NTNU

Kunnskap for en bedre verden

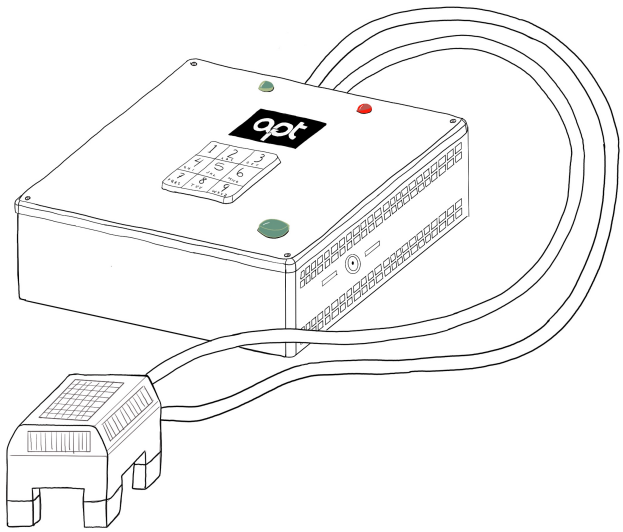
INSTITUTT FOR ELEKTRONISKE SYSTEMER

Videreutvikling av styringsenhet og brukergrensesnitt av system for diabetesbehandling med nær-infrarødt lys

IELET2900 - BACHELOROPPGAVE ELEKTRONIKK OG SENSORSYSTEMER

Gruppemedlemmer:
Ahmed, Taheera
Fyksen, Mina Johanne Eftestøl
Havmo, Anne Elise Ljosdal
Steen, Vilma Andrea

29. mai 2023



Figuren er hentet fra masteroppgaven til
Patrick C. Bösch med tillatelse fra APT [1]

NTNU
Kunnskap for en bedre verden

Tittelside Bacheloroppgave BIELEKTRO

Oppgavetittel (Norsk og Engelsk): Videreutvikling av styringsenhet og brukergrensesnitt av system for diabetesbehandling med nær-infrarødt lys Further development of control unit and user interface of system for diabetes treatment with near infrared light	
Forfattere: Anne Elise Ljosdal Havmo Mina Fyksen Vilma Andrea Steen Taheera Ahmed	**Prosjektnummer:** E2322 **Innleveringsdato:** 26.05.2023 **Gradering:** (x) Åpen () Lukket
Studium: Elektroingeniør - BIELEKTRO	
Studieretning: Elektronikk og sensorsystemer	
Veileder internt: Herman Ranes **Institutt:** Institutt for elektroniske systemer	
Oppdragsgiver: The Artificial Pancreas Trondheim (APT) **Kontaktperson:** Anders Lyngvi Fougner, anders.fougner@ntnu.no, 97158863 Patrick C. Bösch, patrick.c.bosch@ntnu.no, 40109318	
Stikkord norsk: Nær-infrarødt lys, mikrokontrollere, kretskort design, programmering	**Stikkord engelsk:** Near-infrared light, microcontroller, PCB design, programming

Sammendrag

Oppgaven omhandler utviklingen av en prototype for å undersøke effekten av nær-infrarødt lys (NIR) på insulinopptaket i kroppen. Diabetes type 1 er en sykdom som påvirker mange mennesker rundt om i verden og dette krever jevnlige insulininnsprøytninger. Denne prosessen kan være tidkrevende og innebære risiko ved feildosering. Ved å redusere tiden det tar for kroppen å reagere på insulin, kan doseringen og risiko for feil reduseres.

Forskningsgruppen Artificial Pancreas Trondheim (APT) ved NTNU har utviklet maskinvaren til en prototype som skal brukes i forskning på bruken av NIR-lys for å øke blodgjennomstrømningen og dermed insulinopptaket. Prototypen består av et LED-hode som brukes for å påføre NIR-lys og en styringsenhet som LED-hodet skal kobles til.

I denne oppgaven har det blitt utviklet kode for styringsenheten slik at man kan lese av 4 temperatursensorer på prototypen, innehnting av dato og tid fra en real-time clock (RTC), styre LED-hodet og batterilevetiden. Videre har det også blitt laget et system for å loggføre forsøkene som utføres på pasientene og et system for å håndtere eventuelle feilmeldinger ved systemet og maskinvaren. Det har også blitt laget et forslag til forbedringer ved designet til etuiet til styringsenheten. Den originale prototypen til APT brukte to DIP-brytere for å kontrollere forskjellige innstillinger ved forsøket, disse har blitt erstattet til fordel for en skjerm og et tastatur.

Det har blitt avdekket feil ved den originale prototypen, deriblant at temperatursensorene ikke fungerer som forventet og diverse mangler ved LED-hodet. Det har også blitt funnet muligheter for forbedringspotensialer ved designet av kretskortet.

Prosjektets forløp har resultert i en prototype som snart kan benyttes innen diabetesforskning. Det siste arbeidet som gjenstår er å flette sammen koden, teste koden på prototypen og 3D-printe det nye etuiet. Samt. å forbedre feilene som ble avdekket underveis.

Summary

The thesis deals with the development of a prototype to investigate the effect of near-infrared light (NIR) on insulin uptake in the body. Type 1 diabetes is a disease that affects many people around the world, requiring regular insulin injections. This process can be time-consuming and carry risks of dosage errors. By reducing the time it takes for the body to respond to injected insulin, the risk of dosing errors can be minimized.

The research group Artificial Pancreas Trondheim (APT) at NTNU has developed the hardware for a prototype to be used in research on the use of NIR light to increase blood flow and thus insulin uptake. The prototype consists of an LED head used to apply the NIR light and a control unit to which the LED head is connected.

In this task software has been developed for the control unit to read temperature sensors on the prototype, a real-time clock (RTC), control the LED head, and manage battery life. Furthermore, a system has been created to log the experiments conducted on the patients and handle any error messages related to the system and hardware. An improved design for the casing of the control unit has also been developed. The original prototype from APT used two DIP switches to control different settings during the experiment, which have been replaced with a screen and a keypad.

Issues have been identified with the original prototype, including the malfunctioning temperature sensors and deficiencies in the LED head. Opportunities for improvement have also been discovered with regards to the design of the printed circuit board.

The project has resulted in a prototype that can soon be utilized in diabetes research. The remaining work consists of integrating the code, testing it on the prototype and 3D printing the new casing.

Innhold

Figurer

Tabeller

Ordliste

Altium	Design program som blant annet blir brukt til design av kretskort.
APT	”Artificial Pancreas Trondheim”, forskningsgruppe ved NTNU som forsker på teknologi knyttet til diabetes.
Arduino Mega 2560 Rev3	Utviklingskort produsert av Arduino med en ATmega2560 mikrokontroller [2].
ATmega2560	Mikrokontroller prosessor produsert av Microchip [3].
Bitmappe	Skape et bilde på en skjerm hvor hver piksel korresponderer til en eller flere bits i minnet [4].
CAD	”Computer Aided Design” er en programvare som brukes til dataassisterte konstruksjoner av 3D moduleringer.
CSV	”Comma Separated Values” er et filformat hvor data blir lagret med komma separering.
EEPROM	”Electrically erasable programmable read-only memory” er en minnetype. Den vil ikke slette data lagret på minnet når den mister strømtilførsel.
Ekstern periferi	Beskriver komponenter og funksjonaliteter som er bygd på en mikrokontroller. Eksemelvis: LED, trykk knapper og andre sensorer.
Grensesnittskort	Et grensesnittskort er en enhet som brukes til å gjøre tilkoblingen og bruk av spesifikke elektroniske komponenter enklere. Grensesnittkortet fungerer som et mellomledd mellom en mikrokontroller eller et hovedkort og den spesifikke komponenten som trenger å bli tilkoblet.

I2C	”Inter-Integrated Circuit” er en kommunikasjonsprotokoll for overføring av seriell data. Patentert av Philips [5].
Intern periferi	Beskriver funksjonaliteter som er innebygd og kan aksesseres på en mikrokontroller.
ISR	”Interrupt Service Routine” er en delkode som kjøres når et avbrudd er utløst.
LED	”Lysemitterende Diode” er en type diode som stråler ut lys ved strømgjennomgang.
LED-hode	Den delen av prototypen som påfører NIR-belysningen.
Mikrokontroller	En liten programmerbar elektronisk enhet.
NIR	”Nært Infrarødt Lys” befinner seg utenfor det elektromagnetiske spekteret for synlig lys.
Oppregningstyper	En oppregningstype, også kjent som en enum, er en datatype i programmering som lar deg definere en liste over navngitte konstanter.
Pathing	”Pathing” brukes i forhold til planlegging av veien og bevegelsene til printerhodet.
PBM	”Pulsbreddemodulasjon” blir brukt om å sende pulser av spenning med bestemt frekvens.
PLA	”Polylactic acid” eller polymelkesyre er et materiale som typisk blir brukt for 3D-printing.
Prototype v.1	Prototypen til lysterapisystemet utviklet av Patrick C. Bösch [1].
Prototype v.2	Prototypen til lysterapisystemet videreutviklet som en del av bacheloroppgaven.
RTC	”Real Time Clock” er en integrert krets som skal operere som en nøyaktig teller.
Rx	”Recieve” brukes om en overføring der det skal mottas data.
SCL	”Serial Clock Signalbuss” som benyttes for klokkesignal i TWI kommunikasjon.
SDA	”Serial Data Signalbuss” som benyttes for dataoverføring i TWI kommunikasjon.

SPI	”Serial Peripheral Interface” er en synkron seriell kommunikasjonsprotokoll som brukes for å kommunisere mellom enheter, slik som sensorer, skjermer, minneenheter og mikrokontrollere.
Styringsenhet	Boksen som som styrer LED-hodet og NIR-lyset.
Switch-setning	Kode funksjon som tar inn en variabel og utfører ulike handlinger utifra en satt betingelse.
Tekstfil	En TXT-fil (også kjent som en tekstfil) er en type filformat som inneholder ren tekst uten noen form for formatering eller struktur.
Tilstandsmaskin	System som har gitte oppgaver ved ulike statiske tilstander [6].
TWBR	TWI Bit Rate Register.
TWCR	TWI Controll Register.
TWDR	TWI Data Register.
TWI	”Two Wire Interface” er en kommunikasjonsprotokoll for seriell data overføring.
TWSR	TWI Status Register.
Tx	”Transmit” brukes om en overføring der det skal sendes data.
Via-hull	Via-hull er hull som kobler sammen ulike lag på et kretskort.

Kapittel 1

Innledning

1.1 Bakgrunn

Diabetes type 1 påvirker mennesker i alle aldere over hele verden. Det er ingen kjent årsak til sykdommen, men den utløses ofte av en kombinasjon av ukjente miljøfaktorer og arvelige faktorer. Sykdommen fører til økt risiko for hjertesykdommer, hjerteinfarkt, nyresvikt samt skader på føtter, øyne og tenner. I 2017 var det 9 millioner mennesker som levde med diabetes type 1, og per dags dato finnes det ingen kjent kur for sykdommen [7]. Behandlingen av diabetes type 1 innebærer regelmessige insulininjeksjoner for å regulere blodsukkernivået. Det tar i gjennomsnitt 7,6 minutter før kroppen reagerer på den tilførte insulinen [1].

Prosessen er tidkrevende og kan kreve beregning av både dosering og injeksjonstidspunkt. Feil eller dårlig behandling kan medføre alvorlige komplikasjoner. For eksempel, dersom en person med diabetes tar for mye insulin, kan det resultere i insulinsjokk og føre til bevistløshet [8].

Ved å redusere opptakstiden av insulin i blodstrømmen kan dosering av insulin bli enklere å justere, og risikoen for feildosering vil minke. Artificial Pancreas Trondheim (APT) er en forskningsgruppe ved Norges teknisk-naturvitenskapelige universitet (NTNU) som har utført forskning på bruk av nær-infrarødt lys (NIR) for å påvirke opptakstiden av insulin i blodet. I samarbeid med APT har Patrick Christian Bösch utviklet en prototype for å kunne forske på om NIR-lys kan lokalt øke blodgjennomstrømningen som igjen gir økt insulinopptak. Prototypen ble utviklet med formålet å kunne gjennomføre ulike forsøk, inkludert bruk av NIR-lys, varmeeffekt og placeboeffekt.

1.2 Oppgaveteksten

Oppgavebeskrivelsen er utarbeidet i samarbeid med oppdragsgiver og prosjektgruppen i oppstartsfasen av prosjektet. Den er utarbeidet under forprosjektet Tillegg O med hensyn på møter med APT og oppgavebeskrivelsen gitt i høst som kan sees i Tillegg N. Oppgavebeskrivelsen som ble utviklet er som følger:
Forskningsgruppen APT ved NTNU har delt ut bacheloroppgaven med mål om å fullføre prototypen som ble designet av Patrick C. Bösch. Bacheloroppgaven innebærer å programmere mikrokontrolleren som skal styre LED-ene og sjekke tempeartursensorene. Den ferdige prototypen skal brukes

til forsøk og derfor må de målte dataene lagres. Bacheloroppgaven skal også inkludere å fullføre brukergrensesnittet med fokus på brukervennlighet som skal tilpasses for pasient og helsepersonell Tillegg O .

En detaljert beskrivelse av utgangspunktet for prosjektet, med forventningene fra APT er å finne i kapittel 2.

1.3 Rapportens oppbygging

Oppgaven er skrevet med en nyutdannet elektroingeniør som hovedmålgruppe. I tillegg er rapporten tilpasset slik at den kan brukes som referanse for oppdragsgiver APT ved videreutvikling av prototypen. Den består av syv kapitler med flere vedlegg til slutt. I tillegg vil en zip-fil medfølge rapporten, som vil bli henvist til under vedleggslisten.

Innledningen er det første kapittelet og gir en kort introduksjon med bakgrunn og oppgavetekst. Det andre kapittelet går i dybden for bakgrunnen til prosjektet og presenterer oppdragsgiver og prosjektet. Kapittel tre gir den nødvendige teorien for å forstå prosjektet. I kapittel fire beskrives ulike designvalg, produsert kode og valg av teknologier. I det femte kapittelet beskrives hva som er oppnådd som resultat i prosjektet. Kapitel seks drøfter ulike avvik og fremtidig arbeid, mens kapittel syv konkluderer prosjektet.

Kapittel 2

Bakgrunn

Dette kapittelet vil presentere utgangspunktet for prosjektet. Ved å presentere oppdragsgiver, gå grundig inn i forutsetningene som var satt ved prosjektstart, og arbeidet som lå til grunn for gjennomføringen, vil denne delen gi forståelse for gruppens utgangspunkt. Til slutt vil også gruppens tolkning av prosjektets innhold og omfang bli presentert.

2.1 Artificial Pancreas Trondheim (APT)

APT er en forskningsgruppe ved NTNU som ble etablert i 2013. De er en tverrfaglig gruppe bestående av forskere med ulik bakgrunner og høy kompetanse innen sine fagområder, deriblant bioteknologi, kjemi, matematisk modellering, intensivbehandling, anestesi, ingeniøremner og matematisk modellering.

Det langsiktige målet for APT sin forskning er å utvikle og implementere et system som kan overvåke glukosesystemet hos pasienter med diabetes type 1 og type 2, samt hos pasienter som trenger intensivbehandling [9].

2.2 Device for improved insulin absorption in diabetes type 1

Bacheloroppgaven baserer seg på masteroppgaven "Device for improved insulin absorption in diabetes type 1" skrevet av Patrick C. Bösch. Det har blitt utarbeidet en prototype basert på masteroppgaven. Denne prototypen skal brukes i forskning med mål om å gjøre livet til diabetikere lettere ved hjelp av NIR-lys. Mennesker er avhengig av produksjonen av insulin som skjer i bukspyttkjertelen. Dersom produksjonen hemmes, vil det føre til en økning i blodsukkernivået og kan videre medføre sykdommen diabetes. Diabetikere må regelmessig opprettholde blodsukkernivået sitt ved å injisere insulin. Når en diabetiker regulerer blodsukkernivået kan det oppstå forstyrrelser. Insulin tas opp langsomt i kroppen i forhold til blodsukkernivået som reagerer på mat. Dette er problematisk og kan føre til farlige svingninger i blodsukkernivået siden insulinet ikke motvirker blodsukkernivået sine raske endringer [10].

Det er mange teorier knyttet til forbedring av kroppens evne til å absorbere insulin. En av disse teoriene baserer seg på å utføre lysterapi dirkete på hudvevet for å øke blodsirkulasjonen. Dette kan oppnås gjennom stimulering av hudvevet ved oppvarming eller bruk av NIR-lys.

Forskningen skal undersøke om det er oppvarming av hudvevet eller andre effekter som påvirker insulinopptaket. I dette tilfellet er det ønskelig å forske på effekten av NIR-lys på blodsirkulasjonen. Under en studie utført av APT ble det oppdaget at eksisterende kommersielle lysterapisystemene ikke egner seg for forskning. Dette skyldes manglende tilpasningsevne og manglende mulighet for modifikasjoner av lysterapiinnstillingene. Derfor har APT besluttet å utvikle sitt eget lysterapisystem basert på en prototype utviklet av Patrick C. Bösch, som er mer tilpasset forskerne sitt miljø og bruksområder. Videre vil prototypen utviklet av Patrick C. Bösch bli refert til som prototype v.1 og den som har blitt videreutviklet under prosjekteringsfasen vil bli refert til som prototypen v.2 [1].

2.3 Prototype v.1

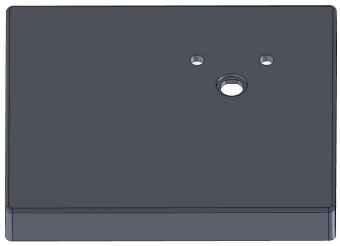
Maskinvaren for prototype v.1 ble ferdigutviklet i midten av Mars 2023, og den danner grunnlagetfor bacheloroppgaven. Prototypen består av en styringsenhet og et LED-hode. LED-hodet skal brukes til å påføre lysterapi under et forsøk, det er avtakbart og kan kobles fra styringenheten. Styringsenheten skal brukes til å kontrollere LED-hodet etter visse krav gitt fra forskeren. Resten av prototypen består av flere LED-indikatorer, vifter og temperatursensorer. Hele systemet er avbildet i Figur 2.2.

Styringsenheten består hovedsakelig av et 3D printet etui, og et kretskort med tilhørende elektronikk inni. Etuiet er en 3D-printet boks laget av svart polymelkesyre (PLA) plast. Den kan åpnes og lukkes for å gi tilgang til den øvre delen av maskinvaren på kretskortet. Dimensjonene er 19,0 cm i bredde, 16,5 cm i lengde og 7,3 (4,8 + 2,5) cm i høyden. Styringsenheten har tre tilkoblingsmuligheter: en USB-C-port, en mini-USB og to tilkoblinger for LED-hodet. Den har også to av/på-knapper, to LED-lys og en temperatursensor. Styringsenheten er avbildet i Figur 2.2b og Figur 2.1.

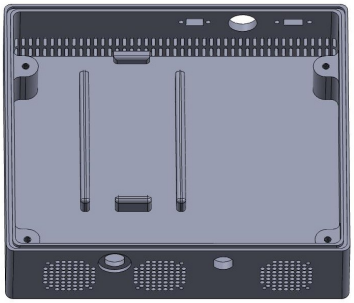
LED-hodet er en separat komponent som kan kobles til maskinvaren i styringsenheten, slik at LED-hodet er utskiftbart. Den er avbildet i Figur 2.2a. LED-hodet er en essensiell del av prototypen da den står for NIR-lysbehandlingen. Styringsenheten styrer LED-hodet for å sikre at det fungerer i samsvar med innstillingene for forsøket. LED-hodet har sitt eget kretskort og består av tre temperatursensorer, en vifte, en motstandsidentifikator og LED-lys for NIR-strålingen.

2.4 Funksjonsspesifikasjoner

Prototype v.1 skal brukes til medisinsk forskning innen diabetes. Overordnet skal prototypen brukes til å utføre forsøk for å teste effekten av NIR-lys på insulinopptaket i blodet. Dette innebærer å gjennomføre forsøk på pasienter med diabetes. Før et forsøk kan igangsettes må prototypen stilles inn med spesifikasjoner om hvilke innstillinger som skal benyttes. Disse skal settes av en forsker før prototypen tas i bruk av en pasient. Innstillingene inkluderer pasient identifikasjonsnummer, varighet av forsøket og modus for LED-hodet. Mer detaljert informasjon om alle innstillingene er forklart nærmere i underseksjon 2.4.1.



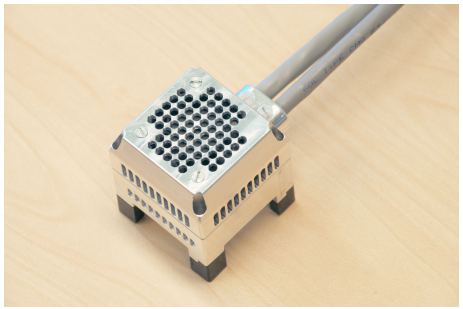
(a) 3D-modellering av etuiet til styringsenheten sitt topplokk.



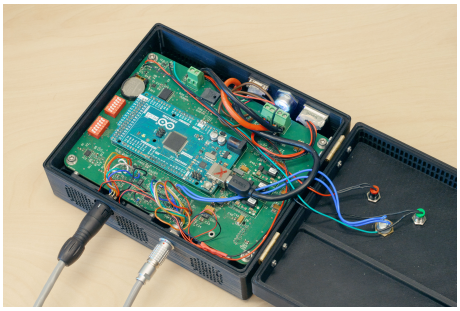
(b) Etuiet til styringsenheten sin bunn.

.

Figur 2.1: 3D-modellering av etuiet til styringsenheten til prototype v.1 som er satt sammen av et lokk og en bunn.



(a) LED-hodet som skal brukes for å påføre NIR-lyset på pasienten.



(b) Styringsenheten som skal brukes for å styre LED-hodet og innstillingene. Disse er skrevet mer om i seksjon 2.4.

.

Figur 2.2: Prototypen utviklet av Patrick C. Bösch. Foto kreditering: Anders Rønning Petersen v/Institutt for Teknisk Kybernetikk (ITK).

Innstilling	**Forklaring**	**Alternativer**
Pasientidentifikasjon	For å kunne identifisere en pasient i loggene	Et heltall mellom 0 og 99
Varighet	Tilsier hvor lenge NIR-lyset skal påføres på pasienten.	(1) 20 min (2) 30 min (3) 40 min
Frekvens for NIR-lys	Det skal kunne gå an å velge frekvens på NIR-lyset	(1) Kontinuerlig (2) Høyfrekvent (3) Lavfrekvent
Modus	Hvilket modus skal NIR-lyset settes til	(1) NIR-lys (2) Placebo (3) Randomisert forsøk

Tabell 2.1: Innstillinger for styringsenheten og LED-hodet som skal settes av forsker før et forsøk gjennomføres.

Funksjonaliteten til styringsenheten som er relevant for prosjektet er oppsummert i Tabell 2.2, mens for LED-hodet er det oppsummert i Tabell 2.3.

Maskinvaren til styringsenheten består av et egendesignet mønsterkort som inneholder flere komponenter, blant annet en mikrokontroller, to DIP-brytere, tilkoblingspunkt til vifter, en lagringsenhet og en temperatursensor. DIP-bryterne brukes til å sette innstillingene for forsøket. Viftene skal fungere som avkjøling for å opprettholde en stabil temperatur i styringsenheten. Lagringsenheten brukes til å loggføre forsøkene pasientene gjennomfører, slik at forskerne kan analysere dataene etter forsøket er gjennomført. Temperatursensoren i styringsenheten skal overvåke temperaturen på mønsterkortet. Styringsenheten har også to LED-indikator: en grønn LED som indikerer om et forsøk er igangsatt eller ikke, og en rød LED for å indikerer eventuelle feil i systemet. Begge LED-ene er plassert på lokket til styringsenheten på prototype v.1. Det er to knapper på styringsenheten, hvor den ene er plassert på siden av boksen. Denne knappen skal brukes til å skru av og på systemet. Den andre knappen er plassert på lokket og brukes til å igangsette forsøket.

	Komponent	**Funksjonalitet**
DIP-brytere	To på kretskortet	Den ene skal brukes for å sette pasient ID, den andre skal brukes for å sette innstillingene på forsøket.
Innganger	USB-C, mini-USB og tilkobling for LED-hodet	Lade styringsenheten og mulighet til å koble på LED-hodet.
Knapper	To stykker	(1) Skru av/på styringsenheten (2) Starte forsøk
Indikatorer	To LED-lys	(1) Grønn, indikerer om et forsøk er igangsatt (2) Rødt, indikere feil v/forsøket som har startet
Vifter	Tre stykker	Brukes for avkjøling av systemet
Temperatursensor	En på kretskortet	Brukes for å monitorere temperaturen på kretskortet
Lagringsenhet	En minneutvidelse for micro-SD-kort	Brukes for å loggføre forsøkene. Deriblant inneholde informasjon om innstillingene for et forsøk, om feil som oppstår og temperatursensor verdier underveis

Tabell 2.2: Oppsummering for ønsket funksjonalitet til styringsenheten som er relevant for prosjektet, utgangspunkt er prototype v.1.

På LED-hodet er det tre temperatursensorer som skal lese tre forskjellige temperaturer: temperaturen på kretskortet, temperaturen i luften nær NIR-strålingen og om LED-hodet har oppnådd hudkontakt eller ikke. Videre har LED-hodet en motstand plassert på mønsterkortet. Denne motstanden brukes til å identifisere hvilket LED-hode som er i bruk under forsøkene, ettersom den er utskiftbar. Det er LED-hodet som står for NIR-strålingen, dette gjøres av en påmontert LED-matrise. Varighet og frekvens av NIR-strålingen er allerede bestemt før forsøket begynner. Der kan man variere mellom 20-40 minutter på aktiv NIR-stråling, eller enten høy-

frekvent, lavfrekvent eller ingen stråling. I tillegg er det en vifte på LED-hodet for avkjøling. Avslutningsvis er det en motstandsindentifikator på LED-hodet.

	Komponenter	**Funksjonalitet**
Temperatursensor	Tre sensorer	Måler temperaturen (1) på kretskortet, (2) i luften i nærheten av NIR-strålingen og (3) hvorvidt LED-hodet har oppnådd hud-kontakt eller ikke.
Motstandsidentifikator	Motstand	Motstand som skal brukes for å identifisere hvilket LED-hode som brukes.
LED-matrise	En på kretskortet	Lyskilden til NIR-lyset og hovedpersonen for hele systemet. Når man initialiserer et forsøk skal man kunne velge mellom tre forskjellige modus (1) kontinuerlig påslått, (2) høyfrekvent pulsering og (3) lavfrekvent pulsering
Vifte	En vifte plassert over LED-matrise og PCB	Skal brukes for avkjøling av systemet

Tabell 2.3: Oppsummering for ønsket funksjonalitet til LED-hodet som er relevant for prosjektet.

Tidlig i prosjektfasen ble det også i sammarbeid med APT utarbeidet et flytskjema for hvordan det da var tenkt at flyten gjennom hele systemet skulle fungere. Dette flytskjemaet er vist i Tillegg A. Dette flytskjemaet beskriver kun utgangspunktet for arbeidet og reflekterer ikke det arbeidet som er gjort, men heller forventningene til systemet fra start.

2.4.1 Forsøksmetodikk

Når prototypen er ferdigutviklet, vil den bli brukt av både forskere og pasienter. Forskeren skal sette innstillingene på systemet slik at pasienten kan ta den i bruk.

En essensiell innstilling for forsøket er om det skal være placebo eller ikke. Et krav som var ønskelig fra oppdragsgiver var at det skulle være en funksjonalitet for blinding eller dobbelt blinding av forsøket. Vanlig blinding vil kunne være at forskeren selv bestemmer om forsøket som skal gjennomføres er placebo (f.eks. at NIR-lysene er avskrudd) eller vanlig (NIR-lys på-skrudd), men at prototypen uansett indikerer at det er påskrudd, slik at pasienten/brukeren ikke kan se forskjell. Dobbelt-blinding vil være at heller ikke forskeren vet om det er placebo/varme eller NIR-lys. Dette kan implementeres ved at systemet bruker randomisering for å avgjøre om det skal være placebo/varme eller NIR-lys. Pasienten og forsker er begge blindet, som betyr at det ikke er synlig om systemet kjøres i placebo-modus eller vanlig modus. Uansett om det er vanlig eller dobbelt blinding, må systemet lagre informasjon om hvilket modus som ble brukt. Denne informasjonen bør lagres et sted som forskeren ikke har tilgang til før etter at forsøket er gjennomført.

Kapittel 3

Teori

3.1 Diabetes type 1

Diabetes type 1 er en autoimmun sykdom. Et friskt immunforsvar eliminerer fremmede celler som ikke er ønsket i kroppen. Ved en autoimmun sykdom vil immunforsvaret også ødelegge egne friske celler. Diabetes type 1 er en tilstand hvor de friske betacellene i bukspyttkjertelen brytes ned. Det er disse cellene som produserer hormonet insulin. Insulin er nødvendig for å slippe glukosen i blodet inn i cellene som enten lagrer eller brukt det. Ved diagnosen diabetes type 1 vil ikke insulin bli skilt ut i blodet som normalt for å regulere blodsukkeret, og per dags dato finnes det ingen kjent kur [11][12].

Det finnes kun en behandling for diabetes type 1 som innebærer å sprøyte insulin direkte inn i underhudsfettet ved hjelp av en pumpe eller insulinpenn. Derfra blir insulinet tatt opp i blodet hvor det insulinet flytter glukosen til celler. Det tar i gjennomsnitt 7,6 minutter før denne prosessen starter, og kan ta opptil 40 til 49 minutter for maksimalt opptak av insulininjeksjonen . I tillegg vil maksimal effekt på blodsukkeret være etter ca 90 til 120 minutter [13, Fig.1d][1].

3.1.1 Nær-infrarødt (NIR) lys

Infrarød stråling er en del av det elektromagnetiske spekteret. Den er definert mellom 0,7 μm og 1 mm og kalles også varmestråling eller IR-stråling. Den tilhører ikke den delen av det elektromagnetiske spekteret som kalles ”synlig lys”. IR-stråling deles ofte i de fem kategoriene: nær-infrarødt, kortbølget infrarødt, mellominfrarødt, langbølget infrarødt og fjernt infrarødt. Nær-infrarød stråling, også kjent som NIR-lys, er definert fra 0,7 μm til 1,4 μm [14].

Det er forsket på om NIR-lys kan påvirke opptakstiden av insulin i blodstrømmen. Konseptet utnytter penetreringsdybden til NIR-lyset som kan være opp til noen millimeter. Dybden stråling kan penetrere er avhengig av flere faktorer som: bølgelengde, energi, bestrålt område og dempning fra materiale som stråles. Menneskelig vev er satt sammen av flere materialer som gjør det vanskelig å regne ut nøyaktig hvor dypt NIR-lyset kan penetrere. Et nær-infrarødt vindu (NIR-vindu) definerer hvilke bølgelengder mellom 650 nm og 1870 nm som penetrerer huden dypest. Det vil si: ved hvilke bølgelengder det menneskelige vevet har minimalt med dempning som gjør det optimalt for optisk behandling. I menneskelig vev er det tre slike vinduer: den første ved bølgelengden 650-950 nm, den andre ved 1100-1350 nm og den siste

ved 1600-1870 nm. I et NIR-vindu får NIR-lyset penetrert huden dypt nok til å utløse en lokal kjemisk reaksjon som produserer nitrogenoksider (NO). NO påvirker muskulaturen rundt blodårene. Den har en avslappende effekt som øker blodgjennomstrømningen. Økt blodgjennomstrømning medfører kjappere insulinopptak [1][15].

3.2 Altium Designer

Altium Designer er et avansert programvareverktøy. Det er et verktøy for utvikling av elektronisk komponenter og kretskort. Det er en plattform som lar ingeniører lage og redigere kretskortdesign på en brukervennlig og effektiv måte. Altium Designer har flere funksjoner slik som: kretsdesign, simuleringsverktøy, PCB-layout, 3D-modellering og produksjonsstyring. Det er enkelt for brukere å implementere store prosjekter og sikre høy kvalitet og pålitelighet i designet deres. Med så mange funksjoner tilbyr Altium Designer er en integrert arbeidsflyt fra første design til produksjonsklare filer. Dette gir mulighet for kontinuerlig feilsøking, simulering, analysering og optimalisering [16][17].

3.3 Visual Studio code

Visual Studio Code (VSCode) er en kodeeditor som gjør det mulig å laste ned forskjellige utvidelser som gjør kodeeditoren mer tilpasset etter behov. Den er kjent for sin evne til å til støtte en rekke ulike utvidelser som gjør programmet fleksibelt og medfører at det kan bli brukt under mange bruksområder [18].

PlatformIO

En utvidelsene av VSCode er PlatformIO. PlatformIO kan brukes for utvikling av innvevde programvaresystemer. Det er en samling av tjenester og verktøy som gjør det enklere å utvikle og bygge programvare for en rekke forskjellige mikrokontrollere og plattformer, deriblant Arduino, Raspberry Pi og mange fler. Det gir utviklere mulighet til å programmere mikrokontrollere ved hjelp av ulike programmeringsspråk som C++, Python og JavaScript [19].

3.4 Git og Github

Git [20] er et versjonskontrollverktøy som er designet til både små og store prosjekter. Det brukes for å spore endringer i filer og prosjekter over tid. Det lar utviklere samarbeide om koding og administrere prosjekter. Git er også godt integrert i VSCode. I kombinasjon med Git ble det bestemt å ta i bruk Github [21]. Github er en nettbasert plattform som tillater håndtering av kode som bruker Git. Det er en skybasert lagringsplass for kodeprosjekter.

3.5 3D-printing

3.5.1 SolidWorks

SolidWorks er et CAD program for design i både 2D og 3D. Det brukes til å visualisere, designe og simulere 3D-modeller av produkter. SolidWorks har funksjoner som gjør det egnet for bruk

i teknisk dokumentasjon, hvor man kan visualisere dimensjonering og lage oversiktlige 3D-animasjoner. Programmvaren er kjent for å være brukervennlig og kompatibelt med Microsoft Windows [22]. Programmet blir brukt av alt fra hobbydesignere, produktutviklere til ingeniører. SolidWorks har mulighet til å konvertere og eksportere 3D-modellene til STL-filer, som er et av de vanligste filformatene som 3D-printere bruker [23] [24].

3.5.2 UltiMaker

UltiMaker er en veletablert produsent for maskinvare, programvare og materialer for 3D-printing. UltiMaker selger 3D-printere og filamenter, og i følge deres nettsiden kan enhver filament med en diameter på 2.85 mm brukes med UltiMakersmaskinvare [25]. UltiMaker 3D-printere kommer i ulike serier, og har derfor forskjellige egenskaper og presisjoner. UltiMaker S5 serien er en robust 3D-printer i den større klassen. Den har fysisk byggevolum på 330x240x300 mm, og er utstyrt med glassdører som skjermer byggeplaten laget av glass fra ytrepåvirkninger i form av temperatur og vind. Dysen hvor filamentet presses ut av har fire forskjellige størrelser: 0,25 mm, 0,4 mm, 0,6 mm og 0,8 mm. S5 serien er kompatibel med SolidWorks og filtypen STL som er foretrukket format når 3D-modellen skjæres i programvaren UltiMaker Cura, men støtter også en rekke andre filformat og «plugin integrations» [26].

3.6 Two Wire Interface (TWI)

TWI kommunikasjonsprotokollen benytter seg av to busser for kommunikasjon mellom flere enheter. TWI er basert på master/slave prinsippet, masteren styrer kommunikasjonen til en eller flere slaver.
De to bussene som benyttes er:

- SDA: For overføringen av data.
- SCL: Fungerer som en felles klokkepuls for alle enhetene koblet til bussen.

SDA og SCL sine funksjoner er avhengige av hverandre. SCL styrer det som skjer på SDA busssen. Når SCL er lav kan data skrives til SDA, og når den er høy kan data leses fra SDA.

TWI protokollen gir mulighet for flere sammenkoblede enheter på bussen. Dette er fordi alle enheter som er designet for TWI kommunikasjon har en adresse som de vil respondere på. Adressene består av 7-bits, bit nummer 8 blir lagt til for å indikere typen kommunikasjon som pågår. Settes den høy skal det leses fra slaven. Settes den lav skal det skrives til [27].

3.6.1 Inter-Intergrated Circuit (I2C)

I2C er en kommunikasjonsprotokoll som tilsvarer TWI. Phillips har patent på navnet I^2C, derfor unngår flere produsenter navnet til fordel for TWI som i praksis er det samme. I dagligtalen blir begge protokollene nevnt om hverandre. I denne oppgaven er det hovedsakelig TWI som blir omtalt da denne protokollen benyttes av Arduino Mega 2560 [5][3].

3.7 Serial Peripheral Interface (SPI)

SPI er en synkron seriell kommunikasjonsprotokoll som brukes i mikrokontrollere for å kommunisere med andre enheter som sensorer, skjermer og minneenheter.

SPI-kommunikasjon innebærer en masterenhet og én eller flere slavenheter. Masterenheten initierer kommunikasjonen og kontrollerer timingen for dataoverføringen. Slavenhetene svarer på masterens kommando. Data overføres serielt, med et klokkesignal som driver timingen til overføringen [28, kap.16].

Grensesnittet for SPI består typisk av fire linjer:

1. MOSI (Master Output Slave Input): Denne linjen bærer data fra masterenheten til slavenheten(e).
2. MISO (Master Input Slave Output): Denne linjen bærer data fra slavenheten(e) til masterenheten.
3. SCK (Serial Clock): Denne linjen bærer klokkesignalet som synkroniserer dataoverføringen mellom master- og slavenhetene.
4. SS (Slave Select): Denne linjen brukes av masterenheten for å velge hvilken slavenhet den vil kommunisere med.

3.8 Digital til Analog omformer (DAC)

DAC er en komponent som tar inn en binær verdi og omformer den til en analog spenningsstørrelse. DAC-en har to referansespenninger, kalt V_{SS} og V_{DD}, som tilsvarer henholdsvis nedre og øvre grense for utgangsspenningen.

3.8.1 MCP4725

I designet til prototypen v.1 og v2 brukes en MCP4725, som er en 12-bit DAC. Det betyr at den har en oppløsning på: $(V_{DD} - V_{SS})/2^{12}$. Dette gir utregningen for utgangsspenningen V_{ut} i Ligning 3.1.

$$V_{ut} = \left(\frac{V_{DD} \cdot D_n}{4096}\right) \cdot \text{12bit\_DAC\_REGISTER} \tag{3.1}$$

For å skrive digitale verdier til MCP4725 brukes kommunikasjon via TWI med en master. Når en verdi skal skrives til DAC-en, kan den skrives til to ulike registre. Et av disse registerene er DAC-registeret, som brukest for å sette den analoge V_{ut} verdien. Registerene inneholder totalt 19-bits, der de 12 minst signifikante bit-ene er betegnet som D_n. Sistnevnte bits bestemmer størrelsen på V_{ut}, mens resten brukes til å lagre innstillinger for TWI-overføringen. Det andre registeret er EEPROM-minne, som inneholder fabrikkinnstillingene for DAC-en.

Hvis EEPROM-minnet ikke skrives til, vil DAC-en alltid bruke fabrikkinnstillingen på $0.5 \cdot V_{REF}$ etter frakoblet strøm [29].

3.9 Tastatur

Tastaturet som brukes i dette prosjektet har varenummer ECO.12150.06 fra produsenten. Det er et tastatur med 12 koblingsterminaler, med oppsett tilsvarende et tastatur på en telefon. Koblingsterminalene er satt sammen i en 4x3 matrise og kan kobles med syv hannkontakter. Sammenhengen mellom koblingsterminalene og pinnene er nærmere beskrevet i databladet under ”Component layout” [30, s.6].

3.10 Mikrokontroller og interne periferier

En mikrokontroller er en fellesbetegnelse for ulike programmerbare prosessorer. Når man bruker en mikrokontroller gjøres det et skille mellom ekstern periferier og intern periferier.

3.10.1 Avbrudd

Avbrudd er en intern periferi i de fleste mikrokontrollere. I programmering av maskinvare er avbrudd et nyttig verktøy for å utnytte prosessorens kapasitet maksimalt. En avbruddsvektor kan aktiveres og videre venter avbruddet på en spesifisert hendelse. Når avbruddsvektoren blir utløst, starter en ”Interrupt Service Routine” (ISR) som håndterer handlingene knyttet til avbruddet. Etter at avbruddshåndteringen er fullført, returnerer koden til der den var da avbruddet ble utløst [31]. Avbruddsprogrammering er en effektiv metode for å unngå venteløkker, der programmet venter på en betingelse før den kan gjennomføre neste oppgave. Istedet kan koden gå videre til andre oppgaver til betingelsen er satt.

3.10.2 Analog-til-digital omformer

Analog-til-digital omformer (ADC) er en elektronisk krets som omformer et analogt inngangssignal til et digitalt signal. I mange elektroniske applikasjoner er det nødvendig å konvertere et analogt signal, for eksempel ved avlesning av lyd, trykk, lysstyrke eller temperatur. Verdiene blir behandlet og omformet til digitale verdier av digitale systemer slik som datamaskiner eller mikrokontrollere. ADC-en utfører denne konverteringen ved å dele det analoge inngangssignalet i diskrete steg for å generere et digitalt tall som representere inngangssignalet. For eksempel vil en 10-bit oppløsning gi 1024 steg [32].

3.11 Pulsbreddemodulasjon (PBM)

PBM er en måte å manipulere signalet på en digital utgang ved å hurtig skru spenningen av og på. Når spenningen raskt veksler mellom av og på, vil prosentandelen der spenningen er høy påvirke den totale spenningen som oppfattes på utgangen. Det meste av elektronikk vil ikke oppfatte de raske endringene og vil ta inn gjennomsnittsverdien. Hvor lenge spenningen er høy blir kalt pulsbredde og er basert på en 8-bits teller. Altså vil pulsbredden kunne settes helt opp till 255, som tilsvarer en konstant på-spenning [33].

3.12 C og C++

C er et mye bruk programmeringsspråk innen elektroingeniørfaget på grunn av effektiviteten og nærheten til maskinvarespråket. Det er et strukturert språk som egner seg for utvikling av programvare for mikrokontrollere og innebygde systemer. Kodespråket C gjør det mulig å skrive kode som kan direkte kjøres på prosessoren og samhandle med elektroniske komponenter. Evnen til å håndtere lavnivåprogrammering gjør det mulig å kontrollere minneallokering og direkte håndtere maskinens ressurser. Det å mestre bitmanipulasjon er essensielt for manipulering av registre og elektroniske signaler. C tilbyr også en rekke bibliotekerfor enheter med for eksempel UART, TWI og SPI som protokoll. Samt. biblioteker for digital signalbehandling og numerisk beregning [34].

AVR C er en variant av C-programmeringsspråket. Det er tilpasset og optimalisert for mikrokontrollere fra Atmel AVR-familien. AVR C inkluderer funksjoner og biblioteker utviklet for mikrokontroller-baserte prosjekter. Det vil gi tilgang til mikrokontrollerens porter, registre og minneadresser som gjør det mulig å styre I/O-enheter [28].

C++ er en utvidelse av C-språket som legger til flere konsepter, datastrukturer og funksjoner. C++ inkluderer funksjoner med dynamisk minnehåndtering som krever mer minne- og prosessorkraft. Kodespråket gjør det enklere å håndtere komplekse systemer og interaksjoner mellom forskjellige enheter og moduler [35].

Kapittel 4

Metode

Denne delen beskriver arbeidsprosessen som har blitt utført gjennom prosjektets forløp. Metoden er grovt sett delt inn i fire deler. Den første beskriver hva som er bevart fra prototypen v.1, hva slags teknologi som er eliminert fra tidligere prosjekt og hva som er valgt å implementere i dette prosjektet på prototypen v.2, seksjon 4.1. Del nummer to omhandler design av styringsenheten til prototype v.2 hvor fremgangsmåten for konstruksjon av den fysiske prototypen v.2. Med beskrivelse av alle delene som tilsammen skaper det fysiske produktet, som kan finnes i seksjon 4.2. Den tredje delen omhandler programmering av prototype v.2, dette gjelder seksjon 4.3, seksjon 4.4, seksjon 4.5 og seksjon 4.6. Programmeringsdelen deles i fire deler hvor grunnstrukturen for kodebasen blir fremlagt etterfulgt av fremgangsmåte til de to programmeringsspråkene som er valgt. I den siste delen fremlegges et tiltenkt system for hvordan hele systemet skal operere når de enkelte delene er knyttet sammen.

4.1 Brukte teknologier og utstyr

En oppsummering av teknologier tatt i bruk for prosjektarbeidet kan sees i Tabell 4.1.

Det har blitt tatt i bruk ulike teknologier for å utvikle prototype v.2 Mikorkontrolleren som ble vallgt var Arduino Mega 2560. Kodespråket som ble brukt for utviklingen var C/C++ som ga mulighet for effektiv og strukturert kode. Versjonskontroll av kodebasen ble oppnådd ved hjelp av Git og GitHub. VSCode og PlatformIO ble brukt for koding og utvikling.

De samme programvarene som ble brukt i Patrick C. Bösch sin masteroppgave har også blitt benyttet for kretskortdesign og 3D-printdesign. Det samme gjelder valg av mikrokontroller.

Underveis i prosjektet har det blitt brukt forskjellig utstyr for feilsøking, videreutvikling og koding av prototype v.2. Utstyrlisten er ramset opp under.

- Oscillioskop
- Multimeter
- Spenningskilde
- Arduino Mega 2560
- Minneutvidelse (MicroSD card breakout board fra Adafruit)

Teknologi	**Valg**	**Seksjon i teori**
Mikrokontroller	Arduino Mega 2560	-
Kodespråk	C / C++	seksjon 3.12
Versjonskontroll	Git og Github	seksjon 3.4
Utviklingsplattform	Visual Studio Code og PlatformIO	seksjon 3.3
Kretskortdesign	Altium	seksjon 3.2
3D-print	SolidWorks og UltiMaker	seksjon 3.5

Tabell 4.1: Oversikt over brukte teknologier med referanser til teoridelen.

- Skjerm (1.8 Inch LCD Display Module SPI Interface TFT Screen Module 128*160 Resolution 16BIT RGB 4 IO ST7735 ST7735S Driver for Arduino)
- Keypad (ECO.12150.06)
- RTC (SD3231)
- DAC (MCP4725)
- Batterivakt (bq27441-G1)
- NTC termistor 2,2 kΩ

4.2 Design av styringsenhet til prototype v.2

Det ble tidlig i prosjektplanleggingen bestemt at gruppen skulle utvide kretskortet til styringsenheten. Idéen var å gjøre prototypen mer intuitiv å bruke ved å legge til en skjerm.

Underveis i designprosessen ble det også bestemt at det de opprinnelige DIP-bryterene skulle byttes med et tastatur. I tilegg ble det ble det bestemt at det skulle inkluderes en WiFi-modul på prototypen. De ulike komponentene som ble valgt var:

- Skjerm: LCD Display Module, 1.8 inch 128x160 TFT LCD Display Module 4-Wire TFT LCD Screen ST7735 TFT Display Module [36]
- Tastatur: ECO.12150.06 [30]
- WiFi-module: ESP8266 ESP-01

4.2.1 Plassering og kobling i Altium

I Tillegg D er alle pinnene som blir benyttet på Arduino Mega 2560 oppsummert. Denne tabellen viser også de ulike pinnene som er brukt for skjerm, tastatur og WiFi-modul. Skjematikken fra Altium med alle tilkoblingspunkter mellom de ulike komponentene er vist i Tillegg E.

Når komponentene skulle plasseres inn i kretskortdesignet for prototype v.1 var det naturlig at de skulle plasseres i tomrommet der DIP-bryterene var plassert. Det er to grunner til at dette området er godt egnet for komponentene som skulle inkluderes. For det første er det kort avstand til Arduino-en sin I/O pinner noe som ville gjøre oppkobling av kretsbanene enklere. Det andre som gjør denne plasseringen godt egnet er at komponentene vil kunne plasseres nært kanten. Når kretskoret ble bestilt var det en ide om at designet på boksen skulle være ganske likt utformet med ett lokk som skulle være festet med hengsler og kunne åpnes. Gitt dette scenariet ville det være en fordel at skjermen og tastaturet står plassert nære hverandre

i lokket. Dette er for at ledningene som kobler dem til kretskortet følger skjøten i lokket og dermed unngår at de beveger seg mye og får slitasje ved åpning av boksen.

Når det gjelder WiFi-modulen var det også ønskelig at denne skulle være plassert ved kanten. Ved å lage ett hull i boksen vil dette grepet tilrettelegge for at antennen på WiFi-modulen får minst mulig forstyrrelse i signaler som sendes fra boksen når den skal brukes.

Hele kretskortet er vist i vedlegg E.

4.2.2 Lodding og feilsøking

Fra start var det et mål om at gruppen skulle følge prosessen for å bygge en fullstendig prototype hele veien. Dette innebar å ha store deler av ansvaret for lodding av komponenter. Etter å ha fått veiledning fra resusser internt på Instituttet og Elektronikk og prototypelabben ble gruppen anbefalt å få gjennomført loddejobben av mer erfarene loddere. Dette var fordi det var svært mange små og varme sensitive komponenter [37].

Når både mønsterkort og komponenter var ankommet ble det gjennomført en grundig jobb av lokalisering og merking av alle komponenter. Det ble etterhvert oppdaget at komponentmerkingene i skjematikken og på mønsterkortet ikke samsvarte. Det gjorde at alle komponentene måtte dobbeltsjekkes at de var riktig type og størrelse. Hele oversikten som ble satt opp for komponentene som skulle loddes er vist i Tillegg G.

Når alle forberedelser var gjort ble kortet levert til Elektronikk og Prototype labben på NTNU. De tar for seg slike oppdrag gratis for studenter. De fikk i oppgave å lodde på alle overflatemonterte komponenter på kortet, bilde av ferdog loddet kretskort er vist i Figur 5.2. De antok at kortet skulle være ferdig loddet i løpet av 1-2 uker. Det resterende arbeidet med oppkobling av kretskortet måtte gjøres samtidig som det monteres i etuiet. Desverre kom gruppen aldri så langt da arbeidet med prototype v.2 ble satt på is av årsaker utenfor gruppens kontroll. Dette er beskrevet i underseksjon 6.3.4.

4.2.3 Design og 3D-printing av etui

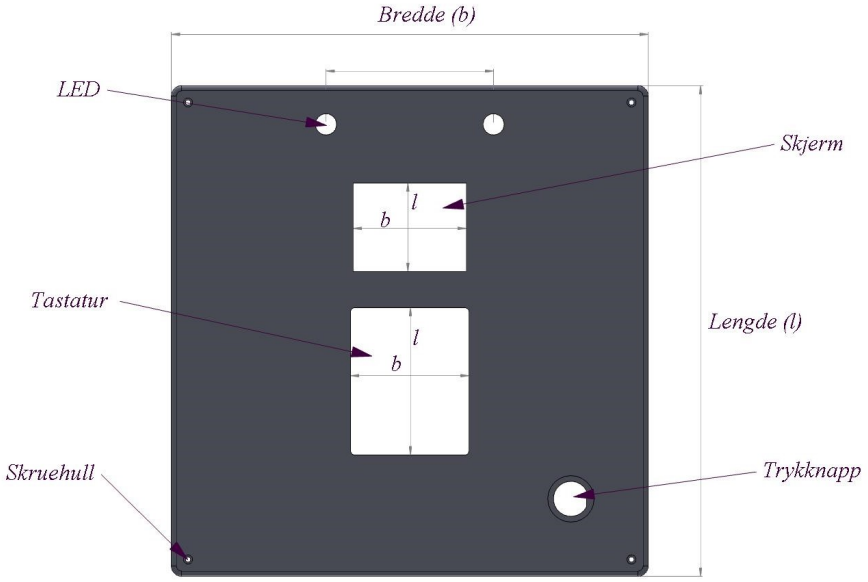
Design

Designet av etuiet ble laget på bakgrunn av designet til prototype v.1. Prototype v.2 ble designet basert på to utlevert SolidWorks delfiler navngitt `LED_Electronics_Top.SLDPRT` og `LED_Electronics_Bottom.SLDPRT`. Etuiet til prototype v.2 ble modifisert til å bestå av tastatur og skjerm, som en erstatning til DIP-bryterne, i tillegg til komponentene som var der fra før. Selve boksen ble utvidet i bredden fra 165 mm til 185 mm og i høyden fra 73 mm til 83 mm, mens lengden forble den samme. Etuiet ble designet til å bestå av tre separate deler: topp, kjerne og bunn. Delene til etuiet for prototype v.2 er avbildet i Figur 5.3 med tilhørende dimensjoner. Oversikt over navn på filene som ble brukt under designprosjektet er vist i Tabell 4.2.

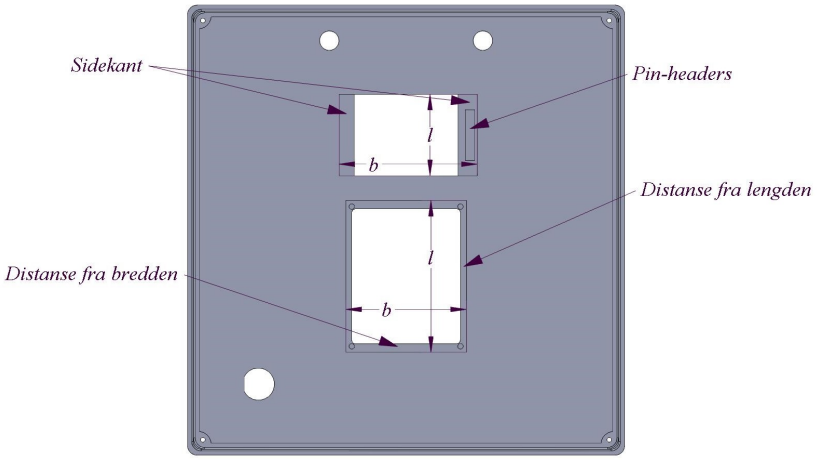
	Filnavn Prototype v.1	**Filnavn Prototype v.2**
Topp	”LED_Electronics_Top.SLDPRT”	”Prototype_v2_lokk.SLDPRT”
Kjerne	”LED_Electronics_Bottom.SLDPRT”	”Prototype_v2_kjerne.SLDPRT”
Bunn		”Prototype_v2_bunn.SLDPRT”

Tabell 4.2: Navn på utgitte- og produserte filer.

Topp



Figur 4.1: Topp sett fra utside markert med dimensjoner.



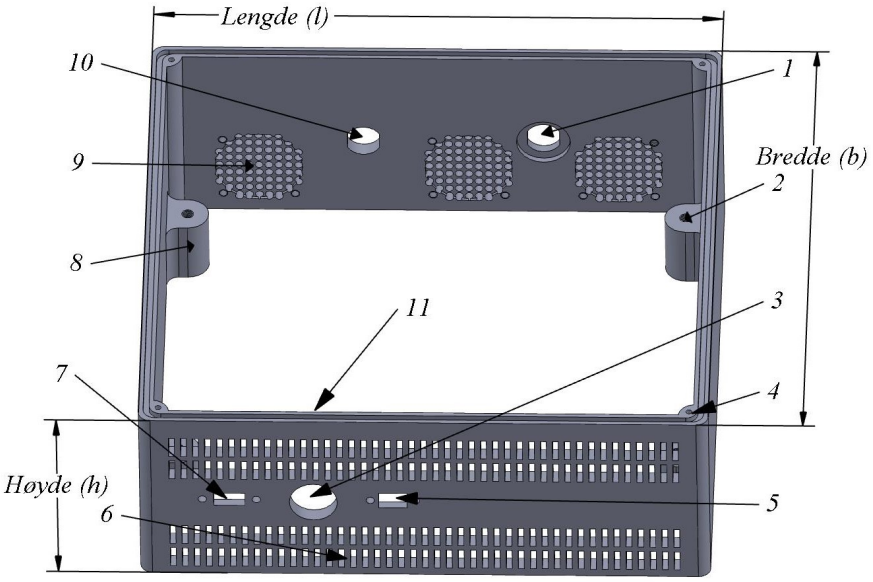
Figur 4.2: Topp sett fra innside markert med dimensjoner.

Nummer	Navn	Utskjæringer og andre forklaringer
1	Skjerm	Utskjæring for skjermen sett fra utsiden
2	Grønn knapp	Utskjæring for trykknapp
3	Skruehull	Utskjæring til skruehull
4	Tastatur	Utskjæring til tastatur
5	LED	Utskjæring til de to LED-ene

Tabell 4.3: Navn på komponenter eller utskjæringer til topplokket.

For design av toppen ble filen `LED_Electronics_Top.SLDPRT` tatt i bruk. I tabellen over i Tabell 4.3 er nummerene fra Figur 4.1 brukt til å beskrive utskjæringene og de tilhørende komponentene. Symmetri ble vektlagt under design av topplokket. Det ble implementert seks nye utskjæringer: skjerm, tastatur og fire skruehull, i tillegg til det originale designet med to LED-lys og en trykknapp på overflaten. Tastaturet sine dimensjoner ble hentet fra databladet under ”Technical drawing” [30, pg.6] og for skjermen ble målene målt manuelt med linjal. Utskjæring sett fra utsiden i Figur 4.1 ble tastaturet satt til en bredde på 46 mm og en lengde på 57 mm. Skjermens bredde og lengde ble satt til hhv. 44 og 34 (oppjustert til 34,30) mm. Skruehullene ble satt til 2 mm i diameter for skruekroppen og 5,5 (ikke fastsatt ennå) mm i diameter for hullet til skruehodet. På topplokket sett fra innsiden i Figur 4.2 ble bredden og lengden til tastaturet satt til hhv. 51- og 64 (oppjustert til 64,10) mm, der distanse fra bredden ble satt til 3,5 mm (64-57/2=3,5) på siden nærmest skjermen, mens 2,6 mm på den nederste siden. Distansen fra lengden ble satt til 2,5 mm på hver side. Dimensjonene for skjermen ble målt manuelt med linjal, og deretter oppjustert etter en test 3D-print, til lengde på 34,3 mm og bredde på 58,5 mm. Sidekantens (uten ”pinheaders”) bredde ble fastsatt til 6,40 mm, og sidekanten med ”pinheaders” ble tilegnet resterende 8,10 mm i bredde og det ble i tilllegg beskjært et 1 mm dypt kvadrat for å gjøre plass til ”pinheaders”-ene. Tastatur og skjerm ble beskjært og deretter plassert langs aksen til boksen. LED-lysene ble forskjøvet ovenfor skjermen, speilet ovenfor aksen. Mens trykkknappen ble plassert nederst i høyre hjørne. Selve lokket sin høyde ble redusert fra 25 mm til 7,5 mm, slik at alle sideflatene ble omgjort til glatte sideflater uten utskjæring. I steden for opplegg til hengsler på den ene kortsiden ble det konstruert hull til fire 2x8TX6 skruer i hvert hjørne av toppflaten på lokket. Bredden til topplokket ble også utvidet fra 165 mm til 185 mm.

Kjerne



Figur 4.3: Kjerne sett fra utsiden nummerert og markert.

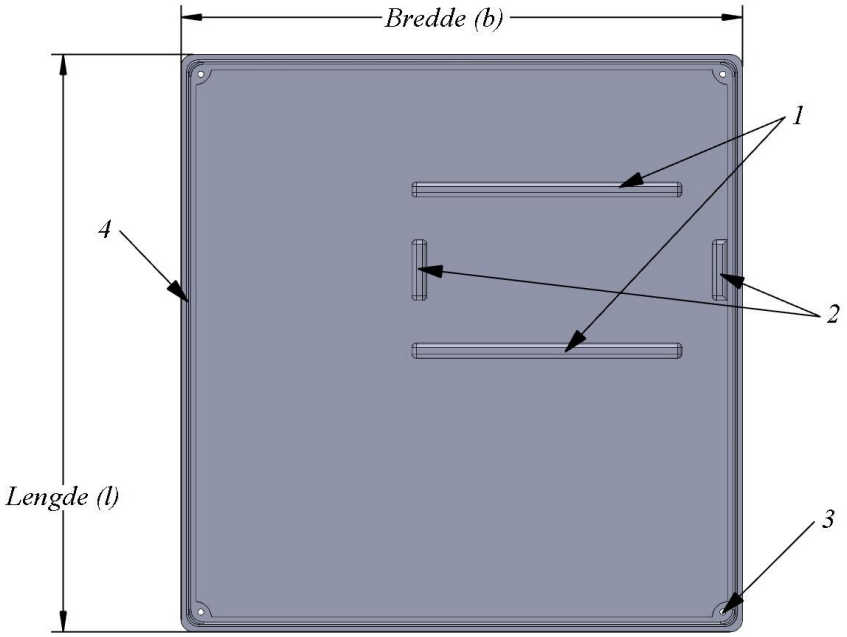
Nummer	Navn	Utskjæringer og andre forklaringer
1	NIR-kabel	Styrer ledmatrise
2	Skruehull	Skruehull for å feste kretskort til feste
3	Av- og påknapp	Til å slå av og på strømmen til styringsenheten
4	Skruehull	Skruehull for å skru fast topp og bunn til kjernen
5	Mini-USB	
6	Ventilasjonsgitter	Rader med utskjæringer
7	USB-C	Til lading
8	Feste	Feste til kretskort
9	Viftehull	Mange runde hull formatert som en sirkel der viftene plasseres
10	Sensor-kabel	
11	Spor	Sørger for riktig plassering når bunn og kjerne føres sammen

Tabell 4.4: Navn på komponenter eller utskjæringer til kjernen.

For design av kjernen ble filen `LED_Electronics_Bottm.SLDPRT` tatt i bruk og modifisert. I tabellen over Tabell 4.4 er nummerene fra Figur 4.3 brukt til å beskrive utskjæringene og de tilhørende komponentene. Bunnen av boksen ble skjært ut, og sporene på den øvre kanten ble speilet til bunnen. Høyden på strukturen ble utvidet til 68 mm med samme høyde og bredde som toppflaten. Også ventilasjonsgitteret (nummer 6) ble speilet om trykknappen. Hjørnene

på innsiden av boksen ble utvidet slik at det ble plass til skruehull til å feste både topp og bunn.

Bunn



Figur 4.4: Bunn sett fra innsiden nummerert og markert.

	Navn	**Forklaring**
1	Batteriholder langside	Hindrer bevegelse på batteriet
2	Batteriholder kortside	Hindrer bevegelse på batteriet
3	Skruehull	Til å skru bunn fast til kjernen
4	Spor	Sørger for riktig plassering når bunn og kjerne føres sammen

Tabell 4.5: Navn på komponenter eller utskjæringer til bunn.

For design av bunn ble filen ”Prototype_v2_lokk” duplisert og tatt i bruk. Alle utskjæringer ble fjernet og målene til batteriholderen fra `LED_Electronics_Bottom.SLDPRT` ble rekonstruert til bunndesignet. Målene til bunnen ble tilsvarende målene til toppen. I Tabell 4.5 er nummerene fra Figur 4.4 brukt til å beskrive utskjæringene og de tilhørende komponentene.

3D-printing

For å få 3D-modellen over til et ferdig produkt ble SLDPRT-filen konvertert til en STL-fil for å forberede den til å lastes opp til UltiMaker Cura som er en ”slicing” programvare. Ved hjelp fra oppdragsgiver ble UltiMaker Cura brukt og 3D-modellen ble posisjonert og klargjort for å skrives ut. ”Infill” ble satt til 20%, ”Profiles” ble satt til 0,1, ”Support” ble satt til normal

og modellen ble deretter lastet opp til en av tre printere på verkstedet: S5_01, S5_02 eller UltiMaker2 Extended +. Alle filene som er beskrevet i avsnittet over ble printet ut etter framgangsmåten beskrevet over og resultatet av printet kan finnes (HER). Mellom hver print ble det gjort forskjellige justeringer som: endring av temperatur på byggeplate, vaske byggeplate med sprit, bruke annen 3D-printer og satt på ekstra brim, i håp om at det skulle bli bedre. Først ble et utsnitt av skjerm og tastatur printet ut. Utsnitt hadde en distanse på ca 10 mm fra skjerm og tastatur og er avbildet i Figur 5.4 , dette tok fire timer å printe. Modifikasjoner på utskjæring av skjerm og tastatur ble utført som er beskrevet i avsnittet over. Etter det ble ”Prototype_v2_lokk” printet som vist i Figur 5.5b, dette tok i overkant av ett døgn. Etterfulgt av ”Prototype_v2_kjerne” som var estimert til å ta ca 2 døgn. Dette printet ble aldri ferdig, og ble avbrutt underveis på grunn av komplikasjoner. På forsøk nummer to av ”Prototype_v2_kjerne” skjedde samme komplikasjon og printeren måtte igjen avbrytes. Siste print var ”Prototype_v2_bunn” som ble estimert til ca et døgn. Dette printet ble også avbrutt underveis vist i Figur 5.6. Det ble totalt gjort seks forskjellige forsøk på å printe de tre forskjellige delene til etuiet uten hell.

4.3 Grunnstruktur for kodebasen

I Tabell 4.6 kan man se en oppsummering av mappene i kodebasen. All koden ligger i en mappe på Github her, men er også lagt til som vedlegg i rapporten. Kodebasen som har blitt utviklet ligger i `src`-mappen. Her kan man finne 13 forskjellige mapper med kode-filer og tilhørende header-filer. Det er i tillegg en `main.cpp`-fil som ligger i `src`-mappen. Den beskriver hvordan alle enkeltfunksjonene kan brukes og settes sammen for å danne systemet etter oppdragsgiver sine ønsker. Kodebasen og alle funksjondefinisjonene er godt dokumentert og kommentert ved å bruke en standard stilart for kodedokumentasjon. Standarden som har blitt brukt kalles Javadoc eller Java-stil dokumentasjon. Det er også inndokumentasjonskommentarer for å beskrive noen spesifikke linjer med kode. Det har også blitt laget en `README.md`-fil for hver mappe. Disse inneholder koden som ble brukt for å teste at funksjonene som har blitt definert fungerer, samt. en beskrivelse av alle funksjonene som er definert i mappen.

Mappenavn	**Beskrivelse**	**Seksjon**
GetError	Definerer en global matrise som skrives til fra andre funksjoner om noe er galt med maskinvaren. Skriver deretter ut de genererte feilmeldingene.	4.4.1
GetTime	Setter opp `Timer1` på mikrokontrolleren for å spore forløpt tid siden mikrokontrollerens oppstart i millisekunder.	4.4.2
Temp	Muliggjør temperaturavlesning ved hjelp av ADC-en og det er definert funksjoner for å beregne temperaturer og LED_ID-motstand basert på ADC-avlesninger.	4.4.3
PWM	Setter opp `Timer3` på mikrokontrolleren for å generere PBM-signaler som styrer viftene på både LED-hodet og styringseneten i tilegg til buzzeren. Funksjonene gir mulighet til å sette tilstandene og ”duty cycle” til disse komponentene basert på gitte parametere.	4.4.4
NIR	Setter opp `Timer2` på mikrokontrolleren for å generere PBM-signaler som styrer NIR-lyset. Funksjonen setNIR() tillater konfigurering av NIR-lysets parametere, og ISR-en (avbuddsvektoren) styrer aktivering og deaktivering av NIR-lyset basert setNIR() sine inngangsvariabler.	4.4.4
TWI	Lager et grensesnitt for kommunikasjon med enhetene koblet til TWI-bussen. Det blir satt opp en rekke funksjoner som automatiserer kommunikasjonen med slavene RTC, DAC og Batterivakten.	4.4.5
Utils	Definerer strukturer, oppregninstyper og hjelpestrukturer	4.5.2
WriteToFile	Sett med forskjellige funksjoner som muliggjør lagring av diverse filer på microSD-kortet ved hjelp av Arduino-biblioteket SD.h	4.5.3
Screen	Initialisering av skjerm og definisjon av de forskjellige skjermbildene som brukes til menyen.	4.5.4
Menu	Menyen setter innstillingene for forsøket og setter sammenhengen mellom skjerm og tastatur.	4.5.4
Settings	Lagrer alle innstillinger forsker har valgt fra menyen og skriver innstillingene til en TXT-fil på microSD-kortet slik at det blir lagret i et ikke-flyktig minne og kan hentes igjen dersom prototypen skrus av.	4.5.4
ledButton-Controlls	Setter opp enkel logikk for styring av LED-ene og avlesning på grønn knapp.	4.5.4
Main-Functions	Setter opp større funksjoner som definerer systemets oppførsel i ulike tilstander. Samler alle deler og setter det i system.	4.6

Tabell 4.6: Oppsummering av bruksområde til alle mappene i kodebasen.

4.4 Koding i C

4.4.1 Håndtering av feilmeldinger

For å samle alle feilmeldingen ble det laget en global matrise som alle kode-mappene kunne skrive til. Den baserer seg på å sette flagg i en matrise dersom en feil oppstår. Matrisen bestod av en rad og fire kolonner. Alle feilmeldinger er indeksert og har et designert plassnummer. Oversikt over alle systemfeil og deres indeks finnes i Tillegg F.

Funksjonen `getError()` kalles på kontinuerlig i `main.c`. Den består av en for-løkke som sjekker om noen av de 26 plassnummerene i matrisen er satt. Om en indeks er satt kjøres en if-setning som fyller den globale matrisen `write_error` med nummeret til den feilmeldingen som var satt.

Kontroll av aktiverte feilmeldinger er i hovedsak tenkt til å kun skje underveis i et forsøk. Det er på dette tidspunktet bestemt at det er mest relevant å overvåke systemets status.

Som beskrevet over kjøres `geterror()` ved hver iterasjon av while-løkken som holder forsøket i gang. Hver gang sjekkes `write_error` for feilmeldinger. Om det detekteres feilmeldinger vil disse skrives til CSV-filen for feilmeldinger, og fjernes fra `write_error`.

Hvis det oppdages en kritisk feil vil skjemen vise ”Kritisk Feil” og bruker blir oppfordret til å skru prototypen av. En kritisk feilmelding refererer til feil som vil hindre gjennomføringen av et forsøk og forhindre at det genereres relevante resultater. Dette vil i hovedsak gjelde feil som medfører at systemet overopphetes og setter pasient i fare, at det ikke kan skrives til fil eller at NIR-lyset ikke kan kontrolleres Figur 4.12.

4.4.2 Oppsett av mindre funksjonaliteter

Avbrudd

For å gjøre koden enklere å strukturere ble det brukt avbruddsprogrammering for å håndtere ulike eksterne og interne hendelser. Dette gjelder søvnmodus og oppsett av sekundteller. I Tabell 4.7 er alle avbruddene som er aktivert samlet.

Søvnmodus

Når prototypen er skrudd på, men ikke brukes aktivt, er det lønnsomt at den minimerer strømbruken. Derfor er det flere forskjellige moduser tilgjengelig for å begrense strømforbruket til Arduino-en mens prototypen er inaktiv. Når prototypen skal gå i søvnmodus, trenger den bare å vente på at en oppvåkning utløses, uten at det er noen annen kode som skal kjøres i bakgrunnen. Dette gir muligheten til å velge den mest aggressive søvnmodusen. Dette er ”Power-DownMode” for ATmega2560 prosessoren. En av de få avbruddene som fortsatt er aktivert i denne modusen er ”Pin Change Interrupt”. Dette avbruddet er det eneste som er nødvendig for dette systemet i sovende tilstand.

For å vekke mikrokontrolleren aktiveres avbrudd på PORTJ PIN1, som tilsvarer pinnen Tx3 på Arduino-en. Valget falt på denne pinnen fordi de resterende pinnene med mulighet for avbrudd var fysisk opptatt. For prototype v.2 vil dette skape en konflikt. Dette er fordi Tx3 skal være

Hva	**Avbruddsvektor**	**ISR oppgave**	**Mappenavn**
Timer1	TIMER1_COMPA_vect	Inkrementerer variabel hvert millisekund	getTime
Timer2	TIMER2_COMPA_vect	Skrur på PWM pinnene koblet til LED-hodet	NIR
Timer2	TIMER2_OVF_vect	Skrur av PWM pinnene koblet til LED-hodet	NIR
Timer3	TIMER3_COMPA_vect	Skrur på PWM8 koblet til buzzer	PWM
Timer3	TIMER3_COMPB_vect	Skrur på PWM10 koblet til vifter på styringsenhet	PWM
Timer3	TIMER3_COMPC_vect	Skrur på PWM11 koblet til vifter på LED-hodet	PWM
Timer3	TIMER3_OVF_vect	Skrur av PWM8,10 og 11	PWM
TWI	TWI_vect	Kjører logikk for TWI kommunikasjon	TWI
Knapp	PCINT1_vect	Vekker systemet fra ”Power Down” / starter forsøk	mainFunctions

Tabell 4.7: Oppsummering av alle avbrudd som benyttes i koden og hvordan de brukes.

koblet opp til WiFi-modulen. WiFi-modulen har blitt skrinlagt, dermed kan pinnen midlertidig kobles til knappen som brukes for å igangsette et forsøk.

Når avbruddet på Tx3 pinnen er aktivert og knappen kobles til, må `ISR(PCINT1_vect)` gjennomføre oppvåkningen.

Oppsett av sekundteller

I mappen getTime i kodebasen kan man finne to funksjoner samt. et avbrudd. Disse funksjonene lager en teller som vil returnere antall millisekunder mikrokontrolleren har vært skrudd på. Koden for dette er basert på en kode fra adnbr.co.uk [38].

Den første funksjonen er `initGetTime()` og setter bit i de tre registrene TCCR1B, OCR1AH og TIMSK1. Dette initialiserer telleren ved å sette ”prescale”, definere verdien som skal trigge en ”overflow”, setter” Time on Compare match (CTC)” modus og skru på avbrudd.

Avbruddhåndteringen er et avbrudd som inkrementerer variabelen `getTime_timer` hver gang den blir kalt. Den blir kalt hvert millisekund.

Den andre funksjonen, `getTime()`, tar i bruk AVR-bibliotek "atomic.h". Atomic-kode brukes for å skru av og på avbrudd for å forsikre at en kodeblokk gjennomføres. Når denne funksjonen kalles i andre funksjoner, vil den gi riktig antall millisekund siden tellingen startet.

4.4.3 Avlesning av analog pinne

Koden som beskrives under er definert i ADC-mappen i kodebasen. Her blir det definert seks forskjellige funksjoner som alle bidrar til avlesning av de analoge pinnene. Deler av programmeringskoden for ADC-en ble utarbeidet fra et eksempel av William Elliot [28]. Koden har blitt omskrevet slik at den avleser fem analoge pinner. En oversikt over de analoge pinnene som skal avlese kan finnes i Tabell 4.8.

	Plassering	**Funksjon**
PIN0	Kretskortet i styringsenheten	Kalt R_ID. Sjekker motstanden på LED-arrayet montert i LED-hodet
PIN1	Kretskortet i styringsenheten	Forsikre seg om at styringsenheten og dens elektronikk har en akseptabel temperatur
PIN2	Kretskortet i LED-hodet	Forsikre seg om at LED-hodet og dens elektronikk har en akseptabel temperatur.
PIN3	Det ene benet til LED-hodet	Forsikre om at hudkontakt mellom pasient og LED-hodet er oppnådd
PIN4	Rett under der NIR-lyset utstråler	Forsikre seg om at NIR-lyset har en akseptabel temperatur

Tabell 4.8: Oversikt over ADC-inngangene og funksjonalitetene deres.

Temperatursensor

Den første funksjonen, `initADC()`, setter ulike bits i de to registerene ADMUX og ADCSRA. Disse registerene setter spenningsreferansen, "prescaler" og slår på ADC-en. Den andre funksjonen, `readADC()`, har en parameter som definerer ønsket avlest analog inngang, en returverdi på 16-bit som returnerer den avleste ADC-verdien og setter ulike bits i ADMUX- og ADSCRA-registerene for å kunne lese av ADC-inngangen. I ADMUX-registeret settes den analoge kanalen som var ønsket. I ADSCRA-registeret blir samtalen startet med ADSC-biten. Resultatet blir lest av med "ADC" kommandoen og brukt som returverdi.

Egenskrevet kode består av fire funksjoner. Den første funksjonen er `initPort()` som setter de fem ønskede analoge pinnene til innganger. Disse pinnene er alle på port F og på pinnene fra 0 til 4.

Den andre funksjonen er `calcADC()`. Funksjonen har den 16-bit avleste ADC-verdien som parameter og en returverdi som er en utregnet temperaturverdi. Funksjonen regner først ADC-verdien om til en ekvivalent spenningsverdi, deretter regnes temperaturen ut. Formelen for utregning av temperaturen er: temperature $= ((V \cdot 100)/V_f) +$ offset [39]. V er spenningen

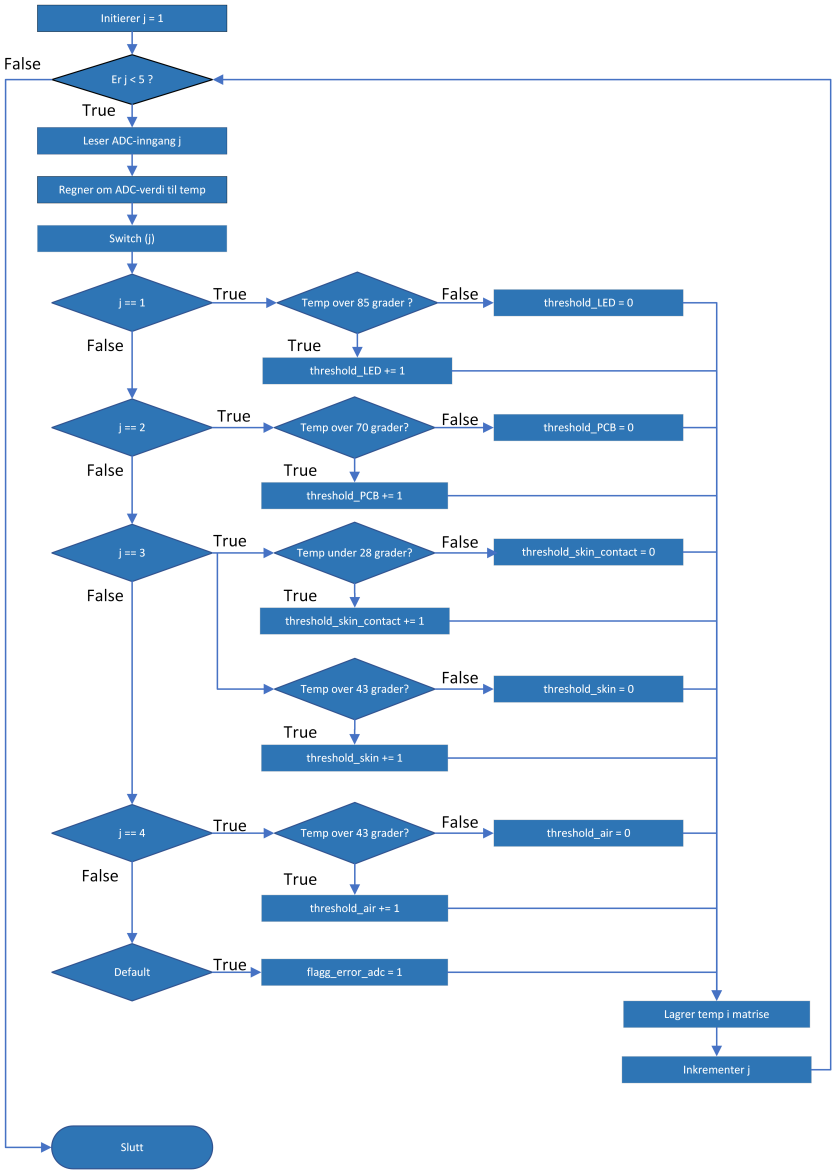
fra ADC-en, V_f er spenningsverdien ved 100 grader celsius minus spenningen ved null grader celsius og offset brukes for å kalibrere temperaturverdien om nødvendig.

Den tredje funksjonen, `setFlaggADC()`, blir kalt på i `main.c` og oppdateres ved hver iterasjon. Denne funksjonen setter ulike varselflagg dersom de fire temperatursensorene registrer temperaturer over de gitte terskelverdiene. I Tabell 4.9 vises de ulike grenseverdiene for temperatursensorene og i flytskjemaet i Figur 4.5 vises kodeflyten til som setter verdier til grenseverdiene.

Pinne	**Temperatursensor**	**Grenseverdi**
PIN1	LED-hodet	I LED-hodet er 85 grader celsius satt som grenseverdi. Dette er basert på LED-matrisen sitt datablad [40] .
PIN2	Styringsenhet	Kretskortet i styringsenheten bør ikke nå temperaturer komponentene ikke tåler. Etter å ha gått gjennom alle komponentenes datablad var lavest. maksimale temperatur 70 grader celsius. Det var batteriet CR-2032/BN som satte denne grensen.
PIN3	Hudkontakt	I masteroppgaven er det konkludert med 42 grader celsius som maksimal temperatur. Denne er til hensyn for diabetikere som kan lide av nevropati og ikke kan reagerer kjapt nok på høye temperaturer. I tilegg er det satt en grenseverdi på 28 grader celsius som skal indikere oppnådd hudkontakt. Er ikke hudkontakt oppnådd skal ikke systemet kunne skru seg på [1].
PIN4	Luft	Her er grenseverdien satt til det samme som ved hudkontakt for og ikke overopphete huden.

Tabell 4.9: Oversikt over temperatursensorene og de ulike grenseverdiene som medfører at et feilmeldingsflagg blir satt.

Funksjonen `getTime()` fra getTime-mappen i kodebasen kjøres hvert femte sekund. Dette realiseres med en if-setning som sammenligner tidspunktet if-setningen sist ble kjørt med det nåværende tidspunktet. Dersom tidsintervallet er større eller lik det som blir definert, vil if-setningen kjøre igjen. I if-setningen er det en for-løkke. Denne for-løkken kjøres fire ganger med variabelen `j` som parameter som teller opp fra en til fire. Den analoge inngangen avleses ved å kalle på funksjonen `readADC()` med `j` som inngangsparameter. Den returnerte verdien regnes om til grader celsius ved å kalle på funksjonen `calcADC()` med den tidligere returnerte ADC-verdien som parameter. I tillegg er det en switch-setning i for-løkken. Når variabel `j` tilsvarer verdien `1` leses den første analoge inngangen av, som er temperatursensoren på kretskortet i styringsenheten. Deretter vil koden videre inn i betingelse `1`. I denne betingelsen vil en variabel for grenseverdien inkrementeres om temperaturen er over 85 grader celsius. Dersom temperaturen er lavere enn grenseverdien vil variabelen tømmes. Dette gjøres for alle de fire inngangene og sjekkes opp mot deres grenseverdier. For-løkken avsluttes ved å lag-



Figur 4.5: Flytskjema som illustrerer gangen i temperaturkoden.

re de fire avleste temperaturene i en global matrise. Til slutt er det fem if-løkker som setter feilmeldingsflagg basert på de avleste temperaturene.

R_ID

Den fjerde og siste funksjonen avleser den analoge inngangen PIN0. Funksjonen, `calcLedID()`, identifiserer hvilket LED-hode som er koblet til styringsenheten. Koden kaller først på funksjonen `readADC()` med null som parameter for å lese av PIN0 og spenningsdeleren. Den avleste ADC-verdien regnes om til en motstandsverdi. Til slutt er det tre if-setninger, som basert på avlest motstandsverdi, setter den globale variabelen `ID_NIR_LED` til en variabelverdi mellom en og tre. Sammenhengen vises i Tabell 4.10.

Bølgelengde	Motstandsverdi	Variabelverdi
660nm	1 kΩ	1
810nm	10 kΩ	2
880nm	51 kΩ	3

Tabell 4.10: Motstandsverdier til LED-matrise ved ulike bølgelengder og ekvivalent variabelverdi.

4.4.4 Pulsbreddemodulasjon (PBM)

PBM kan gjennomføres på alle PBM og I/O pinnene på Arduinoen. Den kontrolleres via en teller som til enhver tid blir sjekket opp mot en ”Compare Match”. ”Compare Match” er et register som aktiverer et avbrudd når den er lik telleren. For hver teller er det to ”Compare Match” registere som kan benyttes. Når tellerene initieres må også tellerens frekvens settes. Dette gjøres ved å stille inn ”precaler” for telleren.

Det er kun `Timer0` og `Timer2` på Arduinoen som har en 8-bits teller og vil fungere for PBM med 255 i oppløsning. Ettersom kontroll av NIR-lyset er den mest kritiske oppgaven som må løses med PBM kontroll blir blir det prioritert å bruke en 8-bits teller for dette. På grunn av problemer med bruk av `Timer0`, ble `Timer2` brukt for kontroll av NIR-lys.

De andre 16-bits tellerene kan også benyttes for PBM, men krever omregning for at `"duty cycle"` skal bli riktig. Derfor blir `Timer3` brukt for å kontrollere buzzeren og viftene. Ligning 4.1 demonstrer hvordan omregning fra 8-bits `"duty cycle"` til 16-bits `dutyCycle` kan gjennomføres.

$$\texttt{"duty cycle"}_{\texttt{16bit}} = 65535 - ((65535/255) \cdot \texttt{"duty cycle"}_{\texttt{8bit}}) \tag{4.1}$$

Kontroll av NIR-lys

LED-matrisene som produserer NIR-lyset kontrolleres via en IC-HG LED-driver, produsert av IC-Haus [41]. Denne LED-driveren kan drive opp til seks kanaler med høy presisjon og strømpulser fra DV nivå helt opp til 200 MHz. IC-HG har tre pinner knyttet til hver av de seks tilgjengelige kanalene.

- ”ENx” tar imot inngangsignalet med PBM puls
- ”CIx” setter nivået på utgangssignalet. Koblet til DAC-en og vil motta 0V / 5V.
- ”LDKx”er utgangen fra LED-driveren. Fungerer som en laser bryter med høy presisjon uten topper.

IC-HG gir muligheten til å benytte totalt seks ”LDK” som kan styre LED-matrisen. Grunnen til at alle seks er koblet opp er fordi noen av LED-matrisene med høyere frekvens på lyset vil kreve større strømforsyning enn kun én LDK pinne kan levere. Som et resultat av dette vil antall PBM pinner og tilsvarende LDK pinner som benyttes være avhengig av hvilket hode som benyttes. Tabell 4.11 viser kort hvilke pinner som styrer hvert av LED-hodene som allerede er definert [1].

Før det kan sendes signal på Arduinoen sine PWM pinner så må først funksjonen `initTimer2()` kjøres. Denne setter opp innstillingene for telleren og aktiverer avbrudd for ”Compare Match A”. For å styre LED-matrisene må det gjøres to ting: DAC-en må settes til å levere 5V inn på

Hvilket LED-hode	Antall pinner	Hvilke pinner
1 - 660 nm	1	PWM: 2
2 - 810 nm	3	PWM: 2, 3, 4
3 - 880 nm	3	PWM: 2, 3, 4

Tabell 4.11: Tabell som oppsummerer dagens LED-hoder og hvilke pinner som skal styre dem.

CI pinnene til IC-HG, og pinnene tilknyttet det gitte hodet må aktiveres med rette innstillinger for PBM. Dette gjennomføres av funksjonen `void startNIR()`. Den aktiverer PBM og lagrer de nødvendige innstillingene i en global matrise kalt `nirData`. Denne variabelen blir brukt når avbruddene knyttet til `Timer2` aktiveres. Inngangsvariablene til funksjonen tilsvarer innstillingene satt i forskermenyen. I koden er høyfrekvent pulsering satt til 75% ”duty cycle”, mens lavfrekvent pulsering er satt til 25% ”duty cycle”. Disse verdiene er lette å modifisere i `startNIR()` funksjonen dersom andre ”duty cycles” skulle vise seg å være mer egnet for forskningen. Når `TIMER0_OVF_vect` trigges har telleren startet på nytt og alle pinnene blir skrudd av. `TIMER0_COMPA_vect` vil trigge de aktuelle pinnene til å skrus på. Dette skjer ved en switch-setning. Denne velger rett antall pinner basert på hvilket LED-hode som er tilkoblet styringsenheten. Funksjonen `void endNIR(void)` vil sette DAC-en sin utgangsspenning til 0V som resulterer i at IC-HG ikke vil gi noe utsignal og LED-hodet vil være avslått.

I kodebasen vil funksjonene som styrer NIR-lyset befinne seg i mappen NIR. Kodeliste 4.1 er det satt opp et enkelt eksempel på hvordan funksjonene brukes. I eksempelet er NIR-lyset skrudd på med innstillingen høyfrekvent pulsering, og skrus av og på med 10 sekunders interval.

```
#include "NIR/NIR.h"

void setup(){
    initTimer2();
}

void loop(){
    // Skrur på LED-hode 2 med PBM instillinger: NIR-belysning med høyfrekvent pulsering
    startNIR(3, 1, 2);
    delay(10000);

    // Skrur av LED-hode
    endNIR();
    delay(10000);
}
```

Kodeliste 4.1: Demonstrerer hvordan funksjonene i NIR.h skal brukes.

Styring av vifter og ”buzzer”

For PBM styring av vifter og ”buzzer” ble `Timer3` benyttet. Denne har tre ”Compare Match” vektorer som gjør at viftene på LED-hodet og styringsenheten kan styres hver for seg. Funksjonen `void setFans()`. Denne funksjonen har mulighet for å skru av/på hver av de ulike viftene, i tilegg til å sette egen ”duty cycle”.

Styring av ”buzzeren” vil gjøres med en tilsvarende funksjon `void setBuzzerAlarm()`. Denne fungerer relativt likt som viftekontrollen. Hovedforskjellen er at ”buzzeren” skal skrus av og på

Hva	Pinne	Formål
Vifte styringsenhet	PWM10	Nedkjøling av elektronikk på styringsenhet
Vifte LED-hode	PWM11	Nedkjøling av LED-hodet
Buzzer	PWM8	Genererer en alarm når det oppstår kritisk feil

Tabell 4.12: Tabell som oppsummerer komponentene som benytter PBM, ekskludert LED-hodet.

som en alarm når den brukes. Derfor har den også parameteren `interval` som definerer hvor mange millisekunder alarmen skal være på av gangen. For å realisere alarmen ble `getTime()` funksjonen benyttet.

Mappen PWM i kodebasen inneholder definisjoner for funksjonene nevnt over. Den inneholder også en `initTimer3()` funksjon som må brukes før vifter og buzzer kan brukes. Denne funksjonen initierer PBM på `timer3` og setter opp alle tre "Compare Match" som er tilgjengelig. I Kodeliste 4.3 er det vist hvordan de ulike funksjonene brukes.

```
#include "PWM/PWM.h"

void setup(){
    initTimer3();
    void setFans(true, false, 200, 0);
}
void loop(){
    // Skrur på alarm med dutyCycle = 170
    // Alarm er på i ettt sekund og så av i ett sekund.
    // Koden må kjøres ved hver iterasjon av løkken for at alarm skal fungere.
    setBuzzerAlarm(true, 170, 1000);
}
```

Kodeliste 4.2: Demonstrerer hvordan funksjonene i NIR.h skal brukes.

4.4.5 Two Wire Interface (TWI)

På kretskortet er det flere komponenter som krever kommunikasjonsprotokoller for overføring av data. Disse er koblet sammen til Arduino Mega 2560 via TWI bussen. I kodebasen kan man finne TWI-mappen og koden beskriver hvordan kommunikasjonsprotokollen gjennomføres. Her er det definert mange forskjellige funksjoner og variabler som muliggjør TWI. Det er fire enheter som er koblet sammen via TWI bussen:

- Arduino Mega 2560
- Real time clock (RTC)
- Digital til Analog konverterer (DAC)
- Batteri Barnevakt

TWI Registerene

ATmega2560 sin TWI protokoll styres av en rekke registre som alle inneholder og oppdateres med informasjon om kommunikasjonen. I denne delen skal de mest relevante registerene beskrives overordnet for å danne et bilde av hvordan kommunikasjonen styres.

TWI Bit Rate Register (TWBR)

TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

Tabell 4.13: Oversikt over bitene i TWI Bit Rate Register [3].

Tabell 4.13 presenterer TWBR-registeret som inneholder en verdi innenfor intervallet 0-255 som spiller inn når klokkefrekvensen på SCL skal bestemmes ved initiering av TWI protokollen. Beregning av verdien som skal skrives til TWBR er beskrevet i Ligning 4.3.

TWI Controll Register (TWCR)

TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	———-	TWIE

Tabell 4.14: Oversikt over bitene i TWI Controll Register [3].

Tabell 4.14 presenterer TWCR-registeret som inneholder bit-ene som styrer dataoverføringene.

- TWINT settes høy hver gang en endring i registeret er satt. Tilbakestilles av maskinvare og signaliserer at forrige operasjon er ferdig.
- TWEA sender ut en ”acknowledge” når det mottas data fra slavene og blir satt høy.
- TWSTA setter start betingelse og aktiverer bussen.
- TWSTO setter stoppbetingelsen og lukker pågående kommunikasjon på bussen.
- TWWC setter flagg når dataskriving skjer samtidig som TWINT er satt høy.
- TWEN må settes høy for å aktivere TWI-protokollen.
- TWIE aktiverer TWI_vect som trigger et avbrudd dersom TWINT er høy.

TWI Status Register (TWSR)

TWS7	TWS6	TWS5	TWS4	TWS3	———-	TWPS1	TWPS0

Tabell 4.15: Oversikt over bitene i TWI Status Register [3].

Tabell 4.13 presenterer TWSR-registeret. Dette registeret inneholder informasjon om hvordan klokken skal stilles inn og status for kommunikasjonen. I TWSR registeret blir to hovedfunksjoner implementert.

- TWPS setter ”prescaler” verdi som benyttes for å beregne klokkefrekvensen på SCL(se Ligning 4.2).
- TWS 3-7 forteller om status for TWI kommunikasjonen. Denne vil gi informasjon om de ulike stadiene i kommunikasjonen, i tillegg til eventuelle feilmeldinger. Det er hovedsakelig to statuskoder som er brukt i koden.
  `0x58` indikerer at det er mottatt data. I koden vil denne fungere som en indikator på at

innholdet i TWDR kan lagres og en stoppbetingelse kan sendes.
`0x38` indikerer at det har skjedd en feil i kommunikasjonen. Om dette skjer sendes en feilmelding, og den pågående kommunikasjonen avsluttes.

TWI Data Register (TWDR)

TWDR7	TWDR6	TWDR5	TWDR4	TWDR3	TWDR2	TWDR1	TWDR0

Tabell 4.16: Oversikt over bitene i TWI Data Register [3].

TWDR vil inneholde neste byte som skal sendes ved Tx-modus, og siste mottatte byte ved Rx-modus.

Beregninger for SCL

Det er viktig at frekvensen på klokkepulsen er innstilt slik at den ikke har for høy hastighet i forhold til slave-enhetene. Dersom hastigheten satt til master er for høy, vil ikke slaven ha tid til å fullføre operasjonen innen SCL settes høy igjen. En konsekvens av dette er at slaven vil tvinge SCL lav til den er ferdig. De høye periodene vil ikke påvirkes av dette, så det vil kun gjøre at hastigheten på dataoverføringene vil gå tregere. Kravet for hastigheten på SCL i forhold til slave-enhetene er at slavens f_{scl} må være 16 ganger så rask som f_{scl}.

Enhet	**Modus**	**Min [kHz]**	**Spesifisert [kHz]**	**Max [kHz]**
RTC	Standard	0		100
	Rask	100		400
DAC	Standard		100	
	Rask		400	
Batteri Barnevakt	Standard		100	
	Rask		400	

Tabell 4.17: Oversikt over SCL hastigheter for hver av slaveenhetene [42][29][43].

Tabell 4.17 viser hastighetene hver av slaveenhetene opererer med for SCL. For oppgavene som skal utføres av prototypen vil standardhastighet være tilstrekkelig. Derfor settes f_{scl} til 100 kHz av master.

Hastigheten på klokkepulsen blir satt av master. Når Arduino Mega 2560 opererer som master er hastigheten bestemt av to bit kalt TWPS i TWSR registeret. De to bit-ene tilsvarer fire forhåndsbestemte verdier for skalering: 1, 4, 16, 64. Registeret TWBR brukes også for å bestemme hastigheten på SCL bussen. Dette registeret er definert for ”R/W”, dermed kan det både leses av og skrives til.

$$f_{scl} = \frac{f_{cpu}}{16 + 2 \cdot TWBR \cdot 4^{TWPS}} \text{Hz [3, kap.24]} \tag{4.2}$$

For å sette frekvensen på SCL (f_{scl}) til 100 kHz brukes verdien på TWBR registeret som en ukjent og verdien fra TWPS bit-ene som konstant som gir Ligning 4.3.

$$TWBR = \frac{\frac{f_{CPU}}{f_{SCL}} - 16}{2 \cdot 4^{TWPS}} \tag{4.3}$$

I Ligning 4.2 og Ligning 4.3 er f_{SCL} frekvensen på klokkepulsen til SCL og f_{CPU} er klokkefrekvensen på mikrokontrollerens CPU. Avlesning av TWPS og utregning av TWBR gjennomføres i funksjonen `init_SCL()`.

Kommunikasjonsprosessen

I sin enkleste form vil det å gjennomføre kommunikasjon via TWI protokollen følge en ganske slavisk oppskrift. En måte dette kan gjennomføres på er nøye beskrevet i databladet til AtMega2560[3, kap.24].

Rx-modus

1. Sett startbetingelse
2. Skriv adressen til slaven som det skal kommuniseres med (R/W bit satt til 0 (W))
3. Skriv registeret som det skal samhandles med
4. Sett ny startbetingelse
5. Skriv adressen til slaven som det skal kommuniseres med (R/W bit satt til 1 (R))
6. Motta data med TWEA satt for å sende ”acknowledge”
7. Ved mottakelse av siste byte settes TWEA til 0 for å signalisere at det ikke skal sendes mer
8. Sett stoppbetingelse

Tx-modus

1. Sett startbetingelse
2. Skriv adressen til slaven som det skal kommuniseres med (R/W bit satt til 0 (W))
3. Skriv registeret som det skal samhandles med
4. Skriv data som skal inn i registeret
5. Sett stoppbetingelse

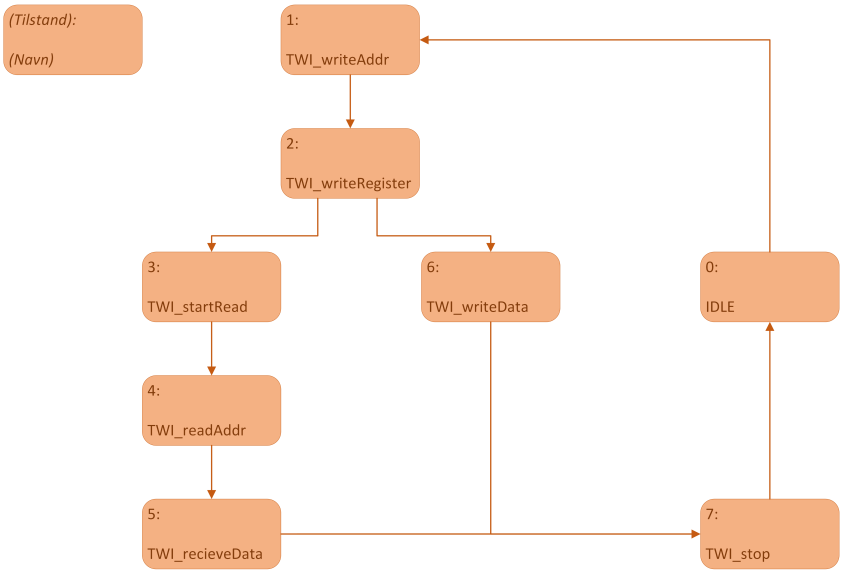
I praksis skal dette gjøre den grunnlegende jobben, men det var et mål om at de ulike stegene skulle trigges av avbruddsvektoren `TWI_vect`. Når dette skulle realiseres i ISR-en ble det satt opp en tilstandsmaskin tilsvarende Figur 4.6 som styrer kommunikasjonsprosessen

Fordi en ISR ikke kan ha inngangsvariabler som en funksjon må variablene som benyttes være globale. Tilstandsmaskinen styres av flere variabler:

- `TWI_case` styrer tilstandsmaskinen med tilstandene beskrevet i Tabell 4.18
- `TWI_state` forteller om det foregår en Rx- eller Tx operasjon. Den sørger for at tilstandsmaskinen gjennomfører rett sekvens.
- `address_byte` lagrer adresse som skal benyttes ved neste data overføring.
- `register_to_read` lagrer det registeret hos slaven som det skal skrives til/leses av.
- `transmit_data` lagrer data-byten som skal sendes ved tx-modus.
- `recieve_data` lagrer data som mottas i rx-modus.

Tilstand	Navn	Oppgave
0	IDLE	Annen kode kjøres frem til TWI trigges
1	TWI_writeAddr	Skriver adresse for slave (tx)
2	TWI_writeRegister	Skriver registeret som det skal kommuniseres med
3	TWI_startRead	Ny start betingelse settes for å starte opp rx
4	TWI_readAddr	Skriver adresse for slave (rx)
5	TWI_recieveData	Mottar data uten ”acknowledge”
6	TWI_writeData	Skriver data til valgt register
7	TWI_stop	Setter stoppbetingelse

Tabell 4.18: Oversikt over tilstander.



Figur 4.6: Flytskjema som illustrerer gangen i TWI kommunikasjonen.

TWI kommunikasjonen må trigges før ISR tar over prosessen. Dette skjer ved å sette en start-betingelse. Dersom denne er satt vil det skje ett avbrudd hver gang TWINT bit-en er satt, frem til dataoverføringen er fullført. For å sette i gang ISR-en er det laget to funksjoner som gjør dette samtidig som variablene for logikken settes riktig:

- `void TWIStartTx(uint8_t address, uint8_t write_reg, uint8_t write_data);`
- `void TWIStartRx(uint8_t address, uint8_t read_register);`

Når logikken for selve dataoverføringen er satt vil disse funksjonene gjøre jobben med å automatisere kommunikasjonen med de ulike slavene lettere og mer oversiktlig.

4.4.6 Feilsøking av TWI kode

Grunnkoden for TWI kommunikasjonen ble relativt raskt satt sammen. Som forventet fungerte ikke koden på første forsøk, problemet var at det var veldig vanskelig å overvåke hva som ble gjort feil og hvorfor den ikke fungerte. Dette førte til at selve feilsøkingen var svært tidkreven-

de. For å undersøke hva slags signal som ble sendt ut av SDA pinnen ble det forsøkt å bruke et oscilloskop. Signalet som ble sendt var 0V til enhver tid som indikerte at det ikke var aktivitet på pinnen. Da ble koden gransket for å se at TWI kommanduene ble utført riktig, det bla da avdekket feil i bit-manipulasjonen av TWI-registerene.

Register	Innhold
0x00	Sekund 0-59
0x01	Minutt 0-59
0x02	Time 0-24
0x04	Dag 1-31
0x05	Måned 1-12
0x06	År 0-99

Tabell 4.19: Oppsummering av registre som blir bruk på DS3231 [42].

Real Time Clock (RTC)

En RTC er en kalibrert klokke som kan gi informasjon om dato og tid. RTC-en som blir benyttet i designet er en DS3231. Den vil operere som slave og vil svare til adressen 110100X. De ulike registerene til RTC-en som blir avles fra er oppsummert i Tabell 4.19.

For å benytte RTC-en og hente ut pålitelige verdier for dato og tid er det to aspekter som må tas hensyn til. Når RTC-en brukes etter å ha vært frakoblet strøm vil ikke de avleste verdiene være riktig. For å løse dette må rett verdi skrives til riktig register. Funksjonen `void resetRTC()` bruker funksjonen `TWIStartTx()` for å gjøre dette.

`resetRTC()` har som input nåverdier for tid og dato som heltall. Kallet `resetRTC(0,15,12,24,5,23)` vil eksempelvis oppdatere RTC-en til *"24. mai 2023 kl. 12:15:00"*. Fra dette tidspunktet er satt vil tiden telles med høy presisjon.

Når dato og tid er oppdatert må man unngå at RTC-en mister strømtilførselen. Om det skulle skje vil verdiene i registerene nullstilles og det vil ikke være mulig å bruke den for å sjekke dato og tid. For å unngå at RTC-en mister strøm hver gang prototypen skrus av benyttes et CR2032 batteri koblet til V_{BAT} pinnen på RTC-en. Denne pinnen er koblet til et reservebatteri mens V_{CC} er koblet til hovedbatteriet som beskrevet i Tabell 4.20. CR2032 er et 3V knappebatteri med en levetid på 460 timer ved kontinuerlig bruk. Så lenge batteriet forsyner RTC-en vil klokken fortsette og telle når prototypen er skrudd av. Men med 460 timer levetid vil den kun forsyne RTC-en i ca. 19 dager [44].

Når RTC-en er riktig innstilt kan de ønskede dataene hentes ut ved å bruke funksjonen `TWIStartRx()`. Målet er å samle innhenting og prosessering av data i ulike funksjoner som kan kalles på der de trengs.

- `getTimeStamp()`: Returnerer klokkeslettet i en matrise [ss,mm,tt].
- `getDate()`: Returnerer dato i et matrise [dd,mm,åå].

Når dato og måned skal brukes i koden er det hensiktsmessig at de er lagret som en `char` slik at det kan skrives direkte til CSV-fil eller skjerm. Funksjonene som utfører denne konverteringen er: '

- `void formatDateToChar(uint8_t dateArray[3])` tar inn matrisen levert fra `getDate()` og gir ut dato som en char-matrise på formatet "dd-mm-åå".
- `void formatTimeToChar(uint8_t timeArray[3])` tar inn matrisen levert fra `getTimeStamp()`

Pinne (RTC)	Koblet til	Funksjon
2: VCC	3.3 Volt	RTC-ens hoved spenningsfirskyning
4: RST	Arduino: Pinne 13	Aktiv lav reset
14: VBAT	CR2032 knappebatteri	Reserveforskyning av spenning til RTC.
15: SDA	Arduino: Pinne 20	TWI kommunikasjon
16: SCL	Arduino: Pinne 21	TWI kommunikasjon

Tabell 4.20: Tabell som oppsummerer alle brukte pinner på RTC-en og deres funksjoner.

og gir ut tid som en char-matrise på formatet ”tt:mm:ss”.

- `void formatDateTimeToChar(uint8_t dateArray[3], uint8_t timeArray[3])` kombinerer dato-matrise og tids-matrise for å sette sammen et char-array med format ”dd-mm-åå tt:mm:ss”

Hvordan de ulike bibliotekene brukes vil avhenge av hvilket format dato og tid må være på for videre bruk. I Kodeliste 4.3 er eksempelkoden til dette vist.

```
#include "TWI/TWI.h"

void setup(){
    Serial.begin(9600);
    initTWI();
}
void loop(){
    // getDate og getTimeStamp må oppdateres før hver gang det skal hentes data fra RTC.
    getDate();
    getTimestamp();

    // Eksempel på hvordan tallverdi for dato kan presenteres
    Serial.print("Dato:");
    Serial.println(calcDate[0]);
    Serial.print("Måned:");
    Serial.println(calcDate[1]);
    Serial.print("År:");
    Serial.println(calcDate[2]);

    // Skriver dato og tid med formateirng
    Serial.println(formatDateTimeToChar(calcDate, calcTime));
}
```

Kodeliste 4.3: Demonstrerer hvordan funksjonene i NIR.h skal brukes.

Digital to Analog Converter (DAC)

Formålet med DAC-en er at den skal fungere tilsvarende de digitale I/O pinnene på Arduino-en, uten å være tilkoblet Arduino-en sin strømforsyning. Dette er for å beskytte Arduino-en fra høyt strømtrekk fra LED-driveren ”IC-HG” som kan trekke opp til 60 mA. For å unngå dette blir DAC-en forsynt av 5V spenningsnettverket fra batteriet mens den skal levere den samme spenningen som en digitalpinne [41].

MCP4725 som benyttes i denne kretsen har en 12-bits oppløsning. Denne oppløsningen har ingen betydning slik DAC-en skal brukes i dette tilfellet. Fremfor å utnytte DAC-ens fulle po-

Pinne (DAC)	**Koblet til**	**Funksjon**
Vout	IC-HG LED-driver sine CI pinner.	Setter nivået på utgangssignalet til IC-HG
Vss	Jord	Setter nedre grense for DAC-en sin utgangsspenning
Vdd	5V	Setter øvre grense for DAC-en sin utgangsspenning
A0	Jord	Del av TWI adresse - satt til ”0”
SCL	Arduino: Pinne 21	TWI Kommunikasjon
SDA	Arduino: Pinne 20	TWI Kommunikasjon

Tabell 4.21: Tabell som oppsummerer pinnene til DAC-en og deres funksjon.

tensial med mange ulike nivåer, skal den heller brukes som en av/på bryter for spenningen inn på LED-driveren. Dette gjør at det kun er to verdier som skrives inn til DAC registeret, enten 0 eller 4095. For å realisere dette blir TWI kommunikasjonen brukt [29].

Funksjonen `void setDAC(bool on_off)` styrer DAC-en. Denne fungerer ved at inngangsverdien velger om den skal skrus av eller på. Deretter sender Arduino-en rett beskjed til DAC-en. Denne funksjonen brukes samtidig som NIR-lyset kontrolleres gjennom PWM-pinnene. Tabell 4.21 beskriver alle pinne til DAC-en og hva de er koblet til. En detalj og legge merke til er at som nevnt tidligere så forsyner DAC-en alle CI pinnene til LED-driveren.

Batterivakt

Batterivakten skal overvåke gjenstående batterilevetid i prosent og batterikapasiteten sammenlignet med kapasiteten i nyfabrikert tilstand. Gjennom TWI kommunikasjon kan det hentes ut flere ulike indikasjoner på batteriets tilstand. Batteriprosenten skal indikere hvor stor prosent av batteriets maksimale kapasitet som gjenstår. Det er også interessant å overvåke batteriets tilstand for å se når det eventuelt må byttes ut. Dette er spesielt viktig med tanke på at prototypen ideelt skal ha nok batterikapasitet til å kunne kjøre tre forsøk på et fulladet batteri som vist i Tillegg C. Batteristatus skal hentes ved prototypens oppstart og oppdateres jevnlig mens prototypen er på. Funksjonen `uint8_t *getBatteryState()` henter ut disse dataene og returnerer en matrise med tre elementer. Det første elementet er batteriprosenten [0 - 100], det andre elementet er verdien for batteriikonet [0 - 5], og tredje elementet forteller om batteriets helse i % [0-100] [43]. Batteriikonet skal brukes for å vises på skjermen.

4.5 Koding i C++

.

4.5.1 Oversikt over brukte biblioteker

Det har blitt tatt i bruk biblioteker for programmering av skjerm, tastatur og minneutvidelsen. Oversikt over bibliotekene som har blitt brukt kan finnes i Tabell 4.22. I PlatformIO blir den nødvendige informasjonen om hvilke biblioteker som blir brukt lagret i `platformio.ini` i rotmappen til kodebasen.

Biblioteknavn	**Utvikler**	**Versjon**	**Bruksområde**
Keypad	chris–a	3.1.1	Tastatur
Adafruit ST7735 and ST7789 Library	Adafruit	1.10	Skjerm
Adafruit GFX Library	Adafruit	1.11.5	Skjerm
SD	Arduino	1.2.4	Minneutvidelsen

Tabell 4.22: Oversikt over bibliotek som brukes i C++ koden.

4.5.2 Hjelpefunksjoner, oppregninstyper og strukturer

I utils-mappen i kodebasen er det definert flere hjelpefunksjoner og strukturer. Disse brukes hovedsakelig i writeToFile-mappen, settings-mappen og menu-mappen. Det er definert en struktur, tre oppregningstyper og seks funksjoner. Formålet med å definere strukturen er å gi en konsistent og helhetlig måte å representere de ulike innstillingene for forsøket på. Strukturene representerer informasjon om de ulike innstillingene for ett forsøk. Funksjonene er hjelpefunksjoner som brukes i sammenheng med strukturene. Dette er mulig siden C++ er et høynivåspråk som tillater mer komplekse datastrukturer som også er mer oversiktlige.

Oppregningstypene er definert i `utils.h` og er definert som typen `enum`. De brukes for å beskrive de forskjellige filtypene og innstillingene forskeren kan velge.

1. `FileType` representerer de forskjellige filtypene. Den har tre mulige verdier: `INFO`, `TEMP` og `ERROR`. De forskjellige filtypene og deres bruksområde er beskrevet nærmere i underseksjon 4.5.3.
2. `Mode` representerer de ulike testmodusene forsker kan velge. Det er fire mulige verdier: `NIR_LIGHT`, `PLACEBO`, `RANDOMIZED_MODE` og `UNKNOWN_MODE`.
3. `Duration` representerer varigheten av et forsøk og har fire mulige verdier: `DURATION_20_MIN`, `DURATION_30_MIN`, `DURATION_40_MIN` og `UNKNOWN_DURATION`.
4. `PwmFreq` representerer PBM-frekvensene og kan ha fire verdier `CONTINUOUS`, `LOW_FREQ`, `HIGH_FREQ` og `UNKNOWN_PWM_FREQ`.

Oppregningstypene `Mode`, `Duration` og `PwmFreq` er satt opp i samsvar med Tabell 2.1, med unntak av de siste alternativene som setter verdiene som ukjente. Disse brukes for å håndtere feilverdier, ettersom det aldri er ønskelig at et forsøk skal ha ukjente verdier.

Strukturen `TestChoices` setter sammen all informasjonen om testinnstillingene og lagrer den. Strukturen har følgende felt: `mode`, `duration`, `pwm_freq`, `patient_id` og `experiment_id`. De tre førstnevnte feltene er av typene `Mode`, og `Duration`, `PwmFreq` (oppregningstypene nevnt over) og de to siste er `uint8_t`.

De seks funksjonene kan deles inn i to kategorier. Den første kategorien har en parameter som tar inn en av oppregningstypene. Den vil returnere en streng som representerer den tilsvarende

verdien. De brukes enten for å skrive innstillingene til en CSV-fil eller for å skrive dem til skjermen.

Den andre kategorien tar imot et heltallsverdi (`int`) og konverterer den til den tilsvarende oppregningstypen, enten `Mode`, `Duration` eller `PwmFreq`. Disse funksjonene brukes i settingsmappen for å lese av de lagrede innstillingene. En oversikt over sammenhengen mellom heltallet og innstillingsverdien kan ses i Tabell 4.23.

Enum	**Int**	**Verdi**	**Streng-verdi**
Mode	1	NIR_LIGHT	NIR-lys
	2	PLACEBO	Placebo
	3	RANDOMIZED	Randomisert
Duration	1	DURATION_20	20 min
	2	DURATION_30	30 min
	3	DURATION_40	40 min
PwmFreq	1	CONTINOUS	Kontinuerlig
	2	LOW_FREQ	Lav frekvent
	3	HIGH_FREQ	Høy frekvent

Tabell 4.23: Oversikt over oppregningstypene, deres tilsvarende heltallsverdi og streng-verdi.

4.5.3 Loggføring av sensorverdier og feilmeldinger

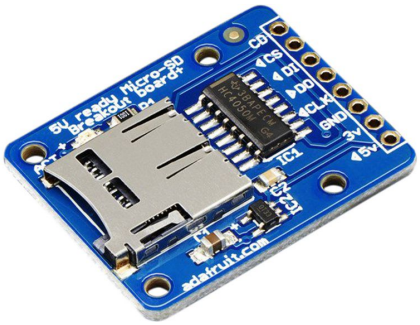
Det er ønskelig at prototypen skal loggføre verdiene på temperatursensorene hvert minutt og knytte dette til et pasient identifikasjonsnummer. I dette tilfellet er det fire forskjellige temperaturverdier som skal loggføres, en oversikt over disse kan sees i Tabell 4.8, samt. det tilhørende tidsstempelet. Det skal også loggføres feilmeldinger som oppstår under forsøket. Feilmeldinger skal loggføres med tidsstempel og kan f.eks. være at pasienten mister hudkontakt med LED-hodet. Feilmeldingene og temperaturverdiene skal lagres i to separate ”comma separated values” (CSV) filer. I tillegg blir innstillingene for forsøket lagret i en tekstfil (Tekstfil-fil). En oversikt over innstillingene som gjelder for et forsøk kan finnes i Tabell 2.1.

MicroSD grensesnittkort fra Adafruit

Prototypen har en minneutvidelse som kan brukes til å lagre og skrive til filene knyttet til et forsøk. Denne minneutvidelsen er av typen ”Micro SD card breakout board+” fra Adafruit som er avbildet i Figur 4.7. Dette grensesnittskortet har plass til et microSD-kort og den har åtte pinner [45] som er beskrevet nærmere i Tabell 4.24. Den pinnen som hovedsakelig brukes for kodingen av loggføringen er pinne `CS` (Chip select), dette er pinnenummer 53 på Arduino-en.

Fillagringssystemet

For å opprette en struktur for lagring av filer for hvert forsøk, ble det tatt utgangspunkt i et system som skal utdypes her. For hvert forsøk opprettes det en mappe, og mappenavnet bestemmes av identifikasjonsnummeret for forsøket. Nummeret genereres kun for å skape en systematisk struktur for mappene. Det starter på null og vil inkrementeres for hvert gjennomførte forsøk. Dette betyr at når NIR-stråling er initialisert, opprettes det en ny mappe på microSD-kortet. Inne i mappen skal det lagres to CSV-filer og en TXT-fil. Filnavnene baserer seg på



Figur 4.7: "microSD card breakout board+" fra Adafruit [46].

identifikasjonsnummeret til pasienten og hvilken type informasjon som lagres i filen. For eksempel, dersom pasientens identifikasjonsnummer er `69` og forsøkets identifikasjonsnummer er `3`, vil det opprettes tre filer med navnene `LOG_69.csv`, `ERR_69.csv` og `INFO_69.csv`. Hele banenavnet til filene vil være `3/LOG_69.csv`, `3/ERR_69.csv` og `3/INFO_69.txt`.

Videre kan man se nærmere på innholdet i disse tre filene. Det er to CSV-filer som brukes til loggføring av feilmeldinger og temperatursensorverdier. Filene vil ha to forskjellige overskrifter ettersom de representerer ulik informasjon for hvert forsøk. I filen `LOG_69.csv` vil den første raden inneholde overskrifter. Den første kolonnen inneholder tidsstempel, etterfulgt av kolonner for de fire temperaturverdiene. Sammenhengen mellom overskriften i CSV-filen og temperatursensoren sin plassering på prototypen er beskrevet nærmere i Tabell 4.25. `ERR_69.csv` vil ha overskriften tidsstempel, feilmeldingkode og den tilhørende feilmeldingen. Tidsstemplene som brukes i loggfilene er på formatet ISO 8606 som inneholder informasjon om klokkeslett, dato og tidssone. Filen `INFO_69.txt` vil innhold innstillingene for forsøket: PBM-frekvensen, pasient- og forsøksidentifikasjonsnummer, varighet og modus for forsøket. Disse innstillingene er beskrevet nærmere i Tabell 2.1. TXT-filen inneholder også informasjon om starttiden. Dette er det tidspunktet forsker eller pasienten igangsetter NIR-belysning.

Arduino biblioteket SD.h

I denne delen av koden har det blitt brukt et bibliotek fra Arduino kalt `SD.h` [47]. Dette biblioteket inneholder funksjoner som gjør det enklere å lese og skrive data til microSD-kortet som er plassert i grensesnittskortet fra Adafruit som er avbildet i Figur 4.7. Ved å inkludere `SD.h` i header-filen kan man initialisere SD-kortet ved å kalle `SD.begin(53)`. Videre har følgende funksjoner blitt brukt:`SD.exists()`, `SD.mkdir()` og `SD.open()`.

- `SD.exists()` sjekker om en fil med det angitte filnavnet eksisterer på SD-kortet. Den tar filnavnet som parameter og returnerer `true` hvis filen eksisterer, eller `false` hvis den ikke eksisterer. Denne brukes til å sjekke om filnavnet allerede eksisterer, før et forsøk igangsettes og oppretter filene.
- `SD.mkdir()` brukes til å opprette en ny mappe på SD-kortet med det angitte katalognavnet. Denne funksjonen tar imot mappenavnet som parameter og returnerer `true` dersom opprettelse av mappen var vellykket, eller `false` hvis det oppstod en feil. Denne funksjonen er nyttig for å organisere filer i ulike mapper og vil skape en bedre struktur. Den brukes til å opprette mappen med forsøksidentifikasjonsnummeret, som vil inneholde

Pinnenummer på Arduino Mega 2560	**Akronym**	**Navn**	**Funksjon**
I/O 48	CD	Card Detecct	Detektere tilstedeværelse eller fravær av microSD kortet
SS 53	CS	Chip Select	Aktivere og deaktivere kommunikasjon med microSD-kortet
MOSI 51	DI	Data In	Dataoverføring fra mikro-kontrolleren til microSD kortet
I/O 49	DO	Data Out	Dataoverføring fra microSD-kortet til mikro-kontrolleren
SCK 52	CLK	Clock	Denne pinnen gir klokke-signalet for å synkronisere dataoverføringen mellom mikontrolleren og microSD kortet
-	GND	Ground	Kobles til jord for prototypen
-	3V	Strømforsyning	Brukes ikke
-	5V	Strømforsyning	Brukes for å gi strøm til kortet

Tabell 4.24: Oversikt over pinnene til grensesnittkortet fra Adafruit [45].

filene relatert til forsøket.

- `SD.open()` brukes til å åpne en fil på microSD-kortet med det angitte filnavnet og åpningsmodusen. Den tar imot to paramtere: filnavnet og åpningsmodusen, og returnerer et `File`-objekt som representerer den åpnede filen. Åpningsmodusen kan være `FILE_READ` for lesing eller `FILE_WRITE` for skriving. Dette muliggjør enten lesing av data fra filen eller skriving av data til fil. Det returnerte`File`-objektet brukes til å åpne logg- og feilmeldingsfilen for skriving.

`File`-objektet som returneres fra `SD.open()` har flere metoder, deriblant `println()`, `write()` og `close()` som har blitt benyttet i denne delen av koden.

- `println()` metoden brukes til å skrive data til den åpne filen og legger til ett linjeskift etter dataene har blitt lagt til. Denne brukes fordi en rad i CSV-filen tilsvarer en lesing av temperaturverdier eller feilmeldinger og vil legge til et linjeskrift automatisk. `write()` gjør essensielt det samme, bare uten å legge til linjeskift.
- `close()` metoden brukes for å lukke den åpne filen når man er ferdig med å lese eller skrive data til den. Det er viktig å lukke filen for å frigjøre ressurser og sikre at eventuelle skrivebuffere blir tømt og at dataene blir lagret på microSD-kortet ved en temperaturav-

	Plassering	**Navn i CSV-fil**
1	Kretskortet i styringsenheten	`temp_pcb`
2	Kretskortet i LED-hodet	`temp_led`
3	Det ene benet til LED-hodet	`temp_skin`
4	Rett under der NIR-lyset utstråler	`temp_air`

Tabell 4.25: Oversikt over navngivingen til de forskjellige temperatursensorene sine verdier loggfilen.

lesning. Det var et krav fra oppdragsgiver at CSV-filene skulle lagres underveis i forsøket. Dette for å hindre at man mister temperatur- eller feilmeldingslogger ved eventuelle systemfeil ved prototypen som kan oppstå underveis i et forsøk.

Skrive data til fil

Koden til prosjektet ligger under mappen writeToFile i kodebasen. I Figur 4.8 kan man se et flytdiagram som beskriver koden og sammenhengen mellom alle funksjonene. Koden er inndelt i tre filer: `helper.cpp`, `writeToFile.cpp` og `writeToFile.h`.

I `helper.cpp` blir det definert flere hjelpefunksjoner som brukes for å håndtere filbehandling, konvertere data til riktig format og gjør om forskjellige oppregningstyper til strenger slik at de også kan skrives til filene.

- `getExperimentId()` henter forsøk identfikasjonsnummeret fra SD-kortet basert på eksisterende mapper. Den returnerer forsøk identfikasjonsnummeret som skal brukes for det gjeldende forsøket.
- `checkIfFileExists(filename)` sjekker om en fil eksisterer på SD-kortet. Den tar filnavnet som input og returnerer 1 hvis filen eksisterer, ellers returnerer den 0.
- `convertErrorToChar(error_code, error_message, timestamp)` konverterer feildata til en char-matrise og returnerer det. Den bruker feilkoden, feilmeldingen og tidsstempelet som parametre.
- `convertDataToChar(temp_pcb, temp_air, temp_skin, temp_led, timestamp)` konverterer temperaturdata til en char-matrise som returneres. Den tar inn temperaturen på PCB, i luften mellom NIR-lyset og huden, mot huden og på LED-matrisen i heltallsformat, samt. tidsstempel som en `char`.
- `modeToString(mode)`, `durationToString(duration)` og `pwmFreqToString(pwmFreq)` konverterer oppregningstypene `mode`, `pwmFreq` og `duration` variablene til en streng slik at det kan skrives til filen.

I `writeToFile.cpp` defineres funksjoner som brukes til å håndtere microSD-kortet, oppretter mapper og filer, skriver data til filene og lage filen som skal inneholde informasjon om forsøkene som gjennomføres.

- `initSD(CS)` initierer microSD-kortet. Den tar inn chip select-pin (CS) som parameter.
- `initCard(CS, card)` sjekker om initieringen av microSD-kortet var vellykket. Hvis initieringen mislykkes, blir det skrevet ut feilmeldinger og informasjon om hva som kan

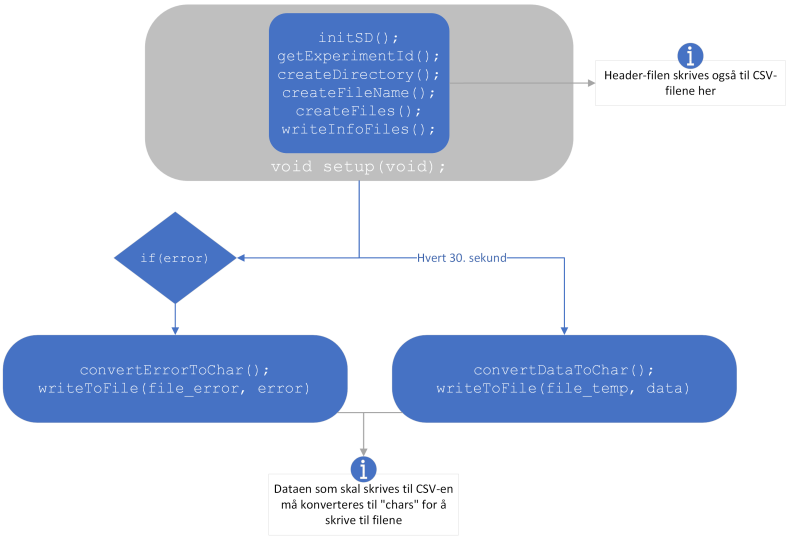
være galt. Den tar inn "Chip Select"-pin (CS) og et `Sd2Card`-objekt (card) som parameter.
- `createDirectory()` oppretter en mappe på microSD-kortet basert på forsøk identifikasjonsnummeret. Den tar inn forsøk identifikasjonsnummeret som parameter.
- `createFile()` oppretter en fil med et gitt filnavn og skriver overskriftene (headers) til både CSV-filene og TXT-filen. Den tar inn overskriftene, filnavnet og en `TestChoices`-struktur som parameter.
- `createFileName()` oppretter et filnavn basert på filtypen (type) og testvalgene (`test_choices`). Den returnerer filnavnet som en char-matrise.
- `writeToFile()` skriver data til en fil med et gitt filnavn. Den tar inn filnavnet og dataene som skal skrives som parameter. For å skrive korrekt datatyper skal skrives til filene må hjelpefunksjonene `convertErrorToChar()` og `convertDataToChar()` brukes.
- `writeInfoFile()` skriver informasjon om forsøket til en fil. Den tar inn testvalgene, starttidspunktet, filnavnet og skriver informasjonen til TXT-filen.

Testkode

For å teste at koden fungerer som ønsket har det blitt laget en testkode som testes på et isolert system med en Arduino Mega 2560, grensesnittskortet fra Figur 4.7, koblingsbrett og noen ledninger. Det ble lastet opp en kode fra `main.cpp` til Arduino-en som kan finnes i `README.md` i `writeToFile`-mappen i kodebasen, koden ligger helt nederst i dokumentet.

Koden besto av en `setup()`-funksjon med en `while()`-løkke. Nødvendige bibliotek som f.eks. `writeToFile.h`, `stdio.h`, `Arduino.h` og `utils.h` ble inkludert. Det ble oprettet flere globale variabler som `temp_headers`, `error_headers`, `info_header`. Disse variablene er `char`-matrise som består av headere til de tre filene som skal skrives til. `card` og `mem_ext_pins` ble også deklarert som globale variabler for å konfigurere grensesnittskortet med Arduino-en. I `setup()`-funksjonen ble seriell kommunikasjon og microSD-kortet initiert. En struktur ble opprettet for å inneholde lagrede innstillinger som PBM-frekvens, pasientidentifikasjonsnummer og forsøksidentifikasjonsnummer. Denne strukturen er nærmere forklart i underseksjon 4.5.2. For å simulere et forsøk ble innstillingene satt uten kommunikasjon med skjerm eller tastatur. Deretter ble de tre filnavnene opprettet ved hjelp av funksjonen `createFilename()`. Videre ble den nødvendige mappen opprettet, som også vil inneholde filene for forsøket. Dette ble gjort ved hjelp av funksjonene `createFile()` og `createDirectory()`. Informasjon om de valgte innstillingene ble også skrevet til tekstfilen ved å bruke `writeInfoFile()`.

Deretter vil koden gå videre inn i den evige `while`-løkken. I denne løkken blir all dataen skrevet til CSV-filene for å loggføre sensorverdier og potensielle feilmeldinger. Feilmeldingene og sensorverdien har blitt simulert slik at man får testet virkemåten til funksjonene. Det har blitt tatt hensyn til at temperatursensorverdiene vil være av datatypen `float`. Raden som skrives til CSV-filen må være på samme format som overskriften (headerene) i CSV-filen. For at dette skal la seg gjøre må `convertDataToChar()`-funksjonen brukes. Denne konverterer de avleste sensorverdiene til en char-matrise `data` som videre sendes inn i `writeToFile()`-funksjonen. Den samme metoden er brukt for feilmeldinger. `convertErrorToChar()` som returner en char-matrise `error` som igjen bruker `writeToFile()`-funksjonen. For å skille mellom hvilken fil som skal skrives til, brukes parameteren `filename` som enten tar inn variabelen `filename_error` eller `filename_temp`.



Figur 4.8: Flytskjema for funksjonene som muliggjør loggføring av sensorverdier og feilmeldinger.

4.5.4 Programmering av brukergrensesnitt

LEDs og knapper

Knappene og LED-indikatorene er de delene av brukergrensesnittet som har opphav fra prototype v.1. Knappene vil ha mer eller mindre samme funksjonalitet i dette systemet som de hadde i versjon en. AV og på knappen på etuiets side skal skru prototypen av og på, mens den grønne knappen skal brukes for å starte forsøk. I prototype v.2 er det også lagt til at den grønne knappen skal kunne vekke systemet fra søvnmodus. Dette skjer via avbruddsvektoren PCINT1_vect som trigges når det skjer en endring på pinnen Tx3.

LED-indikatorene benyttes for å kommunisere systemets status underveis i forsøk. Den grønne LED-en skal lyse kontinuerlig når et forsøk kjøres, og når det er under 3 minutter igjen så vil den begynne å blinke. Den røde indikatoren brukes for å indikere feil på systemet. Den vil begynne å blinke om det oppdages mindre feil i systemet, og vil lyse kontinuerlig ved kritiske feil.

For å gjøre programmering av LED-ene mer oversiktlig i koden er det blir laget en rekke funksjoner som benyttes:

- `initLedPins()` setter alle pinner knyttet til LED-ene som utgang.
- `redLedOn()` skrur rød LED på.
- `redLedBlink()` blinker rød LED med 1 sekunds intervall.
- `redLedOff()` skrur rød LED av.
- `greenLedOn()` skrur grønn LED på.
- `greenLedBlink()` blinker grønn LED med 1 sekunds intervall.
- `greenLedOff()` skrur grønn LED av.
- `testIndicatorOn()` skrur test indikator på LED-hodet på.
- `testIndicatorOff()` skrur test indikator på LED-hodet av.

Skjerm

Mappen ved navnet screen i kodebasen består av de tre filene:

- `logoGray.h`
- `screen.h`
- `screen.cpp`

Til sammen utgjør filene hele systemet for skjermbildene som brukes i mappen menu.

I `logoGray.h` brukes det en spesifikk metode for å "bitmappe" egenlagde ikoner som skal brukes til skjermen [48]. Dette gjøres ved å lage matriser som er konstante av datatypen `unsigned char` og representerer farger med heksadesimale verdier, for eksempel `0cff`. For å minimere minnebruk ble det benyttet "monochrome" representasjon av hver piksel, som tilsvarer én byte. Dette sparer plass i programminnet hvor dataen blir lagret. Hver piksel på skjermen representeres som enten av eller på. Standart oppsett for å lage et ensfarget bitmappet bilde er som følger:

```
const unsigned char navn_til_ikon [] PROGMEM = {};
tft.drawBitmap(x, y, navn_til_ikon, pikselStr.Xakse, pikselStr.Yakse, ST7735_WHITE);
```

Når ikonet er definert, brukes funksjonen `drawBitmap` av `tft` objektet til å tegne ønsket ikon, og å definere startposisjon på x og y aksen. Deretter angis variabelnavnet som representerer ikonet, størrelse på bilde i piksler, og fargen som er ønskelig på bilde.

I `screen` blir skjermens pinner definert i henhold til pinneoversikten vist i Tabell 4.26. I koden er det to alternativer til initiering basert på oppkobling av skjermen. Det første alternativet bruker maskinvare SPI. I dette alternativet defineres 3 pinner, mens to pinner kobles opp til ATmega-en sin SPI pinner, MOSI og SCLK. Det andre alternativet definerer alle de brukte pinnene, inkludert MOSI og SCLK som kan være hvilken som helst I/O-pinne. De to alternativene fører til samme oppførsel av skjerm, mens SPI alternativet får skjermen til å respondere betydelig raskere. I filen `screen.cpp` er funksjoner som er listet opp under definert og forklart i listen under.

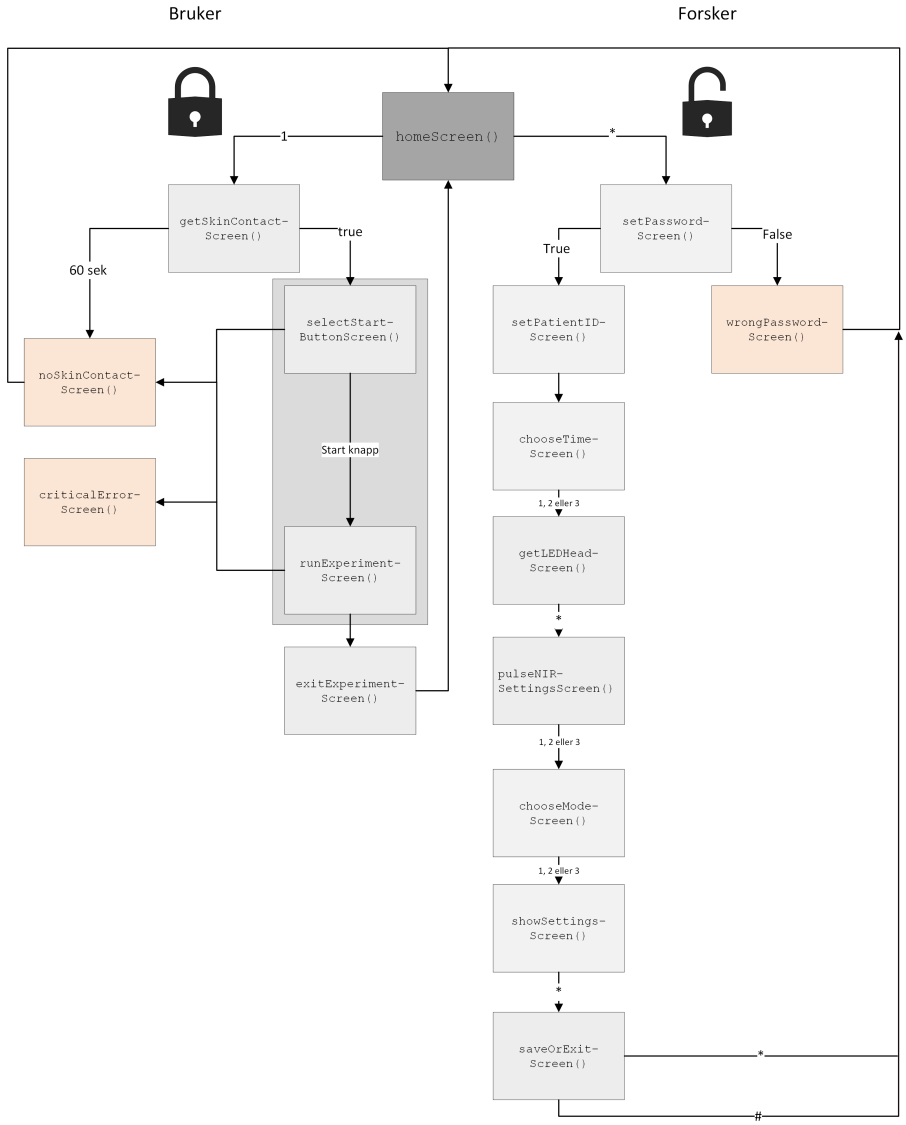
- `initScreen()`: initierer skjermen. Dette gjøres ved å kalle på funksjonen `initR()` av `tft` objektet og farger deretter hele skjermen svart. `tft.fillScreen()` fyller skjermen svart, mens `tft.setRotation()`blir satt til å rotere skjermen 90 grader med klokken.
- `clearScreen()` tømmer skjermen for innhold ved å fylle skjermen med svart farge, nullstiller x og y koordinatene til henholdsvis 0 og 0. Etterfulgt av tilbakestilling av tekststørrelsen til 1.
- `batteryCharge()` funksjonen bruker en unsigned 8-bit int parameter navngitt `battery_charge` i en switch-setning som kaller på fem forskjellige ikoner av et batteriet, avhengig av verdien til `battery_charge`. Ikonene blir tegnet ved hjelp av funksjonen `tft.drawBitmap()`.
- `skinContactStatus()` funksjonen bruker en unsigned 8-bit int parameter navngitt `skin_contact` i en switch-setning som kaller på tre forskjellige funksjoner, som igjen tegner firkanter med farget fyll, avhengig av verdien til `skin_contact`. Firkantene blir tegnet med tilhørende tekst ved hjelp av funksjonen eksempelfunksjonen `xxxRectangle()`.
- `drawDateAndBatteyCharge()` funksjonen bruker unsigned 8-bit int parameter `battery_charge` og parameteren `date[]` som er en char matrise. De to parameterene representerer henholdsvis batteriprosent og dato.

- `homeScreen()` funksjonen nullstiller skjermens bilde, henter funksjonen `drawDateAndBatteyCharge` og bitmapper APT sin logo på hjemskjermen.
- `xxxRectangle()` er en eksempelfunksjon hvor xxx byttes ut med navnet på de tre fargene som skal indikere hudkontaktstatusen på skjermen. xxx erstattes med enten `red`, `yellow` eller `green` og skriver teksten "Hudkontakt: " etterfulgt av en firkant med fargefyll.

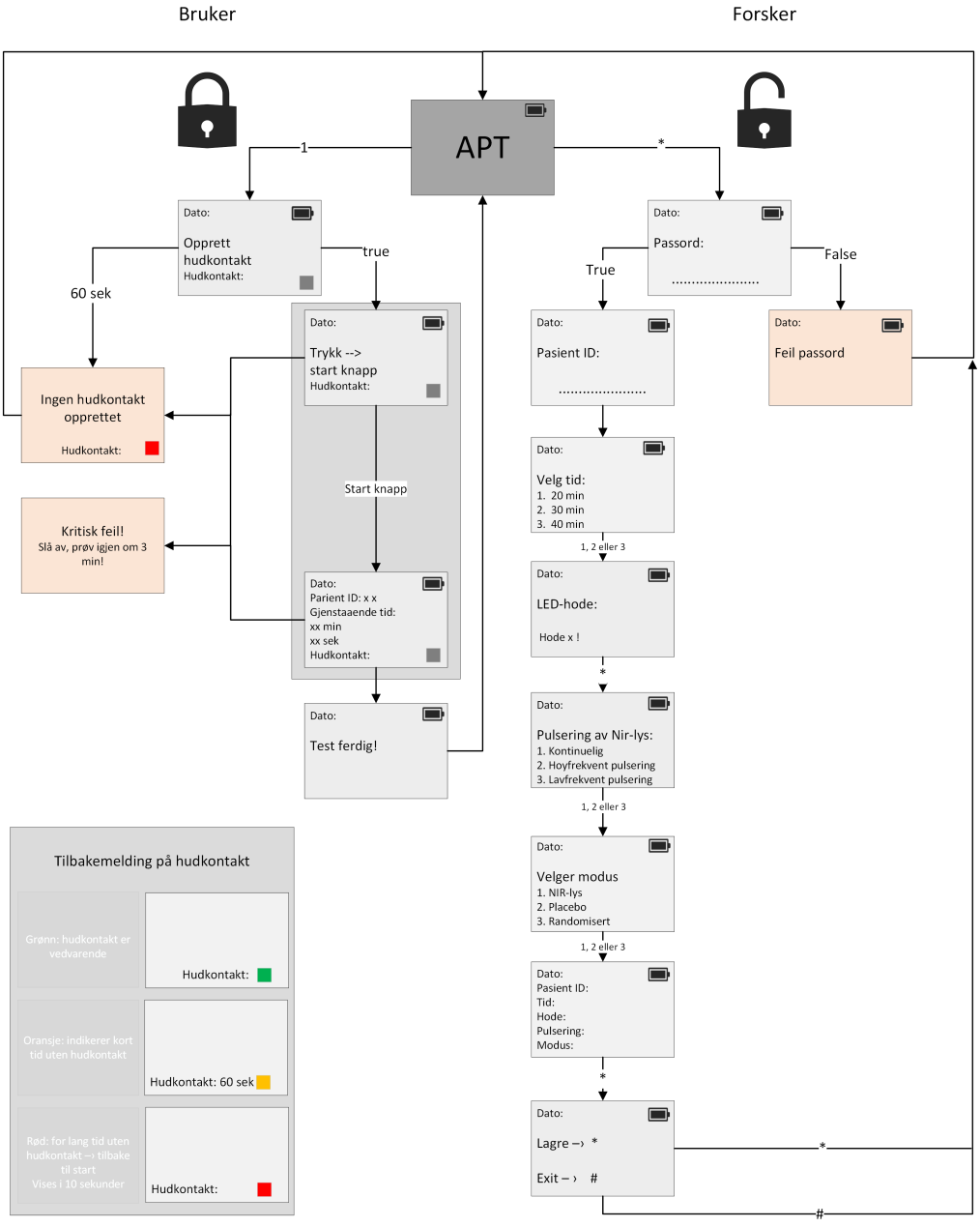
Deretter er alle skjermbilde som blir vist i funksjonene som viser forskerinnstillingene og brukerinnstillingene definert i flytskjema vist i Figur 4.9. Flytskjema er likt strukturert som flytskjema som viser oversikten over funksjonaliteten til skjermbildene vist i Figur 4.10. Det skal være tydelig sammenheng mellom hvilke funksjoner som viser hva og når på skjermen.

Pinnenummer på Arduino Mega 2560	**Akronym på skjerm**	**Navn**	**Funksjon**
5V	VCC	Strømforsyning	Brukes til å gi strøm til skjermen
GND	GND	Jording	Jording for skjermen
32	CS	Chip Select	Aktiverer og deaktiverer kommunikasjon mellom mikrokontroller og skjermen
33	RESET	Reset	Resetter skjermen
34	A0	Data/Command Select	Indikerer om det er data- eller kommandooverføring.
35 (51 - MOSI)	SDA	Seriell data	Sørger for dataoverføringen mellom mikrokontrolleren og skjermen
36 (52 - SCLK)	SCK	Seriell klokke	Gir klokkesignalet for å synkronisere dataoverføringen mellom mikrokontrolleren og skjermen
3.3V	LED	Strømforsyning	Kan brukes til bakgrunnslyset til tft skjermen, for å kontrollere lysstyrken

Tabell 4.26: Pinneoversikt fra Arduino-en til skjermen.



Figur 4.9: Flytskjema for funksjonene brukt i funksjonene RunScientistMenu() og RunUser-Menu().



Figur 4.10: Flytskjema for hvordan tastaturtrykk og skjermbilder fungerer i helhet

Tastatur

I mappen `menu` blir tastaturet initialisert vet hjelp av funksjonen `Keypad` som er hentet fra `Keypad.h`-biblioteket. Denne funksjonen initialiserer tastaturet ved å tilordne tegnene til tastene på tastaturet ved knytte de til rad- og kolonnepinnene. Rad og kolonnepinnene er koblet opp i henhold til pinneoversikten i Tabell 4.27. Hvilke tegn som hører til hvilke pinnekombinasjoner er definert i matrisen med variabelnavn `symbols_on_keypad` som har datatypen char. Dette er en 4x3 matrise som tilsvarer de 12 tastene på tastaturet. Funksjonen `getKey()` av

objektet `customKeypad` leser av tastetrykkverdien som blir tastet inn. Denne koden tilegner verdien som returneres av funksjonen til variabelnavnet `key`.

Pinnenummer på Arduino Mega 2560	**Akronym på tastatur**	**Posisjon på matrisen**	**Funksjon**
22	K	Y1	Rad-pinne
23	J	Y2	Rad-pinne
24	H	Y3	Rad-pinne
25	G	Y4	Rad-pinne
26	F	X1	Kolonne-pinne
27	E	X2	Kolonne-pinne
28	D	X3	Kolonne-pinne

Tabell 4.27: Pinneoversikt fra Arduino-en til tastaturet.

Meny

Oppkobling av menyoppsettet er vist i Figur 5.7. Det isolerte systemet til menyen består av en while-løkke med en if-setning som venter på spesifikke tastetrykk fra tastaturet og funksjonen for denne oppgaven er navngitt `systemWaiting()`. Som navnet tilsier så venter funksjonen på flere bestemte hendelser, og i mellomtiden vises skjermbilde for hjemskjermen `homeScreen`. Dersom '*' er trykket vil funksjonen `RunScientistMenu()` kjøre, dersom tallet 1 blir trykket vil `RunUserMenu()` funksjonen kjøre. Dersom et tall som ikke er av de sistnevnte trykkes inn så vil det ikke skje noe med menyen. Under er det listet opp de tre hovedfunksjonene som lager hele meny-strukturen.

- `systemWaiting()`
- `RunUserMenu()`
- `RunScientistMenu()`

`RunScientistMenu()` er en funksjon som består av en switch-setning som har ni forskjellige tilstander, med ni forskjellige funksjoner som kjører i hvert scenario. Denne switch-setningen "looper" seg kronologisk gjennom forskermeny-innstillingene og tillater forsker å sette en rekke spesifikasjoner for forsøket.

- `waitForPassword`
- `setPatientID`
- `setTime`
- `showLed`
- `chooseNIRsettings`
- `chooseMode`
- `showSettings`
- `SaveOrExit`

`RunUserMenu()` er en funksjon som består av en switch-setning som har fire forskjellige tilstander, med fire forskjellige funksjoner som kjører i hvert scenario. Denne switch-setningen

”looper” seg kronologisk gjennom brukermeny-innstillingene, uten å ha alt for mye funksjonalitet. `RunUserMenu()` er ment til å hente inn data fra forsøket og vise det på skjermen. Da dette systemet foreløpig er isolert fra resten av koden vil switch setningen foreløpig loope seg gjennom skjermbildene hvor det er fastsatt forskjellige parametere på forhånd. Disse parameterne er tiltenkt å knyttes opp mot ”hele systemet” i fremtiden, og batteristatus og dato vil oppdatere seg på skjermen av seg selv.

- `getSkinContact` funksjonen kjører funksjonen for skjermbilde `getSkinContactScreen()`. I koden er det satt tre parametere for skjermen, tilstanden til batteriet, dato, og statusen på hudkontakt.
- `selectStartButton()` funksjonen kjører funksjonen for skjermbilde `selectStartButtonScreen()`. I koden er det også satt tre parametere for skjermen, tilstanden til batteriet, dato, og statusen på hudkontakt.
- `runExperiment` funksjonen kjører funksjonen for skjermbilde `runExperimentScreen()`. I koden er det satt fem parametere for skjermen, tilstanden til batteriet, dato, gjenstående sekunder og minutter av forsøket og statusen på hudkontakt.
- `exitExperiment` funksjonen kjører funksjonen for skjermbilde `exitExperimentScreen`. I koden er det satt to parametere for skjermen, tilstanden til batteriet, dato.

Helhetlig i det isolerte systemet vil både forsker og bruker ha tilgang på menyen. Forskermenyen vil være avgrenset til å sette riktig passord for å sette innstillingene til forsøket, tilbakemelding på innstillingene som har blitt satt og valg om å lagre eller forlate innstillingene. Bruker vil få oppdatering på hva systemet forventer av dem i form av tekst og fargesignalisering av hudkontaktstatus.

Lagring av menyvalg

For å lagre forskers valg av menyvalgene tatt av forsker var det viktig at det ble skrevet til et ikke-flyktig minne. Prototypen må ha mulighet til å skrus av og på, uten at menyvalgene blir glemt. Av den grunn ble det valgt å lagre innstillingene på microSD-kortet. Koden for denne funksjonaliteten ligger i settings-mappen i kodebasen. Den består av to funksjoner: `saveSettingsToFile()` og `getSettingsFromFile()`. Det er antatt at `initSD()` og `initCard()` funksjonene fra writeToFile-mappen blir kjørt først. Disse funksjonene blir forklart nærmere i underseksjon 4.5.3. `TestChoices` er en struktur som brukes og beskrevet nærmere i underseksjon 4.5.2. Videre blir Arduino-biblioteket `SD.h` også tatt i bruk.

`saveSettingsToFile()` tar inn `TestChoices` strukturen og lagrer dette til filen `settings.txt` i rot mappen til microSD-kortet. Filnavnet blir først satt til `settings.txt`, deretter sjekkes det om det allerede eksisterer en fil med navnet `settings.txt` og dersom den eksisterer må den slettes. Her brukes funksjonen `checkIfFileExists()` og `SD.Remove()`. Dette gjøres for å forsikre seg om at tidligere innstillinger ikke påvirker de nye innstillingene som settes av forskeren.

Videre åpnes filen `settings.txt`. Dette gjøres slik at filen kan skrives til ved hjelp av `SD.Open()` funksjonen som settes i skrivemodus (`FILE_WRITE`). Ettersom `saveSettingsToFile()` tar inn testvalgene som en parameter kan denne brukes for å skrive til `settings.txt` filen. Dette vil bli lagret på formatet vist i Kodeliste 4.4. Formatet ble valgt for å gjøre avlesningen enklest mulig. Man kan se at modus, varighet og PBM-frekvensen er heltall og dette forklares nærmere

i underseksjon 4.5.2.

Avslutningsvis lukkes filen, og dersom det skulle oppstå feil ved åpning av filen vil et flagg bli satt til 1.

```
mode:1
duration:2
pwm_freq:2
patient_id:45
experiment_id:105
```

Kodeliste 4.4: Eksempel på formatet til settings.txt-filen som brukes for å lagre menyvalgene.

`getSettingsFromFile()` returnerer `TestChoices` strukturen. Her foregår en del strenghåndtering mens det leser fra filen `settings.txt`. Det første som gjøres er å åpne filen `settings.txt` ved hjelp av `SD.open()` funksjonen i lesemodus (`FILE_READ`). Denne funksjonen er beskrevet nærmere i underseksjon 4.5.3. Dersom filåpningen er vellykket blir innholdet lest linje for linje ved å bruke `File.readStringUntil('\n')` for å definere en linje, som etterfølges av trimming av linjen. Det blir også sjekket for å finne nøkkelord, og disse brukes for å definere hvilken innstilling som blir avlest.

De ulike nøkkelordene som blir søkt etter er `mode:`, `duration:`, `pwm_freq:`, `patient_id:` og `experiment_id:`. Dersom et nøkkelord er funnet begynner strenghåndteringen som kan sees i Kodeliste 4.5. I kodeeksempelet kan man se at verdien bak kolonet hentes ut og konverterer dataen til korrekt datatyper og vil deretter lagres til de tilsvarende feltene i `TestChoices`-strukturen. Til slutt lukkes filen.

Dersom det oppstår en feil ved åpning av `settings.txt`-filen blir et flagg satt til 1. Funksjonen vil videre returnere `TestChoice`-strukturen.

```
if (line.startsWith("mode:"))
{
    String value = line.substring(line.indexOf(":") + 1);
    test_choices.mode = intToMode(value.toInt());
}
```

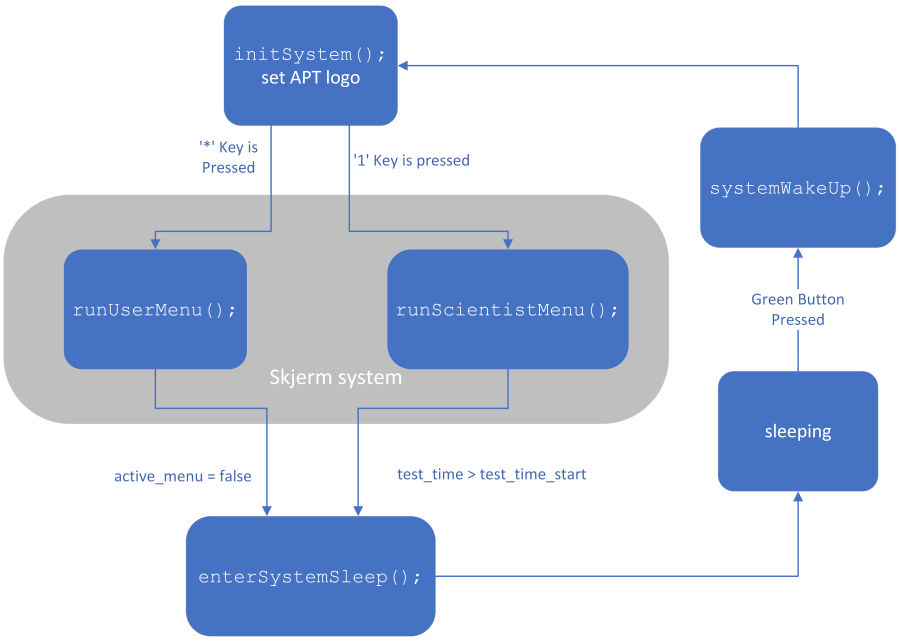
Kodeliste 4.5: Forklarer hvordan strenghåndteringen fungerer for hver linje i settings.txt.

4.6 Hele systemet

Sett i helhetlig perspektiv skal koden bestå av 4 hovedelementer:

- Initiering
- Kjøring av forskermeny og innhenting av innstillinger
- Kjøring av pasientmeny og gjennomføring av forsøk
- Søvnmodus

I Figur 4.11 er systemet overordnet illustrert.



Figur 4.11: Enkel oversikt over systemets flyt.

4.6.1 Oppsett av forsøkslogikk

Det siste som ble gjort av programmering i prosjektet var å utarbeide ett forslag til logikk for forsøket. I denne delen blir alle de ulike delkodene samlet for å danne et helhetlig bilde av hvordan systemet kan operere under forsøk. Denne delen vil ta for seg hvordan forsøket skal gjennomføres, fra den grønne knappen trykkes til forsøket er ferdig og systemet går inn i "power down" modus. Flyten for forsøkslogikken er illustrert i Figur 4.12. Koden for forsøkslogikk finnes i mappen mainFunctions i kodebasen.

Først initieres forsøket. Da settes de ulike timervariablene som skal benyttes for tidtaking. Deretter startes prosessen med å opprette filene som det skal skrives til underveis. Før disse kan opprettes må først dato og tid innhentes for å dokumentere forsøkets oppstart i CSV-filene.

Det vil også bli innhentet de siste lagrede forsøksinstillingene fra TXT-filen `settings.txt`. Når filene er opprettet blir LED-hodets identifikasjon lest av og brukes til å starte NIR-bestråling.

På dette tidspunktet er forsøket offisielt i gang og skjermen skal vise forsøksinformasjonen,

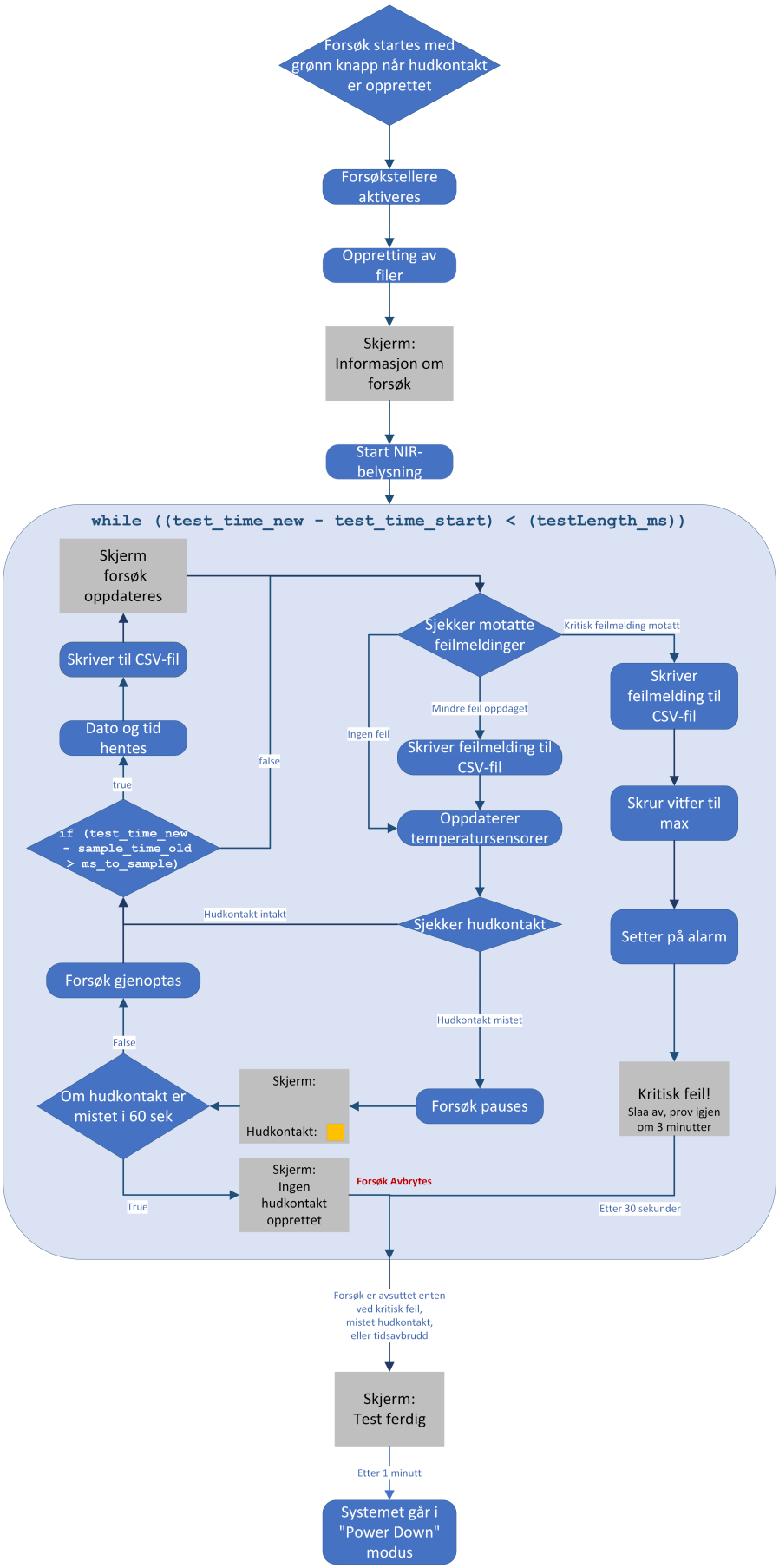
og de grønne LED-indikatorene på styringsenheten og på LED-hodet skal lyse. Når testen er startet vil den kjøre helt til tiden for forsøket er nådd, eller det er oppstått en kritiskfeil.

Hvert minutt vil temperaturverdiene skrives til CSV-filen for forsøksdata. Før dette kan skje må dato og tid innhentes på nytt og formateres som en `dateTimeChar`.

Håndtering av feilmeldinger

En stor del av forsøket vil være å sjekke om det er noen feilmeldinger. Dette skjer ved hver iterasjon av forsøkets while-løkke. Om det blir oppdaget en feilmelding vil denne skrives til CSV-filen for feilmeldinger.

Dersom det skulle oppstå en kritisk feilmelding vil systemet stenges ned. Deretter vil en alarm utløses for å fange brukeren sin oppmerksomhet og viftene settes på maks. Brukeren vil bli bedt om å skru av prototypen, om det ikke er utført innen ett halvt minutt vil systemet gå inn i ”power down” modus.



Figur 4.12: Flyt av programmet under forsøk.

Kapittel 5

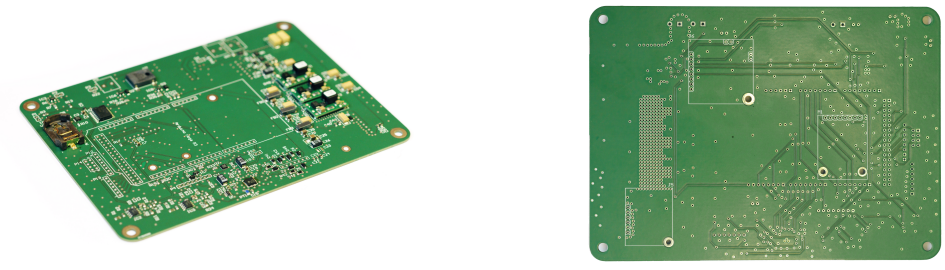
Resultater

5.1 Temperatursensor

Temperatursensoren ble aldri testet, så det er ikke mulig å sjekke om de funket. De analoge inngangene ble testet med termistor i spenningsdeler på en ekstern mikrokontroller. Koden funket fint og ble utviklet videre, uten mulighet for å teste. Det ble utviklet en funksjon for omregning av ADC-verdi til temperatur i grader celsius. Den er basert på en formel fra nett og ikke fra lærebøker. Ved og ikke få testet avlesningene er det ikke mulig å vite om denne omregningen er riktig. Ved en fremtidig arbeid vil dette være mulig å teste og regne om. Det ble laget et varslingsystem som satt ulike flagg basert på om det ble registrert for høy temperatur over for lang tid ved de fire ulike temperatursensorene.

5.2 Design av styringsenheten v.2

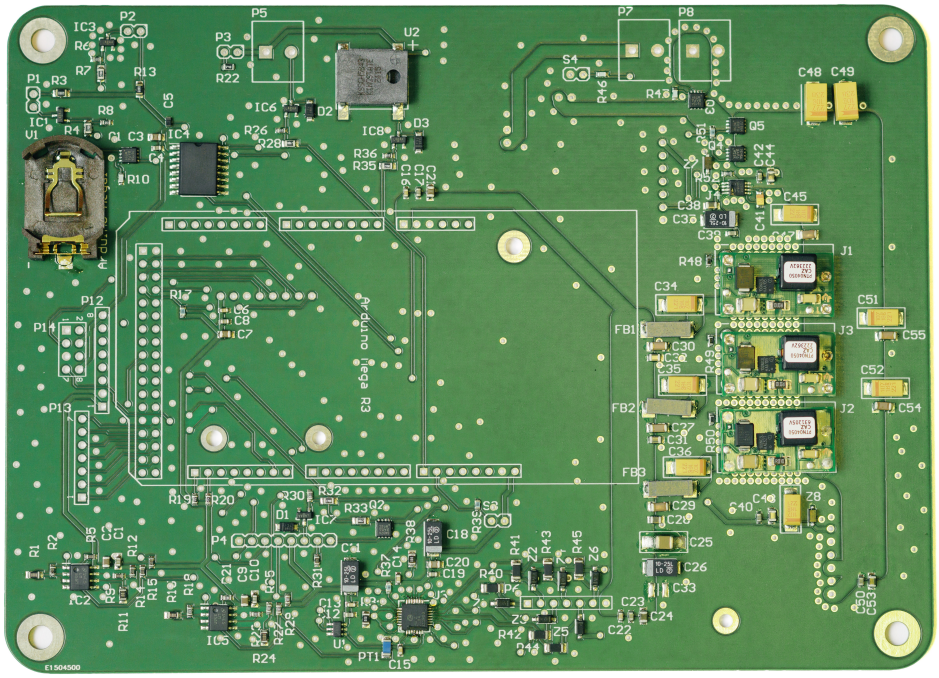
5.2.1 Mønsterkort og kretskort



(a) Kretskort sett fra siden

(b) Mønsterkort sett fra undersiden

Figur 5.1: Bilde av ferdigloddet kretskort og mønsterkort fra undersiden. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.



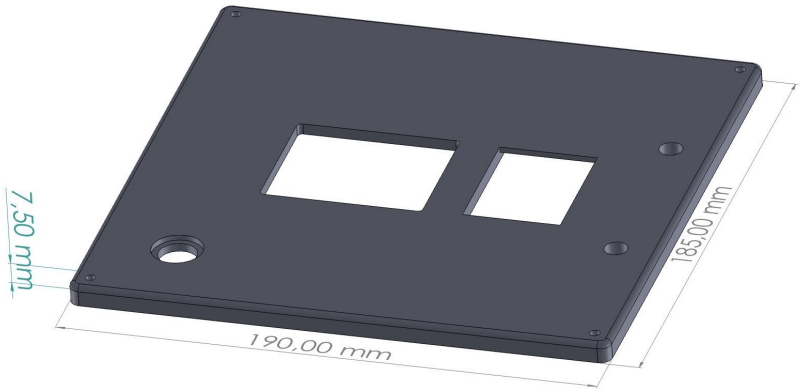
Figur 5.2: Kretskort ferdig loddet. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.

5.2.2 Design og 3D-printing av etui

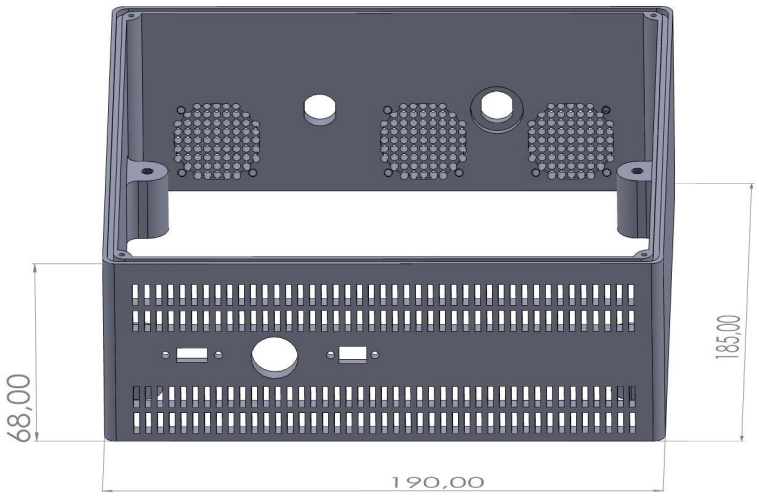
Design av etuiet er basert på to SolidWorks filer: «LED_Electronics_Top» [Tillegg H] og «LED_Electronics_Bottom» [Tillegg I] utarbeidet av Patrick Christian Bösch. Disse filene ble brukt til å videreutvikle etuiet til Prototypen v.2 og resulterte i undernevnte filer.

- «Prototype_v2_lokk» [Tillegg J]
- «Prototype_v2_kjerne» [Tillegg K]
- «Prototype_v2_bunn» [Tillegg L]

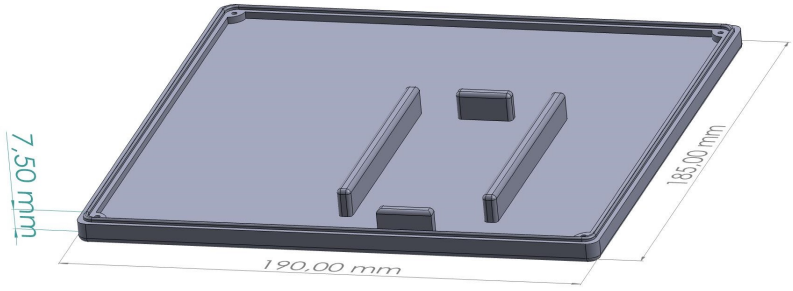
Vedlegg til de fem filene [Tillegg H].



(a) Topp med dimensjoner



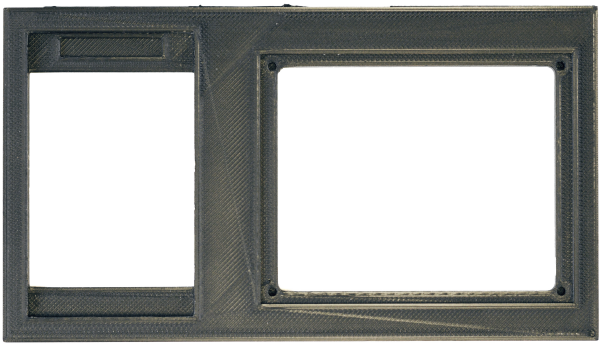
(b) Kjerne med dimensjoner



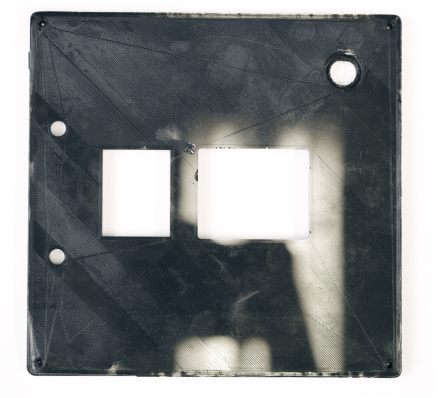
(c) Bunn med dimensjoner

.

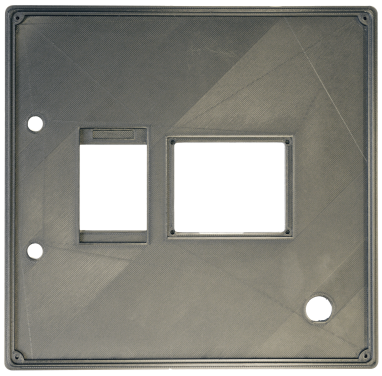
Figur 5.3: Etuiet til styringsenheten til prototype v.2 satt sammen av tre deler



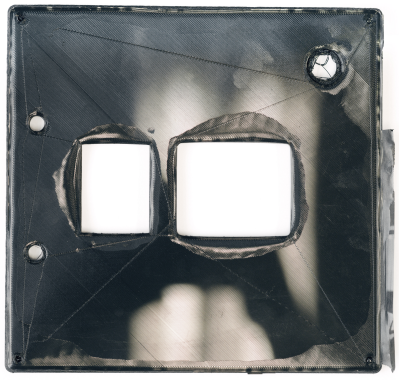
Figur 5.4: Utskjæring av 3D-print for å kontrollere dimensjoner. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.



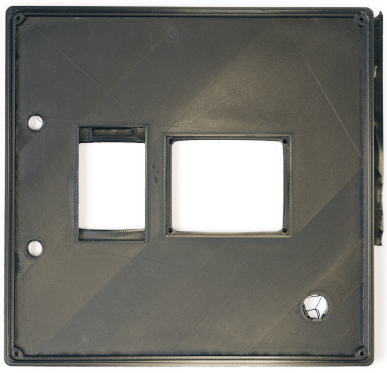
(a) Førsteutkast av topp sett fra utsiden



(b) Førsteutkast av topp sett fra innsiden

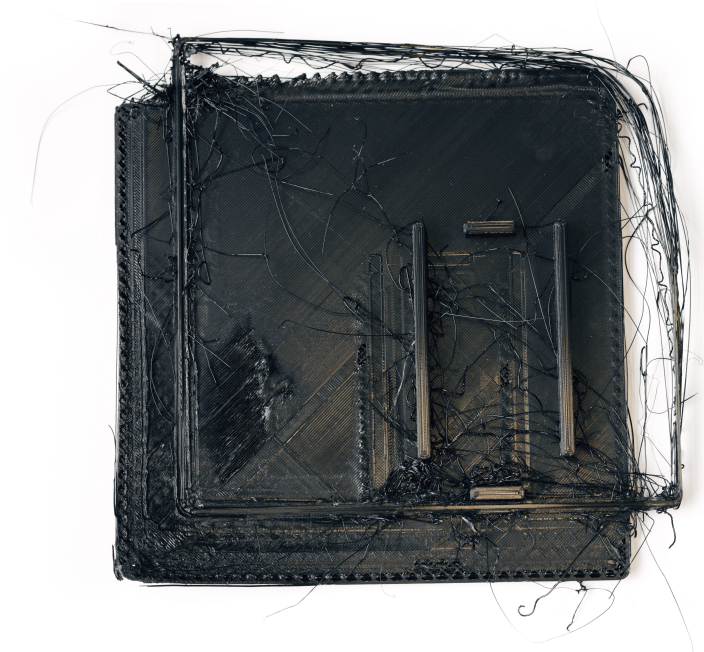


(c) Andreutkast av topp sett fra utsiden



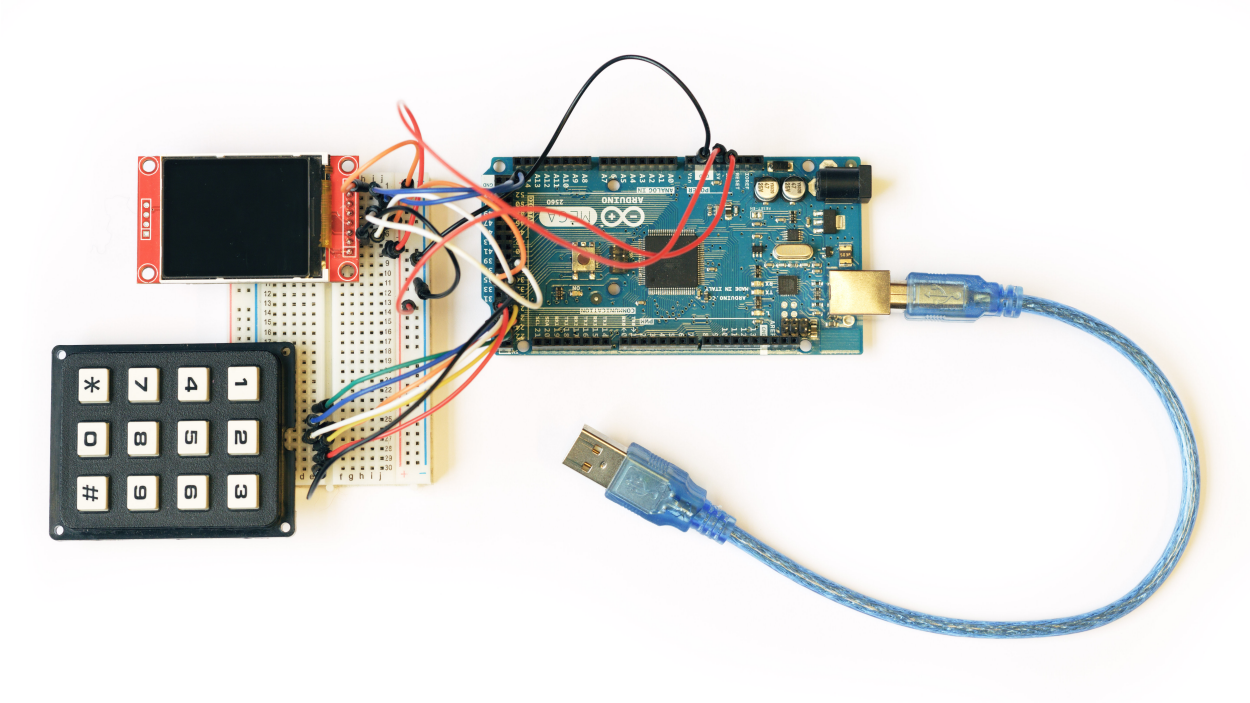
(d) Andreutkast av topp sett fra innsiden

Figur 5.5: 3D-print av lokket til to forskjellige utkast sett fra både innside og utside. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.



Figur 5.6: 3D-print av bunn sett fra innsiden. Fotokreditering: Anders Rønning Petersen v/Institutt for Teknisk Kybernetikk.

5.3 Meny: skjerm og tastatur



Figur 5.7: Oppsett av det isolerte systemet til menyen. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.

5.4 Loggføring av CSV-filer og TXT-filer

Ved bruk av Arduino-biblioteket `SD.h` ble filhåndtering og loggføring på microSD-kortet, som var plassert i grensesnittskortet fra Adafruit, en drøm. Det blir laget tre forskjellige filer: en for sensorverdiene, en for feilmeldinger og en for å lagre innstillingene til forsøket, i en egen mappe. Inntil videre har koden kun blitt testet på et isolert system bestående av en Arduino Mega 2560, et koblingsbrett og noen ledninger. Det ble også laget en egen `main.cpp`-fil for å lage en testkode som kan finnes i `README.md`-filen i `writeToFile`-mappen. Det ble ikke testet på et fullstendig system grunnet ulykken som skjedde med prototype v.1 og strenge beskjeder gitt fra institutt for elektroniske systemer på NTNU.

For å simulere et forsøk og teste hver funksjon og at de fungerer i samsvar med hverandre har det blitt laget testdata. Det har blitt laget en midlertidig `char`-array for både feilmeldingene og sensorverdiene. Verdiene for innstillinger, pasient identifikasjonsnummer og tidsstempel er også simulert. Variablene som brukes er satt i Kodeliste 5.1.

```
TestChoices test;
test.mode = PLACEBO;
test.duration = DURATION_30_MIN;
test.pvm_freq = LOW_FREQ;
test.patient_id = 69;
test.experiment_id = getExperimentId();
const char *start_timestamp = "2021-05-12T12:12:12";
const char *timestamp = "2021-05-12T12:12:12";
char *error = convertErrorToChar(1, error_message, timestamp);
char *data = convertDataToChar(1, 2, 3, 4, timestamp);
```

Kodeliste 5.1: Definisjoner av simulerte innstillinger, pasient identifikasjonsnummer, tidsstempler og data som skal skrives til filene som har blitt brukt for å teste loggføringen.

Etter å ha kjørt testkoden forklart i seksjon 4.5.3 på det isolerte systemet med de simulerte forsøksinnstillingene kunne man koble microSD-kortet til en datamaskin for å se resultatet av filene som ble lagret. Resultatet kan sees i Figur 5.8.

I Figur 5.8a kan man se at det har blitt laget flere filer for hvert forsøk som har blitt gjennomført. I Figur 5.8b kan man se alle filene som tilhører et forsøk og en mappe. Her ligger alle tre filene som er beskrevet tidligere.

Videre kan man se nærmere på innholdet til de genererte filene. Innstillingene for forsøket blir loggført riktig i forhold til Kodeliste 5.1 i Figur 5.8e. I Figur 5.8c og Figur 5.8d kan man se temperatur og feilmeldingene blir loggført i riktig filer og på riktig form i forhold til header-ene i CSV-filene.

(a) Oversikt over mappene som blir generert etter å ha restartet koden flere ganger.

(b) Oversikt over innholdet i en mappe tilhørende et forsøk.

```
1   datetime,temp_pcb,temp_air,temp_skin,temp_led
2   2021-05-12T12:12:12,23.25,22.5,33.2,21.2
3   2021-05-12T12:12:42,23.25,22.5,33.2,21.2
4   2021-05-12T12:13:02,23.25,22.5,33.2,21.2
5   2021-05-12T12:13:32,23.25,22.5,33.2,21.2
6   2021-05-12T12:14:02,23.25,22.5,33.2,21.2
7   2021-05-12T12:14:32,23.25,22.5,33.2,21.2
8   2021-05-12T12:15:02,23.25,22.5,33.2,21.2
9   2021-05-12T12:15:32,23.25,22.5,33.2,21.2
10  2021-05-12T12:16:02,23.25,22.5,33.2,21.2
11  2021-05-12T12:16:32,23.25,22.5,33.2,21.2
12  2021-05-12T12:17:02,23.25,22.5,33.2,21.2
13  2021-05-12T12:17:32,23.25,22.5,33.2,21.2
14  2021-05-12T12:18:02,23.25,22.5,33.2,21.2
```

(c) Oversikt over temperaturene som ble skrevet til LOG_123.csv.

```
1   datetime,error_code,error_msg
2   2021-05-12T12:12:12,1,Error message
3   2021-05-12T12:12:12,1,Error message
4   2021-05-12T12:12:12,1,Error message
5   2021-05-12T12:12:12,1,Error message
6   2021-05-12T12:12:12,1,Error message
7   2021-05-12T12:12:12,1,Error message
8   2021-05-12T12:12:12,1,Error message
9   2021-05-12T12:12:12,1,Error message
10  2021-05-12T12:12:12,1,Error message
11  2021-05-12T12:12:12,1,Error message
12  2021-05-12T12:12:12,1,Error message
13  2021-05-12T12:12:12,1,Error message
14  2021-05-12T12:12:12,1,Error message
```

(d) Oversikt over feilmeldingene som ble skrevet til ERR_123.csv.

(e) Oversikt over informasjonstekstfilen som ble laget.

Figur 5.8: Resultater fra loggføring av sensorverdier, feilmeldinger og innstillinger på forsøket gitt testkoden som ble brukt.

Kapittel 6

Diskusjon

Diskusjonsdelen vil først ta for seg drøfting av de ulike valgene som ble tatt underveis i prosjektet. Deretter vil andre forutsetninger som har påvirket resultatet, bli presentert. Til slutt vil eventuelt videre utvidelser av prototypen bli diskutert.

6.1 Design av styringsenhet til prototype v.2

Denne delen vil diskutere bakgrunnen for de ulike valgene som ble gjort under utviklingen av prototype v.2. Dette vil inkludere valg av komponenter, begrunnelse for plassering og design av etuiet.

6.1.1 Valg av komponenter

Komponentene som ble undersøkt for utvidelsen av kretskortet inkluderte ulike skjermer, tastatur og WiFi-moduler. Det ble og vurdert å benytte en annen mikrokontroller enn Arduino Mega 2560 som ble benyttet på prototype v.1. Det har blitt gjort vurdering på strømtrekk samt egenskapene som vil kreves av komponentene fra systemet.

Vurdering av effektforbruk

Når det ble diskutert hvilke komponenter som kunne inkluderes på det nye kretskortet, var det viktig å vurdere om de oppfylte kravene til effektforbruk.

For å sikre at prototypen er funksjonell i bruk, var det ønsket at den skulle kunne gjennomføre tre fulle forsøk mellom hver lading av batteriet. For å vurdere dette ble regnearket fra ”Device for improved insulin absorption in diabetes type 1” brukt som utgangspunkt [1]. Tidligere beregninger viser at de komponentene som har størst strømtrekk er de ulike LED-hodene mens de er i bruk. De ulike NIR-matrisene vil ha et strømtrekk i området 0,35 til 3,0 Ampere [Tillegg C]. Til sammenlikning vil de fleste komponenter med integrerte kretser ha strømtrekk i mA området. Derfor vil prototypens evne til å oppfylle kravet om 3 forsøk per fulladet batteri avhenge av hvilket LED-hode som blir brukt.

Valg av tastatur

Tidlig i prosjektfasen ble det diskutert alternative løsninger til DIP-bryterene som var plassert på innsiden av boksen i prototype v.1. Disse DIP-bryterne ble ansett som en suboptimal løsning som ville være upraktisk i bruk. I tillegg ville det begrensede antall innstillingsmuligheter med 2x6 brytere være ugunstig. Derfor ble det vurdert å erstatte DIP-bryterene med et tastatur. Ved valg av tastatur ble to ulike alternativer hovedsaklig sammelignet, som vist i Figur 6.1. Siden strømforbruket ikke lenger var avgjørende for valg av tastatur, ble vektleggingen rettet mot enkel monterin, god kvalitet og datablad med beskrivende dimensjoner. EOZ-tastaturet hadde skruehull for enkel montering og var laget av hard plast, mens det andre tastaturet ikke hadde noe form for feste og var laget av et fleksibelt materiale som gjør det bøyelig. Etter å ha sammenlignet de forskjellige egenskapene til tastaturene, besluttet gruppen med å velge tastaturet fra EZO [49] [50].

(a) Tastaturet fra produsenten EOZ [49].

(b) Tastaturet fra ukjent produsenten [50].

.

Figur 6.1: Bilde av de to forskjellige tastaturene som ble sammenlignet.

Valg av skjerm

Etter å ha undersøkt markedet for skjermutvidelser til prosjektet, har gruppen identifisert og vurdert tre forskjellige alternativer. Disse alternativene vil bli presentert, etterfulgt av en diskusjon om hvorfor gruppen valgte den spesifikke skjermen.

- Det første alternativet var skjermen fra Kuongshun som er avbildet i Figur 6.2a.
- Det andre alternativet var skjermen fra Adafruit avbildet i Figur 6.2b
- Det siste alternativet var en skjerm fra ukjent produsent Figur 6.2c.

Valg av pinner for oppkobling av skjerm
Når skjermen skulle kobles opp var det to mulige måter å gjøre dette på. En løsning ville være å koble den opp til I/O pinnene, men den kunne også benytte SPI kommunikasjon for data overføring. Det som taler for å koble opp skjermen til SPI pinnene er at oppdatering av grafikken på skjermen vil gå mye raskere. Det som var mer problematisk var det at SPI pinnene allerede var brukt for minneutvidelsen. Fordi SPI protokollen er laget for at flere enheter skal kunne være koblet på samtidig skulle det likevel være mulig å koble på skjermen til SPI pinnene samtidig som minneutvidelsen. Når det tidlig i prosjektet ble avgjort hvilke pinne skjermen skulle kobles på, ble gruppens manglende erfaring med SPI programmering den faktoren som veide tyngst. Da ble I/O pinnene 32 - 36 på Arduino Mega 2560 valgt.

(a) Skjerm fra Kuongshun ”0.96"OLED Display for Arduino (128x64)” [51].

(b) Skjerm fra Adafruit ”1.8"Color TFT LCD display with MicroSD Card Breakout - ST7735R” [52].

(c) Skjerm fra SparkFun [36] .

.

Figur 6.2: Bilde av de tre forskjellige alternativene som ble vurdert.

Valg av WiFi-modul

Tidlig i prosjektet ble beslutningen tatt om å utsette implementeringen av WiFi-modulen og sette det som videre arbeid (seksjon 6.4.1). Selv om modulen ble inkludert på kretskortet i Prototypen til styring styringsenheten, ble den verken montert eller programmert på styringsenheten til prototype v.2. WiFi-modulen som ble valgt var ESP8266 ESP-01, og dette valget ble gjort av flere grunner:

- Modulen var kompatibel med mikrokontrolleren Arduino Mega 2560
- Hadde en relativt lav pris sammenlignet med andre WiFi-moduler
- Støtter forskjellige WiFi-protokoller
- Hadde god prosesseringskraft
- Hadde en stor brukerbase
- Var designet for å ha et lavt strømforbruk

Strømforbruket var spesielt viktig. I hvilemodus hadde modulen et typisk strømtrekk på rundt 0.9 mA, og under bruk kunne det variere mellom 70-170 mA. Om nødvendig kunne modulen også benytte seg av funksjonen ”deep sleep” for å redusere strømforbruket til 10 µA [53].

Valg av mikrokontroller

Tidlig i prosjektfasen ble det vurdert om det skulle benyttes en annen mikrokontroller enn den som ble brukt på prototype v.1, som ble designet for å bruke utviklingskortet Arduino Mega 2560 Rev3. Dette utviklingskortet er utstyrt med en ATmega2560 prosessor som har et

mangfold av tilgjengelige periferier som er nødvendig for prosjektet:

- 54 Digitale I/O pinner
- 16 Analog input pinner
- 256 kB flash minne for lagring av programkode
- En 8 bits teller
- Fire 16 bits tellere
- TWI kommunikasjon
- SPI kommunikasjon

Utviklingskortene til Arduino bruker ulike mikrokontrollere produsert av Microchip, med noe utvidelse i elektronikken. Generelt sett er Arduino-utviklingskort godt tilpasset utviklingsprosjekter, med lett tilgjengelige pinneinnganger og god dokumentasjon. Arduino tilbyr en rekke veiledninger som er tilpasset nybegynnere, samt flere biblioteker som forenkler utvikling av kode for spesialkomponenter[2].

Når det ble vurdert å bruke en annen mikrokontroller enn ATmega2560, ble det undersøkt det store utvalget av mikrokontrollere tilgjengelige fra Microchip. Dette gjør det mulig å velge en mikrokontroller bedre egnet for prosjekt. Microchip har også utviklet sitt eget utviklermiljø kalt Microchip studio. Microchip studio er regnet som et profesjonelt utviklerverktøy enn Arduino IDE med egen debugger og simuleringsverktøy[54]. Valg av programvare for utvikling av kode ble ikke avgjørende for valg av mikrokontroller, da det allerede var bestemt at koden skulle skrives i "Visual studio codesom gruppen har mer erfaring med å bruke. En vurdering som ble gjort og veide tungt var mengden arbeid som ville være nødvendig for å benytte en annen mikrokontroller. Det ville medføre en større jobb for å omkobling av alle ledningsbanene på kretskortutlegget. For å sikre nøyaktig og riktig utførelse av denne oppgaven ville det ha krevd betydelig tid og ressurser, samtidig som den potensielle gevinsten var begrenset. Konklusjonen ble at det ville være mest hensiktsmessig å beholde Arduino Mega 2560 i designet for dette prosjektet.

6.1.2 Hensyn tatt under design av kretskort

Når komponentene skulle plasseres på kretskortet i Altium var ett av hensynene som ble gjort at det skulle passe inn i det allerede eksisterende designet for prototype v.1. Denne vurderingen ble gjort med tanke på omfanget av arbeid som ble lagt i prototype v.1 og den begrensede tidsperioden for prosjektet. Valget av Altium som programvare for utvikling av kretskortet falt naturlig da det også ble brukt til å designe versjon 1. Derfor var det lett å fortsette på arbeidet som allerede var gjort[1].

På kretskortet for styringsenheten til prototype v.1 var det tatt hensyn til at alle komponentene som skulle være plassert i lokket til boksen skulle være plassert så langt mulig mot hengslene som mulig. Dette var en forutsetning for at ledningene ikke skulle bli utsatt for rykninger og slitasje hver gang DIP-bryterene på innsiden av boksen skulle endres på. Når DIP-brytere skulle fjernes for prototype v.2 til fordel for et tastatur var det ikke lenger like viktig å ta hensyn til komponentenes mobilitet når boksen åpnes og lukkes. Dette er fordi hele brukergrensesnittet skal befinne seg på boksen sin overflate som vil begrense behovet for å åpne boksen jevnlig.

6.1.3 Lodding av kretskort

Etter at det ble produsert feil kretskort, ble det nødvendig å investere ekstra tid og ressurser i å produsere et nytt kort. Dette medførte at arbeidet med lodding ble vesentlig forsinket. Fra starten av prosjektet var det et mål at gruppen skulle gjøre mesteparten av jobben med å lodde komponenter på mønsterkortet selv, med hjelp og veiledning fra Patrick C. Bösch. Etter anbefaling fra ressurser innad på instituttet for elektroniske systemer, ble det konkludert med at gruppen ikke hadde nok kompetanse på lodding av små overflatemonterte komponenter til å utføre en så omfattende loddejobb. Derfor ble det besluttet at arbeidet skulle utføres av elektronikk og prototypelaboratoriet som tar slike oppdrag gratis for studenter.

Tilbakemeldinger etter lodding

Når loddejobben var fullført av elektronikk og prototypelaboratoriet ble det gitt et par tilbakemeldinger til ting som ikke var helt optimalt i designet.

Noen av kommentarene var gruppen allerede klar over, som at det var flere silketrykk som var dårlig plassert og bortgjemt på mønsterkortet Figur 5.2. I tillegg fikk gruppen vite at måten mønsterkortet benyttet jordlag over hele platen ikke var optimalt, da dette vil føre til at oppvarming i reflow ovn ville ta lenger tid.

Det var også en feil knyttet til en kondensator på kortet. Kondensatorer C25 på kreskortet hadde for stort fotavtrykk for komponenten som ble brukt. Det ble forsøkt å lodde den på, men det er usikkert om den vil ha god nok kontakt til å lede strøm. Dette er ikke undersøkt da alt arbeid med utvikling av prototype ble satt på is (underseksjon 6.3.4).

Det er også oppdaget at det er en kondensator som ikke er kommet med i loddejobben. Kondensatoren C33 er merket på mønsterkortet, men grunnet dårlig synkronisering av markeringer på skjematikk og kretsdesign har denne ikke blitt inkludert i komponentlisten [Tillegg G]. I ettertid har hele Altium prosjektet blitt undersøkt for å finne ut hvilken komponent dette tilsvarer, men den har ikke blitt avdekket. Dette må undersøkes videre før prototypen monteres ferdig.

Buzzeren som skulle loddes på, hadde også et problem ved seg. På undersiden var det plast som gjorde loddingen vanskelig. Derfor valgte lærlingen fra elektronikk og prototypelaboratoriet å fjerne denne plasten. Det er antatt at denne plasten skulle dempe vibrasjoner fra buzzeren ned på kretskortet og nærliggende komponenter. Det ble vurdert at dette ikke er et kritisk problem, men gruppen har ikke gjort noen vurdering på hvordan slike vibrasjoner vil kunne påvirke elektronikken over tid.

En kommentar som gjelder kortet som helhet er at det er brukt for mange via-hull. Ressursene på Elektronikk og prototypelaboratoriet mente at antall vias som er benyttet er overflødig og kun tar unødvendig mye plass.

6.1.4 Design og 3D-printing av etui

Design

Grunnen til at boksen ble utvidet til å bestå av tre deler fremfor to, var fordi det var vanskelig å aksessere minneutvidelsen, som er plassert på undersiden av det fastmontert mønsterkor-

tet. Med denne plasseringen måtte deler av prototypen v.1 demonteres for å kunne ta ut hele kretskortet og få tilgang til microSD-kortet. Ved å skjære bort bunnen ble det lettere å få tilgang til minneutvidelsen og batteriet uten å måtte demontere deler av prototypen. Avtakbar bunn reduserer påvirkningene ledningene og elektronikken rundt kretskortet har for slitasje, da aksessering av microSD-kortet gjøres jevnlig. Konklusjonen ble å lage en bunn som kunne skrus av og på. Bredden på boksen ble justert for å ha plass til å ta ut mønsterkortet.
Det ble bestemt å gå bort fra den originale løsningen med DIP-brytere tidlig i prosessen. Det ble fremlagt at DIP-bryterene, som var plassert på innsiden av boksen, ville redusere funksjonaliteten til brukergrensesnittet betydelig. Siden det gjør hele prosessen mindre intuitivt og brukeren blir derfor mer avhengig av en brukermanual for å sette innstillingene på forsøket. Som en erstatning til DIP-bryterne ble det implementert en skjerm og tastatur. Forsker vil ikke lenger være avhengig av å åpne boksen mellom hvert forsøk for å sette innstillingene. Derfor ble det ikke lenger nødvendig med hengsler for åpne- lukkefunksjon og det ble derfor skjært ut skruehull gjennom hele konstruksjonen for å fastmontere lokk og bunn. Skjerm og tastatur ble plassert på lokket for å stemme overens med LED-ene og knappene som var del av brukergrensesnittet til prototype v.1.

3D-printing

Det kan konkluderes med at det var forskjellige faktorer som førte til at alle printene var deformerte etter ytteligere justeringer. Siden 3D-printeren er av S5 serie som har byggeplate i et lukket system uten for mye ytre påvirkninger av hverken vind eller temperaturer, så er det bemerkelsesverdig at det oppstod ”warping” på alle printene [55]. Det viste seg også å ha en sammenheng mellom kontaktflatearealet og hvor stor grad det oppstod ”warping”. Sammentrekning oppstod hyppigere på større kontaktflater. Dette kan derfor skyldes at varmeelementet i byggeplaten er ujevnt fordelt og ikke strekker seg langt nok ut til ytre kanter av byggeplaten. Det kan også ha en sammenheng med at det printes på glass og ikke får godt nok ‘’grep” når etuiet blir printet. Tiltak som ble gjort for å bedre dette var å vaske byggeplaten med sprit for å bedre grepet. Da dette ikke viste noe tegn til forbedring ble temperaturen til byggeplaten oppjustert. Etter dette skjedde det også ujevnheter i 3D-printet, men denne gangen så det ut til at det var på grunn av for varm byggeplate, da store deler av kontaktflaten var ujevn, også ved utskjæring av tastatur og skjerm. Etter x antall mislykkede forsøk og én 3D-printer ned for telling valgte gruppen å avslutte printingen og kun levere ferdige beskjærte SolidWorks-filer da dette var en tidskonsumerende prosess som kan være lett å løse når forholdene er optimale. Det vil derfor være lettere å sette på print i senere tid når 3D-printerne er fikset og feilsøkt og andre printejobber blir gjennomført uten feil. Tiltak som er mulig å gjøre i fremtiden er å påse at byggeplaten ikke har hakk i seg, noe den ene printeren hadde. Det kan også være lurt å spraye byggeplaten med hårspray for å få bedre feste, samt. nedjustere kjøleviften. I tillegg kan det være lurt å ha tilgang på verktøy for overvåking av printeprosessen. Dersom det oppstår problemer er det fort gjort å overvåke og avbryte printe-prosessen og prøve på nytt uten at det får for store konsekvenser for 3D-printerene.
Et annet tiltak som er mulig å gjennomføre er å bestille etuiet laget i feks. metall fra Mekanisk verksted [56]. De har mulighet til å sveise og sette sammen 3D-modellen på bestilling, dersom man legger ved STL-filen har verkstedpersonell mulighet til å gjøre ”pathing”, og frese ut produktet. Ved å bestille et produkt i metall vil det annullere risiko for av boksen smelter under en eventuell brann og redusere feil som oppstår under produksjon, da dette gjøres av

fagpersonell. Gruppens vurdering er at dette vil være en bedre løsning som vil gi en mer robust og gjennomført prototype.

6.2 Kode C++

6.2.1 Valg av kodespråk og bruk av Arduino biblioteker

Fra start var det bestemt at koden skulle skrives i AVR C. Selv om aksessering av den interne logikken og programmering av de enkleste komponentene ble gjort i C, ble det mer avansert når minneutvidelse, tastatur og skjerm skulle programmeres. Disse hadde et grensesnitt som var vanskelig og aksessere uten å laste ned egne biblioteker som kunne automatisere deler av de mer kompliserte prosessene.

Da selve prosjektet ikke er laget for privat bruk, men for en bedrift, var det viktig å unngå biblioteker på nett som ikke er lov til å distribuere kommersielt. Majoriteten av Adafruit og Arduino sine biblioteker er ”open-source” og gjør det derfor lettere å bruke enn de som har strengt regulert opphavsrett. Siden bibliotekene deres er skrevet i C++ falt det seg naturlig å ikke bruke tiden på å oversette bibliotekene til Avr C, men bruke de allerede eksisterende bibliotekene til formålet. De bibliotekene som var ønskelig å bruke i prosjektet krevde at koden måtte skrives i Arduino C, en versjon av C++ kodespråket. Det ble gjort en vurdering på om det ville være rotete å blande kodespråkene i ett og samme prosjekt, men ble besluttet at det var nødvendig. Fokuset ble derfor rettet mot å ha ett klart skille mellom hva som er skrevet i AVR C og Arduino C. Det er først når hele systemet settes sammen i mappen mainFunctions at de to kodespråkene blandes. Som konklusjon er det mulig å bruke koden til kommersielt bruk men for å kunne det må kildekoden være offentlig. Bruk av Arduino-bibliotekt krever at kildekoden til prosjektet er offentlig. `SD.h`-biblioteket er underlagt ”GNU general public license”, anbefales det å gå bort ifra et slikt bibliotek til senere utvikling og erstatte det med et bibliotek med en annen lisens [57]. En oversikt over bibliotekene og deres lisenser kan finnes i Tabell 6.1.

Bibliotek	**Lisensavtale**	**Opphavsrettmerknad**
chris–a/Keypad [58]	GNU GENERAL PUBLIC LICENSE	Copyright (C) 2007 Free Software Foundation, Inc.
adafruit/Adafruit ST7735 and ST7789 Library [59]	MIT License	Written by Limor Fried/Ladyada for Adafruit Industries, all text above must be included in any redistribution.
adafruit/Adafruit GFX Library [60]	BSD License	Copyright (C) 2012 Adafruit Industries. All rights reserved.
arduino-libraries/SD [61]	GNU GENERAL PUBLIC LICENSE	Copyright (C) 2007 Free Software Foundation

Tabell 6.1: Oversikt over bibliotekene brukt for koding av skjerm, tastatur og minneutvidelsen og deres opphavsrettsmerknad.

6.2.2 Loggføring av sensorverdier og feilmeldinger

Gruppen bestemte seg tidlig for å bestille nye komponententer til prototype v.2 og denne bestillingen ble levert rett før påsken (Tillegg B). Dessverre viste det seg at leveransen av "microSD card breakout board+" fra Adafruit var forsinket og ankom først i starten av Mai. Det kunne vært mulig å bruke grensesnittskortet som allerede var på styringsenheten til prototype v.1, men denne var utilgjengelig grunnet designet og monteringen av etuiet og kretskortet. Grensesnittskortet var på innsiden av etuiet og på undersiden av det fastmonterte kretskortet, uten muligheter for å åpnes på undersiden. Som et resultat av dette ble det umulig å aksessere microSD-kortet uten å ødelegge etuiet på undersiden. Det ble derfor både vanskelig å teste om loggføringen fungerte på et isolert system samt. vanskelig å teste om loggføringen fungerte ordentlig uten tilgang til microSD-kortet. Grunnet manglende tilgang på grensesnittkortet på prototype v.1 og forsinkelse på det nye, ble kodingen av loggføringen utsatt til starten av mai, når grensesnittskortet fra Adafruit ble mottatt.

Det har også vært manglende dokumentasjon og tilgjengelige læreressurser for loggføring av sensorverdier og feilmeldinger til CSV-filer uten bruk av Arduino-biblioteker. Dette medførte at det beste valget for denne gangen var å bruke det godt dokumenterte `SD.h`-biblioteket til Arduino. Det som var hovedfokuset var å produsere kode som gjør som forventet og fungerer som ønsket.

6.3 Andre faktorer som har påvirket resultatet

Det har underveis i prosjektfasen oppstått en rekke hendelser som har påvirket forløpet og resultatet av prosjektet. Disse vil drøftes grundig, og konsekvensene vil presenteres.

6.3.1 Problemområder

Denne delen vil ta for seg problemområder i prosjektet med hovedfokus på prototype v.1 designet av Patrick C. Bösch.

6.3.2 Avvik i oppgavebeskrivelse

Oppgavebeskrivelsen til prosjektet ble publisert høsten 2022 [Tillegg N]. Beskrivelsen presiserer at det er påbegynt arbeid som omhandler bygging av den fysiske prototype v.1, og at oppgavens formål er å fullføre det gjenstående arbeidet på prototypen. Deretter vektlegges det at hovedarbeidet består av programmering og utforming av brukergrensesnitt, samt. testing og karakterisering i samråd med forskerne i APT. I oppgavebeskrivelsen blir det fremlagt en lite konkret beskrivelse på hvor langt i prosessen prototypeutviklingen er. Leseren kan få inntrykk av at den fysiske komponenten er tilnærmet ferdigbygget før oppstart av oppgaven, dette stemte ikke. Gruppen ble stående uten noen klare oppgaver å starte på i oppstartsfasen. Når Patrick C. Bösch hintet til at det kunne være aktuelt å designe en versjon 2 av prototypen som en del av bacheloroppgaven. Dette ble lagt til den opprinnelige oppgavebeskrivelsen for å ha noe konkret å jobbe med.

Det er uvisst når utvikling av mønsterkortet til styringsenheten ble påbegynt, men den ferdigstilte prototypen som gruppen basere oppgaven på var ferdigstilt i slutten av mars. Det ble

derfor en overlappende periode i februar-mars der gruppen utviklet sitt eget mønsterkort parallelt som Patrick C. Bösch ferdigstilte sitt. Dette førte til mye inneffektivitet og dårlig utnytting av arbeidskapasitet. Dette medførte videre forsinkelser i programmeringsdelen av oppgaven, fordi gruppen ikke hadde mulighet til å teste den grunnleggende koden før slutten av mars. Siden gruppen kun skulle utvikle brukergrensesnitt og programmere var det mye tid som gikk før gruppen kunne begynne på denne delen av oppgaven. Det er derfor et helhetsinntrykk at prosjektbeskrivelsen samt. hva gruppen og oppdragsgiver avtalte i starten, ikke samsvarte med hendelsesforløpet, da gruppen først fikk startet på det som var beskrevet som oppgaven i slutten av mars.

6.3.3 Feil og mangler

Når ulike deler av koden skulle testes kom det frem at flere deler av maskinvaren ikke fungerte som det skulle. På grunn av den sene leveransen av prototypen v.1 ble ikke disse feilene oppdaget før etter at kretskortet til prototype v.2 ble bestilt. Dermed var det ikke mulig å reparere disse innenfor prosjektets tidsramme. De delene av prototypen som knyttes direkte opp mot sluttresultatet av brukergrensesnittet har blitt feilsøkt med god hjelp fra ressurser internt på NTNU slik at de kan utbedres i nærmeste fremtid. I kommende underkapitler er det listet opp to eksempler på elementer som ble feilsøkt.

Temperatursensorene

En av problemene med maskinvaren ble oppdaget under programmeringen av temperatursensorene. I startfasen ble det tatt i bruk en ekstern mikrokontroller for å isolere systemet. Dette isolerte systemet besto av en analog inngang som ble koblet over en termistor i en spenningsdeler. Avlesningen fungerte og riktig temperatur ble avlest. Da koden ble lastet over på mikrokontrolleren i prototypen v.1 ble det avlest en konstant spenningsverdi. Etter mange timer med feilsøking i koden uten resultat, begynte feilsøkingen på elektronikken til prototypen. Anders R. Petersen v/Institutt for Teknisk Kybernetikk på NTNU bidro med veiledning på denne delen. Med et oscilloskop kunne man se signalet som ble avlest fra de analoge inngangene. De ga en kostant verdi på ca. 0.5 V. Det ble dessverre ikke tatt bilde av dette før ulykken med protypen. På grunn av dette konkluderte gruppen med at arbeidet på temperatursensor-koden var riktig og koden ble videreutviklet med et feilmeldingssystem. Det er derfor ikke mulig å teste koden uten tilgang til prototypen v.2

Det ble videre undersøkt om det var feil med temperatursensoren på LED-hodet. Med hjelp av Anders R. Petersen ble LED-hodet demontert for å sjekke koblingen til temperatursensoren. I LED-hodet var det montert en kjøleribbe. Denne kjøleribben er overdimensjonert i forhold til resten av designet. Dette kan ha resultert i kortslutning mellom ulike loddepunkt på utsiden av kjøleribben. Gruppen oppdaget at det var utført en nødløsning, det hadde blitt smurt på termisk pasta for å heve kjøleribben over loddepunktene, men det var usikkert om det ville løse utfordringene rundt kjøleribben. Om den termiske pastaen ikke dekker nok av overflaten mellom kretskoret og kjøleribben vil det kunne oppstå kortslutninger som fører til feil avlesning på temperatursensor.

DAC

Det ble klart at DAC-en ikke fungerte når modulene koblet til TWI bussen skulle testes på prototype v.1. Når den ble feilsøkt med multimeter var det til enhver tid 0V på utgangen. Dette var en klar indikasjon på feil ved komponenten ettersom den automatisk skal stilles til $0,5 \cdot V_{CC} = 2,5V$ [29] etter å ha vært frakoblet strøm. Som en videre konsekvens av dette var det ingen spenningsforskyning til LED driveren.

Problemer i "Power"-nettverket

Gruppen mottok prototypen fulladet av Patrick C. Bösch, og fikk inntrykk av at alt fungerte som tenkt. Dette viste seg i senere tid å ikke stemme. Det ble avdekket flere problemer med "Power"-nettverket til prototype v.1 etter gruppen fikk tid til å se nærmere på den. Feilene begynte først å dukke opp når batteriet først var utladet. Da ble USB-C porten på prototype v.1 testet for lading som resulterte i at av- og påknappen flimret.

Gruppen valgte å oppsøke Patrick C. Bösch for å undersøke årsaken til feilen. Det ble ikke avdekket noen direkte feil ved kretsen annet enn oppførselen til knappen. Når feilsøkingen var ferdig og kretsen skulle kobles tilbake til batteriet, ble polene byttet om som medførte at en P-MOSFET ble ødelagt. P-MOSFET-en skulle beskytte kretsen mot feilkobling av batteriet, og at den ble ødelagt var en indikasjon på at den gjorde jobben sin. Likevel var kretsen mindre beskyttet for feiloppkobling som gjorde at gruppen måtte være ekstra forsiktige ved videre lading av prototype v.1. Siden problemet med flimringen ikke ble løst var det en bekymring rundt at Arduino-en ikke ville ha godt av hurtig endring av strømtilførsel. Derfor ble gruppen anbefalt å ikke lade batteriet via USB-C porten slik designet egentlig var laget for, men heller lade batteriet direkte via en spenningskilde.

Etter at P-MOSFET-en ble ødelagt ble oppførselen til "Power"-nettverket forverret. På dette punktet lyste av/på knappen uavhengig om den var skrudd av eller på, så lenge prototypen var koblet til enten batteriet eller spenningskilden.

6.3.4 Ulykken og videre konsekvenser

Fredag 5. mai ble det utført tester på prototypen for å sjekke de ulike funksjonene knyttet til TWI kommunikasjonen. Batteriet ble samtidig ladet når prototypen ikke var i bruk slik at den kunne brukes for å teste kode fra hjemmekontor over helgen.

Ca. 22:50 på kvelden begynner batteriet å frese, og den umiddelbare reaksjonen var å koble alt fra. Først blir ledningene til batteriet nappet ut, deretter ble ledningen fra PC-en fjernet. Når oppmerksomheten igjen ble vendt mot batteriet eksploderte det. Selve eksplosjonen opplevdes som voldsom, men den døde ut relativt fort. Når det ble bekreftet at ingen av de tilstede hadde påtatt seg fysiske skader ble fokuset flyttet over til slukking av den gjenværende brannen. Heldigvis var brannslukkerene lett tilgjengelig og slukningsarbeidet gikk fort. Videre ble NTNU sitt beredskapsnummer kontaket og alle tilstede beveget seg ut av rommet for å unngå å puste inn mer røyk.

Etterhvert dukket brannvesenet opp og luftet ut røyken slik at alle kunne gå inn og hente ut sine eiendeler. Senere dukket politiet opp for å forhøre seg om hva som hadde skjedd. Til slutt kom ambulansen som ønsket å ta med alle tilstedeværende ned på akutten for å sjekke om det

ble inhalert farlige gasser. Det ble tatt med et tilsvarende batteri som det som eksploderte slik at de ansatte på akutten kunne sjekke opp hos giftsentralen om det var behov for overvåkning over natten. Det ble klart at det ikke var nødvendig å bli over natten på sykesuset og alle fikk dra hjem etter omtrent 2 timer.

Lørdag morgen ble både veileder og instituttleder kontaktet og informert om hendelsen. Senere på dagen ble det holdt et møte med Dekan Ingrid Schjølberg og instituttleder Thomas Tybell. Alle som var tilstede under ulykken og resten av bachelor gruppen var invitert. Under dette møtet ble hele hendelsesforløpet gjenfortalt. Gruppen ble også oppfordret til å sette alt videre arbeid på prototype v.2 på is grunnet etteforskning internt på NTNU rundt hva som gikk galt med prototype v.1.

Figur 6.3: Bilde av LED-hodet med skader fra brannen. Fotokreditering: Anders Rønning Petersen v/Intitutt for Teknisk Kybernetikk.

Etter ulykken hadde gruppen ikke lenger tilgang på restene av styringsenheten til prototype v.1 ettersom denne ble konfiksert av NTNU. LED-hodet derimot ble ikke konfiskert og er fotografert for å illustrere skadene av brannen (Figur 6.3). Når denne sammenliknes opp mot bilde tatt før brannen, til sammenlikning er LED-hodet i ny tilstand illustrert i Figur 2.2a.

Konsekvenser av ulykken

Som en konsekvens av at prototype v.1 ble ødelagt var det ikke lenger mulig å teste koden. Dermed ble det umulig å teste systemet som en helhet på en fysisk styringsenhet. Likevel kunne mange av enkeltfunksjonene testets hver for seg. De delene som er dokumentert at fungerer hver for seg er ramset opp under.

- Skrive til CSV-fil
- Tid teller `getTime()`
- Avlesning på analog pinnene
- Visning av meny satt av skjerm og tastatur
- Avlesning fra tastatur

Deler som er udokumentert som en konsekvens av ulykken er:

- Temperatursensorer og ADC: Fordi temperatursensorene var koblet opp via en overflatemontert operasjons forsterker ble det ikke mulig å replikere resultatene som er beskrevet i seksjon 6.3.3.
- TWI kommunikasjon: Det var kun mulig å teste TWI funksjonene direkte på prototypen fordi alle komponentene som var koblet til Arduino-en sine TWI busser var overflatemonterte. Det ble observert at koden fungerte, med desverre ble det ikke dokumentert før ulykken.
- Knappene og alle LED-ene skulle brukes i sammenheng med hele systemet, og ble heller ikke dokumentert at fungerte før prototypen ble ødelagt.

Det ble forsøkt å sette sammen hele koden til systemet til prototype v.2, men det ble skrinlagt grunnet mangelen på en prototype. Sammensetningen av koden til hele systemet ble testet med kun tastatur, skjerm og minneutvidelsen tilkoblet Arduino-en. For å kunne gjøre dette måtte resten av variablene og funksjonene som innhenter data underveis deaktiveres for å unngå feiltrigging av feildeteksjon. Som forventet fungerte ikke systemet på første forsøk. Det ble gjennomført feilsøking, men det dukket opp store problemer med når det ikke var mulig å få oversikt over hvilke deler av systemet som trigget den uforutsigbare oppførselen til systemet. Gruppen bestemte at det var best å unngå sammenkobling av forsøkslogikken med menylogikken for å ikke ødelegge det som allerede fungerte.

6.4 Fremtidig arbeid

Denne seksjonen tar for seg potensielle utvidelser som kan implementeres for en senere versjon av prototypen. Gruppen har ikke hatt mulighet til å utforske disse utvidelsene i praksis.

6.4.1 Forslag til prototype v.3

Det har blitt oppdaget flere forbedringsmuligheter i utlegget for både prototype v.1 og v.2, som kan gjøre designet bedre tilpasset formålet.

Som tidligere diskutert var det en mulighet å koble skjermen til SPI-pinnene. Gruppen følte imidlertid at de ikke hadde tilstrekkelig med tid samt. manglende tilgang på prototype v.2 ble

det valgt å ikke koble skjermen til SPI-pinnene. Dette bør vurderes hvis det skal utvikles en prototype v.3 for å øke hastigheten på oppdatering til skjermen.

I tilegg bør komponentene koblet til pinne 9 og 10 på Arduino-en byttes om. Slik det er koblet opp på prototype v.1 og v.2 er den grønne knappen tilkoblet pinne 9, mens viftene til styringsenheten er koblet til pinne 10. Dette skaper en utfordring da det er ønskelig at den grønne knappen som starter forsøk skal aktivere et avbrudd som både starter forsøket og vekker mikrokontrolleren fra "powerDown-modus". Den enkleste måten å løse dette på er å koble knappen til pinne 10, som har mulighet for "Pin Change Interrupt", og koble viften til pinne 9 i stedet.

Når all programvare for prototypen var ferdigutviklet og testet, kan det også være en god idé å bruke ATmega2560 mikrokontrolleren fremfor hele Arduino Mega 2560 utviklingskortet. Dette vil gjøre det mulig å designe prototypen mer kompakt og mer brukervennlig når den skal benyttes av pasienter, samt. redusere kostnadene til et Arduino utviklingsbrett.

WiFi-modul

Selv om WiFi-modulen ble valgt og implementert i kretskortdesignet til styringsenheten, valgte gruppen å sløyfe dette. Det ville blant annet kreve utvikling av et nettbasert brukergrensesnitt, som faller utenfor rammen for denne bacheloroppgaven. Ettersom prototypen er ment å være bærbar, vil det også være nødvendig å programmere styringsenheten med mulighet til å koble seg til ulike WiFi-nettverk. Dette ble ansett som en modul som ikke var nødvendig å prioritere, siden prototypen fortsatt er i forskningsstadiet. Under forskningsstadiet er det ikke like stort behov for at brukeren eller forskeren må være koblet til nettverket. Dette kan begrunnes med at forskningen skjer under oppfølgning av forskerteam og mer kontrollerte omgivelser. Det ble av den grunn klart for gruppen at implementering av en slik modul ville være unødvendig å fokusere på, spesielt med tanke på forsinkelsene som oppstod i prosjektet.

Kapittel 7

Konklusjon

I denne bacheloroppgaven har gruppen utviklet et brukergrensesnitt for en medisinsk prototype for bruk i diabetes forskning. Gruppen har programmert en mikrokontroller til å styre en LED-matrise, overvåke temperatursensorer og implementert en batterivakt som overvåker batteristatusen for å sikre trygge og pålitelige forsøk. Ved hjelp av et tastatur og en skjerm settes innstillingene for forsøket. Etter hvert forsøk genererer systemet to CSV-filer som inneholder de registrerte temperaturverdiene, innstillingene satt av bruker og eventuelle feilmeldingene. Dette tilrettelegger for en grundig dataanalyse for helsepersonell etter fullført forsøk.

Selv om gruppen oppnådde betydelig fremgang i utviklingen av prototypen, er det fremdeles rom for forbedringer. Videre arbeid kan være å implementere SPI-kommunikasjon mellom skjermen og mikrokontrolleren, og endre tilkoblingen til de ulike pinnene på mønsterkortet. Videre kan det være hensiktsmessig å vurdere bruk av mikrokontrolleren ATmega2560 i stedet for hele utviklerkortet Arduino Mega 2560 for å oppnå et mer kompakt design.

Totalt sett har gruppen oppnådd målene som ble satt i forprosjektfasen. Desverre har uforutsette hendelser utenfor gruppens sin kontroll satt det avsluttende arbeidet på is. Likevel har det blitt utviklet et kretskort for prototypen v.2. Det nye kretskortet kan tas i bruk etter at prototype v.1 har blitt feilsøkt og godkjent. Dersom feilene som oppstod på prototypen v.1 og tilsvarende feil rettes opp på prototype v.2, bør det fungere når det eventuelt blir montert. I tillegg har det blitt laget programvare som oppfyller store deler av kravene til systemet som ble satt fra APT i oppstartsfasen. Det mangler fortsatt litt arbeid i sammensettingen av systemet, men det er utarbeidet et forslag til hvordan dette skal gjennomføres. Det ble ikke mulighet for å legge til alle forslagene APT kom med underveis i prosessen, men gruppen har lagt mest mulig til rette for at det skal være lett å redigere koden for å utvide systemet ved en senere anledning.

Som konklusjon har gruppen gjennom denne bacheloroppgaven demonstrert ferdigheter til å programmere og implementere et brukergrensesnitt for en medisinsk prototype. Gruppen har satt opp viktige funksjoner som temperaturovervåkning, batterivakt og datalagring. Videre har gruppen lagt vekt på brukervennlighet for å møte behovene til både pasienter og forskere. Med videreutvikling og forbedringer kan denne prototypen potensielt bidra til overvåkning av glukosesystemer for pasienter med diabetes. Det gjenstår imidlertid en betydelig mengde arbeid før videre testing av systemet med prototypen kan gjennomføres.

Referanser

[1] P. C. Bösch, «Device for improved insulin absorption in diabetes type 1,» 2021.

[2] Arduino®, «Arduino Mega 2560 Rev3 A000067-datasheet.pdf,» 2023, Hentet 2023-05-29. adresse: https://docs.arduino.cc/static/f3b00c25da7477414485f2206af54a2b/A000067-datasheet.pdf.

[3] Microchip Technology Inc, «ATmega640/V-1280/V-1281/V-2560/V-2561/V,» 2018, Hentet 2023-05-29. adresse: https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf.

[4] *Bitmapping definition and meaning | Collins English Dictionary*, i Hentet 2023-05-29. adresse: https://www.collinsdictionary.com/dictionary/english/bitmapping.

[5] P. Semiconductors, «THE I2C-BUS SPECIFICATION VERSION 2.1,» 2000, Hentet 2023-05-29. adresse: https://www.csd.uoc.gr/~hy428/reading/i2c_spec.pdf.

[6] Bjørn B. Larsen, *Tilstandsmaskin*, i *Store Norske Leksikon*, Hentet 2023-05-29, 2021. adresse: https://snl.no/tilstandsmaskin.

[7] World Health Organization. «Diabetes,» Diabetes. Hentet 2023-05-29. (2023), adresse: https://www.who.int/news-room/fact-sheets/detail/diabetes.

[8] Diabetesforbundet. «Insulin,» Insulin. Hentet 2023-05-29. (2023), adresse: https://www.diabetes.no/diabetes-type-1/behandling/insulin/.

[9] Artificial Pancreas Trondheim. «Artificial Pancreas Trondheim.» Hentet 2023-05-29. (2018), adresse: https://www.apt-norway.com/.

[10] Norges Helseinstitutt. «Bukspyttkjertelen.» Hentet 2023-05-29. (2020), adresse: https://nhi.no/kroppen-var/organer/bukspyttkjertelen/.

[11] Helsenorge. «Diabetes type 1 - Helsenorge.» Hentet 2023-05-29. (2018), adresse: https://www.helsenorge.no/sykdom/diabetes/diabetes-type-1/.

[12] Felleskatalogen. «Diabetes type 1.» Hentet 2023-05-29. (2023), adresse: https://www.felleskatalogen.no/medisin/sykdom/diabetes-type-1.

[13] T. Heise, T. R. Pieber, T. Danne, L. Erichsen og H. Haahr, «A Pooled Analysis of Clinical Pharmacology Trials Investigating the Pharmacokinetic and Pharmacodynamic Characteristics of Fast-Acting Insulin Aspart in Adults with Type 1 Diabetes,» *Clinical Pharmacokinetics*, årg. 56, nr. 5, s. 551–559, 1. mai 2017, Hentet 2023-05-29, ISSN: 1179-1926. DOI: 10.1007/s40262-017-0514-8. adresse: https://doi.org/10.1007/s40262-017-0514-8.

[14] Trygve Holtebekk, *Infrarød stråling*, i 2021. adresse: https://snl.no/infrar%C3%B8d_str%C3%A5ling.

[15] Andrew M. Smith, Micheal C. Mancini og Shuming Nie, «Bioimaging: second window for in vivo imaging,» 2023, Hentet 2023-05-29. DOI: 10.1038/nnano.2009.326. adresse: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2862008/.

[16] Altium, *Altium*, Hentet 2023-05-29, 2023. adresse: https://www.altium.com/.

[17] Altium, *Altium Designer*, Hentet 2023-05-29, 2023. adresse: https://www.altium.com/altium-designer.

[18] Microsoft, *Visual Studio Code - Code Editing. Redefined*, 2018. adresse: https://code.visualstudio.com/.

[19] PlatformIO, *A professional collaborative platform for embedded development*, 2018. adresse: https://platformio.org/.

[20] Git, *Git*, 2018. adresse: https://git-scm.com/.

[21] GitHub, *GitHub: Let's build from here · GitHub*, 2018. adresse: https://github.com/.

[22] NTNU. «Solidworks - Kunnskapsbasen - NTNU.» Hentet 2023-05-29. (2023), adresse: https://i.ntnu.no/wiki/-/wiki/Norsk/Solidworks.

[23] M. Dwamena. «do-all-3d-printers-use-stl-files.» Hentet 2023-05-29. (), adresse: https://3dprinterly.com/do-all-3d-printers-use-stl-files/.

[24] cadinterop. «CAD Data Interoperability around SolidWorks by Dassault Systemes.» Hentet 2023-05-29. (), adresse: https://www.cadinterop.com/en/formats/cad-systems/solidworks.html.

[25] U. BV. «About Ultimaker.» Hentet 2023-05-29. (), adresse: https://press.ultimaker.com/about/.

[26] UltiMaker. «UltiMaker S5: Expand your 3D printing ambitions.» Hentet 2023-05-29. (), adresse: https://ultimaker.com/3d-printers/s-series/ultimaker-s5/.

[27] Arduino. «A Guide to Arduino the I2C Protocol (Two Wire) | Arduino Documentation.» Hentet 2023-05-29. (2021), adresse: https://docs.arduino.cc/learn/communication/wire.

[28] Elliot Williams, *Make: AVR programming*. Make: Community, 2014, ISBN: 978-1-4493-5578-4.

[29] Farnell, «12-Bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6,» 2009, Hentet 2023-05-29. adresse: https://www.farnell.com/datasheets/1470577.pdf.

[30] RS-online, «EAO – Your Expert Partner for Human Machine Interfaces,» 2009, Hentet 2023-05-29. adresse: https://docs.rs-online.com/f361/0900766b80de0aff.pdf.

[31] tutorialspoint. «Embedded Systems - Interrupts.» Hentet 2023-05-29. (), adresse: https://www.tutorialspoint.com/embedded_systems/es_interrupts.htm.

[32] E. Tutorials. «Analogue to Digital Converter (ADC) Basics.» Hentet 2023-05-29. (2023), adresse: https://www.electronics-tutorials.ws/combination/analogue-to-digital-converter.html.

[33] Arduino. «Basics og PWM (Pulse Width Modulation).» Hentet 2023-05-29. (2023), adresse: https://docs.arduino.cc/learn/microcontrollers/analog-output.

[34] T. Nätt og E. Rossen, *C – programmeringsspråk – Store norske leksikon*, i Hentet 2023-05-29, 2023. adresse: https://snl.no/C_-_programmeringsspr%5C%C3%C3%A5%5C%A5k.

[35] E. Rossen og S. Almås, *C++ – Store norske leksikon*, i Hentet 2023-05-29, 2023. adresse: https://snl.no/C++.

[36] Amazon. «LCD Display Module, 1.8 inch 128x160 TFT LCD Display Module 4-Wire TFT LCD Screen ST7735 TFT Display Module - -.» Hentet 2023-05-29. (2023), adresse: https://www.amazon.com/Display-Module-128x160-4-Wire-Screen/dp/B0983P263K.

[37] I. f. E. S. (NTNU. «Elektronikk og prototypelaboratoriet - Kunnskapsbasen - NTNU.» Hentet 2023-05-29. (), adresse: https://i.ntnu.no/wiki/-/wiki/Norsk/Elektronikk+og+prototypelaboratoriet.

[38] Adnbr. «Counting milliseconds adnbr.» Hentet 2023-05-29. (2012), adresse: https://adnbr.co.uk/articles/counting-milliseconds.

[39] AEQ-WEB. «PT1000 Converter for Arduino.» Hentet 2023-05-29. (2023), adresse: https://www.aeq-web.com/pt1000-temperature-sensor-arduino-lm358-messwandler/?lang=en.

[40] E. O. Gmb, «EOLS-660-496 SMD LED EPIGAP Optronic,» 2017, Hentet 2023-05-29. adresse: http://www.amstechnologies-webshop.com/media/pdf/c4/28/3b/EOLS-660-496-SMD-LED-EPIGAP-Optronic-Datasheet.pdf.

[41] ICHaus, «iC-HG 3 A LASER SWITCH RevC2,» Hentet 2023-05-29. adresse: https://www.ichaus.de/upload/pdf/HG_datasheet_C2en.pdf.

[42] I. Maxim Integrated Products, «DS3231 Extremely Accurate I2C-Integrated RTC/TCXO/Crystal,» 2015, Hentet 2023-05-29. adresse: https://www.analog.com/media/en/technical-documentation/data-sheets/DS3231.pdf.

[43] T. I. Incorporated, «bq27441-G1 Technical Reference Manual (Rev. A),» 2013, Hentet 2023-05-29. adresse: https://www.ti.com/lit/ug/sluuac9a/sluuac9a.pdf?ts=1682700674427.

[44] V. M. GmbH, «CR2032_eng_TDS.pdf,» 2004, Hentet 2023-05-29. adresse: https://www.elfadistrelec.no/Web/Downloads/_t/ds/CR2032_eng_TDS.pdf.

[45] Digi-Key Electronics. «Micro SD Card Breakout Board Tutorial Datasheet by Lattice Semiconductor Corporation | Digi-Key Electronics.» Hentet 2023-05-29. (), adresse: https://www.digikey.cz/htmldatasheets/production/1800211/0/0/1/micro-sd-card-breakout-board-tutorial.html.

[46] E. Distrelec, «MicroSD card breakout board, Adafruit,» Hentet 2023-05-29. adresse: https://www.digikey.cz/htmldatasheets/production/1800211/0/0/1/micro-sd-card-breakout-board-tutorial.html.

[47] Arduino. «SD - Arduino Reference.» Hentet 2023-05-29. (2023), adresse: https://www.arduino.cc/reference/en/libraries/sd/.

[48] A. Instructables. «Converting Images to Flash Memory Icons/images for TFT (without SD Card) : 9 Steps - Instructables.» (), adresse: https://www.instructables.com/Converting-Images-to-Flash-Memory-Iconsimages-for-/.

[49] RS-online. «ECO.12150.06 | EOZ IP40 12 Key Keypad | RS.» Hentet 2023-05-29. (), adresse: https://no.rs-online.com/web/p/keypads/0146014.

[50] «New 4 X3 12 Key Matrix Array Membrane Switch Keypad Keyboard For Arduino | Fruugo NO.» Hentet 2023-05-29. (), adresse: https://www.fruugonorge.com/new-4-x3-12-key-matrix-array-membrane-switch-keypad-keyboard-for-arduino/p-108740346-229487829?language=en&ac=ProductCasterAPI&asc=pmax&gclid=CjwKCAjwpayjBhAnEiwA-7ena1VGTw9DeYko2VFsfEI1aBXAlOpqZzceaYoZejLR2K7ohII_NY8dZhoCSEwQAvD_BwE.

[51] Elektorstore. «0.96"OLED Display for Arduino (128x64) | Elektor.» Hentet 2023-05-29. (2023), adresse: https://www.elektor.com/0-96-oled-display-for-arduino-128x64.

[52] Shopify, *1.8"Color TFT LCD display with MicroSD Card Breakout - ST7735R*, Hentet 2023-05-29, 2023. adresse: https://cdn.shopify.com/s/files/1/0573/8122/4630/products/18-inch-color-tft-lcd-display-w-microsd-card-breakout-st7735r-img1_79fad123-99b6-4625-9c75-60d8bb04212e_800x.jpg?v=1680814984.

[53] E. Systems, «ESPRESSIF SMART CONNECTIVITY PLATFORM: ESP8266,» 2013, Hentet 2023-05-29. adresse: https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf.

[54] M. T. Inc. «Microcontrollers (MCUs) | Microchip Technology.» Hentet 2023-05-29. (), adresse: https://www.microchip.com/en-us/products/microcontrollers-and-microprocessors.

[55] «Warping - Simplify3D Software.» (2023), adresse: https://www.simplify3d.com/resources/print-quality-troubleshooting/warping/.

[56] Mekanisk Verksted. «Mekanisk verksted - IGP - NTNU.» Hentet 2023-05-29. (2021), adresse: https://www.ntnu.no/igp/lab/mekanisk.

[57] Arduino. «Licensing for products based on Arduino – Arduino Help Center.» Hentet 2023-05-29. (), adresse: https://support.arduino.cc/hc/en-us/articles/4415094490770-Licensing-for-products-based-on-Arduino.

[58] C. A. «Keypad/LICENSE at master · Chris–A/Keypad,» GNU General Public License v3.0. Hentet 2023-05-29. (2023), adresse: https://github.com/Chris--A/Keypad/blob/master/LICENSE.

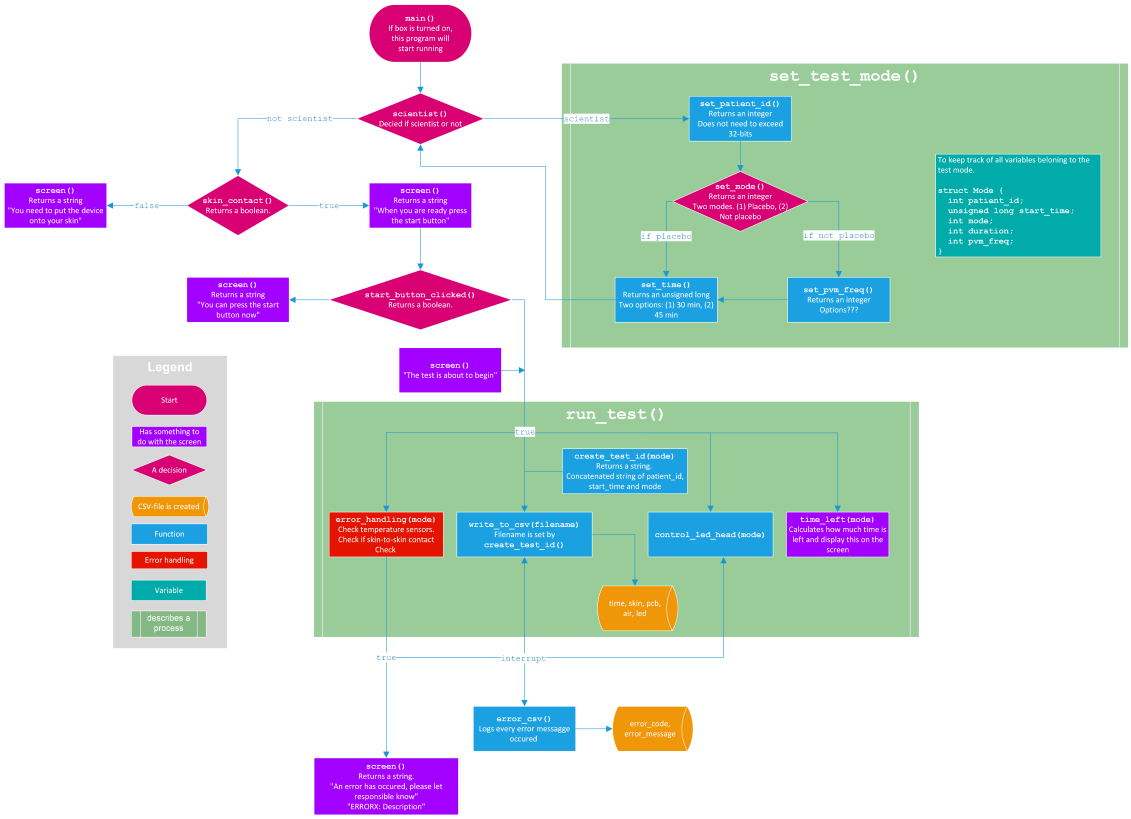
[59] Adafruit. «Adafruit-GFX-Library/license.txt at master · adafruit/Adafruit-GFX-Library.» Hentet 2023-05-29. (2023), adresse: https://github.com/adafruit/Adafruit-GFX-Library/blob/master/license.txt.

[60] Adafruit. «adafruit/Adafruit-ST7735-Library: This is a library for the Adafruit 1.8" SPI display http://www.adafruit.com/products/358 and http://www.adafruit.com/products/618.» Hentet 2023-05-29. (2023), adresse: https://github.com/adafruit/Adafruit-ST7735-Library.

[61] Arduino. «SD/LICENSE.txt at master · arduino-libraries/SD.» Hentet 2023-05-29. (2023), adresse: https://github.com/arduino-libraries/SD/blob/master/LICENSE.txt.

Vedlegg A

Flytskjema for koden


main()
If box is turned on, this program will start running
scientist()
Decied if scientist or not
not scientist
scientist
set_test_mode()
set_patient_id()
Returns an integer
Does not need to exceed 32 bits
set_mode()
Returns an integer
Two modes. (1) Placebo, (2) Not placebo
if placebo
if not placebo
set_time()
Returns an unsigned long
Two options: (1) 30 min, (2) 45 min
set_pvm_freq()
Returns an integer
Options???
To keep track of all variables belonging to the test mode.
struct Mode {
int patiant_id;
unsigned long start_time;
int mode;
int duration;
int pvm_freq;
}
screen()
Returns a string
"You need to put the device onto your skin"
false
skin_contact()
Returns a boolean.
true
screen()
Returns a string
"When you are ready press the start button"
screen()
Returns a string
"You can press the start button now"
start_button_clicked()
Returns a boolean.
screen()
"The test is about to begin"
run_test()
true
create_test_id(mode)
Returns a string.
Concatenated string of patient_id, start_time and mode
error_handling(mode)
Check temperature sensors.
Check if skin-to-skin contact
Check
write_to_csv(filename)
Filename is set by create_test_id()
control_led_head(mode)
time_left(mode)
Calculates how much time is left and display this on the screen
time, skin, pcb, air, led
true
interrupt
error_csv()
Logs every error messagge occured
error_code, error_message
screen()
Returns a string;
"An error has occured, please let responsible know"
"ERRORX: Description"
Legend
Start
Has something to do with the screen
A decision
CSV-file is created
Function
Error handling
Variable
describes a process

Vedlegg B

Bill of Material (BOM)

What?	Total price	Valuta	Supplier	Manufacturer	Link	# of compo	Available on website?	Does Patrick think we	Unit price?	Product number	Part	Where?
BATTERY BABYSITTER LIPO BATTERY	222.30	NOK	Digi-Key	SparkFun	https://www.digikey.no/products/no?keywords=Battery%20B	1	Yes it's availabe	y	222.3	PRT-13777	Control Electronics	Charging Battery
Memory Extension Arduino, MicroSD card breakout board+	77.55	NOK	Digi-Key	Adafruit	https://www.digikey.no/product-detail/no/adafruit-industries	1	Yes it's availabe	y	77.55	254	Control Electronics	Arduino
PowerBoost 1000 Basic - 5V USB Boost @ 1000mA from 1.8V+	154.58	NOK	Digi-Key	Adafruit	https://www.digikey.no/product-detail/no/adafruit-industries	1	Yes it's availabe	y	154.58	2030	Control Electronics	3,7V to 5V
Battery Holder, Coin Cell - 20mm x 1, SMD	8.78	NOK	Farnell	Multicomp	https://no.farnell.com/multicomp/ch7410-2032lf/battery-holc	1	Yes it's availabe	y	8.78	CH7410-2032LF	Control Electronics	RTC
Circular Connector, 1B Series, Panel Mount Receptacle, 7 Contacts, Solder Socket, Push-Pull	339.71	NOK	Farnell	Lemo	https://no.farnell.com/lemo/egg-1b-307-cll/socket-panel-mou	1	Yes it's availabe	y	339.71	EGG.1B.307.CLL	Control Electronics	Connect LEDs
Ferrite Bead, 5 A, 78 ohm	9.06	NOK	Farnell	Fair-Rite	https://no.farnell.com/fair-rite/2773021447/ferrite-bead-120	2	Yes it's availabe	y	4.53	2773021447	Control Electronics	12v Smoothing
LED Panel Mount Indicator, Green, 2 VDC, 8 mm, 20 mA, 8 mcd, IP67	42.58	NOK	Farnell	APEM	https://no.farnell.com/apem/q8f6cxxg02e/led-indicator-panel	1	Yes it's availabe	y	42.58	Q8F6CXXG02E	Control Electronics	Indicator LED
LED Panel Mount Indicator, Red, 2 VDC, 8 mm, 20 mA, 12 mcd, IP67	43.42	NOK	Farnell	APEM	https://no.farnell.com/apem/q8f6bxxr02e/led-indicator-panel	1	Yes it's availabe	y	43.42	Q8F6BXXR02E	Control Electronics	Indicator LED
Operational Amplifier, 2 Amplifier, 1 MHz, 0.1 V/µs, 3V to 32V, ± 1.5V to ± 16V, SOIC, 8 Pins	30.99	NOK	Farnell	TI	https://no.farnell.com/texas-instruments/lm358mx-nopb/op-	3	Yes it's availabe	y	10.33	LM358MX/NOPB	Control Electronics	PT1000
Pin Header, Board-to-Board, 2.54 mm, 1 Rows, 25 Contacts, Through Hole Straight	14.23	NOK	Farnell	Molex	https://no.farnell.com/molex/22-28-4250/connector-header-2	1	Yes it's availabe	y	14.23	22-28-4250	Control Electronics	Connect Arduino
Power MOSFET, P Channel, 20 V, 18 A, 0.0073 ohm, PowerPAK 1212, Surface Mount	38.44	NOK	Farnell	Vishay	https://no.farnell.com/vishay/sis407adn-t1-ge3/mosfet-p-ch-2	4	Yes it's availabe	y	9.61	SIS407ADN-T1-GE3	Control Electronics	Current Protection Battery
PTC Thermistor, 1 kohm, SMD, PTS Series	27.61	NOK	Farnell	Vishay	https://no.farnell.com/vishay/pts120601b1k00p100/thermistc	1	Yes it's availabe	y	27.61	PTS120601B1K00P100	Control Electronics	Temp PCB
Schottky Rectifier, 30 V, 1 A, Single, SMD, 2 Pins, 500 mV	7.56	NOK	Farnell	Taiwan Semiconductor	https://no.farnell.com/taiwan-semiconductor/ss13l-r3/diode-	1	Yes it's availabe	y	7.56	SS13L R3	Control Electronics	Fly Back Fans and Buzzer
SMD Chip Resistor, 1 kohm, ± 0.01%, 100 mW, 0805 [2012 Metric], Thin Film, Precision	59.13	NOK	Farnell	Multicomp	https://no.farnell.com/multicomp/mctf0805ttx1001/res-1k-0-	3	Yes it's availabe	y	19.71	MCTF0805TTX1001	Control Electronics	PT1000
SMD Chip Resistor, 1 ohm, ± 1%, 750 mW, 1206 [3216 Metric], Thick Film, Pulse Proof, High Power	6.70	NOK	Farnell	Vishay	https://no.farnell.com/vishay/crcw12061r00fkeahp/res-1r-1-0	2	Yes it's availabe	y	3.35	CRCW12061R00FKEAHP	Control Electronics	IC-HG
SMD Chip Resistor, 10 kohm, ± 0.01%, 62.5 mW, 0603 [1608 Metric], Thin Film, Precision	236.52	NOK	Farnell	Multicomp	https://no.farnell.com/multicomp/mctf0603ttx1002/res-10k-0	12	Yes it's availabe	y	19.71	MCTF0603TTX1002	Control Electronics	PT1000 / ID Resistor
SMD Chip Resistor, 270 ohm, ± 1%, 125 mW, 0805 [2012 Metric], Thick Film, General Purpose	0.16	NOK	Farnell	Multicomp	https://no.farnell.com/multicomp/mcwr08x2700ftl/res-270r-1	2	Yes it's availabe	y	0.0798	MCWR08X2700FTL	Control Electronics	MOSFET to Arduino
SMD Chip Resistor, Ceramic, 100 kohm, ± 1%, 100 mW, 0603 [1608 Metric], Thick Film	0.56	NOK	Farnell	Multicomp	https://no.farnell.com/multicomp/mcmr06x1003ftl/res-100k-	6	Yes it's availabe	y	0.093	MCMR06X1003FTL	Control Electronics	MOSFET, 3.3V
SMD Multilayer Ceramic Capacitor, 0.1 µF, 50 V, 0603 [1608 Metric], ± 5%, X7R, VJ Series	1.70	NOK	Farnell	Vishay	https://no.farnell.com/vishay/vj0603y104jxacw1bc/cap-0-1-f-	3	Yes it's availabe	y	0.566	VJ0603Y104JXACW1BC	Control Electronics	IC-HG, LED, Power source, DAC
SMD Multilayer Ceramic Capacitor, 10000 pF, 50 V, 0603 [1608 Metric], ± 5%, X7R, VJ Series	1.01	NOK	Farnell	Vishay	https://no.farnell.com/vishay/vj0603y103jxacw1bc/cap-0-01-f	2	Yes it's availabe	y	0.506	VJ0603Y103JXACW1BC	Control Electronics	IC-HG, LED, Power source
Surface Mount Tantalum Capacitor, MnO2, 100 µF, 16 V, 2412 [6032 Metric], T495 Series, ± 10%	82.52	NOK	Farnell	Kemet	https://no.farnell.com/kemet/t495c107k016ate200/cap-100-f	4	Yes it's availabe	y	20.63	T495C107K016ATE200	Control Electronics	3.7V to 12V
Surface Mount Tantalum Capacitor, MnO2, 220 µF, 10 V, 2917 [7343 Metric], T495 Series, ± 10%	36.24	NOK	Farnell	Kemet	https://no.farnell.com/kemet/t495x227k010ate100/cap-220-f	1	Yes it's availabe	y	36.24	T495X227K010ATE100	Control Electronics	Battery Stabiliser
Wire-To-Board Terminal Block, 5 mm, 2 Ways, 24 AWG, 12 AWG, 2.5 mm², Screw	37.60	NOK	Farnell	Phoenix Contact	https://no.farnell.com/phoenix-contact/mkds-3-2/terminal-bl	4	Yes it's availabe	y	9.4	MKDS 3/ 2	Control Electronics	Connect Battery and other I/O
Power MOSFET, N Channel, 30 V, 1.9 A, 0.077 ohm, SOT-23, Surface Mount	111.00	NOK	Farnell	Nexperia	https://no.farnell.com/nexperia/bsh108-215/mosfet-n-ch-30v	5	Yes it's availabe	y	22.2	BSH108,215	Control Electronics	LED and Electronics Fan
LED Panel Mount Indicator, Chrome Plated, Green, 2 VDC, 8 mm, 20 mA, 20 mcd, Not Rated	33.93	NOK	Farnell	Mentor	https://no.farnell.com/mentor/2664-8081/led-indicator-5mm-	1	Yes it's availabe	y	33.93	2664.8081	Control Electroincs	
Power MOSFET, N Channel, 30 V, 2.7 A, 0.08 ohm, SOT-23, Surface Mount	21.65	NOK	Farnell	Infineon	https://no.farnell.com/infineon/irlml2030trpbf/mosfet-n-ch-3	5	Yes it's availabe	y	4.33	1791568	Control Electronics	Audio Alarm, LED and Electronics Fan
Non-Isolated DC/DC Converters 12W 3.3/5V	373.18	NOK	Mouser	TI	https://no.mouser.com/ProductDetail/Texas-Instruments/PTN	2	Yes it's availabe	y	186.59	PTN04050CAZT	Control Electronics	3.7V to 12V
Amphenol ICC, Right Angle, PCB Mount, Socket Type C 3.1 IP67 USB Connector	191.51	NOK	RS Online	Amphenol	https://no.rs-online.com/web/p/usb-connectors/1245632	1	Yes it's availabe	y	191.51	MUSBR-M1C1-30	Control Electronics	Charging Battery
Amphenol ICC, Straight, Through Hole, Socket Type B 2.0 USB Connector	156.33	NOK	RS Online	Arduino	https://no.rs-online.com/web/p/usb-connectors/7255989	1	Yes it's availabe	y	156.33	MUSBB55104	Control Electronics	Arduino
Binder Panel Mount Connector, 8 Contacts, Miniature Connector	142.63	NOK	RS Online	Binder	https://no.rs-online.com/web/p/industrial-circular-connectors	1	Yes it's availabe	y	142.63	99-9128-00-08	Control Electronics	Socket Sensors
Binder, 720 Female Dust Cap IP67 Rated	34.67	NOK	RS Online	Binder	https://no.rs-online.com/web/p/circular-connector-dust-caps/	1	Yes it's availabe	y	34.67	08-2567-000-000	Control Electronics	Connector Sensors
Panasonic CR2032 Button Battery, 3V, 20mm Diameter	0.00	NOK	RS Online	Panasonic	https://no.rs-online.com/web/p/button-batteries/4574757	1	Yes it's availabe	y	0	CR-2032/BN	Control Electronics	RTC
Resistor to Warning LED, Vishay 68Ω, 0805 (2012M) Thick Film SMD Resistor ±1% 0.5W	7.28	NOK	RS Online	Vishay	https://no.rs-online.com/web/p/surface-mount-fixed-resistors	25	Yes it's availabe	y	0.291	CRCW080568R0FKEAHP	Control Electronics	Signal LEDs
TDK, 1206 (3216M) 47µF Multilayer Ceramic Capacitor MLCC 25V dc ±20%	129.74	NOK	RS Online	TDK	https://no.rs-online.com/web/p/mlccs-multilayer-ceramic-cap	10	Yes it's availabe	y	12.974	C3216X5R1E476M160AC	Control Electronics	3.7V to 12V
Vishay 10µF MnO2 Tantalum Capacitor 25V dc, TR3 Series	46.43	NOK	RS Online	Vishay	https://no.rs-online.com/web/p/tantalum-capacitors/684513(	5	Yes it's availabe	y	9.286	TR3C106K025C0600	Control Electronics	DAC, 3.3V
RS PRO Illuminated Momentary Push Button Switch, Panel Mount, SPDT, 16mm Cutout, Blue LED, 2!	168.98	NOK	RS Online	RS Pro	https://no.rs-online.com/web/p/push-button-switches/81184	1	Yes it's availabe	y	168.98	811-8437	Control Electronics	Start Measurement
EOZ IP40 12 Key Keypad	82.48	NOK	RS Online	EOZ	https://no.rs-online.com/web/p/keypads/0146014?cm_mmc=	1	Yes it's availabe	y	82.48	146-014	Control Electronics	Front panel
TDK, 0805 (2012M) 22µF Multilayer Ceramic Capacitor MLCC 16V dc ±20% , SMD C2012X5R1C226M(	67.40	NOK	RS Online	TDK	https://no.rs-online.com/web/p/mlccs-multilayer-ceramic-cap	20	Yes it's availabe	y	3.37	915-9287	Control Electronics	IC-HG, LED, Power source, 3.3V
Memory Extension Arduino, MicroSD card breakout board+	75.00	NOK	Adafruit	Adafruit	https://www.adafruit.com/product/254	1	Yes it's availabe	y	75	254	Control Electronics	Arduino
	3121.15											

Vedlegg C

Power Consumption

Calculation Power Consumption NIR-Device

	What	I(av)[mA]	I(max)[mA]	No. of Dev	No. of Dev(810nm)	No. of Dev(660nm)	I(tot_max)[mA]	I(tot_aver)[mA]	I(tot_810nm)[mA]	I(tot_660nm)[mA]
12V	NIR LED	350.00	500.00	6.00	4.00	1.00	3000.00	2100.00	1400.00	350.00
	Arduino	100.00	200.00	1.00	1.00	1.00	200.00	100.00	100.00	100.00
						I(tot_max)12V[mA]	3200.00	2200.00	1500.00	450.00
5V	Sunon Fan	112.00	112.00	2.00	2.00	2.00	224.00	224.00	224.00	224.00
	IC-HG	300.00	750.00	1.00	0.67	0.17	750.00	300.00	200.00	50.00
	MicroSD card breakout board+	100.00	150.00	1.00	1.00	1.00	150.00	100.00	100.00	100.00
	Buzzer	1.00	60.00	1.00	1.00	1.00	60.00	1.00	1.00	1.00
	DAC (from Arduino)	0.20	0.40	1.00	1.00	1.00	0.40	0.20	0.20	0.20
	Voltage ref PT1000 (Arduino)	0.30	0.30	4.00	4.00	4.00	1.20	1.20	1.20	1.20
	LED in Power Switch	20.00	20.00	1.00	1.00	1.00	20.00	20.00	20.00	20.00
						I(tot_max)5V[mA]	1325.60	741.40	641.40	491.40
3.3V	Warning LED (Head)	20.00	20.00	1.00	1.00	1.00	20.00	20.00	20.00	20.00
	Real Time Clock	0.20	0.20	1.00	1.00	1.00	0.20	0.20	0.20	0.20
	Panel LED red	10.00	20.00	1.00	1.00	1.00	20.00	10.00	10.00	10.00
	Panel LED green	10.00	20.00	1.00	1.00	1.00	20.00	10.00	10.00	10.00
						I(tot_max)3.3V[mA]	160.20	120.20	120.20	120.20

Components to be added to v.2:

	What	I(av)[mA]	I(max)[mA]	No. of Dev	
	Wifi modul:				
3.3V	ESP8266-01	80	100	1	https://www.microchip.ua/wireless/esp01.pdf
	Display:				
3.3V/5V	1.8 tft spi 128x160 v1.1	85	100	1	https://www.buydisplay.com/download/manual/ER-TFTM018-2_Datasheet.pdf
	Keypad:				
5V	EOZ IP40 12 Key Keypad	10	20	1	https://docs.rs-online.com/f361/0900766b80de0aff.pdf

		U(out)[V]	I(out_max)[A]	I(out_aver)[A]	I(out_810nm)[A]	I(out_660nm)[A]	n	I(3.7_max)[A]	I(3.7_aver)[A]	I(3.7_810nm)[A]	I(3.7_660nm)[A]
I(tot_max)3.7V	3.7V -> 12V	12.00	3.20	2.20	1.50	0.45	0.87	11.93	8.20	5.59	1.68
	3.7V -> 5V	5.00	1.33	0.74	0.64	0.49	0.94	1.91	1.07	0.92	0.71
	3.7V -> 3.3V	3.30	0.16	0.12	0.12	0.12	0.61	0.23	0.18	0.18	0.18
							I(tot_max)[A]	14.07			
								I(tot_av)[A]	9.44		
									I(tot_810nm)[A]	6.69	
										I(tot_660nm)[A]	2.56
							P(tot_max)[W]	52.06			
								P(tot_av)[W]	34.94		
									P(tot_810nm)[W]	24.75	
										P(tot_660nm)[W]	9.47

	mAh	C	I(max)[A]
Battery	8200	60.00	492.00
Battery Bank	10000		
Battery(tot)	18200		

Usage	30min/measurement	P(max)[Wh]	26.03	P(av)[Wh]	17.47	P(810nm)[Wh]	12.38	P(660nm)[Wh]	4.74
		C(max)[mAh]	7034.55	C(av)[mAh]	4721.45	C(810nm)[mAh]	3344.81	C(660nm)[mAh]	1279.86
	45min/measurement	P(max)[Wh]	39.04	P(av)[Wh]	26.20	P(810nm)[Wh]	18.56	P(660nm)[Wh]	7.10
		C(max)[mAh]	10551.82	C(av)[mAh]	7082.17	C(810nm)[mAh]	5017.22	C(660nm)[mAh]	1919.79

		No. of Meas (max)	No. of Meas (aver)	No. of Meas (810nm)	No. of Meas (660nm)
30min	Battery	1.17	1.74	2.45	6.41
	Battery with Battery Bank	2.59	3.85	5.44	14.22
45min	Battery	0.78	1.16	1.63	4.27
	Battery with Battery Bank	1.72	2.57	3.63	9.48

All values are for CW-mode of LEDs

Calculations made by: Patrick C. Bösch
Edited by: Vilma A. Steen

Vedlegg D

Pinner brukt på Arduino Mega 2560

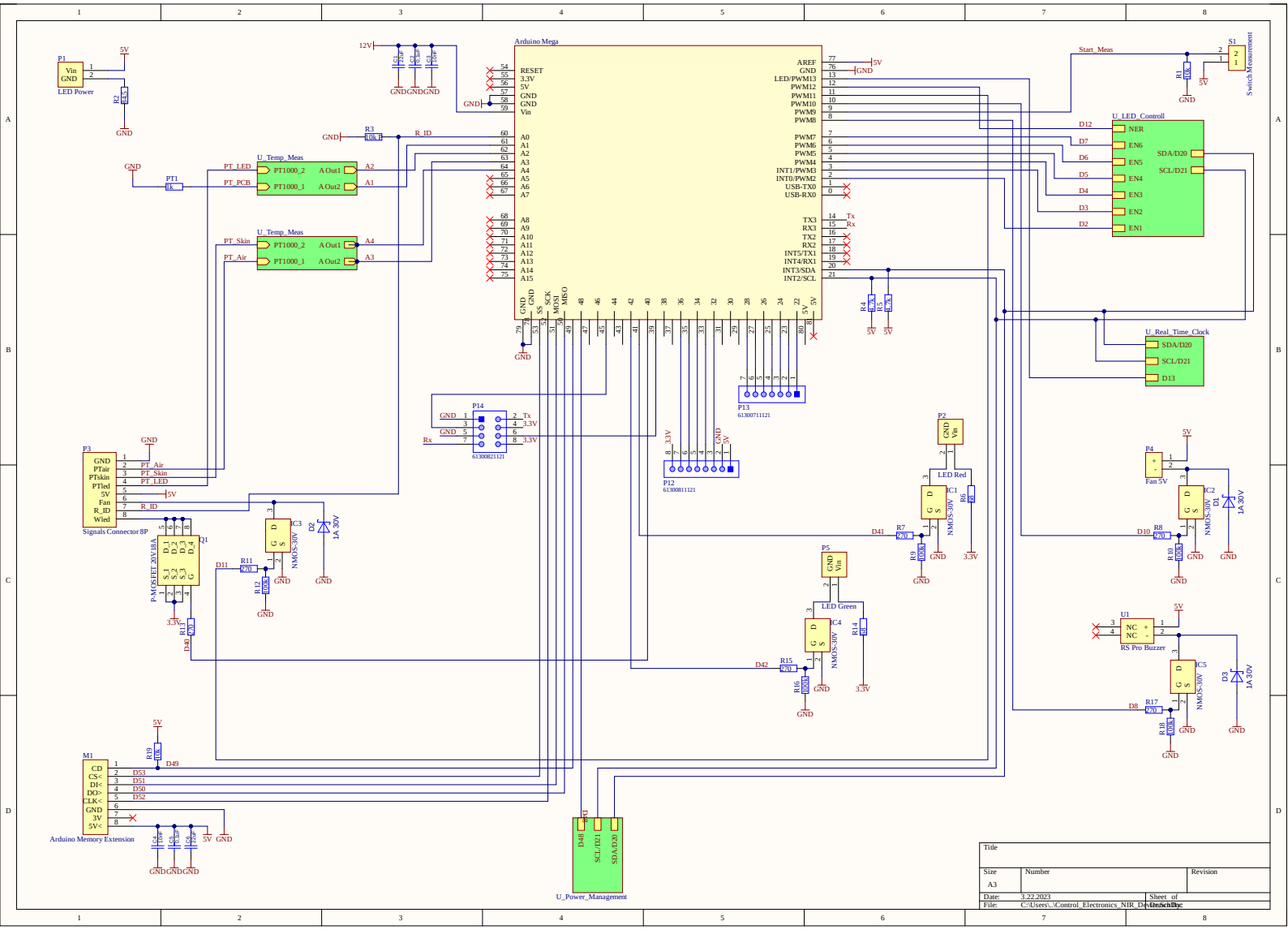
Oversikt alle pinner på Arduinio Mega 2560:
https://docs.arduino.cc/hacking/hardware/PinMapping2560

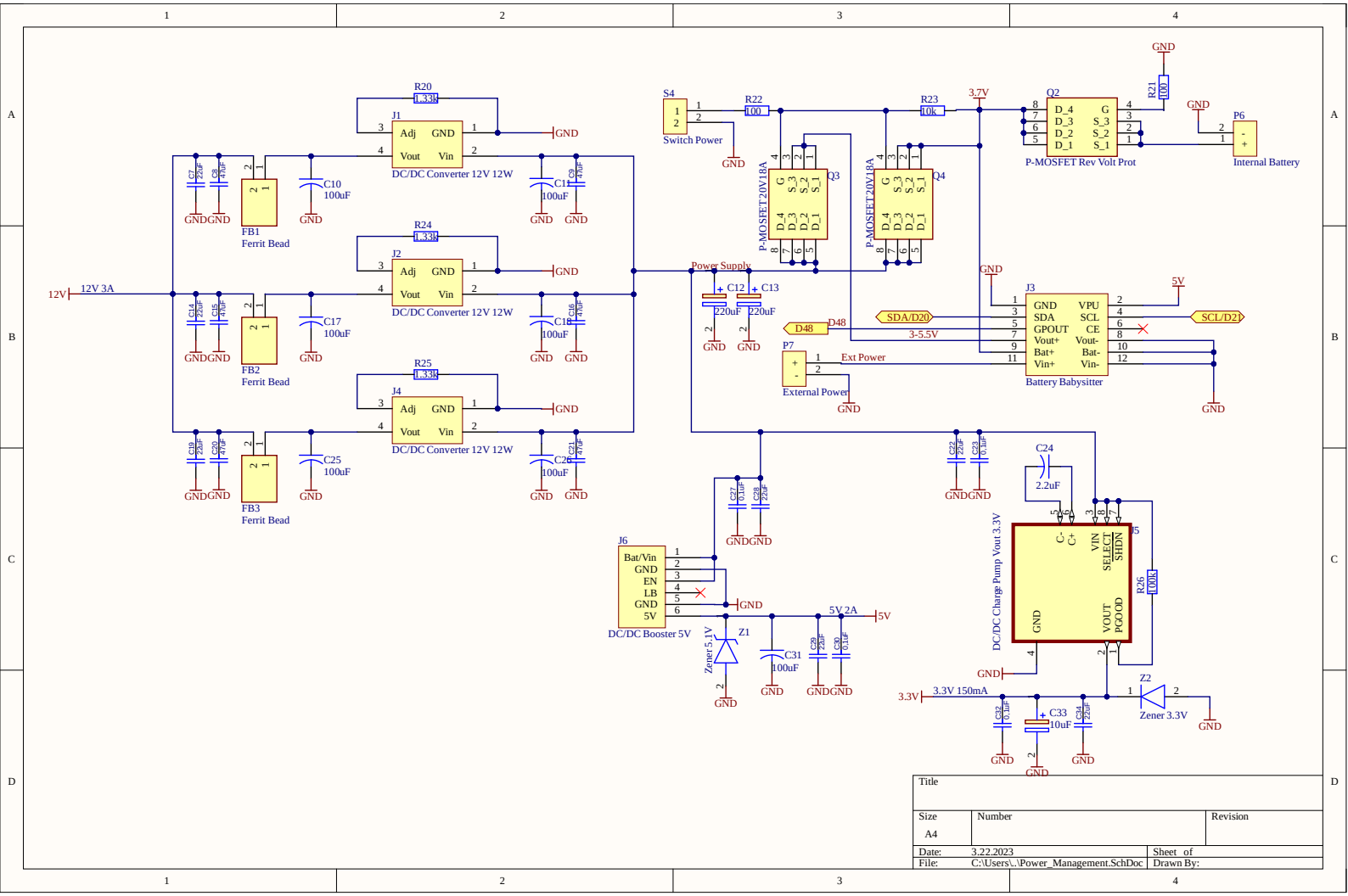
Oversikt over alle pinnene brukt for prototype v.2:

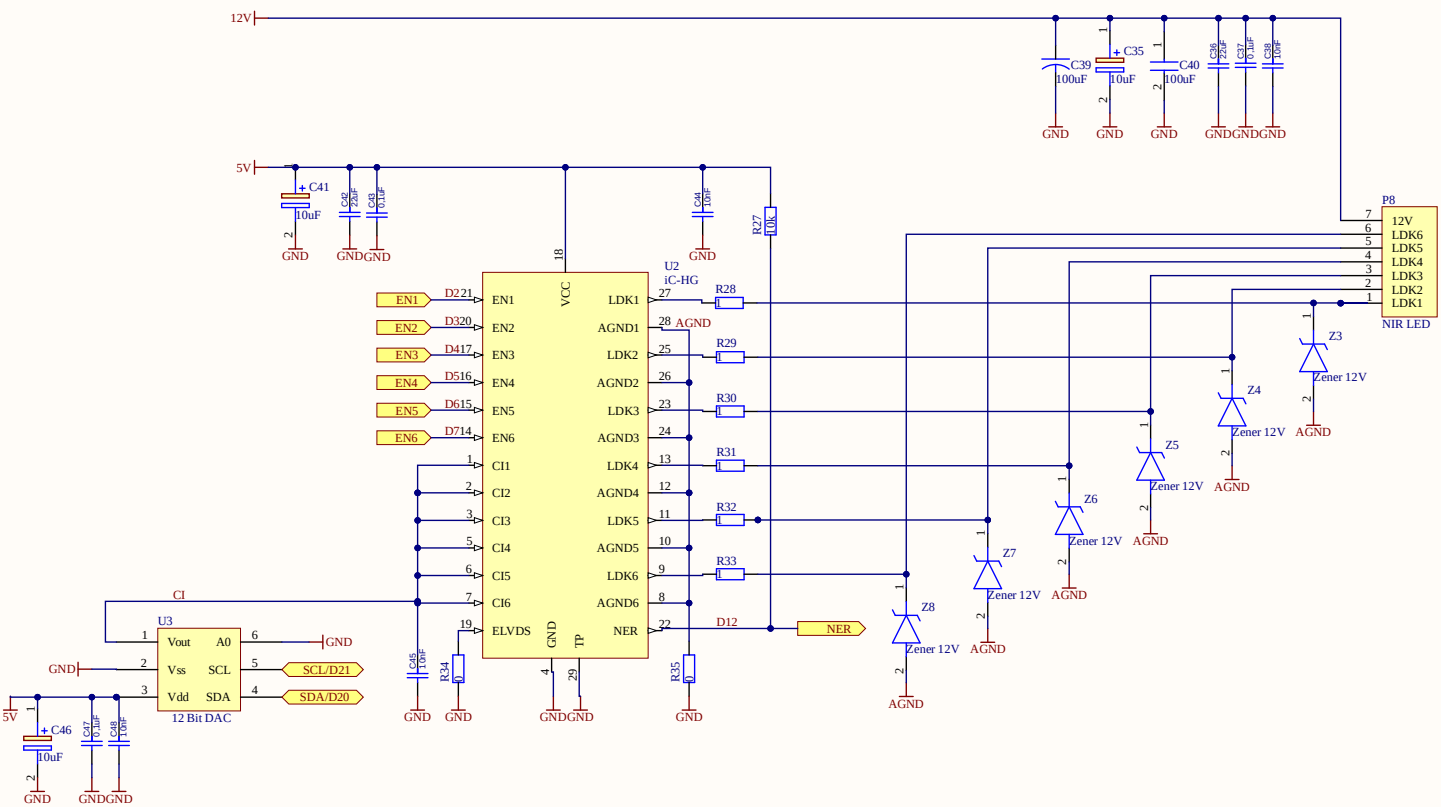
Hva?	Pinne på Arduino Mega:	PORTx	PINx
R_ID - Identifikasjons motstand	A0	PORTF	PIN0
PT1000 - PCB	A1	PORTF	PIN1
PT1000 - LED	A2	PORTF	PIN2
PT1000 - Luft	A3	PORTF	PIN3
PT1000 - Hudkontakt	A4	PORTF	PIN4
Keypad - Y1 (K)	I/O 22	PORTA	PIN0
Keypad - Y2 (J)	I/O 23	PORTA	PIN1
Keypad - Y3 (H)	I/O 24	PORTA	PIN2
Keypad - Y4 (G)	I/O 25	PORTA	PIN3
Keypad - X1 (F)	I/O 26	PORTA	PIN4
Keypad - X2 (E)	I/O 27	PORTA	PIN5
Keypad - X3 (D)	I/O 28	PORTA	PIN6
Skjerm - CS	I/O 32	PORTC	PIN5
Skjerm - Reset	I/O 33	PORTC	PIN4
Skjerm - A0	I/O 34	PORTC	PIN3
Skjerm - SDA	I/O 35	PORTC	PIN2
Skjerm - SCK	I/O 36	PORTC	PIN1
Wifi modul - Reset	I/O 39	PORTG	PIN2
Wifi modul - GPIO-2	I/O 45	PORTL	PIN4
Wifi modul - Tx	Tx3	PORTJ	PIN1
Wifi Modul - Rx	Rx3	PORTJ	PIN0
Indikatorled på Led-hode	I/O 40	PORTG	PIN1
Rød Led	I/O 41	PORTG	PIN0
Grønn Led	I/O 42	PORTL	PIN7
Batterivakt - GPOUT	I/O 48	PORTL	PIN1
Memory Extension - CD	I/O 49	PORTL	PIN0
Memory Extension - DO>	MISO 50	PORTB	PIN3
Memory Extension - DI<	MOSI 51	PORTB	PIN2
Memory Extension - CLK<	SCK 52	PORTB	PIN1
Memory Extension - CS<	SS 53	PORTB	PIN0
NIR LED-driver - EN1	INT0/PWM2	PORTE	PIN4
NIR LED-driver - EN2	INT1/PWM3	PORTE	PIN5
NIR LED-driver - EN3	PWM4	PORTG	PIN5
NIR LED-driver - EN4	PWM5	PORTE	PIN3
NIR LED-driver - EN5	PWM6	PORTH	PIN3
NIR LED-driver - EN6	PWM7	PORTH	PIN4
Buzzer	PWM8	PORTH	PIN5
Knapp - Start Measurement	PWM9	PORTH	PIN6
Vifte - Kontrollenhet	PWM10	PORTB	PIN4
Vifte - LED-hode	PWM11	PORTB	PIN5
LED-driver - Error out	PWM12	PORTB	PIN6
RTC - reset	LED/PWM13	PORTB	PIN7
TWI	INT3/SDA	PORTD	PIN1
TWI	INT2/SCL	PORTD	PIN0

Vedlegg E

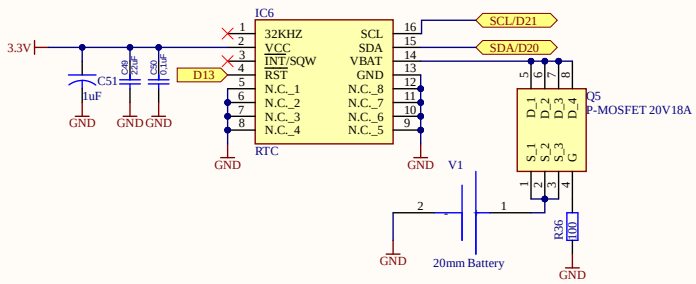
Kontrollenhet v.2 Utlegg


Arduino Mega
P1
LED Power
U_Temp_Meas
PT_LED
PT_PCB
PT_Skin
PT_Air
PT1
P3
Signals Connector 8P
P-MOSFET 20V/8A
U_LED_Controll
U_Real_Time_Clock
Start_Meas
S1
Switch Measurement
P2
LED Red
P4
Fan 5V
P5
LED Green
U1
RS Pro Buzzer
M1
Arduino Memory Extension
U_Power_Management
P12
61300811121
P13
61300711121
P14
61300821121
Title
Size
A3
Number
Revision
Date: 3.22.2023
Sheet of
File: C:\Users\..\Control_Electronics_NIR_Device.SchDoc
Drawn By:

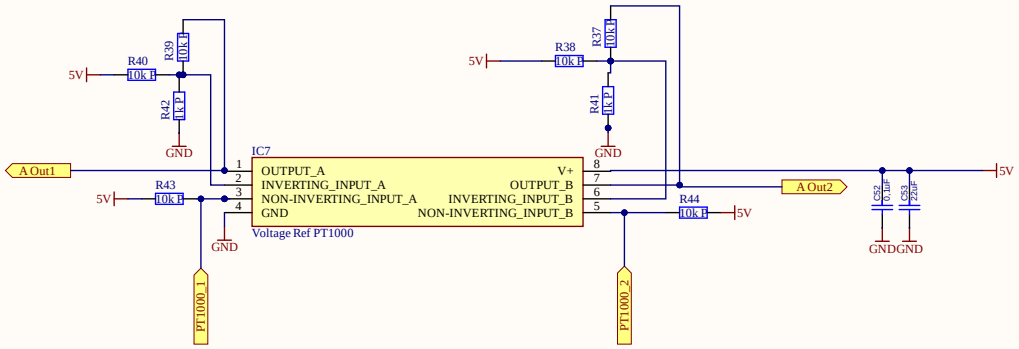

12V
12V 3A
R20
1.33k
J1
Adj GND
Vout Vin
DC/DC Converter 12V 12W
C7 22uF
C8 470nF
FB1
Ferrit Bead
C10 100uF
C11 100uF
C9 470nF
R24
1.33k
J2
DC/DC Converter 12V 12W
C14 22uF
C15 470nF
FB2
C17 100uF
C18 100uF
C16 470nF
R25
1.33k
J4
DC/DC Converter 12V 12W
C19 22uF
C20 470nF
FB3
C25 100uF
C26 100uF
C21 470nF
GND
S4
Switch Power
R22
100
R23
10k
3.7V
Q2
P-MOSFET Rev Volt Prot
R21
100
P6
Internal Battery
Q3
Q4
P-MOSFET 20V 18A
Power Supply
C12 220uF
C13 220uF
D48
SDA/D20
SCL/D21
3-5.5V
5V
J3
GND VPU
SDA SCL
GPOUT CE
Vout+ Vout-
Bat+ Bat-
Vin+ Vin-
Battery Babysitter
P7
Ext Power
External Power
J6
Bat/Vin
GND
EN
LB
GND
5V
DC/DC Booster 5V
5V 2A
Z1
Zener 5.1V
C31 100uF
C29 22nF
C30 0.1uF
C27 0.1uF
C28 22nF
C22 22nF
C23 0.1uF
C24 2.2uF
U5
C+ C-
VIN
SELECT
SHDN
VOUT
PGOOD
DC/DC Charge Pump Vout 3.3V
R26
100k
3.3V
3.3V 150mA
Z2
Zener 3.3V
C32 0.1uF
C33 10uF
C34 22nF
Title
Size
A4
Number
Revision
Date: 3.22.2023
File: C:\Users\..\Power_Management.SchDoc
Sheet of
Drawn By:



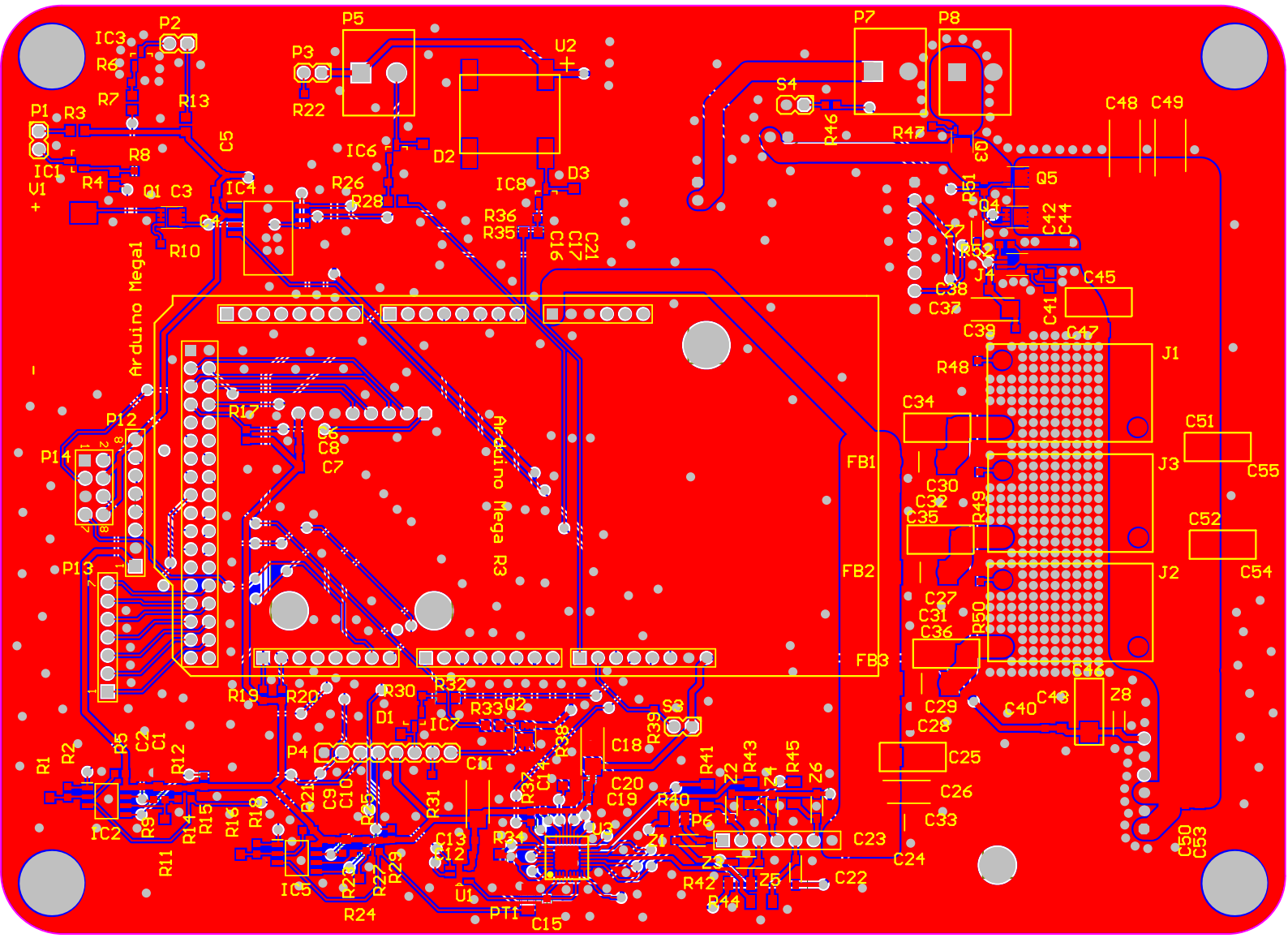
Title		
Size A4	Number	Revision
Date: 3.22.2023		Sheet of
File: C:\Users\..\LED_Controll.SchDoc		Drawn By:



Title		
Size: A4	Number	Revision
Date: 3.22.2023		Sheet of
File: C:\Users\..\Real_Time_Clock.SchDoc		Drawn By:



Title		
Size: A4	Number	Revision
Date: 3.22.2023		Sheet of
File: C:\Users\..\Temp_Meas.SchDoc		Drawn By:


Arduino Mega
Arduino Mega R3
P1
P2
P3
P4
P5
P6
P7
P8
P12
P13
P14
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
U1
U2
J1
J2
J3
J4
Q1
Q2
Q3
Q4
Q5
D1
D2
D3
S4
S6
PT1
FB1
FB2
FB3
VI
+
-

Vedlegg F

Oversikt over feilmeldinger

Feilnummer:	Varsling:	Kategori	Feilmeldingtekst	Del av koden / systemet som har feilet	Mulig feilkilde:
0	Status: Ingen feil	Ingen feil	Alt fungerer som det skal!	Hele	
1	Status: Mindre feil	Lavt batteri	Lavt batteri, bør lades ved neste mulighet	getBatteryState()	Batteri
2	Status: Kritisk feil	SD Card	SD kort initisering failet	initSD()	SD kort ikke lagt inn? Er SD-kortet formatert?
3	Status: Kritisk feil	SD Card	SD kort ikke funnet	initCard()	SD kort ikke lagt inn? Koblet opp korrekt? Er SD-kortet formatert?
4	Status: Kritisk feil	Create directory	Feil ved oppretting av mappe	createDirectory()	SD kort ikke lagt inn? Koblet opp korrekt? Er SD-kortet formatert?
5	Status: Kritisk feil	Create directory	Mappen eksisterer allerede	createDirectory()	Noe galt med systemet som setter eksperiment_id?
6	Status: Kritisk feil	Create file	Filen eksisterer allerede	createFile()	Noe galt med systemet som setter eksperiment_id?
7	Status: Kritisk feil	Create file	Fil ble ikke opprettet	createFile()	SD kort ikke lagt inn? Koblet opp korrekt?
8	Status: Kritisk feil	Write to file	Feil ved skriving til fil	writeFile()	Noe galt med systemet som lager filnavn?
9	Status: Kritisk feil	Write to file	Feil ved å åpne fil	writeFile()	Filen kan være korrupt? Minnekortet er fult
10	Status: Kritisk feil	Write info-fille	Feil ved å åpne fil	writeInfoFile()	Noe galt med systemet som lager filnavn? Er filen korrupt?
11	Status: Ingen feil	Ubrukt			
12	Status: Mindre feil	RTC Dato	Henting av dagens dato fra RTC feilet.	getDate()	TWI kommunikasjon, RTC eller knappebatteri
13	Status: Mindre feil	RTC Tidsstempel	Henting av klokkeslett fra RTC feilet.	getTimeStamp()	TWI kommunikasjon, RTC eller knappebatteri
14	Status: Kritisk feil	DAC	Kritisk feil: DAC	setDAC()	TWI kommunikasjon eller DAC
15	Status: Mindre feil	Batterivakt	Batteristatus ikke tilgjengelig	getBatteryState()	TWI kommunikasjon eller Batterivakt
16	Status: Mindre feil	Batterivakt	Batteriets helse er begynt å bli dårlig	getBatteryState()	Batteri
17	Status: Kritisk feil	Temperatursensor	Kretskort i LED-hodet over 85 grader i ett minutt	setFlagADC()	Overoppheting av system eller feil i operasjonsforsterker
18	Status: Kritisk feil	Temperatursensor	Kretskort i styringsenhet over 70 grader i ett minutt	setFlagADC()	Overoppheting av system eller feil i operasjonsforsterker
19	Status: Kritisk feil	Temperatursensor	Hudtemperatur over 43 grader i et halvt minutt	setFlagADC()	Overoppheting av system eller feil i operasjonsforsterker
20	Status: Kritisk feil	Temperatursensor	Lufttemperatur over 43 grader i ett minutt	setFlagADC()	Overoppheting av system eller feil i operasjonsforsterker
21	Status: Mindre feil	Mangel på hudkontakt	Hudkontakt er fraværende	setFlagADC()	Hudkontakt mistet eller feil på sensor for hudtemperatur
22	Status: Kritisk feil	Setting mode failed	Feil ved setting av modus	intToMode	
23	Status: Kritisk feil	Setting duration failed	Feil ved setting av varighet	intToDuration	
24	Status: Kritisk feil	Setting pvm freq failed	Feil ved setting av PVM frekvens	intToPvmFreq()	
25	Status: Kritisk feil	Error opening file for setting settings	Feil ved åpning av setting fil	getSettingsFromFile()	
26	Status: Kritisk feil	Error saving settings	Feil ved lagring av setting fil	saveSettingsToFile()	

Vedlegg G

Oversikt over komponenter som ble levert til lodding

Tillegg G kan finnes i vedleggsmappen med navnet: ”G_Oversikt komponenter til lodding

Vedlegg H

Solidwork fil 1

Tillegg H kan finnes i vedleggsmappen med navnet: ”H_LED_Electronics_Top.SLDPRT”

Vedlegg I

Solidwork fil 2

Tillegg I kan finnes i vedleggsmappen med navnet: ”I_LED_Electronics_Bottom.SLDPRT”

Vedlegg J

Solidwork fil 3

Tillegg J kan finnes i vedleggsmappen med navnet: ”J_Prototype_v2_lokk.SLDPRT”

Vedlegg K

Solidwork fil 4

Tillegg K kan finnes i vedleggsmappen med navnet: ”K_Prototype_v2_kjerne.SLDPRT”

Vedlegg L

Solidwork fil 5

’ Tillegg L kan finnes i vedleggsmappen med navnet: ”L_Prototype_v2_bunn.SLDPRT”

Vedlegg M

Video av meny

Tillegg L kan finnes i vedleggsmappen med navnet: ”M_Video av Meny”

Vedlegg N

Oppgavebeskrivelsen

NTNU

Institutt for elektroniske systemer
Institutt for elkraftteknikk
Institutt for teknisk kybernetikk

Oppgaveforslag bacheloroppgave elektroingeniør (BIELEKTRO) i Trondheim, vårsemester 2023

Navn forskningsgruppe: **Artificial Pancreas Trondheim (APT)**	**Kontaktperson: Patrick C. Bösch** **Epost: patrick.c.bosch@ntnu.no** **Telefon/mobil: 401 09 318**	
Tittel på oppgave: ***Overvåkningssystem for bruk i lange (flerdagers) dyreforsøk***		
Hvilke studieretninger passer oppgaven for? (kryss av for alle aktuelle retninger; flervalg er mulig):	**Automatisering og robotikk**	☒
	Elektronikk og sensorsystemer	☒
	Elkraft og bærekraftig energi	☐
Er oppgaven reservert for noen bestemte studenter? (skriv navnene på studentene inn til høyre)		
Har dere arbeidsplass for studentene	☐ ja ☐ nei ☒ usikker?	
Er dette en lukket oppgave? Dvs. at sluttrapporten ikke kan publiseres fordi den inneholder sensitiv informasjon.	☐ ja ☐ nei ☒ ikke enda bestemt	

Kort beskrivelse av oppgaven med problemstilling.

Forskningsgruppa Artificial Pancreas Trondheim (APT) gjennomfører forsøk på mennesker hvor det undersøkes om lys av forskjellige bølgelengder kan øke blodsirkulasjonen i underhuden. Målet er å bruke den teknologien for å forbedre blodsukkerreguleringen hos diabetikere.

I et tidligere studentprosjekt ble det påbegynt bygging av fysisk prototypeutstyr som skal brukes i disse forsøkene av blant annet sykepleiere.

I denne oppgaven er målet å fullføre det gjenstående arbeidet på prototypen. Hovedarbeidet som står igjen er programmering og brukergrensesnitt i tillegg til testing og karakterisering av hele prototypen. Oppgaven krever innsikt i instrumentering, programmering, integrerte systemer og måleteknikk. Valg av instrumentering/målemetoder og utforming av brukergrensesnitt må gjøres i samråd med forskerne i APT.

Vedlegg O

Forprosjekt

Tillegg N kan finnes i vedleggsmappen med navnet: ”N_Forprosjekt_Bachelor_NIR_systems

Vedlegg P

Kildekode

Tillegg O kan finnes i vedleggsmappen med navnet: ”O_Kildekode_NIR-system”

Vedlegg Q

Brukermanual NIR-systemet

Tillegg P kan finnes i vedleggsmappen med navnet: ”P_Brukermanual NIR SYSTEM”