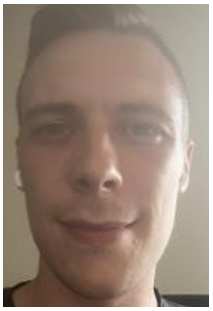
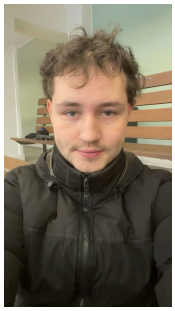
Metaldetektor projekt
Gruppe 1

34621
Electromagnetic sensors and digital signal processing
Danmarks Tekniske Universitet



Sebastian Sørensen,
s233986



Oliver Holm,
s233988

Bilal Alali,
s171678

January 22, 2026

Indholdsfortegnelse

1 Introduktion

Dette projekt omhandler udviklingen af en metaldetektor, herunder skal vi selv designe og implementere hhv. software og hardware. Software delen skrives i embedded C, på en Arduino AtMega2560, hvor data vi sampler fra spolerne skal behandles og outputtes til brugeren.

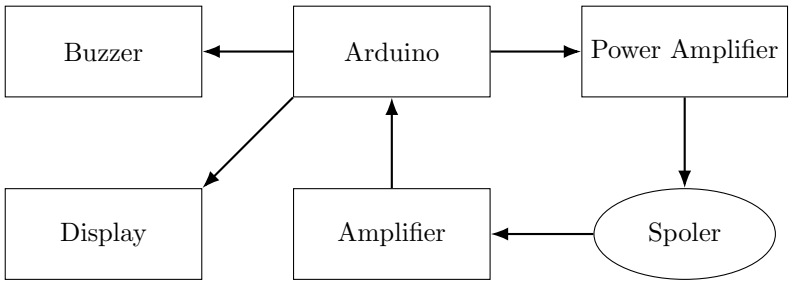
Hardware-delen af projektet omhandler primært design og implementeringen af sensoren til detektion af metaller, samt diverse kredsløb, der har til formål at drive spolen, og filtrere samt forstærke signalet. Systemet skal outputte en beregnet amplitude og fase til OLED-display, men vi har valgt også at implementere en buzzer til audio-feedback.

Herunder ses kravspecifikationen metaldetektor projektet har haft til formål at opnå.

Nr:	Navn:	Beskrivelse:	Prioritet
1	Amplitude/Fase detektion	Det skal være muligt at kunne detektere amplitude og fase i det reflekterede signal	1
2	Metal type	Skal kunne skelne jern fra kobber, messing og aluminium	1
3	Distance krav	Skal kunne detektere en jerngenstand (radius på ca. 15mm. Og en længde på 50mm.) i en dybde på 50mm. I fri luft	1
4	Strømforsyning	Hele metaldetektoren skal kunne køre på et 9 volts batteri (batteristørrelse E/6LR61) - kun et batteri.	1
5	Funktionstid	Metaldetektoren skal kunne køre minimum 100 minutter på batteriet og den ubelastede rest-spænding skal være større end 6 volt efter de 100 minutter	1
6a	Processor design	Kredsløbet skal være baseret på en Arduino	1
6b	Hardware design	Kredsløbene til at drive spolerne og forstærke det modtagne signal skal bygges med diskrete komponenter såsom: transistorer, operationsforstærkere, m.m.	1
7	Muligt software design	Brug Arduino'ens ADC og kontroller denne med timer interrupts	2
8a	Bruger interface display	Skal kunne udskrive amplitude og fase i et display	1

8b	Display udlæsningsstabilitet	Værdierne i displayet skal være læsbar for et flertal af brugere (spørg om hjælp)	1
8c	Forbedret display udlæsningsstabilitet	Der anvendes et FIR eller IIR filter til at lavpas-filtrere (midle) udlæsningen.	2
9a	Bruger interface primære knapper	Der skal være en start/stop knap	1
9b	Bruger intercace sekundære knapper	Der skal være en nulstillingsknap (fratrækker værdierne når detektoren ikke er i nærheden af metal)	1
10	Detektions princip	Very Low Frequency (VLF)	1
11	Samplingsfrekvens	8 kHz +/- 100Hz	1
12	Detektionsfrekvens	2 kHz +/- 100Hz	1
13	Metaldetektorudformning	3D-printet/trækonstruktion/plast	2
14	Spolernes bæreenhed	3D-printet spoleform /læserskåret plastik	2
15	Spoleudformning	Spolerne anvender en centreret udformning med sende-spolen yderst og feedback/modtager spolen inderst.	2
16	Modtager spolens selvinduktans	Modtager spolen skal have en selvinduktans på mindst 10 mHy.	1

2 Analyse



Figur 1: Overordnet moduldiagram af systemet

2.1 Analog

2.1.1 Design af spole

Ved sammenligning af de forskellige spoleudformninger (som er nævnt i "Coil Basics" fra kursets DTU Learn side), ligger vi mærke til følgende forskelle:
DD-spolen har bedre dækning til siderne, hvorimod den konsentriske er mere følsom under selve spolen og kan scanne dybere. Vi ser også at DD-spolen har en mere kompleks struktur, hvilket vil kræve mere arbejde i form af strengere krav til 3D-print. Det er desuden også nævnt i artiklen at det, ved anvendelse af den konsentriske form, er nemmere at udligne feltet fra TX-spolen i Bucking-spolen.
Grundet den mere simple opbygning - og bedre scannings dybde, vælger vi at anvende den konsentriske opbyging.

2.1.2 Signalbehandling

Ved signalfiltreringen ser vi aflæsningen af fasen som noget vi skal være særligt opmærksomme på, da det er denne som skal bruges til at skelne imellem metallerne.
Skin-effekten medfører både amplitudedæmpning og faseskift i det indtrængende felt. For en god leder er dæmpningskonstanten α og fasekonstanten β givet ved $\alpha = \beta = \frac{1}{\delta}$, således at faseskiftet ϕ over en indtrængningsdybde z kan beregnes ved $\phi = \frac{z}{\delta}$, hvor

$$\delta = \sqrt{\frac{2}{\omega\mu\sigma}}$$

Ved detektionsfrekvensen på $f_d = 2\text{kHz}$ (krav 12), og anvendelse af materialernes ledningsevner, forventer vi de relative størrelsesforhold mellem materialernes faseskift til at være

Table 2: Forventet relativt faseskift i materialerne, ved antagelse af $\mu = 1$

Metal	**Ledningsevne [S/m]**	**Faseskift** [°]
Kobber	$5.96 \cdot 10^7$	$75 - 85$
Messing	$1.5 \cdot 10^7$	$60 - 75$
Aluminium	$3.5 \cdot 10^7$	$55 - 70$

Dette er dog kun en approksimation, da faseskiftet er afhængigt af flere faktorer såsom temperatur, materialets geometri, samt afstanden mellem sensoren og materialet.
Da de 3 metaller som vi jf. kravspecifikationen skal kunne skelne imellem ligger relativt tæt i faseskift, er vi særligt opmærksomme på ikke at forstyrre fasen i signalbehandlingen.

2.2 Digital

2.2.1 Timing og sampling

Systemet bruger Timer1 til vores 2 frekvenser, hhv. sampling- /spole frekvens. Selve Timer1 generere de 8 kHz til samplingsfrekvensen og ved at toggle udgangen hver anden interrupt fås spolefrekvensen til 2 kHz ved

$$\frac{8\text{kHz}}{2 \cdot 2} = 2\text{kHz}$$

Ved at sample med 8 kHz kan de 2 kHz signal samples 4 gange per periode, hvilket tillader en simpel DFT implementering.

2.2.2 DFT

Da Arduinos hukommelse er begrænset, undgås den fulde FFT, i stedet benytter vi en hurtigere og mere passende algoritme. Her beregnes kun bin k = 16, udfra de 2 kHz ved Fs = 8 kHz og N = 64.
Dette betyder at vi kan spare CPU tid, fordi kun et frekvensbånd har betydning. Ved $k = 16$ og $N = 64$ kommer $\cos(2 \cdot \pi \cdot k \cdot \frac{n}{N})$ og $\sin(2 \cdot \pi \cdot k \cdot \frac{n}{N})$ til og skifte mellem værdierne “1, 0, -1, 0”. Ved at udnytte dette mønster undgås den tunge multiplikation, som er vigtigt, idet der modtages enorme mængder samples, som skal behandles.
Funktionen DFT_accum() samler hver sample ind i en real og imaginærdel ved brug af switch-casen på (n & 3). Når blokken afsluttes gemmes resultatet, og herefter genstartes processen.

2.2.3 Double buffer

For at undgå konflikt data konflikt mellem ISR og main benyttes et double buffer system, bestående af adc_bud[0] og adc_buf[1]. Bufferne skiftes automatisk når den ene er fuld, så den tomme kan begynde at samle data og den allerede

indsamlede data kan behandles. Et flag giver signal til main, når nye DFT data er klar til behandling.

2.2.4 DSP funktioner

DSP_fast_magnitude() beregner $|Z| = \sqrt{(\Re(z)^2 + \Im(z)^2)}$ ved brug af floating point sqrt(), som hentes fra biblioteket math.h. Denne tilgangsmåde blev valgt grundet dens præcision og enkelthed.
Vi opbyggede som alternativ en algoritme til heltals approksimation (se ukommenteret funktion i appendix DSP.c), men vurderede at den ikke ville være den optimale løsning trods større hastighed, dette skyldes mangel på præcision.
DSP_fast_atan2_deg() er funktionen til fase beregning. Den returnerer fase i grader fra -180 til 180 ved brug af atan2() ligeledes hentet fra math.h biblioteket. Fasen bruges til at skelne mellem metaltyper, hvilket indgår i opgavekravene. Her opbyggede vi ligeledes en algoritme til hastighed og optimering af hukommelses brug, dog kom vi til samme konklusion at atan2() havde bedre præcision.

2.2.5 Display og buzzer

Designvalg og begrundelse

Da kun en frekvens på 2 kHz er relevant, er DFT algoritmen betydeligt mere effektiv end at foretage en stor FFT. Denne løsning vil reducere CPU'ens belastning samt tillade realtidsbehandling på en 16 MHz ATmega2560.

Double buffer system

Sikrer at der kan samples parallelt med at main behandler den samplede data. Denne tilgang benyttes for at undgå timings problemer.

Glidende gennemsnit

Filtrerer støj fra ved at tage gennemsnittet af 32 blokke før data vises på OLED som sikre stabile aflæsninger. Alternativer findes til denne metode, men denne tilgang virkede hermed beholdes den.

Kalibrering

Ved at måle et sted uden metal (tom luft), kompenseres der for DC-offset og miljømæssige faktorer såsom temperaturskift. Dette er nødvendigt for at deltamålingerne er pålidelige. Ved at tage gennemsnittet af 32 baseline samples kommer det på bekostning af CPU'ens hastighed. Hvis der kun benyttes 1 blok i stedet, havde kalibreringen været betydelig hurtigere, dog mere upræcis, her vurderes at præcision var vigtigst.

2.2.6 Sleep

Da krav 5 i kravspecifikationen vedrører strømforbruget i systemet, er vi opmærksomme på at MCU'en kan konfigureres med sleep-funktioner for at forlænge batterilevetiden, hvis nødvendigt.

3 Design

3.1 Analogt design

3.1.1 Energiberegninger

Vi undersøgte os frem til at batteriet har en kapacitet på 550mAh. Da batteriet aflades fra 9V til 6V, har vi lavet udregningen:

$$E_{tot} = V \cdot I \cdot 3600 \iff E_{tot} = \frac{9V + 6V}{2} \cdot 0.55\text{A} \cdot 3600\text{s} = 14850\text{J}$$

Ved måling af strømforbruget finder vi at OLED-display og Arduinoen forbuger ≈ 70.5mA. Da vi skal kunne drive systemet i 100 minutter, kan vi derfor beregne

$$E_{MCU} = 5\text{V} \cdot 70.5 \cdot 10^{-3}\text{A} \cdot (100 \cdot 60)\text{s} = 2115\text{J}$$

Vi reserverer 10% af batteriet som buffer, for at være sikker på at overholde kravene. Resten af energien kan bruges til TX-spolen.

$$E_{TX} = E_{tot} \cdot (1 - 0.15) - E_{MCU} = 11250J$$

3.1.2 Spoleberegninger

TX-spole

Ved resonans kan spændingen gennem spolen approksimeres ved grundtonen i fourierrækken

$$a_n \approx 2\left(\frac{A}{n\pi}\right)\sin\left(\frac{n\pi}{2}\right) = 2\left(\frac{9\text{V}}{\pi}\right)\sin\left(\frac{\pi}{2}\right) = 5.73\text{V}$$

Hvorved RMS-spændingen kan bestemmes

$$V_{RMS} = \frac{a_n}{\sqrt{2}} = 4.05\text{V}$$

Derved kan vi nu beregne strømmen vi kan sende igennem spolen

$$I_{TX} = \frac{E_{TX}}{t \cdot V_{RMS}} = \frac{11250\text{J}}{(60 \cdot 100)\text{s} \cdot 4.05\text{V}} = 0.4628\text{A}$$

Radiussen på TX-spolen vælges til $r_{TX} = 10cm$, med $D_{TX} = Ø0.4$mm. Derved kan antallet af viklinger beregnes ved

$$R_{TX} = \frac{V_{RMS}}{I_{TX}} = 8.754\Omega$$

$$N_{TX} = \frac{R_{TX} \cdot \left(\frac{D_{TX}}{2}\right)^2}{2 \cdot r_{TX} \cdot \rho_{cu}} = \frac{8.754\Omega \cdot \left(\frac{0.4\text{mm}}{2}\right)^2}{2 \cdot 10\text{cm} \cdot 1.77 \cdot 10^{-8}\Omega \cdot m} = 98.9$$

hvilket afrundes til $N_{TX} = 99$.
Selvinduktansen i spolen findes ved

$$L = \frac{0.394 \cdot r^2 \cdot N^2}{9 \cdot r + 10A} = \frac{0.394 \cdot 10^2 \cdot 99^2}{9 \cdot 10 + 10} = 3.855\text{mH}$$

For at kunne lave Arduino'ens firkantsignal til en sinuskurve anvendes et resonanskredsløb, som har resonansfrekvens på 2kHz. Størrelsen af kondensatoren findes ved at løse ligningen

$$f = \frac{1}{2\pi\sqrt{LC}} \iff 2\text{kHz} = \frac{1}{2\pi\sqrt{3.855\text{mH} \cdot C}} \implies C = 1.642\mu\text{F} \qquad (1)$$

Magnetfeltet fra TX-spolen beregnes, da bucking-spolen skal udligne B-feltet fra TX-spolen, for på den måde at tilsikre at vi kun måler på magnetfeltet fra metallet i detektoren.

$$|B| = \frac{\mu_0 I_{TX} N_{TX}}{2r_{TX}} = \frac{4\pi \cdot 10^{-7}\,\frac{\text{H}}{\text{m}} \cdot 0.4628\text{A} \cdot 99}{2 \cdot 10\text{cm}} = 287.6\mu\text{T}$$

Bucking-spole

Derved kan Bucking-spolen beregnes. Vi vælger en radius på $r_{Buck} = 5\text{cm}$, og antallet af viklinger beregnes ved

$$|B| = \frac{\mu_0 \cdot I_{TX} \cdot N_{Buck}}{2 \cdot r_{Buck}} \iff 287.6\mu\text{T} = \frac{4\pi \cdot 10^{-7} \cdot 0.4628\text{A} \cdot N_{Buck}}{2 \cdot 5\text{cm}} \implies N_{Buck} = 49.458$$

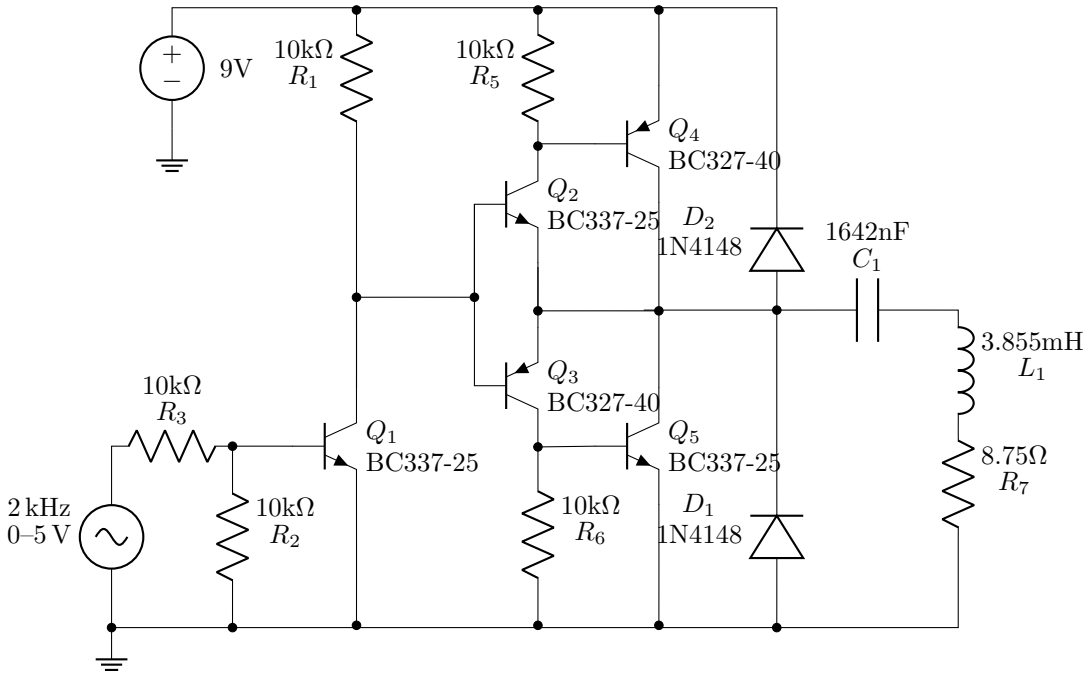
hvilket afrundes til $N_{Buck} = 50$.

RX-spole

For at opnå en god følsomhed på sensoren, er vi opmærksomme på at induktansen i RX-spolen skal være minimum $L_{RX} = 10\text{mH}$. Derved beregnes antallet af viklinger ved

$$10 \cdot 10^3 = \frac{0.394 \cdot r_{RX}^2 \cdot N_{RX}^2}{9r_{RX} + 10A_{RX}} = \frac{0.394 \cdot 5^2 \cdot N_{RX}^2}{9 \cdot 5 + 10} \implies N_{RX} = 236$$

3.1.3 Power Amplifier

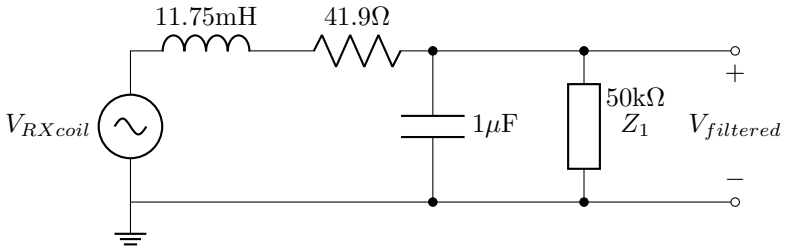


Figur 2: Power amplifier kredsløb til at øge strømmen gennem TX-spolen.

Ovenfor ses vores design af effektforstærkeren, som bruges til at forstærke signalet fra Arduino'en til at kunne drive TX-spolen. Kredsløbet virker ved:

- Firkantsignalet sendes ind i forstærkeren gennem en modstand som beskytter Arduino'en og styrer strømmen ind i transistoren Q_1.
- Transistor-konfigurationen fungerer som en forstærker som via. en styrestrøm kontrollerer en større strøm fra batteriforsyningen (9V batteriet), som driver TX-spolen.
- TX-spolen (L_1) genererer et magnetfelt ved resonansfrekvensen
- Kondensatoren (C_1) er beregnet (Ligning 1) således at der opstår resonans ved detektionsfrekvensen på 2kHz
- Modstanden R_7 repræsenterer den ohmske modstand i TX-spolen.
- Dioderne D_1 og D_2 beskytter kredsløbet fra den induserede spænding i spolen.

3.1.4 Filtrering af spolesignal



Figur 3: Kredsløb til at filtrere støj væk inden signalet bliver forstærket (som vist i Figur 5). Modstanden på 50kΩ repræsenterer indgangsimpedansen i operationsforstærkerkredsløbet.

Som det ses i figuren, er $V_{filtered} = V_{Z_1}$. Filterets overføringsfunktion kan opstilles ved almindelig spændingsdeling, dvs.

$$H(\omega) = \frac{Z_1 \cdot \frac{1}{j\omega C}}{j\omega L + R_1 + Z_1 \cdot \frac{1}{j\omega C}}$$

hvorefter fasen kan analyseres ved

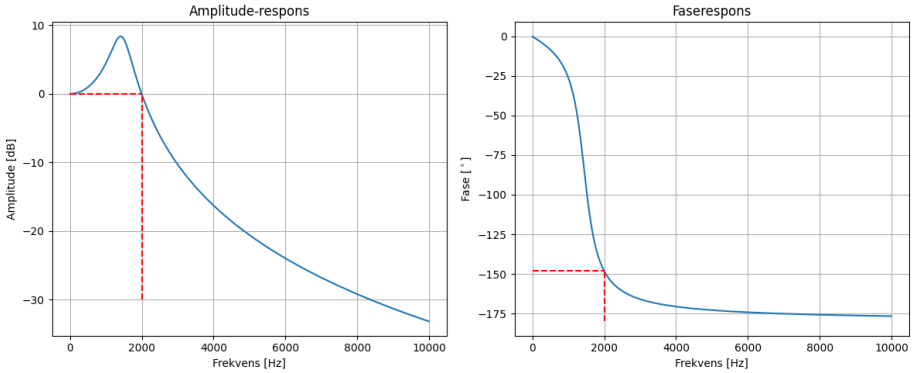
$$\phi(\omega) = \tan^{-1}\left(\frac{\Im\{H\}}{\Re\{H\}}\right)$$

og amplitude responset ved

$$H_{dB} = 20\log 10(H)$$

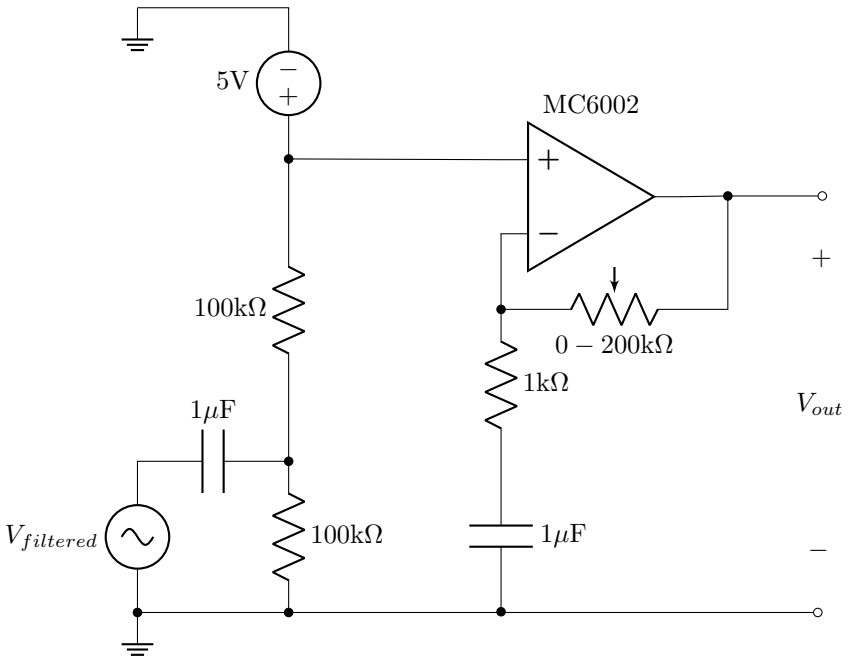
Ifølge Shannon's sampling teorem ved vi at $F_{max} < \frac{F_s}{2}$ skal være overholdt for at forhindre aliaseringer. Det eneste signal som vi ønsker at få igennem, ligger $\approx$ 2kHz, og derfor har vi valgt at alle frekvenser over 2kHz skal attenueres. Designet af RX-filteret er en balancegang imellem at dæmpe amplituden af uønskede frekvenser, mens vi bibeholder en rimeligt præcis faserespons ved 2kHz, for at kunne skelne imellem forskellige metaller.

Figur 4: RX-filterets fase- og magnitude respons, med markering af 2kHz



Da metaldetektorens følsomhed overfor frekvenser afhænger af bl.a temperatur, har vi valgt at designe filteret til at opnå mindst muligt forstyrrelse omkring de 2kHz, samtidigt med vi garanterer at der ikke opstår aliaseringer (Samplingteoremet er overholdt). Som det ses i plottet, ændrer fasen sig ikke voldsomt ved signalfrekvensen, og selv hvis frekvensen skrider ± 100Hz går signalstyrken relativt uberørt igennem.

3.1.5 Forstærkning af spolesignalet

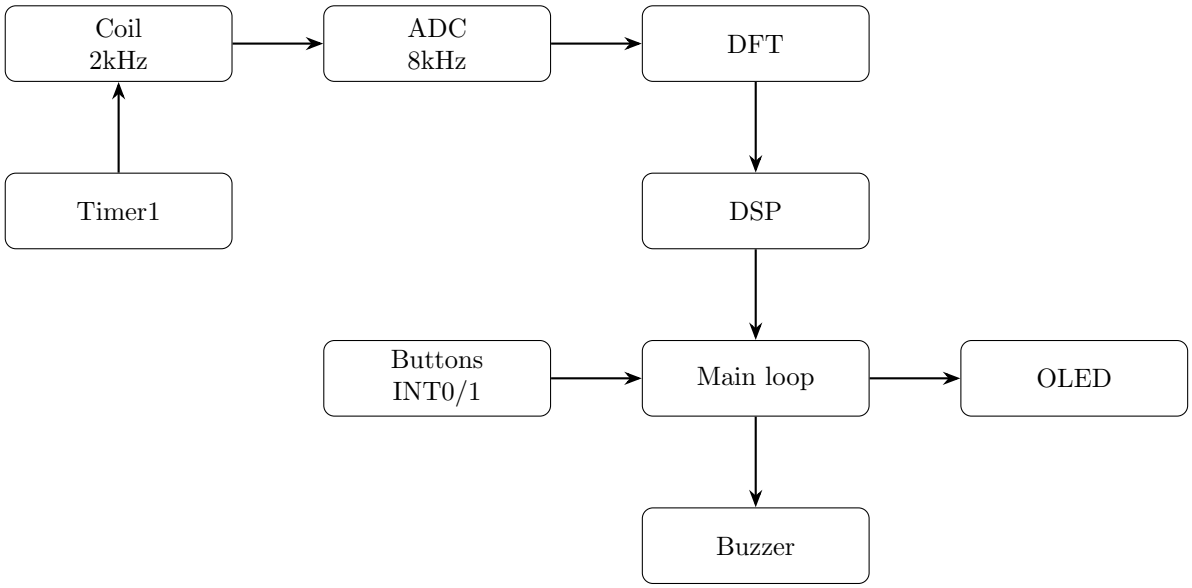


Figur 5: Kredsløb til at forstærke den filtrerede spænding fra RX-spolen op, inden det læses ind i MCU ADC'en. $V_{filtered}$ er outputtet fra RX-filteret.

Da spændingen fra RX-spolen ikke er højt nok til at vi kan aflæse det på MCU'en, bruger vi dette operationsforstærker kredsløb, til at trække spændingen op til $\approx$ 5V, for på den måde at anvende flest mulige bits i ADC'en og opnå bedst mulig opløsning. Vi anvender et potentiometer som feedback-modstand, for på den måde at kunne forstærke et mindre signal op ved behov, så følsomheden på detektoren kan indstilles.

3.2 Digitalt design

3.2.1 Moduldiagram og introduktion



Figur 6: Overordnet moduldiagram af systemet

3.2.2 Timer1

1. Funktion: Generer en samplingsfrekvens på 8 kHz (Fs), og en spolefrekvens på 2 kHz.

2. Designvalg Timer1 opstilles i CTC-mode med en prescaler på 8. Sammenligningsværdien beregnes ved: $OCR1A = \frac{F_{CPU}}{prescaler \cdot F_s} - 1 = \frac{16\text{MHz}}{8 \cdot 8\text{kHz}} - 1 = 249$ Spolefrekvensen sættes ved at toggle PB5 hver 2. interrupt: $\frac{F_s}{2 \cdot 2} = 2\text{kHz}$

3. Begrundelse Krav: Metaldetektor med 2 kHz drivfrekvens og passende samplerate til fasebestemmelse.

   Alternativ: Separat timer til spole og ADC. Denne idé blev droppet, idet en enkel timer med en divider er tilstrækkeligt, det sikrer også synkronisering af ADC og spole, derudover undgås en potentiel konflikt mellem de 2 ISR også.

Register	Værdi	Funktion
TCCR1A	0x00	Normal port operation
TCCR1B	(1 << WGM12) — (1 << CS11)	CTC mode, prescaler 8
OCR1A	249	8 kHz compare match
TIMSK1	(1 << OCIE1A)	Interrupt enable (ISR)

3.2.3 ADC

1. Funktion: Det analoge signal konverteres til digitale samples ved 8 kHz.

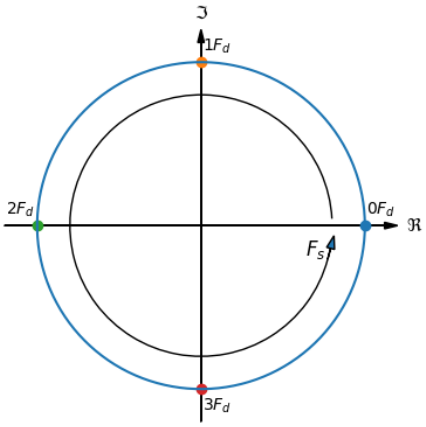
2. Designvalg ADC sættes med AVcc som reference, og en prescaler på 64. Denne opstilling giver en ADC clock på 250 kHz: $F_{clock} = \frac{F_{CPU}}{prescaler} = \frac{16\text{MHz}}{64} = 250\text{kHz}$ En ADC konvertering tager $13 \cdot f_c = 52\mu s$, dette ligger indenfor $125\mu s$ perioden mellem samples, hvor konverteringstiden er givet ved $\delta(t) = \frac{13}{250\text{kHz}} = 52\mu s$

3. Begrundelse Krav: Passende ADC opløsning og hastighed til at kunne sample et 2kHz signal præcist (fra spolen).

   Alternativ: Prescaler på 128, hvilket vil give 125 kHz. Denne konfiguration giver et mindre støjet signal, men en konverteringshastighed på $104\mu s$ som vurderes at være for tæt på sampletiden, dermed benyttes prescaleren på 64.

Register	Værdi	Funktion
ADMUX	(1 << REFS0)	ADC0, Avcc ref
ADCSRA	(1 << ADEN) — (1 <<ADIE) — (1 << ADPS2) — (1 << ADPS1)	ADC Enable, ISR Enable, prescaler 64
DIDR0	(1 << ADC0D)	Disable digital input

3.2.4 DFT



Figur 7: Grafisk illustration over princippet bag DFT-algoritmen. Ved at vælge $F_d = \frac{F_s}{4}$, undgår vi at lave komplicerede komplekse beregninger.

1. Funktion: Beregner amplitude og fase ved 2 kHz fra de indsamlede samples.

2. Designvalg En DFT algoritme opstilles ud fra k, således: $k = \frac{f \cdot N}{Fs} = \frac{2000*64}{8000} = 16$. Dette udnyttes idet k følger $\cos(2\pi\frac{k \cdot n}{N})$ og $\sin(2\pi\frac{k \cdot n}{N})$ gennem "1,0,-1,0". Hermed kan multiplikationer hvilket spare CPU'en en del.

n 4	mod	cos(pi*n/2)	Operation
0		cos = 1, sin = 0	re += sample
1		cos = 0, sin = 1	im -= sample
2		cos = -1, sin = 0	re -= sample
3		cos = 0, sin = -1	im += sample

3. Begrundelse Krav: Realtids bestemmelse af fase ved 2 kHz på MCU.

   Alternativ: En fuld FFT på 64 punkter, men det ville kræve mange komplekse multiplikationer per blok, hvilket ikke ønskes, idet den nærmest konstant skal være i gang, det vil betyde en alt for stor reduktion i CPU-tid.

4. Ingeniørmæssigt bidrag: Genkendelse af den specifikke frekvensrelation

$\frac{Fs}{f} = 4$ til at fjerne trigonomiske beregninger er absolut nødvendig optimering, der muliggør realtidsbehandling.

3.2.5 Double buffer

1. Funktion: Sørger for kontinuerlig dataflow mellem ISR og main.

2. Designvalg buffere adc_buf[0] og adc_buf[1] skiftes automatisk mellem aktivt at sample ved blok afslutning. Flags dft_ready[b] og adc_ready[b] signalerer til main der, med FSM, styre programmets dataflow.

   Begrundelse Krav: Kontinuerlig sampling uden at miste data.

3. Alternativ: En enkelt buffer med et stop-and-go princip vil kunne give huller i samplingen som potentielt kan misse korte metal detektioner. Alternativt er en ringbuffer en løsning, dog mere kompleks og muligvis ville data kunne overskrives hvilket igen leder til missede metaldetektorer.

3.2.6 DSP

1. Amplitude Implementation: abs(z)=sqrt(re*re+im*im) med floating point sqrt() fra math.h.

   Alternativ: En heltals approksimations algoritme: abs(Z) = max + 0.375 * min. Dette blev fravalgt, idet præcision prioriteres frem for hastighed og sqrt() kaldes kun 1 gang hver 8ms (per 64 samples), altså er hastighedsreduktionen ikke kritisk.

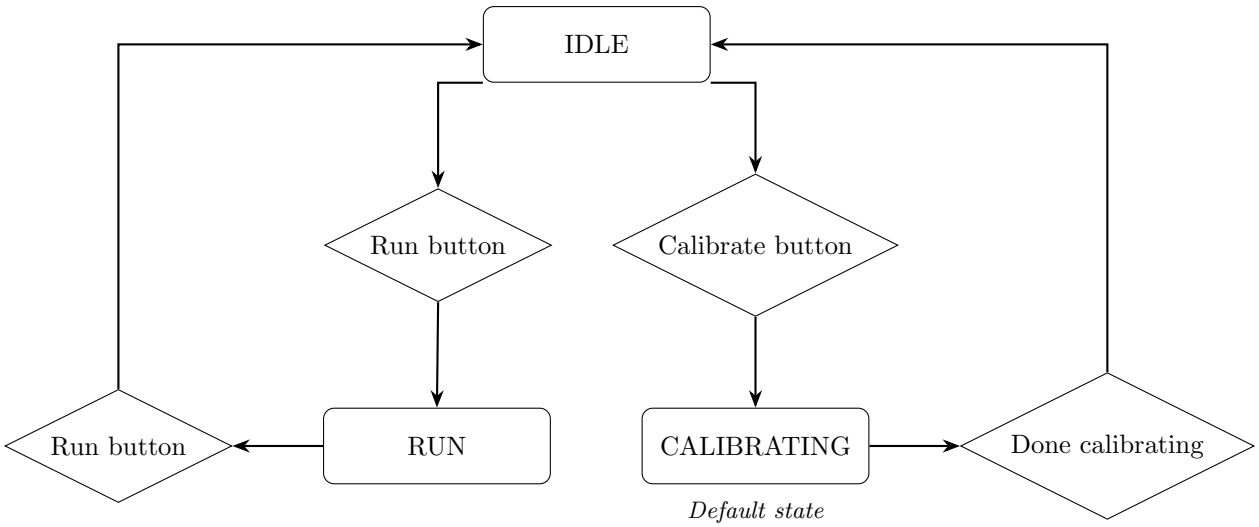
2. Fase Implementation: fase = atan2(im,re) * 180/pi, returner grader i intervallet [-180, 180].

   Alternativ: Atan2 approksimations algoritme, men fravalgt da math.h bibliotekets atan2() er hurtig nok og atan2() yder bedst præcision.

3. Glidende gennemsnit Implementation: 32-sample circulær buffer for amplituden og fasen. Gennemsnittet beregnes ved summation over hele bufferen.

   Tidsrespons: 32 blokke * 8ms/blok = 256 ms for fuld beregning af gennemsnit. Antallet samples blev valgt efter flere test, hvori 32 havde bedst støjfiltrering i forhold til responstid.

3.2.7 Tilstandsmaskine



Figur 8: Flowchart af tilstandsmaskinen

FRA	TIL	Betingelse	Handlin
CALIBRATING	IDLE	NaN	Gem ba
IDLE	RUNNING	NaN	Start sa
RUNNING	IDLE	NaN	Stop sa
IDLE/RUNNING	CALIBRATING	NaN	Nulstil

Softwares dataflow styres i main af en FSM bestående af 3 tilstande, herunder: CALIBRATING, IDLE, RUNNING. Kalibrerings tilstanden indsamler 32 DFT blokke (uden metal), hvor den beregner et gennemsnit af disse som bruges til et nulpunkt.
Denne tilstand styres af en interrupt baseret knap, som tillader adgang til CALIBRATING efter behov, da temperaturskift kan påvirke metaldetektoren er det nødvendigt at kunne danne et nyt nulpunkt.
IDLE tilstanden bruges som en homescreen. Herinde er sampling deaktiveret, altså bruges der betydelig mindre strøm hvilket ligeledes er nødvendigt hvis man ikke aktivt søger metal. Ved opstart af metaldetektoren tilgås IDLE efter en automatisk kalibrering, herfra kan man enten genkalibrere eller gå til RUNNING ved brug af interrupt styret knapper.
RUNNING tilstanden er her metaldetektoren aktivt søger metal, beregner delta

ud fra det kalibrerede nulpunkt, anvender 32-sample glidende gennemsnit til at undgå alt for hurtigere skift i amplitude og fase på OLED display.

3.2.8 Buzzer

1. Funktion: Generer akustisk feedback hvor frekvens indikerer metaltype og lydstyrke indikerer signalstyrke.
2. Designvalg Timer2 opstilles i fast PWM-mode med OCR2A som TOP og OCR2B som duty cycle. PH6 på Arduino ATMega2560 sættes på OC2B som output pin. Frekvens beregnes ved:

   f_out = F_CPU / (prescaler * (1 + OCR2A) = 16 MHz / (64 * (1 + OCR2A)

Register	Værdi	Funktion
TCCR2A	(1 << WGM21) — (1 << WGM20) — (1 << COM2B1)	Fast PWM, OC2B
TCCR2B	(1 << WGM22) — (1 << CS22)	TOP = OCR2A, prescaler 64
OCR2B	0-255 (styret af amplitude)	Volumen

3.2.9 Knap input

1. Funktion: Styre brugerinput på knapperne “run” og “pwr”. Designvalg Knapperne er forbundet til PE4 og PE5 med interne pull-ups. Falling edge trigger starter ISR, som venter 50 ms derefter godkender, om knappen stadig er trykket.
2. Begrundelse Alternativ: Hardware debounce med et RC-filter, ville spare CPU-tid men ligeledes kræve ekstra komponenter, altså er software debounce et valg for simplicitet og fleksibilitet, dette tillades idet software løsningen virker uden problemer.
3. Opsummering af designvalg

Komponen	Valgt løsning	Primær fordel
DFT	algortime for ”1,0,-1,0”	Undgår multiplikationer
Buffering	Double buffer	Ingen datatab
Amp/Fase	Amp() floating point, atan2()	Præcision
Filter	32 tap glidnde gennemsnit	Støjreduktion
Debounce	Software, 50 ms delay	Ingen ekstra hardware

4 Implementering og test

5 Konklusion

6 Ansvarsområder

Fælles

- Vikling af spoler

Bilal

- 3D-print af spole
- Power Amplifier
- Spoleberegninger
- Energiberegninger

Sebastian

- C-programmering
- Test og simulation af C-program
- RX-filter
- Grafik til rapport (plots, kredsløbstegninger,indskrivning af rapport)

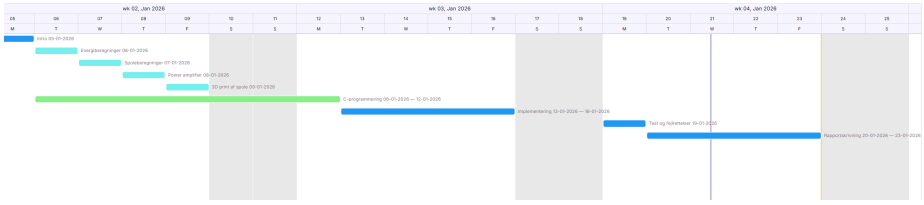
Oliver

- C-programmering
- Test og simulation af C-program
- Implementering af kredsløb

7 Appendix

7.1 Gantt kort

Figur 9: Her ses arbejdsfordelingen. Mørkeblå = Fælles,
Grøn=Sebastian & Oliver, Turkis = Bilal



7.2 Mødereferater

Møde dag 2

- Bilal starter på energiberegninger, samt opstart på beregninger vedrørende spolerne.
- Sebastian og Oliver foretager målinger på Arduino, for bedre at kunne estimere strømforbruget til energiberegningerne. Derefter starter de op på C-programmering.

Møde dag 3

- Sebastian og Oliver fortsætter med C-programmering, herunder moduler til DFT, DSP og tilstandsmaskine, samt generel struktur på programmet.
- Bilal fortsætter med at arbejde på spolen, og får startet på at lave en prototype til Power amplifieren.

Møde dag 5

- Bilal fortsætter med simulation af Power Amplifier, samt 3D tegning af spolen.
- Sebastian og Oliver fortsætter med C-programmering, herunder moduler til PWM, OLED-display, ADC. Efterfølgende påbegyndes modul til buzzer (til lydindikation af fase/amplitude), samt opsætning af sleep-funktion i Arduino for strømbesparelse. Desuden laves der et python-script til simulering af data som ligges i flashhukommelse på Atmega2560'eren, og det kontrolleres om DFT-algoritmen samt display og DSP virker som ønsket, ved sammenligning mellem Atmega2560 output og resultatet fra Python-scriptet.

Møde dag 6

Planen for idag er at vi skal have testet C koden for sig, samt testet Power Amplifier for sig. Hvis begge ting virker enkeltvis, sættes de sammen og testes. Efterfølgende skal vi vikle spolerne, samt have opdateret C-koden til at kunne skelne imellem metaller.

Møde dag 7

Idag har vi fælles foretaget målinger på spolen, samt viklet buck-spolen tilstrækkeligt ud, så vi får udlignet det uønskede felt. Samt fælles kontrol af Power Amplifier. Alt er testet OK, således at der imorgen kan påbegyndes filtrering og forstærkning af spolens output, så vi snarest muligt kan sætte de analoge og digitale dele sammen, og teste det samlede system.

Møde dag 8

- Sebastian og Oliver laver et modul i C koden til sleep-funktion for strømbesparelse. Herefter forbereder de operationsforstærker kredsløbet til imorgen.
- Bilal tegner 3D printet til stangen til metaldetektoren.

7.3 Kodebilag