}

if(Treble == 8){

WDA\_LCD\_DrawBitmap(\_100\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555);

REG\_DATAt = 255;

}

buf[0] = REG\_INS1;

buf[1] = REG\_DATAb;

HAL\_I2C\_Master\_Transmit(&hi2c1, AD5236\_ADDR, buf, 2, HAL\_MAX\_DELAY); //data sturen naar pot 1

buf[0] = REG\_INS2;

buf[1] = REG\_DATAb;

HAL\_I2C\_Master\_Transmit(&hi2c1, AD5236\_ADDR, buf, 2, HAL\_MAX\_DELAY); //data sturen naar pot 2

buf[0] = REG\_INS3;

buf[1] = REG\_DATAt;

HAL\_I2C\_Master\_Transmit(&hi2c1, AD5236\_ADDR, buf, 2, HAL\_MAX\_DELAY); //data sturen naar pot 3

buf[0] = REG\_INS4;

buf[1] = REG\_DATAt;

HAL\_I2C\_Master\_Transmit(&hi2c1, AD5236\_ADDR, buf, 2, HAL\_MAX\_DELAY); //data sturen naar pot 4

}

}

}

/\* USER CODE END 0 \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

int main(void)

{

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* Enable I-Cache---------------------------------------------------------\*/

SCB\_EnableICache();

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

MX\_DMA\_Init();

MX\_LTDC\_Init();

MX\_USART1\_UART\_Init();

MX\_DMA2D\_Init();

MX\_FMC\_Init();

MX\_LWIP\_Init();

MX\_ADC3\_Init();

MX\_TIM8\_Init();

MX\_I2C1\_Init();

MX\_TIM2\_Init();

/\* USER CODE BEGIN 2 \*/

/\* LCD Initialization \*/

BSP\_LCD\_Init();

BSP\_LCD\_LayerDefaultInit(1,LCD\_FB\_START\_ADDRESS);//adres instellen voor externe ram

/\* Enable the LCD \*/

BSP\_LCD\_DisplayOn(); //lcd aan

/\* Select the LCD Background Layer \*/

BSP\_LCD\_SelectLayer(1); //laag 1 van lcd kiezen

/\* Touch veld instellen\*/

BSP\_TS\_Init(480,272);

WDA\_LCD\_DrawBitmap(ACHTERGROND\_DATA,0,0,ACHTERGROND\_DATA\_X\_PIXEL,ACHTERGROND\_DATA\_Y\_PIXEL,ACHTERGROND\_DATA\_FORMAT); //achtergrond tekenen

WDA\_LCD\_DrawBitmap(\_0\_DATA,254,30,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555); //afbeelding 0 tekenen

WDA\_LCD\_DrawBitmap(\_0\_DATA,254,190,145,51,LTDC\_PIXEL\_FORMAT\_ARGB1555); //afbeelding 0 tekenen

HAL\_TIM\_Base\_Start\_IT(&htim2);//timer 2 starten

//printf("DMA test3\r\n");

HAL\_StatusTypeDef stat;

stat = HAL\_ADC\_Start\_DMA(&hadc3, (uint32\_t\*) &pointer[0], aantal\_samples\_d2);//dma starten en kijken of het werkt

if(stat != HAL\_OK){

printf("HAL\_ADC\_Start\_DMA faalt\r\n");

}

stat = HAL\_TIM\_Base\_Start\_IT(&htim8);//timer voor dma starten en kijken of het werkt

if(stat != HAL\_OK){

printf("HAL\_TIM\_Base\_Start\_IT faalt\r\n");

}

HAL\_GPIO\_WritePin(OE\_GPIO\_Port,OE\_Pin,SET); //leds uit

HAL\_GPIO\_WritePin(SER\_GPIO\_Port,SER\_Pin,SET); //ingang hoog

HAL\_GPIO\_WritePin(Latch\_GPIO\_Port,Latch\_Pin,RESET); //clk naar beneden

for (i = 0; i < aantal\_samples; i++) { //hann window constante al uitrekenen

han[i] = (1 - cos(2\*M\_PI\*i/aantal\_samples))/2;

}

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

while (1)

{

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

if(ready == 2){

//printf("ready \n\r");

ready = 0;

HAL\_GPIO\_WritePin(OE\_GPIO\_Port,OE\_Pin,SET);//leds uit

//printf("ready: %d\n\r",ready);

N = aantal\_samples;

//printf("adc: %d \n\r",pointer[0]);

for (i = 0; i < aantal\_samples; i++) { //alles verplaatsen naar float en hann window toepassen

sample[i] = ((pointer[i] -98) /98.0000)\*han[i];

}

//splitsen

while(N != 2){

for(i = 0; i < aantal\_samples\_d2; i++){

even[i]= sample[i\*2];

oneven[i]= sample[i\*2+1];

}

N = N/2;

for(i = 0; i < aantal\_samples\_d2; i++){

sample[i] = even[i];

sample[aantal\_samples\_d2+i] = oneven[i];

}

}

//samenvoegen

for (j = 0; j < 30; j++) {

N = aantal\_samples;

k = terts[j];

while( N != 1){

Cte = (-M\_PI\*N\*k)/aantal\_samples;

Cte = cos(Cte)+sin(Cte);

N = N/2;

for(i=0; i < N;i++){

sample[2\*i]=sample[2\*i]+Cte\*sample[2\*i+1];

//printf("sample: %f\n\r",sample[2\*i]);

}

if (N > 2) {

for(i=0; i < (N/2); i = i +2){

sample[i+1]=sample[i+N];

}

}

if (N == 1) {

sample[1] = sample[2];

}

}

sample[0] = abs(sample[0]);

val[j] = 0;

//waarde bepalen voor op leds

if(((uint8\_t)sample[0] / 1) != 0){

val[j] = 1;

}

if(((uint8\_t)sample[0] / 2) != 0){

val[j] = 1;

}

if(((uint8\_t)sample[0] / 4) != 0){

val[j] = 3;

}

if(((uint8\_t)sample[0] / 8) != 0){

val[j] = 7;

}

if(((uint8\_t)sample[0] / 16) != 0){

val[j] = 15;

}

if(((uint8\_t)sample[0] / 32) != 0){

val[j] = 31;

}

if(((uint8\_t)sample[0] / 64) != 0){

val[j] = 63;

}

if(((uint8\_t)sample[0] / 128) != 0){

val[j] = 127;

}

if(((uint8\_t)sample[0] / 255) != 0){

val[j] = 255;

}

//printf("freq: %dHz: %f\n\r",terts[j],sample[0]);

for(i = 0; i < aantal\_samples\_d2; i++){

sample[i] = even[i];

sample[aantal\_samples\_d2+i] = oneven[i];

}//alle waardes terug resetten om volgende frequentie te berekenen

}

HAL\_ADC\_Start\_DMA(&hadc3, (uint32\_t\*)&pointer[0], aantal\_samples\_d2);//dma starten

}

//ledmatrix

HAL\_GPIO\_WritePin(OE\_GPIO\_Port,OE\_Pin,RESET); // als 32 pulsen zijn geweest, 1 op ingang zetten

if( m == 32){

m = 0;

HAL\_GPIO\_WritePin(SER\_GPIO\_Port,SER\_Pin,SET);

}

else{

HAL\_GPIO\_WritePin(SER\_GPIO\_Port,SER\_Pin,RESET);

}

HAL\_GPIO\_WritePin(SRCLK\_GPIO\_Port,SRCLK\_Pin,SET); //SRCLK is latch tegenovergestelde clk

HAL\_GPIO\_WritePin(Latch\_GPIO\_Port,Latch\_Pin,RESET);

SysTickDelayCount(10000); //wachttijd om leds aan te passen

LedState = GPIOI -> ODR; //zien wat in register zit

LedState = LedState & 0xFFF0; //masker plaatsen

LedState = LedState+((~val[m] & 0xF0)>>4); //waarde op juiste pinnen zetten

GPIOI -> ODR = LedState; //4 eerste leds aansturen

LedState = GPIOF -> ODR; //zien wat in register zit

LedState = LedState & 0xFC3F;//masker plaatsen

LedState = LedState+((~val[m] & 0x0F)<<6); //waarde op juiste pinnen zetten

GPIOF -> ODR = LedState; //4 laatste leds aansturen

HAL\_GPIO\_WritePin(SRCLK\_GPIO\_Port,SRCLK\_Pin,RESET);

HAL\_GPIO\_WritePin(Latch\_GPIO\_Port,Latch\_Pin,SET);

SysTickDelayCount(10000);//wachttijd om leds aan te passen

m = m + 1;

}

/\* USER CODE END 3 \*/

}

/\*\*

\* @brief System Clock Configuration

\* @retval None

\*/

void SystemClock\_Config(void)

{

RCC\_OscInitTypeDef RCC\_OscInitStruct = {0};

RCC\_ClkInitTypeDef RCC\_ClkInitStruct = {0};

RCC\_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};

/\*\* Configure LSE Drive Capability

\*/

HAL\_PWR\_EnableBkUpAccess();

/\*\* Configure the main internal regulator output voltage

\*/

\_\_HAL\_RCC\_PWR\_CLK\_ENABLE();

\_\_HAL\_PWR\_VOLTAGESCALING\_CONFIG(PWR\_REGULATOR\_VOLTAGE\_SCALE1);

/\*\* Initializes the RCC Oscillators according to the specified parameters

\* in the RCC\_OscInitTypeDef structure.

\*/

RCC\_OscInitStruct.OscillatorType = RCC\_OSCILLATORTYPE\_HSE;

RCC\_OscInitStruct.HSEState = RCC\_HSE\_ON;

RCC\_OscInitStruct.PLL.PLLState = RCC\_PLL\_ON;

RCC\_OscInitStruct.PLL.PLLSource = RCC\_PLLSOURCE\_HSE;

RCC\_OscInitStruct.PLL.PLLM = 25;

RCC\_OscInitStruct.PLL.PLLN = 400;

RCC\_OscInitStruct.PLL.PLLP = RCC\_PLLP\_DIV2;

RCC\_OscInitStruct.PLL.PLLQ = 2;

if (HAL\_RCC\_OscConfig(&RCC\_OscInitStruct) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Activate the Over-Drive mode

\*/

if (HAL\_PWREx\_EnableOverDrive() != HAL\_OK)

{

Error\_Handler();

}

/\*\* Initializes the CPU, AHB and APB buses clocks

\*/

RCC\_ClkInitStruct.ClockType = RCC\_CLOCKTYPE\_HCLK|RCC\_CLOCKTYPE\_SYSCLK

|RCC\_CLOCKTYPE\_PCLK1|RCC\_CLOCKTYPE\_PCLK2;

RCC\_ClkInitStruct.SYSCLKSource = RCC\_SYSCLKSOURCE\_PLLCLK;

RCC\_ClkInitStruct.AHBCLKDivider = RCC\_SYSCLK\_DIV1;

RCC\_ClkInitStruct.APB1CLKDivider = RCC\_HCLK\_DIV4;

RCC\_ClkInitStruct.APB2CLKDivider = RCC\_HCLK\_DIV2;

if (HAL\_RCC\_ClockConfig(&RCC\_ClkInitStruct, FLASH\_LATENCY\_6) != HAL\_OK)

{

Error\_Handler();

}

PeriphClkInitStruct.PeriphClockSelection = RCC\_PERIPHCLK\_LTDC|RCC\_PERIPHCLK\_USART1

|RCC\_PERIPHCLK\_I2C1;

PeriphClkInitStruct.PLLSAI.PLLSAIN = 192;

PeriphClkInitStruct.PLLSAI.PLLSAIR = 5;

PeriphClkInitStruct.PLLSAI.PLLSAIQ = 2;

PeriphClkInitStruct.PLLSAI.PLLSAIP = RCC\_PLLSAIP\_DIV2;

PeriphClkInitStruct.PLLSAIDivQ = 1;

PeriphClkInitStruct.PLLSAIDivR = RCC\_PLLSAIDIVR\_4;

PeriphClkInitStruct.Usart1ClockSelection = RCC\_USART1CLKSOURCE\_PCLK2;

PeriphClkInitStruct.I2c1ClockSelection = RCC\_I2C1CLKSOURCE\_PCLK1;

if (HAL\_RCCEx\_PeriphCLKConfig(&PeriphClkInitStruct) != HAL\_OK)

{

Error\_Handler();

}

}

/\*\*

\* @brief ADC3 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_ADC3\_Init(void)

{

/\* USER CODE BEGIN ADC3\_Init 0 \*/

/\* USER CODE END ADC3\_Init 0 \*/

ADC\_ChannelConfTypeDef sConfig = {0};

/\* USER CODE BEGIN ADC3\_Init 1 \*/

/\* USER CODE END ADC3\_Init 1 \*/

/\*\* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

\*/

hadc3.Instance = ADC3;

hadc3.Init.ClockPrescaler = ADC\_CLOCK\_SYNC\_PCLK\_DIV8;

hadc3.Init.Resolution = ADC\_RESOLUTION\_8B;

hadc3.Init.ScanConvMode = ADC\_SCAN\_DISABLE;

hadc3.Init.ContinuousConvMode = DISABLE;

hadc3.Init.DiscontinuousConvMode = DISABLE;

hadc3.Init.ExternalTrigConvEdge = ADC\_EXTERNALTRIGCONVEDGE\_RISING;

hadc3.Init.ExternalTrigConv = ADC\_EXTERNALTRIGCONV\_T8\_TRGO;

hadc3.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;

hadc3.Init.NbrOfConversion = 1;

hadc3.Init.DMAContinuousRequests = DISABLE;

hadc3.Init.EOCSelection = ADC\_EOC\_SINGLE\_CONV;

if (HAL\_ADC\_Init(&hadc3) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.

\*/

sConfig.Channel = ADC\_CHANNEL\_8;

sConfig.Rank = ADC\_REGULAR\_RANK\_1;

sConfig.SamplingTime = ADC\_SAMPLETIME\_3CYCLES;

if (HAL\_ADC\_ConfigChannel(&hadc3, &sConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN ADC3\_Init 2 \*/

/\* USER CODE END ADC3\_Init 2 \*/

}

/\*\*

\* @brief DMA2D Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_DMA2D\_Init(void)

{

/\* USER CODE BEGIN DMA2D\_Init 0 \*/

/\* USER CODE END DMA2D\_Init 0 \*/

/\* USER CODE BEGIN DMA2D\_Init 1 \*/

/\* USER CODE END DMA2D\_Init 1 \*/

hdma2d.Instance = DMA2D;

hdma2d.Init.Mode = DMA2D\_M2M;

hdma2d.Init.ColorMode = DMA2D\_OUTPUT\_ARGB8888;

hdma2d.Init.OutputOffset = 0;

hdma2d.LayerCfg[1].InputOffset = 0;

hdma2d.LayerCfg[1].InputColorMode = DMA2D\_INPUT\_ARGB8888;

hdma2d.LayerCfg[1].AlphaMode = DMA2D\_NO\_MODIF\_ALPHA;

hdma2d.LayerCfg[1].InputAlpha = 0;

if (HAL\_DMA2D\_Init(&hdma2d) != HAL\_OK)

{

Error\_Handler();

}

if (HAL\_DMA2D\_ConfigLayer(&hdma2d, 1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN DMA2D\_Init 2 \*/

/\* USER CODE END DMA2D\_Init 2 \*/

}

/\*\*

\* @brief I2C1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_I2C1\_Init(void)

{

/\* USER CODE BEGIN I2C1\_Init 0 \*/

/\* USER CODE END I2C1\_Init 0 \*/

/\* USER CODE BEGIN I2C1\_Init 1 \*/

/\* USER CODE END I2C1\_Init 1 \*/

hi2c1.Instance = I2C1;

hi2c1.Init.Timing = 0x00C0EAFF;

hi2c1.Init.OwnAddress1 = 0;

hi2c1.Init.AddressingMode = I2C\_ADDRESSINGMODE\_7BIT;

hi2c1.Init.DualAddressMode = I2C\_DUALADDRESS\_DISABLE;

hi2c1.Init.OwnAddress2 = 0;

hi2c1.Init.OwnAddress2Masks = I2C\_OA2\_NOMASK;

hi2c1.Init.GeneralCallMode = I2C\_GENERALCALL\_DISABLE;

hi2c1.Init.NoStretchMode = I2C\_NOSTRETCH\_DISABLE;

if (HAL\_I2C\_Init(&hi2c1) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Analogue filter

\*/

if (HAL\_I2CEx\_ConfigAnalogFilter(&hi2c1, I2C\_ANALOGFILTER\_ENABLE) != HAL\_OK)

{

Error\_Handler();

}

/\*\* Configure Digital filter

\*/

if (HAL\_I2CEx\_ConfigDigitalFilter(&hi2c1, 0) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN I2C1\_Init 2 \*/

/\* USER CODE END I2C1\_Init 2 \*/

}

/\*\*

\* @brief LTDC Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_LTDC\_Init(void)

{

/\* USER CODE BEGIN LTDC\_Init 0 \*/

/\* USER CODE END LTDC\_Init 0 \*/

LTDC\_LayerCfgTypeDef pLayerCfg = {0};

LTDC\_LayerCfgTypeDef pLayerCfg1 = {0};

/\* USER CODE BEGIN LTDC\_Init 1 \*/

/\* USER CODE END LTDC\_Init 1 \*/

hltdc.Instance = LTDC;

hltdc.Init.HSPolarity = LTDC\_HSPOLARITY\_AL;

hltdc.Init.VSPolarity = LTDC\_VSPOLARITY\_AL;

hltdc.Init.DEPolarity = LTDC\_DEPOLARITY\_AL;

hltdc.Init.PCPolarity = LTDC\_PCPOLARITY\_IPC;

hltdc.Init.HorizontalSync = 40;

hltdc.Init.VerticalSync = 9;

hltdc.Init.AccumulatedHBP = 53;

hltdc.Init.AccumulatedVBP = 11;

hltdc.Init.AccumulatedActiveW = 533;

hltdc.Init.AccumulatedActiveH = 283;

hltdc.Init.TotalWidth = 565;

hltdc.Init.TotalHeigh = 285;

hltdc.Init.Backcolor.Blue = 0;

hltdc.Init.Backcolor.Green = 255;

hltdc.Init.Backcolor.Red = 0;

if (HAL\_LTDC\_Init(&hltdc) != HAL\_OK)

{

Error\_Handler();

}

pLayerCfg.WindowX0 = 0;

pLayerCfg.WindowX1 = 480;

pLayerCfg.WindowY0 = 0;

pLayerCfg.WindowY1 = 272;

pLayerCfg.PixelFormat = LTDC\_PIXEL\_FORMAT\_ARGB1555;

pLayerCfg.Alpha = 255;

pLayerCfg.Alpha0 = 0;

pLayerCfg.BlendingFactor1 = LTDC\_BLENDING\_FACTOR1\_PAxCA;

pLayerCfg.BlendingFactor2 = LTDC\_BLENDING\_FACTOR2\_PAxCA;

pLayerCfg.FBStartAdress = 0;

pLayerCfg.ImageWidth = 480;

pLayerCfg.ImageHeight = 272;

pLayerCfg.Backcolor.Blue = 0;

pLayerCfg.Backcolor.Green = 0;

pLayerCfg.Backcolor.Red = 0;

if (HAL\_LTDC\_ConfigLayer(&hltdc, &pLayerCfg, 0) != HAL\_OK)

{

Error\_Handler();

}

pLayerCfg1.WindowX0 = 0;

pLayerCfg1.WindowX1 = 480;

pLayerCfg1.WindowY0 = 0;

pLayerCfg1.WindowY1 = 272;

pLayerCfg1.PixelFormat = LTDC\_PIXEL\_FORMAT\_ARGB1555;

pLayerCfg1.Alpha = 255;

pLayerCfg1.Alpha0 = 0;

pLayerCfg1.BlendingFactor1 = LTDC\_BLENDING\_FACTOR1\_PAxCA;

pLayerCfg1.BlendingFactor2 = LTDC\_BLENDING\_FACTOR2\_PAxCA;

pLayerCfg1.FBStartAdress = 0;

pLayerCfg1.ImageWidth = 480;

pLayerCfg1.ImageHeight = 272;

pLayerCfg1.Backcolor.Blue = 0;

pLayerCfg1.Backcolor.Green = 0;

pLayerCfg1.Backcolor.Red = 0;

if (HAL\_LTDC\_ConfigLayer(&hltdc, &pLayerCfg1, 1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN LTDC\_Init 2 \*/

/\* USER CODE END LTDC\_Init 2 \*/

}

/\*\*

\* @brief TIM2 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM2\_Init(void)

{

/\* USER CODE BEGIN TIM2\_Init 0 \*/

/\* USER CODE END TIM2\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM2\_Init 1 \*/

/\* USER CODE END TIM2\_Init 1 \*/

htim2.Instance = TIM2;

htim2.Init.Prescaler = 10000;

htim2.Init.CounterMode = TIM\_COUNTERMODE\_DOWN;

htim2.Init.Period = 2000;

htim2.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim2.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

if (HAL\_TIM\_Base\_Init(&htim2) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_RESET;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM2\_Init 2 \*/

/\* USER CODE END TIM2\_Init 2 \*/

}

/\*\*

\* @brief TIM8 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_TIM8\_Init(void)

{

/\* USER CODE BEGIN TIM8\_Init 0 \*/

/\* USER CODE END TIM8\_Init 0 \*/

TIM\_ClockConfigTypeDef sClockSourceConfig = {0};

TIM\_MasterConfigTypeDef sMasterConfig = {0};

/\* USER CODE BEGIN TIM8\_Init 1 \*/

/\* USER CODE END TIM8\_Init 1 \*/

htim8.Instance = TIM8;

htim8.Init.Prescaler = 0;

htim8.Init.CounterMode = TIM\_COUNTERMODE\_DOWN;

htim8.Init.Period = 1524;

htim8.Init.ClockDivision = TIM\_CLOCKDIVISION\_DIV1;

htim8.Init.RepetitionCounter = 0;

htim8.Init.AutoReloadPreload = TIM\_AUTORELOAD\_PRELOAD\_ENABLE;

if (HAL\_TIM\_Base\_Init(&htim8) != HAL\_OK)

{

Error\_Handler();

}

sClockSourceConfig.ClockSource = TIM\_CLOCKSOURCE\_INTERNAL;

if (HAL\_TIM\_ConfigClockSource(&htim8, &sClockSourceConfig) != HAL\_OK)

{

Error\_Handler();

}

sMasterConfig.MasterOutputTrigger = TIM\_TRGO\_UPDATE;

sMasterConfig.MasterOutputTrigger2 = TIM\_TRGO2\_UPDATE;

sMasterConfig.MasterSlaveMode = TIM\_MASTERSLAVEMODE\_DISABLE;

if (HAL\_TIMEx\_MasterConfigSynchronization(&htim8, &sMasterConfig) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN TIM8\_Init 2 \*/

/\* USER CODE END TIM8\_Init 2 \*/

}

/\*\*

\* @brief USART1 Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_USART1\_UART\_Init(void)

{

/\* USER CODE BEGIN USART1\_Init 0 \*/

/\* USER CODE END USART1\_Init 0 \*/

/\* USER CODE BEGIN USART1\_Init 1 \*/

/\* USER CODE END USART1\_Init 1 \*/

huart1.Instance = USART1;

huart1.Init.BaudRate = 115200;

huart1.Init.WordLength = UART\_WORDLENGTH\_8B;

huart1.Init.StopBits = UART\_STOPBITS\_1;

huart1.Init.Parity = UART\_PARITY\_NONE;

huart1.Init.Mode = UART\_MODE\_TX\_RX;

huart1.Init.HwFlowCtl = UART\_HWCONTROL\_NONE;

huart1.Init.OverSampling = UART\_OVERSAMPLING\_16;

huart1.Init.OneBitSampling = UART\_ONE\_BIT\_SAMPLE\_DISABLE;

huart1.AdvancedInit.AdvFeatureInit = UART\_ADVFEATURE\_NO\_INIT;

if (HAL\_UART\_Init(&huart1) != HAL\_OK)

{

Error\_Handler();

}

/\* USER CODE BEGIN USART1\_Init 2 \*/

/\* USER CODE END USART1\_Init 2 \*/

}

/\*\*

\* Enable DMA controller clock

\*/

static void MX\_DMA\_Init(void)

{

/\* DMA controller clock enable \*/

\_\_HAL\_RCC\_DMA2\_CLK\_ENABLE();

/\* DMA interrupt init \*/

/\* DMA2\_Stream1\_IRQn interrupt configuration \*/

HAL\_NVIC\_SetPriority(DMA2\_Stream1\_IRQn, 0, 0);

HAL\_NVIC\_EnableIRQ(DMA2\_Stream1\_IRQn);

}

/\* FMC initialization function \*/

static void MX\_FMC\_Init(void)

{

/\* USER CODE BEGIN FMC\_Init 0 \*/

/\* USER CODE END FMC\_Init 0 \*/

FMC\_SDRAM\_TimingTypeDef SdramTiming = {0};

/\* USER CODE BEGIN FMC\_Init 1 \*/

/\* USER CODE END FMC\_Init 1 \*/

/\*\* Perform the SDRAM1 memory initialization sequence

\*/

hsdram1.Instance = FMC\_SDRAM\_DEVICE;

/\* hsdram1.Init \*/

hsdram1.Init.SDBank = FMC\_SDRAM\_BANK1;

hsdram1.Init.ColumnBitsNumber = FMC\_SDRAM\_COLUMN\_BITS\_NUM\_8;

hsdram1.Init.RowBitsNumber = FMC\_SDRAM\_ROW\_BITS\_NUM\_12;

hsdram1.Init.MemoryDataWidth = FMC\_SDRAM\_MEM\_BUS\_WIDTH\_16;

hsdram1.Init.InternalBankNumber = FMC\_SDRAM\_INTERN\_BANKS\_NUM\_4;

hsdram1.Init.CASLatency = FMC\_SDRAM\_CAS\_LATENCY\_1;

hsdram1.Init.WriteProtection = FMC\_SDRAM\_WRITE\_PROTECTION\_DISABLE;

hsdram1.Init.SDClockPeriod = FMC\_SDRAM\_CLOCK\_DISABLE;

hsdram1.Init.ReadBurst = FMC\_SDRAM\_RBURST\_DISABLE;

hsdram1.Init.ReadPipeDelay = FMC\_SDRAM\_RPIPE\_DELAY\_0;

/\* SdramTiming \*/

SdramTiming.LoadToActiveDelay = 16;

SdramTiming.ExitSelfRefreshDelay = 16;

SdramTiming.SelfRefreshTime = 16;

SdramTiming.RowCycleDelay = 16;

SdramTiming.WriteRecoveryTime = 16;

SdramTiming.RPDelay = 16;

SdramTiming.RCDDelay = 16;

if (HAL\_SDRAM\_Init(&hsdram1, &SdramTiming) != HAL\_OK)

{

Error\_Handler( );

}

/\* USER CODE BEGIN FMC\_Init 2 \*/

/\* USER CODE END FMC\_Init 2 \*/

}

/\*\*

\* @brief GPIO Initialization Function

\* @param None

\* @retval None

\*/

static void MX\_GPIO\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStruct = {0};

/\* GPIO Ports Clock Enable \*/

\_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOG\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOB\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOJ\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOD\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOK\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOI\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOF\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOC\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOA\_CLK\_ENABLE();

\_\_HAL\_RCC\_GPIOH\_CLK\_ENABLE();

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(Latch\_GPIO\_Port, Latch\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOI, GPIO\_PIN\_3|GPIO\_PIN\_2|GPIO\_PIN\_1|LCD\_DISP\_Pin

|GPIO\_PIN\_0, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(LCD\_BL\_CTRL\_GPIO\_Port, LCD\_BL\_CTRL\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOC, OE\_Pin|SER\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(SRCLK\_GPIO\_Port, SRCLK\_Pin, GPIO\_PIN\_RESET);

/\*Configure GPIO pin Output Level \*/

HAL\_GPIO\_WritePin(GPIOF, GPIO\_PIN\_7|GPIO\_PIN\_6|GPIO\_PIN\_9|GPIO\_PIN\_8, GPIO\_PIN\_RESET);

/\*Configure GPIO pin : Latch\_Pin \*/

GPIO\_InitStruct.Pin = Latch\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(Latch\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : PI3 PI2 PI1 LCD\_DISP\_Pin

PI0 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_3|GPIO\_PIN\_2|GPIO\_PIN\_1|LCD\_DISP\_Pin

|GPIO\_PIN\_0;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOI, &GPIO\_InitStruct);

/\*Configure GPIO pin : LCD\_BL\_CTRL\_Pin \*/

GPIO\_InitStruct.Pin = LCD\_BL\_CTRL\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(LCD\_BL\_CTRL\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pin : BUTTON\_Pin \*/

GPIO\_InitStruct.Pin = BUTTON\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(BUTTON\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : OE\_Pin SER\_Pin \*/

GPIO\_InitStruct.Pin = OE\_Pin|SER\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOC, &GPIO\_InitStruct);

/\*Configure GPIO pin : SRCLK\_Pin \*/

GPIO\_InitStruct.Pin = SRCLK\_Pin;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(SRCLK\_GPIO\_Port, &GPIO\_InitStruct);

/\*Configure GPIO pins : PF7 PF6 PF9 PF8 \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_7|GPIO\_PIN\_6|GPIO\_PIN\_9|GPIO\_PIN\_8;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_FREQ\_LOW;

HAL\_GPIO\_Init(GPIOF, &GPIO\_InitStruct);

}

/\* USER CODE BEGIN 4 \*/

/\* USER CODE END 4 \*/

/\*\*

\* @brief This function is executed in case of error occurrence.

\* @retval None

\*/

void Error\_Handler(void)

{

/\* USER CODE BEGIN Error\_Handler\_Debug \*/

/\* User can add his own implementation to report the HAL error return state \*/

while(1)

{

}

/\* USER CODE END Error\_Handler\_Debug \*/

}

#ifdef USE\_FULL\_ASSERT

/\*\*

\* @brief Reports the name of the source file and the source line number

\* where the assert\_param error has occurred.

\* @param file: pointer to the source file name

\* @param line: assert\_param error line source number

\* @retval None

\*/

void assert\_failed(uint8\_t \*file, uint32\_t line)

{

/\* USER CODE BEGIN 6 \*/

/\* User can add his own implementation to report the file name and line number,

tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) \*/

/\* USER CODE END 6 \*/

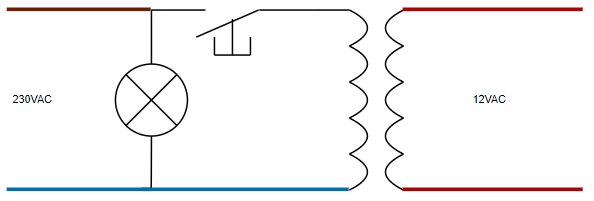
}

#endif /\* USE\_FULL\_ASSERT \*/

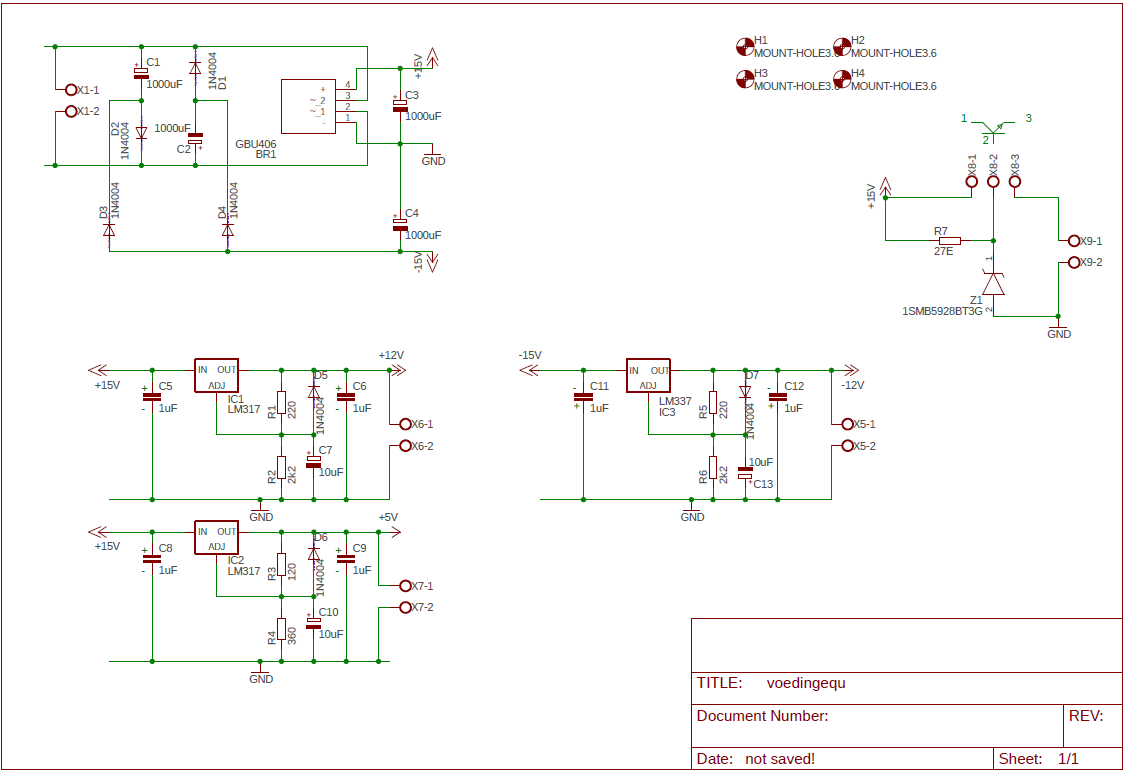
/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* (C) COPYRIGHT STMicroelectronics \*\*\*\*\*END OF FILE\*\*\*\*/

# Bijlage

### Aansluitgegevens Ringkerntransformator

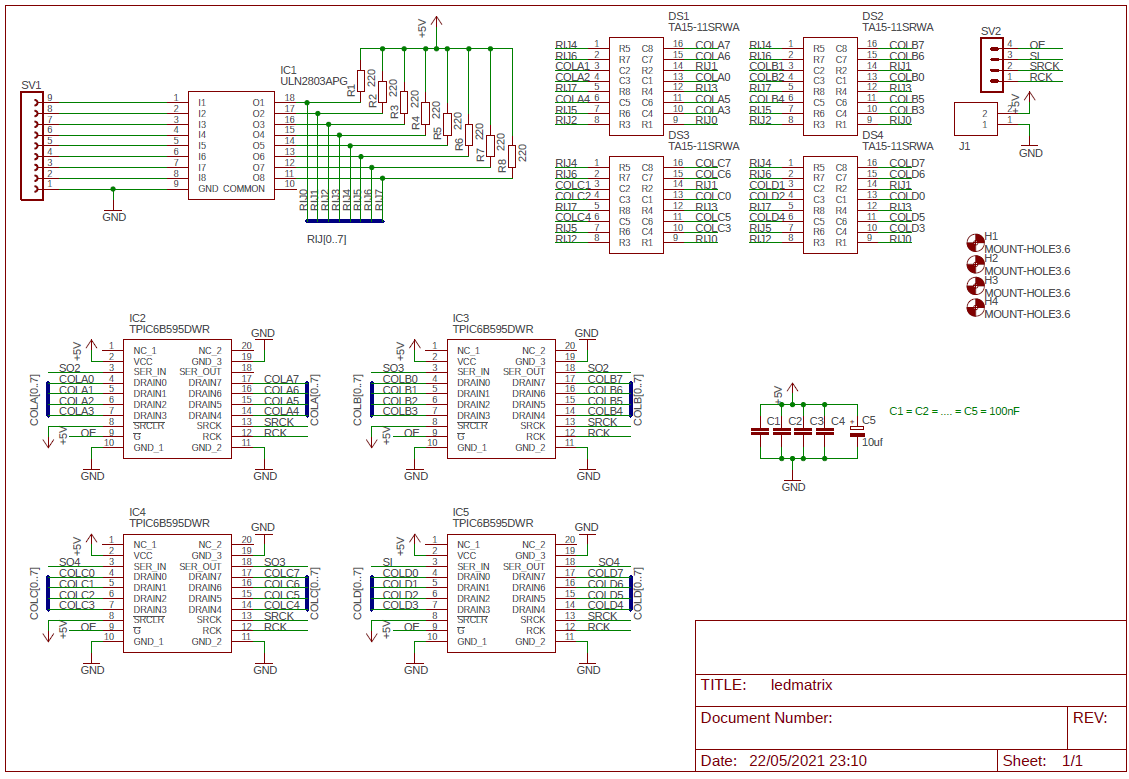


### Schema voeding

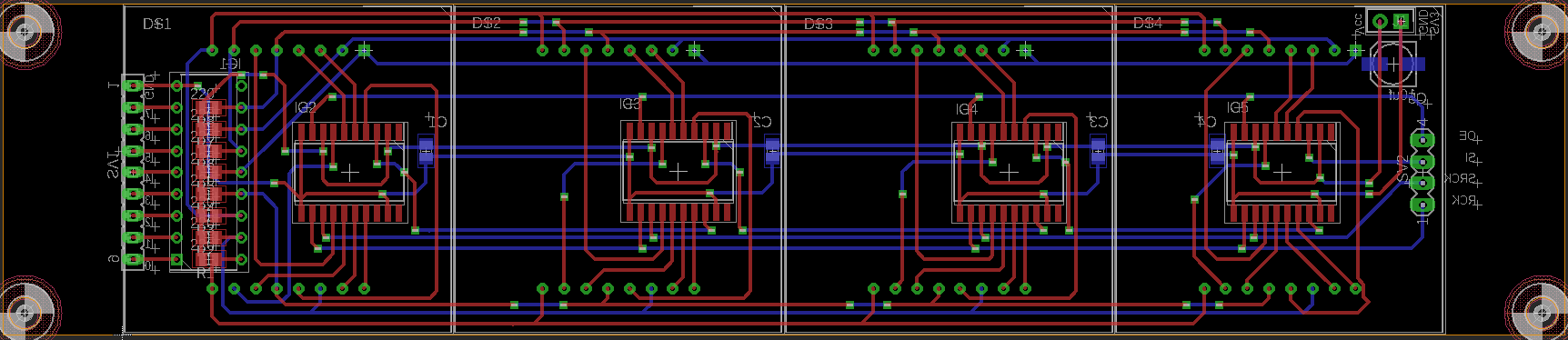


### PCB voeding

### Schema ledmatrix

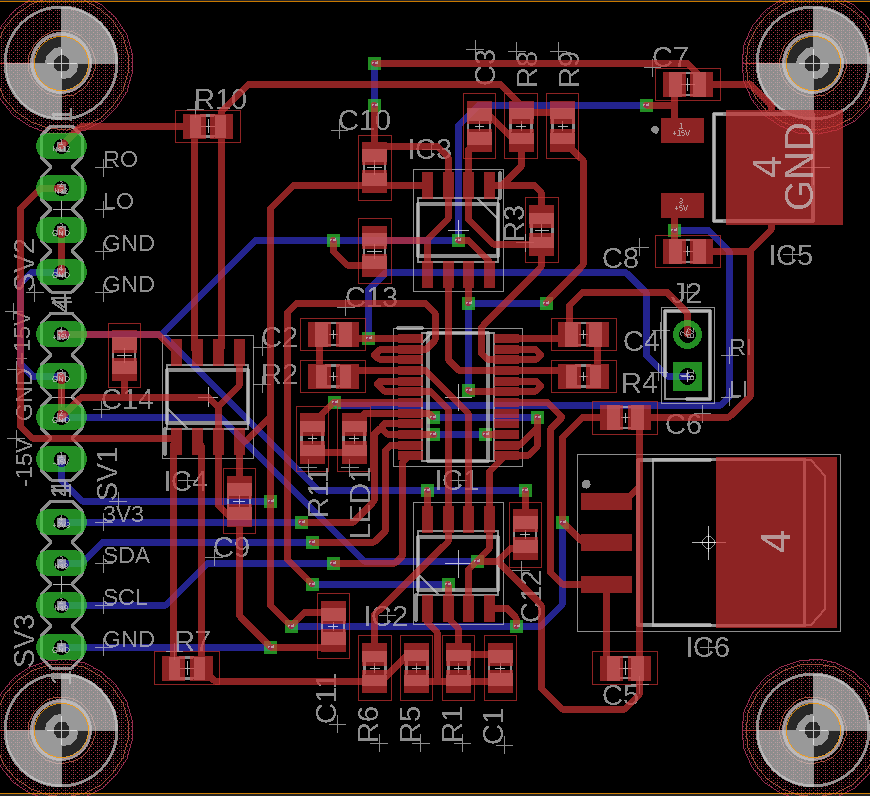


### PCB ledmatrix

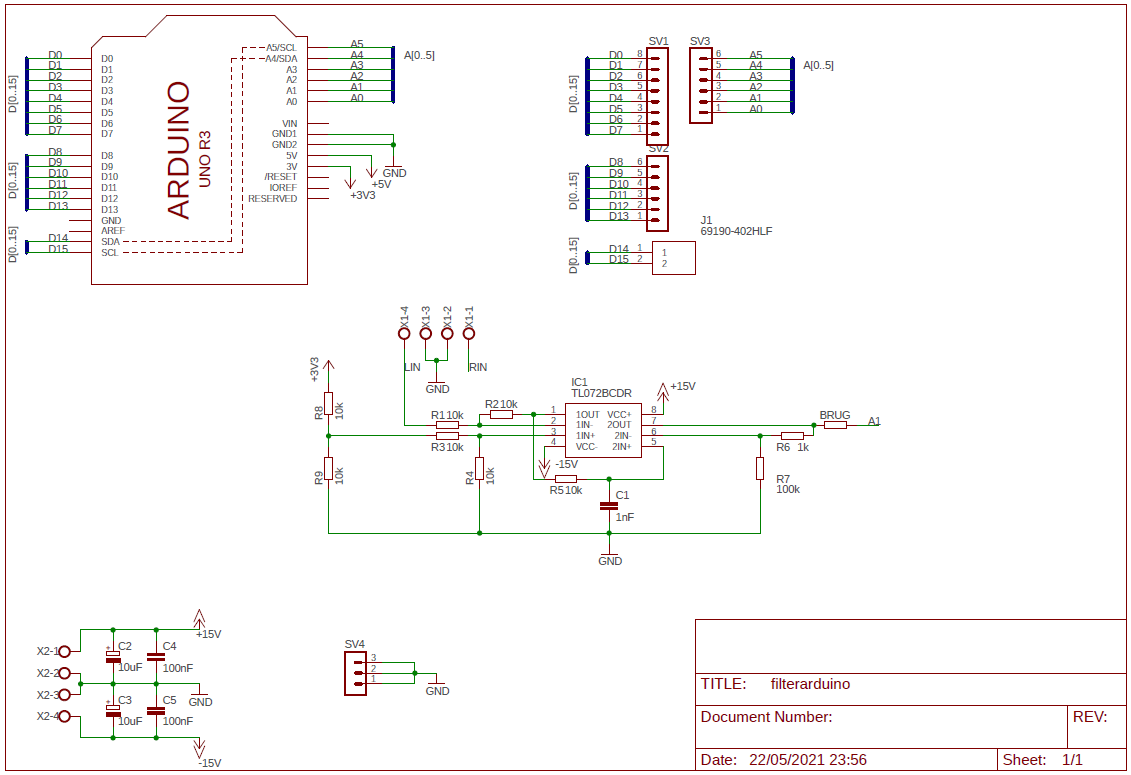


### Schema equalizer

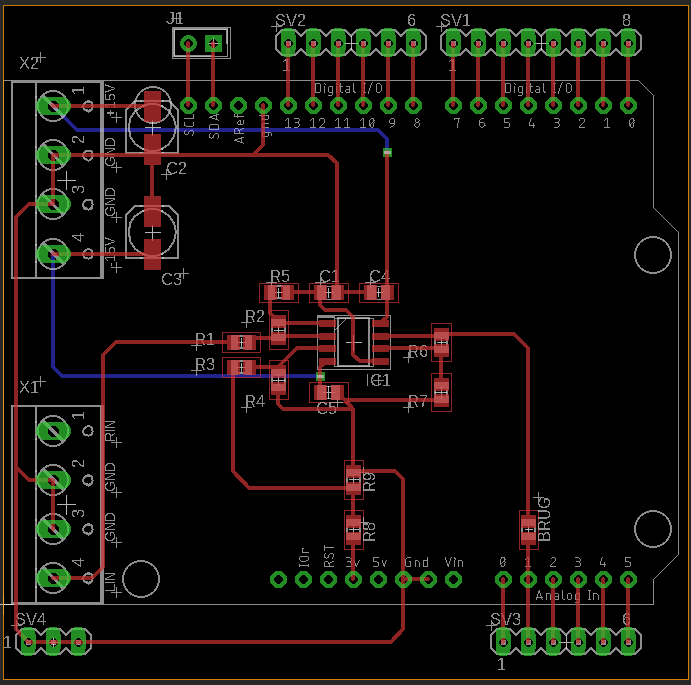
### PCB equalizer



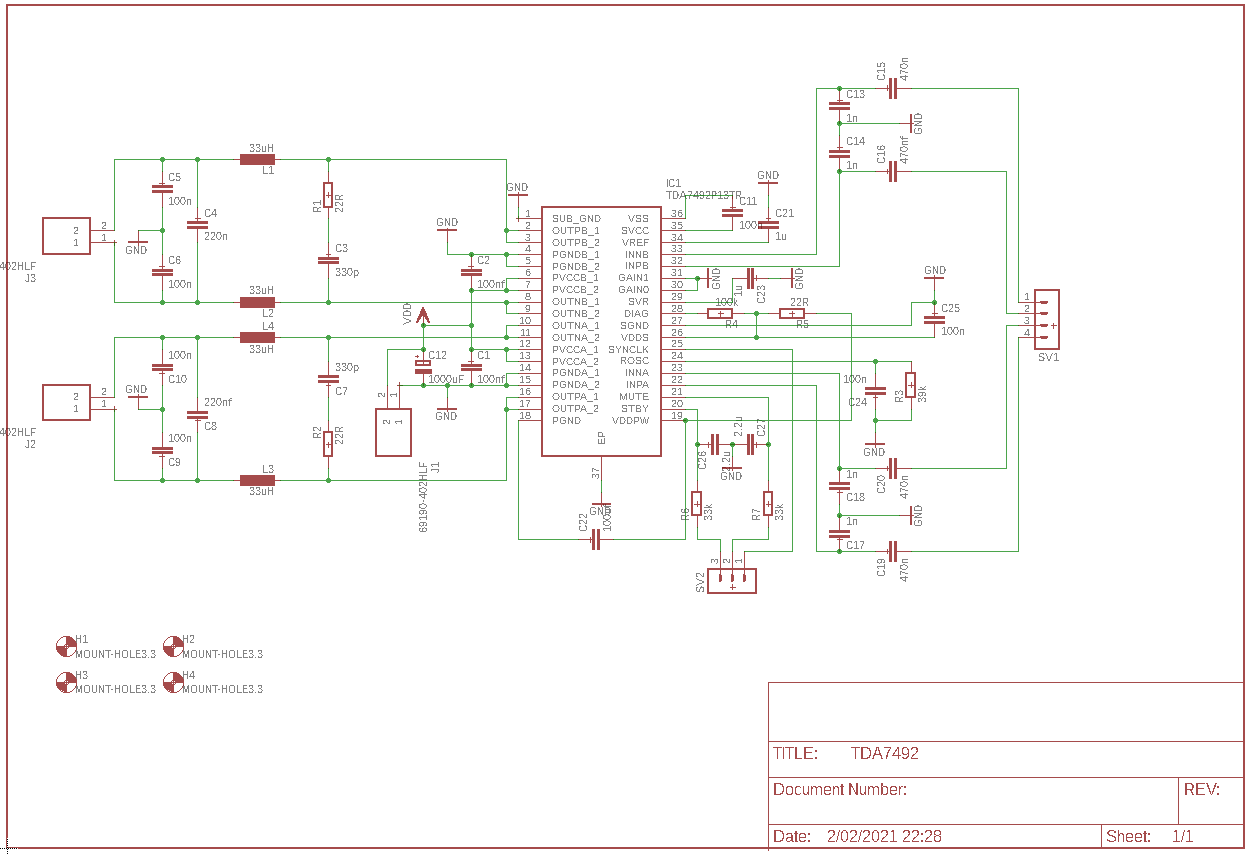
### Schema verschilversterker



### PCB verschilversterker



### Schema versterker



### PCB versterker

