Daniel Carton
Tobias Lien

Bacheloroppgave

Testutstyr for programmerbar ammunisjon

Bachelor i ingeniørfag, Elektro

Mai 2022

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for design

NTNU
Kunnskap for en bedre verden

Bacheloroppgave

2022

NTNU

Kunnskap for en bedre verden

Daniel Carton
Tobias Lien

Testutstyr for programmerbar ammunisjon

Bachelor i ingeniørfag, Elektro

Bacheloroppgave
Mai 2022

NTNU
Norges teknisk-naturvitenskapelige universitet.
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for design

NTNU
Kunnskap for en bedre verden

NTNU
Kunnskap for en bedre verden

INSTITUTT FOR ELEKTRONISKE SYSTEMER

BACHELOR I INGENIØRFAG, ELEKTRO

Testutstyr for programmerbar ammunisjon

Daniel Carton, Tobias Lien
Nammo - NTNU

Mai, 2022

Sammendrag

Oppgavens tittel:	**Dato:**	20.05.2022		
Testutstyr for programmerbar ammunisjon	Antall sider:	43		
	Antall vedlegg:	11		
	Masteroppgave:		**Bacheloroppgave:**	X
Navn: Daniel Carton, Tobias Lien				
Veileder: Knut Wold				
Kontaktperson: Lars-Christian Lund				

Sammendrag

Denne rapporten omhandler utvikling, konstruksjon og testing av testutstyr for programmerbar ammunisjon, en bacheloroppgave gitt av Nammo AS. Hovedmålet med oppgaven var å konstruere en robust løsning som inneholdte støtte for en rekke funksjonsbehov gitt fra oppgavegiver. Oppgaven inneholdte også krav om at det skulle være mulig å bruke testutstyret ute på testområder. Derfor ble det satt krav til at systemet skulle inneholde et batterisystem med mulighet til å kobles på nettstrøm, i tillegg at innkapslingen skulle være vannsikkert. Testutstyret skulle ta imot og behandle telemetridata sendt ut med ammunisjon og eksportere det til gitte format. I tillegg til telemetrien var det også behov for triggersignal for å signalisere start av datainnsamling.

Hele løsningen ble konstruert i en solid koffert med eget batterisystem, selvutviklet kretskort og grensesnittpanel for brukerinput og inn- og utganger. Systemet er basert rundt en ESP32 mikrokontroller som styrer og behandler dataen mottatt fra ammunisjonen. Testutstyret bruker en rekke inn- og utganger som BNC, RS-422, RS-232 og SD-kort for eksportering av data. Det ferdig konstruerte testutstyret ble testet for implementerte funksjoner, samt den mekaniske løsningen.

Testene utført viste lovende resultater i forhold til hva som var forventet og ønsket. Derimot kan det ikke sies at testutstyret er et ferdig utviklet produkt, da systemet fortsatt mangler noen funksjoner. Nåværende løsning har maskinvaren nødvendig slik at videre arbeid skal kun være videre programmering. Prosjektet konkluderes med at selv om løsningen ikke er fullstendig ferdig implementert, er nåværende testutstyr et godt grunnlag for videre arbeid og utvikling.

Nøkkelord: Testutstyr, ESP32, Databehandling, Kretskonstruksjon

Abstract

Title:	**Date:**	20.05.2022		
Testing equipment for programmable ammunition	Number of pages:	43		
	Number of attachments:	11		
	Master's thesis:		**Bachelor's thesis:**	X
Name: Daniel Carton, Tobias Lien				
Supervisor: Knut Wold				
Contact person: Lars-Christian Lund				

Abstract:
This report deals with the development, construction, and testing of testing equipment for programmable ammunition, a bachelor's thesis given by Nammo AS. The main goal of the thesis was to construct a robust solution that included support for several functional needs given by the thesis provider. The thesis also included requirements that it should be possible to use the test equipment out in the field. Therefore, it was required that the system should contain a battery system with the option of connecting to mains power, in addition to the enclosure being waterproof. The testing equipment's purpose is to receive and process telemetry data sent out with ammunition and export it to a given format. In addition to telemetry, there was also need for a trigger signal to signal the start of data collection.

The entire solution was constructed in a solid case with its own battery system, self-developed circuit board and interface panel for user input and also the inputs and outputs to the system. The system itself is based around an ESP32 microcontroller that controls and processes the data received from the ammunition. The testing equipment uses a number of inputs and outputs such as BNC, RS-422, RS-232 and SD cards for data export. The finished testing equipment was tested for the implemented functions as well as the mechanical solution.

The tests performed showed promising results in relation to what was expected and desired. On the other hand, it cannot be said that the testing equipment is a fully developed product, as the system still lacks some functionality. The current solution does have the hardware necessary so that further work will only require further programming. The project concludes that even though the solution has not been fully implemented, the current testing equipment is a good basis for further work and development.

Keywords: Testing equipment, ESP32, Data processing, Circuit construction

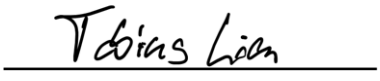
Forord

Denne rapporten markerer slutten på bacheloroppgaven og tre års skolegang på elektroingeniørstudiet på NTNU i Gjøvik. Oppgaven ble startet opp i januar 2022 og stilt ferdig i slutten av mai i samme år. Oppgaven ble gitt av utviklingsavdelingen for elektronikk i ammunisjon ved Nammo AS. Formålet med oppgaven var å utvikle og designe teknisk testutstyr for å ta imot og behandle telemetridata sendt ut fra programmerbar ammunisjon.

Gjennom bacheloroppgaven har vi vært innom flere kjente arbeidsområder som vi har vært gjennom under det tre-årlige løpet på NTNU samt nye og ikke kjente områder. Dette tok vi som en utfordring og som en mulighet til å utvide våre faglige ferdigheter.

Vi ønsker å takke alle som har gitt oss hjelp og veiledning med oppgaven og vi vil spesielt gi vår takk til følgende:

- Vår veileder Knut Wold for veiledning
- Kontaktperson fra Nammo AS, Lars-Christian Lund, for god kommunikasjon og støtte gjennom hele oppgaveprosessen.
- Kjell Are Refsvik for bistand med laserkutting og modellering



Tobias Lien
19.05.2022

Daniel Carton
19.05.2022

Innhold

Figurer

Tabeller

Kodelister

OrdListe

Begreper

Allotrope	En av flere typer forskjellige materialer oppbygd av samme element eller molekyl.
Arduino IDE	Data program for programmering og opplastning til ESP32 og liknende mikrokontroll
Array	Brukt i programmering som en liste med variabler av samme datatype.
Bus-Topologi	Topologi hvor nettverket er koblet slik at alle enhetene leser fra samme datalinje.
Bypass-kondensator	En kondensator brukt for å filtrere ut høyere frekvenser for sensitive integrerte kretser.
Char	En datatype i c++ som lagrer en karakter.
DigitalRead	En funksjon som får mikrokontrolleren til å lese den binære verdien til en pin.
Dropout-spenning	Et spenningskrav mellom spenning ut og spenning inn som må overkommes. Brukt i lineære spenningsregulatorer.
Float	En datatype som kan lagre et desimaltal.
Ikke-flyktig minne	Minne som ikke slettes når strømtilførselen fjernes.
Int	En datatype som kan lagre et heltall.
Interrupt	Funksjon brukt i programmering som kan kjøre med en gang en gitt tilstand er møtt. Interrupt funksjoner trenger ikke å vente på at andre funksjoner er ferdige, så de blir utført effektivt.
Kontroller-mål	Tidligere kalt Mester-slave (master-slave på engelsk), Kontroller-mål (controller-target på engelsk) beskriver hvordan en enhet i et nettverk kontrollerer andre enheter.
Logikk-nivå skifter	Logic Level Shifter på engelsk, en spenningsregulator for datalinjer med høyere frekvensrespons.
Matrise	Brukt i programmering som en flerdimensjonal array.
Pakning	Elementet som forhindrer væskelekkasje.
Pull-Down motstander	En motstand koblet til jord for å trekke spenningen på en inaktiv datalinje ned til 0V.
RX	Recieve-pinnen i en UART tilkobling. RXA og RXB referer til pinnene brukt for måling av differensialspenning.
Seriellbuffer	Brukt i programmering som en midlertidig liste som bevarer serielldata før det leses av mikrokontrolleren.
Topologi	Forklarer hvordan enheter i et nettverk er tilkoblet.
TX	Transmit-pinnen i en UART tilkobling. TXA og TXB referer til pinnene brukt for måling av differensialspenning.
Unicode	En standard for transmisjon, lagring og vising av karakterer.

Forkortelser

Ω	Ohm
A	Ampere
ADC	Analog til digital konverter
CAN	Controller Area Network
DAC	Digital til analog konverter
GPIO	General-Purpose In/Out
Hz	Hertz
IC	Integrated Circuit, integrert krets
ICSP	In-Circuit Serial Programming
IDE	Integrated Development Environment
IoT	Internet of Things
LCD	Liquid Crystal Display
Li-Ion	Litium-ion
LLS	Logic Level Shifter
mAh	Milli ampere-time
MDF	Tre-produkt bestående av tre-fibre kombinert med voks og lim.
OC	Overcharge, overlading
OD	Over-discharge, over-utlading
Op-Amp	Operational Amplifier
PCB	Printed Circuit Board
QSPI	Quad Serial Peripheral Interface
RAM	Random Access Memory
RF	Radio Frekvens
ROM	Read-Only Memory
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
V	Volt
W	Watt

Kapittel 1

Innledning

1.1 Bakgrunn

Nammo AS, som er oppdragsgiveren til denne bacheloroppgaven, er et internasjonalt konsern med virksomhet innen produksjon og utvikling av ammunisjon, romfartsprodukter, og rakettdrift. Gjennom en sammenslåing av tre ammunisjonsbedrifter fra Norge, Sverige og Finland ble Nammo AS stiftet i august 1998. Med hovedkvarter på Raufoss i Norge, er Nammo per dags dato etablert i 12 land deriblant Sverige, Finland, USA, Tyskland og Storbritannia med totalt over 2700 ansatte.

1.2 Introduksjon

Oppgaven var utdelt fra utviklingsavdelingen for elektronikk i ammunisjon ved Nammo, herved oppført som oppgavegiver. Oppgaven innebærte konstruksjon av et generisk testutstyr som skulle kunne behandle data fra flere typer skyte-tester til diverse ammunisjonsutviklingsprogrammer. Testutstyret skulle ha mulighet for oppkobling til en RF-modul som skal motta telemetridata sendt ut fra ammunisjonen under testing. Deretter skulle testutstyret behandle disse dataene og eksportere det ut til gitte format.

1.3 Problemstilling

Fra gitt oppgave og krav ble det bestemt at prosjektet skulle løse problemstillingen under. Problemstillingen er fokusert nok til å være en god veiledning for å oppnå kravene, i tillegg til å være åpen nok for at gruppen kan ta egne valg for funksjonalitet.

"*Utredning av robust testutstyr system som skal kunne behandle telemetridata fra programmerbar ammunisjon*"

Løsning av denne problemstillingen er målet til gruppen under prosjektet, og alle valg tatt skal være for å løse problemstillingen på den mest effektive måten med tilgjengelige ressurser.

1.4 Funksjonalitetsbehov og krav

Følgende funksjonalitetsbehov ble gitt av oppgavegiver:

- Galvanisk skille mellom inngang- og utgangssignaler.
- Mulighet for kalibrering
- Kommunikasjon med følgende protokoller; RS-422, RS-232, CAN-Bus og USB
- Følgende innganger:
    - TTL med Open drain

Det ble også gitt følgende krav til oppgaven:

- Testutstyret skal være bærbart og inkluderer batterisystem med mulighet for tilkobling til strømnettet.
- Innkapslingen til testutstyret skal være robust og vannresistent.

1.5 Rapportinnhold

Rapporten består av ulike kapitler for å vise fremgangen av planleggingen og utførelse, samt grunnleggende teori for å gi en forståelse rundt valgene gjort i prosjektet. Følgende viser hvordan rapporten er inndelt kapittelvis:

Kapittel 2 - Teori: Teorikapittelet inneholder en gjennomgang av det teoretiske grunnlaget for elementer brukt i prosjektet. Dette blir forklart for å gi en forståelse rundt valgene tatt under design og konstruksjon av produktet.

Kapittel 3 - Metode: Dette kapittelet beskriver hvordan konstruksjon av testutstyret planlegges og utføres. Kapittelet skal også gå gjennom de diverse komponentene, bruksområdene deres og hvordan de var brukt i prosjektet. I tillegg blir konstruksjonen av testutstyret forklart for for å vise hvordan kravene var oppnådd.

Kapittel 4 - Resultater: Resultatskapittelet skal gå gjennom diverse tester utført for å verifisere at testutstyret tilfredstiller kravene gitt av oppgavegiver. Kapittelet skal også inkludere resultatene av de gjennomførte testene.

Kapittel 5 - Diskusjon: Dette kapittelet diskuterer resultatene oppnådd fra de diverse testene. Resultatene skal også sammenliknes med målene og kravene satt av oppgavegiver og diskutere hvordan resultatene kunne forbedres.

Kapittel 6 - Konklusjon: Kapittelet skal konkludere prosjektet med en oppsummering av krav og resultater, samt sammenlikning av problemstillingen med resultatet oppnådd.

Kapittel 2

Teori

2.1 Li-Ion Celler

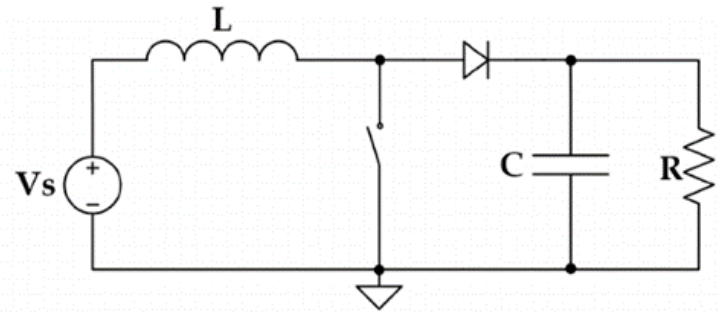
Litium-ion batteri er oppladbare batteri som brukes ofte i bærbar elektronikk. Li-ion batteri er ofte foretrukket fordi de har høyere forhold av energi og vekt sammenliknet med andre typer batteri som alkaliske og sink-karbon batteri. Oppbyggingen av en typisk li-ion celle består av en anode og katode som henholdsvis er dens negative og positive elektroder. Anoden er som regel en allotrope av karbon, typisk grafitt [1]. Katoden består av en type metalloksid, generelt en kombinasjon av litium og et annet metall som f.eks. litium-kobolt oksid eller litium-jern fosfat. Elektrodene er skilt med en elektrolytt bestående typisk av en litium-salt løsning [2]. Flere typer elektrolytter finnes til bruk for forskjellige typer batteri, f.eks. litiumpolymerbatteri som bruker en polymerelektrolytt som er en type halvfast elektrolytt [1].

Under oppladning av litium-ion celler, overføres elektroner fra ladekilden til katoden[3]. Deretter avgir katoden litium ioner som flyter gjennom elektrolytten til anoden til cellen. Med denne bevegelsen av ioner gjennom elektrolytten lagres det energi i form av kjemisk energi. Kun positive litium ioner beveger seg gjennom elektrolytten, mens elektronene blir stoppet av elektrolyttens elektrisk isolerende egenskaper[2]. Ved utladning skjer det motsatte; ionene flyter fra anoden til katoden som deretter frigjør elektroner fra katoden. Cellen telles som utladet når ionene slutter å flyte fra anoden til katoden, mens oppladet når alle ionene er flyttet fra katoden til anoden.

2.2 Boost-konverter

Boost-konvertere er en likespennings konverter som produserer en høyere utspenning i forhold til innspenningen [4]. I en ideell konverter er effekten inn lik effekten ut, som fører til at strømmen ut minker slik at effekten inn og ut er den samme. I figur 2.1 er det vist et kretsdiagram for en ideell boost-konverter. Boost-konvertere baseres på evnen til en spole å motstå endringer i strøm gjennom spolen ved å produsere en motspenning. Dette gjør den ved å øke eller senke mengden energi lagret i det magnetiske feltet til spolen. Når bryteren i figur 2.1 er åpen flyter strømmen med klokken gjennom spolen og ut til dioden, kondensatoren og lasten som her er representert med motstanden R. Dersom bryteren lukkes synker impedansen sett fra kilden og strømmen gjennom spolen øker og dermed også dens magnetiske felt. Når bryteren igjen åpnes, synker strømmen da impedansen øker. Energien lagret i det magnetiske feltet til spolen blir da brukt til å motstå strømendringen, som gjør at

polariteten til spolen snur slik at den negative siden er tilkoblet den positive siden til kilden. Dette danner en seriekobling ekvivalent til to spenningskilder, dermed kombineres spenningen til kilden og spolen som resulterer til en høyere spenning ut over lasten. Når bryteren er lukket blir spenningen over bryteren og dermed utspenningen lik null, derfor brukes kondensatoren til å lagre nok energi slik at spenningen ut til lasten holdes konstant imens spolen lades opp igjen. Dioden brukes til å blokkere strømmen i motsatt retning så kondensatoren ikke lades ut gjennom bryteren når den er i lukket tilstand.



Figur 2.1: Kretsdiagram av en ideell boost-konverter [4]

For at utspenningen skal holdes konstant, må bryteren åpnes og lukkes fort nok at slik at spolen får tid til å lagre nok energi i det magnetiske feltet i tillegg til å lade kondensatoren. Bryteren blir erstattet med en oscillator som sender et pulstog med en hastighet og pulsbredde, brukt for å styre hvor fort boost-konverteren svitsjer. Boost-konvertere kan operere i to moduser; kontinuerlig og diskontinuerlig. I kontinuerlig modus blir strømmen gjennom spolen aldri lik null når boost-konverteren er operativ. Spenningsforholdet er gitt med formel 2.1 der V_O er utspenning, VS er innspenning, og D er pulsbredden. Derimot i diskontinuerlig modus, går strømmen ned til null i av-perioden til pulstoget. Dette fører til at spenningsforholdet mellom inn og ut er avhengig av flere faktorer som spolens induktans, utgangsstrømmen, innspenningen, pulsbredden, og svitsjefrekvensen. Formel 2.2 viser spenningsforholdet for en diskontinuerlig konverter hvor V_O er utspenning, V_S er innspenning, D er pulsbredden, L er induktansen, FS er svitsjefrekvensen og R er resistansen. Full utledning av formel 1 og 2 er hentet fra kilde [4].

$$\frac{V_O}{V_S} = \frac{1}{1-D} \tag{2.1}$$

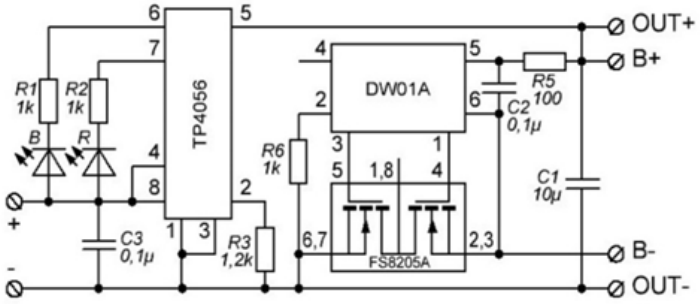
$$\frac{V_O}{V_S} = \frac{1+\sqrt{1+\frac{\frac{2D^2}{LF_S}}{R}}}{2} \tag{2.2}$$

2.3 Lading av batteri

TP4056 er en IC ofte brukt i ladekretser for mindre li-ion batteripakker. Batteriene blir ladet ved bruk av konstant spenning høyere enn den nominelle spenningen på batteriet. Under lading er ladespenningen satt til å være 4.2 V, mens ladestrømmen bestemmes av en ekstern motstand. I databladet til TP4056[5] er ladestrømmen gitt med formel 2.3, der VPROG er 1 V og RPROG er den eksterne motstanden. Når batteriet lades opp og ladestrømmen synker til en tiendedel av satt ladestrøm, avslutter ladekretsen ladingen.

$$I_{BAT} = \frac{V_{PROH}}{R_{PROG}} * 1200 \tag{2.3}$$

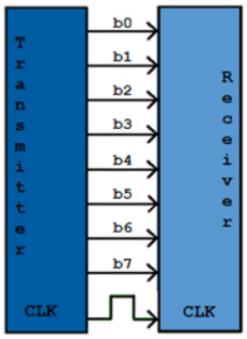
Figur 2.2 viser en krets som baseres på TP4056 [5] som danner en helhetlig ladekrets med DW01A og FS8205A som batteribeskyttelse. Her er ladestrømmen bestemt av motstanden R3, satt til å være 1.2 kΩ som gir en ladestrøm på 1 A. Modulen er utstyrt med en batteribeskyttelses krets bestående av en DW01A IC krets. DW01A baseres rundt komparatorer ved detektering av overlading, overutlading, kortslutning, overstrøm og detektering av lader. Disse komparatorene er koblet til en oscillatorkrets som videre styrer utgangene OD og OC, og dermed lading og utlading av batteriene. Denne kretsen beskytter dermed batteriene mot overlading og overutlading ved bruk av FS8205A som inneholder to MOSFET-transistorer [6] som sperrer for lading og utlading av batteriet ved at gate-inngangene blir aktivert av utgangene OD og OC på DW01A[7].



Figur 2.2: Kretsdiagram av TP4056 ladekrets modul [3]

2.4 Seriell og parallell kommunikasjon

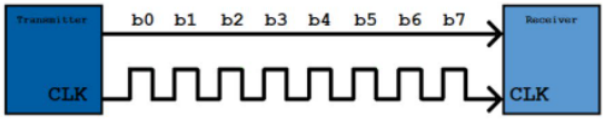
Kommunikasjon mellom to enheter i et nettverk kan deles opp i to forskjellige grupper: parallell og seriell. Dersom et nettverk bruker parallell kommunikasjon, skal alle bits i en byte sendes samtidig over alle åtte datalinjer[8]. Dette betyr at dersom en enhet leser data fra bare den første av de åtte datalinjene, mottar du bare den første bit av alle bytene som sendes. Det er også ikke uvanlig å sende et klokkesignal for å synkronisere når informasjon fra hver bit er sendt. Figur 2.3 viser et eksempel av parallell kommunikasjon.



Figur 2.3: Eksempel av parallell kommunikasjon med 8 bits[8]

Seriell kommunikasjon er en annen måte enheter i et nettverk kan sende informasjon. Der

parallell kommunikasjon sender alle bits av en byte samtidig over 8 datalinjer, sender seriell kommunikasjon alle bits i rekkefølge over bare en datalinje. Dette betyr at senderenheten trenger bare en tilkobling for sending av data, og dermed det samme med mottakerenheten[9]. I likhet med parallell kommunikasjon, finnes det protokoller som krever at et klokkesignal sendes i tillegg til data, derimot finnes det også protokoller som ikke krevet det. Figur 2.4 Viser et eksempel av seriell kommunikasjon.



Figur 2.4: Eksempel av seriell kommunikasjon med 8 bits[8]

Dette viser de forskjellige bruksområdene til seriell og parallell kommunikasjon[10], der parallell er 8 ganger raskere, men trenger 8 datalinjer pluss klokkesignal, og seriell er tregere, men krever kun en datalinje. Det er også mulig å ha nettverk som har mere eller mindre enn 8 bits i en byte, som har samme fordeler og ulemper som et 8-bit nettverk.

UART

Universal asynchronous reciever-transmitter, eller UART, er en ofte brukt protokoll innenfor seriell kommunikasjon som ikke trenger et klokkesignal. Mange mikrokontrollere og datamaskiner bruker samme standardiserte UART protokoll for kommunikasjon, som gjør det lettere å kommunisere mellom to ulike enheter[11]. Fordelen med å bruke UART er at begge enhetene kan både sende og motta data samtidig, som gjør at den er en full dupleks protokoll. En full dupleks protokoll betyr at hver enhet kan både sende og motta data samtidig, i motsetning til halv dupleks, der data kan bare bli send en retning om gangen[12]. Full dupleks oppnås ved å koble en enhets RX (recieve) pin til den andre enhets TX (transmit) pin, og motsatt for den første enhets TX pin.

I tillegg til å være en maskinvare standard, er UART også en standard for hvordan datapakker skal bygges opp for å kunne garantere trygg transmisjon. Figur 2.5 er et diagram av hvordan en UART datapakke er bygd opp.

Start bit 1 bit	Dataramme 5-9 bits	paritetsbit 0 or 1 bits	Stopp bits 1 or 2 bits

Figur 2.5: En UART datapakke[11]

Når en UART transmisjonslinje ikke sender data, er det vanlig at den holdes høy på 5V, som gjør UART til en aktiv-lav protokoll. Dette gir fordelen at brudd i transmisjonslinjen kan lettere oppdages, og redusert sensitivitet til støy. Start-biten trekker da spenningen lav i en klokkesyklus, som viser mottaker at data skal sendes. Neste del av datapakken er datarammen, som inneholder selve dataen som skal overføres. Lengden av datarammen, som kan være 5-9 bits, er bestemt mellom sender og mottaker i forkant av transmisjonen. UART protokollen bestemmer at totalt antall 1-bits sendt skal være et oddetall, så om det blir sendt 8 1-bits i datarammen, skal det legges til en 1-bit på slutten for paritet. Dersom mottaker leser et partall med 1-bits, så kan mottaker bestemme at en bit ble endret i transmisjon, og den må så kaste vekk hele mottatte datapakken[11]. Dermed kan UART inneholde og bruke

feildeteksjon for tryggere transmisjoner. Til slutt trekkes transmisjonslinjen fra lav til høy i form av en eller to stoppbits for å formidle at datapakken er ferdigsendt.

RS-232

RS-232 er en koblingsmetode brukt for seriell UART kommunikasjon. Fordelen med RS-232 er at den bare trenger 3 linjer (TX, RX, Jord) for å kunne sende data pålitelig på en lengde opptil 15m. RS-232 bruker både positive og negative spenninger, der en 0-bit er en positiv spenning mellom 3V og 15V, og en 1-bit er en negativ spenning mellom -3V og -15V som er samme amplitude som en 0-bit bare negativ[13]. Dette betyr at både sender og mottaker i nettverket trenger grensesnitt som kan lage og lese negative spenninger. For å koble to enheter i et nettverk sammen ved bruk av RS-232 skal RX til enhet 1 kobles til TX til enhet 2, og TX til enhet 1 skal kobles til RX til enhet 2. Det er også vanlig å koble til en delt jord på begge enhetene.

RS-422

RS-422 er enda en implementasjon av UART som heller bruker differensiell spenning for å sende data. Der en 0-bit i RS-232 standarden var kjennetegnet som en positiv spenning i forhold til jord, er en 0-bit i RS-422 kjennetegnet som en positiv spenning fra RXA til RXB[9]. En negativ spenning fra RXA til RXB kjennetegnes da med en 1-bit, dette gir også fordelen at det ikke trengs å lage negative spenninger. Fordelen med RS-422 over RS-232 er at transmisjonslinjen kan være mye lengre, der RS-422 kan sende data over en kilometer. Ulempen, derimot, er at en RS-422 kabel trenger minst 4 ledere, der RS-232 trengte bare minst 2 uten å telle jord.

SPI

Serial Preipheral Interface, eller SPI, er et type grensesnitt som er brukt til å kontrollere diverse enheter som sensorer, SD-kort og registere. Et SPI-grensesnitt er som oftest brukt da en krets krever flere kontrollerbare enheter som både sender og mottar informasjon[14]. Alle enhetene i et SPI-nettverk blir koblet til de samme seriell-linjene, men enheten som skal motta informasjonen blir bestemt av en ekstra datalinje kalt chip select. Chip select gjør det mulig å sende serielldata til mange flere enheter i et nettverk med bus-topologi, uten å legge til seriell-linjer for hver av de.

SPI er en full-dupleks kommunikasjonsprotokoll, som betyr at informasjon kan både sendes og mottas samtidig[12]. Dette gjøres ved å ha seriell-linjer liknende UART, der en linje er informasjon ut fra kontroller inn til mål, og en linje er inn til kontroller ut av mål[15]. I tillegg blir et klokkesignal sendt på en ekstra datalinje som synkroniserer kommunikasjonen. Det er også nødvendig å legge til en linje for jord, i tillegg er det ikke uvanlig å inkludere en spenningslinje. De forskjellige linjene beskrives under i tabell 2.1.

Navn	Funksjon
MOSI (Master Out Slave In)	Linje for data som sendes fra mikrokontroller til SPI-enhet
MISO (Master In Slave Out)	Linje for data som sendes til mikrokontroller fra SPI-enhet
SCK (Serial Clock)	Linje for sending av synkroniserende klokkesignal
CS (Chip Select)	Linje for signal som bestemmer hvilken SPI-enhet data sendes til i nettverket

Tabell 2.1: SPI datalinjer

2.5 I2C

Inter-Integrated Circuit, forkortet I2C, er en synkronisert kommunikasjonsbuss som er bygget på seriell kommunikasjon. Denne kommunikasjonsprotokollen er ofte brukt til forskjellige tilleggs enheter styrt av en mikrokontroller, som f.eks. diverse sensorer og LCD-skjermer. I2C benytter en kontroller - målstyring for kontroll av kommunikasjonen mellom to eller flere enheter. Man kan ha en eller flere kontroller-enheter som styrer en eller flere target-enheter[16]. Likt med UART kommunikasjon, bruker I2C kun to ledninger for kommunikasjon mellom to enheter. Fordelen med I2C over f.eks. UART er man kan ha flere enheter tilkoblet samme linje.

I2C krever kun to ledninger, en til data og en til klokkesignal. Datalinjen merket med SDA, Serial Data, kobles til SDA på de andre enhetene og klokkelinjen merket SCL, Serial Clock, kobles til SCL på de andre enhetene[17]. Tilkobling med I2C er dermed enkel og systemet kan lett utvides med flere enheter. I2C er likt med UART hvor når linjen er tom eller ledig er den være satt til høy-tilstand.

Når data overføres med I2C blir de overført i meldinger bestående av flere rammer av informasjon. Figur 2.6 viser hvordan en melding er bygd opp av de forskjellige rammene i datapakken.

Figur 2.6: I2C datapakke [18]

Adresserammen er en 7 eller 10 bits ramme brukt til adressering av de forskjellige mål-enhetene tilkoblet kontroller-enheten. Når kontroller-enheten ønsker å kommunisere med en av mål-enhetene sender den adressen til den enheten den ønsker via SDA linjen. Hver mål-enhet sammenligner den adressen med sin egen adresse og dersom den stemmer sender mål-enheten en en lav-signals bit, eller et lav-signal som en bekreftelse til kontroller-enheten. Dersom adressen ikke stemmer gjør mål-enhetene ingenting.

Read/write bit, eller lese/skrive bit brukes til å informere mål-enheten om at kontroller-enheten ønsker å hente informasjon eller skrive informasjon til den.

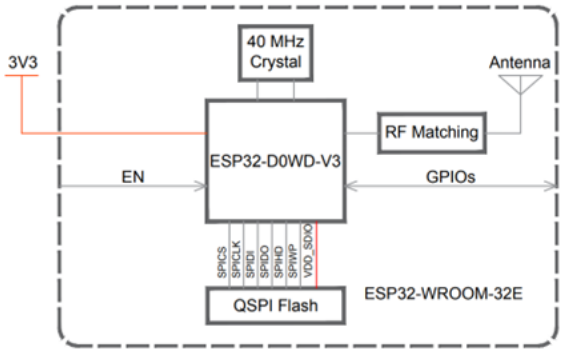
ACK/NACK bit, eller bekreftelses bit brukes til å sende bekreftelse om at informasjon er mottatt eller ikke.

Datarammen er en ramme på 8 bit som inneholder det kontroller-enheten ønsker å hente fra eller skrive til mål-enhetene. Hver dataramme følges av en bekreftelses bit for å bekrefte at datarammen har blitt mottatt. Når alle datarammene har blitt mottatt sender kontroller-enheten en stopp tilstand for å indikere slutten på meldingen og SDA linjen går tilbake til høy-tilstand.

2.6 ESP32

Mikrokontrolleren ESP32 som er utviklet av Espressif er en mikrokontroller med innebygd WiFi og Bluetooth. Kombinert med dens lave kostnad og effektforbruk er ESP32 ofte benyttet til IoT bruksområder, som f.eks. smarthus applikasjoner og annen dataoverføring fra sensorsystemer over Internett. ESP32 baseres på Xtensa LX6 32-bit prosessor med enten en eller to kjerner, som også inneholder 448 kB ROM-minne brukt til fastvare og 520 kB SRAM brukt til register buffer. Noen versjoner av mikrokontrolleren har ikke noe form for innebygd flash-minne og de krever derfor en ekstern lagringsenhet for ikke-flyktig minne. I tillegg til WiFi 802.11 og Bluetooth v4.2, støtter også ESP32 2 x I2C-kommunikasjon linjer, 3 x UART seriell-kommunikasjon linjer, 18 kanaler med 12-bits ADC, 2 x 8-bits DAC og 34 programmerbare GPIO-pinner[19].

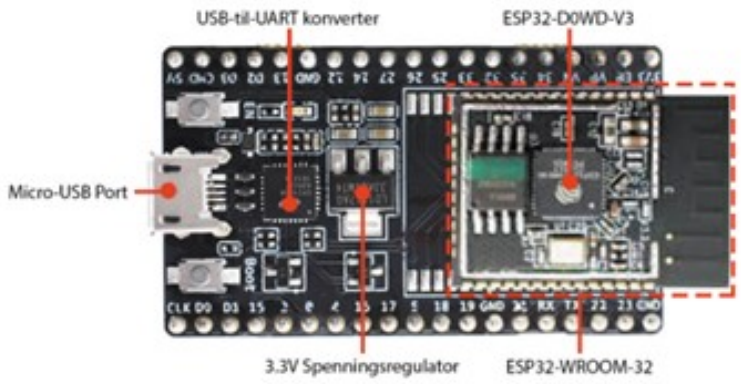
Ofte blir ESP32 benyttet i moduler som inneholder flere nødvendige eksterne komponenter. Figur 3 viser et blokkdiagram for modulen ESP32-WROOM-32E. Denne modulen bruker en ESP32-D0WD-V3 mikrokontroller tilkoblet til en 40 MHz krystalloscillator. Ved bruk av en multiplikator i mikrokontrolleren oppnår den en klokkefrekvens som kan variere mellom 80-240 MHz. I tillegg inneholder modulen en antenne, som på denne versjonen er utformet på PCB, som benytter RF matching for å justere impedansen til antennen lik impedansen til RF kretsen. Dette resulterer at antennen opererer maksimalt effektivt som gir høyere rekkevidde og lavere effektbruk. Modulen bruker også en QSPI.flashminne enhet for ikke-flyktig minne som brukes til lagring av f.eks. programkode. Størrelsen til flashminne enheten kan være 4 MB, 8 MB eller 16 MB, avhengig av modulversjonen.



Figur 2.7: Blokkdiagram ESP32-WROOM-32E[20]

I tillegg til moduler som ESP32-WROOM-32E som inneholder kun de nødvendige komponentene for at mikrokontrolleren skal være funksjonell, er det også større moduler som inneholder tilleggsutstyr som USB-til-UART-konverter og spenningskilder for 5V og 3.3V. ESP32-DevKitC V4 er et eksempel på en slik modul. Som vist i figur 2.7, inkluderer denne modulen en micro-USB port for enkel programmering av mikrokontrolleren. I tillegg bruker

denne modulen en spenningsregulator som konverterer 5V spenningen fra USB-kilden til 3.3V som brukes som forsyning til mikrokontrolleren og USB-til-UART konverteren[20]. For konverteringen fra USB-data til UART-data benytter modulen en CP2102N IC-krets. CP2102N inkluderer full støtte for USB 2.0 med overføringshastigheter opptil 3Mbaud. Grunnet konverteren har integrert USB sender/mottaker og har egen integrert klokke, trenger CP2102N svært får eksterne komponenter.



Figur 2.8: ESP32-DevKitC V4

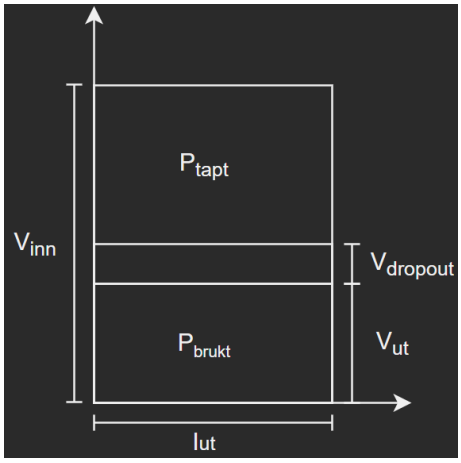
2.7 Triggersignal

Når data skal måles, er det ikke uvanlig at informasjon målt kan endre seg mye fortere enn et menneske kan reagere. Det kan også hende at hendelsen som skal måles varer i så kort tid at et menneske ikke merker at det skjer i det hele tatt. Derfor er det nødvendig når man lager måleinstrumenter å inkludere muligheten å hente data ved en endring i et signal. Triggersignaler, eller utløsende signaler, løser problemet som den menneskelige reaksjonstiden lager, der et måleinstrument begynner/slutter å hente inn data ifølge hvordan et signal endrer seg.

Triggersignaler kan skje i flere forskjellige steg av måleprosessen, og hvor i måleprosessen triggering skjer bestemmer om måleinstrumentet trenger en ekstern eller en intern triggerprosess. Ekstern triggering involverer samme prosess som diskutert tidligere, der måleutstyret kobles til både et grensesnitt for sensoren som henter data, og et triggersignal for å si når data skal hentes. Intern triggering derimot, er en prosess der måleutstyret selv bestemmer når data skal hentes[21]. Det er flere forskjellige typer av intern triggering, men de fleste typene kan reduseres til å være et system som leser fra målt data og trigger når en ønsket endring i data har skjedd. Fordelen med intern triggering er at det ikke krever grensesnitt for et eksternt triggersignal, og siden triggering skjer lokalt i måleutstyret er det mindre sannsynlig at støy i transmisjon kan gi en falsk positiv. Ulempen med intern triggering derimot, er at støy i målingen kan imitere hva triggerprosessen leter etter, som også kan gi en falsk positiv. Dette betyr at bruk av intern eller ekstern triggering må da bestemmes ut ifra nøye analyse at forventet data og støy.

2.8 Lineære spenningsregulatorer

Veldig ofte i design av kretser er det vanlig å trenge en stabil spenningskilde, og da blir spenningsregulatorer ofte brukt. Formålet med spenningsregulatorer er å gi en stabil, konstant spenning med minimal støy. Spenningsregulatorer blir oftest brukt i kretser som er sensitive til endringer i spenning, eller i kretser der spenningskilden er for høy eller ustabil. Den enkleste spenningsregulatoren er en lineær spenningsregulator, som oftest bare trenger et par kondensatorer for å fungere. Problemet med lineære spenningsregulatorer er at de er ofte en veldig ineffektiv løsning, som produserer mye varme, der en økt forskjell mellom V_{inn} og V_{ut} gjør regulatoren mindre effektiv, vist i figur 2.9.



Figur 2.9: Effektivitet av lineær spenningsregulator

Figur 2.9 viser også dropout-spenning, som er et minimum spenningskrav for at den lineære regulatoren kan regulere. Dette betyr at en 5V lineær spenningsregulator kan trenge en minimum inngangsspenning på 10V før den kan regulere[22]. Lineære spenningsregulatorer kan også ha en dropout-spenning basert på utgangsstrømmen, der en dropout-spenning på 2V ved 1A kan øke til 3V ved 2A[23]. En annen type spenningsregulator, kalt Low Dropout regulator, eller LDO, har fordelen med å ha en lavere dropout-spenning. LDO regulatorer kan regulere en spenning som er bare et par millivolt høyere enn utgangspenningen[24].

Som vist i figur 2.9 er effektiviteten mest avhengig av innspenningen, utspenningen, og strømmen. Dette betyr at spenningsregulatorer for kretser som trekker lite strøm, med en innspenning ikke mye høyere enn utspenning pluss dropout-spenning, er mye mere effektive en kretser som trekker mye strøm, med en høy innspenning.

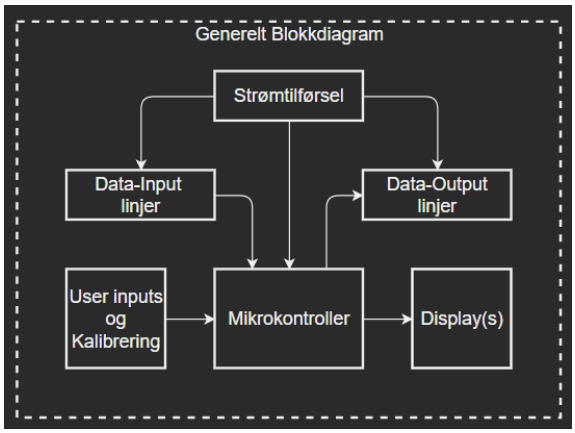
Kapittel 3

Metode

3.1 Funksjonsdiagram

Følgende blokkdiagrammer beskrevet i denne seksjonen er deler av et større blokkdiagram som kan finnes i vedlegg . Blokkdiagrammene ble diskutert og godkjent av oppgavegiver til å være de nye, mer fokuserte funksjonsbehovene.

Formålet med prosjektet er å lage testutstyr for programmerbar ammunisjon der utstyret skal få data via en radiomottaker. Mottatt data skal bli prosessert og eksportert via diverse medium og porter. Et forenklet blokkdiagram av funksjonene til utstyret er vist i figur 3.1. Dette blokkdiagrammet viser hvordan de forskjellige elementene i testutstyret mottar og sender informasjon til hverandre.



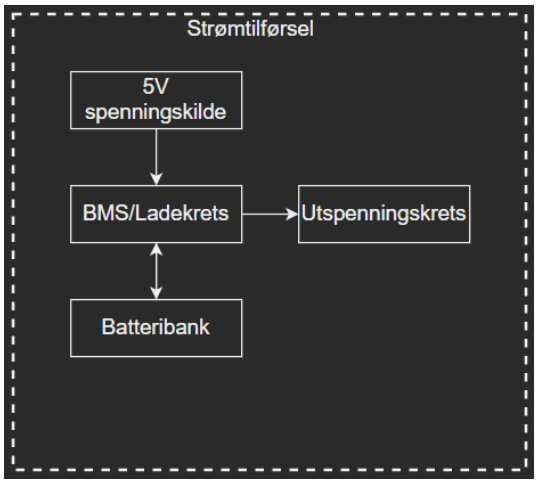
Figur 3.1: Blokkdiagram av funksjoner i testutstyret

Dette funksjonsdiagrammet blir også brukt som en hjelp for å forstå hvordan de forskjellige elementene påvirker resten av testutstyret da de blir mer nøyaktig forklart. Forklaring av de forskjellige elementene skal også hjelpe forklare hvordan kretsene til elementene er oppbyggd, og designvalg som var tatt.

Strømtilførsel

Funksjonen til strømtilførselen er å kunne produsere en spenning som spenningsregulatoren kan regulere. Det har også mål å kunne trygt lade og utlade batteripakken. Dette skal bli

oppnådd ved å dele strømtilførselen inn i forskjellige moduler som kan oppfylle kravene til kretsen. Figur 3.2 er et blokkdiagram av strømtilførselen delt inn i de forskjellige modulene.



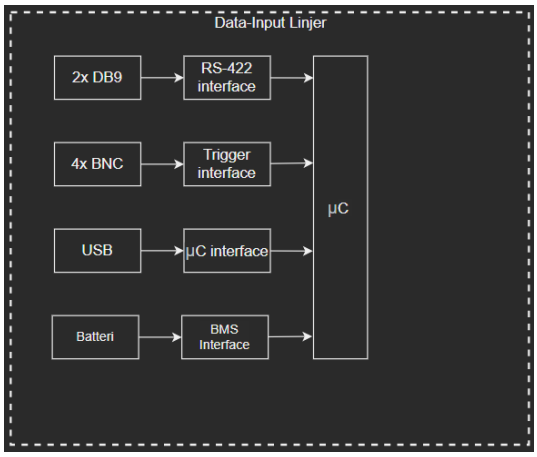
Figur 3.2: Blokkdiagram av strømtilførselen

For trygg lading og utlading av battericellene, blir en kretsmodul brukt som skal kunne kobles til en ekstern 5V spenningskilde for å trygt lade batteriene. Modulen skal også ha evnen til å trygt utlade batteriene dersom den blir koblet til en last.

Kretsen skal bruke en 5V spenningsregulator med en 5V dropout-spenning, så det er nødvendig å bruke en boost-konverter for å øke spenningen ut fra batteriene slik at strømtilførselen gir ut 10V ved den laveste batterispenningen. Denne utspenningen skal kunne justeres ved bruk av potentiometeret på modulen.

Data-input linjer

Data-input linjene er elementene som faktisk skal ta imot dataen mottatt fra de eksterne kildene. Sett under i figur 3.3 er et blokkdiagram av hvordan de forskjellige kildene kobles til kretsen, og hvordan grensesnittene blir brukt for å gjøre dataoverføringen mulig.



Figur 3.3: Blokkdiagram av linjene for innkommende data

For å bruke de digitale pinnene som RX og TX pinnene i UART-protokollen, skal det blir brukt

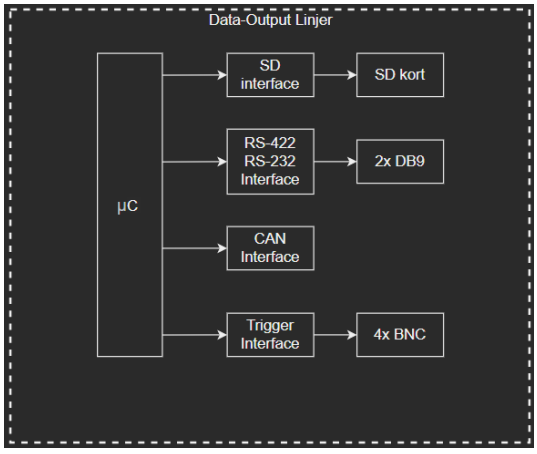
et offentlig kodebibliotek. Dette biblioteket skal da bli brukt der mer komplisert data blir sendt eller mottatt, som fra RF mottakerne som sender dataen sin via RS-422. Siden RS-422 er en seriellprotokoll som sender og mottar data over 4 transmisjonslinjer, brukes et grensesnitt for å transformere protokollen til en protokoll mikrokontrolleren lettere kan lese.

Triggersignalene skal bli sendt via en koaksial kabel som ender med en BNC kobling. Siden triggersignalene kan variere fra 3V-15V må det legges til et grensesnitt som kan kalibreres for å utløse da spenningsnivået er over en gitt spenning. Dette grensesnittet skal bruke en spenningsdeler for å redusere den innkommende spenningen med en gitt faktor, og skal bruke filtrerte PWM signaler for å lage en terskelspenning. Ved å bruke en komparator kan den reduserte triggerspenningen og det filtrerte PWM signalet sammenliknes og da gi ut enten 3.3V eller 0V basert på forholdet.

Det siste grensesnittet for innkommende data er måling av batterispenningen. Dette grensesnittet skal bestå av en pinne på mikrokontrolleren som leser spenningen på batteriet. Siden batterispenningen kommer til å falle dersom batteriet blir mer brukt, kan batterinivået kalkuleres og vises på en skjerm. Dette kan også brukes for å unngå å utlade batteriet for mye som kan være skadelig for batteriene.

Data-output linjer

For å kunne eksportere data som måleinstrumentet henter, er det nødvendig med et grensesnitt liknende de grensesnittene brukt for innkommende data. Et liknende blokkdiagram for kobling mellom utgående datalinjer og mikrokontroller kan bli sett i figur 3.4:



Figur 3.4: Blokkdiagram av linjene for utgående data

Et av målene for måleinstrumentet er å kunne overføre dataen til en datamaskin, og en av de valgte metodene for dataoverføring ble valgt å være SD-kort. Fordelen med å bruke et SD-kort er at det øker det begrensede minnet til mikrokontrolleren, som gjør det mulig å bevare resultater fra flere testforsøk. Ved å bruke et SD-kort blir det også lettere å lese data på en datamaskin. Grensesnittet mellom SD-kortet og mikrokontrolleren er bestemt å være en modul med en lineær 3.3V LDO spenningsregulator, som er nødvendig siden SD-kort er sensitive for spenninger høyere enn 3.3V. På grunn av denne sensitiviteten trenger grensesnittet en logikk-nivå skifter, som er en spenningsregulator for individuelle datalinjer[25]. Fordelen med å bruke en logikk-nivå skifter over spenningsregulatorer er at er

LLS har som oftest en mye raskere responstid, så den kan regulere mye høyere frekvenser enn en lineær spenningsregulator. Grensesnittet trenger ingen andre integrerte kretser, siden alle SD-kort allerede har et SPI [15] grensesnitt, som mikrokontrollere kan kommunisere med.

Likt som data-input linjene, skal data-output linjene sende data via UART og seriellprotokoller. Grensesnittene som mottar data fra mikrokontrolleren skal lese utgående data fra samme TX linje, som betyr at både RS-422 og RS-232 kontaktene skal sende samme data, men via de forskjellige protokollene. Grensesnittet for CAN-bussen er liknende grensesnittet for seriellportene, der en IC kommuniserer med mikrokontrolleren via seriell kommunikasjon.

Måleinstrumentet skal kunne kommunisere videre med annet utstyr, så det kreves et grensesnitt for å kunne sende triggersignaler videre i et nettverk. Dette grensesnittet skal bruke en MOSFET for å sette en spenning høy eller lav, kontrollert av et signal fra mikrokontrolleren. En MOSFET blir brukt for å unngå at enheten som kobler til trigger utsignalet trekker strømmer for høye for mikrokontrolleren.

Display og Brukerinput

Måleinstrumentet skal vise informasjon og data via to LCD-skjermer, der LCD-skjermene skal ha forskjellige funksjoner for grensesnittet. Den øverste LCD-skjermen skal vise grunnleggende informasjon om de forskjellige elementene i kretsen, som batterinivå, terskelspenning for de forskjellige triggerportene og om en test er aktiv eller ikke. Den nederste skjermen skal vise de forskjellige menyfunksjonene som brukeren kan velge via de fire knappene på siden. Skjermene som skal brukes er begrenset i hvor mye informasjon de kan vise, der hver skjerm har fire linjer med maksimalt 20 karakterer hver[26]. Denne begrensingen betyr at det er viktig å være nøye med hva slags informasjon som vises.

Menyfunksjoner

Det skal være totalt 4 hovedmenyer tilgjengelig da testutstyret blir slått på, oppgitt som 1-4 i listen under. Siden det er bare 4 linjer med informasjon på skjermen skal ingen menyer ha flere enn 4 menyer eller funksjoner i seg.

1. Ny Test
    a. Stopp og lagre test
2. Eksporter data
    a. Eksport til seriell
    b. Tilbake
3. Triggerinnstillinger
    a. Trigger inn status
        i. Bytt Port A status
        ii. Bytt Port B status
        iii. Tilbake
    b. Juster trig.Spenning

        i. Juster Port A
        ii. Juster Port B
        iii. Tilbake
    c. Trigger ut status
        i. Bytt Port A status
        ii. Bytt Port B status
        iii. Tilbake
    d. Tilbake
4. Datainnstillinger
    a. SD-kort
        i. Slett kort
        ii. Tilbake
    b. Tilbake

Disse menyfunksjonene oppnår de ønskede funksjonene og innstillingene planlagt. Menyen har også ha plass for flere funksjoner som kan legges til i fremtiden.

3.2 Design av krets

For å forklare de forskjellige elementene i kretsen blir den delt inn i deler liknende elementene i funksjonsdiagrammet som forklarte funksjonene til elementene. Hele kretsen finnes som vedlegg 3.

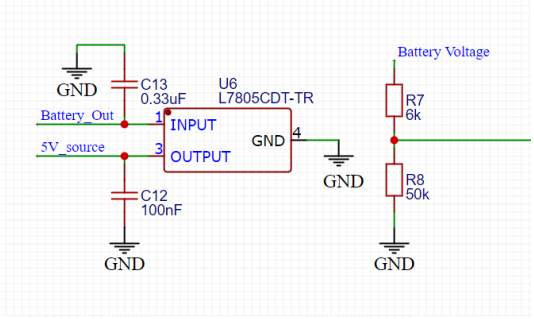
Valg av mikrokontroller

Grunnet behovet for å kunne behandle data tilsendt fra en RF-modul, var det nødvendig å benytte en mikrokontroller. Derimot er det på nåværende tidspunkt en global mangel på mikrokontrollere og andre integrerte kretser. Dette gjorde valg av mikrokontroller vanskelig da mange mikrokontrollere lå på restordre til 2023. Dermed måtte mikrokontrolleren ikke bare ha støtte for alle funksjonene som trengs, men i tillegg også være tilgjengelig. Etter en del undersøking ble det bestemt å bruke et ESP32-modulkort. Dette er en mikrokontroller som gruppemedlemmene er godt kjent med fra før i tillegg at det også finnes mye støtte på Internett i form av kodebiblioteker og artikler. For å programmere mikrokontrolleren, benyttes den innebygde ISCP-chipen på ESP32-devkit modulen. Modulen programmeres ved å koble modulkortet til en PC hvor program som Arduino IDE kan brukes for kompilering og opplasting til mikrokontrolleren. ESP32 er en kjent mikrokontroller ofte brukt til IoT-prosjekter grunnet dens støtte for WiFi og Bluetooth. Det er ikke noe funksjonsbehov som krever hverken WiFi eller Bluetooth, men grunnet mikrokontrollerens antall av GPIO-pinner og dens andre egenskaper som hastighet og minne, dekker ESP32 behovene som mikrokontroller.

Batteri

Kretsen kobler til batteriet via tre kontakter: Batterispenning, konvertert batterispenning og jord. Batterispenningen blir brukt for å måle batterinivået, og den konverterte

batterispenningen skal være spenningskilden for den lineære spenningsregulatoren. Batteridelen av kretsen er vist i figur 3.5.

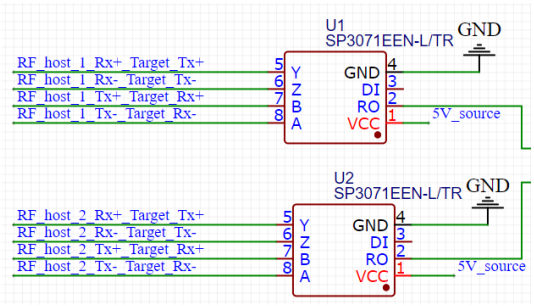


Figur 3.5: Krets for spenningskilde og spenningsregulator

Her er Battery_Out flagget den boost-konverterte spenningen ut fra batteriet, som blir regulert til 5V_source flagget som er 5V kilden til hele kretsen. Flagget Battery_Voltage er spenningen fra batteriet før det var konvertert som brukes for å regne ut batterinivå. Mikrokontrolleren har en ADC som kan lese spenninger lavere enn 3.3V, som er lavere enn den maksimale 3.7V spenningen fra batteriet. For å løse dette er batterispenningen delt med en faktor av $\frac{50k}{56k}$ som deler batterispenningen fra 3.7V til 3.3V. Siden Batteriet er tomt ved 3V, skal batterinivået vise 0% ved en målt spenning av 2.7V.

Seriell

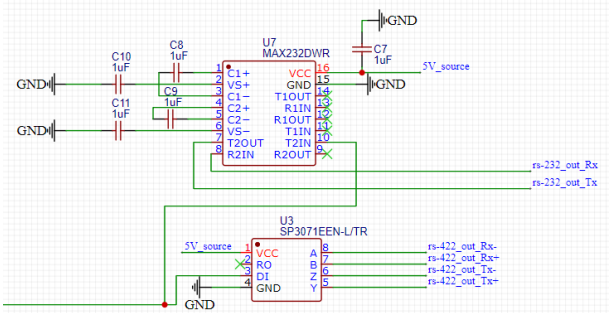
Testutstyret skal motta data fra RF mottakeren over en seriell-kobling ved hjelp av RS-422 protokollen. For å konvertere mellom RS-422 protokollen og UART protokollen mikrokontrolleren bruker, blir en SP3071EEN-L IC brukt[27]. Denne chipen blir valgt grunnet dens evne for å effektivt konvertere mellom seriellprotokollen til mikrokontrolleren og RS-422. Kretsen for innkommende seriell data kan bli funnet i figur 3.6.



Figur 3.6: Krets for seriell inn

Et økende problem var mangelen på tilgjengelige GPIO-pinner på mikrokontrolleren, så for å redusere bruk av pinner var noen unødvendige pinner lagt flytende. Siden det bare skal være innkommende informasjon fra RF-modulen, blir TX-pinnene på mikrokontrolleren aldri brukt for den seriellkoblingen Dette er grunnen til at pinnen som kobles til RX på mikrokontrolleren er de eneste som kobles til.

For å forenkle kretsen, koden, og brukergrensesnittet, var det bestemt at begge protokollene for seriell ut skulle sende samme data samtidig. En fordel med at begge protokollene skal sende samme data er at begge integrerte kretsene som skal konvertere dataen kan lese data fra samme pin. I tillegg til å lese data fra samme pin, trenger ikke seriell ut tilkoblingene for å motta data, som betyr at det ikke er nødvendig å koble til mottakerpinner. Alle de nevnte faktorene betyr at det bare er behov for en GPIO-pinne for all serielldata som sendes, som sett i kretsen i figur 3.7.



Figur 3.7: Krets for seriell ut

Protokollen for RS-232 krever positive og negative spenninger for transmisjon av data, som spenningskilden ikke produserer. Derfor er det nødvendig at den integrerte kretsen som driver RS-232 protokollen kan også produsere negative spenninger fra en positiv spenningskilde. Ut ifra dette kravet var den integrerte kretsen MAX232DWR[28] valgt. Denne chipen er en effektiv RS-232 driver, sender og mottaker som kan konvertere positive til negative spenninger. For å produsere de negative spenningene bruker den integrerte kretsen en negativ ladningspumpe, eller negative charge pump. Negative ladningspumper bruker PWM signaler til å lade og utlade kondensatorer slik at andre kondensatorer i kretsen lader med negative spenninger relativt til jord. I figur 3.7 er disse kondensatorene navngitt C8-C11 og har alle en verdi av $1\mu F$. Siden kretsen er sensitiv for endringer i spenning, er en $1\mu F$ bypasskondensator brukt for å filtrere ut støy eller raskt endrende komponenter av kildespenningen. Bypasskondensatoren er ofte ikke nødvendig i kretser med en regulert spenningskilde, men er nyttig dersom det oppstår interferens fra andre elementer eller komponenter i kretsen. Den integrerte kretsen har to drivere som gjør at den kan konvertere to seriellkoblinger samtidig, men bare en kommer til å bli brukt i denne kretsen.

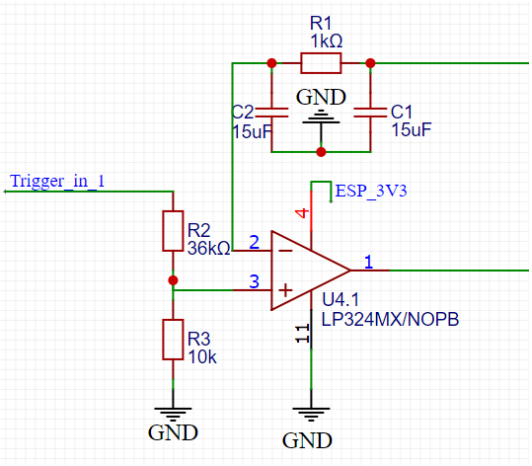
Trigger

Målet med input triggerkretsen er å kunne sammenlikne en inputspenning med en terskelspenning, der grensesnittet skal gi ut en binær 1 dersom inngangsspenningen er høyere, og en binær 0 dersom terskelspenningen er høyere. Sammenlikningen av spenningene gjøres av en op-amp koblet som en komparator[29]. En op-amp brukt som en komparator gir maksimal kildespenning dersom spenningen inn i den ikke-inverterende pinnen har en spenning høyere enn det på den inverterende pinnen. Dersom spenningen på den inverterende pinnen er høyere enn spenningen på den ikke inverterende gir den ut laveste spenningen tilgjengelig, som kan være en negativ spenning eller jord.

Terskelspenningen produseres ved å sende et PWM-signal gjennom en lavpassfilter for å få DC-komponenten av signalet. Dette signalet lader opp en kondensator som bevarer spenningen som måles av komparatoren. Dersom et 3.3V PWM-signal har en pulsbredde på

100%, er terskelspenningen tilnærmet til å være 3.3V. Dette forholdet mellom pulsbredde og terskelspenning er lineært fra 0% (0V) til 100% (3.3V).

Triggerspenningen som skal sammenliknes med terskelspenningen varierer fra 3V til 15V, som er for høye spenninger enn det kretsen kan tåle. Derfor må triggerspenningen gå gjennom en spenningsdeler for å maksimalt gi 3.3V. Forholdet for spenningsdeleren var kalkulert til å være $\frac{10k}{46k}$, der den maksimale triggerspenningen på 15V blir delt ned til 3.26V. Kretsdiagrammet av en slik triggermodul vises i figur 3.8.



Figur 3.8: Trigger inn modul

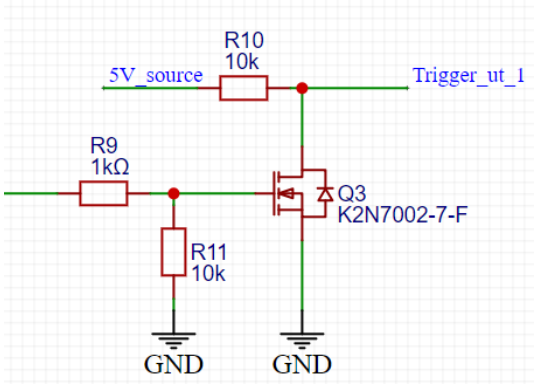
Kretsen i figur 3.8 viser hvordan triggerkretsen var koblet opp for å oppnå ønsket resultat. Triggerspenningen blir delt av motstandene R2 og R3 i forholdet $\frac{10k}{46k}$, som da går inn i den ikke-inverterende pinnen på op-ampen. PWM-signalet blir filtrert av lavpassfilteret laget av R1 og C1, med knekkfrekvens gitt av likning 3.1 og 3.2.

$$f = \frac{1}{2\pi RC} \tag{3.1}$$

$$\frac{1}{2\pi 1k\Omega * 15\mu F} = 10.6Hz \tag{3.2}$$

Et filter med lav knekkfrekvens filtrerer ut de høyere frekvensene, og beholder de lavere frekvensene og DC komponenten. DC komponenten lader kondensator C2, som holder spenningen inn til den inverterende pinnen til op-ampen. Dersom triggerspenningen er høyere enn terskelspenningen gir op-ampen ut 3.3V, og ellers gir den ut 0V.

Måleinstrumentet skal kunne sende ut triggersignaler i tillegg til å kunne motta triggersignaler, så et ekstra grensesnitt er benyttet for å oppnå dette. Ut fra testutstyret skal det sendes ut et 5V triggersignal, da dette var et ønske fra oppgavegiver. Den høyeste spenningen GPIO-pinnene på mikrokontrolleren kan gi ut er 3.3V, og kan ikke trekke mere enn 50mA. Grensesnittet må da kunne ta et signal fra mikrokontrolleren og konvertere det til et 5V triggersignal med et høyere maksimalt strømtrekk. Løsningen for triggerspenningen var å bruke en N-kanal MOSFET for å kunne sende 5V kildespenningen som triggersignalet, som kunne kontrolleres av mikrokontrolleren med en GPIO-pinne uten høye strømtrekk. Kretsdiagrammet av trigger ut vises i figur 3.9 under.



Figur 3.9: Kretsdiagram av trigger ut port

I denne kretsen skal trigger ut være vanligvis på, siden V_{GS} på MOSFET ikke er positiv. Dersom et signal sendes til gaten til MOSFET Q3 som gjør V_{GS} positiv, skal trigger ut heller kortsluttes til jord og ha en utspenning på 0V. For at testutstyret da skal sende et triggersignal, må signalet fra mikrokontrolleren slås av for at kortslutningen til jord blir frakoblet. Et problem med noen mikrokontrollere er at når en pinne slås av betyr det ikke at spenningen ut fra pinnen er 0V, men at mikrokontrolleren heller ikke prøver å holde den på en spesifikk spenning. I det tilfellet at mikrokontrolleren slår pinnen av i stedet for å ha en spenning på 0V var en $10k\Omega$ pull-down motstand (R11) nødvendig for at gaten til Q3 ikke ble aktivert av støy eller interferens.

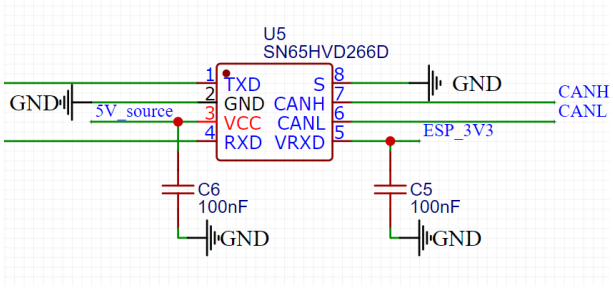
Display og Brukerinput

Det var bestemt at to LCD-skjermer skulle bli brukt for å vise informasjonen og menyfunksjoner fra testutstyret. Det oppstod et problem derimot, der de fleste LCD-skjermene tilgjengelig trengte 6 GPIO-pinner for å operere riktig, som er et problem grunnet antall GPIO-pinner tilgjengelig. Løsningen for dette problemet var å bruke skjermer med I2C grensesnitt, som krever bare 2 pinner og kan kobles i bus-topologi. Siden LCD-skjermene kunne kontrolleres via I2C, var det ikke nødvendig å lage noe grensesnitt for dem. For å koble skjermene til mikrokontrolleren var det bare nødvendig å koble pinnene fra skjermen til 5V, jord, I2C SCL, og I2C SDA.

Siden knappene på brukergrensesnittet skal kunne kontrollere alle porter og funksjoner i hele testutstyret, er det viktig å implementere de slik at støy eller interferens ikke påvirker knappene når de ikke trykkes. For å oppnå dette blir pull-down motstander koblet til pinnene til mikrokontrolleren som leser statusen til knappene..

CAN-bus

Grensesnittet for CAN-bus var implementert liknende seriellprotokollene, der en IC kunne motta UART data fra mikrokontrolleren og konvertere det til CAN-bus protokollen. Forskjellen mellom grensesnittet for seriell og CAN er at CAN skal motta og sende data samtidig, som gjør at både RX og TX pinnene ble brukt. Siden CAN-bus ikke er en protokoll som mikrokontrolleren har innebygd støtte for, blir det nødvendig å bruke et offentlig bibliotek for å gjøre rede for kommunikasjon mellom mikrokontrolleren og den integrerte kretsen. Figuren 3.10 viser hvordan den integrerte kretsen SN65HVD var implementert for å gjøre CAN-bus mulig.



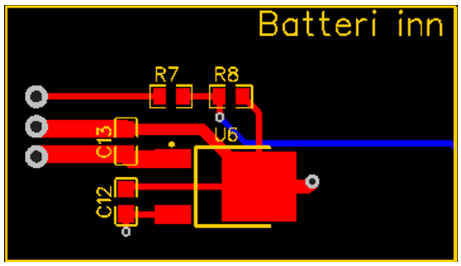
Figur 3.10: Grensesnitt for CAN-bus

3.3 Utlegg av PCB

Utlegget til kretskortet var designet likt som funksjonsdiagrammet, der kretsen er delt inn i moduler som kobler til en sentral mikrokontroller. Ved å dele en slik krets inn i moduler blir det lettere å unngå uønsket interferens mellom to elementer. I kretsen er det to forskjellige kildespenninger: 5V som blir regulert der batteriene kobles til kretsen, og 3.3V som blir regulert i mikrokontrolleren og kan hentes via 3.3V pinnen. 5V, 3.3V, og jord skal ha egne lag i kretskortet, der det øverste laget skal være mesteparten av banene. Dette betyr at kretskortet skal ha fire lag som gjør utlegg lettere, siden elementer kan kobles til spenningslagene ved bruk av et via-hull. Det var også bestemt tidlig i prosjektet at kretskortet skulle bruke overflateloddet komponenter i 0805 størrelse, som gir god balanse av størrelse og tilgjengelighet for lodding. Utlegg av hele kretskortet er lagt til som vedlegg 4 og 5.

Kildespenning

Formålet med spenningskretsen var å både regulere en boost-konvertert batterispenning, og måle batterinivået. Dette elementet av kretsen skal levere en regulert 5V kildespenning til resten av kretsen, i tillegg til å være koblingspunktet for jord mellom kretsen og batteriet. Et kretsdiagram av kildespenningen kan bli sett i figur 3.11

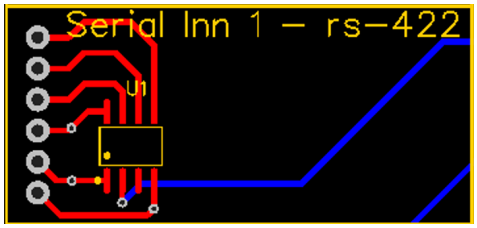


Figur 3.11: Utlegg av batteri

Den øverste pinnen tilsvarer den ikke-konverterte batterispenningen, pinnen i midten tilsvarer jord, og den siste er den konverterte batterispenningen. Via-hullet som kobler jord til spenningsregulatoren tilsvarer punktet der jordplanet kobles til batterikretsen, derfor måtte både banen og via-hullet være større enn de andre.

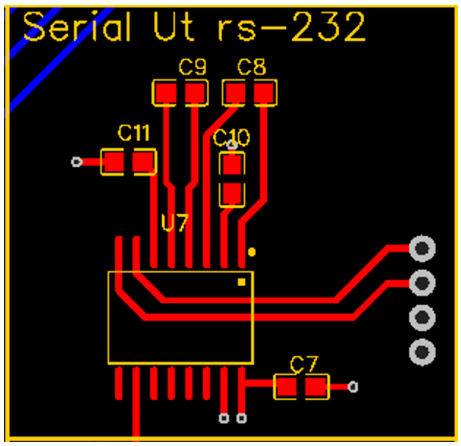
Seriell

Den integrerte kretsen som konverterer RS-422 protokollen til UART protokollen mikrokontrolleren bruker trenger ingen andre komponenter for å drives. Dette betyr at bare chipen er nødvendig for oversetting for seriell inn og ut, som gjør at alle RS-422 seriellportene bruker samme oppsett som kretsen i figur 3.12, med bare bytting av RX og TX pinne.



Figur 3.12: Utlegg av seriell port

Kretsen for å konvertere serielldata fra mikrokontrolleren til RS-232 er litt mere komplisert, der ladningspumpen trenger kondensatorer for å produsere de negative spenningene. Det var også iterert i databladet til den integrerte kretsen at bypasskondensatoren skal kobles til jord nærme chipen for å unngå at interferens forstyrrer spenningsnivået til transmisjonen. Figur 3.13 viser utlegget for seriellporten for RS-232.

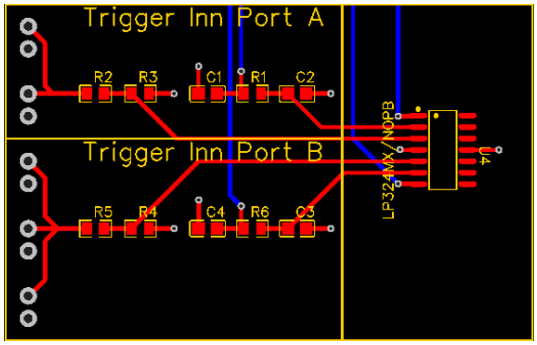


Figur 3.13: Utlegg av RS-232 for seriell

Utlegget viser de to pinnene, RX og TX , i tillegg til 5V og jordpinner. Kondensator C7 er bypasskondensatoren som stabiliserer inngangsspenningen til chipen, i tillegg til kondensatorene C8-C11 som brukes i ladningspumpen.

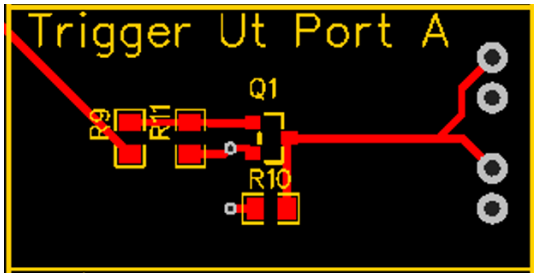
Trigger

Det var bestemt at 8 BNC kontakter for trigger skulle brukes i testutstyret, i tillegg til en 8-pin 2M805 konnektor, lagt til som vedlegg 9. Fire av BNC kontaktene skulle være for trigger inn, og fire skulle være for trigger ut, der både trigger inn og trigger ut skulle deles inn i 2 porter som kunne operere individuelt i forbindelse med de to seriellportene.



Figur 3.14: Utlegg for trigger inn

Figur 3.14 viser utlegget for begge triggerportene inn i utstyret. trigger port A har totalt 2 innganger som tilsvarer de to BNC kontaktene som kobles til den porten, og Trigger port B har 3 innganger, som inkluderer 8-pin 2M805 kontakten i tillegg til to BNC kontakter. PWM signalet fra mikrokontrolleren kommer via de blå banene inn i motstandere R1 og R6, som sammen med C1 og C4 henholdsvis skaper to lavpassfiltre. C2 og C3 holder terskelspenningen ut fra lavpassfiltrene, som begge går inn i de inverterende pinnene i op-ampen. LP324MX[30] er en integrert krets som inneholder 4 op-amper, som får lik maksimum og minimum inngangsspenning. Op-ampene som blir brukt trenger begge en inngangsspenning på 3.3V, som er grunnen til at LP324MX var brukt. R2 og R3 sammen med R5 og R4 lager spenningsdelerne for triggersignalene, som går inn i hver sin ikke-inverterende pin på hver sin op-amp.



Figur 3.15: Utlegg for trigger ut

Figur 3.15 viser hvordan trigger ut portene bruker en MOSFET for å kontrollere trigger ut signalene til begge BNC-kontaktene som er tilkoblet. R11 tilsvarer $10k\Omega$ pull-down motstanden som forsikrer at V_{GS} er null dersom det ikke kommer et signal inn i gaten til MOSFET-transistoren, og R10 tilsvarer motstanden som kobler 5V til drain-pinnen. Siden trigger ut portene bruker bare BNC-plugger for kontakter, er begge trigger ut portene identiske utenom hvilke GPIO-pinner som kobler de til mikrokontrolleren.

Display og Brukerinput

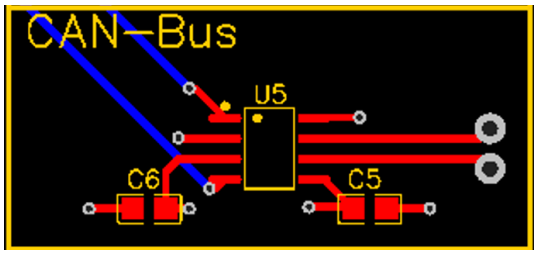
Måten brukeren kan kontrollere funksjoner for testutstyret er gjennom navigasjon av menyer ved bruk av knapper ved siden av skjermene. Skjermene kobles til mikrokontrolleren via I2C, som kontrollerer begge skjermene samtidig. Knappene er satt til å være aktivt-høye og dermed blir det benyttet pull-down motstander. Dersom brukeren trykker på knappen skal GPIO-pinnen heller være kortsluttet til 3.3V, som kan leses som en binær 1 i mikrokontrolleren. Figur 3.16 viser hvordan knappene skal kobles til kretskortet.

Figur 3.16: Utlegg av knapper for brukerinput

Motstandene R15-R18 tilsvarer pull-down motstandene, som kobles til jord gjennom via-hullene plassert over. Den nederste raden med pinner er koblet til 3.3V spenningskilden som knappene tilkobler til de øverste pinnene koblet til mikrokontrolleren.

SD-kort og CAN-bus

SD-kortet er en SPI-enhet som betyr at kortet inneholder et innebygd SPI-grensesnitt. På grunn av dette kunne et SD-kort kobles direkte til mikrokontrolleren. En SD-kort modul med spenningsregulator og logikk-nivå skifter var derimot brukt for å forsikre at en rask uventet økning av spenningen ikke kan skade kortet.



Figur 3.17: Utlegg for CAN-bus

Kretsen for CAN-bus chipen var tilkoblet som anbefalt i databladet[31], der bypasskondensatorene C5 og C6 er plassert nærme 3.3V og 5V kildespenningene henholdsvis. Kontakter for tilkobling av CAN trenger ikke en spenningskilde inkludert, så bare kontakter for pinnene CANH og CANL var plassert.

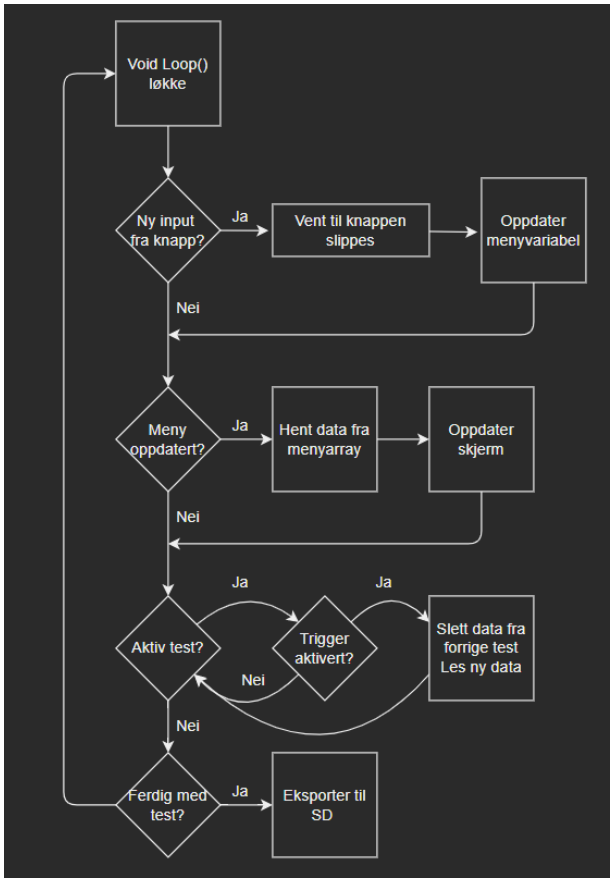
3.4 Kodeforklaring

Koden er ansvarlig for logikken og funksjonaliteten som ikke kunne gjøres av kretsen, eller for funksjoner laget med hensikten for å kunne endres senere. Hva som kan programmeres for dette prosjektet er begrenset til hva som kan testes, der noen protokoller i kretsen ikke kan testes med tilgjengelig utstyr. Funksjonaliteten som ikke kan testes pålitelig kan dermed ikke programmeres i den siste versjonen av testutstyret, og blir igjen som en videre arbeid for

oppgavegiver. Målet med koden, med tanke på begrensningene, er da å programmere funksjonalitet for følgene grensesnitt og kretselementer:

- Motta data via seriell inn
- Justere terskelspenning for trigger
- Datainnsamling utløst av triggergrensesnitt
- Motta og behandle data fra to uavhengige kilder samtidig
- Vise informasjon angående testutstyret på en skjerm
- Navigere menyer på en skjerm ved bruk av knapper.
- Eksportere data til et SD-kort
- Måle og vise batterinivå som en prosentandel
- Eksportere data til en datamaskin

Et blokkdiagram ble tegnet for enklere oversikt av programmering og forklaring av kode. Dette blokkdiagrammet forklarer loop funksjonen i koden, som er delen av koden som repeterer så lenge mikrokontrolleren er på. Figur 3.18 viser et blokkdiagram av hvordan koden fungerer, og utelater mindre detaljer som blir diskutert videre i denne seksjonen.



Figur 3.18: Blokkdiagram av koden

Det er generelt foretrukket at en knapphandling oppstår på synkende flanke i stedet på stigende flanke, så koden registrerer et knappetrykk dersom knappen slippes. Da knappen for en menyfunksjon er sluppet, blir variabelen som bestemmer hvilken meny som vises for øyeblikket endret. Denne funksjonen informerer meny-skjermen at ny informasjon skal vises, og må oppdatere med de nye menyelementene.

Dersom skjermene skulle oppdateres hver gang løkken repeterer, blir det mye flimring fra den korte tiden det tar å oppdatere skjermen, som skjer flere ganger per sekund. For å unngå flimring skal skjermene bare oppdatere dersom det er noen oppdateringer av informasjonen som vises.

Da en ny test starter skal mye informasjon komme inn i testutstyret veldig fort, så det er viktig å programmere en testprotokoll som ikke sløser tid på unødvendige funksjoner som ikke har noen innflytelse på testen. Dermed skal en ny løkke startes samtidig som en ny test starter, som er programmert for å bare plukke opp data etter triggering. Når en test er utført, blir dataen lagret permanent til et SD-kort som videre kan bli behandlet på PC.

Skjermene brukt i prosjektet er ikke kompatible med de vanlige bibliotekene tilgjengelig, så for å kommunisere med de må individuelle HEX-kommandoer bli sendt over I2C. Det et viktig å huske at funksjonene disse kommandoene utføres trenger tid for å gjennomføres, så data kan ikke sendes til disse skjermene før kommandoen er utført. Kommandoene brukt kan finnes i tabell 3.1.

Prefiks	HEX-kommando	Parameter	Beskrivelse av kommando	Tid for utføring
0xFE	0x45	1 Byte	Beskriver hvor tekst plasseres	$100\mu s$
		1 Byte	Skriver tekst ved markør (0x45)	$100\mu s$
0xFE	0x51	0 Bytes	Tømmer skjermen	$1,5ms$
0xFE	0x62	1 Byte	Lagrer ny I2C adresse	$3ms$
0xFE	0x72	0 Bytes	Viser I2C adresse	$4ms$

Tabell 3.1: HEX-koder og funksjoner

Skjermene kommer fra produsent med samme I2C adresse (0x18), så adressene må byttes for å kunne kommunisere med de individuelt.

```
const byte LCD_func = 0x18, LCD_info=0x28;
const byte line[4]={0x00, 0x40, 0x14, 0x54};

const char *funcMenu[11][4] = {
{"Ny Test",              "Eksporter Data",      "Triggerinstillinger", "Datainstillinger"},
{"Stopp og lagre Test",  "-",                    "-",                  "-"},
{"Eksporter til Seriell", "-",                   "-",                  "Tilbake"},
{"Trigger inn Status",   "Juster Trig.Spenning", "Trigger ut Status",  "Tilbake"},
{"Bytt Port A Status",   "Bytt Port B Status",   "-",                  "Tilbake"},
{"Juster Port A",        "Juster Port B",        "-",                  "Tilbake"},
{"Opp",                  "Ned",                  "-",                  "Tilbake"},
{"Opp",                  "Ned",                  "-",                  "Tilbake"},
{"Bytt Port A Status",   "Bytt Port B Status",   "-",                  "Tilbake"},
{"SD-kort",              "-",                    "-",                  "Tilbake"},
{"Slett Kort",           "-",                    "-",                  "Tilbake"},
};
```

Kodeliste 3.1: Matrise med menyvalg

Den første linjen i koden beskriver de forskjellige erklærer variabler med de forskjellige adressene etter de var ender. Deretter er HEX-adressen til de fire linjene bevart i en array. Den store matrisen beskriver de forskjellige menyvalgene som skal vises, der alle valgene i

menyen kan finnes i samme rad.

```
using namespace std;
typedef float (*functionPointer) (float a);

functionPointer menuFunctions[][4] = {
{beginCapture,          returnA,                returnA, returnA},
{returnA,               returnA,                returnA, returnA},
{returnA,               returnA,                returnA, returnA},
{returnA,               returnA,                returnA, returnA},
{byttTriggerInn,        byttTriggerInn,         returnA, returnA},
{returnA,               returnA,                returnA, returnA},
{setTriggerVoltageUpA,  setTriggerVoltageDownA, returnA, returnA},
{setTriggerVoltageUpB,  setTriggerVoltageDownB, returnA, returnA},
{byttTriggerUt,         byttTriggerUt,          returnA, returnA},
{returnA,               returnA,                returnA, returnA},
{returnA,               returnA,                returnA, returnA},
};
```

Kodeliste 3.2: Funksjonsmatrise for meny

Denne matrisen gjør det mulig å legge til funksjonalitet til de forskjellige menyvalgene, der hvert element i matrisen er en peker til en funksjon. Ved å gjøre det på denne måten kan en menyfunksjon både returnere en verdi som tilsvarer adressen til en meny, og utfør en funksjon. Ulempen med å bruke en funksjonsmatrise er at det er nødvendig å knytte hvert element i matrisen med en funksjon, selv der det ikke er krav for funksjon i menyen. For å løse dette problemet var funksjonen returnA laget, som returnerer verdien til variabelen brukt i argumentet til funksjonen.

```
A.triggerInPWMPin = 12;
B.triggerInPWMPin = 14;
A.triggerInReadPin = 13;
B.triggerInReadPin = 35;

pinMode(A.triggerInPWMPin, OUTPUT);
pinMode(B.triggerInPWMPin, OUTPUT);
pinMode(A.triggerInReadPin, INPUT);
pinMode(B.triggerInReadPin, INPUT);
```

Kodeliste 3.3: Erklæring av triggerpinner

Programmet til mikrokontrolleren er delt i to deler. Den første delen, setup, skal bare kjøre en gang. Loop, den andre delen av programmet, skal repetere uendelig så lenge mikrokontrolleren er på og annen kode ikke forstyrrer løkken. Oftest skal setup-delen av koden indentifisere de forskjellige måtene mikrokontrolleren samhandler med elementene i kretsen den er koblet til. I koden for dette prosjektet var setup koden brukt for å registrere hvordan trigger og knappene skulle fungere. I eksempelet over er triggerpinnene først bevart i variablene, så blir pinnene registrert som input eller output.

```
but1.buttonVal=digitalRead(but1.buttonPin);

if(but1.buttonVal!=but1.lastButtonVal){
    if(but1.buttonVal == 0){
        menulevel=menuFunctions[menulevel][0](menuAddr[menulevel][0]);
        funcScreenUpdate = 1;
    }
    delay(10);
}
```

```
but1.lastButtonVal = but1.buttonVal;
```

Kodeliste 3.4: Lesing av knapptilstand

Loop løkken starter med å først lese tilstanden til knappene, og lagrer verdien i buttonVal variabelen. Deretter sjekker koden om tilstanden til knappene har endret seg fra forrige løkke. Siden knappen skal registrere knappetrykket på fallende flanke, er menyvalget bare registrert da både knappverdien har byttet tilstand, og den nye tilstanden er 0 (åpen). Menyen som skal vises er gitt av variabelen menulevel, som endres av matrisene menuFunction og menuAddr. MenuFunctions er matrisen som gir funksjonalitet til menyen som diskutert tidligere, og menuAddr er en matrise som inneholder adressene til menyene knappene tilsvarer. Eksempelet over bruker knapp 1, som betyr at adressen til menyen som knapp 1 hører til kan hentes med funksjonen `menuAddr[menulevel][0]`. Adressen som tilsvarer knapp 2 kan finnes på liknende måte med `menuAddr[menulevel][1]`, osv. For å bruke menuFunctions matrisen, må det legges til et argument i linjen funksjonen er brukt, som `menuFunctions[a][b](c)` der a og b tilsvarer en posisjon i matrisen, og c tilsvarer argumentet til funksjonen i posisjon [a,b]. For plassene i matrisen menuFunction som inneholder funksjonen returnA, returnerer funksjonen bare adressen som blir brukt for argumentet, som da lagres som den nye verdien til menulevel. Andre funksjoner kan returnere menulevel som betyr at funksjonen har kjørt men menyen ikke skal endres.

Til slutt forteller variabelen funcScreenUpdate at meny-skjermen har ny informasjon den skal vise, og en 10*ms* forsinkelse passer på at knappen ikke spretter. Den forrige knappverdien blir også endret til den nåværende knappverdien, som gjør at knappehandlingene kan bli programmert i neste løkkeomgang.

```
if(infoScreenUpdate==1){
    Wire.beginTransmission(LCD_info);
    Wire.write(0xFE);
    Wire.write(0x51);
    Wire.endTransmission();
    delay(2);

    //linje1 konstrueres

    Wire.beginTransmission(LCD_info);
    Wire.write(0xFE);
    Wire.write(0x45);
    Wire.write(line[0]);
    Wire.write(line1);

    //resten av linjene konstrueres og sendes

    Wire.endTransmission();
    infoScreenUpdate=0;
}
```

Kodeliste 3.5: Oppdatering av informasjonsskjerm

Over er et eksempel av hvordan den øverste linjen på informasjonsskjermen blir oppdatert dersom informasjonen på skjermen har endret seg. Eksempelet kuttet ut fra koden er hvordan teksten på den første linjen (strengen linje1) er laget fra informasjon hentet fra diverse variabler, i tillegg til de andre to linjene.

Etter opprettelse av en I2C tilkobling mellom skjermene og mikrokontrolleren blir HEX-kommandoen 0x51 sendt for å tømme alt innhold på skjermen. Denne kommandoen tar

1.5*ms* å utføre, så programmet venter 2*ms* før mer informasjon sendes. Etter skjermen er tømt starter I2C tilkoblingen igjen, og kommandoen 0x45 sendes for å begynne å skrive. Neste HEX-kommando funnet i linje[0] er HEX-adressen for første karakter på første linje, fulgt av strengen linje1 som inneholder hva som skal stå i det feltet. Til slutt stopper mikrokontrolleren tilkoblingen til skjermen, og endrer variabelen infoScreenUpdate til å fortelle at skjermen ikke trenger å oppdatere igjen før ny data kommer inn.

```
if(funcScreenUpdate==1){
    Wire.beginTransmission(LCD_func);
    Wire.write(0xFE);
    Wire.write(0x51);
    Wire.endTransmission();
    delay(2);

    for(int i = 0; i<4; i++){
        Wire.beginTransmission(LCD_func);
        Wire.write(0xFE);
        Wire.write(0x45);
        Wire.write(line[i]);
        Wire.write(funcMenu[menulevel][i]);
        Wire.endTransmission();
    }

    funcScreenUpdate=0;
}
```

Kodeliste 3.6: Oppdatering av menyskjerm

Likt som på infoskjermen blir skjermen tømt med en gang menyen endrer seg, bestemt av variabelen funcScreenUpdate. Menyen som skal vises er gitt av variabelen menulevel, så for-løkken henter menyvalgene på hver linje ved bruk av i-variabelen til det tilsvarende menynivået. For å igjen passe på at meny-skjermen ikke flimrer blir variabelen funcScreenUpdate endret slik at menyen forblir uendret.

```
while(captureCondition==1){
    if(firstLoop==1){
        firstLoop=0;
        captureDone=1;

        analogWrite(A.triggerInPWMPin, A.triggerInStatus*triggerVoltageToDuty(A.portVoltage));
        analogWrite(B.triggerInPWMPin, B.triggerInStatus*triggerVoltageToDuty(B.portVoltage));

        attachInterrupt(digitalPinToInterrupt(A.triggerInReadPin), portAcheckCapture, RISING);
        attachInterrupt(digitalPinToInterrupt(B.triggerInReadPin), portBcheckCapture, RISING);

        SerialInPortA.begin(34800);
        SerialInPortB.begin(34800);

        delay(50);

        for(int i = 0; i < 1000; i++){
            SerialPortAPermData[i]="";
            SerialPortBPermData[i]="";
        }
    }
```

Kodeliste 3.7: Første funksjoner i datainnsamling

Løkken brukt for datainnsamling følger hovedstrukturen til resten av koden, der et kodestykke kjører en gang i starten og resten repeterer til koden blir stoppet. Datainnsamlingen inkluderer også et stykke kode som kjører en gang i slutten, som blir

diskutert senere.

For å starte datainnsamlingen er variabelen captureCondition endret til en 1, som gjør argumentet til while-løkken sann. Koden i første if-setning, bestemt av verdien til variabelen firstLoop, starter med å bytte verdi til firstLoop for å forsikre at koden bare kjører en gang. Variabelen captureDone er ansvarlig for at et stykke kode kjører i slutten av datainnsamlingen, så den endres i tillegg.

Terskelspenningen er gitt av variablene portVoltage, som konverteres til en 8-bits verdi ved bruk av funksjonen triggerVoltageToDuty. Denne 8-bits verdien ganges med triggerstatusen (1 eller 0) for å unngå at en port ikke produserer et PWM signal på en inaktiv triggerport. Deretter etableres interrupt-pinner for begge trigger inn portene, selv om portene ikke brukes. Så åpnes seriellportene for kommunikasjon, som gjøres i starten av datainsamlingsprosessen slik at seriellpluggene kan kobles til og fra utenom datainnsamlingsprosessen. Til slutt tømmes arrayene med informasjon fra forrige datainnsamling, slik at tidligere data ikke blander seg med de nye testene.

```
    if(A.capture==1){
        char characterA = SerialInPortA.read();

        if (characterA != (char)255){
            SerialPortATempData.concat(characterA);
        }

        if (characterA == '\n'){
            SerialPortAPermData[SerialPortAPos]=SerialPortATempData;
            SerialPortATempData = "";
            SerialPortAPos++;
        }
    }
```

Kodeliste 3.8: Datainnsamling

Det er bare mulig å lese data fra seriell en karakter om gangen, så data hentet fra seriell blir bevart som en karakter characterA. Dersom ingen data har blitt sendt enda og seriell er tom, kommer lesing av seriell til å gi char verdien 255. Koden sjekker om den leste karakteren er 255, og om den ikke er det legges karakteren til en streng SerialPortATempData, som er en streng for midlertidig data. Da en linje er ferdigsendt leser seriell en \n karakter, som gjør at koden legger til strengen med midlertidig data til en array for permanent data, og midlertidig data tømmes for fremtidig datainnsamling. Posisjonen i det permanente minne er gitt av SerialPortAPos, som øker hver gang data legges til minne.

```
if(captureDone==1){
    captureDone=0;
    firstLoop=1;
    SerialPortAPos=0;

    SerialInPortA.end();

    detachInterrupt(digitalPinToInterrupt(A.triggerInReadPin));

    SD.begin();

    if(A.capture==1){
        A.capture=0;
        captureDataA = SD.open("/DataA.txt", FILE_WRITE);
```

```
        for(int y=0; y<1000; y++){
            if(SerialPortAPermData[y]!=""){
                captureDataA.println(SerialPortAPermData[y]);
            }
        }
        captureDataA.close();
    }
    SD.end();
}
```

Kodeliste 3.9: Sluttprosessen av datainnsamling

Kodestykket over er programmet som kjøres i slutten av datainnsamlingen for en av de to seriellportene. Etter koden endrer verdien på de kritiske variablene, stenger den linjen for seriell kommunikasjon med den eksterne enheten og slår av interruptfunksjonen for triggeren. Deretter opprettes det kommunikasjon med SD-kortet for å begynne eksportering av dataen bevart i arrayen med permanent. Til slutt skal kommunikasjon med SD-kortet stenges slik at det kan fjernes uten at testutstyret må slås av. Fullstendig kode med kommentarer er oppgitt i vedlegg 1.

3.5 Bygging av testutstyret

I fra kravene til oppgaven er det gitt at testutstyret skal være robust og skal kunne være vannsikkert. Valget ble derfor mellom å designe en egen innkapsling fra grunnen eller gå for en eksisterende løsning. Design og konstruksjon av egen innkapsling krever grundig planlegging av hvordan selve boksen bygges opp for å kunne tilfredsstille kravene som ble gitt. Derfor ble det bestemt å gå for en eksisterende løsning og heller bestille det. Etter noe undersøking på nettet etter noe som tilfredsstiller kravene, ble det bestemt å gå for en Peli Case 1400 koffert med panelramme for å kunne installere et grensesnittpanel. Peli Case ble valgt da deres kofferter er kjente for å være robuste og sikre, samt var typen 1400 av gode dimensjoner og som kunne passe alt som skulle innkapsles. Kofferten med panelrammen inkluderer to pakninger av gummi; en mellom koffert og lokk, og en for mellom kofferten og panelrammen. Med disse pakningene kan kofferten sies å være vannresistent nok til å dekke kravene fra oppgavegiver.

Det ble bestemt etter diskusjon med oppgavegiver at det skulle følge med et type grensesnitt for testutstyret. Grunnet valg av koffert ble det naturlig å lage et grensesnittpanel som lå i toppen av kofferten med all elektronikk under. Oppgavegiver ønsket at grensesnittet skulle bestå av følgende utstyr:

- Skjermer for visning av status og informasjon om testutstyret
- Knapper for navigering av skjermer og endring av verdier.
- Diverse inn- og utformater for data

grensesnittpanelet ble tegnet i Inkscape som er et bilderedigeringsprogram som har støtte til redigering av vektorgrafikk, og ved bruk av laserkutter kunne grensesnittpanelet skjæres ut presist fra tegningene. Institutt for design ved NTNU ble kontaktet for bistand med både tegninger og utskjæring. Panelet ble først skjært ut i MDF som test for å være sikre på at alle komponentene passer og at panelet passet i kofferten. Etter testing av at MDF panelet passet til rammen og komponentene, ble panelet til slutt skjært ut i akryl.

Komponentene i kofferten kan deles inn i tre større deler; batterisystem, kretskort og grensesnittpanel. Grunnet at oppgavegiver skal kunne gjøre justeringer og endringer i

fremtiden om ønskelig, ble det valgt å ikke feste komponenter permanent i kofferten. Siden kofferten skal være vanntett, var det ikke gunstig å lage hull i bunnen for å feste delene med skruer. Derfor ble det foreslått å bruke borrelås som feste. Borrelåsen sin festeevne til komponentene ble testet, og resultatene viste at borrelåsen kunne holde komponentene til ønsket grad. Kretskortet og batterisystemet ble festet med borrelås, mens grensesnittpanelet ble festet med skruer til panelrammen. For å forsikre at testutstyret var mest mulig vannsikkert, ble det brukt silikonfuge rundt komponentene i panelet som ikke hadde inkludert egen pakning.

Batterisystemet ble planlagt skulle være sin egen modul som kunne kobles fra alle andre komponenter slik at det eventuelt kan erstattes om ønskelig. Patteripakken består av 4 li-ion celler koblet i parallell som resulterte i en kapasitet på anslått 10 000 mAh. Ettersom ingen av komponentene skal trekke særlig strøm under maksimal bruk, gir dette batteriet god levetid til testutstyret. Batteriet ble påloddet til ladekretsen TP4056 til inngangene B+ og B- for henholdsvis batteriets positive og negative ledninger. Deretter ble utgangene til ladekretsen tilkoblet den variable boost-konverteren som ble justert til å gi ut 10V ved den laveste batterispenningen. Imellom ladekretsen og konverteren ble det også installert av/på bryteren til testutstyret. Det ble valgt å plassere den mellom ladekretsen og konverteren for at det skal være mulig å lade batteriet mens systemet er slått av, og i tillegg at konverteren ikke skal kunne lade ut batteriet selv om systemet er avslått. Konverterens positive og negative utganger ble koblet til kretskortet med en 3-pin-kontakt i lag med en ledning tilkoblet batteriet for måling av batterispenningen.

For tilkobling av skjermene til kretskortet ble det brukt kontakter slik at installering til panelet ble enklere. Inn- og utgangene samt meny-knappene ble derimot påloddet direkte til kretskortet med ledninger grunnet at kontakter ikke ble tatt hensyn til under tegning av utlegget til kretskortet. Seriell portene bruker følgende koblinger i tabell 3.2 og 3.3 for henholdsvis RS-232 og RS-422:

Navn	**Pinne på kontakt**	**Farge på ledning**
TX	2	Oransje
RX	3	Gul
GND	5	Sort

Tabell 3.2: RS-232 DB9 koblingsskjerma

Navn	**Pinne på kontakt**	**Farge på ledning**
TX+	7	Oransje
TX-	2	Grønn
RX+	3	Gul
RX-	8	Brun
GND	6	Sort
GND	4	Sort

Tabell 3.3: RS-422 DB9 koblingsskjerma

Kapittel 4

Resultater

4.1 Funksjonalitetstesting

På grunn av begrensninger av diverse utstyr som var tilgjengelig, var det ikke mulig å implementere kode for all maskinvaren i utstyret, som diskutert i kodeforklaringen. Listen av funksjoner som kunne både programmeres og testes var sentrert rundt mottakelse av data som hentes da et justerbart triggersignal mottas, i tillegg til eksportering av data i forskjellige medier.

Menneskelig grensesnitt

Det menneskelige grensesnittet fungerte nøyaktig som håpet, der skjermene aldri sviktet under standarddrift. Skjermene mislyktes bare i spesielle tilfeller der strømmen til utstyret raskt ble slått av og på, som forhindret mikrokontrollerne til skjermene fra å skru på bakgrunnsbelysningen.

Knappene fungerte til ønsket grad og viste ingen tegn til feil i de diverse testene som ble utført. Testene inkluderte rask trykking av alle knappene tilfeldig, trykking av flere knapper samtidig, bruk av knapper som ikke hadde menyfunksjoner i den nåværende menyen, i tillegg til de andre testene som diskuteres senere.

Trigger inn

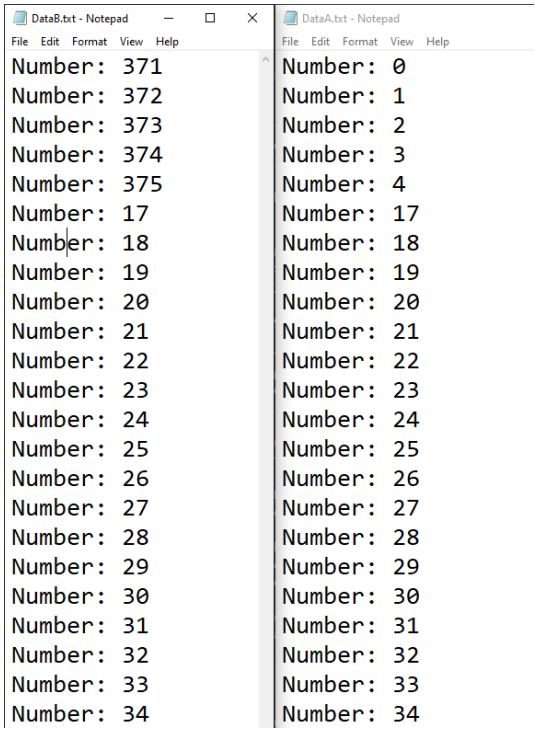
Funksjonene tilknyttet til triggerportene fungerte som ønsket, og kunne pålitelig utløses av triggersignaler kortere enn $1ms$, som var det korteste triggersignalet som kunne lages av testutstyret. Det var også mulig for begge triggerportene å aktiveres uavhengig av hverandre, som gjør det mulig å utføre to tester samtidig.

Terskelspenningen var ikke like nøyaktig som forventet, der et triggersignal kunne aktivere en triggerport med en terskelspenning 1-2V høyere. Denne forskjellen gjorde det mulig at et 3.3V triggersignal kunne pålitelig utløse en triggerport med en 4.1V terskelspenning.

Seriell inn

Fra testene utført, var det aldri oppdaget manglende data, eller data som var endret i transport og behandling. Det var også ingen tap i funksjonalitet der begge seriell portene fikk data samtidig i samme test. De forskjellige seriell-linjene var testet ved å bruke en annen

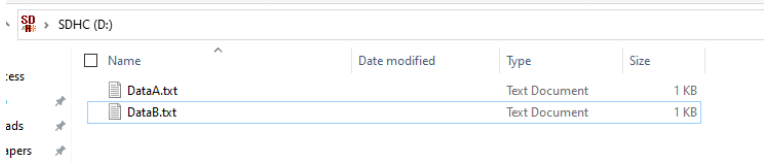
mikrokontroller for å sende tall fra 1-500, der trigger ble aktivert etter tall nr. 16. Dette betydde at data med tall fra 17 og oppover skulle bli lagret. En uventet feil derimot, oppstod der mottatt data inkluderte noen få linjer med data fra før trigger ble aktivert, som var inkludert i dataen som var eksportert. Denne feilen kommer fra løsningen brukt for å lese data etter trigger var aktivert, som involverte å tømme seriellbufferen da triggeren var utløst. Funksjonen for å tømme seriellbufferen tar 100ms for å utføres, så kretsen leser fortsatt data samtidig som dataen blir fjernet. Et eksempel av dette kan ses i resultatene fra testene i figur 4.1.



Figur 4.1: Data fra utført test

SD-kort og eksportering

Eksportering til SD-kort fungerte også som ønsket. Likt som seriellportene var koden designet slik at SD-kortet trengte bare å være plugget inn under testen, og kunne fjernes uten at det oppsto noen problemer med lagret data. I den nåværende koden er all dataen fra port A lagret på en tekstfil DataA.txt, som gjør at dataen fra flere tester blir blandet sammen. Fremtidig arbeid inkluderer da separering av data fra forskjellige tester, ikke bare separering mellom porter. Figur 4.2 viser de to filene som var laget i samme uavbrutte test fra begge portene.



Figur 4.2: Filene laget av testutstyret

4.2 Mekanisk testing

Figur 4.3: Ferdig konstruert testutstyr

Figur 4.3 viser resultatet av konstrueringen av testutstyret i Peli Case kofferten. Grunnet kravene gitt av oppgavegiver at testutstyret skulle være robust og vannsikker, ble kofferten testet for hvor vannsikkert panelet er og hvorvidt komponentene løsner under vanlig bruk.

Testing av vannsikkerhet

Vanntetningen til grensesnittpanelet ble testet med å sprute lett vann over panelet for å simulere lett regn. Under testingen var batteriet koblet fra og fjernet fra kofferten. Selve testen gikk ut på for å se om vann kunne trenge seg gjennom panelet og under til komponentene nederst i kofferten når lokket er åpent under bruk. For at testen skulle gi mest mulig realistiske resultater, sto kofferten i 4 timer med vann påført på panelet. For å kunne se om vann har gjennomtrengt panelet ble det benyttet tørkepapir som skulle absorbere vannet og vise tydelig på papiret om vesentlige vannmengder har penetrert vanntetningen.

Etter gjennomført test ble det nøye gjennomgått om noe vann har gått gjennom panelet og ned til kretskortet. Etter det som ble observert, viste det seg at ingen tydelig mengde vann hadde gått gjennom tetningen. Det viste seg også at ingen komponenter etter testing hadde tegn på at vann hadde gjennomtrengt. Tørkepapiret berørte alle mulige steder hvor vannet kan ha trengt gjennom, men viste ingen tegn på fukt.

Vibrasjonstest

Det ble valgt at vibrasjonstesten skulle være overdrevet til en viss grad, og skulle ikke simulere hvordan kofferten faktisk skal bli håndtert. Dermed var testen av typen verste fall scenario. Formålet med testen var for å forsikre at alle komponentene, spesielt batterisystemet og kretskortet, som er festet med borrelås, sitter sikkert og ikke løsner.

Vibrasjonstesten ble gjennomført med å riste kofferten med forskjellige styrker. Ved hard risting løsnet ingen av komponentene inne i kofferten og det eneste hørbare er ledningene til testutstyret.

Kapittel 5

Diskusjon

5.1 Analyse av resultater

Som nevnt i kodeforklaringen var funksjonalitet begrenset til det som kunne testes og verifiseres. Begrensningene betydde at seriell ut, trigger ut, og CAN ikke kunne testes, så de ble heller gitt som videre arbeid til oppdragsgiver. Til slutt var en liste laget som understreket funksjonaliteten som kunne legges til:

- Motta data via seriell inn.
- Justere terskelspenning for trigger.
- Datainnsamling utløst av trigger.
- Motta og behandle data fra to uavhengige kilder samtidig .
- Vise informasjon angående testutstyret på en skjerm.
- Navigere menyer på en skjerm ved bruk av knapper.
- Eksportere data til et SD-kort.
- Måle og vise batterinivå som en prosentandel.
- Eksportere data til en datamaskin.

Denne seksjonen går gjennom de forskjellige elementene og diskuterer resultatene i forhold til hva som var forventet.

Motta data via seriell inn.
Mottakelsen av data via seriell endte opp med å være en av de siste funksjonene som var implementert, selv om det var en av de viktigere elementene av produktet. Dette kom for det meste fra at etter kretsen var loddet var det vanskelig å teste seriellportene uten annen programinfrastruktur. Programinfrastrukturen som gjorde det mulig å navigere funksjonene seriell krevde, som trigger eller en prosess for henting av data, gjorde testing vanskelig. Selv om det var implementert såpass sent i prosessen virket det som planlagt med ingen tap, med ulempen av at litt data før trigger var aktivert ble inkludert.

Justere terskelspenning for trigger.
Justering av terskelspenning var funksjonen som ikke gikk like bra som forventet. Trigger kunne pålitelig bli utløst av spenninger 1-2V lavere enn terskelspenningen, som påvirket den generelle presisjonen til utstyret. Triggersignalet var implementert for det meste for å unngå at støy eller interferens startet datainnsamlingen, som det effektivt klarer. Tapet av presisjon i terskelspenningen, har derimot påvirket utstyrets evne til å motta mer komplekse triggersignaler som ikke er en digital høy eller lav.

Datainnsamling utløst av trigger.
Utstyret var laget for å ha en rask responstid til et triggersignal, som var et krav for slike tester utstyret måtte gjennomføre. Dette var grunnen til at det var brukt interrupts under datainnsamlingen, som reduserte responstiden markant fra en vanlig digitalRead funksjon. En uventet effekt av å bruke interruptfunksjoner var den korte varigheten på triggersignalene. For å teste testutstyret var en sekundær testkrets laget som kunne sende både trigger- og serielldata. Det ble oppdaget at den sekundære testkretsen ikke kunne lage korte nok triggersignaler som ikke kunne leses av testutstyret. Siden klokkehastigheten til mikrokontrolleren kan justeres mellom 80MHz og 240MHz, kan interruptfunksjonen ha en kortest mulig responstid rundt $2\mu s - 6\mu s$ dersom behandling av interrupts tar 500 klokkesyklusser. Denne korte responstiden virker veldig lovende for ment bruksområde til testutstyret.

Motta og behandle data fra to uavhengige kilder samtidig.
Ved bruk av det sekundære testutstyret fikk testutstyret tilsendt både serielldata og triggersignaler samtidig i samme uavbrutte test. Resultatet fra denne testen viste at det ikke var noen tap i den generelle ytelsen til testutstyret sammenliknet med en test som bare brukte en trigger- og seriellport. Dette resultatet viste også at det er mulig for et fremtidig testsystem å ha mange flere porter med lite eller ingen effekt i ytelsen.

Vise informasjon angående testutstyret på en skjerm og navigere menyer på en skjerm ved bruk av knapper.
Skjermene og knappene var de første funksjonene som var programmert. Dette var gjort for å kunne implementere og teste nye funksjoner mere effektivt, som justering av triggerporter. Dette endte opp med å være en effektiv løsning, siden alle funksjonene etter skjermen tok mye kortere tid å programmere. Det var også viktig å lage menyfunksjonalitet så effektivt som mulig, så det ikke var nødvendig å endre eller fikse den hver gang en ny funksjon skulle legges til. Som sagt i resultatseksjonen var både knappene og skjermene veldig effektive, og feilet aldri under normal operasjon, og bare noen få ganger under veldig unormale stresstester.

Eksportere data til et SD-kort.
Eksportering av data til SD-kort endte opp med å være en annen vanskelig funksjon å legge til, siden det var vanskelig å forstå hvor et problem oppstod. Det var også noen ukjente problemer som oppstod rundt behandling av noen Unicode-karakterer, men som også ble løst. I de siste testene var det ikke noen problemer med eksportering av data, selv med data fra begge portene.

Måle og vise batterinivå som en prosentandel.
Batterinivået var en annen funksjon som ikke gikk så bra som håpet, der en fortsatt ukjent feil i kretskortet gjør at batterinivået ligger alltid på 100%. Koden som regner ut en prosent basert på batterispenningen er testet og virker som den skal, men kretsfeilen gjør at mikrokontrolleren prøver å lese en spenning høyere enn den høyeste spenningen den kan lese.

Eksportere data til en datamaskin.
Eksportering av data til en datamaskin via seriell var en funksjon som kom naturlig samtidig

som lesing av seriell data. Dette er fordi koden fikk mikrokontrolleren til å sende data via seriell for testing av forskjellige funksjoner som mottakelse av seriell data. Eksportering var ikke satt inn som en funksjon i menyen, men skal alltid eksportere til seriell i slutten av en test.

Mekanisk test

Testingen av den mekaniske delen av prosjektet ga resultater om løsningen til oppgaven tilfredsstiller kravene gitt av oppgavegiver. Krav til innkapslingen var at det skulle være robust og vannresistent siden utstyret skal kunne brukes ute på testområder. Fra testingen som ble utført på innkapslingen kan det konkluderes at løsningen oppfyller kravene.

Fra vannmotstands testen ble det testet hvorvidt elektronikkpanelet med lokket åpent kunne motstå vann fra simulert regn. Etter det ble sprayet på lett vann over panelet, sto det i en periode for å se om vannet kom seg igjennom tetningen. Testen ble konkludert med at vanntetningen fungerte bra og tilsynelatende kom det ikke noe vann gjennom. Derimot bør det også nevnes at utført test ikke er perfekt og har sine mangler. Det ble kun påført et lett lag med vann en gang i starten av testperioden, dermed er det vanskelig å si hvorvidt panelets evne til å motstå tungt regn er. Men grunnet testen gikk ut over i en periode på 4 timer og tilsynelatende null vann gikk gjennom, er resultatene gode nok til å konkludere at kravet om vannsikkerhet er oppfylt. Det ble ikke utført noen ytterlige tester med større mengder vann grunnet under testing er testutstyret utilgjengelig for annet arbeid. Under bruk i et verste fall scenario med tungt regn anbefales det at lokket er halvveis lukket slik at minst mulig regn faller på panelet og man kan fortsatt operere testutstyret.

Vibrasjonstesten av testutstyret ble utført med å riste kofferten med alle komponentene installert. Grunnlaget for denne testen var siden batterisystemet og kretskortet var festet med borrelås for at komponentene skulle være mulig å lett justere eller erstattes senere av oppgavegiver. Som nevnt under kapittelet om resultater, ble det bestemt at vibrasjonstesten skulle være overdrevet slik at det er sikkert at komponentene ikke løsner under vanlig bruk. Resultatene fra testen viste at ingen av komponentene løsnet eller raslet inne i kofferten. Fra dette kan løsningen kalles robust og kravene til oppgavegiver oppfylt.

5.2 Etisk grunnlag

Grunnet oppgaven er gitt av utviklingsavdelingen for elektronikk i ammunisjon ved Nammo AS, omhandler prosjektet grunnleggende om våpen- utvikling og produksjon. Funksjonen til testutstyret er at det skal ta imot telemetridata sendt ut med utskutt ammunisjon for å behandle og eksportere dataen. Formålet med testutstyret er da ikke direkte tilkoblet selve produksjonen av våpen, men heller testing. Spørsmålet om våpen hjelper verden mer enn den skader er veldig komplisert og kan knyttes til FNs bærekraftdelmål 16.1: ”Betydelig redusere alle former for vold og dødelighet knyttet til vold i hele verden"[32]. Om våpen er et nødvendig verktøy som trengs for å oppnå dette er vanskelig å bekrefte, men er en viktig baktanke å ha som produsent og utvikler av våpenutstyr.

Bærekraftsmål 12[33] går ut på ansvarlig forbruk og produksjon. Da dette prosjektet går ut på å produsere et testutstyr som skal brukes til å gjennomføre testing av utviklingsprodukt, kan det bistå med avdekking av feil som kan oppstå senere i produktene. Dersom feil ikke blir

oppdaget før produktene blir produsert i større skalaer, kan det føre til at produktene må bli tilbakekalt som kan kreve enorme ressurser. Siden testutstyret ikke er ment til å produseres i noe større skala, ble det ikke sett på som nødvendig å gjennomføre noe form for livsløpsanalyse av komponentene brukt i prosjektet.

5.3 Forbedringer

Terskelspenning for trigger
Ytterligere testing av den justerbare trigger terskelspenningen kunne ha avslørt problemet utstyret endte opp med, og gjøre det mulig å fikse det tidligere. At terskelspenningen ikke er nøyaktig like høy som forventet kan være et resultat av en ukjent feil med enten kretsen eller mikrokontrolleren. Siden terskelspenningen til kretsen ikke var ekstremt viktig, og kretsen fortsatt ignorerer støy og interferens er triggerportene fortsatt vellykket.

Et annet uventet problem var at kretsen til trigger ut portene krevde pull-down motstandere for å gjøre triggersignalet mer stabilt og pålitelig. GPIO-pinne 0 som var tildelt en av triggerportene har en funksjon som gjør at programmet ikke blir lastet opp riktig om den er dratt ned til en spenning lik 0V. Dette var løst ved å lodde vekk motstanderen som koblet pinnen til jord, men dette gjorde sannsynligvis trigger pinnen mindre pålitelig.

Lading og batteri
Et krav for testutstyret var at det må kunne bruke strøm fra egne batterier, så det var bestemt å bruke batterier som kunne vare lenge nok til at testutstyret slo seg av under en test. For å forsikre at batteriene ikke gikk tomme var et 10Ah batteri brukt, som kunne forsyne testutstyret 80 timer uten stopp. Et uventet problem oppstod der ladekretsen kunne lade med en strøm rundt 0.5A, som gjorde at ladetiden varer rundt 20 timer fra utladet til full. En mulig løsning til dette kan være å erstatte motstanden på ladekretsen som bestemmer ladestrøm. Dette kan også omgås ved å lade testutstyret ofte når det ikke er i bruk. Det var også veldig uventet at kretsen for å måle batterinivået ikke fungerte som det skulle.

Innkapsling
Fra resultatene til testingen av innkapslingen, ble det bekreftet at kravene fra oppgavegiver om at testutstyret skal være robust og vannsikkert er oppfylt. Derimot kan noen deler forbedres ytterligere. Grunnet vannmotstands testen kun ble testet med en lett mengde vann, er det ikke sikkert hvor vannsikkert panelet er mot større mengder. Dermed er en mulig forbedring av løsningen være å ha testet panelet med større mengder vann og gjort endringer i henhold til resultatene deretter. I tillegg hadde det vært en fordel om komponentene i kofferten hadde blitt festet med en bedre løsning enn borrelås. Selv om vibrasjonstesten viste gode resultater, kommer borrelås eventuelt til å slites ut og festeevnen degraderes.

Grensesnittutlegg
Nåværende grensesnittpanel kan virke noe forvirrende med at det ikke er tydelig at øverste seriell-inn og øverste trigger-inn er sammenkoblet, og det samme med nederste seriell-inn og trigger-inn. Derfor kan panelet forbedres med enten å flytte seriell-inn og trigger-inn parvis sammen eller merke panelet med Seriell-inn 1 og Trigger-inn 1, samme med Seriell-inn 2 og Trigger-inn 2.

Bruk av kontakter på kretskort
Grunnet at kontakter på kretskortet ikke ble konsiderert før utlegget var ferdig og bestilt, passet ikke de kontaktene som var tilgjengelige til de fleste portene på kretskortet. Der noen porter hadde like dimensjoner som pinnene til kontaktene, ble det benyttet kontakter. Det inkluderte batterisystem- og skjermtilkobling. Resten av portene ble påloddet direkte til ledningene fra komponentene til grensesnittet. Dette resulterte til en utfordring med monteringen av kretskortet og panelet. En stor forbedring er derfor være å bruke kontakter til alle portene på kretskortet slik at det kan skilles fra panelet og installeres individuelt i kofferten.

5.4 Videre arbeid

Selv om testutstyret på nåværende tidspunkt er i en funksjonell tilstand, og kan brukes effektivt i tester som er kompatible med inngangsprotokollene, er det fortsatt noen funksjoner som kan legges til for å forbedre testutstyrets funksjonalitet.

- Tilkobling av 8-pin kontakt
- CAN
- Testing av seriell ut
- Testing av trigger ut
- Bytting av mikrokontroller / multipleksing
- Langtidsminne

Kapittel 6

Konklusjon

Hensikten med denne rapporten var å beskrive bakgrunnen og metodikken for konstruksjon av robust testutstyr for utviklingsavdelingen for elektronikk i ammunisjon ved Nammo AS, som gitt med følgende problemstilling:

"*Utredning av robust testutstyr system som skal kunne behandle telemetridata fra programmerbar ammunisjon*"

Denne problemstillingen kom fra Nammo AS sitt ønske om å bytte ut sitt eldre testutstyr, som ikke fungerte like effektivt som de ønsker, med manglende funksjonalitet. Etter et møte med kontaktpersonen fra Nammo AS var det bestemt hva slags input og output protokoller som måtte inkluderes i testutstyret. Kravene bestemt i møtet var ikke identiske til kravene som var gitt originalt i oppgaveteksten, som inkluderte litt mindre spesifikk informasjon om de nødvendige protokollene og hvordan de skulle brukes.

Testsystemet er bygd rundt en sentral mikrokontroller som tar imot og behandler data gjennom grensesnittene i kretsen. Ut ifra kravene gitt av oppgavegiver, ble testutstyret konstruert i en robust koffert med batteriløsning. Funksjonene som kunne programmeres var begrenset til de funksjonene som kunne testes med utstyret som var tilgjengelig, så implementering for resten av funksjonalitet blir igjen som videre arbeid for oppdragsgiver.

Ved å bruke en sekundær mikrokontroller ble alle ferdig implementerte funksjoner til testutstyret testet og verifisert til å virke på et ønsket nivå. Robustheten til kofferten ble testet både for vannsikkerhet og løse komponenter ved bruk av en vibrasjonstest.

Resultatene til testene viste at testutstyret presterte bedre enn det som var forventet. Responstiden til triggersignalene er teoretisk sett tusen ganger raskere enn de korteste signalene som det sekundære testutstyret kunne produsere. Testene viste også at det var ingen tap av informasjon under transmisjon og behandling av seriell data, selv når to kilder sendte data samtidig. Resultatene fra testene kunne lett eksporteres til en datamaskin via seriellkommunikasjon, eller ved bruk av SD-kort. Testutstyret kunne også enkelt kontrolleres ved bruk av LCD-skjermer og et knappbasert grensesnitt. Testutstyret var også testet for robusthet, som viste at det var vannsikkert og vibrasjonssikkert under normal bruk.

Som følge av resultatene oppnådd blir prosjektet sett på som en suksess. Selv om ikke alle funksjonene ønsket av oppgavegiver var implementert er det fortsatt god mulighet for videre

utvikling av nåværende krets. Oppgaven har gitt veldig gode muligheter å bruke og videre utvikle ferdigheter lært i det tre-årlige løpet på NTNU.

Bibliografi

[1] «BU-205: Types of Lithium-ion,» Battery University. (18. sep. 2010), adresse: https://batteryuniversity.com/article/bu-205-types-of-lithium-ion (sjekket 24.04.2022).

[2] C. Woodford. «How Do Lithium-Ion Batteries Work?» Explain that Stuff. (27. jul. 2009), adresse: http://www.explainthatstuff.com/how-lithium-ion-batteries-work.html (sjekket 24.04.2022).

[3] «Basics: Project 082a Lithum Battery Charger TP4056 at Acoptex.Com / ACOPTEX.COM.» (), adresse: https://acoptex.com/project/9446/basics-project-082a-lithum-battery-charger-tp4056-at-acoptexcom/ (sjekket 24.04.2022).

[4] K. Fronczak, «Stability Analysis of Switched DC-DC Boost Converters for Integrated Circuits,» 1. aug. 2013. DOI: 10.13140/RG.2.1.2646.7606.

[5] «TP4056.Pdf.» (), adresse: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Prototyping/TP4056.pdf (sjekket 24.04.2022).

[6] «FS8205A.Pdf.» (), adresse: http://j5d2v7d7.stackpathcdn.com/wp-content/uploads/2021/03/FS8205A.pdf (sjekket 18.05.2022).

[7] «DW01A.Pdf.» (), adresse: http://hmsemi.com/downfile/DW01A.PDF (sjekket 24.04.2022).

[8] «Serial vs. Parallel.» (), adresse: https://focuslcds.com/serial-vs-parallel/ (sjekket 20.04.2022).

[9] J. Sonnenberg, «Serial Communications RS232, RS485, RS422,» s. 6, 2018.

[10] «The Difference Between Parallel & Serial Communication | Techwalla.» (), adresse: https://www.techwalla.com/articles/the-difference-between-parallel-serial-communication (sjekket 20.04.2022).

[11] «UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter | Analog Devices.» (), adresse: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html (sjekket 20.04.2022).

[12] «What is full-duplex?» SearchNetworking. (), adresse: https://www.techtarget.com/searchnetworking/definition/full-duplex (sjekket 20.04.2022).

[13] «Fundamentals of RS-232 Serial Communications.» (), adresse: https://www.maximintegrated.com/en/design/technical-documents/tutorials/8/83.html (sjekket 20.04.2022).

[14] «Serial Peripheral Interface (SPI) - Learn.Sparkfun.Com.» (), adresse: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all (sjekket 17.05.2022).

[15] «Introduction to SPI Interface | Analog Devices.» (), adresse: https://www.analog.com/en/analog-dialogue/articles/introduction-to-spi-interface.html (sjekket 17.05.2022).

[16] «I2C Primer: What Is I2C? (Part 1) | Analog Devices.» (), adresse: https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html (sjekket 18.05.2022).

[17] «I2C - Learn.Sparkfun.Com.» (), adresse: https://learn.sparkfun.com/tutorials/i2c/all (sjekket 18.05.2022).

[18] «Basics of the I2C Communication Protocol.» (), adresse: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/ (sjekket 18.05.2022).

[19] «ESP32 Pinout Reference: Which GPIO pins should you use? | Random Nerd Tutorials.» (2. okt. 2019), adresse: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/ (sjekket 01.04.2022).

[20] «ESP32-DevKitC V4 Getting Started Guide - ESP32 - — ESP-IDF Programming Guide Latest Documentation.» (), adresse: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html (sjekket 18.05.2022).

[21] «Triggering | Articles | TiePie - USB Oscilloscopes, Spectrum Analyzers, Data Loggers, Multimeters, Arbitrary Waveform Generators.» (), adresse: https://www.tiepie.com/en/articles/triggering (sjekket 21.04.2022).

[22] «En.CD00000444.Pdf.» (), adresse: https://www.st.com/content/ccc/resource/technical/document/datasheet/41/4f/b3/b0/12/d4/47/88/CD00000444.pdf/files/CD00000444.pdf/jcr:content/translations/en.CD00000444.pdf (sjekket 18.05.2022).

[23] «Linear_reg_basic_appli-e.Pdf.» (), adresse: https://www.mouser.com/pdfdocs/linear_reg_basic_appli-e.pdf (sjekket 25.04.2022).

[24] B. S. Lee, «Technical Review Of Low Dropout Voltage Regulator Operation And Performance,» s. 30,

[25] «Logic Level Shifting: What Does a Level Shifter Do? | Arrow.com,» Arrow.com. (), adresse: https://www.arrow.com/en/research-and-events/articles/using-a-level-shifter-to-integrate-different-voltage-signals (sjekket 30.04.2022).

[26] «NHD_0420D3Z_NSW_BBW_V3-11396.Pdf.» (), adresse: https://no.mouser.com/datasheet/2/291/NHD_0420D3Z_NSW_BBW_V3-11396.pdf (sjekket 18.05.2022).

[27] «Rs422.» (), adresse: https://assets.maxlinear.com/web/documents/sp3070e-sp3078e.pdf (sjekket 18.05.2022).

[28] «Max232.Pdf.» (), adresse: https://www.ti.com/lit/ds/symlink/max232.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1652854834477&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fmax232 (sjekket 18.05.2022).

[29] «Op-Amps, Comparator Circuit | Renesas.» (), adresse: https://www.renesas.com/us/en/support/engineer-school/electronic-circuits-03-op-amps-comparator-circuit (sjekket 02.03.2022).

[30] «Lp2902-n.Pdf.» (), adresse: https://www.ti.com/lit/ds/symlink/lp2902-n.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1652889214073&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Flp2902-n (sjekket 18.05.2022).

[31] «CAN Datasheet.» (), adresse: https://www.ti.com/lit/ds/symlink/sn65hvd265.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1652814474344&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fsn65hvd265 (sjekket 18.05.2022).

[32] «Fred, rettferdighet og velfungerende institusjoner.» (), adresse: https://www.fn.no/om-fn/fns-baerekraftsmaal/fred-rettferdighet-og-velfungerende-institusjoner (sjekket 20.05.2022).

[33] «Ansvarlig forbruk og produksjon.» (), adresse: https://www.fn.no/om-fn/fns-baerekraftsmaal/ansvarlig-forbruk-og-produksjon (sjekket 20.05.2022).

Vedlegg

Vedlegg 1: HovedKode

```
    #include <Wire.h>
#include <SoftwareSerial.h>
#include <SPI.h>
#include <SD.h>
#include <FS.h>



//----------------Ærklærer funksjoner----------------------
float setTriggerVoltageDownA(float voltage);
float setTriggerVoltageUpA(float voltage);
float setTriggerVoltageDownB(float voltage);
float setTriggerVoltageUpB(float voltage);
float returnA(float a);
float byttTriggerUt(float port);
float byttTriggerInn(float port);
float beginCapture(float addr);
int triggerVoltageToDuty(float TriggerVoltage);


//---------------Informasjonsskjermen-----------------
char line1[] = "Batteri:            ";
char line2[] = "T.A:                ";
char line3[] = "T.B:                ";


//------------------Data og variabler for triggerporter--------------
struct {
  float portVoltage=5;
  int triggerInReadPin;
  int triggerInPWMPin;
  int triggerOutPin;
  int triggerInStatus = 0;
  int triggerOutStatus = 0;
  int capture=0;
} A,B;

//---------------pins og data for inpuknapper----------------
struct {
  int buttonPin;
  int lastButtonVal=0;
  int buttonVal;
}but1, but2, but3, but4;

//---------------pins og funksjoner for seriell kommunikasjon----------------
struct serialLines{
  int RxPin;
  int TxPin;
};
```

```
serialLines SerialInPortAPins {33, 255}; //trenger ikke en TX kobling, så TX pinnen settes til pin 255, en pin som i
serialLines SerialInPortBPins {26, 255};

SoftwareSerial SerialInPortA(SerialInPortAPins.RxPin, SerialInPortAPins.TxPin);
SoftwareSerial SerialInPortB(SerialInPortBPins.RxPin, SerialInPortBPins.TxPin);

String SerialPortATempData="";
String SerialPortBTempData="";
String SerialPortAPermData[1000] = {""};
String SerialPortBPermData[1000] = {""};
int SerialPortAPos=0, SerialPortBPos=0;


//-----------------SD card variables-------------
File captureDataA;
File captureDataB;


//---------------ekstra variabler for andre funksjoner
int infoScreenUpdate = 1, funcScreenUpdate = 1;

int captureCondition=0, firstLoop=1, captureDone=0;

int batteryPin = 34;


//-------------------------LCD meny-------------------
const byte LCD_func = 0x18, LCD_info=0x28; //I2C addressene
const byte line[4]={0x00, 0x40, 0x14, 0x54};//Hex-addressen til de forskjellige linjene på skjermene

const char *funcMenu[11][4] = {{"Ny Test",                "Eksporter Data",
"Triggerinstillinger",    "Datainstillinger"},  //0
                              {"Stopp og lagre Test",     "-",
"-",                      "-"}, //1
                              {"Eksporter til Seriell",   "-",
"-",                      "Tilbake"}, //2
                              {"Trigger inn Status",
"Juster Trig.Spenning", "Trigger ut Status",      "Tilbake"}, //3
                              {"Bytt Port A Status",      "Bytt Port B Status",
"-",                      "Tilbake"}, //4
                              {"Juster Port A",           "Juster Port B",
"-",                      "Tilbake"}, //5
                              {"Opp",                     "Ned",
"-",                      "Tilbake"}, //6
                              {"Opp",                     "Ned",
"-",                      "Tilbake"}, //7
                              {"Bytt Port A Status",      "Bytt Port B Status",
"-",                      "Tilbake"}, //8
                              {"SD kort",                 "-",
"-",                      "Tilbake"}, //9
                              {"Slett Kort",              "-",
"-",                      "Tilbake"}, //10
                              };


float menuAddr[11][4] = {{1, 2, 3, 9},
                         {0, 1, 1, 1},
                         {0, 0, 2, 0},
                         {4, 5, 8, 0},
                         {0, 1, 4, 3},
                         {6, 7, 5, 3},
                         {A.portVoltage, A.portVoltage, 6, 5},
                         {B.portVoltage, B.portVoltage, 7, 5},
                         {0, 1, 8, 3},
```

```
                    {10, 9, 9, 0},
                    {0, 0, 10, 9},
                    };

//------------lager type for array med funksjonspointers-------------------
using namespace std;
typedef float (*functionPointer) (float a);

functionPointer menuFunctions[][4] = {{beginCapture, returnA, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      {byttTriggerInn, byttTriggerInn, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      {setTriggerVoltageUpA, setTriggerVoltageDownA, returnA, returnA},
                                      {setTriggerVoltageUpB, setTriggerVoltageDownB, returnA, returnA},
                                      {byttTriggerUt, byttTriggerUt, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      {returnA, returnA, returnA, returnA},
                                      };

int menulevel = 0;



void setup(){

  //--------standard setup kode----------
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(50000); //Skjermne kan bare kommunisere under 50kHz



  Serial.println("Started");

  //------------Setter opp knapper og trigger pins----------------
  but1.buttonPin=32; //
  but2.buttonPin=25; //
  but3.buttonPin=27; //
  but4.buttonPin=17; //

  pinMode(but1.buttonPin, INPUT);
  pinMode(but2.buttonPin, INPUT);
  pinMode(but3.buttonPin, INPUT);
  pinMode(but4.buttonPin, INPUT);

  A.triggerInPWMPin = 12;
  B.triggerInPWMPin = 14;
  A.triggerInReadPin = 13;
  B.triggerInReadPin = 35;

  pinMode(A.triggerInPWMPin, OUTPUT);
  pinMode(B.triggerInPWMPin, OUTPUT);
  pinMode(A.triggerInReadPin, INPUT);
  pinMode(B.triggerInReadPin, INPUT);
}

void loop(){

  // ----------Leser knapper-----------
  but1.buttonVal=digitalRead(but1.buttonPin);
  but2.buttonVal=digitalRead(but2.buttonPin);
  but3.buttonVal=digitalRead(but3.buttonPin);
  but4.buttonVal=digitalRead(but4.buttonPin);
```

```
  if(but1.buttonVal!=but1.lastButtonVal){ //om knappen har endret status (høy-lav, lav-høy)
    if(but1.buttonVal == 0){ // om knappen nå er lav, endret status må være høy-lav, eller når knappen slippes
      menulevel=menuFunctions[menulevel][0](menuAddr[menulevel][0]); // gå til menyen, og gjør funksjonen knyttet ti
      funcScreenUpdate = 1; //skjermen som viser menyer skal nå oppdateres
    }
    delay(10); //trenger en liten delay for å
unngå at knappen trykkes mere enn en gang, kalt bouncing
  }
  but1.lastButtonVal = but1.buttonVal;

  if(but2.buttonVal!=but2.lastButtonVal){
    if(but2.buttonVal == 0){
      menulevel=menuFunctions[menulevel][1](menuAddr[menulevel][1]);
      funcScreenUpdate = 1;
    }
    delay(10);
  }
  but2.lastButtonVal = but2.buttonVal;

  if(but3.buttonVal!=but3.lastButtonVal){
    if(but3.buttonVal == 0){
      menulevel=menuFunctions[menulevel][2](menuAddr[menulevel][2]);
      funcScreenUpdate = 1;
    }
    delay(10);
  }
  but3.lastButtonVal = but3.buttonVal;

  if(but4.buttonVal!=but4.lastButtonVal){
    if(but4.buttonVal == 0){
      menulevel=menuFunctions[menulevel][3](menuAddr[menulevel][3]);
      funcScreenUpdate = 1;
    }
    delay(10);
  }
  but4.lastButtonVal = but4.buttonVal;

  //----------Oppdater infoskjermen-------------
 if(infoScreenUpdate==1){

    Wire.beginTransmission(LCD_info); //starter i2c kommunikasjon med infoskjermen
    Wire.write(0xFE);//tømmer skjermen
    Wire.write(0x51);
    Wire.endTransmission();
    delay(2); // må vente siden tømming av skjermen tar 1.5ms

    Serial.println(analogRead(34));
    Serial.println(batteryPin);

    int batteryVoltage = map(analogRead(batteryPin), 0, 4095, 0, 420); //kalkulerer batterinivået i forhold til inng
    int batteryLevel = map(batteryVoltage, 310, 420, 0, 100);
    batteryLevel=constrain(batteryLevel, 0, 100);

    int batHun = batteryLevel/100; //konverterer prosent batterinivå til forskjellige siffer
    int batTen = (batteryLevel-(batHun*100))/10;
    int batOne = (batteryLevel-(batTen*10)-(batHun*100));



    //konverterer fra int til char, som kan skrives på skjermene
    char batBufHun[2], batBufTen[2], batBufOne[2];
    String(batHun).toCharArray(batBufHun, 2);
    String(batTen).toCharArray(batBufTen, 2);
    String(batOne).toCharArray(batBufOne, 2);
```

```
line1[10]=batBufHun[0]; //oppdaterer hva skjermen skal vise
line1[11]=batBufTen[0];
line1[12]=batBufOne[0];
line1[13]='%';

Wire.beginTransmission(LCD_info);//sender data til skjermen
Wire.write(0xFE);
Wire.write(0x45);
Wire.write(line[0]);
Wire.write(line1);


int tensA = ((int)A.portVoltage-((int)A.portVoltage%10))/10; //konverterer fra triggerpsenning til sifre for vis
int onesA = (int)A.portVoltage%10;
int tenthsA = 10*(A.portVoltage-(tensA*10+onesA));

char Abuffer10[2], Abuffer1[2], Abuffer01[2];

String(tensA).toCharArray(Abuffer10, 2);
String(onesA).toCharArray(Abuffer1, 2);
String(tenthsA).toCharArray(Abuffer01, 2);

char Abufferinn[2], Abufferut[2];

String(A.triggerInStatus).toCharArray(Abufferinn, 2); //viser status til trigger inn og ut som en 1 eller 0
String(A.triggerOutStatus).toCharArray(Abufferut, 2);

line2[4]=Abuffer10[0]; //oppdaterer hva som sendes til skjermen på line 2
line2[5]=Abuffer1[0];
line2[6]='.';
line2[7]=Abuffer01[0];
line2[8]='V';
line2[9]=' ';
line2[10]='I';
line2[11]='n';
line2[12]='n';
line2[13]=':';
line2[14]=Abufferinn[0];
line2[15]=' ';
line2[16]='U';
line2[17]='t';
line2[18]=':';
line2[19]=Abufferut[0];

Wire.write(0xFE);
Wire.write(0x45);
Wire.write(line[1]);
Wire.write(line2);



int tensB = ((int)B.portVoltage-((int)B.portVoltage%10))/10;
int onesB = (int)B.portVoltage%10;
int tenthsB = 10*(B.portVoltage-(tensB*10+onesB));

char Bbuffer10[2], Bbuffer1[2], Bbuffer01[2];

String(tensB).toCharArray(Bbuffer10, 2);
String(onesB).toCharArray(Bbuffer1, 2);
String(tenthsB).toCharArray(Bbuffer01, 2);

char Bbufferinn[2], Bbufferut[2];

String(B.triggerInStatus).toCharArray(Bbufferinn, 2);
```

```
   String(B.triggerOutStatus).toCharArray(Bbufferut, 2);

   line3[4]=Bbuffer10[0];
   line3[5]=Bbuffer1[0];
   line3[6]='.';
   line3[7]=Bbuffer01[0];
   line3[8]='V';
   line3[9]=' ';
   line3[10]='I';
   line3[11]='n';
   line3[12]='n';
   line3[13]=':';
   line3[14]=Bbufferinn[0];
   line3[15]=' ';
   line3[16]='U';
   line3[17]='t';
   line3[18]=':';
   line3[19]=Bbufferut[0];

   Wire.write(0xFE);
   Wire.write(0x45);
   Wire.write(line[2]);
   Wire.write(line3);



  Wire.write(0xFE);
  Wire.write(0x45);
  Wire.write(line[3]);
  Wire.write(" ");

 Wire.endTransmission();
   infoScreenUpdate=0;
}

if(funcScreenUpdate==1){
   Wire.beginTransmission(LCD_func); //Start og tøm skjermen
   Wire.write(0xFE);
   Wire.write(0x51);
   Wire.endTransmission();
   delay(2);

   for(int i = 0; i<4; i++){ // skriv på skjermen de forskjellige menyvalgene
     Wire.beginTransmission(LCD_func);
     Wire.write(0xFE);
     Wire.write(0x45);
     Wire.write(line[i]);
     Wire.write(funcMenu[menulevel][i]);
     Wire.endTransmission();
   }
   funcScreenUpdate=0;
}


 //----------capture-----------
 while(captureCondition==1){
   if(firstLoop==1){ //kjører bare første gangen i løkken
     Serial.println("New Capture");
     analogWrite(A.triggerInPWMPin, A.triggerInStatus*triggerVoltageToDuty(A.portVoltage));//lager terskelspenninge
     analogWrite(B.triggerInPWMPin, B.triggerInStatus*triggerVoltageToDuty(B.portVoltage));

     firstLoop=0; //endrer denne for at disse funksjonene ikke kjører igjen
     captureDone=1; //denne variablen er ansvarlig for at funksjoner kjører EN gang etter capture-løkken

     attachInterrupt(digitalPinToInterrupt(A.triggerInReadPin), portAcheckCapture, RISING);//slår på interrupts for
```

```
    attachInterrupt(digitalPinToInterrupt(B.triggerInReadPin), portBcheckCapture, RISING);

    SerialInPortA.begin(34800); //åpner kommunikasjon for de forskjellige seriell portene
    SerialInPortB.begin(34800);

    delay(50);
    Serial.print("Serial port A availability: ");
    Serial.println(SerialInPortA.available());
    Serial.print("Serial port B availability: ");
    Serial.println(SerialInPortB.available());
    for(int i = 0; i < 1000; i++){
      SerialPortAPermData[i]="";
      SerialPortBPermData[i]="";
    }

  }

  if(digitalRead(but1.buttonPin)==HIGH){ //denne er ansvarlig for å avslutte dataprosessen
    captureCondition=0;
  }

  if(A.capture==1){
    char characterA = SerialInPortA.read(); //programmet kan bare lese fra seriell en karakter om gangen, der "\n"
    if (characterA != (char)255){
      SerialPortATempData.concat(characterA); //legger til data som ikke er -1 til midlertidig data
      Serial.print("A: ");
      Serial.print((int)characterA);
      Serial.println(characterA);
    }
    if (characterA == '\n') //om linjen slutter, skal linjen lest legges til arrayen med permanent data, så resett
      {
        SerialPortAPermData[SerialPortAPos]=SerialPortATempData;
        SerialPortATempData = "";
        SerialPortAPos++;
      }
  }

  if(B.capture==1){
    char characterB = SerialInPortB.read();
    if (characterB != (char)255){
      SerialPortBTempData.concat(characterB);
      Serial.print("B: ");
      Serial.print((int)characterB);
      Serial.println(characterB);
    }
    if (characterB == '\n')
      {
        SerialPortBPermData[SerialPortBPos]=SerialPortBTempData;
        SerialPortBTempData = "";
        SerialPortBPos++;
      }
  }

}

//------------------etter capture--------------
if(captureDone==1){
  //Reset variabler og interrupts om de ikke var triggered
  captureDone=0;
  firstLoop=1;
  SerialPortAPos=0;
  SerialPortBPos=0;
  SerialInPortA.end();
  SerialInPortB.end();
  detachInterrupt(digitalPinToInterrupt(A.triggerInReadPin));
```

```
    detachInterrupt(digitalPinToInterrupt(B.triggerInReadPin));
    SD.begin(); //Starter SD kortet for å skrive til det


    if(A.capture==1){ //skriver bare til SD kortet om det var mottatt data fra den porten
      A.capture=0;
      captureDataA = SD.open("/DataA.txt", FILE_WRITE); //åpner/lager en fil med navn DataA.txt
      for(int y=0; y<1000; y++){
        if(SerialPortAPermData[y]!=""){
          captureDataA.print(SerialPortAPermData[y]);
          Serial.println(SerialPortAPermData[y]);
        }
      }
      captureDataA.close(); //lukker filen
    }


    if(B.capture==1){
      B.capture=0;
      captureDataB = SD.open("/DataB.txt", FILE_WRITE);
      for(int y=0; y<1000; y++){
        if(SerialPortBPermData[y]!=""){
          captureDataB.print(SerialPortBPermData[y]);
          Serial.println(SerialPortBPermData[y]);
        }
      }
      captureDataB.close();
    }

    SD.end(); // lukker sd kortet, så det kan bli plugget ut uten å måtte slå av tetutstyret
  }

}



//Justerer tærskelspenningen for trigger inn
//tar inn spenningog legger til eller tar fra 0.1V fra spenningen
//Spennigen kan ikke være høyere enn 15V, og ikke lavere enn 0V
//trenger en funksjon for hver port og hver retning på grunn av hvordan funksjon-arrayen er satt opp
float setTriggerVoltageDownA(float voltage){
  if(A.portVoltage>0){
    voltage=A.portVoltage;
    A.portVoltage=voltage-0.1;
    infoScreenUpdate=1;
  }
  return(menulevel);
}

float setTriggerVoltageUpA(float voltage){
  if(A.portVoltage<=15){
    voltage=A.portVoltage;
    A.portVoltage=voltage+0.1;
    infoScreenUpdate=1;
  }
  return(menulevel);
}

float setTriggerVoltageDownB(float voltage){
  if(B.portVoltage>0){
    voltage=B.portVoltage;
    B.portVoltage=voltage-0.1;
    infoScreenUpdate=1;
  }
  return(menulevel);
```

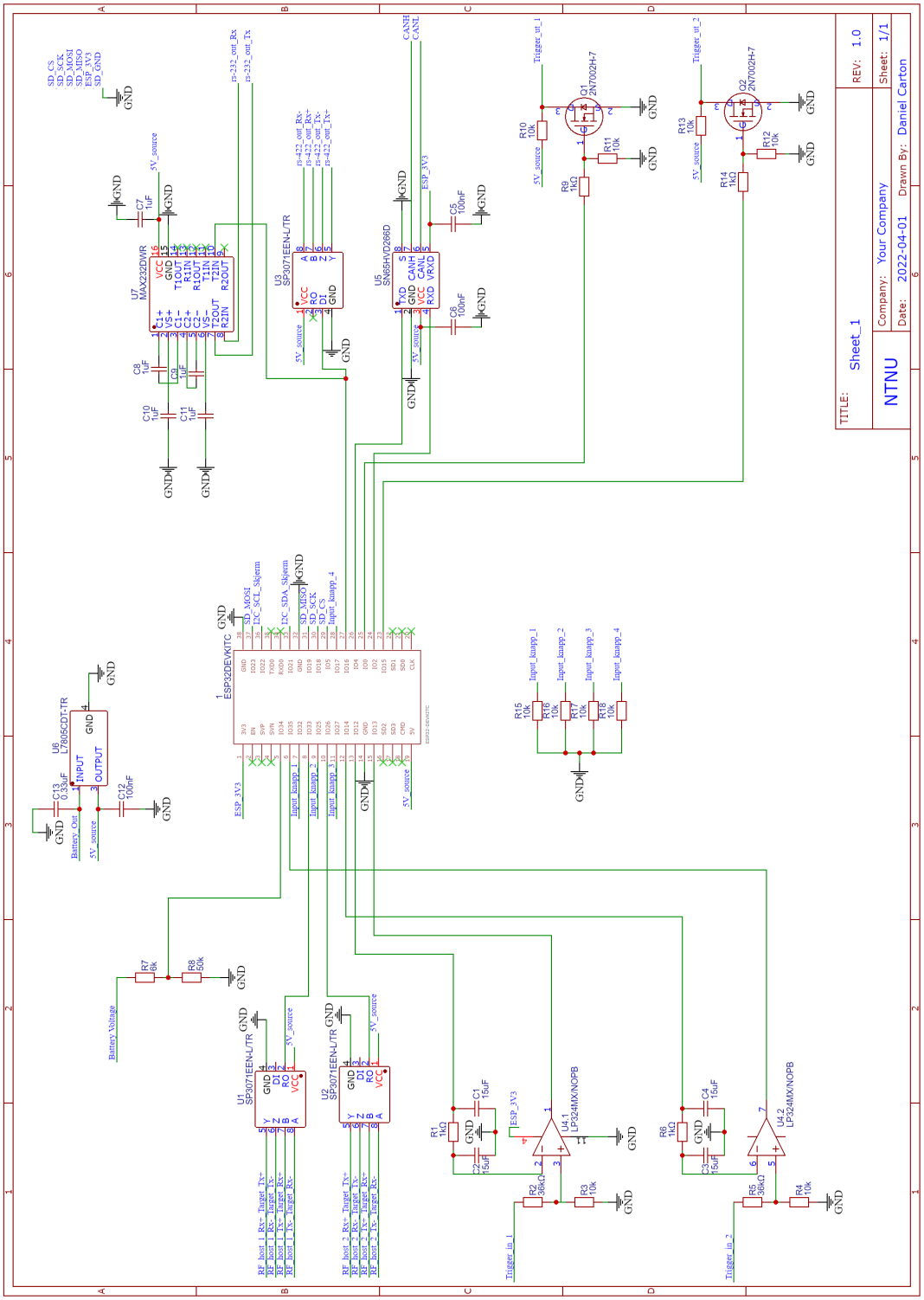
```
}

float setTriggerVoltageUpB(float voltage){
  if(B.portVoltage<=15){
    voltage=B.portVoltage;
    B.portVoltage=voltage+0.1;
    infoScreenUpdate=1;
  }
  return(menulevel);
}

//en funksjon for å
returnere verdien sendt til den, dette er en nødvendig funksjon for menyvalg som bare videresender til en annen meny
float returnA(float a){return(a);}


//Bytter trigger status for trigger inn og ut, for port A eller B. Å
bruke denne funksjonen vil bytte til det motsatte av hva som er satt nå
float byttTriggerUt(float Port){
  if((int)Port==0){
    A.triggerOutStatus=(A.triggerOutStatus+1)%2;
  }
  if((int)Port==1){
    B.triggerOutStatus=(B.triggerOutStatus+1)%2;
  }
  infoScreenUpdate=1;
  return(menulevel);
}

float byttTriggerInn(float Port){
  if((int)Port==0){
    A.triggerInStatus=(A.triggerInStatus+1)%2;
  }
  if((int)Port==1){
    B.triggerInStatus=(B.triggerInStatus+1)%2;
  }
  infoScreenUpdate=1;
  return(menulevel);
}

//funksjon for å starte en capture
float beginCapture(float addr){
  captureCondition=1;
  return(addr);
}

//funskjon som kalkulerer en duty cycle fra 0 til 255 ut fra en gitt inngangspenning
int triggerVoltageToDuty(float TriggerVoltage){
  float duty = TriggerVoltage * (10.0 / 46.0) * (256.0 / 3.26);
  return ((int)duty);
}


//interruptfunksjonene for å
starte henting av data for de forskjellige portene. interrupten vil bli slått av selv om porten blir brukt eller ikk
void portAcheckCapture(){
  if(A.triggerInStatus==1){
    A.capture=1;
    SerialInPortA.flush();
  }
  detachInterrupt(A.triggerInReadPin);
}

void portBcheckCapture(){
  if(B.triggerInStatus==1){
```

```
    B.capture=1;
    SerialInPortB.flush();
  }
  detachInterrupt(B.triggerInReadPin);
}
```

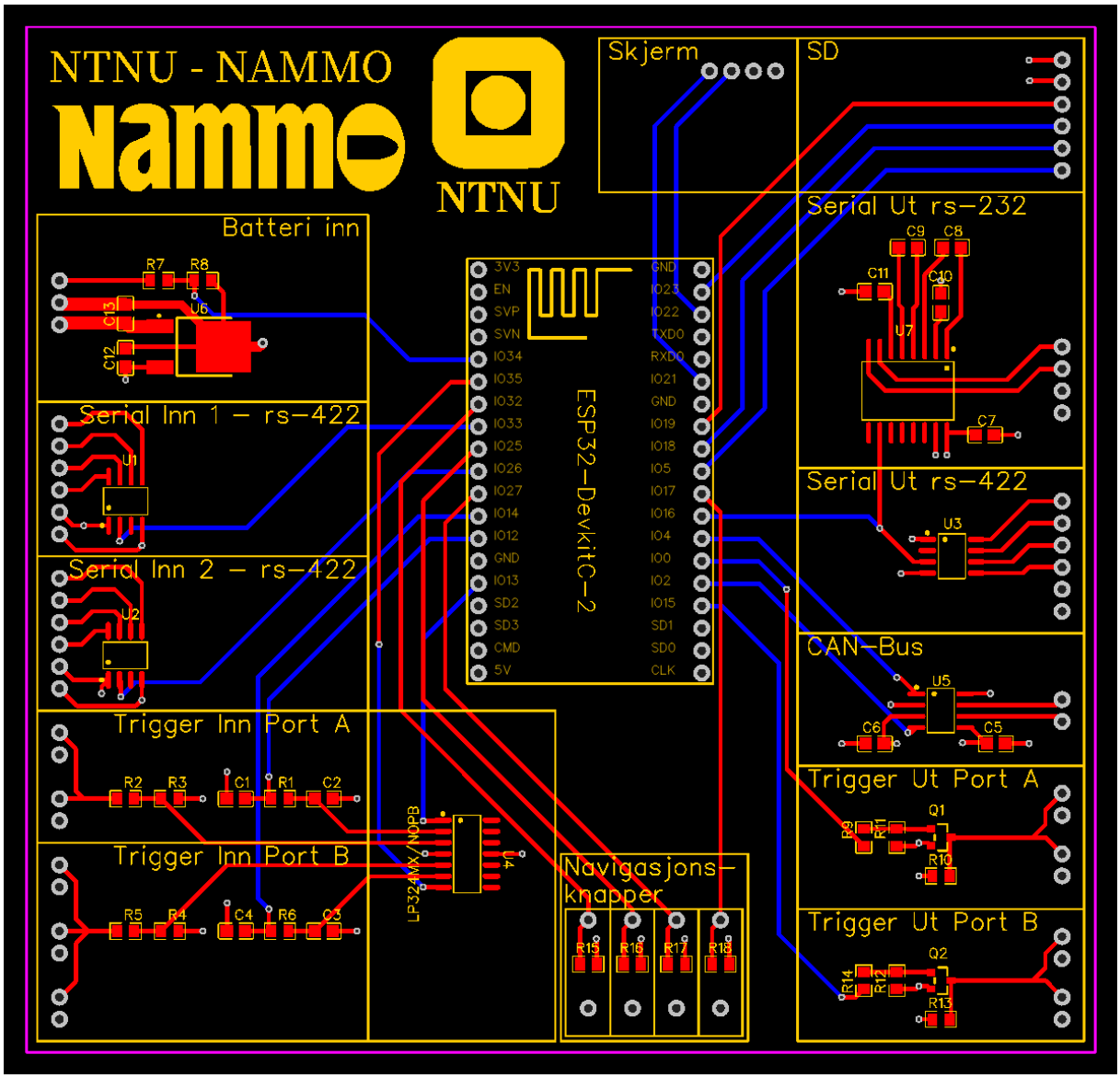
Vedlegg 2: Kode for sekundær testutstyr

```
#include <SoftwareSerial.h>

SoftwareSerial SerialInPortA(255, 22);


void setup() {
  Serial.begin(115200);
  SerialInPortA.begin(34800);
  pinMode(4, OUTPUT);
}

void loop() {
  for(int i = 0; i<650; i++){
    SerialInPortA.print("Number: ");
    SerialInPortA.println(i);
    Serial.print("Number: ");
    Serial.println(i);
    delay(150);
    if(i==16){
      Serial.println("triggering");
      digitalWrite(4, HIGH);
      delay(1);
      digitalWrite(4, LOW);
    }
  }
  return;
}
```

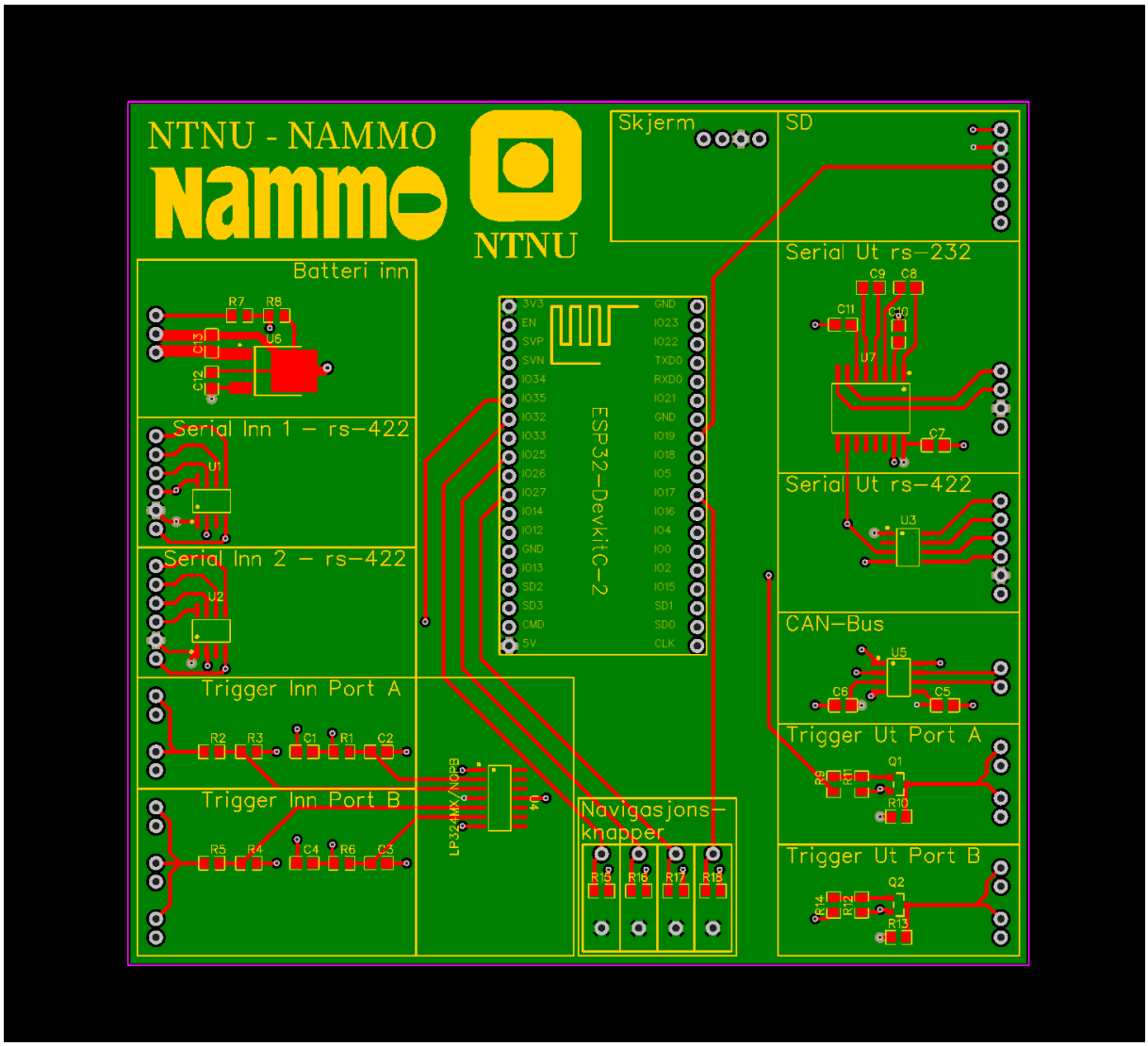
Vedlegg 3: Kretsdiagram



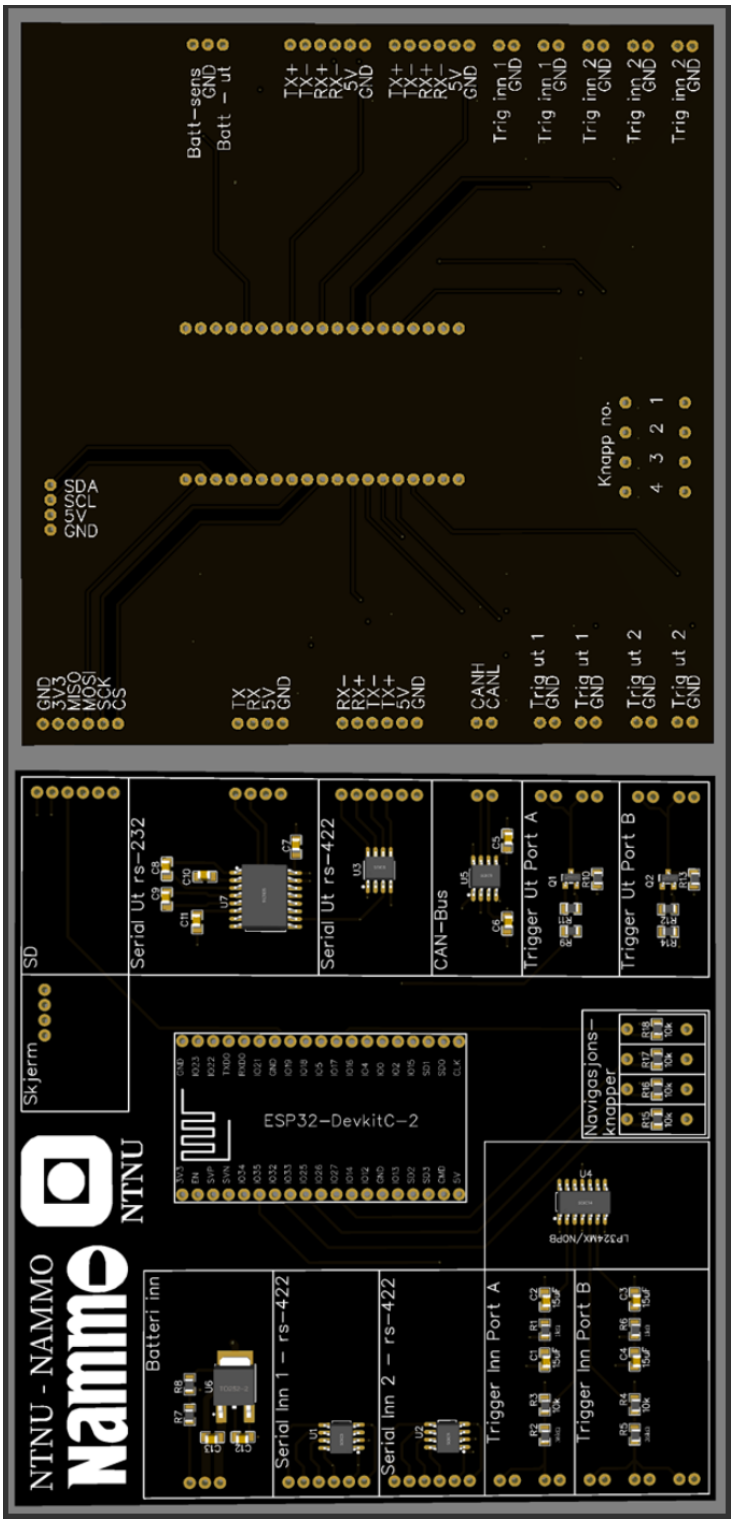
Vedlegg 4: PCB utlegg uten kopperfelt



Vedlegg 5: PCB utlegg med kopperfelt



Vedlegg 6: Kretskort

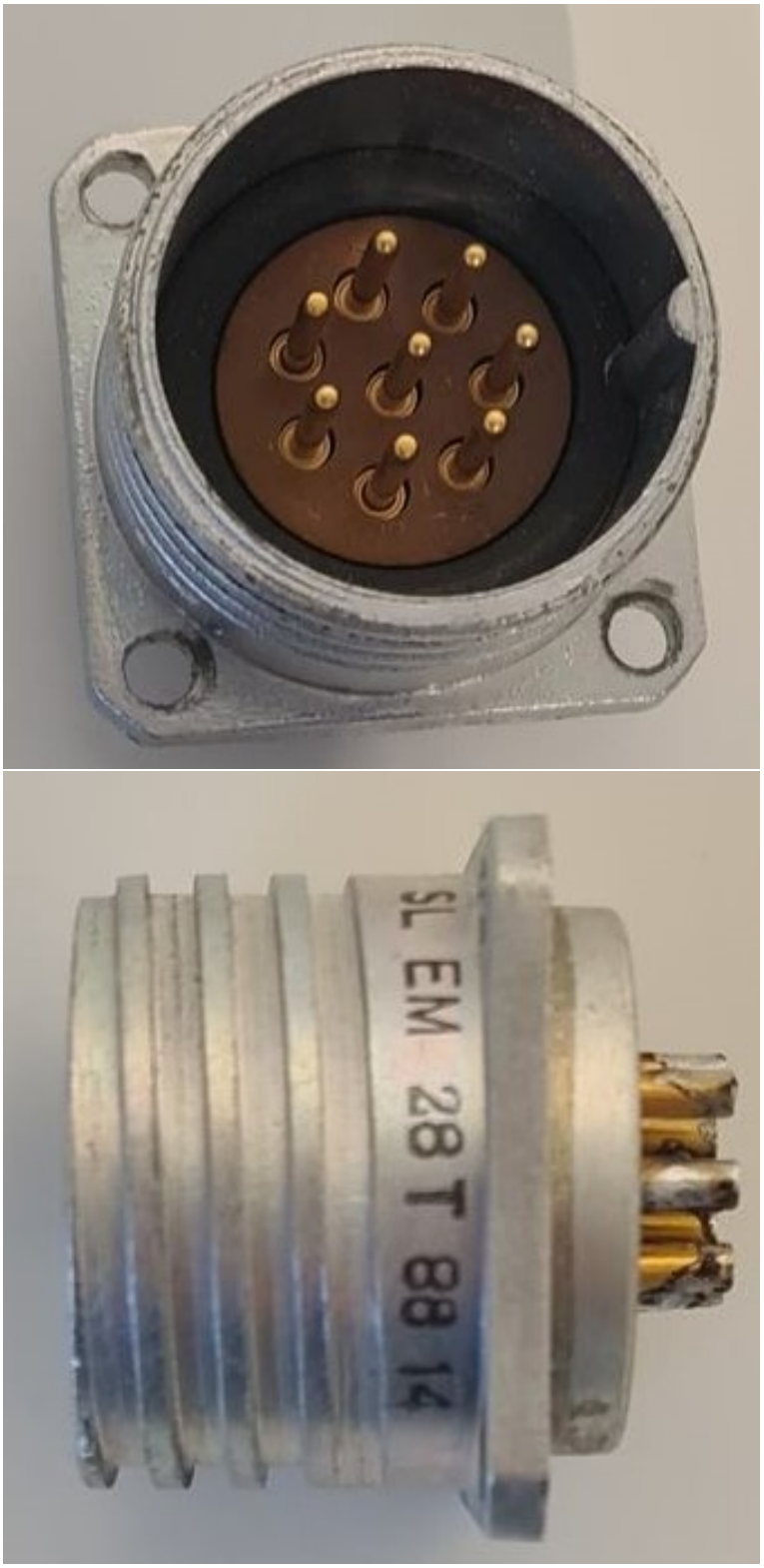


Vedlegg 7: Ferdig produkt

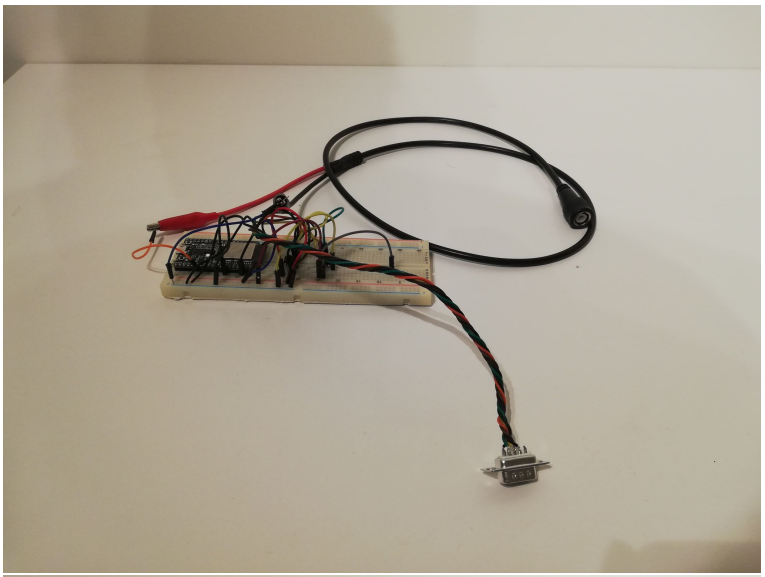
Vedlegg 8: Tabell av GPIO-pinner

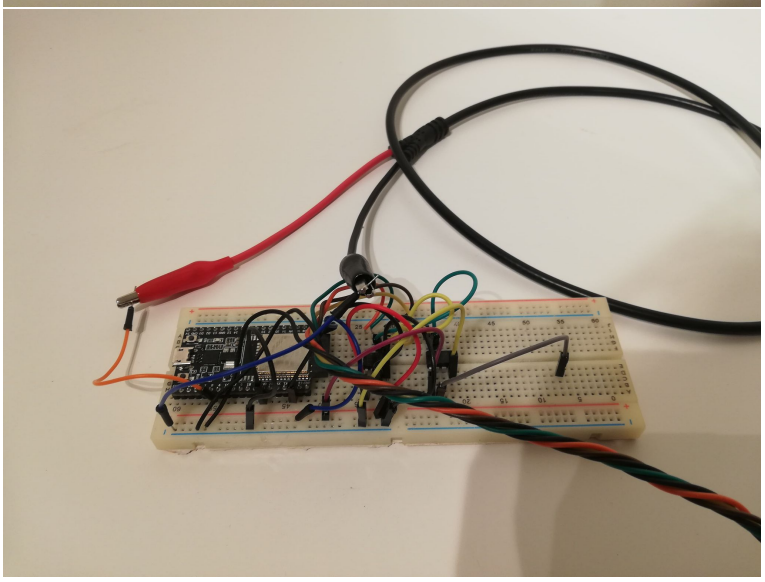
Type	**Type pinner**	**Pin nums + adresses**
2x Trigger innsignal	4x GPIO (2x pwm, 2x digitalRead)	**PWM1:** 12, **Trigger1:** 13, **PWM2:** 14, **Trigger2:** 35
2x trigger utsignal	2x GPIO	**Trigger 1:** 0, **Trigger 2:** 15
4x navigasjonsknapper	4x GPIO (4xdigitalRead)	**1:** 17, **2:** 32, **3:** 25, **4:** 27
2x Rs422 in for RF	2x GPIO	**Rx1:** 33, **Tx1:** 255 (recieve data only), **Rx2:** 26, **Tx2:** 255 (recieve data only)
Serial ut	1x GPIO (1x softwareserial)	**TX:** 16, **RX:** 255 (transmit data only)
Sd-kort	1x MOSI, 1x MISO, 1x SCK, 1x CS	**Mosi:** 23, **Miso:** 19, **SCK:** 18, **CS:** 5
Skjerm	1x SDA, 1x SCL	**SDA:** 21, **SCL:** 22
Batterinivå	1x GPIO (analogRead)	34
CAN	2x GPIO	**RX:** 2, **TX:** 4

Vedlegg 9: 8-pin 2M805 konnektor



Vedlegg 10: Sekundert testutstyr





Vedlagg 11: Blokkdiagram