Practice Enterprise Electronics

BlueTooth Speaker

Janssens Robbe

Schooljaar: 2020 - 2021

Docenten: Dhr. Dams – Dhr. Steen

Introductie

Voorwoord

Mijn naam is Janssens Robbe. Afgelopen jaren ben ik gestart aan mijn studies in Thomas More te Sint-Katelijne-Waver. Ik heb in het middelbaar in de eerste en tweede graad ASO-onderwijs les gevolgd. Daarna ben ik overgeschakeld naar de TSO-richting Elektronica-ICT omdat ik niet alleen meer praktijk gericht wou werken, maar ook meer met elektronica in het algemeen. Voor het Practice Enterprise 2 van 2020-2021 kregen we, net zoals bij Practice Enterprise 1, de vrijheid om zelf een project te kiezen. Mijn keuze was al snel gemaakt: De BlueTooth muziek speler. Het is niet alleen een kleinigheidje, maar ook wel leuk om thuis te hebben. Het ontwerpen en realiseren van zowel de hardware als de software leek mij een zeer toffe uitdaging! Ikzelf had nog nooit eerder gewerkt met BlueTooth en batterij-gevoede projecten. Dit doel heb ik behaald met behulp van trukjes en snufjes die ik de voorbije jaren geleerd heb.

Informatie auteur

Naam: Janssens Robbe
E-mail thuis: robbe.janssens27@gmail.com
E-mail school: r0787876@student.thomasmore.be
r-nummer: r0787876

Dankwoord

Om mijn project te kunnen realiseren, kon ik gelukkig meerdere keren terugvallen op het geloof en de steun van verschillende mensen. Als eerste willen we graag de docenten van het vak Practice Enterprise 2, Dhr. Dams en Dhr. Steen, bedanken. Vooral dankzij hun lessen, tips en advies is het me gelukt om mijn project te kunnen realiseren. Ook wil ik graag mijn klasgenoot Stijn Vandenbosch bedanken voor zijn hulpvolle ingevingen. Elke tip helpt!

Inhoud

1. Inleiding

Als “Practice Enterprise 2-project” heb ik gekozen om een eigen BlueTooth Speaker te maken. Het was een gerichte keuze omdat dit project verschillende domeinen van onze opleiding combineert, wat een leuke uitdaging was. Zo moest ik op een creatieve manier te werk gaan om mijn programmeer-, mechanische en elektronische kennis te gebruiken om een werkend geheel te verkrijgen.

Ik ben gestart met het zoeken naar de juiste componenten voor ons project. De grootste belangrijke componenten toch. De onderdelen in dit project waren nieuw voor mij, dus heeft het een tijd geduurd voordat ik op de geschikte componenten terecht kwam. De belangrijkste componenten in het geheel zijn: het grafische TFT LCD scherm met touch, STM32L443CCT microcontroller en de RN52 BlueTooth-module. Gaandeweg ben ik enkele keren tegen de lamp gestoten omdat ik telkens als ik een aanpassing deed niet altijd oplette op de rest van mijn schema. Na veel opzoekwerk en verbeteringen is het me toch gelukt.

Vóór ik meteen ga werken en me verder in mijn project ga wikkelen, is het natuurlijk een goed idee om aandacht te besteden aan wat ik juist wou dat het doet en hoe ik het ga realiseren. De beste manier om dat te bekomen, en ook de manier die ik iedereen aanraad, was om een blokschema op te stellen (verdere uitleg later). Nadat dat was gebeurd konden we beginnen met specifieker te werk te gaan:

- Het ontwerpen van de pcb
- Het opzoeken van informatie om
    - De microcontroller juist aan te sturen
    - De BlueTooth-module juist aan te sturen
    - Het scherm juist aan te sturen
    - De speciale RGB leds aan te sturen
    - …

Ik heb zelf mijn PCB gemaakt. Dit is altijd wat puzzelen: componenten zoeken, combineren, solderen, testen. Het hoort er altijd bij. Aan de andere kant wil ik niet zeggen dat de software dan weer het gemakkelijkste is. De juiste instellingen voor specifiek mijn project vinden en keer op keer opnieuw proberen tot ik het juiste heb is ook niet altijd even eenvoudig.

Mijn project bestaat grotendeels uit elektronica. Bij een BlueTooth Speaker komt er natuurlijk ook een mechanische behuizing aan te pas. Hoewel ik hier niet al te veel ervaring mee heb, is het mij toch gelukt om mijn idee met wat hulp te realiseren. Het totale idee is om iets in deze aard te maken: een hoofdschermpje, speakers, lichtjes en enkele bedieningsknoppen.

2. Gevolgde oplossingsstrategie

Zoals bij elk project, moet men eerst beginnen met een duidelijk beeld te creëren van wat hij/zij eigenlijk wilt. Hoe simpel het ook klinkt, dit eerste deel is al zeker niet het gemakkelijkste. Een accuraat idee hebben is al meteen van start gaan met een voorsprong!

De tweede stap is het verzamelen van componenten en de gegevens erover. Hierbij moet ik zeker opletten. Elke component heeft zijn eigen vereisten. Zo kan de ene als uitgang te weinig stroom of spanning leveren waardoor de volgende component de juiste signalen niet kan opvangen. Misschien heeft het uitgangssignaal niet de juiste vorm om op een andere ingang ingelezen te kunnen worden. Je moet echt naar alles kijken voor duidelijke en correcte communicatie. De juiste componenten zoeken neemt meer tijd in beslag dan dat men zou denken.

Nu dat ik een idee en de componenten heb, kan ik beginnen met het ontwerpen van een pcb. Ik ben begonnen met aparte delen stuk voor stuk te tekenen. Als je in 1 keer het hele plaatje zou beginnen tekenen, dan ga je gegarandeerd meer fouten tegenkomen dan wanneer je het stuk voor stuk gaat doen. Niet alleen ga je minder fouten krijgen, maar je zal ook zien dat het duidelijk blijft vanaf de start van het ontwerpen. Zelfs als je stuk voor stuk werkt, garandeer ik je niet dat het van de eerste keer correct zal zijn. Ik zelf ben 5x opnieuw moeten beginnen. Je moet als ontwerper met veel rekening houden. Kijk na of de banen niet te dik of te dun zijn of dat ze niet te dicht bij een andere baan liggen. Kijk na of de componenten in het echt ook geplaatst kunnen worden. Ook bij het ontwerpen van de pcb is er veel aandacht nodig om zo veel mogelijk fouten te voorkomen.

Nu de pcb klaar is, kan ik software code beginnen schrijven terwijl men mijn pcb maakt. Voordat je roekeloos code begint te typen, denk eerst na over wat je programma allemaal moet kunnen. Dit kan je bijvoorbeeld doen aan de hand van een flowchart. Het biedt niet alleen structuur aan, maar ook een mooie handleiding. Natuurlijk is dit niet verplicht, maar het helpt.
Ook hier zou je het best stuk voor stuk te werk gaan. Als je je programma van de eerste keer helemaal maakt, dan zal het zeer moeilijk zijn om de fouten terug op te sporen in je code. Als je het stuk voor stuk programmeert, dan zal je duidelijk zien waar de fout zit. Goed geschreven code moet ook duidelijk geschreven zijn.

Nu dat zowel de hardware als de software werkt, kunnen we beginnen aan de mechanische behuizing. Dit is niet zo moeilijk. Hier moet je goed opletten dat je dingen ontwerpen die in het echt ook mogelijk zijn om te maken. Kijk na wat de afmetingen zijn van je componenten zodat je iets niet te groot of te klein maakt. Mijn behuizing moet niet de stevigste zijn die er bestaat, maar natuurlijk moet de elektronica niet alle kanten op vliegen.

3. Projectvoorstel

3.1 Korte beschrijving

Als project voor Practice Enterprise 2 heb ik gekozen om een Bluetooth Music Speaker te maken. Het was een gerichte keuze omdat deze opdracht verschillende domeinen van mijn opleiding gaat combineren en omdat ik zelf graag naar muziek luister. Zo zal ik op een individuele en creatieve manier te werk gaan om mijn programmeer-, mechanische en elektronische kennis te kunnen gebruiken om een werkend geheel te verkrijgen.

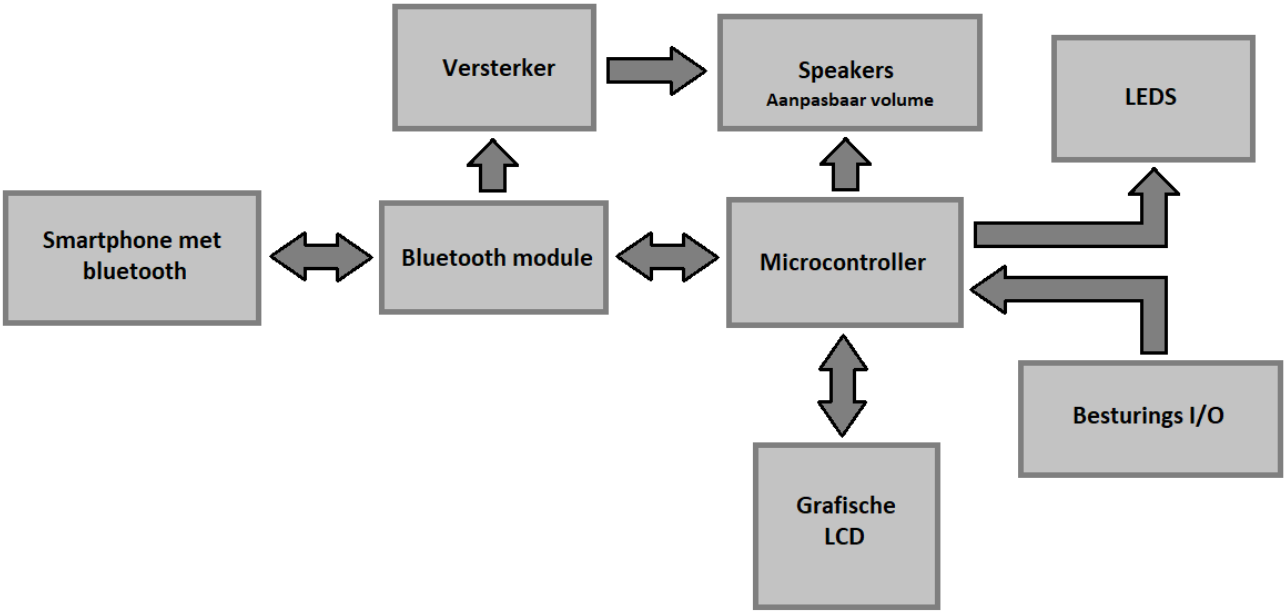
Als ik aan een Music Speaker denk, dan denk ik niet alleen muziek spelen, alhoewel dit wel het hoofddoel is, maar ook aan alle bijhorende parameters:

- Volume kunnen aanpassen
- Liedjes aanpassen (volgende, vorige, pauze)
- Ingebouwd scherm met
    - Klok (eventueel)
    - Naam van het liedje dat speelt
- Drukknoppen
    - Aan / uit
    - Zoeken naar apparaten om te verbinden
- Sfeerverlichting
- Batterijduur

Het is voor mij de eerste keer dat ik ga werken rond Bluetooth. Ook dit is een reden waarom ik dit project heb gekozen. Met de tijd mee zal alles hoe langer, hoe meer draadloos gebeuren. Alles zal draaien rond de communicatie tussen de microcontroller en de bluetooth-module.

Van dit project ga ik ongetwijfeld veel bijleren! Hopelijk zullen de Corona-toestanden zich hier niet te veel mee bemoeien…

3.2 Blokdiagram + beschrijving



Smartphone met Bluetooth:
Spreekt voor zich denk ik. Aangezien we niet willen rondlopen met kabels gaan we gebruik maken van Bluetooth om met het systeem te communiceren.

Bluetooth module:
Mijn project is een Bluetooth Music Speaker dus natuurlijk moet er een Bluetooth module inzitten. Deze module zal de “brug” zijn om met je smartphone te communiceren met de microcontroller. Men verbindt zijn/haar smartphone met de module en de module zal dan zelf een sessie opstarten met de microcontroller voor verdere communicatie. Welke Bluetooth module ik voor ogen heb, wordt later nog aangehaald.

Versterker:
Een music speaker moet in mijn ogen geen te laag volume kunnen afspelen. Aan de hand van een versterker gaan we dus het signaal versterken.

Microcontroller:
Net zoals in bijna elke schakeling is de microcontroller het “hart” van het systeem. Hier zal alles gebeuren wat men kan waarnemen. Bluetooth-signalen worden hier ontvangen en omgezet. De leds, het grafische scherm en de speakers worden aangestuurd door de microcontroller. Ook de microcontroller die ik wil gaan gebruiken, wordt later nog aangehaald.

LED’s:
Om het er allemaal een beetje mooi uit te laten zien, ga ik gebruik maken van zowel normale leds als ook specialere RGB’s met ingebouwde driver. Creatieve led-effects kunnen hiermee worden getoond. Met de normale leds kan ik bijvoorbeeld laten aantonen of het geheel aan staat of niet.

Grafische LCD:
Niet alle muziekboxen hebben een ingebouwd schermpje. De meesten hebben wat knoppen en speakers. Ik denk dat ik met een extra schermpje meer kan doen. Zo zal ik het proberen te gaan gebruiken om te laten zien welk liedje er speelt en om eventueel het volume aan te passen (er zal sowieso een externe potmeter voorzien worden voor het volume van de speakers). Ik zat er ook aan te denken om er een klein real time klokje op te weergeven.

Besturings-I/O:
Op de behuizing ga ik extra (draai)knoppen voorzien voor het gebruik te verbeteren. Er zullen knoppen komen om het volgende of het vorige liedje te kiezen. Met een draaiknop zal het volume kunnen worden geregeld. Een switch zal ook worden voorzien om het geheel aan en uit te zetten.

Speakers:
Spreekt voor zich denk ik. Hier komt het geluid uit.

Natuurlijk moeten de aparte delen ook gevoed worden. Later, bij de componentenkeuze, zal u zien dat zo goed als al mijn componenten met 3.3V kunnen werken.

3.3 Wat bestaat er al?

Mijn project dat ik heb gekozen is een zeer gekend systeem. Ik denk dat iedereen er al eens over heeft gehoord of toch al op zen minst reclame ervan heeft gezien. Zeker jongeren hebben meestal een eigen Bluetooth-speaker. Er bestaan honderden, misschien zelfs duizenden, soorten Bluetooth-speakers.

Eén van de speakers die ik voor het laatst nog heb mogen gebruiken is de JBL Charge 3 (rechts). Deze speaker heeft zijn unieke output en besturingssysteem om maar even een voorbeeld te geven.

Zo zijn er nog vele andere Bluetooth-speakers, maar binnenkort zal weer ééntje extra uitkomen.

3.4 Technologie verkenning

De belangrijkste componenten in mijn project zijn: de Bluetooth module, de microcontroller en het grafische scherm. Als we voor elke component even de benodigdheden opsommen, dan zullen we al sneller een keuze kunnen maken.

Bluetooth module:

- Moet met audio kunnen werken
- Crystal en oscillator voor digitale klok
- Operating voltage liefst rond de 3.3V-5V
- Range liefst tussen 5m-10m

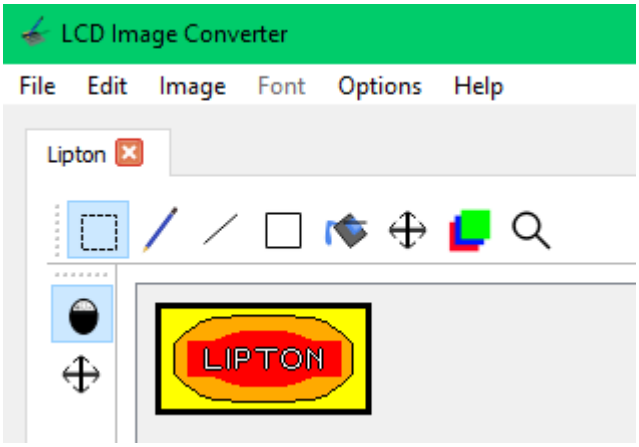
Microcontroller:

- Operating voltage rond de 3.3V-5V
- Geen klein geheugen aangezien we een grafisch scherm aansturen
- Verschillende communicatiemogelijkheden
- Laag verbruik

Grafisch scherm:

- Touch screen LCD
- Operating voltage rond de 3.3V-5V
- Laag verbruik
- Niet té klein (resolutie)
- Eventueel extra SSD-kaart houder voor als meer geheugen als het nodig is

Om alles juist te doen laten werken, ga ik gebruik maken van de software “MPLAB X IDE”. De code zelf gaat waarschijnlijk in C-code geschreven worden. Naast de code kan ik ook gebruik maken van de software “lcd image converter”. Hierin kan ik zelf tekeningen/symbolen maken met instelbare resolutie. De software genereerd dan zelf de .xml-file. Deze file kan ik dan gebruiken om op het scherm de afbeelding te weergeven. Hier is een voorbeeldje:



3.5 Component- en toolkeuze

Na enige tijd opzoekwerk heb ik mijn keuzes gemaakt. Als Bluetooth module ga ik de OVC3860 Bluetooth stereo audio module gebruiken. Waarom?

Hier zijn enkele features van de component:

- Heeft alle mogelijke functies die ik nodig heb (volgend/vorig lied, pauze/play, volume regeling, …)
- Heeft een analoge output
- Heeft een range van +/- 10m
- Kan met een operating voltage werken van 3.3V wat heel goed is aangezien de andere componenten ook met deze spanning werken

Ook staat er bij de applicaties dat deze module specifiek gebruikt wordt voor audio. Dus ik denk dat deze module een goede keuze is voor mijn project

Als grafische LCD heb ik gekozen voor de “2.8 inch ILI9341 240x320 SPI TFT LCD Display touch panel SPI Serial port module”.

Waarom?

Enkele features van het scherm:

- Het is een touchscreen LCD
- Heeft een operating voltage van 3.3V en 5.5V
- Laag verbruik aangezien de achtergrond uit LEDS bestaat
- Heeft een resolutie van 320x240 pixels verspreidt over een lengte van 8,6cm op 5cm
- Er is een mogelijkheid om een extra SSD-kaart toe te voegen voor geheugen

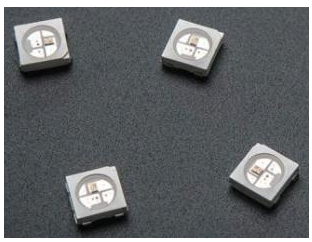
Als microcontroller heb ik gekozen voor de PIC18F27k40. Waarom?

Enkele features van deze microcontroller:

- Het geheugen is groot genoeg voor onze toepassing
- Heeft verschillende communicatie mogelijkheden waaronder SPI. Dit is handig aangezien het Grafische scherm aangestuurd wordt met SPI
- Operating voltage ligt tussen 1.8V en 5.5V
- Heeft een laag verbruik. Heeft zelfs een zekere “sleep-mode”.

Als speakers heb ik gekozen voor 2x 6.5inch (+/- 16,45cm) speakers.

Zoals ik reeds al heb aangehaald, ga ik gebruik maken van speciale RBG leds met ingebouwde driver. De WS2812B driver ic zal mij toelaten om leuke lichtsferen te creëren.



3.6 Budget raming

Met alle bovenstaande informatie kunnen we een gerichte schatting maken:

Component	Prijs	Aantal
Leds strips met WS2812B driver	€6 - €8 (€0,15 / led)	1
OVC3860 Bluetooth module	€3,5	1
Grafische LCD Touchscreen	€5,73	1
PIC18F27K40	€1,80	1
Speaker	€11,00	1 of 2
Besturings-I/O	€3,00	5
TOTAAL	€42,02 - €44,02	1
TOTAAL (zonder componenten die ik al heb)	€6,50	1

Het ziet er veel uit. Maar een paar van de componenten heb ik al in mijn bezit. Die kosten vallen dus al weg.
Componenten die ik al bezit:

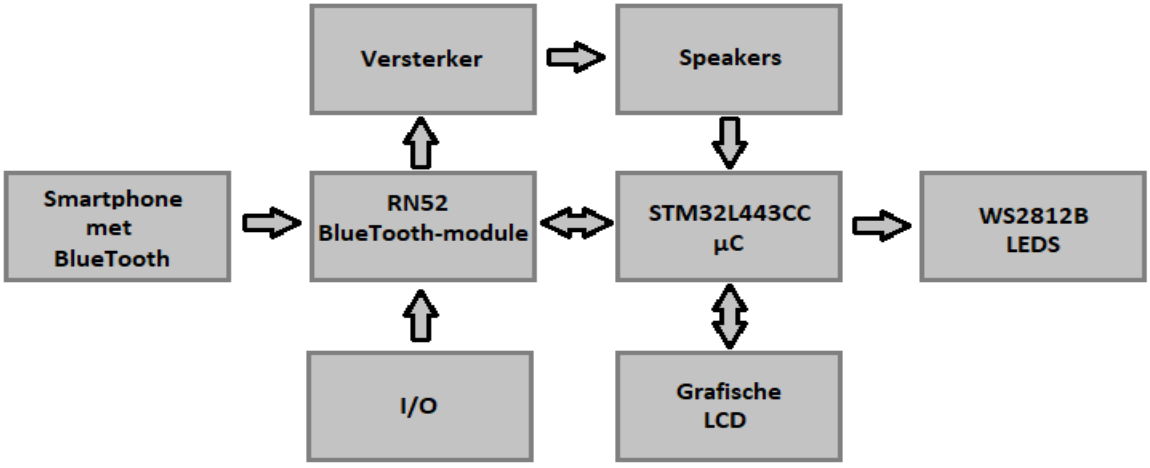
- 20 leds
- 1 grafische lcd scherm
- 2 pic microcontrollers
- 2 speakers (wel niet diegene die ik heb opgezocht)

4. Eindverslag

Hierboven werd mijn projectvoorstel beschreven. Dat was het “idee” en een “ruime schatting” van wat ik graag zou willen verwezenlijken. Het moest nog niet gedetailleerd zijn omdat het ook maar een voorstel moest zijn voordat het semester van Practice Enterprise 2 ook echt begonnen was. Vanaf dat ik echt aan mijn project mocht beginnen, had ik ook al een veel beter beeld van hoe het ik graag wil én wat ik graag wil. De uitleg die nu dus komt zal hier een daar verschillen tegenover mijn projectvoorstel.

4.1 Blokschema

Zoals altijd kunnen we best beginnen met een duidelijk beeld van mijn project.



Smartphone met BlueTooth:

Hier is niet veel extra uitleg voor nodig. Je moet natuurlijk een smartphone hebben om de muziek mee af te spelen via BlueTooth. De smartphones van tegenwoordig hebben allemaal BlueTooth dus is dit geen probleem. (andere devices met BlueTooth kan men ook koppelen, maar daar zie ik het nut niet echt van in)

Versterker:

Ik had de kans om heel vroeg in het jaar al eens speakers rechtstreeks via kabel met mijn smartphone te verbinden. Toen bleek dat de muziek redelijk goed werd afgespeeld, maar wel zeer zacht. Ik heb dat dan ook onthouden en er zeker voor gezorgd dat er in mijn project een versterker in ging zitten. De versterker ontvangt de “muzieksignalen” van de BlueTooth-module en versterkt deze wat resulteert in luidere muziek.

RN52 BlueTooth-module:
Deze module is de verbinding tussen mijn project en de smartphone. Het ontvangt de BlueTooth-signalen en stuurt deze door naar de versterker. Naast muziek kan deze module ook communiceren met mijn microcontroller voor extraatjes. Zo zou ik kunnen zien in welke mode de module zit, of er een event gebeurt, data halen uit de signalen van mijn gsm, … .

I/O:
Dit zijn slechts enkele drukknoppen om extern op de behuizing te zetten. De drukknoppen zorgen voor de “basis-handelingen”. Door de knoppen te bedienen kan men de muziek pauzeren en afspelen. Maar ook luider als zachter zetten. We hebben ook de keuze om het vorige liedje of het volgende liedje op te zetten.

Speakers:
Hier komt de muziek uit.

STM32L443CC µC:
De microcontroller is zowat het hartje van een project. In mijn project zal de microcontroller niet alleen communiceren met de BlueTooth-module het LCD scherm, maar ook de muzieksignalen inlezen en a.d.h.v deze signalen de LEDS aansturen.

Grafische LCD:
Op het scherm zal een soort van hoofdmenuutje worden afgebeeld. Daarop zal staan welk liedje er afspeelt en hoeveel het percentage van de batterij bedraagt. Naast deze dingen zullen er ook enkele symbooltjes worden weergeven om bijvoorbeeld de muziek zachter te zetten of om het volgende liedje af te spelen.

WS2812B LEDS:
Dit zijn RGB LEDS. Ze staan allemaal in serie waardoor we in 1 serieel datasignaal een hele LED-strip kunnen aansturen. Het toffe aan deze adresseerbare RGB LEDS is dat we er leuke sfeerlichtjes mee kunnen maken die gebaseerd zijn op de spelende muziek.

4.2 Componenten

Met de volgende componenten heb ik mijn project samengesteld:

N°	References	Value	Quantity	Footprint/ datasheet
1	C1, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22, C23, C24, C25, C26, C27, C40, C41, C48, C49, C51, C52, C53, C54, C55, C57	100nF	35	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
2	C28, C29, C30, C31, C32, C33	1µF	6	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
3	C35, C36, C37, C38	220nF	4	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
4	C44, C45, C46, C47	2µ2F	4	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
5	C2, C50	10uF	2	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
6	C34, C39	1nF	2	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
7	C56	10nF	1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
8	C3, C42, C43	100µF	3	C_0805_2012Metric_Pad1.18x1.45

				mm_HandSolder
9	R2, R3, R8, R9, R31	100k	5	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
10	R4, R5, R10, R11	220K	4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
11	R6, R7, R18, R28	10k	4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
12	R24, R25, R27, R30	33k	4	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
13	R12, R13, R17	100	3	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
14	R1	1k	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
15	R14	560	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
16	R15	562K	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
17	R16	107K	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder

18	R23	39k	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
19	R26	56k	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
20	R29	10	1	R_0805_2012Metric_Pad1.20x1.40mm_HandSolder
21	L1, L2, L3, L4	15µH	4	L_0805_2012Metric_Pad1.15x1.40mm_HandSolder
22	L6	1.5µH	1	L_1210_3225Metric_Pad1.42x2.65mm_HandSolder
23	L5	2.2uH	1	L_tps563200
24	D28	D_Schottky	1	D_SOD-128
25	D27, D29	RED	2	LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
26	D26	BLUE	1	LED_0603_1608Metric_Pad1.05x0.95mm_HandSolder
27	D1	LED	1	LED_D5.0mm
28	D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16, D17, D18, D19, D20, D21, D22, D23, D24, D25	WS2812B	24	LED_WS2812B_PLCC4_5.0x5.0mm_P3.2mm

29	U6	LM1117-3.3	1	LM1117
30	U4	STM32L443CCTx	1	LQFP-48_7x7mm_P0.5mm
31	U9	Spi_header	1	PinHeader_1x14_P2.54mm_Vertical
32	U12	LTC4001	1	QFN-16-1EP_4x4mm_P0.65mm_EP2.1x2.1mm
33	U3	RN52	1	RN52
34	U1	TL072	1	SO-8_3.9x4.9mm_P1.27mm
35	U2	SN74LVC1T45DBV	1	SOT-23-6
36	U7	TLV61048	1	SOT-23-6_Handsoldering
37	U5	FT232RL	1	SSOP-28_5.3x10.2mm_P0.65mm
38	U11	tpa3140	1	tpa3140
39	SW9	Switch	1	JST_XH_B2B-XH-A_1x02_P2.50mm_Vertical
40	SW1, SW2, SW3, SW4, SW5, SW6	SW_Push	6	SW_PUSH_6mm_H7.3mm
41	c2, c7, c10	10µF	3	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder

42	c1	1uF	1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
43	c8	0.22µF	1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
44	c9	0.1µF	1	C_0805_2012Metric_Pad1.18x1.45mm_HandSolder
45	BT3	Battery_Cell	1	JST_PH_B2B-PH-K_1x02_P2.00mm_Vertical
46	LS1, LS2	8 Ohm	2	PinHeader_1x02_P2.54mm_Vertical
47	JP5, JP6, JP8	Jumper	3	PinSocket_1x02_P2.54mm_Vertical
48	J1	Conn_01x04	1	PinSocket_1x04_P2.54mm_Vertical
49	J3	Conn_01x05_Male	1	PinSocket_1x05_P2.54mm_Vertical
50	J2	USB_B_Micro	1	USB_Micro-B_Amphenol_10104110_Horizontal

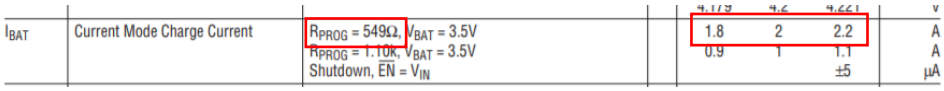
4.3 Hardware

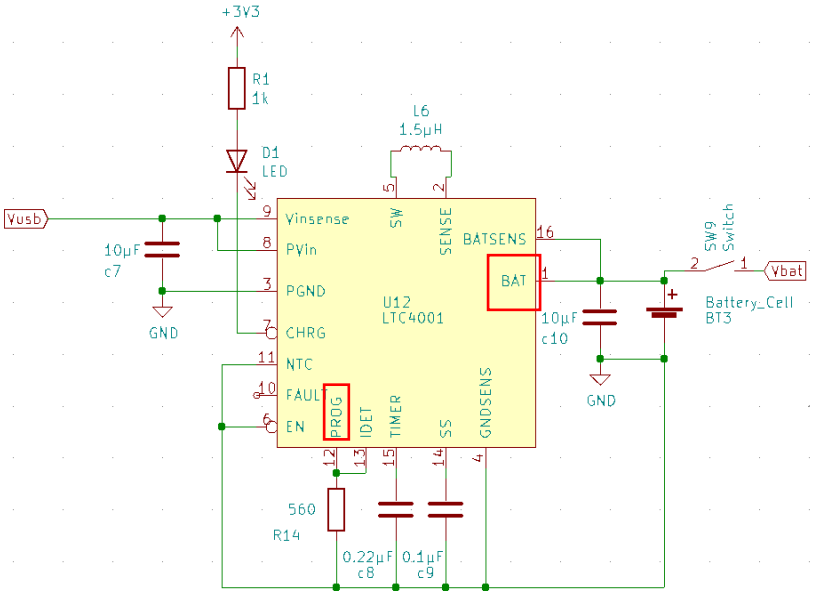
4.3.1 Schematic

Dit was voor mij een hele opgave. Zoals eerder vermeld zitten er in dit project enkele zaken waar ik persoonlijk voor het eerst mee in contact kom: BlueTooth, batterijen en audio. In de *documentatiebestanden* zal u zien dat ik meerdere malen van schematic ben veranderd. Mijn meest voorkomende fouten waren dat ik mijn batterij- en oplaad-gedeelte niet helemaal correct opstelde. Sommige componenten konden ook niet volledig correct met elkaar verbonden worden. Of het was een hele puzzel.

Iets waar ik ook zeer veel uit heb geleerd is de manier van schematics tekenen en hoe je juist alles moet plaatsen om een duidelijk geheel te krijgen.

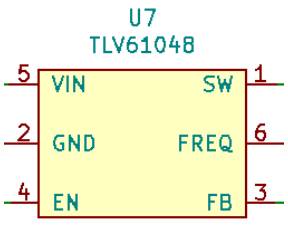
Na zeer veel opzoekwerk, verbeteringen en veranderingen ben ik toch op mijn finale schematic gekomen. Het grootste deel voor het schema spreekt voor zich: pins naar µc, i/o, leds, … . Toch wil ik graag slechts enkele zaken kort aanhalen.





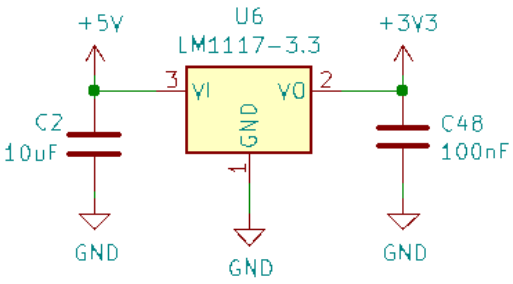
De batterij wordt opgeladen met een stroom van 2A.

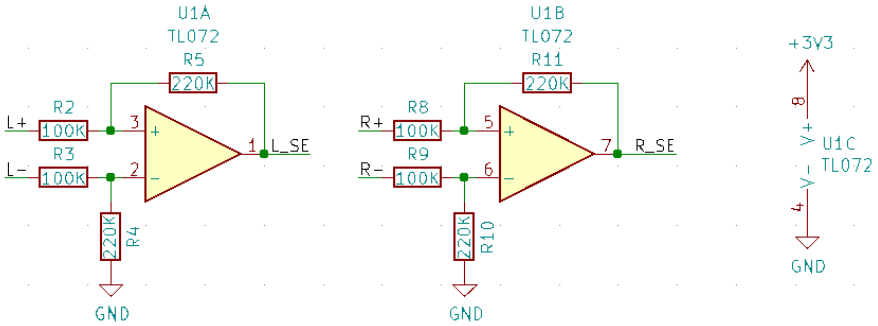
Een foutje dat ik pas later heb ondervonden is dat het ledje (linksboven) pas brand als het 3.3V krijgt. Deze spanning wordt pas gecreëerd als het hele systeem aan staat. Dus dat is niet echt nuttig als je de batterij moet opladen.



Dit is een chip die ervoor zorgt dat de batterijspanning wordt omgezet naar 5V. Deze 5V wordt gebruikt voor de versterker en de WS2812B LEDS te voeden.

De LM1117 zal de +5V omzetten naar +3.3V om de rest van het circuit de voeden (µC, BlueTooth-module, LCD, …).



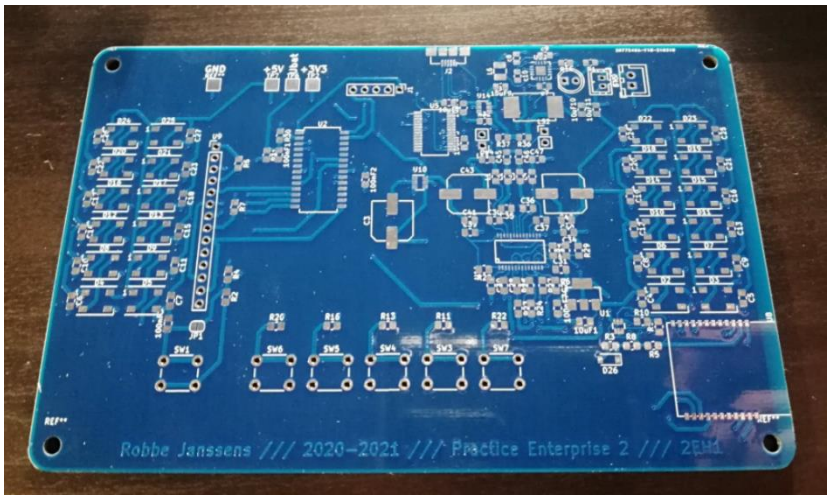


Bovenstaande screenshot weergeeft een verschilversterker. Audiosignalen bestaan uit differentiële signalen. Door de signalen door een opamp te sturen krijgen we het verschil tussen audio – en audio +.
Het verschil wordt weergeven met een positief signaal. Dit positief signaal lees ik in op 1 van mijn ingangen. Hoe groter het verschil, hoe groter de spanning, hoe groter mijn ingelezen waarde is.
Met deze ingelezen waarde kan ik sfeerlichten ontwerpen op mijn leds.

4.3.2 PCB

Ik heb verschillende schematics gehad, en dus ook meerdere pcb's getekend. Er zat niet overal evenveel verschil in, dus laat ik hier even mijn eerste pcb tegenover mijn finale.

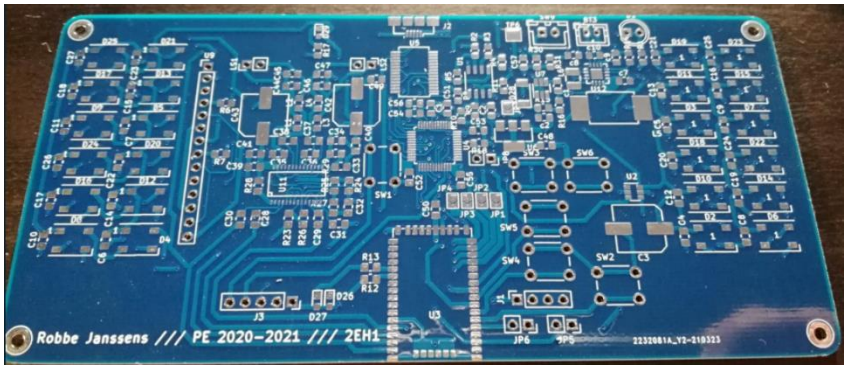
Op onderstaande foto ziet u mijn eerste versie van mijn pcb:



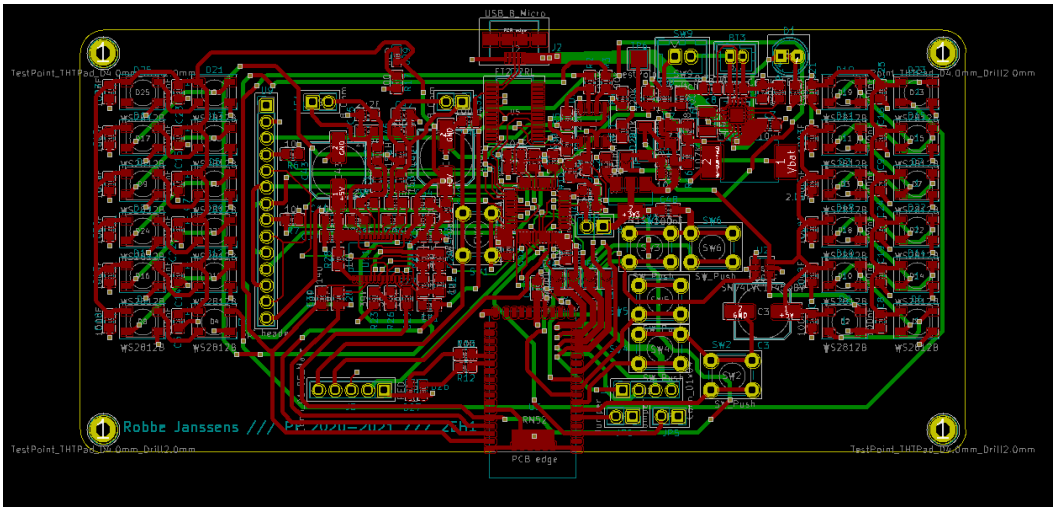
We zien hier duidelijk een heel slecht punt…

Ik maakte geen goed gebruik van mijn plaats wat resulteerde in een pcb met veel open ruimte. Het ziet er ook niet netjes uit…

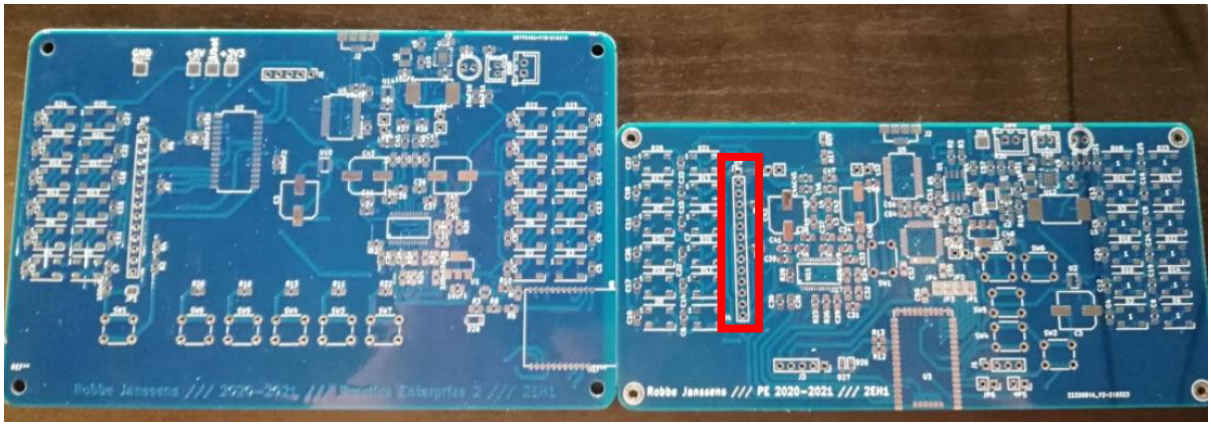
Op de volgende foto krijgt u te zien hoe mijn finale pcb eruit ziet:

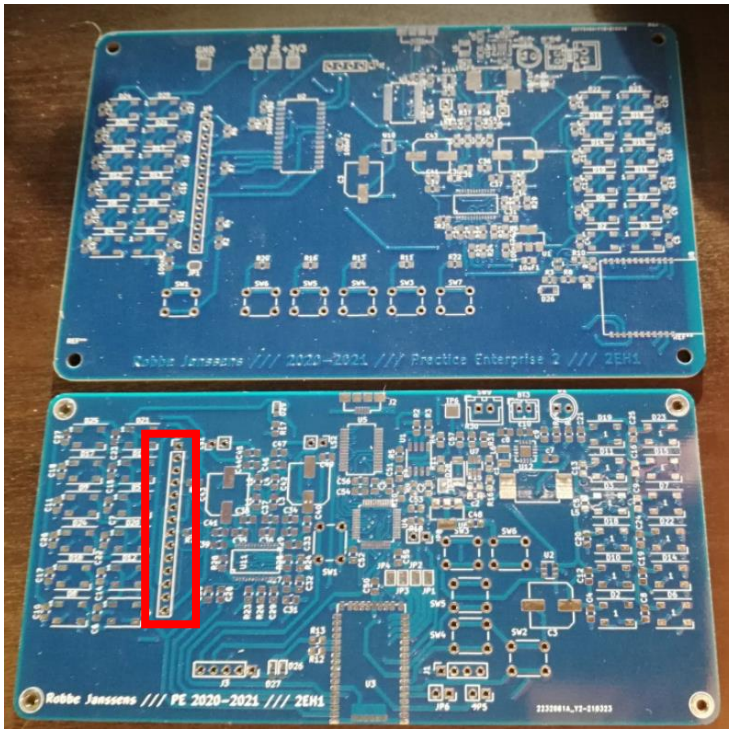


Dit ziet er al veel beter uit tegenover mijn eerste print.
Er staat niet alleen veel meer op, maar het is ook allemaal veel beter geplaatst. Het ziet er netjes uit en ik moest minder betalen.



Om ze toch even beter met elkaar te kunnen vergelijken heb ik ze dan ook naast elkaar gelegd:

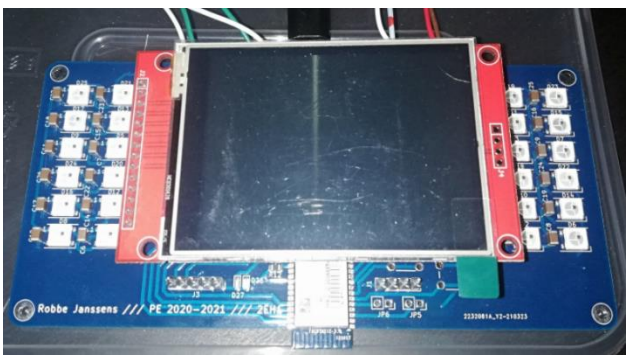


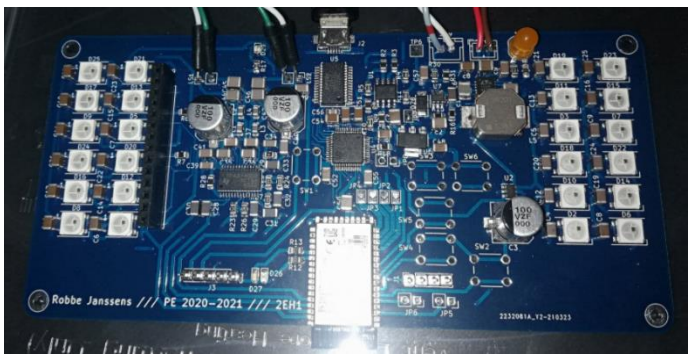


Zowel in de lengte als in de breedte zien we verschillen.

Ik heb expres de componenten zo geplaatst zodat we niets (of toch zeer weinig) van de elektronica zien. Deze is namelijk “bedekt” door het LCD-scherm (rode rechthoek).

Dit is zeer goed te zien op onderstaande foto’s:





4.4 Software

De volledige code kan u terugvinden in de bestanden. Toch heb ik mijn “main.c”-code onderaan in de bijlagen gestoken. Daar staan zo goed als alle belangrijke punten in.

4.4.1 LCD

Zie gebruikte library in de code in de projectdocumentatie

4.4.2 BlueTooth-module

Zie user guide voor commando’s en datasheet voor extra info over de module zelf

4.4.3 WS2812B

In mijn code zal u waarschijnlijk wel de volgende 2 lijnen hebben opgemerkt:

```
setLedColor( 1, 0, 0, 0); // (index, r, g, b)
updateLed( ); //kleur op ledje(s) zetten
```

Met “setLedColor” stel ik de kleur van een bepaald ledje in. In het gegeven voorbeeld laat ik het eerste ledje uitgaan. Waarom? De RGB-waarde 0 0 0 straalt nu eenmaal geen licht uit. Als er nu een waarde “0, 255, 0” stond, dan zou het eerste ledje een groene kleur hebben (rood, GROEN, blauw).
Met “updateLed” laat ik de ingestelde waarden ook daadwerkelijk naar buiten gaan.

Hoe deze dingen nu juist werken, daarvoor moeten we even in de datasheet van de WS2812B Led gaan kijken. Deze leds stuur ik aan a.d.h.v pwm. In de datasheet krijgen we te zien hoe een signaal eruit ziet om 1 led aan te sturen:

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

We merken op dat voor elke kleur 8 bits moeten verstuurd worden van MSB naar LSB.

Dus…we beginnen vanaf het begin. Met de eerste regel stellen we de buffer van de led in. Dit gebeurt als volgt:

```
void setLedColor(uint8_t ledIndex, uint8_t r, uint8_t g, uint8_t b)
{
    ledIndex--; //for people who ignore 0 -> suckers
    ledBuffer[ ledIndex * 3 ] = g;
    ledBuffer[ (ledIndex * 3) + 1 ] = r;
    ledBuffer[ (ledIndex * 3) + 2 ] = b;
}
```

“ledBuffer” is gedeclareerd als een array met 72 plaatsen. Waarom 72? Voor elke led heb ik 24 bits (3 bytes) nodig om elke kleur (rood, groen, blauw) voor te stellen. 24 (#leds) * 3 (#bytes/led) = 72

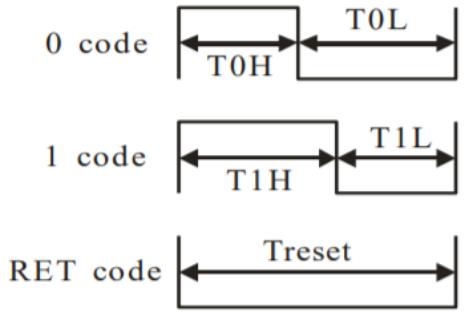
Daarna roep ik de functie “updateLed” op. Met behulp van naaststaande screenshot leg ik u uit wat er juist gebeurt.

Eerst maak ik een variabele om de informatie uit te buffer van de led te halen. Omdat deze leds heel tijdsgevoelig zijn, zet ik de interrupts voor de zekerheid uit (“__disable_irq();”).
Dan staan er enkele for-loops om voor elke led (24x) en elke kleur (3x) en voor elke bit van de kleur (8x) een juiste tijd te wachten. Als we terug het voorbeeldje “0, 255, 0” voor ALLE leds gebruiken, dan krijgen we dit tijdsschema:

24x (8xWSZERO, 8xWSONE, 8x WSZERO)

```
void updateLed( void )
{
uint8_t tempByte = 0x00;
//time sensitive code, disable incoming interrupt requests
__disable_irq( );
    //for every led
    for( uint8_t i = 0; i < NUM_PIXELS; i++ ) // 24 keer
    {
        //for every color
        for( uint8_t j = 0; j < NUM_BPP; j++ ) // 3 keer
        {
            tempByte = ledBuffer[ (i*3) + j ];
            //for every bit
            for( uint8_t q = 0; q < 8; q++ )
            {
                if( tempByte & (0x80 >>q) )
                {
                    WSONE
                }
                else
                {
                    WSZERO
                }
            }
        }
    }
    __enable_irq( );
    HAL_Delay( 1 );
}
```

In de datasheet kunnen we terugvinden hoelang we juist moeten wachten (zowel hoog als laag) om een “0” of “1” door te sturen.



We zien dat we voor een “0” het signaal hoog moeten maken voor een tijdsduur “T0H” en dan het signaal laag moeten maken voor een tijdsduur “T0L”.

Voor een “1” hebben we tijdsduren “T1H” en “T1L” nodig.

Wat een “probleem” is, is dat deze timings ZEER kritisch zijn zoals in de datasheet weergeven:

T0H	0 code ,high voltage time	0.4us	±150ns
T1H	1 code ,high voltage time	0.8us	±150ns
T0L	0 code , low voltage time	0.85us	±150ns
T1L	1 code ,low voltage time	0.45us	±150ns
RES	low voltage time	Above 50μs	

In het labo van het vak Datacommunicatie heb ik gewerkt met een functie dat instructies telt. Met behulp van enkele berekeningen kunnen we berekenen hoeveel keer we deze functie moeten oproepen om de juiste delay te bekomen.

```
void __attribute__((naked)) waitCycles(unsigned long ulCount)
{
    __asm("    subs    r0, #1\n"
          "    bne     waitCycles\n"
          "    bx      lr");
}
```

Mijn systeem werkt met een kloksnelheid van 80MHz.

$$\frac{1}{80\text{M}} = 13\text{ns per instructie}$$

3 nodig voor 1 count :

$3 * 13\text{ns} = 39\text{ns}$ (wij pakken 40ns omdat dit makkelijker is om mee te rekenen)

Dus nu hebben we 4 “delays” dat we moeten berekenen om op een correcte manier “0” en “1” uit te schrijven.

$$\mathrm{T0H} = \frac{400\mathrm{n}}{40\mathrm{n}} = 10 \text{ cycles} \qquad \mathrm{T0L} = \frac{850\mathrm{n}}{40\mathrm{n}} = 21 \text{ cycles}$$

$$\mathrm{T1H} = \frac{800\mathrm{n}}{40\mathrm{n}} = 20 \text{ cycles} \qquad \mathrm{T1L} = \frac{450\mathrm{n}}{40\mathrm{n}} = 11 \text{ cycles}$$

Nu we alle aantal cycles weten, kunnen we de juiste delays instellen. Op de screenshot van de vorige pagina kunnen we zien dat de macro’s (rood hieronder) “WSONE” en “WSZERO” werden opgeroepen:

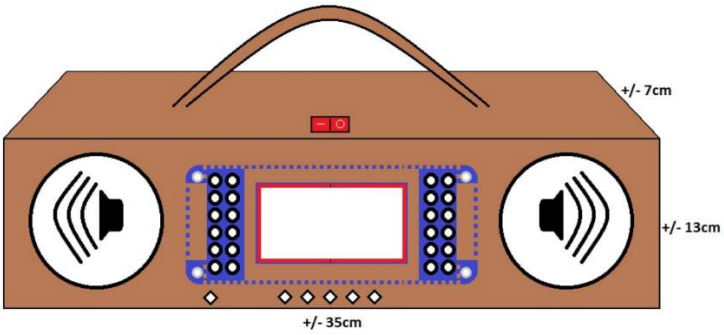
```
//led timing calculated for 80MHz clock speed
#define WSONEH  20
#define WSONEL  11
#define WSZEROH 10
#define WSZEROL 21

//led pins
#define LEDPORT GPIOB
#define LEDPIN  GPIO_PIN_13

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define SETHIGH(x,y) ((x)->BSRR = (uint32_t)(y))
#define SETLOW(x,y) ((x)->BRR = (uint32_t)(y))
#define WSONE   do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSONEH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSONEL);}while (0);
#define WSZERO  do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSZEROH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSZEROL);}while (0);
```

Om toch even een duidelijker beeld van te maken, verduidelijk ik even hoe een “1” wordt gevormd.

```
//led timing calculated for 80MHz clock speed
#define WSONEH  20
#define WSONEL  11
#define WSZEROH 10
#define WSZEROL 21

//led pins
#define LEDPORT GPIOB
#define LEDPIN  GPIO_PIN_13

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define SETHIGH(x,y) ((x)->BSRR = (uint32_t)(y))
#define SETLOW(x,y) ((x)->BRR = (uint32_t)(y))
#define WSONE   do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSONEH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSONEL);}while (0);
#define WSZERO  do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSZEROH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSZEROL);}while (0);
```

De macro’s “SETHIGH” en “SETLOW” maken setten en resetten de juist pin op de juiste port (waar mijn leds zijn op aangesloten).

4.5 Behuizing



Ik ben voor een meer standaard behuizing gegaan. Handvatje bovenaan, aan de voorkant het schermpje, de lichtjes, knopjes en speakers. De aan/uit knop heb ik ook bovenaan gezet zodat je deze minder makkelijk per ongeluk kan omschakelen.

Zo ziet de behuizing eruit (getekend in met het programma “Fusion 360”).

In realiteit ziet het er zo uit:

4.6 Projectdocumentatie

Gebruikte library voor LCD:
https://github.com/afiskon/stm32-ili9341/tree/master/Lib/ili9341

Datasheet WS2812B:
https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

RN52 datasheet + user guide/manuel:
http://ww1.microchip.com/downloads/en/DeviceDoc/rn-52-ds-1.0r.pdf
http://ww1.microchip.com/downloads/en/devicedoc/50002154a.pdf

Voor de rest van de componenten zijn de datasheets makkelijk terug te vinden op het internet. Bovenstaande links heb ik gebruikt voor het aansturen van mijn meest gebruikte componenten.

4.7 Filmpjes

Om de (test)filmpjes te kunnen bekijken, verwijs ik je door naar de bestanden in de zipfile.

4.8 Besluit

Doelstelling bereikt?

Ja, mijn doelstelling is bereikt. Ik heb een BlueTooth speaker kunnen maken, met een schermpje en lichteffecten erbij.

Discussiepunten vermelden

Mijn project doet wel wat het moet doen (muziek afspelen), maar toch zijn er enkele puntjes waar ik het zelf ook wat moeilijk mee heb. Alles werkt, slechts 2 puntjes niet. De ADC heb ik niet aan de praat gekregen, dus ik kan mijn audiosignalen en batterijspanning niet inlezen. Het gevolg hiervan is dat men niet weet wanneer mijn batterij leeg is en dat mijn lichtjes niet “dansen” op basis van de muziek.

Het 2e puntje dat niet werkt is de volledige communicatie mijn BlueTooth-module. Ik heb wel een opgenomen filmpje dat ik via mijn laptop (uart) kan communiceren naar de module, maar het communiceren tussen mijn µC en de module is niet gelukt. De aanpassingen die ik met mijn laptop kon doen, waren niet allemaal permanent. Dus elke keer dat mijn systeem uitschakelde, werden de default instellingen terug geactiveerd voor sommige settings.

Reflectie over mijn project

Voor mij was dit een project waar ik zeer veel uit geleerd heb. Als ik het opnieuw zou moeten doen, dan zou ik niet al te veel willen veranderen. Ik denk wel dat ik misschien mijn analoge spanningen op een andere manier zou willen binnenlezen. Hoe exact weet ik niet, maar op de manier dat ik het nu deed is het niet gelukt.

Misschien nog een extra puntje was om mijn pcb op te splitsen in meerdere stukken. Zo zouden bijvoorbeeld mijn leds, oplaadcircuit en scherm op verschillende printen kunnen liggen zodat het makkelijker zou zijn om in de behuizing te steken.

Reflectie tegenover bestaande projecten

Qua basisprincipe denk ik niet dat mijn project veel verschilt met bestaande speakers. Het enige wat mij wel opvalt, is dat ik niet zo veel kleine speakers met lichtjes en schermpjes te koop zie staan. Ondanks dat het schermpje niet helemaal volledig werkt, denk ik toch dat dit een pluspuntje is.

Eventuele aanbevelingen en/of opmerkingen

Ondanks het feit dat dit zo goed als allemaal nieuw was voor mij, denk ik dat ik wel mag vaststellen dat het project niet slecht is. Het is natuurlijk niet perfect, maar ik denk toch dat ik tevreden mag zijn met wat ik heb kunnen realiseren.

4.9 Bijlage

```
/* 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 BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "ili9341.h"
#include "test.h"
#include "connect.h"
#include <stdio.h>
//#include "ili9341_touch.h";
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define PWM_HI (38)
#define PWM_LO (19)

// LED parameters
#define NUM_BPP (3)
#define NUM_PIXELS (24)
#define NUM_BYTES (NUM_BPP * NUM_PIXELS)

//led timing calculated for 80MHz clock speed
#define WSONEH  20
#define WSONEL  11
#define WSZEROH 10
#define WSZEROL 21

//led pins
#define LEDPORT GPIOB
#define LEDPIN  GPIO_PIN_13

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
#define SETHIGH(x,y) ((x)->BSRR = (uint32_t)(y))
#define SETLOW(x,y) ((x)->BRR = (uint32_t)(y))
#define WSONE   do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSONEH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSONEL);}while (0);
#define WSZERO  do{ SETHIGH(LEDPORT,LEDPIN); waitCycles(WSZEROH); SETLOW(LEDPORT,LEDPIN); waitCycles(WSZEROL);}while (0);
/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart1;
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
//variables for led effects
uint8_t red=130, blue, green;

//led enable, batterijbesparing
uint8_t ENABLE_LEDS=1;

//coordinates touch
uint16_t x, y;

//battery voltage
char *Ubat;
uint8_t ready=0;
//max value voor ingelezen waarde
uint16_t testValue=0;

//pointer voor adc
uint8_t pointer[80];

//led buffer
```

```
uint8_t ledBuffer[NUM_BYTES];

/* 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_SPI1_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */
void __attribute__((naked)) waitCycles ( unsigned long ulCount ) ;
void ledClearBuffer ( void );
void setLedColor ( uint8_t ledIndex , uint8_t r , uint8_t g , uint8_t b ) ;
void updateLed ( void ) ;
void Check_ADC ( void ) ;
void Led_Effect ( void ) ;
void LCD_DrawMenu ( void ) ;
void Check_Power ( void ) ;
/* 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) //functie wordt opgeroepen als conversie klaar is (callback)
{
    // 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(&hadc1); //adc stond op continuous ingesteld
    //setLedColor(1, 0, 0, 0);
    //updateLed();
    ready = 1;
//  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);
}*/
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* 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_SPI1_Init();
  MX_USART1_UART_Init();
  MX_ADC1_Init();
  MX_USART2_UART_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  //HAL_ADC_Start( &hadc1 );
  HAL_StatusTypeDef stat ;
  stat = HAL_ADC_Start_DMA ( &hadc1 , ( uint32_t* ) &pointer[0] , 80 ) ;
      if ( stat != HAL_OK )
      {
          //setLedColor ( 1 , 0 , 0 , 0 ) ; // (index, r, g, b)
          //updateLed ( ) ; //kleur op ledje(s) zetten
      }

      stat = HAL_TIM_Base_Start_IT ( &htim2 ) ;
      if ( stat != HAL_OK )
      {
          //setLedColor ( 1 , 0 , 0 , 0 ) ; // (index, g, r, b)
          //updateLed ( ) ; //kleur op ledje(s) zetten
      }

      //alles initialiseren
      ILI9341_Unselect ( ) ;
      ILI9341_TouchUnselect ( ) ;
      ILI9341_LCD_Init ( ) ;
      //ILI9341_TouchSelect();

      HAL_GPIO_WritePin ( BACKLIGHT_GPIO_Port , BACKLIGHT_Pin , GPIO_PIN_SET ) ;
      HAL_GPIO_WritePin ( MODE_GPIO_Port , MODE_Pin , GPIO_PIN_RESET ) ;

      LCD_FillScreen ( LCD_BLACK ) ; //"clear" screen
      ledClearBuffer ( ) ; //inhoud buffer = cleared
      //alle leds uit
      for ( uint8_t i = 1 ; i<25 ; i++ )
      {
            setLedColor ( i , 0 , 0 , 0 ) ;
            updateLed ( ) ;
      }
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

    //Check_ADC();
    LCD_DrawMenu ( ) ;
    Check_Power ( ) ;
    Led_Effect ( ) ;

    //Even wachten
    HAL_Delay ( 100 ) ;
  }
  /* 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 PeriphClkInit = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 10;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != 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_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
  {
```

```
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_USART2
                              |RCC_PERIPHCLK_ADC;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInit.AdcClockSelection = RCC_ADCCLKSOURCE_PLLSAI1;
  PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC_PLLSOURCE_HSI;
  PeriphClkInit.PLLSAI1.PLLSAI1M = 1;
  PeriphClkInit.PLLSAI1.PLLSAI1N = 8;
  PeriphClkInit.PLLSAI1.PLLSAI1P = RCC_PLLP_DIV7;
  PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC_PLLQ_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1R = RCC_PLLR_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1ClockOut = RCC_PLLSAI1_ADC1CLK;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure the main internal regulator output voltage
  */
  if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{

  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
  hadc1.Init.Resolution = ADC_RESOLUTION_8B;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  hadc1.Init.LowPowerAutoWait = DISABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.NbrOfConversion = 1;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIG_T2_TRGO;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
  hadc1.Init.DMAContinuousRequests = DISABLE;
  hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
  hadc1.Init.OversamplingMode = DISABLE;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_6;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
```

```
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 60-1;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */

  /* USER CODE END TIM1_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 = 100;
  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_UPDATE;
  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 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 = 9600;
  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_RTS_CTS;
  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 */

}

/**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

/**
  * Enable DMA controller clock
  */
static void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Channel1_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

}

/**
  * @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_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, LCD_RST_Pin|MODE_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, TOUCH_CS_Pin|BACKLIGHT_Pin|LCD_CS_Pin|LCD_DC_Pin
                          |GPIO_PIN_13, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(WAKE_GPIO_Port, WAKE_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin : LCD_RST_Pin */
  GPIO_InitStruct.Pin = LCD_RST_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LCD_RST_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : TOUCH_CS_Pin BACKLIGHT_Pin LCD_CS_Pin LCD_DC_Pin */
  GPIO_InitStruct.Pin = TOUCH_CS_Pin|BACKLIGHT_Pin|LCD_CS_Pin|LCD_DC_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : TOUCH_IRQ_Pin */
  GPIO_InitStruct.Pin = TOUCH_IRQ_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(TOUCH_IRQ_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : PB13 */
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : EVENT_Pin */
  GPIO_InitStruct.Pin = EVENT_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(EVENT_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : WAKE_Pin */
  GPIO_InitStruct.Pin = WAKE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(WAKE_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin : MODE_Pin */
  GPIO_InitStruct.Pin = MODE_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_OD;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(MODE_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void __attribute__((naked)) waitCycles ( unsigned long ulCount )
{
    __asm("    subs    r0, #1\n"
          "    bne     waitCycles\n"
          "    bx      lr");
}

void ledClearBuffer ( void )
{
    memset ( ledBuffer , 0 , NUM_BYTES ) ;
}

void setLedColor ( uint8_t ledIndex , uint8_t r , uint8_t g , uint8_t b )
{
    ledIndex-- ;    //for people who ignore 0 -> suckers
    ledBuffer [ ledIndex * 3 ] = g ;
    ledBuffer [ (ledIndex * 3) + 1 ] = r ;
    ledBuffer [ (ledIndex * 3) + 2 ] = b ;
}

void updateLed ( void )
{
    uint8_t tempByte = 0x00 ;
    //time sensitive code, disable incoming interrupt requests
    __disable_irq ( ) ;
    //for every led
    for ( uint8_t i = 0 ; i < NUM_PIXELS ; i++ ) // 24 keer
    {
        //for every color
        for ( uint8_t j = 0 ; j < NUM_BPP ; j++ ) // 3 keer
        {
            tempByte = ledBuffer [ ( i * 3 ) + j ] ;
            //for every bit
            for ( uint8_t q = 0 ; q < 8 ; q++ )
            {
                if ( tempByte & ( 0x80 >> q ) )
                {
                    WSONE
```

```
                }
                else
                {
                    WSZERO
                }
            }
        }
    }
    __enable_irq ( ) ;
    HAL_Delay ( 1 ) ;
}

void Check_ADC()
{
    if ( ready )
    {
        Check_Power ( ) ;
        testValue = 0 ;
        //40MHz is frequentie bass
        //minstens het dubbele van gewenste frequentie gebruiken => 80
        //80 samples tegen 80Hz => per seconde in deze loop komen
        for ( uint8_t i = 0 ; i < 80 ; i++ )//80 waarden binnenlezen en gemiddelde berekenen
        {
            testValue = testValue + pointer [ i ] ;
        }
        testValue = testValue / 80 ;

        //LCD_WriteString(0, 20, &testValue, Font_11x18, LCD_WHITE, LCD_BLACK);
        //waarde omzetten naar string (om op lcd te zetten)
        itoa ( testValue , pointer , 10 ) ;// 10 = decimal
        /*if(ENABLE_LEDS==1)
        {
            setLedColor( 2, 0, 0, pointer);
            updateLed();
        }
        else
        {
            for(uint8_t i = 1; i<25; i++)
            {
                setLedColor( i, 0, 0, 0);
                updateLed();
            }
        }*/
        LCD_WriteString (150 , 110 , pointer , Font_11x18 , LCD_WHITE , LCD_BLACK ) ;
        ready = 0 ;
        HAL_ADC_Start_DMA ( &hadc1 , ( uint32_t* ) &pointer [ 0 ] , 80 ) ;
    }


    // Starten van de ADC Conversie
    HAL_ADC_Start ( &hadc1 ) ;
    // Wachten tot de conversie gedaan is
    // timeout=1mS maar kijken we niet na
    HAL_ADC_PollForConversion ( &hadc1 , 1 ) ;
    // ADC waarde inlezen en printen
    uint16_t result = HAL_ADC_GetValue ( &hadc1 ) ;
    //printf("result = %d\r\n", result);
    Check_Power ( ) ;
}

void Led_Effect()
{
    if ( ENABLE_LEDS == 1 )
    {
        for (/*var already declared globally*/; red >= 0 , blue < 130 ; red-- , blue++ )
        {
            for ( uint8_t led_row = 1 ; led_row <= 12 ; led_row = led_row + 2 )
            {
                if ( ENABLE_LEDS == 1 )
                {
                    setLedColor ( led_row , red , 0 , blue ) ;
                    setLedColor ( led_row + 1 , red , 0 , blue ) ;
                    setLedColor ( led_row + 12, red , 0 , blue ) ;
                    setLedColor ( led_row + 13, red , 0 , blue ) ;
                    updateLed ( ) ;
                    Check_Power ( ) ;
                    HAL_Delay ( 1 ) ;
                }
                else
                {
                    for ( uint8_t i = 1 ; i < 25 ; i++ )
                    {
                        setLedColor ( i , 0 , 0 , 0 ) ;
                        updateLed ( ) ;
                    }
                }
            }
        }
        Check_Power ( ) ;

        for (/*var already declared globally*/; blue >= 0 , green < 130 ; blue-- , green++ )
        {
            for ( uint8_t led_row = 1 ; led_row <= 12 ; led_row = led_row + 2 )
            {
                if ( ENABLE_LEDS == 1 )
                {
                    setLedColor ( led_row , 0 , green , blue ) ;
                    setLedColor ( led_row + 1 , 0 , green , blue ) ;
                    setLedColor ( led_row + 12 , 0 , green , blue ) ;
                    setLedColor ( led_row + 13 , 0 , green , blue ) ;
                    updateLed ( ) ;
                    Check_Power ( ) ;
                    HAL_Delay ( 1 ) ;
```

```
                }
                else
                {
                    Check_Power ( ) ;
                    for ( uint8_t i = 1 ; i < 25 ; i++ )
                    {
                        setLedColor ( i , 0 , 0 , 0 ) ;
                        updateLed ( ) ;
                    }
                }
            }
        }
        Check_Power ( ) ;

        for (/*var already declared globally*/; green >= 0 , red < 130 ; green-- , red++ )
        {
            for ( uint8_t led_row = 1 ; led_row <= 12 ; led_row = led_row + 2 )
            {
                if( ENABLE_LEDS == 1 )
                {
                    setLedColor ( led_row , red , green , 0 ) ;
                    setLedColor ( led_row + 1 , red , green , 0 ) ;
                    setLedColor ( led_row + 12 , red , green , 0 ) ;
                    setLedColor ( led_row+13 , red , green , 0 ) ;
                    updateLed ( ) ;
                    Check_Power ( ) ;
                    HAL_Delay ( 1 ) ;
                }
                else
                {
                    Check_Power ( ) ;
                    for ( uint8_t i = 1 ; i < 25 ; i++ )
                    {
                        setLedColor ( i , 0 , 0 , 0 ) ;
                        updateLed ( ) ;
                    }
                }
            }
        }
        Check_Power ( ) ;

    }
    else
    {
        Check_Power ( ) ;
        for ( uint8_t i = 1 ; i < 25 ; i++ )
        {
            setLedColor ( i , 0 , 0 , 0 ) ;
            updateLed ( ) ;
        }
    }
}

void LCD_DrawMenu()
{
    LCD_DrawVLine ( 59 , 0 , 59 , 28 , LCD_YELLOW ) ;
    LCD_DrawHLine ( 0 , 28 , 59 , 28 , LCD_YELLOW ) ;
    LCD_WriteString ( 4 , 6 , "Song:" , Font_11x18 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 65 , 11 , "Life is a Highway - Rascal Flatts" , Font_7x10 , LCD_WHITE , LCD_BLACK ) ;
    LCD_DrawHLine ( 0 , 80 , 320 , 80 , LCD_YELLOW ) ;
    LCD_DrawVLine ( 86 , 80 , 86 , 160 , LCD_YELLOW ) ;

    Check_Power ( ) ;

    LCD_DrawHLine ( 101 , 95 , 228 , 95 , LCD_YELLOW ) ;
    LCD_DrawHLine ( 101 , 145 , 228 , 145 , LCD_YELLOW ) ;

    LCD_DrawVLine ( 101 , 95 , 101 , 144 , LCD_YELLOW ) ;
    LCD_DrawVLine ( 229 , 95 , 229 , 144 , LCD_YELLOW ) ;

    Check_Power ( ) ;

    LCD_DrawVLine ( 244 , 80 , 244 , 160 , LCD_YELLOW ) ;
    LCD_DrawHLine ( 0 , 160 , 320 , 160 , LCD_YELLOW ) ;
    LCD_WriteString ( 18 , 105 , "LED" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 247 , 105 , "!LED" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 35 , 190 , "+" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 80 , 190 , "<<" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 143 , 190 , "PP" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 207 , 190 , ">>" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;
    LCD_WriteString ( 273 , 190 , "-" , Font_16x26 , LCD_WHITE , LCD_BLACK ) ;

    /*
    Check_Power();
    LCD_WriteString(4, 165, "Effect:", Font_11x18, LCD_WHITE, LCD_BLACK);

    LCD_DrawHLine(19,190,60,190,LCD_YELLOW);
    LCD_DrawVLine(20,190,20,225,LCD_YELLOW);
    LCD_WriteString(35, 200, "1", Font_11x18, LCD_WHITE, LCD_BLACK);
    LCD_DrawHLine(20,225,60,225,LCD_YELLOW);
    LCD_DrawVLine(60,190,60,225,LCD_YELLOW);

    LCD_DrawHLine(134,190,175,190,LCD_YELLOW);
    LCD_DrawVLine(135,190,175,225,LCD_YELLOW);
    LCD_WriteString(150, 200, "2", Font_11x18, LCD_WHITE, LCD_BLACK);
    LCD_DrawHLine(135,225,175,225,LCD_YELLOW);
    LCD_DrawVLine(175,190,175,225,LCD_YELLOW);

    Check_Power();

    LCD_DrawHLine(257,190,298,190,LCD_YELLOW);
    LCD_DrawVLine(258,190,298,225,LCD_YELLOW);
    LCD_WriteString(273, 200, "3", Font_11x18, LCD_WHITE, LCD_BLACK);
```

```
    LCD_DrawHLine(258,225,298,225,LCD_YELLOW);
    LCD_DrawVLine(298,190,298,225,LCD_YELLOW);*/
}

void Check_Power()
{
    if ( LCD_TGetC ( &x , &y ) )
    {
        if ( x > 80 && x < 160 && y > 0 && y < 80 )
        {
            //HAL_GPIO_WritePin(LEDPORT, LEDPIN, RESET);
            ENABLE_LEDS ^= 1UL << 1 ;
            HAL_Delay ( 200 ) ;
        }

        if ( x > 95 && x < 145 && y > 101 && y < 240 )
        {
            HAL_GPIO_TogglePin ( BACKLIGHT_GPIO_Port , BACKLIGHT_Pin ) ;
            HAL_Delay ( 500 ) ;
        }

        if ( x > 80 && x < 160 && y > 244 && y < 320 )
        {
            //HAL_GPIO_WritePin(LEDPORT, LEDPIN, SET);
            ENABLE_LEDS ^= 1UL << 1 ;
            HAL_Delay ( 200 ) ;
        }
    }
}
/* 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 */
  __disable_irq();
  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,
     ex: 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****/
```