

METALDETEKTOR PROJEKT

24. januar 2021

Udarbejdet af:



A handwritten signature in black ink, reading "Kim Christensen".

Kim H. Christensen - s181554



A handwritten signature in black ink, reading "J.D. Greve".

Jørgen D. Greve - s181519



A handwritten signature in black ink, reading "Mads Ludvig Stender".

Mads Ludvig Stender - s195130

DTU - DIPLOM

62739 - Elektromagnetiske sensorer og digital signalbehandling

Contents

1	Indledning	4
1.1	Metaldetektor kravspecifikation	5
2	Analyse	6
2.1	State of the Art (SOTA)	6
2.2	Metaldetektor blokdiagram	6
3	Design	8
3.1	Opstart	8
3.2	Tilstandsmaskine	8
3.3	Detektions signal	8
3.4	Analog til digital konvertering	8
3.5	Digital signalbehandling	9
3.6	Display	10
3.7	Effekt forstærker	10
3.8	Spoler	12
3.8.1	TX-spole	12
3.8.2	Feedback-spole	13
3.8.3	RX-spole	13
3.9	LC lavpas filter	14
3.10	Operationsforstærker	15
4	Implementering & test	16
4.1	Opstart	16
4.2	Tilstandsmaskine	16
4.3	Detektions signal	16
4.4	Analog til digital konvertering	17
4.5	Digital signalbehandling	17
4.6	OLED display	18
4.7	Start/stop knap	18
4.8	Nulstilling	18
4.9	Buzzer	18
4.10	Detektionsfølsomhed	18
4.11	Spoler	19
4.12	Effekt forstærker	19
4.13	Operationsforstærker	20
4.14	LC lavpas filter	21
4.15	Video demonstration	21
5	Konklusion	22
6	Bidrag	23
6.1	Bidrag	23
7	References	24
A	Appendix	26
A.1	Produkt fotos	26
A.2	System	28
A.3	Søgehoved	31
A.4	Operationsforstærker	32
A.5	C code	33

A.5.1	main.c	34
A.5.2	calcResult.h	36
A.5.3	OLEDinit.h	40
A.5.4	piezoBuzzer.h	41
A.5.5	startUp.h	42
A.5.6	timerCounter.h	48
A.5.7	ssd1306.c	49
A.5.8	ssd1306.h	57
A.5.9	I2C.c	59
A.5.10	I2C.h	61
A.5.11	data.h	62

1 Indledning

Denne rapport bygger på 3-ugers projektet ”62739 Elektromagnetiske sensorer og digital signalbehandling”, som er et selvstændigt projekt, betydende at det er gruppens eget ansvar at anvende de lærte metoder til at opbygge projektet, samt finde relevant viden. Under dette projekt lægges der stor vægt på den viden, som der er givet fra kurserne ”31035 Anvendt elektromagnetisme” og ”62743 Digital signalbehandling” for at kunne bygge en VLF¹ metaldetektor. Der er herunder givet en kravspecifikation til denne metaldetektor med prioriteten 1 og 2, som henholdsvis betyder **skal opfyldes** og **kan opfyldes**.

Ydermere, bruges projektet til at reflektere over vores lærte ingeniørmæssige færdigheder og tilgang til problemløsning. Der ligges dermed vægt på at kunne planlægge og dokumentere arbejdsfremgangen, samt kunne samarbejde i en gruppe, herunder at opgaverne bliver ligeligt fordelt, mens samtidig at kunne bidrage til hinandens opgaver hvis nødvendigt.

Sidst reflekteres der yderligere på, at kunne relatere og kunne formulere klare konklusioner mellem beregninger, simuleringer og ikke mindst implementering af sit design. Herunder, er det specielt vigtigt at kunne forklare og begrunde forskellen mellem simuleringerne og de målte resultater.

Planlægning af projektførløbet:



Figur 1: Gantt kort over projektførløbet

¹Very Low Frequency

1.1 Metaldetektor kravspecifikation

Nr.:	Navn:	Beskrivelse:	Prio.:	Opfyldt:
1	Amplitude/fase detektion	Det skal være muligt at kunne detektere amplitude og fase i det reflekterede signal	1	Ja
2	Metal type	Skal kunne skelne jern fra kobber, messing og aluminium	1	Ja
3	Distance krav	Skal kunne detektere en jerngenstand (radius på ca. 15 mm. og en længde på 50 mm) i en dybde på 50 mm. i fri luft.	1	Ja
4	Strømforsyning	Hele metaldetektoren skal kunne køre på et 9 volts batteri (batteristørrelse E / 6LR61) – kun et batteri.	1	Ja
5	Funktionstid	Metaldetektoren skal kunne køre minimum 100 minutter på batteriet og den ubelastede restspænding skal være større end 6 volt efter de 100 minutter.	1	Ja
6a	Processor design	Kredsløbet skal være baseret på en 1 Arduno	1	Ja
6b	Hardware design	Kredsløbene til at drive spolerne og forstærke det modtagne signal skal bygges med diskrete komponenter såsom: transistorer, operationsforstærkere, m.m.	1	Ja
7	Muligt software design	Brug Arduinoens ADC og kontroller den med timer interrupts	2	Ja
8a	Bruger interface display	Skal kunne udskrive amplitude og fase i et display	1	Ja
8b	Display udlæsningsstabilitet	Værdierne i displayet skal være læsbar for et flertal af brugerne (spørg om hjælp)	1	Ja
8c	Forbedret display udlæsningsstabilitet	Der anvendes et FIR eller IIR filter til 2 at lavpasfiltrere (middle) udlæsningen.	2	Nej*
9a	Bruger interface primære knapper	Der skal være en start/stop knap	1	Ja
9b	Bruger interface sekundære knapper	Der skal være en nul-stillingsknap (fratrækker værdierne når detektoren ikke er i nærheden af metal)	2	Ja
10	Detektions princip	Very Low Frequency (VLF)	1	Ja
11	Samplingsfrekvens	8 kHz +/- 100 Hz	1	Ja
12	Detektionsfrekvens	2 kHz +/- 100 Hz	1	Ja
13	Metaldetektorudformning	3D- printet / trækonstruktion	2	Ja
14	Spolernes bæreenhed	3D- printet spoleform / læserskåret plastik	2	Ja
15	Spoleudformning	Spolerne anvender en centreret udformning med sende-spolen yderst og feedback / modtager-spolen inderst.	2	Ja
16	Modtager spolens selvinduktans	Modtager spolen skal have en selvinduktans på mindst 10 mHy	1	Ja**

* Et IIR filter blev implementeret men er udeladt da det ikke forbedrede den i forvejen gode display stabilitet.

** Målt værdi af modtager spole er 9.9 mHy, dog grundet så lille en afvigelse, kan dette skyldes måleusikkerhed.

2 Analyse

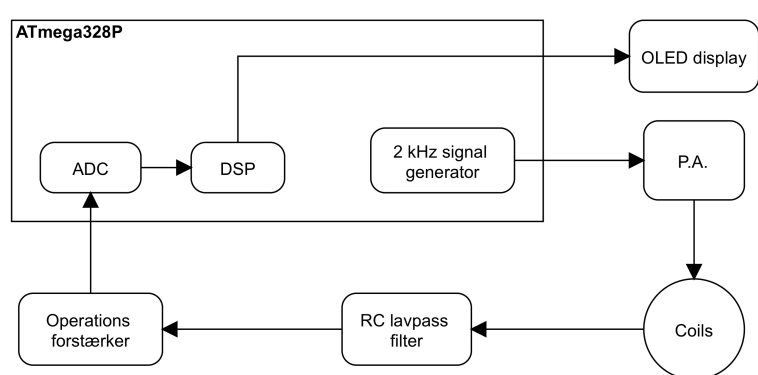
2.1 State of the Art (SOTA)

Der er efterhånden mange typer VLF metaldetektorer med meget forskellige kvalifikationer. I artiklen "Coil Basics" beskriver forfatteren omkring de bedste og mest omfattende metal detektorer, hvoraf de tre bedste er "DD"², concentric og 8-tal spoler. "DD" går under typen overlappende spoler, hvor disse typer daterer helt tilbage til 1830'erne, dog er den stadig meget anvendt, specielt anvender Minelab denne type detektor.[1]. En anden meget hyppigt anvendt metaldetektor er concentric-coil, som bygger på at have en ydre spole, med en mindre spole inde i sig. For at have en chance for at detektere en reflekteret signal, laves der en såkaldt "feedback" spole rundt om den inderste spole. Denne roteres modsat den yderstre, for at lave en modsatrettet magnetfelt. For at få et den mest præcise detektion, sættes modtager spolen inderst og så sender spolen yderst. Her vil feedback-spolen dermed vikles rundt om modtager spolen.

Til sidst er 8-tals spolen. Det smarte ved denne konfiguration er, da spolen vender om sig selv, vil det magnetiske felt gå ud med hinanden og derfor vil der være et klart signal til modtager spolen. Dog vil der være en fase skift på 180° fra den foreste del til den bagerste del og kan dermed er det kun muligt at korrekt detektere et metal på én af enderne. Dette kan reddes ved at lave en ekstra vending på spolen, hvilket vil give et center hvor faserne er balanceret.

8-tals spolen blev udelukket fordi denne virkede mere avanceret at designe og fordi det kunne være svært at detektere en forskel på metaller, grundet faseskiftet fra den forreste del til den bagerste del af søgehovedet. Valget mellem "DD"-coil og concentric-coil blev taget efter at have læst "Advanced Instrumentation for Polyharmonic Metal Detectors"[2] hvor "DD"-coil har sin fordel i mineraliseret jord, men er mindre sensitiv end en concentric-coil, af samme størrelse, i ikke mineraliseret jord. Concentric-coil har et bredere søgefelt end for eksempel "DD"-coil, men da der ikke var noget krav om at kunne pinpointe et objekt sås dette ikke som en ulempe. Derfor blev concentric-coil valgt til søgehoved i projektet, da vi ud fra de stillede krav skulle kunne skelne mellem jern og kobber, messing, aluminium og fordi vi ønskede at have et søgehoved der gav maksimal sensitivitet.

2.2 Metaldetektor blokdiagram



Figur 2: Blokdiagram over metaldetektor

Ovenstående blokdiagram viser den planlagte fremgangsmåde til designet af metaldetektoren. Det ses at micro controlleren(MC) styrer en Power Amplifier (P.A.) der skal booste 2kHz signalet fra MC op således at amplituden bliver taget fra 6LR61 (9-volt) batteriet for at få mest mulig strøm ud til spolen i søgehovedet. Coils bliver søgehovedet der laves som en concentric-coil og derfor består af en TX-spole der udsender et magnetfelt drevet af P.A. En feedback spole skal udsende et modsatrettet magnetfelt af samme størrelse som

²Også kendt som double-D

TX-spolen, for at RX-spolen kan opfange reflekterede magnetfelter fra metalobjekter. RX-spolen kobles til et lavpas filter for at blokere højere frekvenser end 2kHz og dernæst forstærkes det modtagne signal vha. en operationsforstærker for at MC kan registrere det. ADC omsætter det analoge signal til digitale værdier og disse bliver processeret vha. digital signal behandling (DSP) hvorefter amplitude og fase, på det modtagne signal, bliver udskrevet på et OLED display.

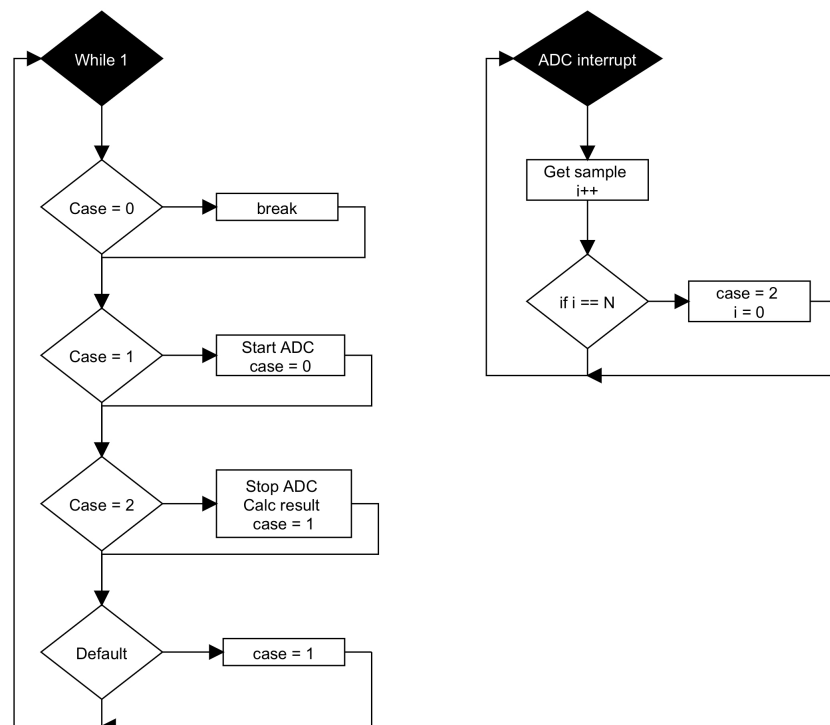
3 Design

3.1 Opstart

Når metaldetektoren starter op skal den nulstille sig selv så den er klar til brug.

3.2 Tilstandsmaskine

Tilstandsmaskinen skal sørge for at holde processerne i MCU'en³ adskilt således at der ikke samples når der beregnes. Den starter op i tilstand 1 og dermed starter samplingen. Så tager ADC⁴ interrupt over og styrer samplingen indtil det ønskede antal samples er lagret i et array hvorefter der ændres tilstanden til 2 hvor ADC'en stoppes, resultatet beregnes og sendes til displayet. Nu ændres tilstanden til 1 og der startes forfra. Se flowchart herunderfor overblik.



Figur 3: Tilstandsmaskine

3.3 Detektions signal

Detektionssignalet skal være et 2 kHz firkant signal der bliver genereret i Arduinoen. Dette gøres ved at udnytte at ADC samplingen sker fire gange hurtigere således at en port henholdsvis sættes høj og lav hver anden gang ADC interruptet sker.

3.4 Analog til digital konvertering

Når detektionssignalet kommer retur til Arduinoen skal det konverteres til et digitalt signal der kan beregnes på i MCU'en. Dette skal foregå ved 8 kHz og det gøres mest præcist ved at udføre samplingen hver gang en timer giver interrupt. En 8 bit timer tæller fra 0 til 255 og hvis den prescales med 8 kommer den ned

³Microcontroller unit

⁴Analog to digital conversion

på en tællefrekvens på 2 MHz og hvis man så sætter den til at give interrupt hver gang den har talt til 249, så rammer man en sampling frekvens på præcis 8000. Beregningen af sammenligningsværdien kan gøres på følgende måde.

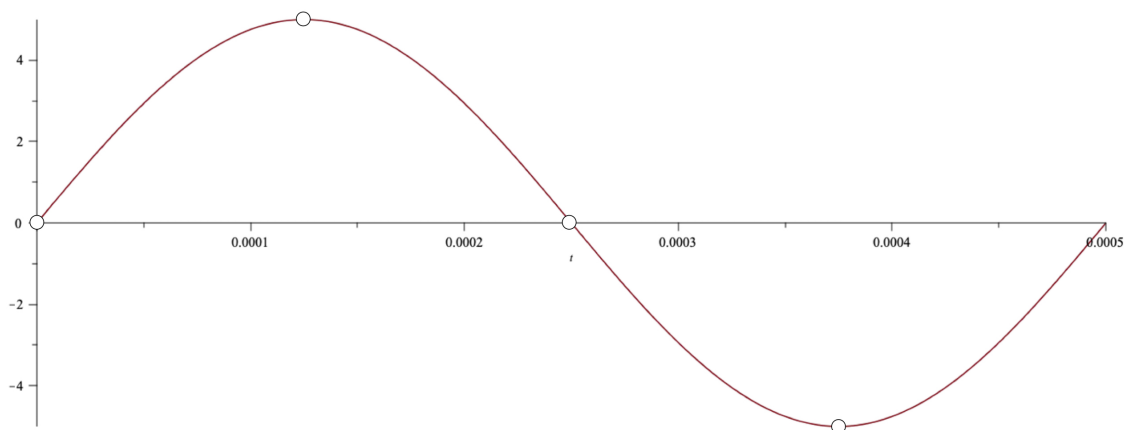
$$OCRnA = \frac{CLK \cdot T - N}{N} = \frac{16e6 \cdot \frac{1}{8000} - 8}{8} = 249$$

Hvor $OCRnA$ er sammenligningsværdien, CLK er system klokfrekvensen, T er den ønskede tid mellem samples og N er prescale faktoren.

Hver gang ADC'en har taget en sample skal værdien gemmes i et array og når den har taget det ønskede antal samples bliver der beregnet og udskrevet til displayet.

3.5 Digital signalbehandling

Den digitale signalbehandling går ud på at finde den information man skal bruge ved en bestemt frekvens i detektionssignalet og derefter beregne amplitude og fase som skal udskrives til displayet. Da detektionssignalet er et 2 kHz signal og der samples ved 8 kHz betyder det at den 2 kHz sinus kurve vi ønsker information omkring bliver samplet de samme fire steder hver periode.



Figur 4: Sampling af 2 kHz sinus med 8 kHz (hvide prikker)

Med den information kan vi se at vi helt kan slippe for diskret fourier transformation til at trække information ud af signalet. Hvis man tænker på enhedscirklen når man ser på sinus kurven kan man se at hvis vi tager henholdsvis cosinus og sinus til 0 , $\frac{\pi}{2}$, π og $3\frac{\pi}{2}$ så får vi følgende resultat:

	0	$\frac{\pi}{2}$	π	$3\frac{\pi}{2}$
cos	1	0	-1	0
sin	0	-1	0	1

Nu kan vi blot multiplicere vores resultat array med ovenstående på de respektive pladser og summere det indtil N samples.

ReSum = Σ	$s1 \cdot 1$	$s2 \cdot 0$	$s3 \cdot -1$	$s4 \cdot 0$	$s5 \cdot 1$	$s6 \cdot 0$	$s7 \cdot -1$	...	$sN \cdot 0$
ImSum = Σ	$s1 \cdot 0$	$s2 \cdot -1$	$s3 \cdot 0$	$s4 \cdot -1$	$s5 \cdot 0$	$s6 \cdot -1$	$s7 \cdot 0$	...	$sN \cdot -1$

$s1 = \text{sample } 1$, $s2 = \text{sample } 2$, ..., $sN = \text{sample } N$

For at komme frem til det endelige resultat som skal udskrives på displayet skal vi finde amplituden som er modulus af signalet og fassen som er argumentet af signalet.

$$A = \sqrt{ReSum^2 + ImSum^2}$$

$$\theta = \arctan\left(\frac{ImSum}{ReSum}\right)$$

3.6 Display

Amplituden og fassen af signalet skal udskrives på et display. Til det formål har vi valgt et OLED display, dels fordi vi synes at det har en fornuftig udskriftskvalitet og dels fordi det er en type display vi er vant til at arbejde med. For at undgå at det står og skifter for hurtigt så man ikke kan nå at se hvad der står på displayet skal resultatet filtreres inden det bliver sendt til displayet. Dette kan gøres med et IIR filter hvor man tager en procentdel af det forgående resultat og lægger sammen med en procentdel af det nuværende resultat:

$$x_{out} = (x_{in} \cdot 0.9) + (0.1 \cdot (x_{in} - 1))$$

3.7 Effekt forstærker

For at designe P.A'en⁵ anvendes bipolære transistorer, grundet deres evne til at styre op til strømme, som kan være op til 400 gange basestrømmen (afhængig af transistormodellen). Dette er specielt en god egenskab, da jo større strøm der løber gennem sender-spolen, jo kraftigere er magnetfeltet spolen laver. Dog, da hele metal detektoren skal udelukkende køre på ét 9V batteri, og samtidig kunne køre i 100 min med en efterfulgt restspænding på over 6V, skal strømforbruget begrænses der hvor det kan.

Samtidig kan der ved brug af NPN og PNP transistorer laves en såkaldt AB-klasse forstærkning, hvilket er specielt vigtigt, da spolen skal have vekselspænding/strøm for at fungere. Ved at bruge Arduinoen som signalgenerator, som sender et 2kHz firkant signal (med 50% duty-cycle), kan der laves en tilnærmet sinuskurve over sender-spolen.

Til at designe P.A'en er der blevet valgt NPN transistoren BC337-25 og PNP transistoren BC327-25, da disse har en god forstærkning ($\beta \approx H_{FE} = [160, 400]$), samt både komponentshoppen og LTspice har disse modeller. Collector strømmen bestemmes ved

$$I_c = \beta \cdot I_b \quad (1)$$

Ud fra databladet på et Duracell 9V batteri (6LR61), kan ses at ved en strøm på 200mA, vil der efter 100 min. være over 6 volt i restspænding [3]. Heraf, kan der altså bruges 200mA på hele systemet og stadig at kunne opfylde krav 5.

For at udregne en modstand til styre base strømmen, I_b , kræves den konkrete forstærkning kendes. Og som forklaret tidligere, ses der i begge datablade for BC337[4] og BC327[5], at forstærkning er mellem 160-400. Dog gælder disse værdier for en collector-strøm på 100mA og en collector-spænding på 1V (-100mA og -1V for PNP-transistoren), som der nødvendigvis ikke vil være det ønskede kredsløb. Dernæst, er det et stort interval forstærkningen kan ligge imellem, og gør det svært at beregne en konkret modstand. Dog ved hurtig udregning på en størrelse der kan justeres lidt på, kan der regnes baglæns. Der er 200mA tilgængelig for hele systemet, og der afsættes 50mA til Arduinoen m.m., hvilket giver en tilgængelig collector strøm $I_c = 150mA$. Heraf kan der udledes

$$I_b = \frac{V_b}{R_b} \quad (2)$$

Og med ligning (1) kan disse samles til følgende:

⁵Power Amplifier

$$I_c = \beta \cdot \frac{V_b}{R_b} \quad (3)$$

Og ud fra karakteristikken ved bipolære transistorer, vil dioderne trækker omkring 0.7V og der vil derfor være omkring 8.3V på base benet. Forstærkningen sættes på dens minimum på 160, og nu vil der kun være en ukendt:

$$150 \text{ mA} = 160 \cdot \frac{8.3 \text{ V}}{R_b}$$

⇕

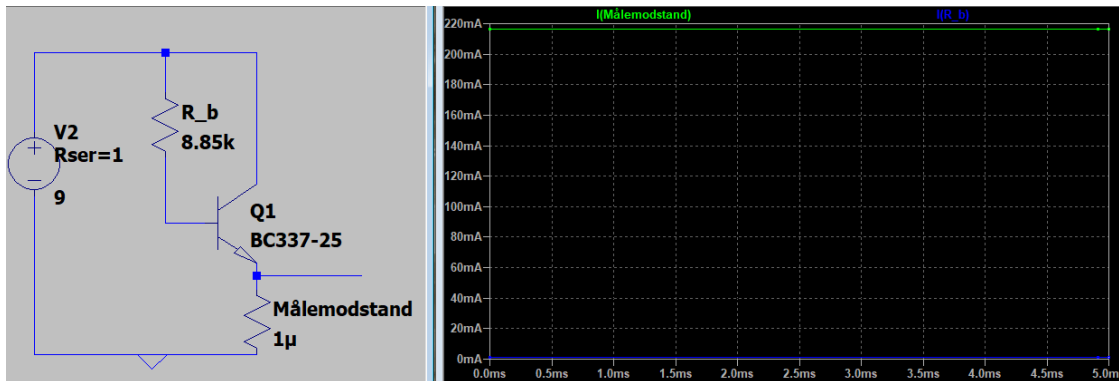
$$R_b = 160 \cdot \frac{8.3 \text{ V}}{150 \text{ mA}} = 8.85 \text{ k}\Omega$$

Dog, når der laves en simulering ses det at forstærkningen ikke er som den mindste der er angivet. Dette kan ses på figur 3.7. Da det ikke muligt at aflæse de konkrete værdier for R_b og måle modstanden, men de kan aflæses i LTspice direkte og bliver målt til

$$I_{R_b} = 902.73 \mu\text{A} \quad \wedge \quad I_{\text{Måle modstand}} = 216.46 \text{ mA}$$

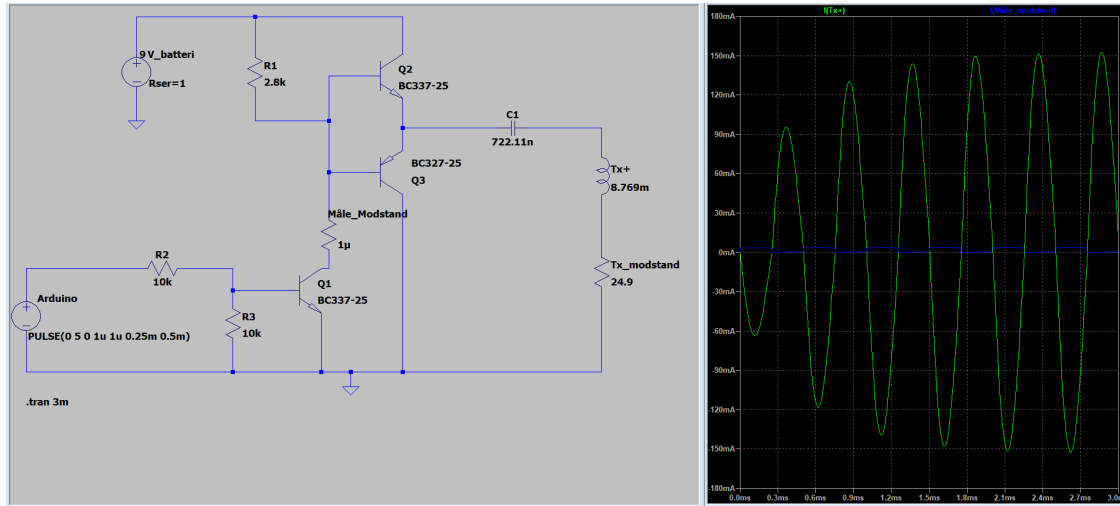
Forstærkning kan findes ved se hvor mange gange base strømmen går op i collectorstrømmen:

$$\beta_{\text{sim}} = \frac{I_{\text{Måle modstand}}}{I_{R_b}} = \frac{216.46 \text{ mA}}{902.73 \mu\text{A}} = 239.79$$



Figur 5: LTspice simulering af forstærkning med bipolar transistor

Simuleringen viser netop det kan være svært at bestemme forstærkningen. Det gælder således også når det kommer til AB-klasse forstærkningen. Da der i udleveret et P.A kredsløb som vil kunne fungere til metal detektoren, anvendes denne og via simuleringer vil base-modstandene blive justeret. Her kræver det en idé om hvad induktansen og modstanden er i sender-spolen. Værdierne fra **3.8.1 TX-spole**, kan der P.A kredsløbet opsættes og efter lidt justeringer ses det, at med en modstand på 2.8kΩ vil der opnås en strøm gennem spolen som svinger mellem 150mA og -150mA, se figur 3.7.



Figur 6: LTspice simulering af P.A kredsløb med udregnede værdier.

3.8 Spoler

3.8.1 TX-spole

Til designet af spolerne blev TX-spolen først udregnet. Ud fra udregningerne effektforstærkeren vidste vi at der ville være $V_L \approx 8.3V$ til rådighed for spolen og ønskede at trække $I_L \approx 150mA$ gennem spolen. Desuden var radius af de tre spoler fastlagt i designet af søgehovedet $r_{TX} = 13cm$. Derudover blev det besluttet at bruge en kobbertråd med en diameter på 0.3mm da en for tyk kobbertråd ville betyde flere viklinger på spolen og en for tynd kobbertråd risikerede at knække.

Da vi vidste at effektforstærkeren leverede et firkantsignal kunne den fundamentale frekvens Fourier ko-efficient findes ved

$$a_n = 2 \cdot \frac{A}{n \cdot \pi} \cdot \sin\left(\frac{n \cdot \pi}{2}\right) = 2 \cdot \frac{8.3V}{1 \cdot \pi} \cdot \sin\left(\frac{1 \cdot \pi}{2}\right) = 5.283V \quad (4)$$

Hvor A er amplituden af firkantsignalet, n er den fundamentale frekvens som er 1.

Nu kunne RMS spændingen findes ved at dividere a_n med kvadratroden af 2.

$$RMS = \frac{a_n}{\sqrt{2}} = \frac{5.283V}{\sqrt{2}} = 3.736V \quad (5)$$

Modstanden i spolen kunne nu findes ved.

$$R_L = \frac{RMS}{I_L} = \frac{3.736V}{150mA} = 24.904\Omega \approx 25\Omega \quad (6)$$

Da metaldetektoren designes som VLF og frekvensen derfor er forholdsvis lav kan det antages at strømmen i spolen er ligeligt fordelt i hele tværsnittet af kobbertråden og med radius af TX-spolen kan antallet af viklinger til findes.

$$N_{TX} = \frac{R_L \cdot r t^2}{2 \cdot r_{TX} \cdot \rho_c} = \frac{25\Omega \cdot \frac{0.3mm^2}{2}}{2 \cdot 13cm \cdot 1.68 \cdot 10^{-8} \cdot \Omega \cdot m} = 128.307 \quad (7)$$

Hvor rt er kobbertrådens radius, r_{TX} er spolens radius og ρ_c er kobbers resistivitet ($\rho_c = 1.68 \cdot 10^{-8} \cdot \Omega \cdot m$).

Dernæst blev selvinduktansen udregnet med.

$$L_{TX} = \frac{0.349 \cdot r_{TX}^2 \cdot N_{TX}^2}{9 \cdot r_{TX} + 10 \cdot A} = \frac{0.349 \cdot 13cm^2 \cdot 128.307^2}{9 \cdot 13cm + 10 \cdot 0.8cm} = 8769.536\mu H = 8.77mH \quad (8)$$

Hvor r_{TX} er radius af spolen(cm), N_{TX} er antallet af viklinger på spolen og A er højden af spolen(cm).

Og til sidst blev magnetfeltet for spolen udregnet.

$$|B| = \frac{\mu_0 \cdot I_L \cdot N_{TX}}{2 \cdot r_{TX}} = \frac{4 \cdot \pi \cdot 10^{-7} \frac{H}{m} \cdot 150mA \cdot 128.307}{2 \cdot 13cm} = 93.02 \cdot 10^{-6}T \quad (9)$$

Hvor $\mu_0 = 4 \cdot \pi \cdot 10^{-7} \frac{H}{m}$

3.8.2 Feedback-spole

Feedbackspolen bliver viklet uden om RX-spolen og skal have et modsatrettet magnetfelt i forhold til TX-spolen, for at skabe et "vakuum" så RX-spolen får mindst mulig interferens og dermed kan detektere reflekterede signaler fra metal objekter. Feedback spolen har en radius på $r_{feedback} \approx 6cm$. Grunden til at det er en cirka radius er at feedbackspolen bliver viklet udenpå RX-spolen adskilt af isoleringstape.

Ved at omskrive ligning (9) kan antallet af viklinger på feedbackspolen $N_{[feedback]}$ findes, da det ønskede magnetfelt for spolen skal være det samme som TX-spolen.

$$|B| = \frac{\mu_0 \cdot I_L \cdot N_{TX}}{2 \cdot r_{TX}} \Rightarrow N_{feedback} = \frac{2 \cdot |B| \cdot r_{feedback}}{\mu_0 \cdot I_L}$$

$$N_{feedback} = \frac{2 \cdot |B| \cdot r_{feedback}}{\mu_0 \cdot I_L} = \frac{2 \cdot 93.02 \cdot 10^{-6}T \cdot 6.005cm}{4 \cdot \pi \cdot 10^{-7} \cdot 150mA} = 59.268$$

Dette antal af viklinger ville dog kun være vejledende da vi med formlerne gik ud fra at magnetfelterne var homogene. I realiteten skulle feedback-spolen trimmes, dette bliver gennemgået under implementerings afsnittet.

3.8.3 RX-spole

For at få en god sensitivitet i RX-spolen skulle denne have en selvinduktans på mindst $10mH$, desuden var vores søgehoved designet således at RX-spolen ville få en radius på $6cm$. Det nødvendige antal viklinger til spolen kunne findes ved at benytte formel (8), her indsættes $L_{RX} = 10mH$ og Maples solve kommando bruges til at finde antallet af viklinger. Men da formlen giver selvinduktansen i μH krævede den lidt modifikation:

$$10^{-3}H = \frac{0.349 \cdot r_{RX}^2 \cdot N_{RX}^2}{9 \cdot r_{RX} + 10 \cdot A} \cdot 10^{-6}$$

Vha. Maples solve kommando kunne N_{RX} nu findes:

$$10 \cdot 10^{-3}H = \frac{0.349 \cdot r_{RX}^2 \cdot N_{RX}^2}{9 \cdot r_{RX} + 10 \cdot 1cm} \cdot 10^{-6}$$

$$\Downarrow$$

$$N_{RX} = 212.417$$

3.9 LC lavpas filter

Når at Tx-spolen udsender sit magnetiske felt, med en frekvens på 2kHz, vil Rx-spolen ikke nødvendigvis udelukkende modtage 2kHz reflekterede signaler, tværtimod. Det er derfor essentielt at tilføje et LC lavpas filter efter modtager spolen, da ellers det modtagne signal vil være ulæseligt. Herudover, er det også vigtigt, at sætte cut-off frekvensen større end 2kHz, da på grund af filteret er et LC-lavpas, vil der ved cut-off frekvensen være forstærkning af signalet, grundet et resonans-peak. Forstærkningen øges jo større modstanden R bliver, og da der er koblet en ikke-inverterende operationsforstærker på dette trin, vil modstanden blive uendeligt stor. Derfor vil forstærkningen ved cut-off frekvensen være markant og ydermere ligger faseskiftet også i dette punkt. Derfor blev det besluttet at sætte cut-off frekvensen til 4kHz da signaler med andre frekvenser bliver frasortet i samplingen på microcontrolleren og forstærkningen ved 2kHz ville, i følge simuleringer, blive under 3dB.

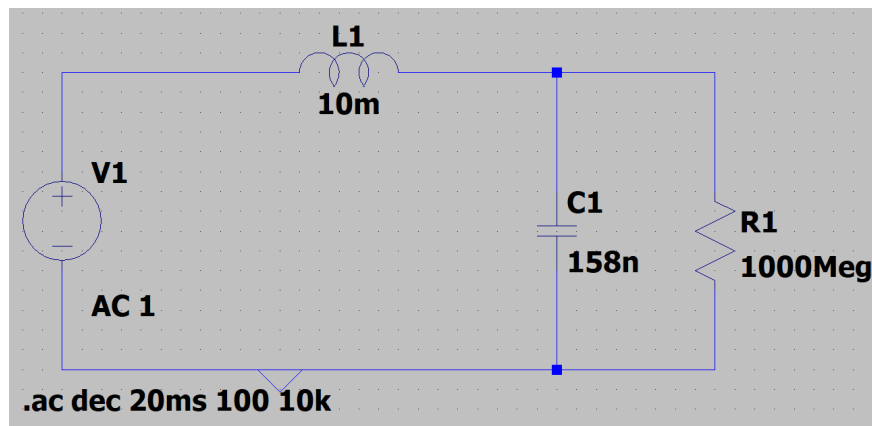
Da spolen i kredsløbet er RX-spolen, $L_{RX} = 10mH$ og cut-off var 4kHz, kan den nødvendige kondensator bestemmes ved følgende formel:

$$f_c = \frac{1}{2\pi\sqrt{LC}} \quad (10)$$

⇕

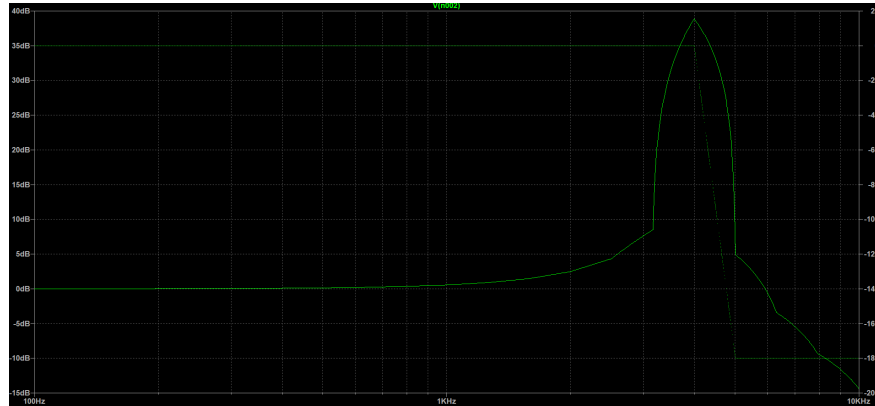
$$4\text{ kHz} = \frac{1}{2\pi\sqrt{10mH \cdot C}} \quad \Leftrightarrow \quad C = 158nF$$

Med dette kan der opsættes en LTspice model med de givne værdier. Denne simulering kan ses på figur 3.9.



Figur 7: LTspice model med udregnede værdier

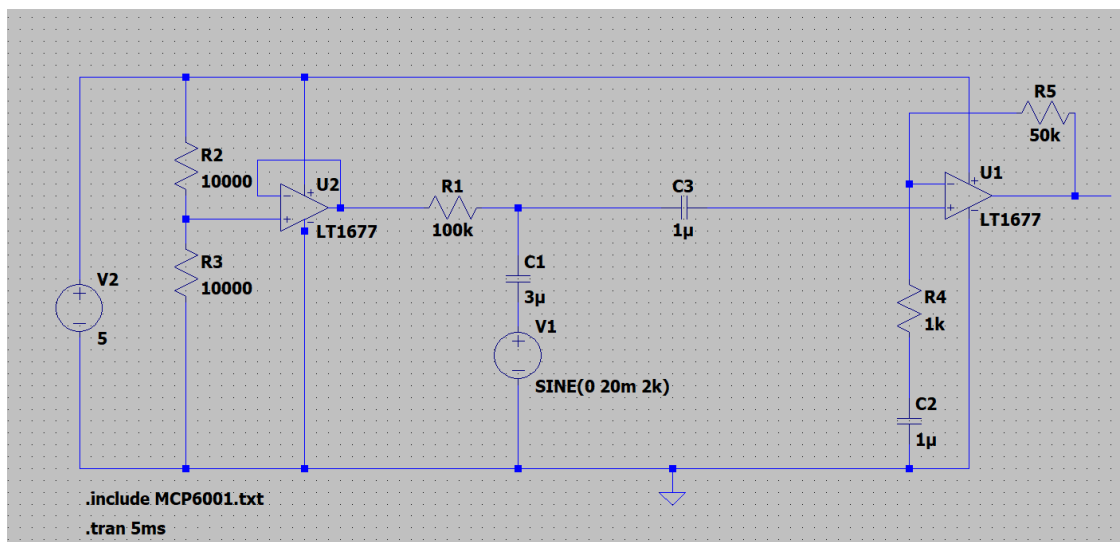
Det ses at på figur 3.9 at R1 (OP-amp indgangsmodstand) modstanden er på 1000MΩ, hvilket ikke nødvendigvis er den præcise modstand i virkeligheden, men det skal blot simulere en meget stor modstand, da det ikke er muligt at have en uendelig stor modstand i virkeligheden.



Figur 8: LTspice simulering af LC lavpas filter

3.10 Operationsforstærker

Da det signal RX-spolen opfanger er forholdsvis lavt ($20\text{mV} - 100\text{mV}$) er det nødvendigt at forstærke signalet for at micro controllerens ADC kan registrere signalet. Derudover skal signalet have et offset på 2.5V da micro controllerens ADC kun kan modtage fra $0\text{V} - 5\text{V}$.



Figur 9: LTspice model af operationsforstærker kredsløb

På figur 3.10 ses det at en 5V spændingskilde driver de to operationsforstærkere (U1 og U2). En spændingsdeler der halvere $V2$ går ind på U2s plus indgang, U2 der er en spændingsfølger, fungerer som en buffer således at signalet $V1$ (RX-spolen) ikke bliver påvirket af $V2$. Kondensatorene $C1$ er tiltænkt som DC-spær derudover er $C2$ og $C3$ placeret da hjælpelærer foreslog at dette ville hjælpe med at fjerne støj på signalet. U1 er koblet som en ikke-inverterende grundkobling og får på plus indgangen signalet fra RX-spolen med et offset på 2.5V og forstærkningen er på V_{out} er 50, dette kan tilpasses så RX-signalet kan få den ønskede forstærkning med ligningen:

$$A_v = 1 + \frac{R5}{R4}$$

Men dette kan først testes når de forskellige dele er blevet konstrueret.

4 Implementering & test

4.1 Opstart

Når man tænder for metaldetektoren viser den en lille animation med lidt lyd til via en piezo/buzzer. Herefter bruger den et kort øjeblik på at nulstille værdierne i displayet inden den er klar til brug.

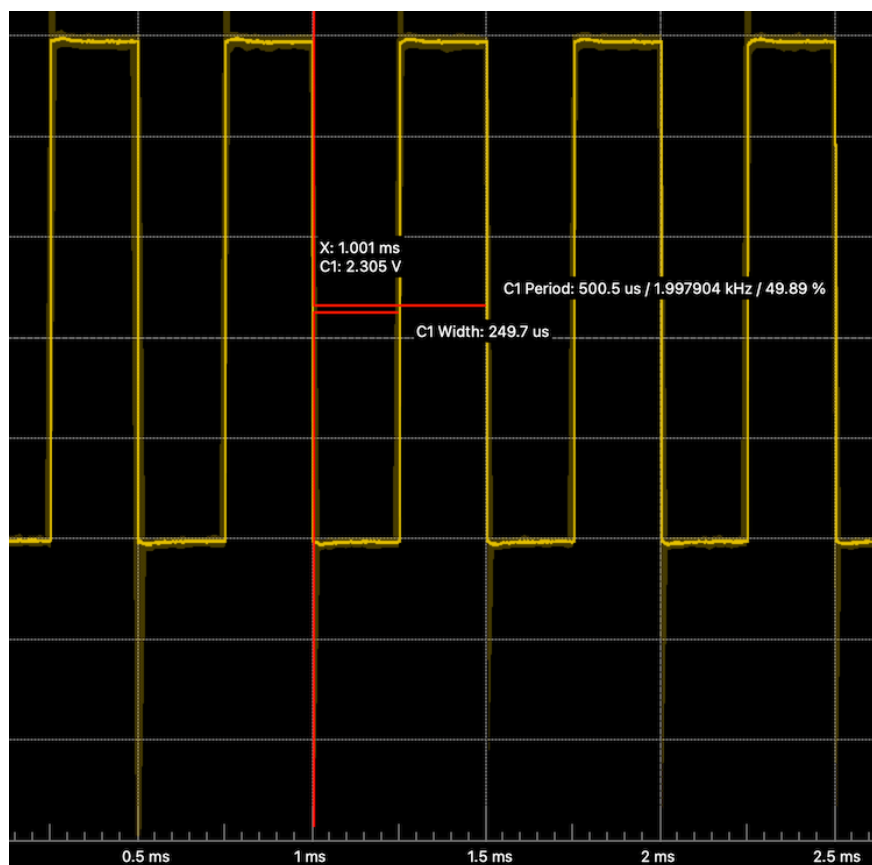
4.2 Tilstandsmaskine

Kernen i softwaren er en tilstandsmaskine der sørger for at holde sampling og beregning adskilt. Den starter med at tænde for ADC'en og sample, for herefter at slukke ADC'en, beregne og sende resultatet til OLED displayet. Dette bliver den ved med indtil metaldetektoren slukkes.

4.3 Detektions signal

Detektionssignalet er et 2 kHz firkant signal der sendes ud fra Arduinoen igennem effekt forstærkeren, ud i spolerne i søgehovedet og tilbage gennem operationsforstærkeren og ind på ADC'en. Detektionssignalet genereres i Arduinoen ved at tænde og slukke PORT D pin 7. En variabel kaldet *timer0count* tæller én op hver gang der genereres et ADC interrupt hvilket sker 8000 gange i sekundet. I selve interruptet bliver pin 7 sat høj når *timer0count* er 0 og lav når *timer0count* er 2, dette generer et 2 kHz firkant signal. Til sidst når *timer0count* rammer 4 nulstilles den.

Frekvensen på detektionssignalet er målt ved at sætte et oscilloskop på PORT D pin 7, afbilledet herunder.

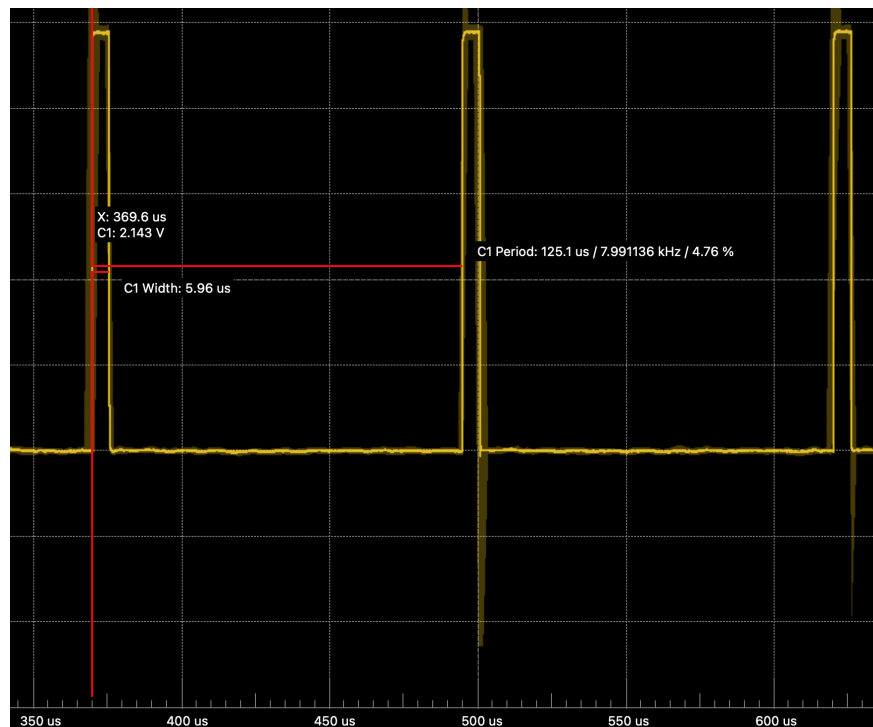


Figur 10: 2 kHz detektionssignal

Som det ses er frekvensen målt til 1.998 kHz hvilket er vel inden for krav nr. 12 om en detektionsfrekvens på $2 \text{ kHz} \pm 100 \text{ Hz}$.

4.4 Analog til digital konvertering

Detektionssignalet kommer retur til Arduinoen og bliver samlet i ADC'en med 8 kHz. ADC'en er sat op til at være interrupt baseret som ønsket i krav nr. 7 således at den tager en ny sample hver gang der kommer et interrupt fra timer 0. Timer 0 er en 8 bit timer (tæller fra 0-255) som er prescalet med 8. Det vil sige at den tæller med en hastighed svarende til $16e6/8 = 2e6$ gange i sekundet. Hver gang den rammer sammenligningsværdien OCR0A giver den et interrupt der gør at ADC'en tager en sample. Når ADC'en har samlet, gemmes hver sample på en plads i et array kaldet adcSampleArr. Dette gøres 48 gange hvorefter tilstandsmaskinen stopper samplingen, og sender adcSampleArr til beregning i funktionen calcResult.



Figur 11: 8 kHz sampling

Som det ses på billedet samples der med 7.99 kHz hvilket er inden for krav nr. 11 om en samplingfrekvens på $8 \text{ kHz} \pm 100 \text{ Hz}$.

4.5 Digital signalbehandling

I funktionen calcResult foregår den digitale signalbehandling ved først at multiplicere de respektive samples i adcSampleArr med værdierne fra de to arrays cos1 og sin1. cos1 indeholder værdierne 1, 0, -1, 0 og bruges til at finde den reelle del af signalet Xre. sin1 indeholder værdierne 0, -1, 0, 1 og bruges til at finde den imaginære del af signalet Xim. Herefter normaliseres Xre og Xim ved at multiplicere med $\frac{5}{1024}$ og det hele udføres for samtlige 48 samples.

Realdelen Xre og imaginærdelen Xim af signalet summeres hver for sig fra 0 til 47 og gemmes i henholdsvis ampSum og phaseSum inden modulus og argumentet udregnes som beskrevet under design hvorefter det bliver udskrevet til OLED displayet.

Oprindeligt var det tænkt at udføre filtrering af signalet inden udskrivning som beskrevet i design sektionen. Men efter implementering har det vist sig at være unødvendigt da signalet i forvejen er ret stabilt samt opbygningen af programmet med en tilstandsmaskine gør at en periode fra sampling start til sampling start tager rundt regnet 135 ms. Derfor opdateres OLED displayet kun omkring 7.5 gange i sekundet hvilket er nok til at tallene ikke er ulæselige. Filteret har været implementeret, men er blevet fjernet igen da det dæmpede resultaterne unødigt i forhold til at det ikke er nødvendigt med et filter.

4.6 OLED display

Inden amplituden og fasen kan udskrives til displayet bliver de konverteret ved hjælp af funktionen `dtostrf` (Decimal to string function) som tager en float og konverterer den til et character array som funktionen `sendStrXY` kan håndtere. `sendStrXY` er en funktion der ligger i header filen `ssd1306.h` og den bruges til at udskrive et character array til OLED displayet. I den forbindelse skal det nævnes at filerne ⁶ der bruges til håndtering af OLED displayet er udarbejdet og udleveret af Ole Schultz - `osch@dtu.dk`.

4.7 Start/stop knap

En start/stop eller snarere en tænd/afbryd knap er blevet implementeret således at den afbryder forbindelsen til batteriets plus pol.

4.8 Nulstilling

Allerede når man tænder for metaldetektoren nulstilles værdierne i displayet således at metaldetektoren er klar til brug. Derudover har systemet også en reset knap der er forbundet til PORT B bit 2 således at når knappen aktiveres sættes bit 2 høj og programmet trækker herefter de aktuelle værdier fra de kommende værdier indtil man eventuelt aktiverer resetknappen på ny.

4.9 Buzzer

Der er blevet implementeret en buzzer som er sat til at give lyd når en given grænse for henholdsvis amplitude og fase bliver overskredet, således at man ikke behøver at holde øje med displayet kontinuerligt. Buzzeren er koblet til PORT D bit 3 som er sat op til 8 bit timer 2 fast PWM. Frekvensen styres af OCR2A sammenligningsværdien således at man kan variere lyden. Lyden kan slås til og fra ved at holde reset knappen inde i et par sekunder.

4.10 Detektionsfølsomhed

Detektionsfølsomheden er blevet testet med fire forskellige slags metal: jern, messing, kobber og aluminium ved 5 cm afstand fra søgehovedet med følgende resultater:

Metal:	Amplitude:	Fase:
Jern ø13 x 70mm	10	-4
Messing	5	-74
Kobber	5	-147
Aluminium	13	-129

OBS. det benyttede jern er rundt regnet 5% større end det i kravspecifikationen angivende, men da den benyttede jerngenstand ikke er massiv antages det at det går nogenlunde lige op.

Ud fra ovenstående målinger kan man konkludere at metaldetektoren kan detektere forskel på jern og mere ædle metaller da fasen ændrer sig væsentligt mere ved ædelmetallerne kobber og aluminium end ved jern og endda også messing som er en legering mellem kobber og zink.

⁶"I2C.c", "I2C.h", "ssd1306.c", "ssd1306.h", "data.h"

4.11 Spoler

Da søgehovedet var blevet 3D-printet kunne spolerne konstrueres og spolerne selvinduktans og modstand kunne efterprøves med et LC meter. Alle tre spoler blev viklet med kobbertråd med en diameter på $0.3mm$. Søgehovedets konstruktion gjorde at rillerne, til spolerne, buede svagt hvorfor det var en udfordring at ligge viklingerne pænt ved siden af hinanden. Ydermere fandt vi ud af at inderkonstruktionen til RX-spole og feedbackspolen var besværlig at vikle, det havde været mere ideelt at have de to dele adskilt og derefter samle dem.

Under design afsnittet blev TX-spolens antal viklinger og selvinduktans udregnet til:

$$N_{TX} = 128.307$$

$$L_{TX} = 8.77mH$$

I bilag A.3 kan selvinduktansen for TX-spolen og feedbackspolen ses til at være $9.5mH$ målingen blev først foretaget efter de to spoler var blevet loddet sammen. Desuden ses modstanden også 28.2Ω , under design af P.A. havde vi forventet en modstand på 25Ω .

Feedbackspolen blev viklet uden på RX-spolen, de to spoler blev adskilt af isoleringstape. Feedbackspolen skulle modvirke det magnetfelt TX-spolen udsendte hvorfor feedbackspolen blev viklet i den modsatte retning i forhold til TX-spolen. Under design afsnittet blev feedbackspolens antal viklinger udregnet til:

$$N_{Feedback} = 59.268$$

RX-spolen blev i designafsnittet fastsat til $10mH$ da det var minimumskravet for at vi fik en god sensitivitet på metaldetektoren. Antallet af viklinger blev udregnet:

$$N_{Feedback} = 212.417$$

Her var det en udfordring at spolen da det som beskrevet i starten af afsnittet, var svært at vikle tråden da søgehovedets konstruktion blokerede for et lige træk på tråden.

I bilag A.3 kan selvinduktansen og modstanden for RX-spolen ses til at være $9.90mH$ og 20.6Ω . Vi havde forventet af selvinduktansen skulle være $10mH$, men dette er desværre ikke tilfældet. Det har dog vist sig at metaldetektoren kan leve op til samtlige krav bortset fra denne måling af selvinduktansen. Der kan selvfølgelig også være en afvigelse i måleren, men det har ikke været muligt at finde nogle valide data for præcisionen af måleren.

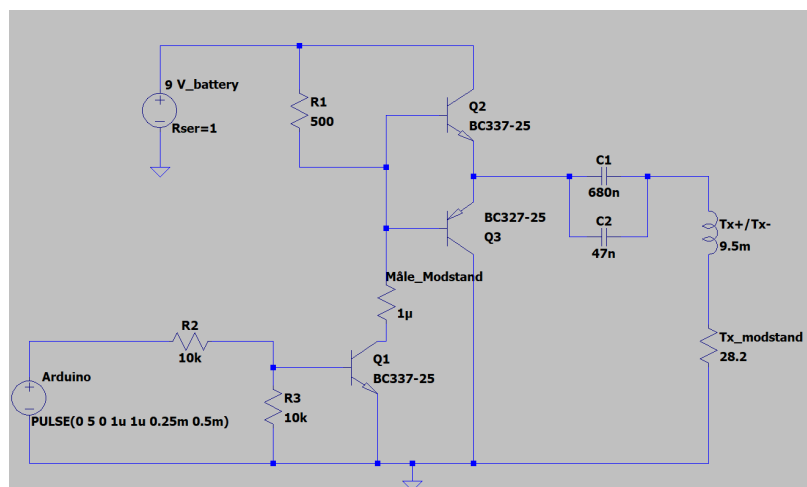
Da alle tre spoler var blevet konstrueret blev TX-spolen og feedbackspolen loddet sammen og vha. et oscilloskop der var koblet til RX-spolen kunne feedbackspolen trimmes. Rent praktisk foregik trimningen ved at tage viklinger af feedbackspolen indtil den målte sinuskurve var så lille som muligt. Her kunne der tydeligt ses en forskel efter blot ti viklinger var blevet taget af, da sinuskurven var på sit laveste havde den en peak-peak spænding på cirka $20mV$ og der var blevet viklet cirka 30 viklinger af feedbackspolen. Efterhånden som projektet skred frem var det dog nødvendigt at trimme den løbende når vi for eksempel flyttede P.A, LC filter og operationsforstærker kredsløbene fra breadboard til hulprint.

4.12 Effekt forstærker

Under designet af P.A'en blev der udelukkende kigget Tx^+ (sender spolen), uden der blev taget hensyn til Tx^- (feedback-spolen). Derfor, efter de målte værdier for selvinduktansen for Rx-spolen samt Tx-spolen (sender + feedback samlet) skulle de valgte kondensatorer ændres. Ydermere, viste det sig at der slet ikke blev trukket den ønskede strøm igennem P.A kredsløbet. Med en base-modstand på $1k\Omega$ var der et strømforbrug på $50mA$, når P.A kredsløbet var sat op på et breadboard. Til vores overraskelse, var der et endnu mindre strømforbrug efter kredsløbet blev opsat og loddet på et hulprint, hvor der blev trukket mellem $20-30mA$. Derfor blev det nødvendigt at skifte base-modstanden ned til 500Ω på hulprintet.

Yderligere, fordi spole-værdier varierede en smule fra det tidligere designede, skulle der bruges en kondensator på $722nF$. Da denne størrelse ikke findes som produktion, blev der anvendt en $680nF$ med en $47nF$

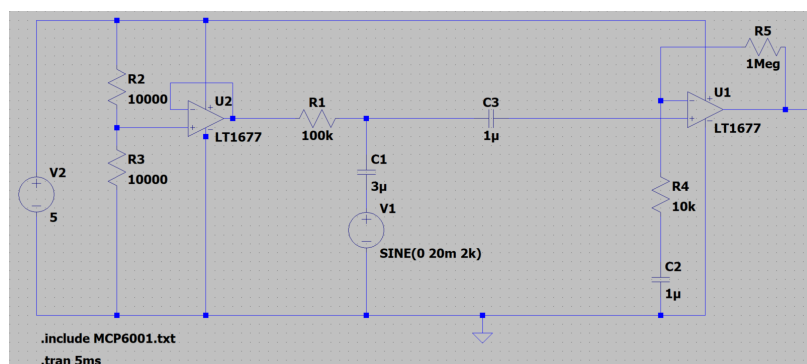
kondensator i parallel forbindelse for at så opnå en kapacitans værdi på 727nF. Med dette vil det nye P.A kredsløb se ud som vist på figur 4.12.



Figur 12: LTspice model af P.A kredsløb med målte værdier

4.13 Operationsforstærker

Til opbygningen af operationsforstærker kredsløbet blev MCP6002[6] brugt. Operationsforstærkeren blev først opbygget på et bread board, da det opstillede kredsløb fungerede som det ønsket blev det lavet på et hulprint sammen med P.A og LC filter, dog var offset generelt lidt lavere end det halve af forsyningsspændingen, men stadig brugbart da det var indenfor 0.1 – 0.2V.



Figur 13: LTspice model af operationsforstærker kredsløb med 100 gange forstærkning

Da søgehovedet blev koblet sammen med P.A, filter og operationsforstærkeren viste det sig at den optimale forstærkning var 100 gange, hvorfor modstanden R5 i den ikke-inverterende grundkobling blev justeret som det ses på 4.13. Oprindeligt havde vi kun brug for 15 ganges forstærkning af RX-spolens signal, hvilket vi fandt ud af var fordi LC filtret havde en cut-off frekvens på 2kHz, som gav en forstærkning i sig selv. Da dette blev ændret var det brug for en forstærkning på 100 gange, se evt bilag A.4 hvor forstærkningen er målt i simuleringen:

$$A_v = \frac{4.034V}{39.93mV} \approx 101$$

Ved fysiske målinger på kredsløbet blev det brugt en signalgenerator med en sinuskurve med en amplitude på 40mV peak-peak og et oscilloskop der målte det ikke forstærkede og forstærkede signal, på bilag A.4 kan

værdierne ses til:

$$A_v = \frac{3.889V}{39.91mV} \approx 97$$

Dermed er den ønskede forstærkning opnået.

4.14 LC lavpas filter

Tilføjelsen af lavpas filteret gjorde en markant ændring på det modtagne reflekterede signal, hvor uden dette filter vil støjniveauet være alt for stort til at kunne arbejde med signalet. Der blev testet med både en cut-off frekvens på henholdsvis 2kHz og 4kHz. Det viste sig, at med et filter med 2kHz cut-off frekvens blev signalet vist som en næsten ideel sinuskurve, hvilket ikke er overraskende, da den vil fjerne meget støj og blive forstærket grundet resonans-peaket. Dog ved anvendelse i metaldetektoren, var både amplituden og faseren en smule ustabil. Herunder var faseren specielt påvirket.

Ved brug af 4kHz lavpas filteret, blev det modtagne signal ikke ligeså pæn en sinuskurve, men snarere en superposition af sinuskurver med forskellige frekvenser. Dette var forventet da det kun er frekvenser større end 4kHz, der gradvist ville blive filteret fra. Men ved brug af dette filter var faseren langt mere stabilt. Amplituden var ikke længere ligeså kraftigt og derfor var vi nødsaget til at hæve forstærkning på operationsforstærker kredsløbet med næsten 100 gange. Grunden til at amplituden ikke længere var ligeså kraftigt, er pga. resonans-peaket (forklaret i **3.9 LC lavpas filter**) og som vist på figur 3.9 er der i dette peak næsten en forstærkning på $40dB \approx 100 \times U$, hvor U er spændning.

4.15 Video demonstration

Da vi har konstrueret en fuldt funktionsdygtig metaldetektor og ikke har haft mulighed for at vise den frem, har vi lavet en lille demonstrationsvideo og lagt på youtube. Videoen kan ses af alle der har linket til den. <https://youtu.be/FCOfsdaurbA>



Figur 14: QR kode med link til youtube demonstrations video

Se også produkt fotos i appendix A.1 og A.1

5 Konklusion

Udarbejdelsen af projektet har været lærerigt og udfordrende, ikke mindst COVID-19 situationen der har begrænset tid i laboratorier og kvaliteten af udstyret i disse. Gruppen har brugt egne digitale oscilloskoper, strømforsyninger og signalgeneratorer, da dem i laboratorierne ikke fungerede optimalt. Dog er det lykkedes over al forventning at få lavet en metaldetektor der lever op til alle krav bortset fra RX-spolens selvinduktans som er $0.1mH$ fra minimumskravet. Dette er rigtig ærgerligt og det blev overvejet at vikle en ny RX-spole, men eftersom at metaldetektoren kunne detektere en metal genstand som specificeret i kravene blev det besluttet at det kunne accepteres, siden vi havde beregnet en selvinduktans der levede op til minimumskravet. For LC filtret viste det sig at en cut-off frekvens på 2kHz gav en væsentligt flottere sinuskurve på RX-spolen, men det gjorde fasen meget ustabil, dette blev ændret til en cut-off frekvens på 4kHz hvilket resulterede i en mere skarp (trekantet) sinuskurve, men med en fase det var langt mere stabil. Vi har også haft problemer med at kurverne "hænger", dette viste sig at skyldes kablerne til søgehovedet og kablerne til strømforsyningen. Dette kunne tildels afhjælpes ved at vikle kablerne om hinanden, men en bedre løsning havde måske været at bruge COAX kabler fra søgehovedet til kredsløbene. Da vi skulle teste hele vores setup med micro controller, kredsløb og søgehoved kørte alle delene stort set ved første forsøg, dette vidner om at vi hver især har formået at teste og afprøve vores individuelle bidrag til projektet inden det blev koblet sammen. Testen har vist at vi bruger væsentligt mindre strøm end vi havde forventet, dette har vi dog ikke formået at finde en forklaring på, men med en metaldetektor der kan leve op til kravene ser vi det blot som positivt at arbejdstiden er væsentligt længere end forventet. Tiden i laboratorierne har af gode grunde været begrænset, hvilket har givet lidt frustration når fejlfinding måtte afbrydes og skubbes til næste dag, men det har også givet tid til at gennemtænke problematikkerne inden man stirrede sig blind i laboratorierne.

6 Bidrag

6.1 Bidrag

Emne:	Navn:
ADC	Jørgen
DSP	Mads, Kim, Jørgen
2 kHz signal	Jørgen
OLED	Jørgen
Statemachine	Jørgen
Effektforstærker	Mads
Coils	Kim & Mads
Operationsforstærker	Kim
LC lavpas filter	Kim & Mads

7 References

- [1] Coil Basics - af Carl Moreland
<https://learn.inside.dtu.dk/d2l/le/content/57854/viewContent/190503/View>
- [2] Advanced Instrumentation for Polyharmonic Metal Detectors
<https://learn.inside.dtu.dk/d2l/le/content/57854/viewContent/190505/View>
- [3] Duracell Ultra Power 9V battery datasheet
<http://www.farnell.com/datasheets/1842389.pdf>
- [4] Amplifier Transistor BC337 (NPN) datasheet
<https://pdf1.alldatasheet.com/datasheet-pdf/view/2888/MOTOROLA/BC337-25.html>
- [5] Amplifier Transistor BC327 (PNP) datasheet
<https://pdf1.alldatasheet.com/datasheet-pdf/view/2885/MOTOROLA/BC327-25.html>
- [6] OP AMP MCP6002 datasheet
<https://html.alldatasheet.com/html-pdf/195202/MICROCHIP/MCP6002/486/1/MCP6002.html>

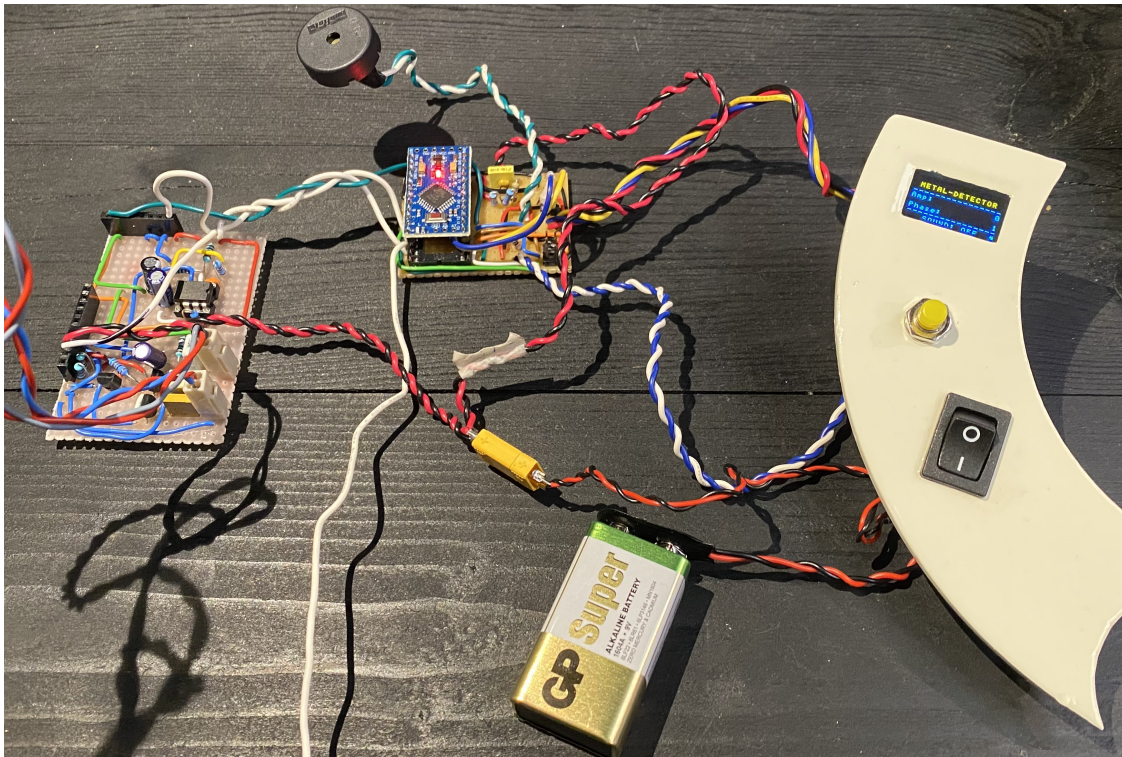
A Appendix

A.1 Produkt fotos

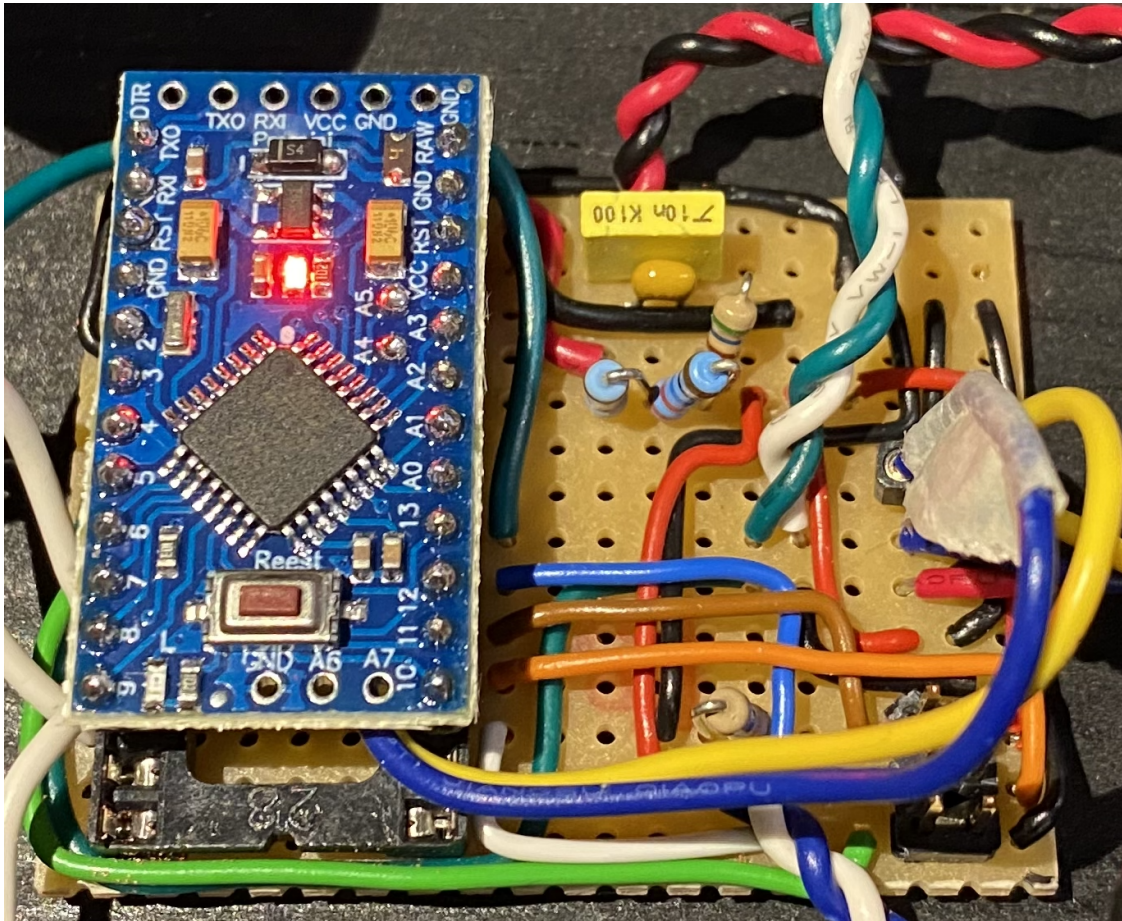




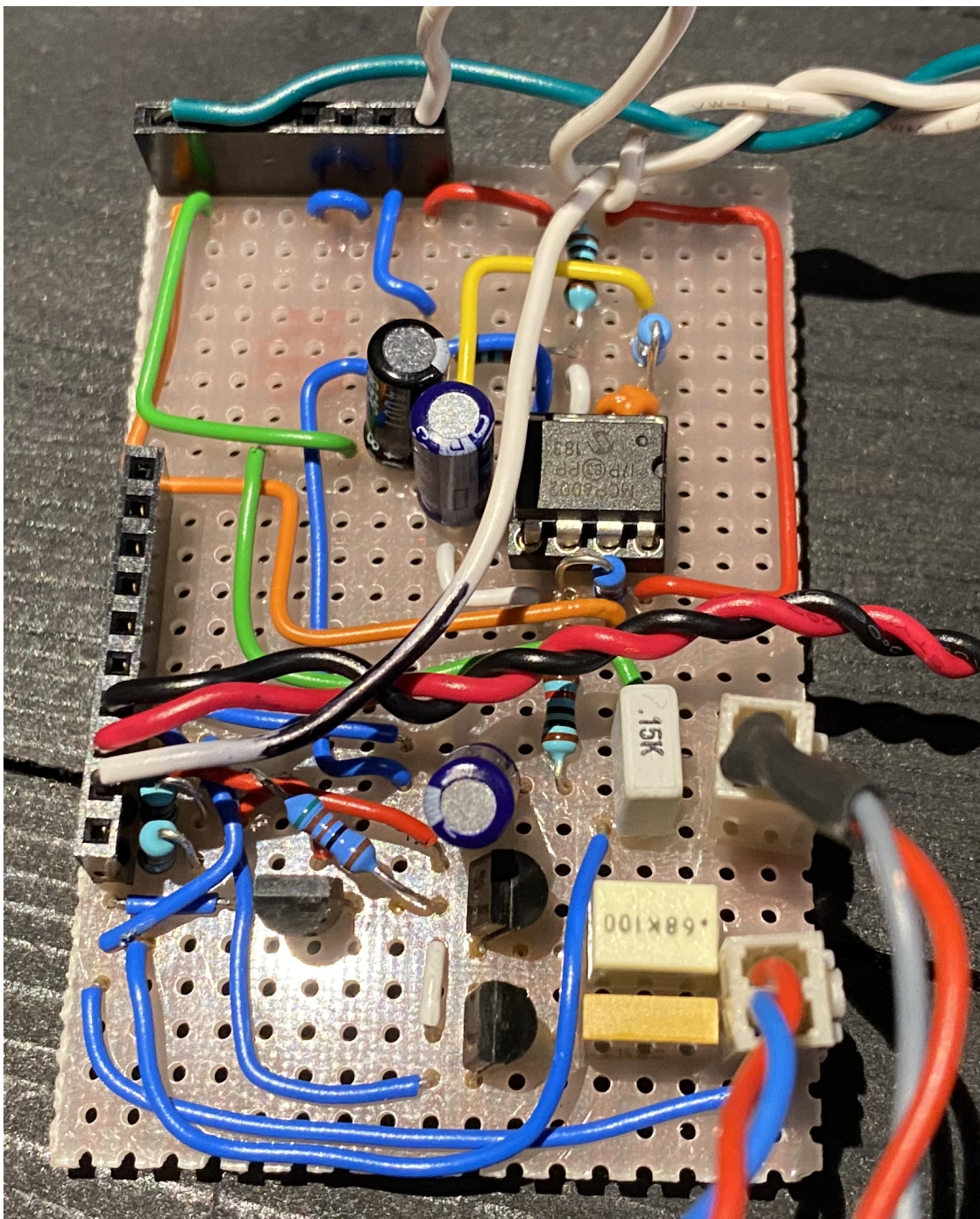
A.2 System



Figur 15: Det samlede system på nær søgehovedet

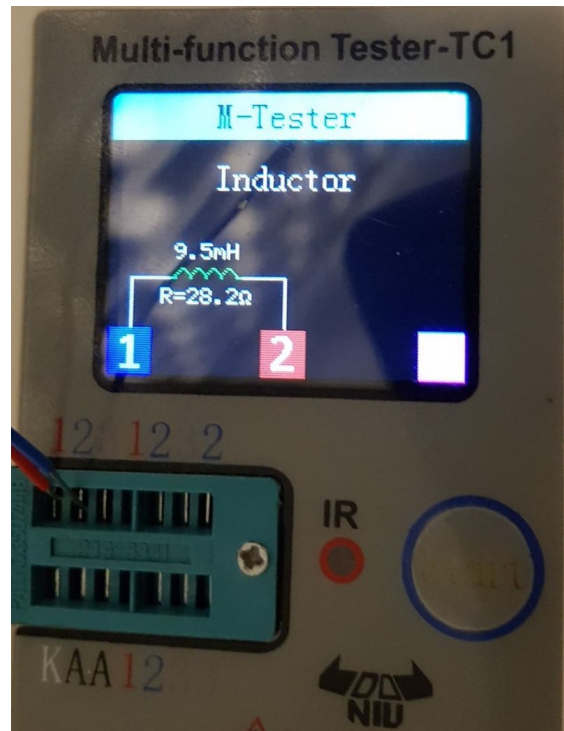


Figur 16: Den digitale del af systemet



Figur 17: Den analoge del af systemet

A.3 Søgehoved

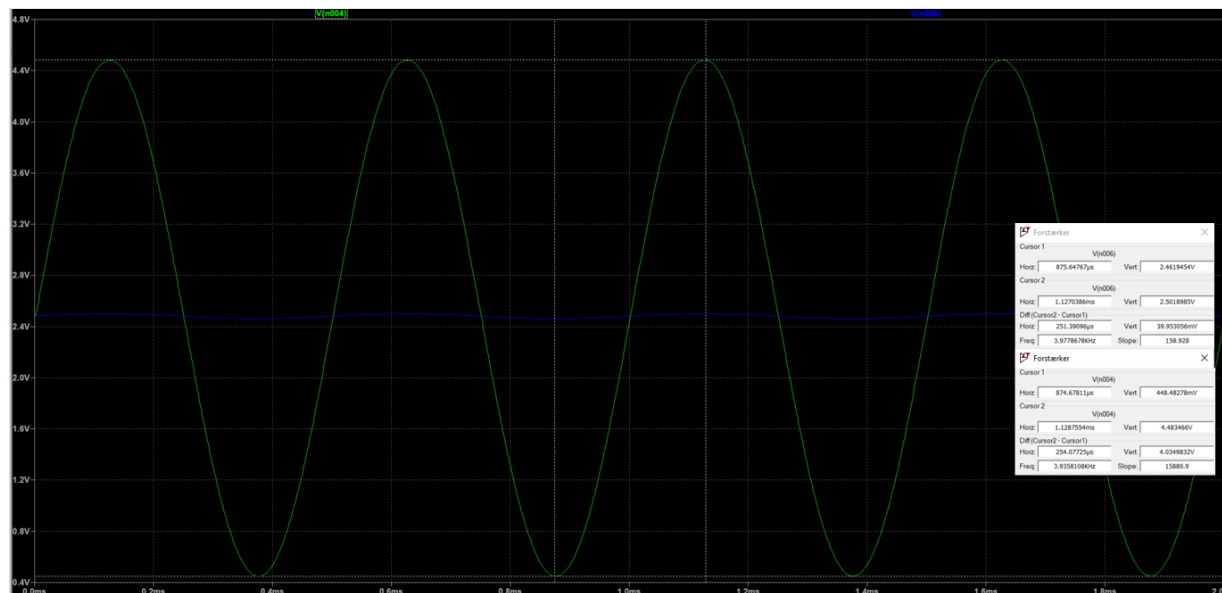


Figur 18: Målt selvinduktans og modstand for TX-spole og feedbackspole

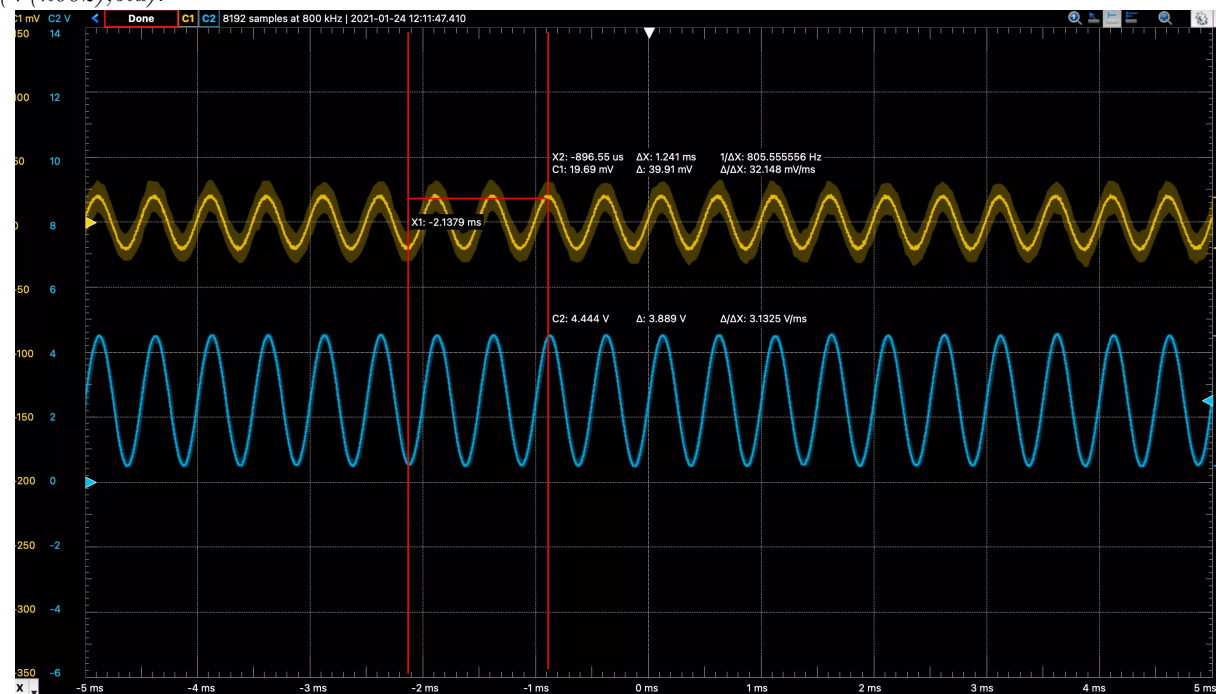


Figur 19: Målt selvinduktans og modstand for RX-spole

A.4 Operationsforstærker



Figur 20: Simulation med værdier for det forstærkede signal (V(n004), grøn) og det ikke forstærkede signal (V(n002), blå).



Figur 21: Målt forstærkning med signalgenerator og oscilloskop, det forstærkede signal i blå og det ikke forstærkede signal i gul

A.5 C code


```

...1 Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\main.c 1
1  /*****
2  * DTU - ELECTRO          62739 - METAL DETECTOR          *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519          *
5  * Creation date:         24.01.2021                          *
6  * Update date:          24.01.2021                          *
7  * Version:               1.0                                  *
8  *****/
9  #include <avr/io.h>
10 #include <stdio.h>
11 #include <avr/interrupt.h>          // Includes interrupt functionality
12 #include <stdlib.h>                // Includes the dtostrf for OLED
13 #include "adc.h"
14 #include "calcResult.h"
15 #include "timerCounter.h"
16
17 #define N 48                      // Number of samplings before result calculation
18
19 /***** GLOBAL VARIABLES *****/
20 volatile int timer0count = 0;      // Scaling of timer 0 for signal output
21 volatile int i = 0;               // Counting variable used in ISR(ADC_vect)
22 volatile int state = 1;           // Variable used in the switch
23 volatile double adcSampleArr[N] = {0}; // Array with samples from ADC
24 volatile int btn = 0;             // Storing pin state (for zeroing OLED button)
25 volatile int btnPress = 0;        // Stores the zeroing OLED button press
26
27
28
29 int main(void)
30 {
31     /***** INITIALIZATION *****/
32     counter0_init();               // Initialize timer for ADC sampling
33     adc0_init();                  // Initialize ADC
34     initOLED();                   // Initialize OLED
35     runOnce();                    // Load startup screen on OLED
36
37     /***** OUTPUT PINS *****/
38     DDRB |= (1<<DDB1);            // Physical pin 8 (OpAmp)
39     DDRD |= (1<<DDD7);            // Physical pin 7 (Signal out)
40     DDRB |= (1<<DDB5);            // Physical pin 13 (LED FOR TEST)
41     DDRD |= (1<<DDD6);            // Physical pin 6 (FOR OSCILOSCOPE TEST)
42
43     /***** INPUT PINS *****/
44     DDRB |= (0<<DDB2);            // Physical pin 10 (for zeroing OLED values)
45
46     /***** INTERRUPT *****/
47     sei();                        // Enable global interrupt
48
49
50     /***** ENABLE AMPLIFIER *****/
51     PORTB |= (1<<1);              // Enable OpAmp
52
53
54     while (1)
55     {
56         /***** RESET BTN PRESS *****/
57         if ((PINB & 0b00000100) >= 1) // Store button press from PB2
58         {                               // (Physical pin 10)
59             btnPress = 1;
60         }
61
62         /***** STATE MACHINE *****/
63         switch(state)
64         {

```


...1 Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\main.c

2

```

65         case 0:
66             break;
67
68         /* Enable ADC and start sampling */
69         case 1:
70             counter0_start();           // Start ADC sample counter
71             adcON();                   // Enable ADC
72             state = 0;
73             break;
74
75         /* Disable ADC and stop sampling */
76         case 2:
77             adcOFF();                  // Disable ADC
78             counter0_stop();           // Stop ADC sample counter
79             calcResult(adcSampleArr, N, btnPress); // Calculate result and send to OLED
80             state = 1;
81             break;
82
83         default:
84             state = 1;
85             break;
86     }
87 }
88 }
89
90
91 /***** ADC INTERRUPT SAMPLE READY *****/
92 ISR(ADC_vect)
93 {
94     adcSampleArr[i] = adc0_getSample(); // Get sample from ADC and store in adcSampleArr
95     i++;
96
97     if (i == N) // Step state machine when buffer full
98     {
99         state = 2;
100        i = 0;
101    }
102 }
103
104
105 /***** ADC INTERRUPT BEGIN SAMPLE *****/
106 ISR(TIMER0_COMPA_vect)
107 {
108     if (timer0count == 0) // Scale the 8 kHz timer for 2 kHz signal output
109     {
110         PORTD |= (1<<7);
111     }
112
113     if (timer0count == 2)
114     {
115         PORTD &=~ (1<<7);
116     }
117
118     timer0count++;
119
120     if (timer0count == 4)
121     {
122         timer0count = 0;
123     }
124 }

```

```

...io\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\calcResult.h 1
1  /*****
2  * DTU - ELECTRO          62739 - METAL DETECTOR          *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519          *
5  * Creation date:         24.01.2021          *
6  * Update date:          24.01.2021          *
7  * Version:               1.0          *
8  *****/
9  #include <math.h>
10 #include <string.h>
11 #include "OLED_init.h"
12 #include "startUp.h"
13 #include "piezoBuzzer.h"
14
15
16 /***** GLOBAL VARIABLES *****/
17 volatile int subtractValAmp = 0;          // Subtract this from result for zeroing OLED
18 volatile int subtractValPhase = 0;        // Subtract this from result for zeroing OLED
19 volatile int initOutput = 1;              // Pre-zero OLED values on startup
20 volatile char flip1 = 0;                  // Flip buzzer between two sounds
21 volatile char flip2 = 1;                  // Flip buzzer between two sounds
22 volatile unsigned int count1 = 0;          // OLED zeroing initialization delay
23 volatile unsigned int count2 = 0;          // Buzzer button delay
24 volatile char a = 0;                      // Buzzer sound ON/OFF switch variable
25 volatile char c = 0;                      // Buzzer sound ON/OFF switch variable
26 volatile char state2 = 0;                 // Buzzer sound ON/OFF switch variable
27
28
29 /***** INITIALIZE OLED AND BUZZER *****/
30 void initOLED()
31 {
32     OLEDinit();                          // Initialize OLED
33     piezoBuzzer_init();                  // Initialize piezo/buzzer
34 }
35
36
37 /***** DSP AND PRINT TO OLED *****/
38 void calcResult(volatile double adcSampleArr[], int N, int btnPress)
39 {
40     /***** VARIABLES *****/
41     char outputAmp[5] = {0};              // OLED character array
42     char outputPhase[5] = {0};             // OLED character array
43     unsigned char buzz1 = 0;               // Buzzer variable for phase value
44     unsigned char buzz2 = 0;               // Buzzer variable for amplitude value
45     unsigned char buzzSwitch = 0;          // Buzzer switch between sounds variable
46     int cos1[4] = {1,0,-1,0};              // 2 kHz real part factor
47     int sin1[4] = {0,-1,0,1};              // 2 kHz imaginary part factor
48     int i = 0, j = 0, k = 0;               // Counting variables
49     double Xim[N], Xre[N];                 // Imaginary part and Real part of the signal
50     double pi = 3.1415926540;              // Pi
51     double ampSum = 0;                     // Used to sum up the real part
52     double phaseSum = 0;                   // Used to sum up the imaginary part
53     double smp11 = 0;                      // Temporary variable for conversion to OLED
54     double smp12 = 0;                      // Temporary variable for conversion to OLED
55     double amplitude = 0;                  // Temporary variable for reset of OLED values
56     double phase = 0;                      // Temporary variable for reset of OLED values
57     double newAmplitude = 0;               // Temporary variable for amplitude
58     double newPhase = 0;                   // Temporary variable for phase
59
60
61     /***** RESET BTN PRESS *****/
62     btnPress = 0;
63     if ((PINB & 0b00000100) >= 1)          // Store zeroing OLED values button press from PB2
64     {                                         // (Physical pin 10)

```

```

...io\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\calcResult.h 2
65     btnPress = 1;
66 }
67
68
69 /***** DFT *****/
70 for (i=0; i<N; i++)
71 {
72     Xre[i] = adcSampleArr[i]*cos1[j]; // Calculate the real part of 2 kHz signal
73     Xre[i] = Xre[i]*5/1024;          // Normalize the real part
74
75     Xim[i] = adcSampleArr[i]*sin1[j]; // Calculate the imaginary part of 2 kHz signal
76     Xim[i] = Xim[i]*5/1024;          // Normalize the imaginary part
77
78     if (j == 3)
79     {
80         j = 0;
81     }
82     else
83     {
84         j++;
85     }
86 }
87
88 for (k=0; k<N; k++) // Sum up the real and imaginary parts
89 {
90     ampSum = ampSum + Xre[k];
91     phaseSum = phaseSum + Xim[k];
92 }
93
94
95 /***** AMP & PHASE CALCULATION *****/
96 amplitude = sqrt( (ampSum * ampSum) + (phaseSum * phaseSum) );
97 phase = atan2( phaseSum, ampSum ) * 180/pi;
98
99
100 /***** RESET OLED VALUES *****/
101 if (initOutput == 1 && count1 < 10) // Pre-zero OLED values on startup
102 {
103     subtractValAmp = amplitude;
104     subtractValPhase = phase;
105     piezoBuzzer_stop();
106     count1++;
107
108     if (count1 >= 10) // Delay for stability
109     {
110         initOutput = 0;
111     }
112 }
113
114
115 if (btnPress == 1) // Reset OLED values ob button press
116 {
117     subtractValAmp = amplitude;
118     subtractValPhase = phase;
119 }
120
121 newAmplitude = amplitude - subtractValAmp;
122 newPhase = phase - subtractValPhase;
123
124
125 /***** CONVERT RESULT FOR OLED *****/
126 smpl1 = abs(newAmplitude);
127 dtostrf(smpl1, 4, 0, outputAmp);
128

```

```

...io\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\calcResult.h 3
129     smpl2 = newPhase;
130     dtostrf(smpl2, 4 , 0 , outputPhase);
131
132
133     /***** BUZZER *****/
134
135     if ((PINB & 0b0000100) >= 1)          // Count when BTN pressed
136     {
137         count2++;
138     }
139
140     if (a == 0 && (PINB & 0b0000100) >= 1 && count2 == 7) // Do after BTN press delay
141     {
142         a = 1;
143         c++;
144     }
145     if (c == 1)
146     {
147         state2 = 1;
148     }
149     if (c == 2)
150     {
151         state2 = 0;
152         c = 0;
153     }
154     if (a == 1 && (PINB & 0b0000100) == 0)
155     {
156         a = 0;
157         count2 = 0;
158     }
159
160     buzz1 = abs(newPhase);
161     buzz2 = abs(newAmplitude);
162
163     if (buzz1 > 10 && buzz1 < 150 && state2 == 0) // Buzz if phase is above 10
164     {
165         buzzSwitch = 1;
166     }
167     if (buzz2 > 2 && buzz2 < 150 && state2 == 0) // Buzz if amplitude is above 2
168     {
169         buzzSwitch = 1;
170     }
171
172     if (buzzSwitch == 1)
173     {
174         if (flip1 == 0 && flip2 == 1) // Switch between two sounds
175         {
176             OCR2A = 30; // Update compare value for sound change
177             flip2 = 0;
178         }
179         else
180         {
181             flip2 = 0;
182         }
183         if (flip1 == 1 && flip2 == 0) // Switch between two sounds
184         {
185             OCR2A = 50; // Update compare value for sound change
186             flip1 = 0;
187             flip2 = 1;
188         }
189         else
190         {
191             flip1 = 1;
192         }
193     }

```

```
...io\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\calcResult.h 4
193     piezoBuzzer_start();           // Start sound
194     buzzSwitch = 0;
195 }
196 else
197 {
198     piezoBuzzer_stop();           // Stop sound
199 }
200
201
202 //***** PRINT TO OLED *****/
203 sendStrXY("-----",0,0);
204 sendStrXY(" METAL-DETECTOR ",1,0);
205 sendStrXY("-----",2,0);
206 sendStrXY("Amp:"           ,3,0);
207 sendStrXY(outputAmp        ,3,12);
208 sendStrXY("Phase:"         ,5,0);
209 sendStrXY(outputPhase      ,5,12);
210 sendStrXY("-----",6,0);
211
212 if (state2 == 0)
213 {
214     sendStrXY("# SOUND: ON #",7,0);
215     sendStrXY("# SOUND: ON #",7,0);
216     sendStrXY("# SOUND: ON #",7,0);
217     sendStrXY(" SOUND: ON  ",7,0);
218 }
219 else
220 {
221     sendStrXY("# SOUND: OFF #",7,0);
222     sendStrXY("# SOUND: OFF #",7,0);
223     sendStrXY("# SOUND: OFF #",7,0);
224     sendStrXY(" SOUND: OFF  ",7,0);
225 }
226 }
```

```
...dio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\OLED_init.h 1
1  /*****
2  * DTU - ELECTRO          62739 - METAL DETECTOR          *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519      *
5  * Creation date:         24.01.2021                          *
6  * Update date:          24.01.2021                          *
7  * Version:               1.0                                  *
8  *****/
9  #include "I2C.h"
10 #include "ssd1306.h"
11
12
13 /***** INITIALIZE OLED *****/
14 void OLEDinit()
15 {
16     I2C_Init();          // OLED initialization
17     _i2c_address = 0X78; // OLED address
18     InitializeDisplay(); // OLED initialization
19     reset_display();     // OLED reset
20 }
```

```

...o\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\piezoBuzzer.h 1
1  /*****
2  * DTU - ELECTRO          62739 - METAL DETECTOR          *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519      *
5  * Creation date:         24.01.2021                          *
6  * Update date:           24.01.2021                          *
7  * Version:               1.0                                  *
8  *****/
9
10
11 /***** PIEZO/BUZZER PWM OUT INITIALIZATION *****/
12 void piezoBuzzer_init()
13 {
14     /* TIMER/COUNTER 2 FAST PWM OUTPUT ON OC2B/PD3 */
15     DDRD  |= (1<<DDD3);
16     TCCR2B |= (1<<WGM22);          // Enable Fast PWM OCRA = TOP
17     TCCR2A |= (1<<WGM21) | (1<<WGM20); // Enable Fast PWM OCRA = TOP
18     TCCR2A |= (1<<COM2B1) | (0<<COM2B0); // Clear OC2B on compare match
19     OCR2A  = 255;                  // 61 Hz with prescale 1024 and TOP = 255
20 }
21
22
23 /***** PIEZO/BUZZER START *****/
24 void piezoBuzzer_start()
25 {
26     TCCR2B |= (1<<CS22) | (1<<CS21) | (1<<CS20); // Prescale F_CPU/1024
27 }
28
29
30 /***** PIEZO/BUZZER STOP *****/
31 void piezoBuzzer_stop()
32 {
33     TCCR2B &= 0b11111000;          // Prescale F_CPU/1024
34 }
35

```



```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h 1
1  /*****
2  * DTU - ELECTRO                      62739 - METAL DETECTOR                      *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519                      *
5  * Creation date:         24.01.2021                                           *
6  * Update date:          24.01.2021                                           *
7  * Version:               1.0                                                 *
8  *****/
9
10 #define tone1 10
11 #define tone3 30
12 #define tone5 50
13
14
15 /***** STARTUP ANIMATION AND SOUND *****/
16 void runOnce()
17 {
18     OCR2A = tone5;
19     piezoBuzzer_start();
20     sendStrXY("-----",2,0);
21     sendStrXY("-----",4,0);
22     sendStrXY(" METAL-DETECTOR ",3,0);
23     sendStrXY(" M ",3,0);
24     sendStrXY(" ME ",3,0);
25     sendStrXY(" E ",3,0);
26     sendStrXY(" ET ",3,0);
27     sendStrXY(" T ",3,0);
28     sendStrXY(" TA ",3,0);
29     sendStrXY(" A ",3,0);
30     OCR2A = tone5;
31     sendStrXY(" AL ",3,0);
32     sendStrXY(" L ",3,0);
33     sendStrXY(" L- ",3,0);
34     sendStrXY(" - ",3,0);
35     sendStrXY(" -D ",3,0);
36     sendStrXY(" D ",3,0);
37     sendStrXY(" DE ",3,0);
38     sendStrXY(" E ",3,0);
39     sendStrXY(" ET ",3,0);
40     sendStrXY(" T ",3,0);
41     OCR2A = tone3;
42     sendStrXY(" TE ",3,0);
43     sendStrXY(" E ",3,0);
44     sendStrXY(" EC ",3,0);
45     sendStrXY(" C ",3,0);
46     sendStrXY(" CT ",3,0);
47     sendStrXY(" T ",3,0);
48     sendStrXY(" TO ",3,0);
49     sendStrXY(" O ",3,0);
50     sendStrXY(" OR ",3,0);
51     sendStrXY(" R ",3,0);
52     OCR2A = tone5;
53     sendStrXY(" OR ",3,0);
54     sendStrXY(" O ",3,0);
55     sendStrXY(" TO ",3,0);
56     sendStrXY(" T ",3,0);
57     sendStrXY(" CT ",3,0);
58     sendStrXY(" C ",3,0);
59     sendStrXY(" EC ",3,0);
60     sendStrXY(" E ",3,0);
61     sendStrXY(" TE ",3,0);
62     sendStrXY(" T ",3,0);
63     OCR2A = tone5;
64     sendStrXY(" ET ",3,0);

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h

2

```

65     sendStrXY("      E      ",3,0);
66     sendStrXY("      DE     ",3,0);
67     sendStrXY("      D      ",3,0);
68     sendStrXY("      -D     ",3,0);
69     sendStrXY("      -      ",3,0);
70     sendStrXY("      L-     ",3,0);
71     sendStrXY("      L      ",3,0);
72     sendStrXY("      AL     ",3,0);
73     sendStrXY("      A      ",3,0);
74     OCR2A = tone3;
75     sendStrXY("      TA     ",3,0);
76     sendStrXY("      T      ",3,0);
77     sendStrXY("      ET     ",3,0);
78     sendStrXY("      E      ",3,0);
79     sendStrXY("      ME     ",3,0);
80     sendStrXY("      M      ",3,0);
81     sendStrXY("      M      ",3,0);
82     sendStrXY("      ",3,0);
83     sendStrXY("      ",3,0);
84     sendStrXY("      ",3,0);
85     sendStrXY("      METAL-DETECTOR ",3,0);
86     sendStrXY("      METAL-DETECTOR ",3,0);
87     sendStrXY("      ",3,0);
88     sendStrXY("      ",3,0);
89     sendStrXY("      METAL-DETECTOR ",3,0);
90     piezoBuzzer_stop();
91     sendStrXY("      METAL-DETECTOR ",3,0);
92     sendStrXY("      ",3,0);
93     sendStrXY("      ",3,0);
94     sendStrXY("      METAL-DETECTOR ",3,0);
95     sendStrXY("      METAL-DETECTOR ",3,0);
96     sendStrXY("      METAL-DETECTOR ",3,0);
97     sendStrXY("      METAL-DETECTOR ",3,0);
98     sendStrXY("      METAL-DETECTOR ",3,0);
99     sendStrXY("      *      ",0,0);
100    sendStrXY("      *      ",7,0);
101    sendStrXY("      *      ",5,0);
102    sendStrXY("      *      ",1,0);
103    sendStrXY("      *      ",6,0);
104    sendStrXY("      *      ",0,0);
105    sendStrXY("      *      ",7,0);
106    sendStrXY("      *      ",5,0);
107    sendStrXY("      *      ",1,0);
108    sendStrXY("      *      ",6,0);
109    sendStrXY("      *      ",0,0);
110    sendStrXY("      *      ",7,0);
111    sendStrXY("      *      ",5,0);
112    sendStrXY("      *      ",1,0);
113    sendStrXY("      *      ",6,0);
114    sendStrXY("      *      ",0,0);
115    sendStrXY("      *      ",7,0);
116    sendStrXY("      *      ",5,0);
117    sendStrXY("      *      ",1,0);
118    sendStrXY("      *      ",6,0);
119    sendStrXY("      *      ",0,0);
120    sendStrXY("      *      ",7,0);
121    sendStrXY("      *      ",5,0);
122    sendStrXY("      *      ",1,0);
123    sendStrXY("      *      ",6,0);
124    sendStrXY("      *      ",0,0);
125    sendStrXY("      *      ",7,0);
126    sendStrXY("      *      ",5,0);
127    sendStrXY("      *      ",1,0);
128    sendStrXY("      *      ",6,0);

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h

3

```
129     sendStrXY("          *",0,0);
130     sendStrXY("      *          ",7,0);
131     sendStrXY("      *          ",5,0);
132     sendStrXY("          *          ",1,0);
133     sendStrXY("          *          ",6,0);
134     sendStrXY("          *          ",0,0);
135     sendStrXY("          *          ",7,0);
136     sendStrXY("          *          ",5,0);
137     sendStrXY("      *          ",1,0);
138     sendStrXY("          *          ",6,0);
139     sendStrXY("      *          ",0,0);
140     sendStrXY("          *",7,0);
141     sendStrXY("          *",5,0);
142     sendStrXY("          *          ",1,0);
143     sendStrXY("      *",6,0);
144     sendStrXY("          *          ",0,0);
145     sendStrXY("          *          ",7,0);
146     sendStrXY("      *          ",5,0);
147     sendStrXY("      *          ",1,0);
148     sendStrXY("          *          ",6,0);
149     sendStrXY("      *          ",0,0);
150     sendStrXY("      *          ",7,0);
151     sendStrXY("          *          ",5,0);
152     sendStrXY("      *",1,0);
153     sendStrXY("          *          ",6,0);
154     sendStrXY("          *",0,0);
155     sendStrXY("      *          ",7,0);
156     sendStrXY("      *          ",5,0);
157     sendStrXY("          *          ",1,0);
158     sendStrXY("          *          ",6,0);
159     sendStrXY("          *          ",0,0);
160     sendStrXY("          *          ",7,0);
161     sendStrXY("          *          ",5,0);
162     sendStrXY("      *          ",1,0);
163     sendStrXY("          *          ",6,0);
164     sendStrXY("          *          ",0,0);
165     sendStrXY("          *",7,0);
166     sendStrXY("          *",5,0);
167     sendStrXY("          *          ",1,0);
168     sendStrXY("      *",6,0);
169     sendStrXY("          *          ",0,0);
170     sendStrXY("          *          ",7,0);
171     sendStrXY("      *          ",5,0);
172     sendStrXY("      *          ",1,0);
173     sendStrXY("          *          ",6,0);
174     sendStrXY("      *          ",0,0);
175     sendStrXY("      *          ",7,0);
176     sendStrXY("          *          ",5,0);
177     sendStrXY("      *",1,0);
178     sendStrXY("          *          ",6,0);
179     sendStrXY("          *",0,0);
180     sendStrXY("      *          ",7,0);
181     sendStrXY("      *          ",5,0);
182     sendStrXY("          *          ",1,0);
183     sendStrXY("          *          ",6,0);
184     sendStrXY("          *          ",0,0);
185     sendStrXY("          *          ",7,0);
186     sendStrXY("          *          ",5,0);
187     sendStrXY("      *          ",1,0);
188     sendStrXY("          *          ",6,0);
189     sendStrXY("          *          ",0,0);
190     sendStrXY("          *",7,0);
191     sendStrXY("          *",5,0);
192     sendStrXY("          *          ",1,0);
```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h

4

```
193     sendStrXY(" *      ",6,0);
194     sendStrXY("      *      ",0,0);
195     sendStrXY("      *      ",7,0);
196     sendStrXY(" *      ",5,0);
197     sendStrXY(" *      ",1,0);
198     sendStrXY("      *      ",6,0);
199     sendStrXY("      *      ",0,0);
200     sendStrXY(" *      ",7,0);
201     sendStrXY("      *      ",5,0);
202     sendStrXY(" *      ",1,0);
203     sendStrXY("      *      ",6,0);
204     sendStrXY("      *      ",0,0);
205     sendStrXY(" *      ",7,0);
206     sendStrXY(" *      ",5,0);
207     sendStrXY("      *      ",1,0);
208     sendStrXY("      *      ",6,0);
209     sendStrXY("      *      ",0,0);
210     sendStrXY("      *      ",7,0);
211     sendStrXY("      *      ",5,0);
212     sendStrXY(" *      ",1,0);
213     sendStrXY("      *      ",6,0);
214     sendStrXY("      *      ",0,0);
215     sendStrXY("      *      ",7,0);
216     sendStrXY("      *      ",5,0);
217     sendStrXY("      *      ",1,0);
218     sendStrXY(" *      ",6,0);
219     sendStrXY("      *      ",0,0);
220     sendStrXY("      *      ",7,0);
221     sendStrXY(" *      ",5,0);
222     sendStrXY(" *      ",1,0);
223     sendStrXY("      *      ",6,0);
224     sendStrXY(" *      ",0,0);
225     sendStrXY(" *      ",7,0);
226     sendStrXY("      *      ",5,0);
227     sendStrXY(" *      ",1,0);
228     sendStrXY("      *      ",6,0);
229     sendStrXY("      *      ",0,0);
230     sendStrXY(" *      ",7,0);
231     sendStrXY(" *      ",5,0);
232     sendStrXY("      *      ",1,0);
233     sendStrXY("      *      ",6,0);
234     sendStrXY("      *      ",0,0);
235     sendStrXY("      *      ",7,0);
236     sendStrXY("      *      ",5,0);
237     sendStrXY(" *      ",1,0);
238     sendStrXY("      *      ",6,0);
239     sendStrXY("      *      ",0,0);
240     sendStrXY("      *      ",7,0);
241     sendStrXY("      *      ",5,0);
242     sendStrXY("      *      ",1,0);
243     sendStrXY(" *      ",6,0);
244     sendStrXY("      *      ",0,0);
245     sendStrXY("      *      ",7,0);
246     sendStrXY(" *      ",5,0);
247     sendStrXY(" *      ",1,0);
248     sendStrXY("      *      ",6,0);
249     sendStrXY(" *      ",0,0);
250     sendStrXY(" *      ",7,0);
251     sendStrXY("      *      ",5,0);
252     sendStrXY(" *      ",1,0);
253     sendStrXY("      *      ",6,0);
254     sendStrXY("      *      ",0,0);
255     sendStrXY(" *      ",7,0);
256     sendStrXY(" *      ",5,0);
```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h

5

```
257     sendStrXY("      *      ",1,0);
258     sendStrXY("      *      ",6,0);
259     sendStrXY("      *      ",0,0);
260     sendStrXY("      *      ",7,0);
261     sendStrXY("      *      ",5,0);
262     sendStrXY("    *      ",1,0);
263     sendStrXY("      *      ",6,0);
264     sendStrXY("      *      ",0,0);
265     sendStrXY("      *      ",7,0);
266     sendStrXY("      *      ",5,0);
267     sendStrXY("      *      ",1,0);
268     sendStrXY(" *      ",6,0);
269     sendStrXY("      *      ",0,0);
270     sendStrXY("      *      ",7,0);
271     sendStrXY(" *      ",5,0);
272     sendStrXY(" *      ",1,0);
273     sendStrXY("      *      ",6,0);
274     sendStrXY("    *      ",0,0);
275     sendStrXY(" *      ",7,0);
276     sendStrXY("      *      ",5,0);
277     sendStrXY(" *      ",1,0);
278     sendStrXY("      *      ",6,0);
279     sendStrXY("      *      ",0,0);
280     sendStrXY("    *      ",7,0);
281     sendStrXY("    *      ",5,0);
282     sendStrXY("      *      ",1,0);
283     sendStrXY("      *      ",6,0);
284     sendStrXY("      *      ",0,0);
285     sendStrXY("      *      ",7,0);
286     sendStrXY("      *      ",5,0);
287     sendStrXY("    *      ",1,0);
288     sendStrXY("      *      ",6,0);
289     sendStrXY("      *      ",0,0);
290     sendStrXY("      *      ",7,0);
291     sendStrXY("      *      ",5,0);
292     sendStrXY("      *      ",1,0);
293     sendStrXY(" *      ",6,0);
294     sendStrXY("      *      ",0,0);
295     sendStrXY("      *      ",7,0);
296     sendStrXY(" *      ",5,0);
297     sendStrXY(" *      ",1,0);
298     sendStrXY("      *      ",6,0);
299     sendStrXY("    *      ",0,0);
300     sendStrXY(" *      ",7,0);
301     sendStrXY("      *      ",5,0);
302     sendStrXY(" *      ",1,0);
303     sendStrXY("      *      ",6,0);
304     sendStrXY("      *      ",0,0);
305     sendStrXY("    *      ",7,0);
306     sendStrXY("    *      ",5,0);
307     sendStrXY("      *      ",1,0);
308     sendStrXY("      *      ",6,0);
309     sendStrXY("      *      ",0,0);
310     sendStrXY("      *      ",7,0);
311     sendStrXY("      *      ",5,0);
312     sendStrXY("    *      ",1,0);
313     sendStrXY("      *      ",6,0);
314     sendStrXY("      *      ",0,0);
315     sendStrXY("      *      ",7,0);
316     sendStrXY("      *      ",5,0);
317     sendStrXY("      *      ",1,0);
318     sendStrXY(" *      ",6,0);
319     sendStrXY("      *      ",0,0);
320     sendStrXY("      *      ",7,0);
```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\startUp.h

6

```
321     sendStrXY(" *           ",5,0);
322     sendStrXY(" *           ",1,0);
323     sendStrXY(" *           ",6,0);
324     piezoBuzzer_start();
325     OCR2A = tone1;
326     clear_display();
327 }
```

```

...\\7.0\\3UGERS-V2021\\Samlet projekt\\Version01\\Version01\\timerCounter.h 1
1  /*****
2  * DTU - ELECTRO          62739 - METAL DETECTOR          *
3  *****/
4  * Author:                Jørgen Drelicharz Greve - s181519      *
5  * Creation date:         24.01.2021                          *
6  * Update date:          24.01.2021                          *
7  * Version:               1.0                                *
8  *****/
9
10
11 /***** ADC SAMPLE TIMER 8 KHZ *****/
12 void counter0_init()
13 {
14     TCCR0A |= (1<<WGM01);          // Enable CTC, TOP = OCR0A, TOV Flag set on MAX
15     OCR0A = 249;                  // 8000 Hz with prescale 8 and TOP = OCR0A = 249
16 }
17
18
19 /***** ADC SAMPLING BEGIN *****/
20 void counter0_start()
21 {
22     TIMSK0 |= (1<<OCIE0A);          // Output Compare A Match Interrupt Enable
23     TCCR0B |= (0<<CS02) | (1<<CS01) | (0<<CS00); // Prescale F_CPU/8
24 }
25
26
27 /***** ADC SAMPLING END *****/
28 void counter0_stop()
29 {
30     TCCR0B &= 0b11111000;          // No clock = counter stopped
31     TIMSK0 |= (0<<OCIE0A);          // Output Compare A Match Interrupt disable
32 }
33

```

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c 1
1  /**
2   * ssd1306.c
3   *
4   * Created: 03-01-2018 17:33:26
5   * Author: osc
6   *this modules purpose is to initiate and control the OLED display SSD1306 driver for the 128x64
   OLED display
7   *The plus can be connected to PIN 24 and the GND to PIN 26 and sda goes to PIN 21 and the sck
   goes to PIN 20
8   *http://microcontrolandos.blogspot.dk/2014/12/pic-ssd1306.html
9   * SSD1306 Advanced information Matrix apr. 2008 rev. 1.1 www.solomon-systech.com describes all
   used commands and data used for the display 128x64 dot matrix
10  *for Mega2560 arduino board
11  *p reference to this documentation
12  *initialization is from the DD-2864bY-3A rev c p 19
13  *both data sheets are at git hub in this project
14  */
15  #include <math.h>
16  #include <string.h>
17  #include "I2C.h"
18  // #define F_CPU 16000000UL
19  #include <util/delay.h>
20  #include <avr/pgmspace.h>
21  #include "ssd1306.h"
22  #include "data.h"
23  #define ssd1306_swap(a, b) { int16_t t = a; a = b; b = t; }
24  #define _vccstate 1 //externalVcc
25
26  uint8_t _i2c_address=0x78; //display write address
27
28  /**write a command to the ssd1306*/
29  void ssd1306_command(uint8_t c)
30  {
31      uint8_t control = 0x00; // some use 0X00 other examples use 0X80. I tried both
32      I2C_Start(_i2c_address);
33      //I2C_Write();
34      I2C_Write(control); // This is Command
35      I2C_Write(c);
36      I2C_Stop();
37  }
38  //////////////////////////////////////
39  //
40  /**write a a data byte to the ssd1306*/
41  void ssd1306_data(uint8_t c)
42  {
43      I2C_Start(_i2c_address);
44      I2C_Write(_i2c_address);
45      I2C_Write(0X40); // This byte is DATA
46      I2C_Write(c);
47      I2C_Stop();
48  }
49  //////////////////////////////////////
50  /** Used when doing Horizontal or Vertical Addressing*/
51  void setColAddress()
52  {
53      ssd1306_command(SSD1306_COLUMNADDR); // 0x21 COMMAND
54      ssd1306_command(0); // Column start address
55      ssd1306_command(SSD1306_LCDWIDTH-1); // Column end address
56  }
57  //////////////////////////////////////
58  /** Used when doing Horizontal or Vertical Addressing*/
59  void setPageAddress()
60  {
61      ssd1306_command(SSD1306_PAGEADDR); // 0x22 COMMAND

```



```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c 2
62     ssd1306_command(0); // Start Page address
63     ssd1306_command((SSD1306_LCDHEIGHT/8)-1); // End Page address
64 }
65 ///////////////////////////////////////////////////////////////////
66 /** init according to SSD1306 data sheet and using the plus can be connected to PIN 24 and the
    GND to PIN 26 */
67
68 void InitializeDisplay()
69 {
70
71
72
73     // Init sequence for 128x64 OLED module
74     ssd1306_command(SSD1306_DISPLAYOFF); // 0xAE
75
76     ssd1306_command(SSD1306_SETDISPLAYCLOCKDIV); // 0xD5
77     ssd1306_command(0x80); // the suggested ratio 0x80
78
79     ssd1306_command(SSD1306_SETMULTIPLEX); // 0xA8
80     ssd1306_command(0x3F);
81
82     ssd1306_command(SSD1306_SETDISPLAYOFFSET); // 0xD3
83     ssd1306_command(0x0); // no offset
84
85     ssd1306_command(SSD1306_SETSTARTLINE); // 0x0; // line #0
86
87     ssd1306_command(SSD1306_CHARGEPUMP); // 0x8D
88     ssd1306_command(0x14); // using internal VCC
89
90     ssd1306_command(SSD1306_MEMORYMODE); // 0x20
91     ssd1306_command(0x00); // 0x00 horizontal addressing automatic line
    shift
92
93     ssd1306_command(SSD1306_SEGREMAP | 0x1); // rotate screen 180
94
95     ssd1306_command(SSD1306_COMSCANDEC); // rotate screen 180
96
97     ssd1306_command(SSD1306_SETCOMPINS); // 0xDA
98     ssd1306_command(0x12);
99
100    ssd1306_command(SSD1306_SETCONTRAST); // 0x81
101    ssd1306_command(0xCF);
102
103    ssd1306_command(SSD1306_SETPRECHARGE); // 0xD9
104    ssd1306_command(0xF1);
105
106    ssd1306_command(SSD1306_SETVCOMDETECT); // 0xDB
107    ssd1306_command(0x40);
108
109    ssd1306_command(SSD1306_DISPLAYALLON_RESUME); // 0xA4
110
111    ssd1306_command(SSD1306_NORMALDISPLAY); // 0xA6
112
113    ssd1306_command(SSD1306_DISPLAYON); //switch on OLED
114
115 }
116
117 /** reset the display*/
118 void reset_display(void)
119 {
120     displayOff();
121
122     clear_display();
123

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c

3

```
124
125     displayOn();
126 }
127
128
129
130
131 //=====//
132 /** Turns display on.*/
133 void displayOn(void)
134 {
135     ssd1306_command(0xaf);    //display on p. 28
136 }
137
138 //=====//
139 /** Turns display off.*/
140 void displayOff(void)
141 {
142     ssd1306_command(0xae);    //display off p. 28
143 }
144
145 //=====//
146 /** Clears the display by sending 0 to all the screen map.*/
147 void clear_display(void)
148 {
149     unsigned char i,k;
150     for(k=0;k<8;k++)
151     {
152         setXY(k,0);
153         {
154             for(i=0;i<128;i++)    //clear all COL
155             {
156                 SendChar(0);    //clear all COL
157                 //delay(10);
158             }
159         }
160     }
161 }
162
163
164
165
166
167
168 //=====//
169 /**print integer string in big font*/
170 void printBigTime(char *string)
171 {
172
173     int Y=0;
174     int lon = strlen(string);
175     if(lon == 3) {
176         Y = 0;
177     } else if (lon == 2) {
178         Y = 3;
179     } else if (lon == 1) {
180         Y = 6;
181     }
182
183     int X = 4;
184     while(*string)
185     {
186         printBigNumber(*string, X, Y);
187     }
```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c

4

```

188     Y+=3;
189     X=4;
190     setXY(X,Y);
191     string++;
192 }
193 }
194
195
196 //=====//
197 /** Prints a display big number (96 bytes) in coordinates X Y,
198 * being multiples of 8. This means we have 16 COLS (0-15)
199 * and 8 ROWS (0-7).*/
200 void printBigNumber(char string, int X, int Y)
201 {
202     setXY(X,Y);           //set the cursor to start
203     int salto=0;
204     for(int i=0;i<96;i++)
205     {
206         if(string == ' ') {
207             SendChar(0);
208         } else
209             SendChar(pgm_read_byte(bigNumbers[string-0x30]+i));
210
211         if(salto == 23) {
212             salto = 0;
213             X++;
214             setXY(X,Y);
215         } else {
216             salto++;
217         }
218     }
219 }
220
221 //=====//
222 /** Actually this sends a byte, not a char to draw in the display.
223 * Displays chars uses 8 byte font the small ones and 96 bytes
224 * for the big number font.*/
225 void SendChar(unsigned char data)
226 {
227     I2C_Start(_i2c_address); // begin transmitting
228     I2C_Write(0x40); //data mode
229     I2C_Write(data);
230     I2C_Stop(); // stop transmitting
231 }
232
233 //=====//
234 /** Prints a display char (not just a byte) in coordinates X Y,
235 * being multiples of 8. This means we have 16 COLS (0-15)
236 * and 8 ROWS (0-7).*/
237 void sendCharXY(unsigned char data, int X, int Y)
238 {
239     setXY(X, Y);
240     I2C_Start(_i2c_address); // begin transmitting
241     I2C_Write(0x40); //data mode
242
243     for(int i=0;i<8;i++)
244         I2C_Write(pgm_read_byte(myFont[data-0x20]+i));
245
246     I2C_Stop(); // stop transmitting
247 }
248
249 //=====//
250 /** Set the cursor position in a 16 COL * 8 ROW map.*/
251 void setXY(unsigned char row,unsigned char col)

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c

5

```

252 {
253     ssd1306_command(0xb0+row);           //set page address    p. 31
254     ssd1306_command(0x00+(8*col&0x0f));  //set low col address  p. 30
255     ssd1306_command(0x10+((8*col>>4)&0x0f)); //set high col address  p.30
256 }
257
258
259 //=====//
260 /** Prints a string regardless the cursor position.*/
261 void sendStr(char *string)
262 {
263     unsigned char i=0;
264     while(*string)
265     {
266         for(i=0;i<8;i++)
267         {
268             SendChar(pgm_read_byte(myFont[*string-0x20]+i)); //look up ascii chars (no danish)
269                 defined in data.h
270         }
271         string++;
272     }
273
274 //=====//
275 /** Prints a string in coordinates X Y, being multiples of 8.
276 * This means we have 16 COLS (0-15) and 8 ROWS (0-7).*/
277 void sendStrXY( char *string, int X, int Y)
278 {
279     setXY(X,Y);
280     unsigned char i=0;
281     while(*string)
282     {
283         if (*string=='\n'){
284             setXY(X+1,0);
285             string++;
286         }
287         for(i=0;i<8;i++)
288         {
289             SendChar(pgm_read_byte(myFont[*string-0x20]+i));
290         }
291         string++;
292     }
293 }
294 void ssd1306_setpos(uint8_t x, uint8_t y)
295 {
296     ssd1306_command(0xb0 + y);
297     ssd1306_command(((x & 0xf0) >> 4) | 0x10); // | 0x10
298
299
300 }
301 void print_fonts(){
302     clear_display();
303
304     uint8_t data=32;
305     for(int k=0;k<6;k++){
306         setXY(k,0);
307
308         for (int j=0;j<16;j++)
309         {
310             for(int i=0;i<8;i++){
311                 SendChar(pgm_read_byte(myFont[(data+j)-0x20]+i));
312             }
313         }
314     }

```

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c
315     data=data+16;
316 }
317 }
318 void ssd1306_draw_bmp(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1, const uint8_t bitmap[])
319 {
320     uint16_t j = 0;
321     uint8_t y;
322     if (y1 % 8 == 0) y = y1 / 8;
323     else y = y1 / 8 + 1;
324     for (y = y0; y < y1; y++)
325     {
326         ssd1306_setpos(x0,y);
327
328         for (uint8_t x = x0; x < x1; x++)
329         {
330             ssd1306_data(pgm_read_byte(&bitmap[j++]));
331         }
332     }
333 }
334 }
335 /*
336 void drawPixel(int16_t x, int16_t y, uint16_t color) {
337     if ((x < 0) || (x >= width()) || (y < 0) || (y >= height()))
338         return;
339
340     // check rotation, move pixel around if necessary
341     switch (getRotation()) {
342         case 1:
343             ssd1306_swap(x, y);
344             x = WIDTH - x - 1;
345             break;
346         case 2:
347             x = WIDTH - x - 1;
348             y = HEIGHT - y - 1;
349             break;
350         case 3:
351             ssd1306_swap(x, y);
352             y = HEIGHT - y - 1;
353             break;
354     }
355
356     // x is which column
357     switch (color)
358     {
359         case WHITE:  buffer[x+ (y/8)*SSD1306_LCDWIDTH] |= (1 << (y&7)); break;
360         case BLACK:  buffer[x+ (y/8)*SSD1306_LCDWIDTH] &= ~(1 << (y&7)); break;
361         case INVERSE: buffer[x+ (y/8)*SSD1306_LCDWIDTH] ^= (1 << (y&7)); break;
362     }
363
364 }
365 */
366 void invertDisplay(uint8_t i) {
367     if (i) {
368         ssd1306_command(SSD1306_INVERTDISPLAY);
369     } else {
370         ssd1306_command(SSD1306_NORMALDISPLAY);
371     }
372 }
373 /** startscrollright
374  * Activate a right handed scroll for rows start through stop
375  * Hint, the display is 16 rows tall. To scroll the whole display, run:
376  * scroll right(0x00, 0x0F)*/
377 void startscrollright(uint8_t start, uint8_t stop){
378     ssd1306_command(SSD1306_RIGHT_HORIZONTAL_SCROLL);

```

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c
379     ssd1306_command(0X00);
380     ssd1306_command(start);
381     ssd1306_command(0X00);
382     ssd1306_command(stop);
383     ssd1306_command(0X00);
384     ssd1306_command(0XFF);
385     ssd1306_command(SSD1306_ACTIVATE_SCROLL);
386 }
387 /** startscrollleft
388  * Activate a right handed scroll for rows start through stop
389  *Hint, the display is 16 rows tall. To scroll the whole display, run:
390  * scrollleft(0x00, 0x0F) */
391 void startscrollleft(uint8_t start, uint8_t stop){
392     ssd1306_command(SSD1306_LEFT_HORIZONTAL_SCROLL);
393     ssd1306_command(0X00);
394     ssd1306_command(start);
395     ssd1306_command(0X00);
396     ssd1306_command(stop);
397     ssd1306_command(0X00);
398     ssd1306_command(0XFF);
399     ssd1306_command(SSD1306_ACTIVATE_SCROLL);
400 }
401 /** startscrollldiagright
402  *Activate a diagonal scroll for rows start through stop
403  * Hint, the display is 16 rows tall. To scroll the whole display, run:
404  * display.scrollright(0x00, 0x0F)*/
405 void startscrollldiagright(uint8_t start, uint8_t stop){
406     ssd1306_command(SSD1306_SET_VERTICAL_SCROLL_AREA);
407     ssd1306_command(0X00);
408     ssd1306_command(SSD1306_LCDHEIGHT);
409     ssd1306_command(SSD1306_VERTICAL_AND_RIGHT_HORIZONTAL_SCROLL);
410     ssd1306_command(0X00);
411     ssd1306_command(start);
412     ssd1306_command(0X00);
413     ssd1306_command(stop);
414     ssd1306_command(0X01);
415     ssd1306_command(SSD1306_ACTIVATE_SCROLL);
416 }
417 /** startscrollldiagleft
418  * Activate a diagonal scroll for rows start through stop
419  *Hint, the display is 16 rows tall. To scroll the whole display, run:
420  * display.scrollright(0x00, 0x0F)*/
421 void startscrollldiagleft(uint8_t start, uint8_t stop){
422     ssd1306_command(SSD1306_SET_VERTICAL_SCROLL_AREA);
423     ssd1306_command(0X00);
424     ssd1306_command(SSD1306_LCDHEIGHT);
425     ssd1306_command(SSD1306_VERTICAL_AND_LEFT_HORIZONTAL_SCROLL);
426     ssd1306_command(0X00);
427     ssd1306_command(start);
428     ssd1306_command(0X00);
429     ssd1306_command(stop);
430     ssd1306_command(0X01);
431     ssd1306_command(SSD1306_ACTIVATE_SCROLL);
432 }
433 void stopscroll(void){
434     ssd1306_command(SSD1306_DEACTIVATE_SCROLL);
435 }
436
437 // Dim the display
438 // dim = true: display is dimmed
439 // dim = false: display is normal
440 void dim(bool dim) {
441     uint8_t contrast;
442

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.c

8

```
443     if (dim) {
444         contrast = 0; // Dimmed display
445     } else {
446         if (_vccstate == SSD1306_EXTERNALVCC) {
447             contrast = 0x9F;
448         } else {
449             contrast = 0xCF;
450         }
451     }
452     // the range of contrast to too small to be really useful
453     // it is useful to dim the display
454     ssd1306_command(SSD1306_SETCONTRAST);
455     ssd1306_command(contrast);
456 }
```

```

...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.h 1
1  /*
2   * ssd1306.h
3   *based upon Adafruit_SSD1306 at github -adjusted to C using i2c address 0x78 for writing and 0x7A for reading
4   * Created: 03-01-2018 17:33:51
5   * Author: osch
6   *page refers to: SSD1306 Advanced information Matrix apr. 2008 rev. 1.1 www.solomon-systech.com describes all used commands and data used for the display 128x64 dot matrix
7   *p reference to this documentation
8   *initialization is from the DD-2864bY-3A rev c p 19
9   *both data sheets are at git hub in this project
10  */
11
12  #include <stdbool.h>
13
14  #define SSD1306_128_64
15  //#define SSD1306_128_32
16  // #define SSD1306_96_16
17  #if defined SSD1306_128_64 && defined SSD1306_128_32
18  #error "Only one SSD1306 display can be specified at once in SSD1306.h"
19  #endif
20  #if !defined SSD1306_128_64 && !defined SSD1306_128_32 && !defined SSD1306_96_16
21  #error "At least one SSD1306 display must be specified in SSD1306.h"
22  #endif
23
24  #if defined SSD1306_128_64
25  #define SSD1306_LCDWIDTH 128
26  #define SSD1306_LCDHEIGHT 64
27  #endif
28  #if defined SSD1306_128_32
29  #define SSD1306_LCDWIDTH 128
30  #define SSD1306_LCDHEIGHT 32
31  #endif
32  #if defined SSD1306_96_16
33  #define SSD1306_LCDWIDTH 96
34  #define SSD1306_LCDHEIGHT 16
35  #endif
36
37  // #define pgm_read_byte(addr) (*(const unsigned char *)(addr))
38  //command data defined p 28 - 32
39  #define SSD1306_LCDWIDTH 128
40  #define SSD1306_LCDHEIGHT 64
41  #define SSD1306_SETCONTRAST 0x81
42  #define SSD1306_DISPLAYALLON_RESUME 0xA4
43  #define SSD1306_DISPLAYALLON 0xA5
44  #define SSD1306_NORMALDISPLAY 0xA6
45  #define SSD1306_INVERTDISPLAY 0xA7
46  #define SSD1306_DISPLAYOFF 0xAE
47  #define SSD1306_DISPLAYON 0xAF
48  #define SSD1306_SETDISPLAYOFFSET 0xD3
49  #define SSD1306_SETCOMPINS 0xDA
50  #define SSD1306_SETVCOMDETECT 0xDB
51  #define SSD1306_SETDISPLAYCLOCKDIV 0xD5
52  #define SSD1306_SETPRECHARGE 0xD9
53  #define SSD1306_SETMULTIPLEX 0xA8
54  #define SSD1306_SETLOWCOLUMN 0x00
55  #define SSD1306_SETHIGHCOLUMN 0x10
56  #define SSD1306_SETSTARTLINE 0x40
57  #define SSD1306_MEMORYMODE 0x20
58  #define SSD1306_COLUMNADDR 0x21
59  #define SSD1306_PAGEADDR 0x22
60  #define SSD1306_COMSCANINC 0xC0
61  #define SSD1306_COMSCANDEC 0xC8
62  #define SSD1306_SEGREMAP 0xA0

```



```
...tudio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\ssd1306.h 2
63 #define SSD1306_CHARGE_PUMP 0x8D
64 #define SSD1306_EXTERNAL_VCC 0x1
65 #define SSD1306_SWITCHCAPVCC 0x2
66 // Scrolling #defines
67 #define SSD1306_ACTIVATE_SCROLL 0x2F
68 #define SSD1306_DEACTIVATE_SCROLL 0x2E
69 #define SSD1306_SET_VERTICAL_SCROLL_AREA 0xA3
70 #define SSD1306_RIGHT_HORIZONTAL_SCROLL 0x26
71 #define SSD1306_LEFT_HORIZONTAL_SCROLL 0x27
72 #define SSD1306_VERTICAL_AND_RIGHT_HORIZONTAL_SCROLL 0x29
73 #define SSD1306_VERTICAL_AND_LEFT_HORIZONTAL_SCROLL 0x2A
74
75 #define BLACK 0
76 #define WHITE 1
77 #define INVERSE 2
78
79 typedef uint8_t bitmap_t[8][128];
80 uint8_t _i2c_address;
81 void InitializeDisplay();
82 void sendStrXY( char *string, int X, int Y);
83 void sendStr( char *string);
84 void setXY(unsigned char row,unsigned char col);
85 void sendCharXY(unsigned char data, int X, int Y);
86 void SendChar(unsigned char data);
87 void displayOn(void);
88 void displayOff(void);
89 void clear_display(void);
90 void printBigTime(char *string);
91 void reset_display(void);
92 void printBigNumber(char string, int X, int Y);
93 void bmp(bitmap_t b);
94 void setPageAddress();
95 void setColAddress();
96 void ssd1306_setpos(uint8_t x, uint8_t y);
97 void ssd1306_draw_bmp(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1, const uint8_t bitmap[]);
98 //scroll
99 void startscrollright(uint8_t start, uint8_t stop);
100 void startscrollleft(uint8_t start, uint8_t stop);
101
102 void startscrollrightdiagright(uint8_t start, uint8_t stop);
103 void startscrollleftdiagleft(uint8_t start, uint8_t stop);
104 void stopscroll(void);
105 void dim(bool dim);
106 void print_fonts();
107 void drawPixel(int16_t x, int16_t y, uint16_t color);
108
109
```

...e1 Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\I2C.c

1

```

1  /*
2  * I2C.c
3  *
4  * Created: 22-12-2017 19:00:37
5  * Author: osc
6  *sda goes to PIN 21 and the sck goes to PIN 20*
7  */
8  #include "I2C.h"
9  #include <avr/io.h>
10 /**init for I2C scl set to 100000 kHz*/
11 void I2C_Init()          /* I2C initialize function */
12 {
13     DDRC|=(1<<DDC4);
14     PORTC|=(1<<4);
15     _delay_ms(1000);
16     TWBR=18;
17     TWSR&=0xFC;
18     TWCR=0x05;
19 }
20 /** I2C start function
21 * Return 0 to indicate start condition fail
22 * Return 1 to indicate ack received
23 * Return 2 to indicate nack received*/
24 uint8_t I2C_Start(char write_address)
25 {
26     uint8_t status;      /* Declare variable */
27     TWCR=(1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /* Enable TWI, generate START */
28     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
29     status=TWSR&0xF8;      /* Read TWI status register */
30     if(status!=0x08)        /* Check weather START transmitted or not? */
31     return 0;              /* Return 0 to indicate start condition fail */
32     TWDW=write_address;    /* Write SLA+W in TWI data register */
33     TWCR=(1<<TWEN)|(1<<TWINT); /* Enable TWI & clear interrupt flag */
34     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
35     status=TWSR&0xF8;      /* Read TWI status register */
36     if(status==0x18)        /* Check for SLA+W transmitted &ack received */
37     return 1;              /* Return 1 to indicate ack received */
38     if(status==0x20){      /* Check for SLA+W transmitted &nack received */
39
40     return 2;              /* Return 2 to indicate nack received */
41     }
42 }
43 else
44     return 3;              /* Else return 3 to indicate SLA+W failed */
45 }
46 /** I2C repeated start function */
47 uint8_t I2C_Repeated_Start(char read_address)
48 {
49     uint8_t status;      /* Declare variable */
50     TWCR=(1<<TWSTA)|(1<<TWEN)|(1<<TWINT); /* Enable TWI, generate start */
51     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
52     status=TWSR&0xF8;      /* Read TWI status register */
53     if(status!=0x10)        /* Check for repeated start transmitted */
54     return 0;              /* Return 0 for repeated start condition fail */
55     TWDW=read_address;    /* Write SLA+R in TWI data register */
56     TWCR=(1<<TWEN)|(1<<TWINT); /* Enable TWI and clear interrupt flag */
57     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
58     status=TWSR&0xF8;      /* Read TWI status register */
59     if(status==0x40)        /* Check for SLA+R transmitted &ack received */
60     return 1;              /* Return 1 to indicate ack received */
61     if(status==0x20)        /* Check for SLA+R transmitted &nack received */
62     return 2;              /* Return 2 to indicate nack received */
63     else
64     return 3;              /* Else return 3 to indicate SLA+W failed */

```

...el Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\I2C.c

2

```
65 }
66 uint8_t I2C_Write(char data)    /* I2C write function */
67 {
68     uint8_t status;             /* Declare variable */
69     TWDR=data;                  /* Copy data in TWI data register */
70     TWCR=(1<<TWEN)|(1<<TWINT); /* Enable TWI and clear interrupt flag */
71     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
72     status=TSR&0xF8;            /* Read TWI status register */
73     if(status==0x28)             /* Check for data transmitted &ack received */
74         return 0;               /* Return 0 to indicate ack received */
75     if(status==0x30)             /* Check for data transmitted &nack received */
76         return 1;               /* Return 1 to indicate nack received */
77     else
78         return 2;               /* Else return 2 for data transmission failure */
79 }
80 char I2C_Read_Ack()             /* I2C read ack function */
81 {
82     TWCR=(1<<TWEN)|(1<<TWINT)|(1<<TWEA); /* Enable TWI, generation of ack */
83     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
84     return TWDR;                 /* Return received data */
85 }
86 char I2C_Read_Nack()            /* I2C read nack function */
87 {
88     TWCR=(1<<TWEN)|(1<<TWINT); /* Enable TWI and clear interrupt flag */
89     while(!(TWCR&(1<<TWINT))); /* Wait until TWI finish its current job */
90     return TWDR;                 /* Return received data */
91 }
92
93 void I2C_Stop()                 /* I2C stop function */
94 {
95     TWCR=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN); /* Enable TWI, generate stop */
96     while(TWCR&(1<<TWSTO)); /* Wait until stop condition execution */
97 }
```

...el Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\I2C.h

1

```
1  /**
2   * I2C.h
3   * driver for I2C from AVR freaks adjusted with an init function
4   * Created: 22-12-2017 19:00:53
5   * Author: osch
6   */
7
8
9  #ifndef I2C_H_
10 #define I2C_H_
11 #define SCL_CLK 100000
12 #define F_CPU 16000000UL
13
14 char write_address;
15
16 #include <util/delay.h>
17
18 #define BITRATE(TWSR) (((F_CPU/SCL_CLK)-16)/(2*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1))))))
19 char read_address;
20
21 void I2C_Init() ;
22 uint8_t I2C_Start(char write_address);/* I2C start function */
23 uint8_t I2C_Repeated_Start(char read_address); /* I2C repeated start function */
24 uint8_t I2C_Write(char data); /* I2C write function */
25 char I2C_Read_Ack() ; /* I2C read ack function */
26 char I2C_Read_Nack(); /* I2C read nack function */
27 void I2C_Stop() ; /* I2C stop function */
28 #endif /* I2C_H_ */
```

```

...1 Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\data.h 1
1  /*
2   * This file contains all the static arrays to draw things in the display.
3   *you can create your own fonts using this tool https://www.mikroe.com/glcd-font-creator
4   */
5 #include <avr/pgmspace.h>
6 typedef uint8_t bitmap_t[8][128];
7 typedef char PROGMEM prog_uchar;
8
9 // Big numbers font, from 0 to 9. 96 byte each.
10 const prog_uchar bigNumbers[][96] PROGMEM = {
11 {0x00, 0x00, 0x00, 0xC0, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
12 0xF0, 0xF0, 0xF0, 0xE0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x01,
13 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00,
14 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
15 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0F, 0x0F, 0x0F,
16 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x03, 0x00, 0x00, 0x00},
17
18 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xF0,
19 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
20 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
21 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF,
22 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
23 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x07, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
24
25 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
26 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81, 0xC1, 0xC0, 0xC0, 0xC0,
27 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE1, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
28 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x87, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03,
29 0x03, 0x83, 0x81, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x0F, 0x0F,
30 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
31
32 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
33 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x81, 0xC1, 0xC0, 0xC0, 0xC0,
34 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE1, 0xFF, 0xFF, 0xFF, 0x7F, 0x00, 0x00, 0x00, 0x00,
35 0x00, 0x00, 0x00, 0x81, 0x83, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x87,
36 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x0F, 0x0F,
37 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
38
39 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30,
40 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xE0,
41 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
42 0x00, 0x00, 0x00, 0x81, 0x83, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07,
43 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
44 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
45
46 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
47 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1,
48 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC1, 0x81, 0x00, 0x00, 0x00, 0x00,
49 0x00, 0x00, 0x00, 0x81, 0x83, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x87,
50 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x0F, 0x0F,
51 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
52
53 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
54 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1,
55 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC1, 0x81, 0x00, 0x00, 0x00, 0x00,
56 0x00, 0x00, 0x00, 0x81, 0x83, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x87,
57 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x0F, 0x0F,
58 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
59
60 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
61 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x01, 0x01, 0x00, 0x00, 0x00,
62 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
63 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
64 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```

```

...l Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\data.h 2
65 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
66
67 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
68 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xFF, 0xFF, 0xE1,
69 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE1, 0xFF, 0xFF, 0xFF, 0x3F, 0x00, 0x00, 0x00, 0x00,
70 0x00, 0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0x87, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x87,
71 0xFF, 0xFF, 0xFF, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x0F, 0x0F,
72 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
73
74 {0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0,
75 0xF0, 0xF0, 0xF0, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0xE1,
76 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xE1, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,
77 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x03, 0x07,
78 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
79 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00},
80
81 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
82 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
83 0x00, 0x3C, 0x7E, 0x7E, 0x7E, 0x7E, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
84 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0xF8, 0xF8, 0xF8, 0xF8, 0xF0, 0x00,
85 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
86 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
87 };
88
89 // Big numbers minus symbol.
90 const prog_uchar minus [] PROGMEM = {
91 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
92 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
93 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
94 0x00, 0x00, 0x0C, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
95 0x1E, 0x1E, 0x1E, 0x1E, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
96 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
97 };
98
99 // degrees, outside the ascii table myFont
100 const prog_uchar myDregree [8] PROGMEM = {
101 0x00, 0x00, 0x0C, 0x12, 0x12, 0x0C, 0x00, 0x00
102 };
103
104
105 // Small 8x8 font
106 const prog_uchar myFont[][8] PROGMEM = { //ascii start number 32
107 {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00},
108 {0x00, 0x00, 0x5F, 0x00, 0x00, 0x00, 0x00, 0x00},
109 {0x00, 0x00, 0x07, 0x00, 0x07, 0x00, 0x00, 0x00},
110 {0x00, 0x14, 0x7F, 0x14, 0x7F, 0x14, 0x00, 0x00},
111 {0x00, 0x24, 0x2A, 0x7F, 0x2A, 0x12, 0x00, 0x00},
112 {0x00, 0x23, 0x13, 0x08, 0x64, 0x62, 0x00, 0x00},
113 {0x00, 0x36, 0x49, 0x55, 0x22, 0x50, 0x00, 0x00},
114 {0x00, 0x00, 0x05, 0x03, 0x00, 0x00, 0x00, 0x00},
115 {0x00, 0x1C, 0x22, 0x41, 0x00, 0x00, 0x00, 0x00},
116 {0x00, 0x41, 0x22, 0x1C, 0x00, 0x00, 0x00, 0x00},
117 {0x00, 0x08, 0x2A, 0x1C, 0x2A, 0x08, 0x00, 0x00},
118 {0x00, 0x08, 0x08, 0x3E, 0x08, 0x08, 0x00, 0x00},
119 {0x00, 0xA0, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00},
120 {0x00, 0x08, 0x08, 0x08, 0x08, 0x08, 0x00, 0x00},
121 {0x00, 0x60, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00},
122 {0x00, 0x20, 0x10, 0x08, 0x04, 0x02, 0x00, 0x00},
123 {0x00, 0x3E, 0x51, 0x49, 0x45, 0x3E, 0x00, 0x00},
124 {0x00, 0x00, 0x42, 0x7F, 0x40, 0x00, 0x00, 0x00},
125 {0x00, 0x62, 0x51, 0x49, 0x49, 0x46, 0x00, 0x00},
126 {0x00, 0x22, 0x41, 0x49, 0x49, 0x36, 0x00, 0x00},
127 {0x00, 0x18, 0x14, 0x12, 0x7F, 0x10, 0x00, 0x00},
128 {0x00, 0x27, 0x45, 0x45, 0x45, 0x39, 0x00, 0x00},

```

...l Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\data.h

3

```
129 {0x00,0x3C,0x4A,0x49,0x49,0x30,0x00,0x00},
130 {0x00,0x01,0x71,0x09,0x05,0x03,0x00,0x00},
131 {0x00,0x36,0x49,0x49,0x49,0x36,0x00,0x00},
132 {0x00,0x06,0x49,0x49,0x29,0x1E,0x00,0x00},
133 {0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00},
134 {0x00,0x00,0xAC,0x6C,0x00,0x00,0x00,0x00},
135 {0x00,0x08,0x14,0x22,0x41,0x00,0x00,0x00},
136 {0x00,0x14,0x14,0x14,0x14,0x14,0x00,0x00},
137 {0x00,0x41,0x22,0x14,0x08,0x00,0x00,0x00},
138 {0x00,0x02,0x01,0x51,0x09,0x06,0x00,0x00},
139 {0x00,0x32,0x49,0x79,0x41,0x3E,0x00,0x00},
140 {0x00,0x7E,0x09,0x09,0x09,0x7E,0x00,0x00},
141 {0x00,0x7F,0x49,0x49,0x49,0x36,0x00,0x00},
142 {0x00,0x3E,0x41,0x41,0x41,0x22,0x00,0x00},
143 {0x00,0x7F,0x41,0x41,0x22,0x1C,0x00,0x00},
144 {0x00,0x7F,0x49,0x49,0x49,0x41,0x00,0x00},
145 {0x00,0x7F,0x09,0x09,0x09,0x01,0x00,0x00},
146 {0x00,0x3E,0x41,0x41,0x51,0x72,0x00,0x00},
147 {0x00,0x7F,0x08,0x08,0x08,0x7F,0x00,0x00},
148 {0x00,0x41,0x7F,0x41,0x00,0x00,0x00,0x00},
149 {0x00,0x20,0x40,0x41,0x3F,0x01,0x00,0x00},
150 {0x00,0x7F,0x08,0x14,0x22,0x41,0x00,0x00},
151 {0x00,0x7F,0x40,0x40,0x40,0x40,0x00,0x00},
152 {0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00,0x00},
153 {0x00,0x7F,0x04,0x08,0x10,0x7F,0x00,0x00},
154 {0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x00},
155 {0x00,0x7F,0x09,0x09,0x09,0x06,0x00,0x00},
156 {0x00,0x3E,0x41,0x51,0x21,0x5E,0x00,0x00},
157 {0x00,0x7F,0x09,0x19,0x29,0x46,0x00,0x00},
158 {0x00,0x26,0x49,0x49,0x49,0x32,0x00,0x00},
159 {0x00,0x01,0x01,0x7F,0x01,0x01,0x00,0x00},
160 {0x00,0x3F,0x40,0x40,0x40,0x3F,0x00,0x00},
161 {0x00,0x1F,0x20,0x40,0x20,0x1F,0x00,0x00},
162 {0x00,0x3F,0x40,0x38,0x40,0x3F,0x00,0x00},
163 {0x00,0x63,0x14,0x08,0x14,0x63,0x00,0x00},
164 {0x00,0x03,0x04,0x78,0x04,0x03,0x00,0x00},
165 {0x00,0x61,0x51,0x49,0x45,0x43,0x00,0x00},
166 {0x00,0x7F,0x41,0x41,0x00,0x00,0x00,0x00},
167 {0x00,0x02,0x04,0x08,0x10,0x20,0x00,0x00},
168 {0x00,0x41,0x41,0x7F,0x00,0x00,0x00,0x00},
169 {0x00,0x04,0x02,0x01,0x02,0x04,0x00,0x00},
170 {0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00},
171 {0x00,0x01,0x02,0x04,0x00,0x00,0x00,0x00},
172 {0x00,0x20,0x54,0x54,0x54,0x78,0x00,0x00},
173 {0x00,0x7F,0x48,0x44,0x44,0x38,0x00,0x00},
174 {0x00,0x38,0x44,0x44,0x28,0x00,0x00,0x00},
175 {0x00,0x38,0x44,0x44,0x48,0x7F,0x00,0x00},
176 {0x00,0x38,0x54,0x54,0x54,0x18,0x00,0x00},
177 {0x00,0x08,0x7E,0x09,0x02,0x00,0x00,0x00},
178 {0x00,0x18,0xA4,0xA4,0xA4,0x7C,0x00,0x00},
179 {0x00,0x7F,0x08,0x04,0x04,0x78,0x00,0x00},
180 {0x00,0x00,0x7D,0x00,0x00,0x00,0x00,0x00},
181 {0x00,0x80,0x84,0x7D,0x00,0x00,0x00,0x00},
182 {0x00,0x7F,0x10,0x28,0x44,0x00,0x00,0x00},
183 {0x00,0x41,0x7F,0x40,0x00,0x00,0x00,0x00},
184 {0x00,0x7C,0x04,0x18,0x04,0x78,0x00,0x00},
185 {0x00,0x7C,0x08,0x04,0x7C,0x00,0x00,0x00},
186 {0x00,0x38,0x44,0x44,0x38,0x00,0x00,0x00},
187 {0x00,0xFC,0x24,0x24,0x18,0x00,0x00,0x00},
188 {0x00,0x18,0x24,0x24,0xFC,0x00,0x00,0x00},
189 {0x00,0x00,0x7C,0x08,0x04,0x00,0x00,0x00},
190 {0x00,0x48,0x54,0x54,0x24,0x00,0x00,0x00},
191 {0x00,0x04,0x7F,0x44,0x00,0x00,0x00,0x00},
192 {0x00,0x3C,0x40,0x40,0x7C,0x00,0x00,0x00},
```

```

...l Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\data.h 4
193 {0x00,0x1C,0x20,0x40,0x20,0x1C,0x00,0x00},
194 {0x00,0x3C,0x40,0x30,0x40,0x3C,0x00,0x00},
195 {0x00,0x44,0x28,0x10,0x28,0x44,0x00,0x00},
196 {0x00,0x1C,0xA0,0xA0,0x7C,0x00,0x00,0x00},
197 {0x00,0x44,0x64,0x54,0x4C,0x44,0x00,0x00},
198 {0x00,0x08,0x36,0x41,0x00,0x00,0x00,0x00},
199 {0x00,0x00,0x7F,0x00,0x00,0x00,0x00,0x00},
200 {0x00,0x41,0x36,0x08,0x00,0x00,0x00,0x00},
201 {0x00,0x02,0x01,0x01,0x02,0x01,0x00,0x00},
202 {0x00,0x02,0x05,0x05,0x02,0x00,0x00,0x00}
203 };
204
205
206 /*
207 const uint8_t buffer[SSD1306_LCDHEIGHT * SSD1306_LCDWIDTH / 8] = {
208     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
209     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
210     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
211     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
212     0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
213     0x00, 0x80, 0x80, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
214     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
215     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
216     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
217     0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF8, 0xFC, 0xF8, 0xE0, 0x00, 0x00, 0x00, 0x00,
218     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80,
219     0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00,
220     #if (SSD1306_LCDHEIGHT * SSD1306_LCDWIDTH > 96*16){
221     0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00,
222     0x80, 0xFF, 0xFF, 0x80, 0x80, 0x00, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x80, 0x00, 0x80,
223     0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x8C, 0x8E, 0x84, 0x00, 0x00, 0x80,
224     0xF8, 0xF8, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
225     0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xF0, 0xE0, 0xE0, 0xC0,
226     0x00, 0xE0, 0xFC, 0xFE, 0xFF, 0xFF, 0xFF, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
227     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xFF, 0xC7, 0x01,
228     0x01, 0x01, 0x83, 0xFF, 0xFF, 0x00, 0x00, 0x7C, 0xFE, 0xC7, 0x01, 0x01, 0x01, 0x01, 0x83,
229     0xFF, 0xFF, 0x00, 0x38, 0xFE, 0xC7, 0x83, 0x01, 0x01, 0x01, 0x83, 0xC7, 0xFF, 0xFF, 0x00,
230     0x01, 0xFF, 0xFF, 0x01, 0x01, 0x00, 0xFF, 0xFF, 0x07, 0x01, 0x01, 0x01, 0x00, 0x00, 0x7F,
231     0x80, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x7F, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x01,
232     0xFF, 0xFF, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```


...	1	Studio\7.0\3UGERS-V2021\Samlet projekt\Version01\Version01\data.h	5
233		0x03, 0x0F, 0x3F, 0x7F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xE7, 0xC7, 0xC7, 0x8F,	↗
234		0x8F, 0x9F, 0xBF, 0xFF, 0xFF, 0xC3, 0xC0, 0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFC, 0xFC,	↗
235		0xFC, 0xFC, 0xFC, 0xFC, 0xFC, 0xF8, 0xF8, 0xF0, 0xF0, 0xE0, 0xC0, 0x00, 0x01, 0x03, 0x03,	↗
236		0x03, 0x03, 0x01, 0x03, 0x03, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x01,	↗
237		0x03, 0x01, 0x00, 0x00, 0x00, 0x01, 0x03, 0x03, 0x03, 0x03, 0x01, 0x01, 0x03, 0x03, 0x00,	↗
238		0x00, 0x03, 0x03, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
239		0x03, 0x03, 0x03, 0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x01, 0x03, 0x01, 0x00, 0x00, 0x00,	↗
240		0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
241		#if (SSD1306_LCDHEIGHT == 64){	
242		0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0xF0, 0xF9, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x1F,	↗
243		0x87, 0xC7, 0xF7, 0xFF, 0xFF, 0x1F, 0x1F, 0x3D, 0xFC, 0xF8, 0xF8, 0xF8, 0xF8, 0x7C, 0x7D,	↗
244		0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x3F, 0x0F, 0x07, 0x00, 0x30, 0x30, 0x00,	↗
245		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
246		0x00, 0x00, 0x00, 0x00, 0xFE, 0xFE, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
247		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xC0,	↗
248		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00,	↗
249		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
250		0x00, 0xC0, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7F, 0x3F,	↗
251		0x0F, 0x07, 0x1F, 0x7F, 0xFF, 0xFF, 0xF8, 0xF8, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xF8,	↗
252		0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xFE, 0x00,	↗
253		0x00, 0xFC, 0xFE, 0xFC, 0x0C, 0x06, 0x06, 0x0E, 0xFC, 0xF8, 0x00, 0x00, 0xF0, 0xF8, 0x1C,	↗
254		0x06, 0x06, 0x06, 0x0C, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0xFE, 0xFE, 0x00, 0x00, 0x00, 0x00,	↗
255		0xFE, 0xFC, 0x00, 0x18, 0x3C, 0x7E, 0x66, 0xE6, 0xCE, 0x84, 0x00, 0x00, 0x06, 0xFF, 0xFF,	↗
256		0x06, 0xFC, 0xFE, 0xFC, 0x0C, 0x06, 0x06, 0x06, 0x00, 0x00, 0xFE, 0xFE, 0x00, 0x00, 0xC0,	↗
257		0xFC, 0x4E, 0x46, 0x46, 0x46, 0x4E, 0x7C, 0x78, 0x40, 0x18, 0x3C, 0x76, 0xE6, 0xCE, 0xCC,	↗
258		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	↗
259		0x00, 0x00, 0x00, 0x00, 0x01, 0x07, 0x0F, 0x1F, 0x1F, 0x3F, 0x3F, 0x3F, 0x3F, 0x1F, 0x0F,	↗
260		0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00,	↗
261		0x00, 0x0F, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x03, 0x07, 0x0E,	↗
262		0x18, 0x18, 0x0C, 0x06, 0x0F, 0x0F, 0x0F, 0x00, 0x00, 0x01, 0x0F, 0x0E, 0x0C, 0x18, 0x0C,	↗
263		0x07, 0x01, 0x00, 0x04, 0x0E, 0x0C, 0x18, 0x0C, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x0F, 0x0F,	↗
264		0x00, 0x0F, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00,	↗
265		0x07, 0x0C, 0x0C, 0x18, 0x1C, 0x0C, 0x06, 0x06, 0x00, 0x04, 0x0E, 0x0C, 0x18, 0x0C, 0x0F,	↗

