

# Trådløs EKG måling

Gruppe 28, Kandidatnummer: 10025 & 10008

**Abstract**—I dette prosjektet skal studenter ved NTNU designe en EKG-måler. Kretsen skal først simuleres i LT-spice, for så å konstrueres på et breadboard. Studentene skal deretter designe en PCB for å foreta EKG-målingene. Denne skal loddas og testes. Testresultatene viste at EKG måleren fungerte, men den trengte digital filtrering for å fungere optimalt, da det oppstod mye bakgrunnstøy.

**Keywords**—EKG, Avanserte Sensorsystem, Simulering, Kretsdesign, LabVIEW.

## I. INTRODUCTION

I dagens samfunn er vi omgitt av sensorer som et bindeledd mellom mennesket og maskin. Sensorene henter inn måledata fra omgivelsene sine og danner elektriske signaler som kan tolkes av maskiner for å gjøre alt fra å åpne dører til å styre et fly på autopilot. Spesielt i systemer som det sistnevnte er det ekstremt viktig med nøyaktige og pålitelige signaler. De rå signalene fra sensoren må ofte behandles med for eksempel forsterkning eller filtrering før den er brukbar. I denne oppgaven skal det konstrueres en krets for å måle EKG.

## II. BAKGRUNN - TEORETISK GRUNNLAG

### A. EKG

EKG, eller electrocardiography, måler de elektriske pulsene til hjertet, for å sjekke tilstanden til hjertet. Dette blir ofte anvendt innen medisin, for å sjekke at hjerterytmen er som den skal. En EKG måling kan gi informasjon man ellers ikke kan finne, som uregelmessig hjerterevirksomhet, ledningsforstyrrelser i hjertet og oksygenmangel som følge av blodpropp. EKG måles normalt mens pasienten er helt rolig. En EKG krets består hovedsakelig av 5 deler: Inngang, Preamplifier, High-pass filter, Isolation Amplifier og Utgangen

### B. Elektronisk støy

Elektronisk støy er en uønsket forstyrrelse i et elektronisk signal. Støy generert av elektroniske komponenter varierer veldig, da det kan stamme fra flere ulike kilder. Noen typer støy kan være: Crosstalk, kosmisk støy, burst støy og termisk støy.

### C. LTspice

LTspice er et simuleringsprogram for analoge elektriske kretser. Programmet gjør det mulig å konstruere, simulere, probe og analysere alle deler av en krets, noe som kan være veldig nyttig før en etter hvert går videre med designet sitt for å bestille deler osv. I tillegg til en krets-editor, Tilbyr LTspice en innebygd waveform diagram.

### D. Ekvivalent Støybandbredde

Ekvivalent støybandbredde, eller EKBW, er bredden til et perfekt rektangel, som tilatter like mye gjennomtrengning som det samlede filteret. Se figur 3.

### E. Operasjonsforsterker

Operasjonsforsterker, eller Op-Amp, er en elektronisk forsterker som brukes i analoge elektroniske systemer. Den brukes sammen med passive komponenter, og kan da forsterke et signal eller utføre vanlige matematiske funksjoner på et eller flere signal.

### F. Instrumenteringsforsterker

Instrumenteringsforsterker, eller In-Amp, er en type differensialforsterker som er utstyrt med inngangs buffer forsterkere. Dette eliminerer nødvendigheten for å matche impedansene til inngangen, noe som gjør forsterkeren godt egnet for måleutstyr og testutstyr. Dette fører også til at forsterkeren har lite støy, lavt DC offset, lav drift, og høy open-loop gain. Instrumenteringsforsterkere brukes i system hvor både høy nøyaktighet og stabilitet er nødvendig.

### G. Metning

Metning vil si full konduktivitet i en halvleder. En Op-Amp vil ha metning når spenningen som ifølge kretsen skulle ha vært på utgangen går utenom det mulige operasjonsområdet. Op-Ampen vil sende ut det maksimale eller minimale mulige spenningsnivået istedenfor. Operasjonsområdet er bestemt av suppliedspennningen til op-ampen. Ideelle op-amper vil derfor ikke oppleve metning.

### H. Common Mode Rejection Ratio

CMRR, eller Common Mode Rejection Ratio, beskriver hvor godt en forsterkerkrets klarer å nulle ut Common Mode signaler. Common mode signaler er signaler som oppstår på begge inngangene, samtidig og i fase. En ideell differensialforsterker vil ha uendelig høy CMRR, dette skjer ikke i praksis.

### I. ESP32

Mikrokontrollere brukes i systemer der noe skal styres eller overvåkes. ESP32 er en mikrokontroller med WiFi funksjonalitet. En ESP32 integrert i et system gjør det mulig å sette opp trådløs kommunikasjon med andre enheter.

### J. Webserver

På hardware siden er en webserver typisk en datamaskin som lagrer informasjon og kode for en nettside. Webserverens oppgave er å gjøre dette tilgjengelig for andre enheter på nettverket.

### K. Websocket

Websocket API er en teknologi som åpner for to-veis kommunikasjons sesjon mellom nettleser og server. En Websocket gjør det mulig å motta data fra server uten å aktivt sjekke etter oppdateringer.

## III. METODE - FREMGANGSMÅTE

Det meste av denne oppgaven er basert på innhold og informasjon som ble gitt ut i NTNU faget IELET3109: Avanserte Sensorsystemer. Foreleser i dette faget delte ut en eksempel krets, som studentene har brukt som referanse-design. Denne kretsen benytter en TL071 operasjonsforsterker, og en INA126 instrumenteringsforsterker. Kretsen er simulert i LTspice, hvor formålet er å sjekke ulike karakteristikk, som noise, forsterkningsrespons, båndbredde og EKBW.

### A. Kretsens oppbygging

Kretsen kan tenkes er delt opp i flere deler. Inngang. Instrumenteringsforsterker, Høypassfilter, Opamp og utgang.

1) *Inngang*: Inngangen på kretsen er en av de enklere delene av kretsen. Den består av motstander og TVS dioder. Diodene er av typen INA1004. Motstandene er ikke spesifiserte. Diodene er for å beskytte mot overspenning, i form av ESD. Diodene gjør at spenningstoppen fra ESD, blir minimert, og dette beskytter de videre delene av kretsen. Motstandene er der for å passe på at inngangsstrømmen ikke blir for høy.

2) *Instrumenterings forsterker*: De beskyttede signalene går så videre til In-Ampen. In-Ampen tar signalene og forsterker de. Signalene blir da enklere å håndtere for de videre delene av kretsen.

3) *Høypassfilter*: Det forsterkede signalet går da gjennom et passivt høypassfilter. Høypassfilteret sørger for å filtrere bort uønsket støy som kan forekomme av eksterne kilder. Knekkfrekvens til et høypassfilter er gitt ved:

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

4) *Operasjons forsterker*: Op-Ampen forsterker det filtrerte signalet nok en gang. Denne op-ampen fungerer også som et ikke-inverterende lavpassfilter, som betyr at fasen til signalet ikke blir forskøvet.

5) *Utgang*: Signalet går til slutt gjennom et lavpassfilter, som kommer fra den inverterende Op-Ampen. Dette for å glatte ut signalet, og gjøre det enklere å tyde. Knekkfrekvensen til et lavpassfilter er igjen gitt ved:

$$f_c = \frac{1}{2 \cdot \pi \cdot R_f \cdot C_f}$$

### B. Dimensjonering av In-Amp

In-Ampen burde ha en gain på ca 10V/V. EKG signalet som normalt ligger mellom 1mV og 5mV, ligger da mellom 10mV og 50mV. In-Amp'en har dedikert plasing for  $R_g$ :

$$G = 5 + \frac{80k\Omega}{R_g} = 10 \Rightarrow R_g = \frac{80k\Omega}{G - 5} = \frac{80k\Omega}{10 - 5} = 16k\Omega$$

### C. Dimensjonering av Høypassfilteret

Det passive høypassfilteret er dimensjonert til å ha en knekkfrekvens på 33.9mHz. Dette er gitt ved

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C} = \frac{1}{2 \cdot \pi \cdot 1M\Omega \cdot 4.7\mu F} = 33.86mHz$$

### D. Dimensjonering av In-Amp

Op-Ampen burde ha en gain på ca 100V/V. Ettersom Op-Ampen er inverterende, blir fasen snudd med 180°. Formelen for gain hos en inverterende forsterker er:

$$G = -\frac{R_2}{R_1}$$

I denne kretsen er det valgt  $R_1 = 1k\Omega$  og  $R_2 = 100k\Omega$ . Dette resulterer i en gain på 100 V/V

### E. Altium

For å designe selve PCB'en brukt for dette prosjektet, ble Altium utnyttet. Dette er et skjematikk og PCB utlegg programvare.

1) *Skjematikk*: Skjematikken er delt opp i flere sub-skjematikker. Dette er for å gjøre skjematikken mer forståelig da den følger blokkdiagrammet som ble lagt ved i emnet, se figur 14 for blokkdiagram. Blokkene i skjematikken er flyttet litt på, men har samme navn som de i figur 14. Noen av de individuelle blokkene inneholder ganske enkle komponenter som ESP32, ADS1115, Power-Boost 1000 og DCDC-15V. Disse blokkene inneholder kun enkeltkomponenter, eller headers ment for komponentene. Den mest avanserte blokken, er EKG-interface. Denne inneholder bolken av selve EKG måleren, altså IN-amp og OP-amp kretsen. Det var mulighet for å bestille komponenter fra Digikey, men ettersom markedet er ustabilt, ble det benyttet komponenter NTNU allerede hadde på lager.

2) *Utlegg*: Utleget ble laget med bærbarhet i fokus. Utleget skulle bli så lite som mulig, slik at EKG måleren enkelt skulle kunne fraktes, og var enkel å handtere. For at dette skal være mulig, er det ikke mye klaring mellom komponentene. Ettersom EKG kretsen i seg selv er en sensor, er det kritisk med signal integritet. Det er derfor tatt hensyn til hvor de forskjellige signalbærende tracene er plassert. Utleget og skjematikken har også to individuelle jordbaner, analog og digital jord. I dette tilfellet kommer analog jord fra 3.5mm jack inngangen. Den blir ledet direkte til ADC, for minst mulig forstyrrelse.

### F. LabVIEW

For EKG målinger med myDAQ ble det tatt inspirasjon fra et labVIEW VI fra lab 4 lesson 2. Vårt GUI er vist i figur 23

1) *Raw Data*: Den første waveform charten har som oppgave å vise de rå datapunktene som blir målt av myDAQ.

2) *Raw Collected Data*: Raw collected data er raw data samlet opp i en graf. Grafen viser alle datapunkter de siste antall sekunder beskrevet i Samples[s]. Dette gjøres ved å konkatinerer sammen verdien fra forrige iterasjon med verdien fra den mest nylige, i et en-dimensjonalt array. Dermed slettes alle verdier som er eldre en antall sekunder delt på frekvensen på målingene. Altså antall målinger per sekund ganger antall sekunder.

3) *Filtered Data*: Filtered data tar Raw Collected Data og filterer den i et forsøk på å fjerne så mye støy som mulig og få frem hjerterytmen som typisk ligger på mellom 60 og 100 bpm. I et subVI, "filter waveform", tas det inn Raw Collected Data og et kontroll signal (Additional Signals to remove[s]).

For å filtrere ut støy brukes det en "unbundle" som tar ut kun datapunktene av signalet i et array i dette tilfellet. Det er disse som kan manipuleres og filtreres og dermed bundles sammen igjen med dataen fra det originale signalet. Slik kan vi endre på datapunkt verdiene uten å ta hensyn til eller miste noe metadata fra signalet. Etter unbundle blokken sendes arrayet med punkter inn i et Butterworth båndpass-filter. Filter-blokken tar også inn frekvensen på punktene. Dette får vi ved å ta dt invers. Dt får vi fra Get wfm blokken som tar inn hele waveformen og sender ut det som trengs av data, i dette tilfellet er det Y verdiene som skal filtreres og dt som vi bruker til å regne ut frekvensen og sende inn i filteret. For å fullføre båndpass-filteret trenger vi cutoff-frekvenser for båndet. Vi regner med at en realistisk maks bpm er 180 for våre ekg målinger, og filtrerer derfor ut verdier over 180bpm/60sek eller 3 hjerteslag per sekund. Det samme gjøres med low-cutoff frequency, her bruker vi en minimum 20 bpm delt på 60sekunder = ca. 0,3. Utgangen av filteret sendes inn i build waveform-blokken eller "bundle". Deretter sender vi dataen ut av subVI og signalet vises på grafen "Filtered Data". Koden vises i figur 24

4) *Filtered Data Adjustments*: For å forbedre filtrerings VI så mye som mulig kan man bruke en metode som kalles padding. Slik som det står nå har den filtrerte dataen en del urolighet i oppstartsfasen. Padding brukes for å fjerne dette uten å bare klippe det bort. I dette tilfellet tar vi en klonen av data-arrayet vårt og inverterer det. Deretter tar vi den inverterte klonen og konkatinerer det inn foran de ikke-inverterte punktene. Videre bruker vi dette som input inn i filteret. På denne måten kommer hele den ujevne delen av det filtrerte signalet på det inverterte delen, og det originale ikke-inverterte signalet blir riktig filtrert med en gang. Hvis vi nå velger å fjerne den fabrikkerte inverterte delen av det endelige signalet, som inneholder hele den uønskede responsen, ender vi opp med kun det ønskede signalet. Dette er nå filtrert med et filter som på forhånd

ble "trent" på å fjerne akkurat det støyet den skulle. Koden vises i figur 25

Additional Signals to remove[s] brukes til å fjerne deler av signalet etter fjerningen av det inverterte padding signalet. Dette er stort sett ikke nødvendig når vi bruker padding-metoden i filtreringen av signalet.

### G. ESP32 Kode

1) *Bibliotek*: Esp koden importerer 5 biblioteker som vist i figur 26.

WiFi.h brukes for å koble ESP32 opp mot WiFi nettverk. Webserver.h for å sette opp en Webserver og behandle Get-requests som skal mottas. WebSocketServer.h er for å sette opp Web socketen som kontinuerlig sender den målte dataen over WiFi. SPIFFS.h brukes for behandling av HTML filen vi laster opp på ESP separat fra koden. slik at vi slipper å skrive HTML og CSS kode rett inn i ESP koden. Adafruit\_ADS1X15.h brukes for kommunikasjon mellom ESP og ADC modulen som er montert på kretskortet over I2C.

2) *Handle Get Request()*: Funksonen Handle Get Request leser av index.html filen og sender deretter html koden til den som la inn get request, kode i figur 27

3) *broadcastWebSocket()*: broadcastWebSocket funksjonen Leser av ADC outputen dersom den er klar, formaterer det med litt tekst og broadcaster resultatet til websocket. Figur 28

4) *Esp Loop*: Loop koden består av tre funksjonskall. Den første heter http.handleClient() og kjører funksjonene satt i http.on. det vil si HandleGetRequest() funksjonen. WebSocket.loop() venter til systemet er klar for neste websocket pakke. deretter kjøres BroadcastWebSocket() funksjonen. Loop kode i figur 29

## IV. BEARBEIDING OG RESULTATER

### A. Simulering

Milepæl 1 av prosjektoppgaven gikk ut på å finne tre egenskaper for kretsen:

- Grafer av bode plot
- 3db-signals båndbredde
- EKBW

Disse egenskapene er med på å fortelle hvilke signaler som vil bli forsterket gjennom kretsen, og hvilke som blir filtert bort. Etter å ha redigert noen av symbolene i LTspice, endte kretsen slik som figur 6. Ved å simulere, og probe utgangen, ble bodeplottet skapt, se figur 7.

Bodeplottet i figur 7 viser at kretsen har to 3-db knekkfrekvenser. En ved 33mHz, og en ved 15.9Hz. Videre ble formler for å finne EKBW brukt i LTspice. Dette gir et tall direkte ut i "error log" vinduet. Resultater kan sees i figur 9.

Ut fra bildet ser vi at simulasjonen ga en støybandbredde lik 43.2Hz. Videre ble kretsen testet for hvordan

TABLE I  
METNINGSPENNING VS VED FORSKJELLIGE FREKVENSER

Freq	1mHz	1Hz	100Hz
Vs	0.4V	0.012V	0.078V

den oppførte seg ved forskjellige frekvenser, og hvilke spenninger som førte til metning ved forskjellige frekvenser. Kretsen testes ved 1mHz, 1Hz og 100Hz

Videre støyanalyse av modellen førte til resultateene:

$V_{norms} = 1.1\text{mV}$  og  $SNR0 = 58.8\text{db}$

### B. Prototype

Før milepæl 2, var det en lab hvor vi kunne teste oppkoblingen av kretsen. Denne labben var mest for å sjekke hvorvidt den oppkoblede kretsen oppførte seg som simulert. Det ga også studentene en mulighet for å justere komponenter og koblinger før det endelige designet ble låst. Det fungerte altså som en revisjon 1, eller prototype, for videre produksjon. Resultatet fra denne kretsen kan sees i figur 33

### C. Testing av kretskort

Kretskortet ble som nevnt tidligere i rapporten, designet i Altium, med tanker om at den skulle være så liten og kompakt som mulig. Kortet ble innsisert da det kom tilbake, med ingen synlige defekter. Det ble loddet opp og testet på samme vis som kretsen fra labben. Kretskortet gikk da gjennom de samme testene for frekvensrespons som prototypen i figur 32.

Det ble også utregnet CMRR til ca 70db. Dette kan sees i figur 34.

Etter dette ble kretsen koblet opp til en student, og det ble foretatt EKG målinger. Disse resulterte i målingene i labVIEW som kan sees på figur 31

## V. DRØFTING

Med baktanke om at EKG måleren skulle være en kompakt og bærbar enhet, ble prosjektet gjennomført på godt vis. Selve kretskortet er litt større enn selve batteriet. Batteriet får plass på undersiden av kortet 38 For å spare plass, ble det gjort noen kompromiss, som å ikke inkludere batteri-lader og bryter. Dette kunne ha blitt lagt til, men det hadde gått på bekostning av plass. Batterilader ble i ettertid montert, se figur 36, men dette krevde modifikasjoner til batterilader-PCB'en. Batteriladeren kunne ha blitt integrert i kretsen, men denne ville da ha krevd at en bryter ble benyttet. Ettersom bryteren tildelt av NTNU var relativt stor. Og ettersom gruppen var inneforstått med at kretskortet skulle være liten, og kun benytte seg av komponenter tildelt av NTNU, bestemte gruppen seg for å ikke ha med noen av delene. Gruppen designet også et analogt twin-T notch filter 21 for 50Hz, for at filteret skulle ha en effektiv frekvens ved 50Hz krevde det kondensatorverdier som ikke var innenfor E12-standard. 50Hz

er en kjent trøblete frekvens som stammer fra omgivelsene, og ble informert om i faget.

Målingene som ble gjennomført ved hjelp av MyDAQ'en frekvensrespons måler. Disse gav resultater som minner mye om de simulerte målingene, men ikke helt. Dette er trolig grunnet at simuleringen tar for seg ideelle forhold, noe som ikke er tilfellet for kretskortet. Transmisjonelinjene fungerer både som motstander, spoler og kondensatorer til en vis grad, og dette vil ha en påvirkning på hvordan kretsen oppfører seg. Spesielt "Via'er" er kjent for å ha egenskapene til en kondensator. Derfor er det prøvd å unngå bruken av for mange via'er, og alle komponentene til selve EKG-måleren er på samme side av kretskortet.

Dataen kunne bli overført fra ESP32 til en nett-basert tjener. Ettersom denne nett-tjeneren hadde en del forsinkelser, ble det vanskelig å lese av signalene. Det ble derfor benyttet mest Labview for testing og videre forsøk. Labview programmet fungerte bra, men det visste seg at kretsen trengte en del filtrering for at signalet skulle bli avlesbart. Når signale først var avlesbart, målte kretsen verdier som stemte temmelig godt med forventningene. I noen tilfeller ble signalet vanskelig å lese, men dette kan være på grunn av at probene ble plassert på feil sted.

Dette er et lite prosjekt, men det inneholder ganske mange ledd. Det kan tenkes at hvert av disse leddene kan være en mulig kilde for feil. Den største feilkilden er nok selve PCB'en da denne ikke er grundig testet ved flere ulike komponenter. De fleste "off the shelf" komponentene som ble brukt i kretsen, er allerede testet, og har oppgitt funksjonalitet i databladene sine. Til tross for dette, er det ikke gitt at de fungerer uten feil. De fleste passive komponentene er nokså feilfrie, men de mer innviklede komponentene kan tenkes å ha blitt ESD skadet dersom de ikke har blitt behandlet og oppbevart på riktig vis.

En stor synder når det kommer til støy, er strømførende plan. Kretsen benytter to store jordingsplan, men er allikevel ikke immun mot støy. Ettersom det er flere ulike spenningsnivåer på kretsen, burde disse være meget separerte fra signal-banene. I noen tilfeller er de ganske nærme hverandre, noe som kan føre til "cross-talk". Selve EKG probene og kablene har heller ikke skjerming. Dette fører til at de er meget utsatt for eksterne støykilder. Gruppen oppdaget blant annet at kretsen ble meget støyutsatt dersom signal-ledningene lå kveilet rundt seg selv. Hypotesen til gruppen er at den har da fungert som en stor jordings-sløyfe (grunnet Analog jord), og tatt til seg mye støy.

## VI. KONKLUSJON - ERFARING

Gjennom dette prosjektet, har gruppen erfart hvor mye støy og signal integritet har å si for konkrete gode målinger. Gjennom ulike tester har vi laget en krets som oppfører seg tilnærmet likt simuleringen den er basert på. For at kretskortet skulle bli så bra som mulig, har det blitt gått gjennom flere versjoner, hvor resultatet endte opp som en kompakt og bærbar EKG måler egnet godt til bruk i felten. Gruppen fikk også erfare hvor mye jording av bioelektriske objekter har å si for utsenede av signalet. Alt i alt er gruppen godt fornøyd med prosjektet.

## REFERENCES

- 1 Instructables, "ECG Signal Modeling in LTspice," Instructables, Dec. 2020. <https://www.instructables.com/ECG-Signal-Modeling-in-LTspice/> (accessed Nov. 28, 2022).
- 2 W. Storr, "Band Stop Filter and Notch Filter Design Tutorial," Basic Electronics Tutorials, Oct. 20, 2015. <https://www.electronics-tutorials.ws/filter/band-stop-filter.html> (accessed Nov. 28, 2022).
- 3 Mathuranathan, "Equivalent noise bandwidth (ENBW) of window functions - GaussianWaves," GaussianWaves, Sep. 13, 2020. <https://www.gaussianwaves.com/2020/09/equivalent-noise-bandwidth-enbw-of-window-functions/> (accessed Nov. 28, 2022).
- 4 A. Kumar, "Three(3),Five(5),Ten(10) Lead ECG Lead /Cable/Electrode Placement," Biometriccables, Jan. 08, 2020. <https://www.biometriccables.in/blogs/blog/three3-five5-ten10-lead-ecg-cable-electrode-placement> (accessed Nov. 28, 2022).
- 5 "INA126 datasheet" Ti.com, 2022. <https://www.ti.com/lit/ds/symlink/ina126.pdf> (accessed Nov. 28, 2022).
- 5 "TL071 datasheet" Ti.com, 2022. <https://www.ti.com/lit/ds/symlink/tl071.pdf> (accessed Nov. 28, 2022).

## VII. APPENDIX

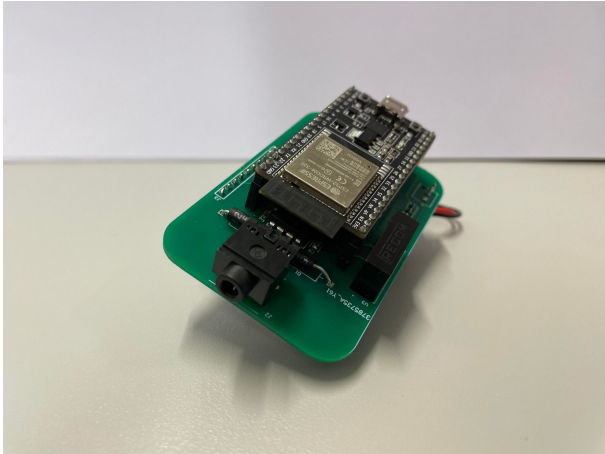


Fig. 1. PCB bilde 5

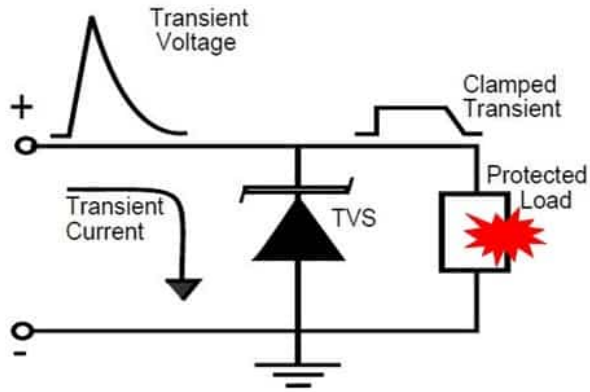


Fig. 2. Overspennings beskyttelse fra diode

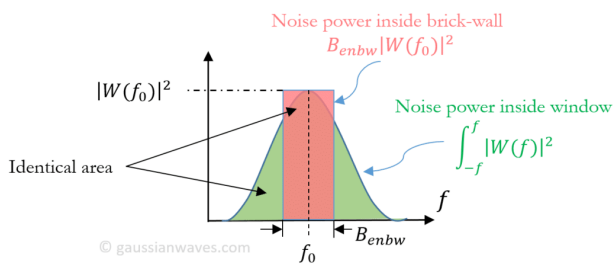


Fig. 3. EKBW figur, [3]

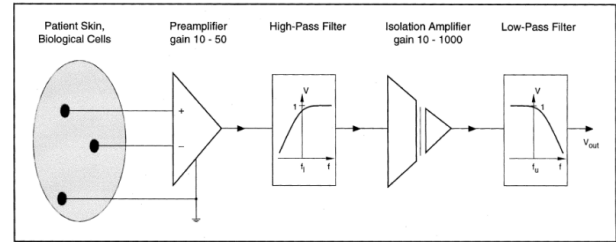


Fig. 4. Forenklet model av en EKG-Krets.

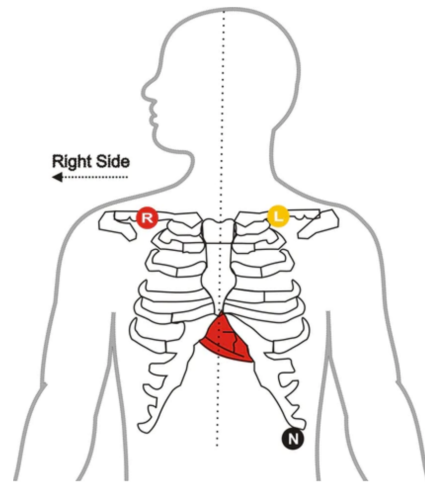


Fig. 5. Plassering av prober, [4]

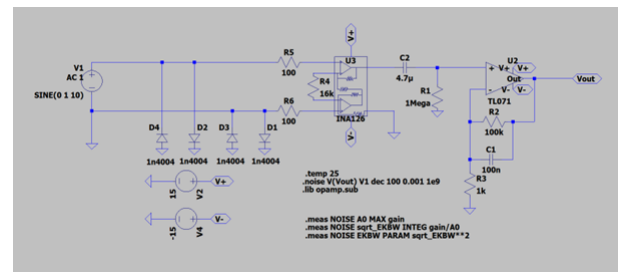


Fig. 6. Skjermdump fra LTspice

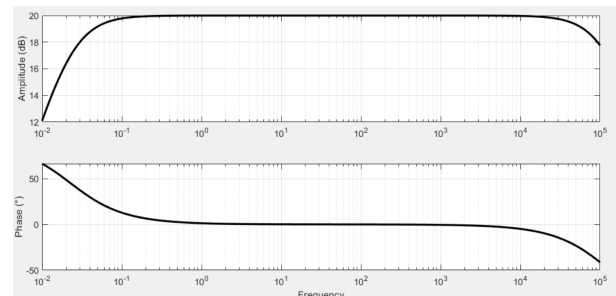


Fig. 7. BodePlot fra den simulerte kretsen

```

.temp 25
.noise V(Vout) V1 dec 100 0.001 1e9
.lib opamp.sub

.meas NOISE A0 MAX gain
.meas NOISE sqrt_EKBW INTEG gain/A0
.meas NOISE EKBW PARAM sqrt_EKBW**2

```

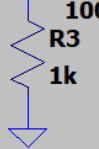


Fig. 8. Kommandoer brukt for å finne støybandbredde

```

sqrt_ekbw: INTEG(gain/a0)=6.572 FROM 0.001 TO 1e+009
a0: MAX(gain)=1007.27 FROM 0.001 TO 1e+009
ekbw: sqrt_ekbw**2=43.1912

```

Fig. 9. Errorlog

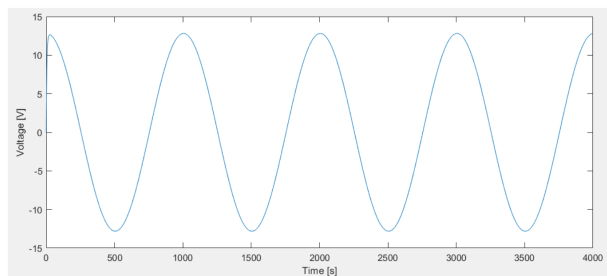


Fig. 10. Utgang av den simulerte kretsen 1mHz-0.43v

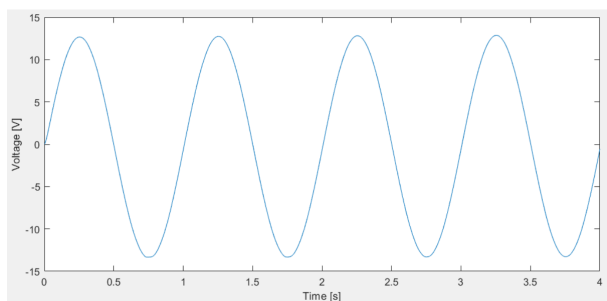


Fig. 11. Utgang av den simulerte kretsen 1Hz-0.013v

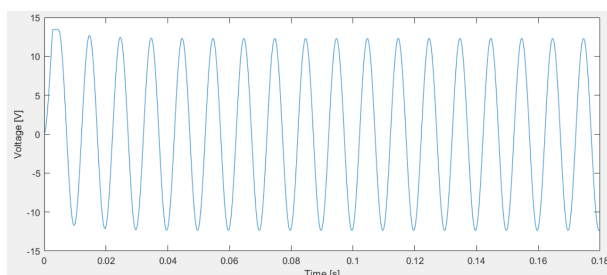


Fig. 12. Utgang av den simulerte kretsen 100Hz-0.078v

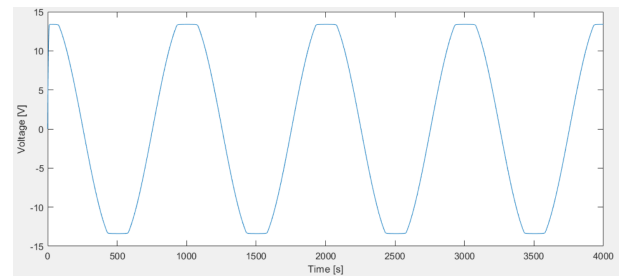


Fig. 13. Metning i kretsen

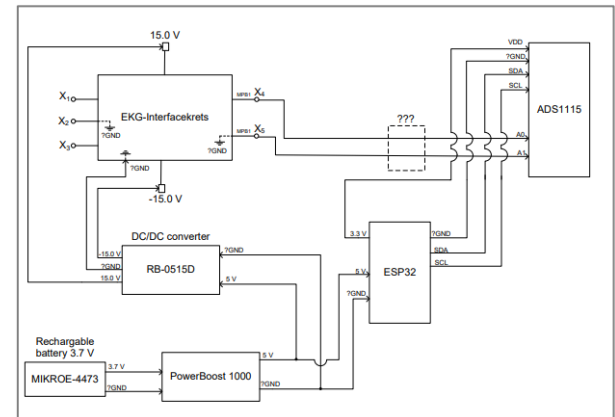


Fig. 14. Blokkdiagram som ble utlevert i emnet

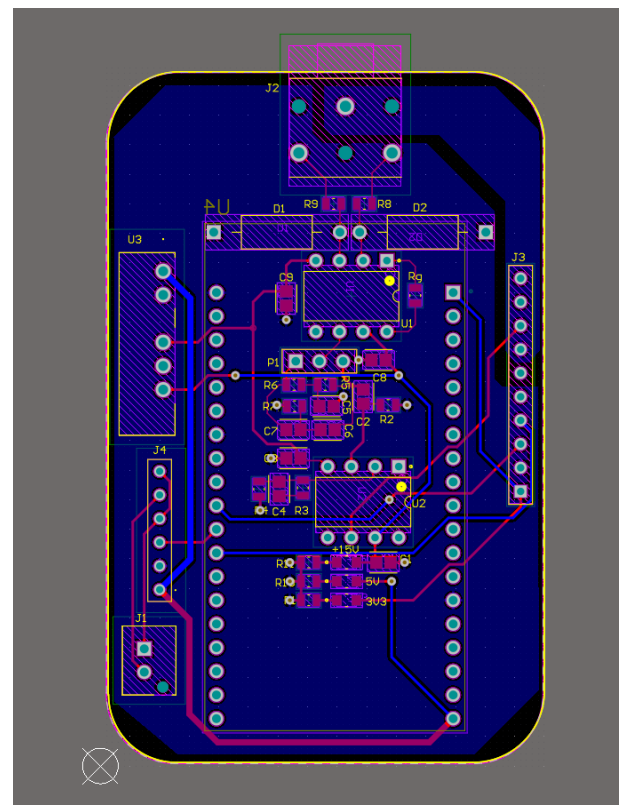


Fig. 15. 2D Utlegg

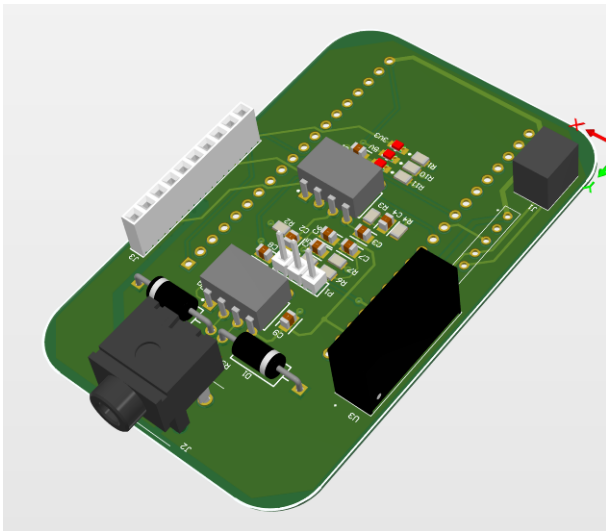


Fig. 16. 3D render av PCB'en

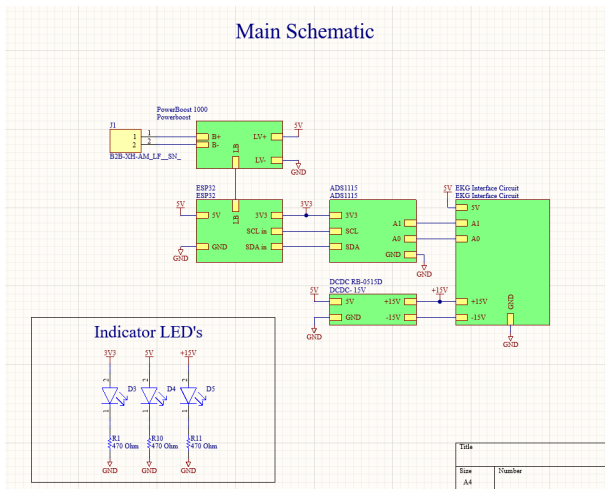


Fig. 17. Hoved skjematikken

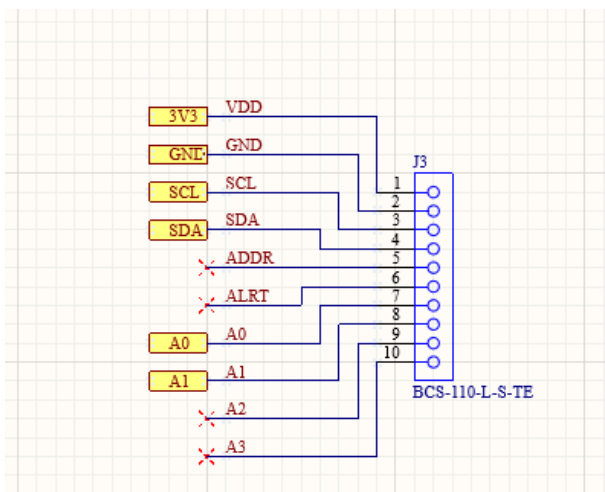


Fig. 18. ADS1115 Skjemattikk

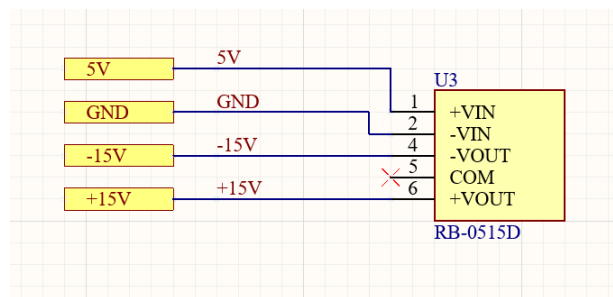


Fig. 19. DCDC skjemattikk

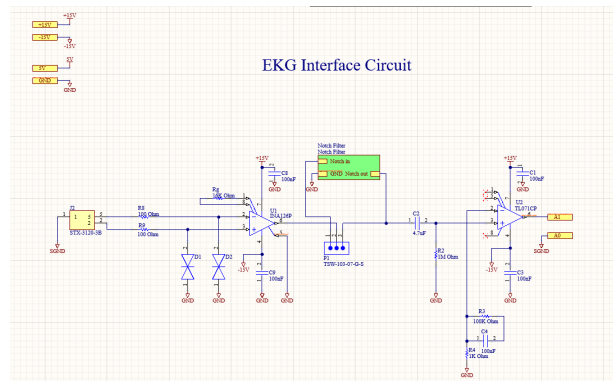


Fig. 20. Interface skjemattikk

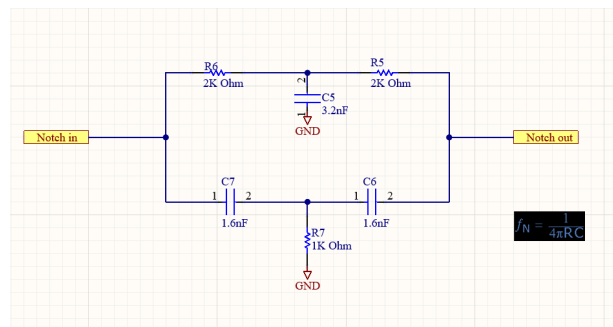


Fig. 21. Notch filter skjemattikk

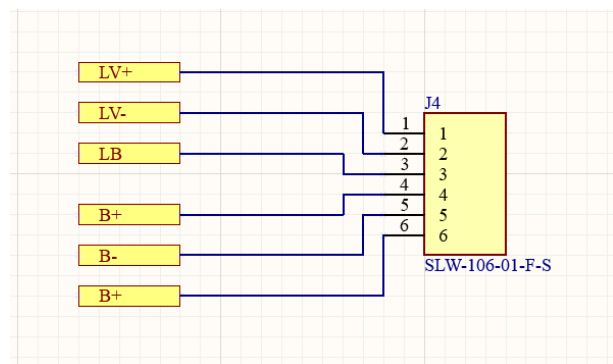


Fig. 22. PowerBoost skjemattikk



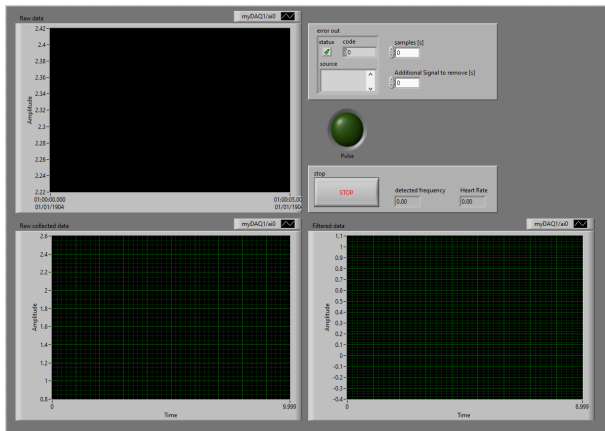


Fig. 23. Main,vi

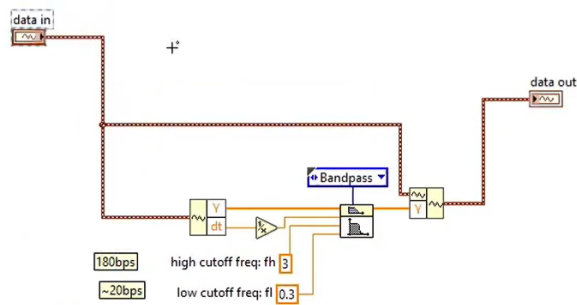


Fig. 24. Kode for Filtrering

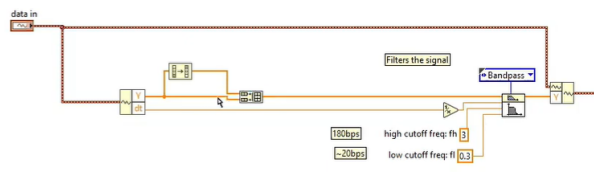


Fig. 25. Padding Metode

```

1 #include <WiFi.h>
2 #include <WebServer.h>
3 #include <WebSocketsServer.h>
4 #include <SPIFFS.h>
5 #include <Adafruit_ADS1X15.h>

```

Fig. 26. ESP biblioteker

```

17 void handleGetRequest() {
18     File webpage = SPIFFS.open("/index.html", "r");
19     http.streamFile(webpage, "text/html");
20     webpage.close();
21 }

```

Fig. 27. Handle get Request funksjon

```

23 void broadcastWebSocket() {
24     if (adc.conversionComplete()) {
25         int16_t reading = adc.getLastConversionResults();
26         adc.startADCReading(ADS1X15_REG_CONFIG_MUX_DIFF_0_1, false);
27         volts = adc.computeVolts(reading);
28         char json[20];
29         sprintf(json, "{\"ECG\":%f}", volts);
30         websocket.broadcastTXT(json);
31     }
32 }

```

Fig. 28. broadcastWebSocket funksjon

```

71 void loop() {
72     http.handleClient();
73     websocket.loop();
74     broadcastWebSocket();
75 }

```

Fig. 29. ESP Loop

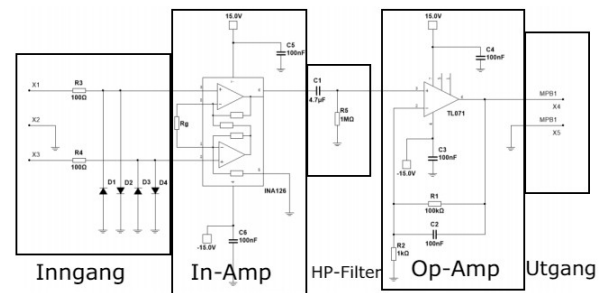


Fig. 30. Kretsen og dens respektive deler

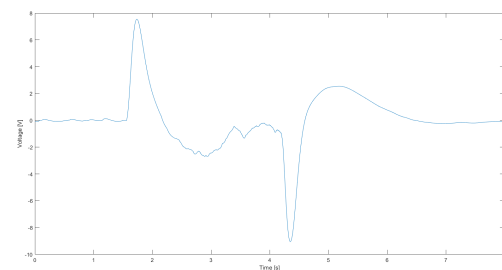


Fig. 31. EKG signal fra LabVIEW

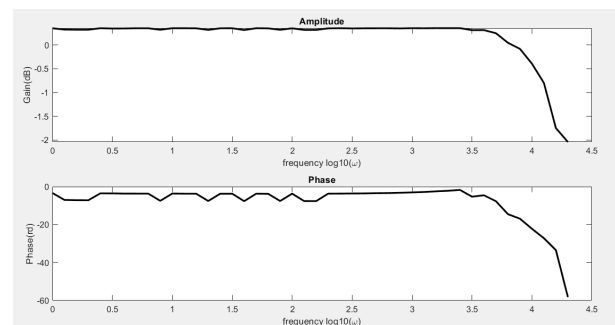


Fig. 32. Bodediagram for kretskortet

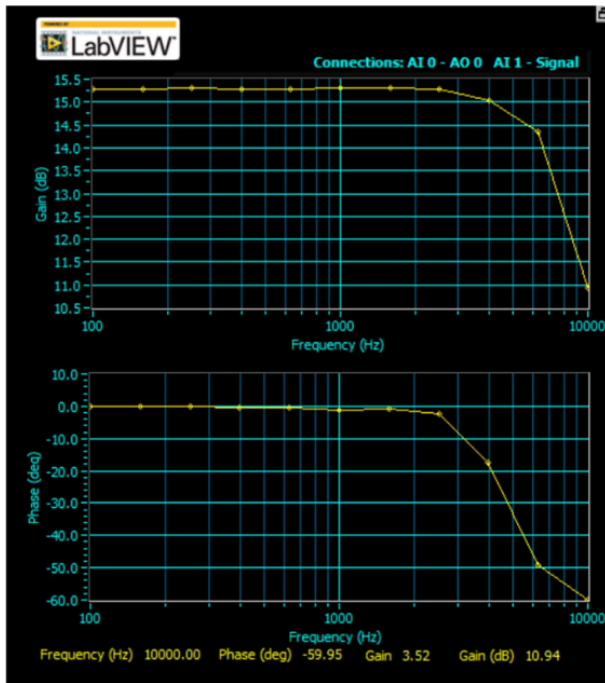


Fig. 33. Bodediagram for prototype

Beregn CMRR

$$G_{diff} = \frac{V_{out}}{V_{in}} \Rightarrow 20 \log_{10} G_{diff}$$

$$G_{cm} = \frac{V_{cm}}{V_{cm}} \Rightarrow 20 \log_{10} G_{cm}$$

$$CMRR = \frac{G_{diff}}{G_{cm}} = G_{diff}[dB] - G_{cm}[dB]$$

Målinger:  $V_{cm} = 1.44mV$

$$G_{diff} = \frac{V_{out}}{V_{in}} = \frac{20V}{2V} = 10, \quad 20 \log(10) = 20db$$

$$G_{cm} = \frac{V_{out}}{V_{cm}} = \frac{1.44 \cdot 10^{-3}}{5V} = 2.88 \cdot 10^{-4}, \quad 20 \log(2.88 \cdot 10^{-4}) = -70db$$

$$CMRR = \frac{G_{diff}}{G_{cm}} = \frac{20db}{-70db} = -0.285 \text{ ved } 1 \text{ kHz}$$

Hva er dårlig CMRR?

“-50 dB CMRR poor noise rejection. -60 dB CMRR average noise rejection. -70 dB CMRR good noise rejection. -80 dB CMRR very good noise rejection. -90 dB CMRR excellent noise rejection.”  
(Cmrr-What-Is-It-And-How-Do-You-Get-a-Good-Number, 2012)

Fig. 34. Beregning av common mode rejection ratio



Fig. 35. PCB bilde 1



Fig. 36. PCB bilde 2

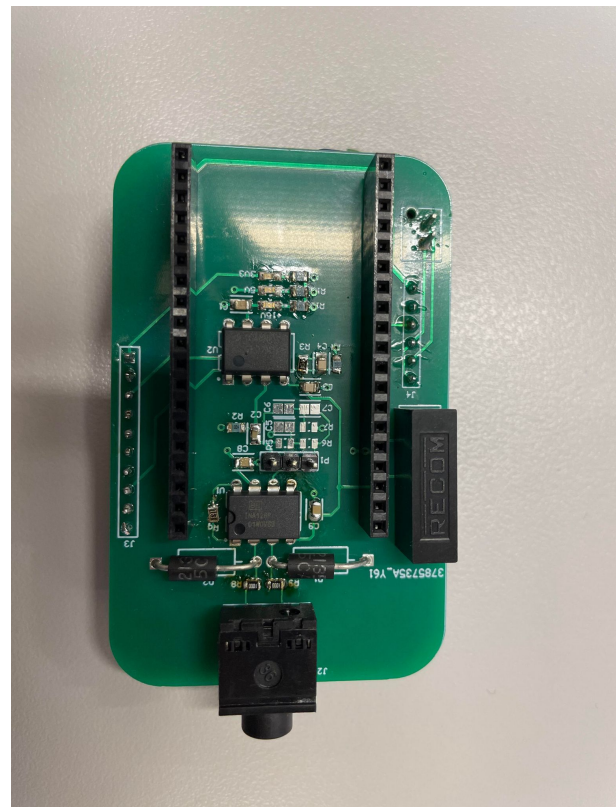


Fig. 37. PCB bilde 3

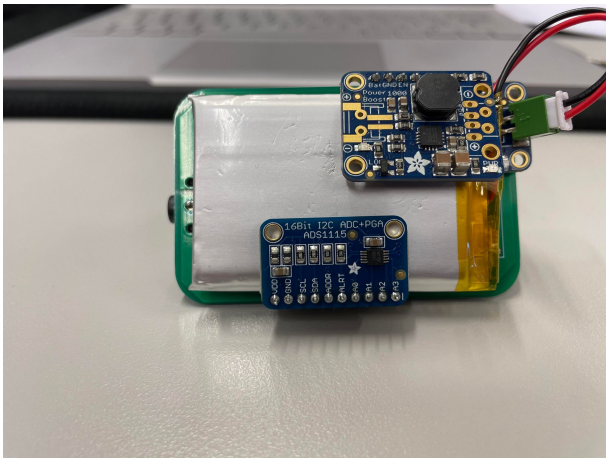


Fig. 38. PCB bilde 4