
Bacheloroppgave

Prosjektets tittel / Project title: <i>LightBlu – Bluetooth Smart Lysstyringssystem</i> <i>LightBlu – Bluetooth Smart Light Control System</i>	Gitt dato: 05.01.2015 Innlevert dato: 26.05.2015
Prosjektets webside / Project webpage: http://hekta.org/~hpe1534b	Antall sider/bilag 96/26
Gruppedeltakere: <i>Mads Ellingsen Stephansen (MES)</i> Tlf.: 971 14 436 Email: mads.stephansen@hotmail.com <i>Erlend Røed Myklebust (EM)</i> Tlf.: 416 93 261 Email: erlend.r.myklebust@gmail.com <i>Petter Haugen (PH)</i> Tlf.: 454 33 856 Email: petter15@hotmail.com	Veileder: <i>Stein Øvstdal</i> stein.ovstdal@hist.no
Studieretning: <i>ELE12H (Elektronikk)</i>	Prosjektnummer: <i>E1534B</i>
Oppdragsgiver: <i>Nordic Semiconductor ASA</i>	Kontaktperson hos oppdragsgiver: <i>Christian Wilhelmsen</i> christian.wilhelmsen@nordicsemi.no

Fritt tilgjengelig X

Tilgjengelig etter avtale med oppdragsgiver

Rapporten frigitt etter

Høgskolen i Sør-Trøndelag
2015

"LightBlu"

HOVEDRAPPORT

Bachelorprosjekt for avgangsstudenter ved studieretning for elektronikk (ELE12H)
Oppgave: *E1534B Light Dimmer*

Gruppedeltakere

Erlend R. Myklebust

Mads E. Stephansen

Petter Haugen

Oppdragsgiver

Christian Wilhelmsen
Nordic Semiconductor ASA

Rådgiver

Stein Øvstedral
HiST

FORORD

Denne sluttrapporten er siste og avgjørende del av Bacheloroppgave E1534B «Light Dimmer», som utføres av elektronikkstudentene Erlend, Mads og Petter på oppdrag for Nordic Semiconductor ASA.

Studieretning for elektronikk er et treårig bachelorprogram som er underlagt Avdeling for Teknologi ved Høgskolen i Sør-Trøndelag. Studiet består av fag med til sammen 180 poengs vekting. Av disse, allokeres de 20 siste studiepoengene til et bachelorprosjekt, som utgjør studiets avsluttende fase.

Hensikten med denne sluttrapporten er å dokumentere og presentere arbeidet vi har gjennomført i løpet av våren 2015. På denne måten kan de som skulle ha interesse av det få et innblikk i erfaringene vi har opparbeidet oss gjennom bachelorprosjektet. Dersom det skulle være aktuelt å gjenskape prosjektet, eller videreføre det i fremtiden, vil man kunne finne nyttig informasjon i denne rapporten. Vi viser valgt teknisk løsning samt relevant teori for utførelsen.

Sluttrapporten spiller også en vesentlig rolle i forbindelse med karaktersettingen. Oppdragsgiver og veiviser vil bruke denne rapporten som et av kriteriene når karakteren skal bestemmes.

Vi ønsker å benytte anledningen til å takke linjeforeningen Elektra for lån av utstyr og donasjon av mikrokontrollere, og retter samtidig en stor takk til Nordic Semiconductor og Christian Wilhelmsen for glimrende veiledning og en spennende oppgave.

SAMMENDRAG

I dette prosjektet har vi utviklet og realisert et trådløst lysstyringssystem som kontrolleres av en Android-basert enhet gjennom Bluetooth Smart.

Prosjektet er utført på oppdrag for den Trondheimsbaserte teknologibedriften Nordic Semiconductor ASA og har til hensikt å demonstrere egenskapene til Bluetooth Smart gjennom en praktisk anvendelse av Nordics nRF51422 System on Chip (SoC).

Vi har valgt en modulbasert hardware-løsning for dette systemet, der de forskjellige modulene oppfyller følgende roller:

- Kontrollmodul: Håndterer Bluetooth-kommunikasjon
- Binærmodul: Styrer lyset binært (enten av eller på)
- Dimmermodul: Dimmer lyset i 6 nivåer (av, 20%, 40%, 60%, 80% og 100%)

Hver av disse modulene styres av mikrokontrollere som er programmert med separate firmwareversjoner. Det er denne koden som definerer funksjonaliteten til hver enkel modul. Vi har selv utviklet det meste av firmware, dog med noe hjelp.

Når en Kontrollmodul kombineres med enten en Binærmodul eller en Dimmermodul skapes det vi kaller et «produkt». I gjennomførelsen av dette prosjektet har vi realisert to forskjellige produkt; en Binærplugg og en Dimmerplugg. Begge disse kommer i form av et adapter man kan koble mellom lys og stikkontakt.

I tillegg har vi utviklet en mobilapplikasjon for Android versjon 4.3 og nyere. Enheter som kjører denne applikasjonen fungerer dermed som en trådløs fjernkontroll for lyset.

Gjennom denne utviklingsprosessen har vi lært mye om planlegging og utførelse av store prosjektarbeid. Vi har også fått smake på fordelene og ulempene med gruppearbeid etter en gang.

Ved prosjektets slutt gjennomførte vi en rekke sluttmålinger og fikk bekreftet at de endelige versjonene av produktene våre fungerer som tiltenkt. Vi anser derfor prosjektgjennomførelsen som vellykket.

ANSVARSFORDELING PROSJEKT

Erlend	Petter	Mads
Utvikling av firmware Testing av firmware Webside Hovedansvar for gjennomføring	Utvikling av hardware Produksjon av Hardware Testing av Hardware Økonomi	Utvikling av mobilapplikasjon Testing av mobilapplikasjon

ANSVARSFORDELING HOVEDRAPPORT

Erlend	Petter	Mads
2.4 – Mikrokontrollere 2.5 – Kommunikasjonsmetoder 2.8 – Bluetooth Smart (BLE) 4.2 – Firmware for lokal MCU 4.3 – Bluetooth firmware 4.4 – Testing 7.3 – Arbeidsflyt 8.1.2 - Diskusjon FW 8.2 – Konklusjon Forord Sammendrag Plakat	2.1 – Elektroniske komponenter 2.2 – Styring og dimming 2.3 – Kapasitiv PSU 3.1 – Systemløsning 3.2 – Dim. Kapasitiv PSU 3.3 – Simulering og testing 3.4 – Endelig design 6.3 – Sluttresultat 6.4 – Sluttmålinger 7.5 – Økonomi 8.1.1 – Diskusjon HW 8.2 - Konklusjon Forord Sammendrag	1.1 – Oppgaven 1.2 – Problemstilling 1.3 – Mål 1.4 – Rapportens oppbygging 2.6 – Java 2.7 – Android 4.1 – Løsning 5.1 – Opprinnelig plan 5.2 – Utviklingsverktøy 5.3 – Grafisk design 5.4 – Implementering av BLE 5.5 – Testing 6.1 – Systemsammensetning 6.2 – Kommunikasjonsflyt 7.1 – Prosjektgruppa 7.2 – Prosjektplan 7.4 – Avvik 8.1.3 Diskusjon App 8.2 – Konklusjon Forord Sammendrag

Erlend Røed Myklebust

Mads Ellingsen Stephansen

Petter Haugen

INNHOLDSFORTEGNELSE

KAP.1	INNLEDNING	1
1.1	OPPGAVEN.....	1
1.2	PROBLEMSTILLING	1
1.3	PROSJEKTMÅL.....	1
1.4	RAPPORTENS OPPBYGGING.....	2
KAP.2	TEORI	3
2.1	VIRKEMÅTE ELEKTRONISKE KOMPONENTER.....	3
2.1.1	RELÉ.....	3
2.1.2	HALVLEDERTEORI.....	4
2.1.3	DIODE	4
2.1.4	BIPOLAR TRANSISTOR.....	4
2.1.5	TRIAC	4
2.1.6	DIAC	6
2.1.7	OPTOKOBLER	6
2.2	STYRING OG DIMMING AV LYSINSTALLASJONER	7
2.2.1	RELÉSTYRING AV LYS	7
2.2.2	DIMMING AV LYS	7
2.3	KAPASITIV STRØMFORSYNING	10
2.3.1	METALL OKSID VARISTOR	10
2.3.2	FILMKONDENSATOR	11
2.3.3	LIKERETTERDIODER.....	11
2.3.4	GLATTEKONDENSATOR.....	11
2.3.5	ZENERDIODE	12
2.3.6	SPENNINGSREGULATOR.....	12
2.4	MIKROKONTROLERE	14
2.4.1	ATMEL ATTINY45V OG ATMEGA48V.....	14
2.4.2	ARM CORTEX M0 OG NRF51422	15
2.5	KOMMUNIKASJONSMETODER	17
2.5.1	TWO WIRE INTERFACE (TWI).....	17
2.5.2	SERIAL PERIPHERAL INTERFACE (SPI).....	19
2.6	JAVA	20
2.6.1	GENERELT OM JAVA	20
2.6.2	DATATYPER	20
2.6.3	OBJEKTER.....	20
2.6.4	KLASSER OG METODER	20
2.6.5	METODER	21
2.6.6	EKSEMPELPROGRAM.....	21
2.7	ANDROID.....	24
2.7.1	HISTORIE OG GENERELT OM UTVIKLING.....	24
2.7.2	MAPPESTRUKTUR OG FILORGANISERING	24
2.7.3	AKTIVITETER	25
2.7.4	INSTALLASJON AV APPLIKASJONER	26
2.8	BLUETOOTH SMART	27
2.8.1	KORT OPPSUMMERING.....	27
2.8.2	GENERIC ACCESS PROFILE (GAP)	28
2.8.3	GENERIC ATTRIBUTE PROFILE (GATT).....	29

KAP.3	HARDWARE.....	30
3.1	SYSTEMLØSNING	30
3.1.1	ENDRINGER FRA OPPRINNELIG LØSNING.....	30
3.1.2	FYSISK GRENSESNITT.....	31
3.1.3	VALG AV RELÉ-TYPE.....	31
3.1.4	VALG AV DIMMERTYPE.....	31
3.1.5	VALG AV INTERN STRØMFORSYNING	32
3.1.6	VALG AV LOKAL HOVEDKONTROLLER	32
3.2	DIMENSJONERING AV KAPASITIV STRØMFORSYNING	33
3.2.1	EFFEKTBEREGNINGER.....	33
3.2.2	DIMENSJONERING AV FILMKONDENSATOR	34
3.2.3	VALG AV REGULATORTYPE	34
3.2.4	DIMENSJONERING AV ZENERDIODE	34
3.2.5	DIMENSJONERING AV GLATTEKONDENSATOR.....	35
3.3	SIMULERING OG TESTING	36
3.3.1	SIMULERING	36
3.3.2	TESTING AV PROTOTYPER.....	40
3.3.3	ENDRINGER SOM FØLGE AV TESTING	45
3.4	ENDELIG DESIGN.....	46
KAP.4	FIRMWARE.....	47
4.1	LØSNING.....	47
4.2	FIRMWARE FOR LOKAL HOVEDKONTROLLER	48
4.2.1	KODE FOR TWI	48
4.2.2	FIRMWARE FOR BINÆRMODUL (LDFBIN1).....	50
4.2.3	FIRMWARE FOR DIMMERMODUL (LDFDIM1).....	52
4.3	BLUETOOTH SMART FIRMWARE.....	57
4.3.1	OVERORDNET FUNKSJONSFORKLARING.....	57
4.3.2	KODEFLYT.....	57
4.4	TESTING	59
KAP.5	MOBILAPPLIKASJON.....	61
5.1	OPPRINNELIG PLAN	61
5.2	UTVIKLINGSVERKTØY.....	62
5.3	GRAFIK DESIGN.....	62
5.4	IMPLEMENTERING AV BLUETOOTH SMART.....	63
5.5	TESTING	66
KAP.6	INTEGRERING OG RESULTAT.....	67
6.1	SYSTEMSAMMENSETNING	67
6.2	KOMMUNIKASJONSFLYT.....	67
6.3	SLUTTRESULTAT	69
6.4	SLUTTMÅLINGER	71
6.4.1	BINÆRMODUL.....	71
6.4.2	DIMMERMODUL	78
KAP.7	PROSESSEN	87
7.1	PROSJEKTGRUPPA.....	87
7.2	PROSJEKTPLAN.....	88
7.3	ARBEIDSFLYT	89
7.4	AVVIK	91
7.5	ØKONOMI.....	92

KAP.8	DISKUSJON OG KONKLUSJON	93
8.1	DISKUSJON.....	93
8.1.1	HARDWARE	93
8.1.2	FIRMWARE	94
8.1.3	MOBILAPPLIKASJON	95
8.2	KONKLUSJON	96
FIGURLISTE		I
TABELLISTE		III
KILDELISTE		IV
VEDLEGG		IX
	VEDLEGG 1 PROSJEKTOPPGAVEN.....	X
	VEDLEGG 2 UTLEGG OG SKIEMA.....	XII
	VEDLEGG 3 KOMPONENTLISTER	XVIII
	VEDLEGG 4 DIGITALT ARKIV / FIRMWARE	XXI
	VEDLEGG 5 GANTT-DIAGRAM.....	XXIII
	VEDLEGG 6 PLAKAT	XXV

KAP.1 INNLEDNING

1.1 OPPGAVEN

Oppgaven gikk ut på å benytte seg av en smart-telefon med Bluetooth Smart teknologi til å kontrollere et lysstyringssystem basert på Nordic Semiconductors nRF51822 Bluetooth Smart chip (senere erstattet med nRF51422). Det skulle designes firmware for en dimmerenhet. I tillegg skulle det være mulig å koble til flere ulike lysenheter og justere lyset fra en og samme smarttelefon. Se Vedlegg 1.

Etter å ha gått igjennom oppgaveteksten kom vi frem til at det var flere sentrale elementer. Naturligvis var bruk av Nordics chip og lysdimming noe vi vektla under planleggingen, men også muligheten for å konstruere noe som enkelt kan installeres inn i en helt ordinær husholdning. Å lage et lysstyringssystem som appellerer til et vidt spenn av forbrukere, og ikke kun for teknisk interesserte, var viktig for oss. Oppdragsgiver var klar på at vi stod helt fritt til å velge selv hvordan oppgaven skulle løses. Fjernstyring av lysstyringssystemer eksisterer allerede på markedet, men vi har forsøkt å være kreative med å komme frem til nyskapende løsninger.

1.2 PROBLEMSTILLING

Trådløs regulering av lys i bolig er tradisjonelt blitt utført av systemer som kommuniserer trådløst på 400 MHz-båndet. Denne løsningen er forholdsvis enkel og tilbyr god rekkevidde, men systemet mangler vesentlig sikkerhetsfunksjonalitet i og med at det kommuniserer ukryptert. I senere tid har det blitt etablert løsninger for regulering av lys over WiFi, men i og med at WiFi-kommunikasjon er energikrevende gjør det systemet lite fleksibelt ovenfor enheter med redusert batterikapasitet. Ved å benytte Bluetooth Smart som kommunikasjonsgrensesnitt vil vi kunne kombinere funksjoner som kanalvalg, kryptering og lavt energiforbruk.

Vi har derfor valgt følgende problemstilling: Utvikle og realisere et trådløst lysstyringssystem basert på Bluetooth Smart, som skal fungere som en «accessory» til Android-baserte enheter (versjon 4.3 og nyere).

1.3 PROSJEKTMÅL

Målene for prosjektet definerte vi i forprosjektrapporten og lyder som følger:

Effektmål:

- Demonstrasjone interoperabiliteten til Bluetooth Smart.
- Demonstrasjone lav-effekt-egenskapene til Bluetooth Smart.
- Demonstrasjone en praktisk anvendelse av Bluetooth Smart.

Resultatmål:

- Lage et enkelt og fleksibelt lysstyringssystem.
- Lage en oversiktlig mobilapplikasjon som skal kunne styre og overvåke lysstyringssystemet.
- Lage firmware for tilgang til lysstyringssystemet.

- Lage en demonstrasjonsenhet som oppfyller effektmålene.

Prosessmål:

- Opparbeide erfaring innen profesjonell prosjektstyring og utførelse.
- Opparbeide erfaring innen profesjonelt kundeforhold, gruppearbeid og problemløsning.
- Opparbeide kunnskap om Bluetooth Smart.
- Opparbeide kunnskap om komponenter og kretser brukt i forbindelse med styring og dimming av lys.
- Opparbeide erfaring innen utvikling av firmware og appdesign.
- Øke innsikten i hvilke ansvarsområder og arbeidsoppgaver en ingeniør er pålagt.

1.4 RAPPORTENS OPPBYGGING

Rapporten er satt opp slik at ved å lese kapitlene kronologisk, skal leseren kunne sitte igjen med såpass mye kunnskap om prosjektet at han/hun er i stand til å gjenfortelle hva vi har gjort. Det er en gradvis oppbygging, med en beskrivelse av oppgaven i starten. Deretter følger en forklarende teoridel, som gir forståelse for hvordan vi har anvendt teorien i praksis. Hoveddelen består av en presentasjon av løsningene innenfor fagfeltene hardware, firmware og applikasjon som vi har kommet frem til. Vi forklarer hvilke valg vi har blitt nødt til å ta, og hvorfor. Rapporten avsluttes med informasjon rundt prosessen og diskusjon rundt produktet vårt, før en endelig konklusjon.

Merk: For å best forstå utgangspunktet til denne rapporten, bør man først lese gjennom forprosjektrapporten som ble utviklet i starten av prosjektet.

KAP.2 TEORI

2.1 VIRKEMÅTE ELEKTRONISKE KOMPONENTER

Styring av lys-systemer utføres i dag ved bruk av forskjellige løsninger, med bruk av ulike komponenter. Dette underkapitlet tar for seg komponenter som er sentrale for å kunne styre og kontrollere lys, henholdsvis av/på-styring og dimming.

2.1.1 RELÉ

Reléer har over lengre tid blitt brukt til å styre spenninger i elektriske anlegg og har et vidt bruksområde, for eksempel i industri, boliginstallasjoner og kjøretøy.

Et relé er en komponent som styrer en større spennin ved hjelp av et mindre styresignal. [1] Det eksisterer et bredt utvalg av forskjellige typer reléer, hver og en med forskjellige virkemåter og spesifikasjoner.

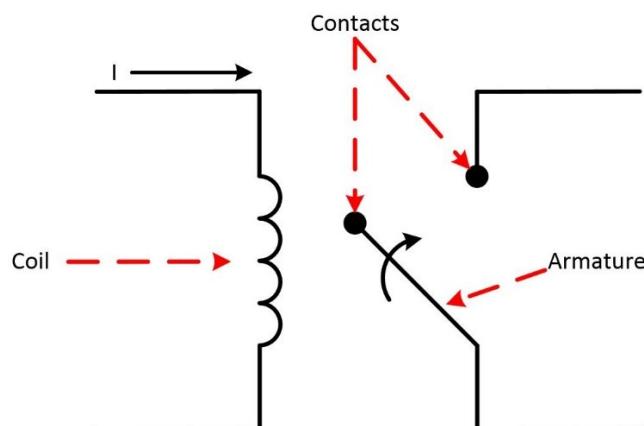
Solid-State-Reléer benytter seg av halvlederteknologi, hvor en lysdiode overfører et elektrisk styresignal for å styre en større spennin. [2] Dette anses som den mest nyvinnende teknologien brukt for reléstyring i dagens marked, men er fortsatt ikke den teknologien som er mest benyttet.

Den mest kjente teknologien som benyttes i reléer i dag er elektromagnetisme. Magnetisme er en naturkraft hvor elektriske ladninger og deres kraft kan danne spesifikke kraftfelt. Magnetisme består av et positivt og negativt kraftfelt, hvor stoffer med like kraftfelt vil frastøtes hverandre og stoffer med ulike kraftfelt vil tiltrekkes. [3]

Vi vil fokusere på å forklare virkemåten for det elektromekaniske reléet, som har tre sentrale komponenter: Spole, armatur og kontaktsett.

Spolen og kontaktene er fastmontert i reléet, mens armaturen opererer som en bryter. Når det tilføres en strøm til spolen induseres det et magnetfelt, og dermed styres posisjonen til armaturen mot en av kontaktene i reléet. [2] Ved å koble et styresignal til spolen og den ene spenningsfasen til kontaktsettet kan man kontrollere spenningstilførselen til ønsket utstyr, ved å styre om spenningstilførselen skal være til- eller frakoblet.

Reléer kan konfigureres i et bredt utvalg konstellasjoner, og kan blant annet konfigureres med flere spoler og flere kontaktsett i samme enhet. Ettersom armaturen alltid står tilkoblet et av kontaktsettene kan man konfigurere reléet for et rikt utvalg av bruksområder, avhengig av behov.



FIGUR 1. ELEKTROMEKANISK RELÉ

2.1.2 HALVLEDERTEORI

Et sentralt tema for forståelsen av all moderne elektronikk er halvlederteori, og hvordan bruken av den kan gi ulik funksjonalitet. En halvleder er et kjemisk materiale som leder strøm dårligere enn en rent metallisk leder. [4]

Halvledere kan bygges opp av flere forskjellige materialer, men det mest brukte i dag er silisium. Ved å kjemisk påvirke ("dope") silisiumet kan det gis egenskaper som gjør det mer egnet til å lede strøm, ved at det naturlig innehar et overskudd (N-type) eller underskudd (P-type, hull) av elektroner i materialet. [5]

Ved å kombinere N-type og P-type materialer i ulike konstellasjoner kan man konstruere en rekke halvlederkomponenter med forskjellige egenskaper, deriblant diode, transistor, tyristor og TRIAC.

2.1.3 DIODE

En diode er en halvlederkomponent som leder strøm i kun en retning. Dioden består av ett p-materiale og ett n-materiale satt sammen i en PN-overgang, og danner tilkoblingspunktene anode og katode. Ved å påføre en spenning med mer positiv verdi på anode i forhold til katode vil dioden transportere p-materialene fra anode til katode, og n-materialene fra katode til anode. Dette tilsvarer en strømgjennomgang i komponenten. Når komponenten påføres en spenning med mer negativ verdi på anode i forhold til katode vil den sperre for strømgjennomgang. [6] Dette er en av de mest grunnleggende egenskapene ved halvlederkomponenter, som benyttes i flere komponenter enn dioden, blant annet transistor og tyristor.

2.1.4 BIPOLAR TRANSISTOR

En bipolar transistor er en halvlederkomponent som består av to PN-overganger. De to overgangene danner tre soner med hvert sitt tilkoblingspunkt, kalt base, emitter og kollektor. Med like elektriske egenskaper som PN-overgangen i en diode kan man styre en stor strøm til den ene terminalen ved å påføre en liten strøm på en av andre terminalene (base). [6] Transistoren eksisterer i en rekke konfigurasjoner, og dens egenskaper benyttes i likhet med dioden i flere komplekse elektroniske komponenter.

2.1.5 TRIAC

Triac er en halvlederkomponent som brukes til å regulere strømgjennomgangen til en last. Ved å benytte en triac og noen støttekomponenter kan man regulere strømgjennomgangen og dermed kontrollere bruken av for eksempel induksjonsmotorer, dimbare lamper og varmeovner. [7]

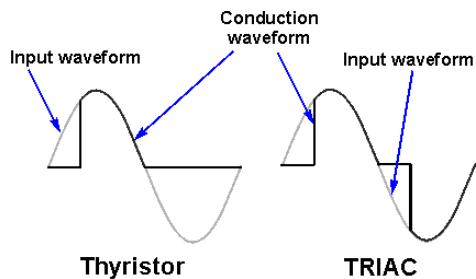
Komponenten triac har lignede virkemåte som komponenten tyristor, men med et par tilleggsegenskaper.

Tyristoren består av 4 lag silisium, og opererer som en styrt likeretterdiode bestående av anode, katode og gate.

Hvis det påføres en vekselspenning med potensial over en grenseverdi på anode og det påføres et styresignal til gate vil det gå en strøm i dioden i den ene halvperioden. Strømmen som går gjennom dioden føres til lasten, hvor det oppstår et spenningsfall. Strømgjennomgangen i dioden vil vedvare til spenningen på anoden krysser grenseverdien på vei mot nullpunktet. Dette vil si at man kan regulere effekten til lasten ved å endre på tidspunktet for når styresignalet skal være aktivt.

For den andre halvperioden vil dioden sperre for strømgjennomgang, og vil ikke lede strøm før den igjen har et positivt spenningspotensiale på anode og et styresignal på gate. Dermed regulerer man spenningstilførselen i kun den ene halvperioden, noe som gir en dårlig utnyttelse av den tilførte spenningen.

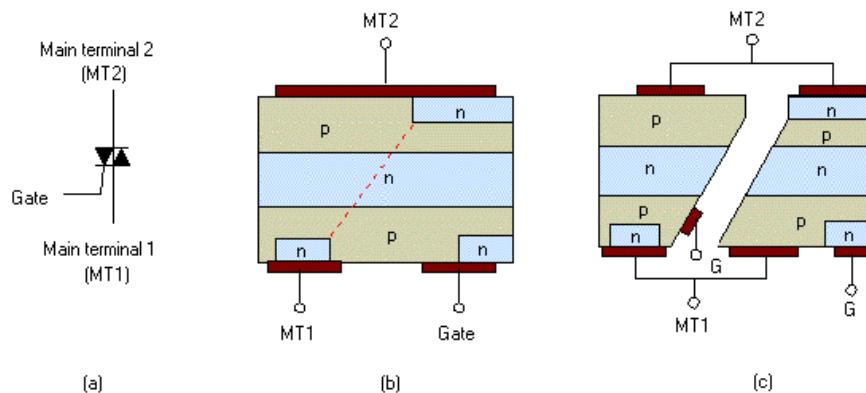
Ved å konfigurere to tyristorer i anti-parallell får man en komponent med tilsvarende egenskaper som en triac. En triac kan regulere begge halvperiodene av en vekselspenning ved å la de to tyristorene regulere hver sin halvperiode.



FIGUR 2. REGULERING TYRISTOR OG TRIAC

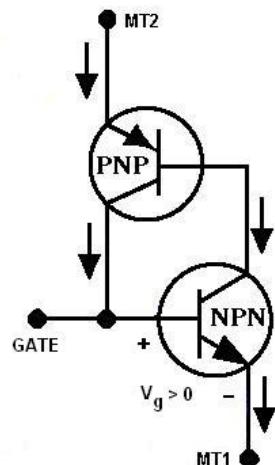
En triac er bygd opp av 6 lag silisium, konfigurert med to terminaler og en gate. [7] Figur 3 viser det skjematiske tegnet for en triac (a), triacens interne konstruksjon (b) og dens interne konstruksjon som to tyristorer (c).

Triacens parallelitet med tyristoren gjør at man kan se triacen som to tyristorer i henholdsvis PNPN- og NPNP-konstellasjon. [8] Den interne plasseringen av n- og p-materialer i triacen, samt deres fysiske størrelse avgjør hvor stor strøm og hvilken polaritet som trengs på gate for at triacen skal begynne å lede. [8]



FIGUR 3. TRIAC, SKJEMASYMBOL OG OPPBYGNING

Triacens interne konstruksjon kan sammenlignes med PN-overgangene som benyttes i transistorer, og den kan derfor tenkes å ha samme funksjon som to transistorer, som vist i Figur 4.



FIGUR 4. TRIAC, EKVIVALENT

Plassert i serie med en last opererer triacen som en regulator, og vil regulere strømgjennomgangen til lasten ved hjelp av et styresignal på gate. Spenningen over lasten vil variere etter når styresignalet aktiverer utgangen til triacen. På grunn av naturlige ulikheter vil ikke de to tyristorene i triacen ha mulighet til å oppføre seg helt likt hverandre, og man vil derfor kunne oppleve at den ene tyristoren regulerer ulikt i forhold til den andre i samme periode. Denne ulikheten i reguleringen vil påvirke spenningstilførselen til lasten negativt, og vil blant annet kunne føre til harmoniske svingninger og EMI-støy. [7]

2.1.6 DIAC

En diac er konstruert som to tyristorer i anti-parallel, men er konfigurert med to terminaler og ingen gate.

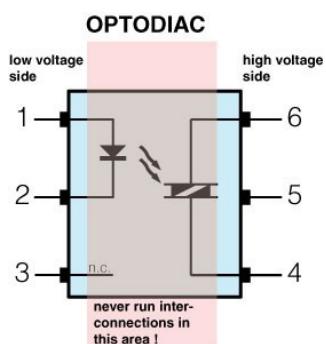
Siden diacen ikke har noen gate vil den operere som to likeretterdioder i anti-parallel, hvor diodene er aktive i hver sin halvperiode. Diodene har lik karakteristikk og vil derfor begynne å lede strøm ved samme grenseverdi, både for positiv og negativ halvperiode. Triacen får dermed et tilpasset styresignal med like stor strømstyrke for hver halvperiode, og vil derfor regulere likt for begge halvperiodene. [9]

2.1.7 OPTOKOBLER

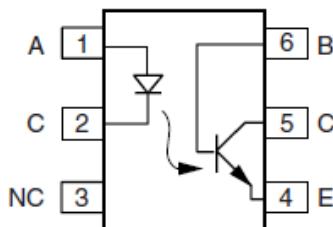
En optokobler er en halvlederkomponent som binder sammen to separate elektriske kretser ved hjelp av lys.

Optokobleren er representert med en primær- og sekundærside, og benytter i hovedsak en lysdiode, en lukket optisk kanal og en fotosensitiv diode for å overføre elektriske signaler fra primær til sekundärsiden. [10] Optokobleren eksisterer i flere konstellasjoner, blant annet med integrert transistor-krets og integrert diac plassert på sekundärsiden. [11] Det at optokobleren kan konfigureres i flere konstellasjoner gjør den veldig fleksibel, og den kan brukes både for signalering fra en svakstrømskrets til sterkstrømskrets og omvendt.

For eksempel kan en styrekrets med likespenning plasseres på primärsiden, og en utgangskrets med høy vekselspenning på sekundärsiden. Ved bruk av lys for signaloverføring kan man trygt styre vekselspenningskretsen fra likespenningskretsen uten å være redd for spenningsoverslag eller støy mellom de to kretsene.



FIGUR 5. OPTOKOBLER MED DIAC



FIGUR 6. OPTOKOBLER MED TRANSISTOR

2.2 STYRING OG DIMMING AV LYSINSTALLASJONER

2.2.1 RELÉSTYRING AV LYS

Styring av lys i boliginstallasjoner har tradisjonelt blitt utført ved bruk av bryterstyring. Dette er kjent som en forholdsvis enkel og økonomisk metode for å styre lys da den krever få eksterne komponenter for å styre en spenningsfase av eller på. En av ulempene er at bryterstyring gir lite fleksibilitet i forhold til implementering i andre systemer da dens utforming i stor grad består av enkle mekaniske komponenter. For å kunne styre lys fra et felles system velger man derfor ofte å benytte seg av reléstyring, hvor en sentral styreenhet skal kunne operere en eller flere lyskilder.

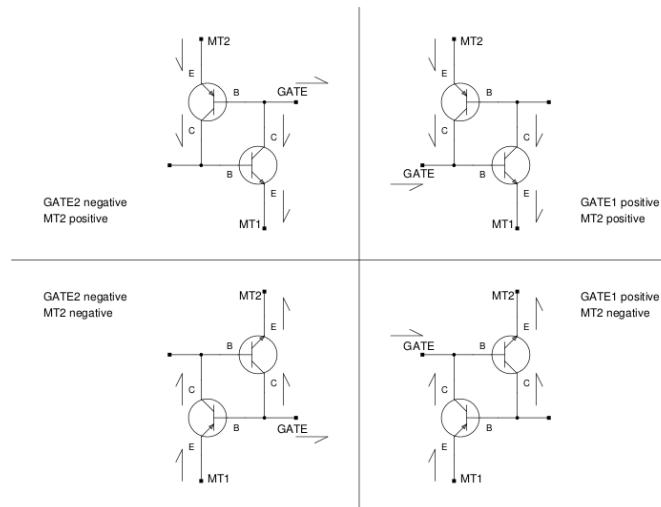
Elektromekaniske relé vil i de fleste tilfeller ha en påslagstid på 5 til 15 ms, og vil kunne levere full effekt til påsatt last såfremt de er dimensjonert for det. Dermed vil bruk av elektromekaniske relé gi en tilnærmet lik brukeropplevelse av lys som ved bryterstyring. Ettersom påslagstiden til reléet er 5 til 15 ms vil det derimot ikke egne seg til dimming av lys, da armaturen ikke vil kunne operere raskt nok til å regulere utgangsspenningen. [2]

2.2.2 DIMMING AV LYS

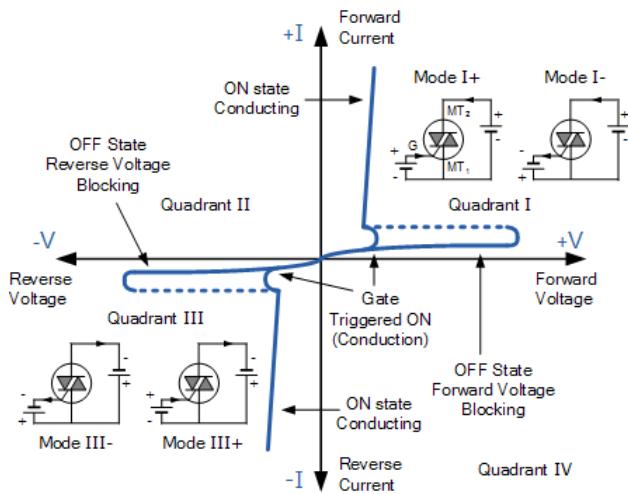
Dimming av lys ble tidligere utført ved hjelp av variable motstander, noe som var en ineffektiv og potensielt farlig løsning med tanke på effektforbruk og varmetap. [12] I nyere tid har man funnet løsninger som gjør det mulig å regulere lys ved hjelp av halvlederkomponenter, blant annet ved bruk av mosfet-transistor og triac.

En mosfet-transistor er konstruert med terminalene gate, drain og source. Ved å påføre en spenning på gate vil det gå en strøm mellom source og drain, en strøm som vil øke med økt spenning på gate. Siden mosfet-transistoren opererer som en enkeltstående transistor vil det kun være mulig å regulere den ene halvperioden av vekselspenningen, samtidig som transistoren trenger et kontinuerlig styresignal på gate for å holde i gang reguleringen.

Et triacbasert dimmesystem muliggjør dimming av lys ved å regulere strømgjennomgangen til lyset med et styresignal på gate. Triacsens regulering er delt inn i fire modus, hvor triacsens interne oppbygging av n- og p-materialer i sammenheng med spenningsstyrken og polariteten på gate og MT2 bestemmer hvor sensitiv triacen er for å starte en strømgjennomgang.



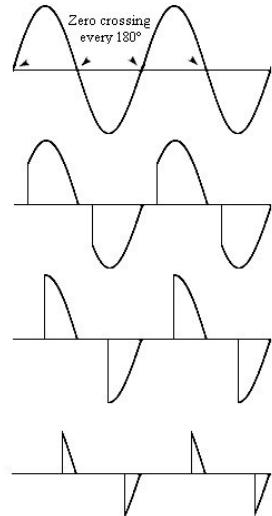
FIGUR 7. TRIAC, MODUSER FOR TRIGGING



FIGUR 8. TRIAC, STRØM- OG SPENNINGSKARAKTERISTIKK

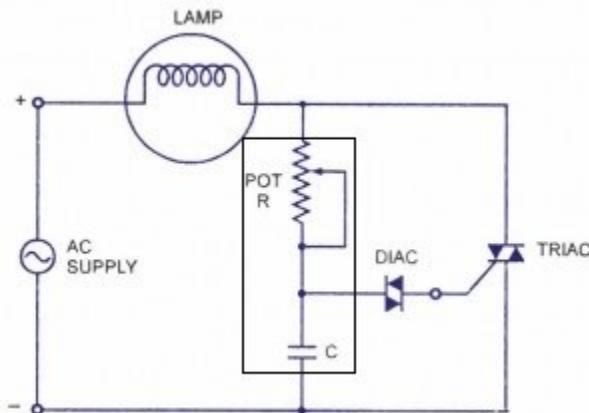
Figur 7 viser strømretningen i triacen når den triggges i de fire ulike operasjonsområdene, mens Figur 8 viser strøm- og spenningskarakteristikken for de ulike strøm- og spenningskvadrantene.

Triacen er aktiv i kvadrant en og tre, og kan triggges med både positiv og negativ puls for disse kvadrantene. Et typisk bruksmønster for triacen er å påføre et styresignal med lik polaritet som tilførselsspenningen, da triacen krever minst strøm på gate for å starte strømgjennomgang hvis gate og terminal 2 er i fase. [8] Ved å regulere tiden for å påføre styresignalet til gate kan man regulere spenningen til lasten over en periode.



FIGUR 9. DIMMING, SPENNING OVER LAST

I en praktisk sammenheng vil styresignalet til Triacen være et modifisert AC-signal med lik oppførsel som en puls. Påføringen av styresignalet vil avhenge av hvilket styresystem som benyttes for å regulere lyset. Et analogt styresystem, for eksempel en variabel motstand og en kondensator vil operere som en vri-bryter og gi en forholdsvis enkel kontroll av lysreguleringen.



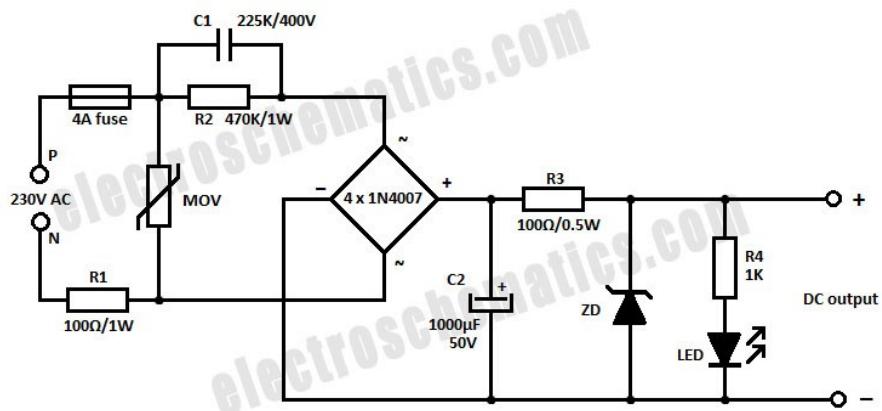
FIGUR 10. DIMMING, ANALOG EKSEMPELKRETS

Et digitalt styresystem vil gi en større fleksibilitet og økte muligheter for bruk i lignende applikasjoner, men vil øke kompleksiteten til dimmerkretsen betraktelig. Når dimmeren styres digitalt må den digitale styrekretsen oppfatte når tilførselsspenningen krysser nullpunktet, og deretter beregne tidspunktet for når styresignalet skal påføres gaten på triacen for å oppnå ønsket dimmenivå. Hvis det benyttes et digitalt styresystem med dedikert strømforsyningskrets er det også en fordel å isolere likespenningskretsen fra vekselspenningskretsen. Dette gjøres ved å benytte optokoblere, både for å detektere nullgjennomgang og for å påføre styresignal til gate.

2.3 KAPASITIV STRØMFORSYNING

Strømforsyninger utgjør en vesentlig del av dagens elektronikkindustri, og benyttes i så godt som alle forbrukerapplikasjoner. Strømforsyninger for likespenning eksisterer i flere varianter, og tjener med ulike egenskaper og fysisk utforming til ulike formål. Den mest kjente strømforsyningskretsen som brukes i dag er switch-mode-forsyning. Den opererer som en strømforsyning med tilbakekobling, hvor den overvåker og sammenligner utgangsspenningen med en referanseverdi. Hvis utgangsspenningen ikke stemmer med referanseverdien vil strømforsyningen regulere hvor stor andel av den påførte spenningen som skal slippe til på utgangen. [13]

En kapasitiv strømforsyning benytter seg av relativt få og enkle komponenter for å konvertere en høy vekselspenning til en lavere likespenning. Kapasitive strømforsyninger eksisterer i flere varianter. Figur 11 viser en typisk kapasitiv strømforsyning.



FIGUR 11. KAPASITIV STRØMFORSYNING, EKSEMPEL

Vi har gjort noen modifiseringer på vår kapasitive strømforsyning ved å fjerne LED og R4 ettersom de kun er plassert for å påvise at strømforsyningen er aktiv. Samtidig har vi plassert en regulator på utgangen av kapasitivforsyningen for å levere 5 volt til styrekretsen vår. Regulatoren er i så måte ikke en del av den originale kapasitive strømforsyningen, men vi ser den som så vesentlig at vi allikevel har valgt å forklare dens virkemåte.

2.3.1 METALL OKSID VARISTOR

Første parallellegren i den kapasitive strømforsyningen består av en metall oksid varistor (RV1). Denne opererer som en spenningsavhengig ikke-lineær motstand, og er plassert i kretsen som en sikkerhet komponent. Ved normal drift opererer varistoren som en høy motstandsverdi, og leder tilnærmet ingen strøm. Hvis den tilførte spenningen stiger over en grenseverdi vil motstandsverdien i varistoren synke, og den vil begynne å lede strøm. Dette gjør varistoren egnet som et overspenningsvern, da den ikke forstyrrer den øvrige kretsen ved normal drift, samtidig som den kan ta opp et større spenningsfall hvis kretsen skulle oppleve ustabil forsningsspenning [37]. Motstand R1 er plassert i serie med RV1 fram til N-forsyning for å hjelpe overspenningsvernet med å omsette effekt ved et større spenningspåtrykk.

2.3.2 FILMKONDENSATOR

En kapasitiv strømforsyning har en filmkondensator (C1) plassert i parallel med en utladingsmotstand (R2). En filmkondensator består av en tynn plastikkfilm som dielektrisk isolasjonsmateriale. Karakteristikken til plastikkfilmen tilsier at polariteten til den påsatte spenningen er uvesentlig, noe som gjør en slik type kondensator velegnet til bruk i vekselspenningsnett. Filmkondensatoren anses som en stødig komponent med lave ESR- og ESL-tap, men vil ikke kunne levere større mengder effekt ettersom bruk av plastikkfilm gir den en relativt liten kapasitansverdi og ugunstig fysisk størrelse. [14]

Filmkondensatorens oppgave i kretsen er å filtrere støy fra tilførselsspenningen, samt transformere ned tilførselsspenningen. Transformasjon av spenningen gjøres ved at parallellkoblingen C1 -R1 plasseres i serie med resten av kretsen, hvor blant annet kondensator C2 befinner seg. Disse to vil utgjøre en kapasitiv spenningsdeler, som deler tilførselsspenningen mellom parallellkobling C1-R1 og C2. Spenningsforholdet mellom parallellkoblingen og C2 vil variere i forhold til den videre konfigurasjonen av kretsen, men vil på generell basis avgjøres av størrelsесforholdet mellom C1 og C2. [15]

$$\frac{V_{C1}}{V_{C2}} = \left(\frac{C_1}{C_1+C_2} \right)^{-1}$$

Dette vil si at en økning i kapasitansverdi i kondensator C1 vil føre til en økt spenning over kondensator C2, og omvendt. Hamill & Hamill har utredet en teori for en lignende kapasitiv kobling uten zenerdiode. Der er spenningen over kondensator C2 utredet i følgende formel:

$$V_{C2}(t) = \sqrt{2 * V_I} * \cos(\omega * t) + V_0 + 2 * V_D$$

Denne formelen kan også benyttes i våre beregninger, hvor V_0 vil representere spenningsfallet over zenerdiode Z1 og motstand R3, mens V_D vil representere spenningsfallet over en enkelt likeretterdiode.

2.3.3 LIKERETTERDIODER

Diodene D1, D2, D3 og D4 opererer som likeretterdioder, og er i vårt tilfelle konfigurert som en likeretterbro. En likeretterdiode er en halvlederkomponent som kun leder strøm når den har et spenningspotensial på anode-terminalen med større positiv verdi enn katode-terminalen. Likeretterdioden vil ikke lede strøm når spenningspotensialet over anode-terminalen er mer negativt enn katode-terminalen, noe som gjør den optimal for likerettingsoperasjoner. Dioden er konstruert for å sperre for negativt spenningspotensiale opp til en gitt verdi, og vil kunne feile ved å begynne og lede strøm hvis spenningspotensialet overstiger denne verdien. [16]

Likeretterbroens fire dioder jobber to og to i par, som konverterer vekselspenningen fra kondensator C1 om til en pulserende likespenning. [17] Ved å benytte en likeretterbro kan man utnytte begge halvperiodene av vekselspenningen, med en effektivitet på opp til 81,2 %. [18]

2.3.4 GLATTEKONDENSATOR

Foruten å operere som en spenningsdeler, har kondensator C2 også oppgaven som glattekondensator. Glattekondensator C2 plasseres i parallel med zenerdiode Z1 og motstand R3, og påføres den pulserende likespenningen fra likeretterbroen. C2 vil forsøke å påføre toppverdien fra den pulserende likespenningen over til zenerdioden, uavhengig av endring i den påførte spenningen.

Kondensatorens evne til å gjennomføre oppgaven vil avgjøres av hvor stor ladning den kan bære, med andre ord hvor stor kapasitansen i den er. Glattekondensatorer benytter ofte materialet elektrolytt som dielektrisk isolasjon. Materialet inneholder en stor ansamling av ioner, som gir mulighet for å bære en stor ladning over et mindre område. Dette gjør at elektrolyttkondensatoren i mange tilfeller kan operere med en stor kapasitansverdi, vesentlig større enn

en filmkondensator. [19] Egenskapene i det dielektriske materialet er forøvrig ikke helt optimale. Elektrolyttmaterialet vil i de fleste tilfeller være avhengig av å ha det største spenningspotensialet fast på en terminal, noe som gjør elektrolyttkondensatoren polaritetsavhengig. Elektrolyttmaterialet fører også med seg ulemper i forhold til økte ESR-tap, lekkasjestrømmer og kontinuitet i kapasitansverdien. [20] For bruksområder hvor disse elementene vil utgjøre en vesentlig faktor må systemdesigneren ta høyde for de ulemper som kan oppstå og eliminere problemene med støttekretser.

2.3.5 ZENERDIODE

Parallelt med glattekondensator C2 står motstand R3 og zenerdiode Z1 plassert i serie. Zenerdioden er en halvlederkomponent som har til oppgave å gi en tilnærmet konstant utgangsspenning, selv om spenningen som tilføres den og strømforbruket i kretsen skulle variere. Zenerdioden har lik karakteristikk som likeretterdioden for positivt spenningspotensial, men ulik for negative potensial. Ved å påføre positiv spenning på katode-terminalen vil katode-terminalen ha et større elektrisk potensiale enn anode, noe som vil si at anoden har et lavere og dermed negativt spenningspotensial. Zenerdioden står nå i sperreretning og vil blokkere for strømgjennomgang fram til den påførte spenningen på katode når en grenseverdi (Vz). Når denne verdien nås vil zenerdioden åpne for strømgjennomgang, samtidig som spenningsfallet over zenerdioden vil bli tilnærmet konstant likt grenseverdien. For å kunne opprettholde et konstant spenningsfall over zenerdioden er man avhengig av å være i zenerdiodens operative område. Zenerdiodens operative område er bestemt av strømforbruket som kan gå gjennom dioden, med en minimum og en maksimumsverdi. [21]

For å kontrollere at zenerdioden holder seg i det operative området benytter man en seriometstand for å begrense strømgjennomgangen. Seriemotstanden begrenser det totale strømforbruket til zenerdioden, samtidig som den får det resterende spenningsfallet fra glattekondensator C2 påført. [22]

Dette er egenskaper som gjør en seriometstand og zenerdiode ypperlig som spenningsregulator, da zenerdioden gir en tilnærmet konstant utgangsspenning, uavhengig av kretsens strømforbruk, så lenge det er innen det operative området. Zenerdioder eksisterer i en rekke konfigurasjoner, spesifisert for ulike grenseverdier og operative områder.

2.3.6 SPENNINGSREGULATOR

Parallelt med zenerdiode Z1 står kondensator C3, spenningsregulator VR1 og kondensator C4, hver i sin egen parallelleggren. Disse tre komponentene utgjør det siste trinnet i strømforsyningen, og skal sørge for en stabil likespenning på 5 volt levert til utgangen. Kondensator C3 og C4 står plassert henholdsvis parallelt mellom Z1 og VR1, og VR1 og utgangen. De opererer som glattekondensatorer for innførelsspenningen og utførelsspenningen til regulatoren.

Spenningsregulatorens oppgave er å levere en stabil utgangsspenning på en gitt verdi til en last, såfremt lastens strømforbruk er innen det operative området til regulatoren. En spenningsregulator er en integrert krets, som består av en rekke halvlederkomponenter. Spenningsregulatorer eksisterer i en rekke konfigurasjoner, både som lineærregulatorer og vekselregulatorer. Disse to regulatortypene har ulike virkemåter, og derav også ulike operasjonsområder. Faktorer som tilført effekt og avgitt effekt er vesentlig når man skal velge regulatortype for sitt bruksområde. [23]

En lineær regulator leverer en stabil utgangsspenning ved å kompensere for endringer i lasten som vil påvirke strømforbruket. Dette gjøres ved at regulatoren overvåker utgangsspenningen i en feildeteksjonskrets (operasjonsforsterker), og sammenligner den faktiske utgangsspenningen med en spenningsreferanse. Ved avvik mellom den faktiske utgangsspenningen og referansen vil feildeteksjonskretsen sende et signal til en intern regulator,

som så regulerer hvor stor strøm som skal leveres til lasten. Dette sørger for at utgangsspenningen stabiliserer seg etter et kort innsvingningsforløp.

Lineærregulatorer er delt inn i tre hovedgrupper; standard, low dropout og quasi low dropout. Det som i hovedsak skiller disse tre typene er konstruksjonen til den interne regulatorkretsen, regulatorens krav for inngangsspenning og dens hvilestrømegenskaper. [24]

Regulatoren som benyttes for prosjektet er en standard lineær spenningsregulator, levert av Texas Instruments. En standard lineær regulator benytter en NPN Darlington transistor, samt en NPN- og PNP transistor som intern regulatorkrets. Standardregulatoren er den regulatortypen som har det strengeste kravet til stabil inngangsspenning; For at den interne regulatoren skal være i det operative området kreves det at regulatoren påsettes en spenning som tilsvarer spenningen over base-emitter for NPN Darlington transistoren, samt en hvilespenning for kollektor-emitter overgangen. [25] I de fleste tilfeller vil denne verdien tilsvare mellom 2,5 til 3 volt. [24] Med andre ord bør spenningsregulatoren påføres en spenning med verdi 2,5 til 3 volt over den ønskede utgangsverdien for at regulering skal være mulig.

2.4 MIKROKONTROLLERE

2.4.1 ATMEL ATTINY45V OG ATMEGA48V

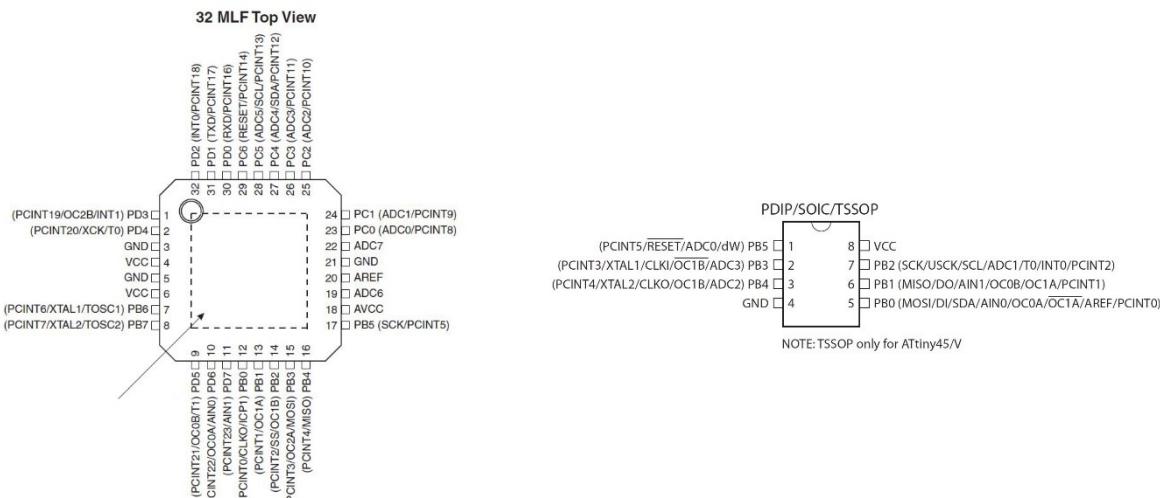
Den første AVR-prosessoren ble utviklet i Trondheim i 1996 av NTNU-studentene Alf-Egil Bogen og Vegard Wollan. En ubekreftet kilde hevder Bogen og Wollan jobbet hos Nordic VLSI (nå kjent som Nordic Semiconductor) i denne perioden og at AVR-prosessoren ble utviklet her. Konseptet ble imidlertid solgt til Atmel, og i 1997 ble den første offisielle 8-bits AVR-arkitekturen lansert. Det mytes at navnet AVR refererer til de første bokstavene i skaperne sine fornavn og kontrollerens prosessorarkitektur, med andre ord Alf-Egils og Vegards RISC-prosessor. Dette er riktig nok ikke bekreftet faktta, men virker troverdig.

Reduced Instruction Set Computing (RISC) er prosessordesignfilosofi som sier at når et simplifisert instruksjonssett består av få, enkle og generelle instruksjoner implementeres i en mikroprosessorarkitektur som kan håndtere disse på en effektiv måte, vil man oppnå høyere ytelse enn ved bruk av et mer komplekst instruksjonssett bestående av mange og svært spesifikke instruksjoner.

ATtiny45V og ATmega48V er begge 8-bits AVR-kontrollere og deler den samme grunnleggende arkitekturen. Tabell 1 viser en oversikt over kontrollernes relevante spesifikasjoner.

	ATtiny45V	ATmega48V
Prosessorarkitektur	8-bit AVR 120 instruksjoner 32x 8-bits register	8-bit AVR 131 instruksjoner 32x 8-bits register
Flash programminne	4 kB	4 kB
EEPROM / SRAM	256 B / 256 kB	256 B / 512 kB
Antall I/O-register	1	3
Antall I/O-pins	6	23
Teller(e)	2x 8-bit	2x 8-bit og 1x 16-bit
TWI	Begrenset	Ja
SPI	Ja	Ja
JTAG-programmering	Nei	Ja
Operasjonsspenning	1,8 – 5,5 V	1,8 – 5,5 V

TABELL 1. SPESIFIKASJONER FOR ATTINY45V OG ATMEGA48V

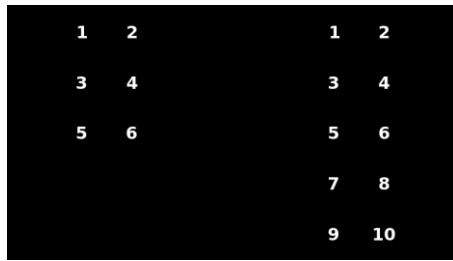


FIGUR 12. ATMEGA48V

FIGUR 13. ATTINY45V

ATmega48V har støtte for både SPI og JTAG-programmering, mens ATtiny45V kun har støtte for SPI-programmering. Seriell innlasting av firmware gjennom SPI settes opp på følgende måte for begge kontrollerne:

1. Forsyningsspenningen VCC slås av.
2. RESET-pinnen (PB5 og PC6 for hhv. ATtiny45V og ATmega48V) kortsluttes til jord.
3. VCC-, GND-, SCK-, MISO- og MOSI-pinnene kobles til respektive pinner på SPI-konnektoren. Se Figur 15Feil! **Fant ikke referansekilden..**
4. Forsyningsspenningen VCC slås på.
5. Firmware lastes inn fra PC gjennom en SPI-kompatibel programmerer (f.eks. JTAGICE v.3).



FIGUR 15. 6-PIN OG 10-PIN SPI-KONNEKTOR

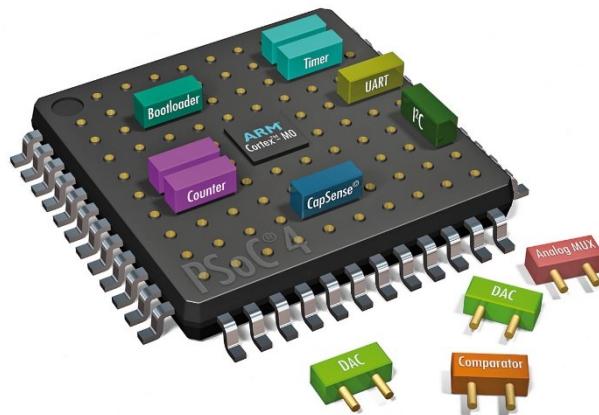


FIGUR 14. ATMEL JTAGICE v.3

2.4.2 ARM CORTEX M0 OG NRF51422

Den første Acorn RISC Machine (ARM)-arkitekturen ble utviklet av selskapet Acorn Computers i Cambridge mot slutten av 1980-tallet. Dagens ARM-arkitekturer eies, utvikles og selges av selskapet ARM Holdings i form av Intellectual Property (IP). Teknologiprodusenter som ønsker å produsere egne mikrokontrollere, men som ikke ønsker å utvikle egne prosessorarkitekturen, kan dermed kjøpe rettigheter til å implementere ARM-arkitekturen i produktene sine. Et veldig relevant eksempel på dette er Nordic Semiconductors nRF51422.

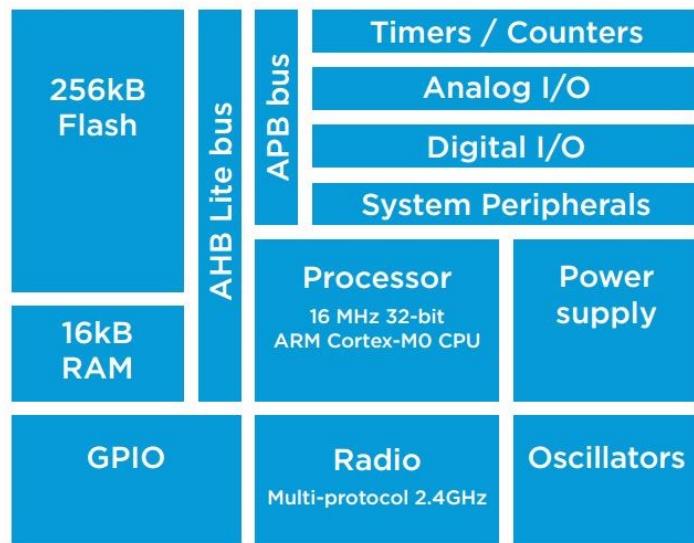
ARM Cortex M0 er en enkel 32-bits RISC-prosessor som er optimalisert for bruk i små og billige mikrokontrollere. Den har kun støtte for et begrenset antall instruksjoner, men er til gjengjeld rask, energieffektiv og har stor ekspansjonsmulighet takket være måten tilleggsmoduler kan implementeres på. Figur 16 illustrerer hvordan ekspansjonsmoduler bygges rundt en Cortex M0-prosessor for å skape funksjonaliteten til en komplett mikrokontroller, i dette tilfellet en Cypress PSoC4.



FIGUR 16. ILLUSTRASJON AV CYPRESS PSoC4

Nordic Semiconductor benytter en lignende metodikk i deres nRF51422, som også bygger på ARM Cortex M0. nRF51422 er en 32-bits multi-protokoll System on Chip (SoC)-løsning for radiosystemer som benytter Bluetooth Smart og/eller ANT. Kontrolleren inneholder blant annet en kraftig radiotransiver med sendestyrke fra -20 dBm til +4 dBm og sensitivitet ned mot -93 dBm for Bluetooth Smart-kommunikasjon. Den inneholder også ekspansjonsmoduler for TWI, SPI, Tellere og ADC, og har i tillegg 31 fullt konfigurerbare GPIO-pinner. Figur 17 gjenspeiler arkitekturen til kontrolleren ved bruk av illustrative funksjonsblokker.

Energieffektivitet er en viktig del av designet, og er avgjørende for at implementering av Bluetooth Smart firmware skal resultere i de lavenergiegenskapene teknologien ble utviklet for å oppnå. Strømforbruk ved tomgang (IDLE) er oppgitt til å være 2 µA.



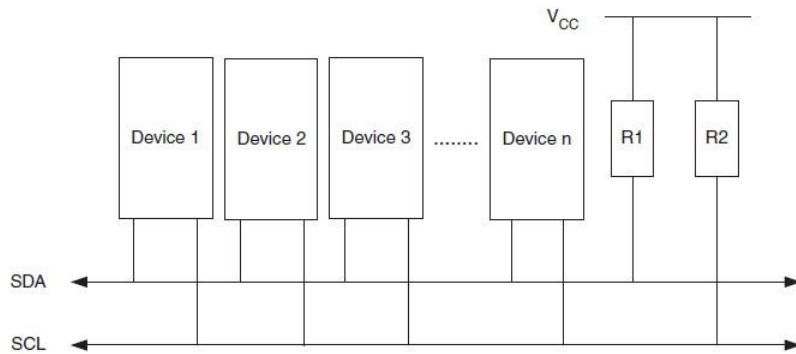
FIGUR 17. INTEGRERTE FUNKSJONSBLOKKER I NRF51422

2.5 KOMMUNIKASJONSMETODER

2.5.1 Two Wire Interface (TWI)

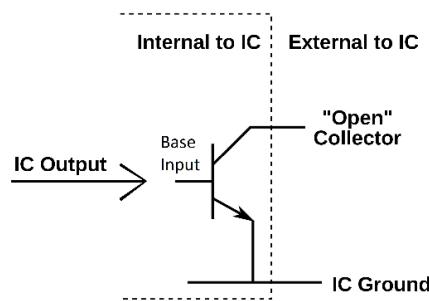
Two Wire Interface (TWI) er et begrep som brukes av ulike teknologiprodusenter for å beskrive et serielt kommunikasjonsgrensesnitt som funksjonelt sett er enten identisk med eller i det minste kompatibelt med Philips Semiconductors (nå kjent som NXP Semiconductors) velkjente I2C-grensesnitt.

Årsaken til at enkelte produsenter velger å benytte TWI i stedet for I2C kan være fordi det åpner for å avvike fra den offisielle I2C-spesifikasjonen der dette er ønskelig og/eller hensiktsmessig. Det spekuleres også i om dette gjøres for å unngå eventuell problematikk knyttet til varemerkebeskyttelse og misbruk av intellektuell eiendom. Det opprinnelige patentet til I2C utløp først i 2004.



FIGUR 18. I2C-BUSS MED N ANTALL ENHETER

I2C, eller *Inter-Integrated Circuit*, er en lavhastighets seriell kommunikasjonsbuss som kun benytter to linjer for å oppnå dataoverføring og synkronisering; *Serial Data Line* (SDA) og *Serial Clock Line* (SCL). Disse to linjene er av typen «Open Drain» og drives av VCC gjennom hver sin pull-up-motstand. «Open Drain» vil si at linjene ikke kobles til jord som en del av bussen, men at de i stedet jordes på kommando gjennom en tilkoblet enhets interne koblinger. Linjene begrenses av en totalkapasitans på 400pF, som i praksis betyr at maksimal linjelengde kun er noen få meter.



FIGUR 19. OPEN DRAIN, TRANSISTORKOBLING

Eksempel: SDA-linjen kobles til «Open Collector» -terminalen på enhet nr.1. Dette gjør det mulig for enhet nr.1 å manipulere spenningsnivået på SDA-linjen ved å styre «Base Input» -signalet til transistoren. Settes «Base Input» høy aktiveres transistoren og SDA-linjen får kobling til jord. De andre enhetene på bussen vil deretter kunne «lese» spenningsnivået på SDA-linjen som $\approx 0\text{ V}$, og man kan dermed si at enhet nr.1 har «kommunisert» med disse enhetene.

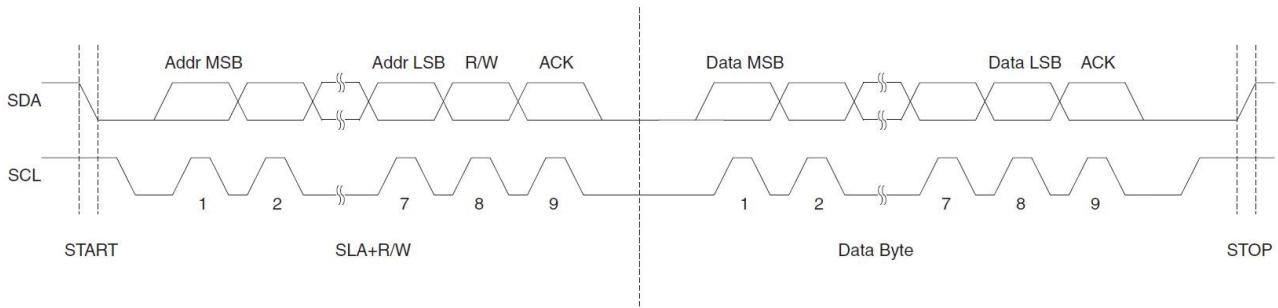
Dette er et simplifisert eksempel som kun er ment å forklare hvordan enheter kan manipulere linjene på bussen og dermed hvordan de kan påvirke hverandre. I realiteten inngår det langt mer enn så i en I2C-kompatibel enhet. Nøyaktig hvilke kontroll-, og deteksjonskretser som benyttes i en gitt enhet avhenger helt av hvilke oppgaver som er tiltenkt denne enheten. Noen kan være utelukkende basert på hardware, som for eksempel EEPROM- eller sensormoduler, mens andre kan inneholde hardware som også krever implementering av software for å kunne håndtere I2C-kommunikasjonen. Mikrokontrollere er et typisk eksempel på sistnevnte.

Felles for alle typer enheter derimot er måten de må operere på for at kommunikasjon skal være mulig. En I2C-buss er tross alt en delt ressurs og må behandles deretter. Det er derfor viktig at alle tilkoblede enheter opererer i henhold til I2C-protokollen slik at det ikke oppstår uønskede situasjoner som for eksempel «låsing» av busslinjene. I2C-protokollen danner dessuten et felles «språk» som forsikrer at alle enheter som bruker denne protokollen forstår hverandre.

I2C-protokollen har kun støtte for to rolletyper, master og slave, men kan inneholde flere av begge typer så lenge bare én kommunikasjonsøkt foregår samtidig. Hver enhet kan adresseres individuelt med en 7 bits adresse, men enkelte adresser er enten reservert eller anbefales ikke å bruke, så det totale antall mulige adresserbare enheter på bussen er begrenset til 119. All kommunikasjon foregår i form av datapakker, der hver datapakke er 9 bit lang og består av 8 payload-bits og 1 ACK-bit. Payload er selve innholdet i datapakken, mens ACK er en bekreftelse som mottaker har ansvar for å sende i retur ved mottatt payload. Data i payload transmitteres med Most Significant Bit (MSB) først og Least Significant Bit (LSB) sist.

En typisk kommunikasjonsøkt mellom to enheter foregår på følgende måte:

1. Masterenheten sender en START-tilstand ved å trekke SDA-linjen lav.
2. Masterenheten starter samtidig klokkegenereringen, men venter én klokkesyklus før dette klokkesignalet påtrykkes SCL-linjen.
3. Master sender deretter en adresseringspakke som inneholder en 7 bit lang adresse til den aktuelle slaveenheten master vil kommunisere med samt en les/skriv-bit (R/W). Er R/W satt høy, vil master skrive data til slave, mens dersom R/W er satt lav vil master lese data fra slave.
4. Oppdager slaven at den har blitt adressert og R/W er lav, har den mottatt SLA+W fra master og vil umiddelbart svare med en lav ACK i 9. klokkeperiode.
5. Oppdager slaven at den har blitt adressert og R/W er høy, har den mottatt SLA+R fra master og vil umiddelbart svare med en lav ACK i 9. klokkeperiode.
6. Dersom SLA+W ble mottatt vil master sende aktuell data til slave i form av x antall datapakker. Slaven vil svare hver mottatte datapakke med en lav ACK.
7. Dersom SLA+R ble mottatt vil slaven sende aktuell data til master i form av x antall datapakker. Master vil svare hver mottatte datapakke med en lav ACK. Når master har mottatt det den anser som siste datapakke vil den i stedet svare med en høy ACK, eller NACK (not-ACK).
8. Master avslutter økten ved å sende en STOP-tilstand. Dette gjøres ved å koble klokkesignalet fra SCL-linjen, vente én klokkesyklus og deretter trekke SDA-linjen høy.

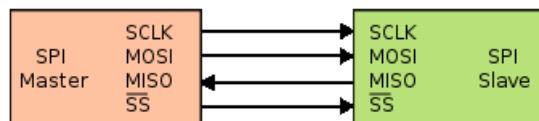


FIGUR 20. TWI, TYPISK TRANSMISJONSSTRUKTUR

2.5.2 SERIAL PERIPHERAL INTERFACE (SPI)

Serial Peripheral Interface (SPI) er en datakommunikasjonsbuss med tilhørende kommunikasjonsprotokoll som tilbyr synkron seriell kommunikasjon for integrerte systemer.

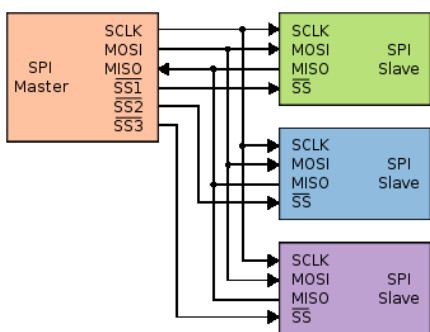
En typisk singel master, singel slave SPI-buss består av fire linjer; Serial Clock (SCK), Master Output Slave Input (MOSI), Master Input Slave Output (MISO) og Slave Select (SS). De to klokkelinjene gjør det mulig for SPI-enheter å kommunisere med full dupleks, altså i begge retninger samtidig, noe TWI / I2C ikke har støtte for.



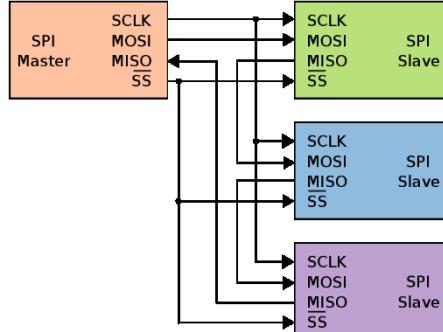
FIGUR 21. SINGLE MASTER, SINGLE SLAVE SPI-BUSS

SPI-busser kan inneholde flere slaveenheter, men er låst til å bare inneholde én masterenhet. En slik multiple-slave-buss kan konfigureres på følgende to måter:

1. Uavhengig slave: Alle slaveene på bussen deler de samme klokke- og datalinjene, men har hver sin egen SS-linje og kan dermed «adresseres» spesifikt. På denne måten kan kommunikasjonen skje direkte mellom master og en gitt slave. Ulempen er at master må ha støtte for like mange SS-linjer som det er slaveenheter på bussen.
2. Kjedekobling: I denne konfigurasjonen deler slaveene samme SCK og SS, men datalinjene kaskadekobles i en kjede. Hver enkel transmisjon skiftes dermed sekvensielt gjennom alle slaveenhetene på bussen før dataene når frem til målet.



FIGUR 23. SPI-BUSS, UAVHENIG SLAVE



FIGUR 22. SPI-BUSS, KJEDEKOBLING

Typiske slaveenheter kan være alt fra LCD-skjermer til sensormoduler. For at kommunikasjonen skal foregå riktig må derfor datatransmisjonen konfigureres riktig. I alt 8 operasjonsmoduser er tilgjengelig og baseres på om klokkesignalet er aktiv lavt eller høyt, hvilke klokkeflanke data skal sendes på og om data sendes med MSB først eller sist.

Selve datatransmisjonen fungerer på følgende måte:

1. Master konfigurerer først parameterne som klokkefrekvens og operasjonsmodus.
2. Master velger deretter hvilken slave den vil kommunisere med ved å trekke aktuell SS lav.
3. Master påtrykker klokkesignalet på SCK-linen og kommunikasjonen starter. For hver klokkepuls sendes én bit over både MOSI og MISO samtidig. Spesifikt hvordan data synkroniseres og behandles avhenger av operasjonsmodusen.
4. Kommunikasjonen stanses ved at master kobler av klokkesignalet fra SCK-linen og tilbakestiller SS.

2.6 JAVA

2.6.1 GENERELT OM JAVA

I 1991 gikk tre amerikanere sammen for å utvikle programmeringsspråket som vi i dag kjenner som Java. Teamet bestod av Mike Sheridan, Patrick Naughton og mannen som har fått mye av æren for prosjektet, James Gosling. [26] Fire år senere ble det annonsert at nettleseren Netscape Navigator Internet Browser ville benytte seg av Java. [27] Dette markerte starten på en enorm utvikling, og i dag benyttes Java i et stort utvalg applikasjoner og elektroniske enheter. Det er altså hovedsakelig dette programmeringsspråket, i tillegg til noe C/C++ og XML, vi har anvendt i utviklingen av Android-applikasjonen vår, som kontrollerer lysstyringssystemet.

2.6.2 DATATYPER

Når man programmerer i Java, jobber man mye med tall. For ordens skyld «gjemmer» man tallene i ord, kalt variabler, slik at det er lettere å holde styr på tallene. Disse variablene må deklarereres med datatyper som sier noe om lengden på variablene og hvor mye minne de opptar. Tabell 2 viser ofte brukte datatyper.

Type Name	Kind of Value	Memory Used	Range of Values
<code>byte</code>	Integer	1 byte	-128 to 127
<code>short</code>	Integer	2 bytes	-32,768 to 32,767
<code>int</code>	Integer	4 bytes	-2,147,483,648 to 2,147,483,647
<code>long</code>	Integer	8 bytes	-9,223,372,036,8547,75,808 to 9,223,372,036,854,775,807
<code>float</code>	Floating-point	4 bytes	$\pm 3.40282347 \times 10^{38}$ to $\pm 1.40239846 \times 10^{-45}$
<code>double</code>	Floating-point	8 bytes	$\pm 1.79769313486231570 \times 10^{308}$ to $\pm 4.94065645841246544 \times 10^{-324}$
<code>char</code>	Single character (Unicode)	2 bytes	All Unicode values from 0 to 65,535
<code>boolean</code>		1 bit	True or false

TABELL 2. DATATYPER I JAVA

Datatypen byte er en integer, som vil si at den består kun av heltall. En byte tilsvarer 8-bit, som gir en range fra -128 til 127 ($2^8 = 256$).

2.6.3 OBJEKTER

Java er et objektorientert programmeringsspråk (OOP), et konsept som kan være vanskelig å forstå. I prosedyreorientert programering er koden skrevet i en lang bokført med funksjoner og rutiner blandet sammen. Med OOP splitter man koden opp i flere ulike *objekter*, som tar seg av forskjellige oppgaver innad i programmet, noe som gjør det ryddigere og skaper bedre oversikt. Et objekt representerer en modell av et objekt i den virkelig verden, enten det er abstrakt eller konkret. Hvert objekt inneholder også sine egne attributter og logikk, som blir kommunisert mellom objektene. [28]

2.6.4 KLASSER OG METODER

En klasse er en oppskrift på hva en bestemt type objekt skal inneholde. Oppskriften sier noe om hvilke egenskaper objektet skal inneholde og hvilke operasjoner som kan utføres. F.eks. så kan klassen fortelle at objektet skal ha et navn, at det skal inneholde bestemte typer attributter og definere hvilke operasjoner det skal være mulig å utføre. Klassen

definerer derimot ikke hva navnet skal være, størrelsen på attributtene eller utfallet av operasjonene, det er det objektet i seg selv som gjør. [28]

2.6.5 METODER

Når man har opprettet ulike objekter og klasser, vil man gjerne at disse skal brukes til noe. Som nevnt tidligere kan man definere operasjoner når man lager en klasse. Disse operasjonene blir til metoder når de utføres på et objekt. [28]

2.6.6 EKSEMPELPROGRAM

For å få en viss forståelse av hvordan objekter, klasser og metoder anvendes i praksis, har vi laget et lite eksempelprogram med forklaring.

```
/* Oppretter klassen "Konto" */
class Konto {

    /* Starter med å initiere tre variabler; KontoNummer, KontoNavn
       og KontoSaldo av henholdsvis datatypen long, String og double */

    private final long KontoNummer ;
    private final String KontoNavn ;
    private final double KontoSaldo ;
```

Her ser vi hvordan klassen Konto opprettes, som ganske enkelt deklarerer med *class* foran navnet. Navnet på klassen kan man velge helt selv, men det er en fordel å bruke noe som har sammenheng med resten av programmet.

Etter at klassen er opprettet starter vi med å lage tre variabler KontoNummer, KontoNavn og KontoSaldo. Det at variablene er private betyr at de skjules i objektene. Final vil si at variablene er *read-only*, eller med andre ord, når variablene har fått sin verdi kan de ikke endres, kun leses.

```
/* Oppretter konstruktøren "Konto" som tar inn tre variabler,
   de tre som allerede er blitt initiert */

public Konto(long KontoNummer, String KontoNavn, double KontoSaldo) {
    this.KontoNummer = KontoNummer;
    this.KontoNavn = KontoNavn;
    this.KontoSaldo = KontoSaldo;
}
```

Videre oppretter vi *konstruktøren* Konto. En konstruktør brukes til å lage et objekt av klassen. For hver gang et objekt blir opprettet av denne konstruktøren, forventes det at de tre variablene skal bli fylt med verdier. At Konto er public vil si at innholdet ikke gjemmes inni objektet, men er tilgjengelig utenfor objektet.

```
/* Oppretter metoden "getKontoSaldo" som returnerer  
verdien i variabelen "KontoSaldo" */  
  
public double getKontoSaldo() {  
    return KontoSaldo;  
}  
  
}
```

Til slutt oppretter vi metoder. Over ser vi metoden getKontoSaldo. get, engelsk for «hent», er standard javapraksis, men man kunne like gjerne kalt den noe annet. Så det som egentlig skjer når getKontoSaldo-operasjonen kalles opp er at vi ber om at verdien i variabelen KontoSaldo skal hentes. Man får da den nevnte verdien i retur.

```
/* Oppretter klassen "BankKonto" */  
class BankKonto {  
  
    /* Deretter kommer mainfunksjonen, alt som skal utføres i programmet skjer her */  
    public static void main(String[] args) {  
  
        /* Oppretter objektet "MinEgenKonto" med KontoNummer: "123456789", KontoNavn:  
        "Ola Nordmann", og KontoSaldo: "50.25" */  
        Konto MinEgenBankKonto = new Konto(123456789, "Ola Nordmann", 50.25);  
  
        /* Utfører operasjonen getKontoSaldo, og putter den returnerte verdien inn i  
        en ny variabel kalt "Saldo" */  
        double Saldo = MinEgenBankKonto.getKontoSaldo();  
  
        /* Skriver ut teksten "Saldoen på kontoen din er:" etterfulgt av verdien  
        for Saldo */  
        System.out.println("Saldoen på kontoen din er: " + Saldo);  
    }  
}
```

Her ser vi et program der vi oppretter et objekt som heter MinEgenKonto av klassen Konto. Som vi så av konstruktøren tidligere skulle det være tre variabler som måtte fylles. I dette eksemplet har vi bare valgt helt tilfeldige tall, kun for å forklare hvordan oppretting av objekter virker.

Videre ser vi at en metode utføres. Vi ønsker å vite saldoen på MinEgenBankKonto, for så å skrive den ut til skjerm.

Eksemplet som er gjennomgått er veldig simpelt, men demonstrerer bruken av objekter, klasser og metoder på en god måte.

```
/* Oppretter klassen "Konto" */
class Konto {

    /* Starter med å initiere tre variabler; KontoNummer, KontoNavn
     og KontoSaldo av henholdsvis datatypen long, String og double */

    private final long KontoNummer ;
    private final String KontoNavn ;
    private final double KontoSaldo ;

    /* Oppretter konstruktøren "Konto" som tar inn tre variabler,
     de tre som allerede er blitt initiert */

    public Konto(long KontoNummer, String KontoNavn, double KontoSaldo) {
        this.KontoNummer = KontoNummer;
        this.KontoNavn = KontoNavn;
        this.KontoSaldo = KontoSaldo;
    }

    /* Oppretter metoden "getKontoNummer" som returnerer
     verdien i variabelen "KontoNummer" */

    public long getKontoNummer() {
        return KontoNummer;
    }

    /* Oppretter metoden "getKontoNavn" som returnerer
     teksten i tekstvariabelen "KontoNummer" */

    public String getKontoNavn() {
        return KontoNavn;
    }

    /* Oppretter metoden "getKontoSaldo" som returnerer
     verdien i variabelen "KontoSaldo" */

    public double getKontoSaldo() {
        return KontoSaldo;
    }
}
```

2.7 ANDROID

2.7.1 HISTORIE OG GENERELT OM UTVIKLING

Selskapet Android Incorporation ble opprettet i 2003 av Andy Rubin, Rich Miner, Nick Sears og Chris White. Senere kjøpte Google opp selskapet, og i 2008 kom den første mobile enheten som benyttet Android som operativsystem. [29] Utviklingen har vært stor siden den gang, og Android er per dags dato det mest installerte operativsystemet i verden for mobile enheter. [30] Kildekoden er åpen, det vil si at oppskriften til hvordan man kan utvikle Android programvare (Apps) er gratis tilgjengelig for alle som skulle ønske å benytte seg av den. [31]

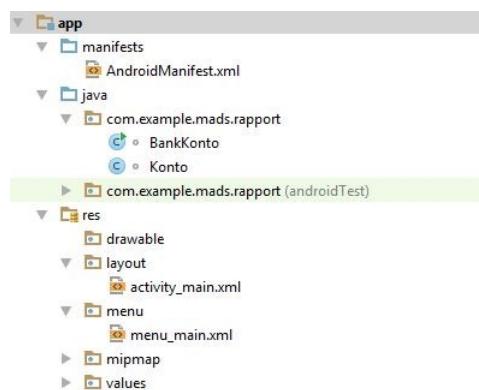
Android er basert på Linux kernelen. En kernel er kildekode som har som funksjon å være et bindeledd mellom hardware og software. Når man trykker på skjermen til en smarttelefon, f.eks. at man ønsker å justere lysnivået på skjermen, sendes det en forespørrelse fra software via kernel til hardware. Dette gjøres for å slippe å skrive mengder med kode for hver enkel handling og kode som er unik for hver enkel hardware-del. Utviklere trenger derfor kun å skrive kode slik at hardware og software kommuniserer med kernel. [32]

Operativsystemet tillater bruk av applikasjoner, eller såkalte «apps». Dette kan blant annet være spill, nytteverktøy (eksempler; kalkulator, lommelykt) eller snarveier til sosiale medier. Programmeringsspråkene Java, C / C++ og XML brukes til utvikling av apps. Hvilket IDE (Integrated Development Environment), eller enklere forklart, hvilken programvare man ønsker å bruke til utvikling har man mulighet til å velge selv, da det finnes flere forskjellige IDE-er. Android Studio er et eksempel på et IDE, og er det offisielle programmet brukt til apputvikling, utgitt av Android selv. Andre eksempler på IDE er Eclipse og NetBeans. [33]

Android tilbyr også omfattende utviklingsverktøy med deres SDK (Software Development Kit), som består av blant annet en debugger, som man kan bruke til feilsøking dersom ikke appen fungerer fullstendig. En emulator er også inkludert, som brukes til testing av apps. Dessuten følger det med eksempekkode og øvinger i SDK-et. [34]

2.7.2 MAPPESTRUKTUR OG FILORGANISERING

Figur 24 viser mappe- og filstrukturen når man lager en applikasjon i Android Studio. Det er tre hovedmapper; *manifest*, *java* og *res*. I manifest-mappen finner vi filen *AndroidManifest.xml*. Dette er en fil alle applikasjoner må inneholde, og man kan ikke forandre navnet på den. Innholdet i manifest-fila er informasjon om applikasjonen som er kritisk for at den skal kunne kjøre på Android-enheter. [35] Java-mappen inneholder filer med forskjellige klasser og programkode. Til slutt ligger mappen *res*, simpelthen en forkortelse for resources. I denne mappa skal alt av grafisk tilbehør til appen ligge. [36]



FIGUR 24. ANDROID STUDIO, MAPPE- OG FILSTRUKTUR

2.7.3 AKTIVITETER

En *aktivitet* (*activity*) er en applikasjonskomponent som sørger for at brukeren av applikasjonen kan se et skjermbilde med innhold. Innholdet kan være enkel tekst, men også knapper og andre elementer. Ved å trykke på en av disse elementene, er det mulig at nye aktiviteter blir åpnet, alt etter hvordan applikasjonen er designet. I det man starter en app er det første man møter en aktivitet, dette kan være en splash screen, som er en logo eller et bilde som varer i noen sekunder før man blir tatt videre til hovedaktiviteten. Mange utviklere velger derimot at det første som skal vises er hovedaktiviteten, ofte kalt «main activity», for å spare ressurser. Når man oppretter nye aktiviteter må de deklarerdes i AndroidManifest-fila.

For å unngå at applikasjonen man lager opptar unødvendige systemressurser implementerer man en livssyklus til en aktivitet. Ved bruk av denne metoden definerer man hva som skal skje når aktiviteten åpnes, blir pauset eller avsluttet.

onCreate()

For hver gang en aktivitet blir startet for første gang, blir denne metoden kalt opp. Aktiviteten klargjøres før den vises til brukeren. Her definerer man hvilken layout aktiviteten skal ha ved oppstart

onRestart()

Metoden blir kalt opp dersom det har vært stopp i aktiviteten.

onStart()

Metoden blir kalt opp rett før aktiviteten er synlig til brukeren

onResume()

Metoden blir kalt opp dersom det har vært pause i aktiviteten. Her kan man f.eks. fortsette oppgaver som pågikk før aktiviteten ble pauset.

onPause()

Metoden kalles når man setter aktiviteten på pause. En grunn til at en aktivitet blir pauset kan være at man starter en annen aktivitet, eller at telefonen går i sleep modus. Det vil f.eks være unødvendig å kjøre en Bluetooth-scan, dersom aktivitetsinduet ikke vises. Derfor har man muligheten med denne metoden å avslutte ressurskrevende oppgaver rett før aktiviteten blir pauset.

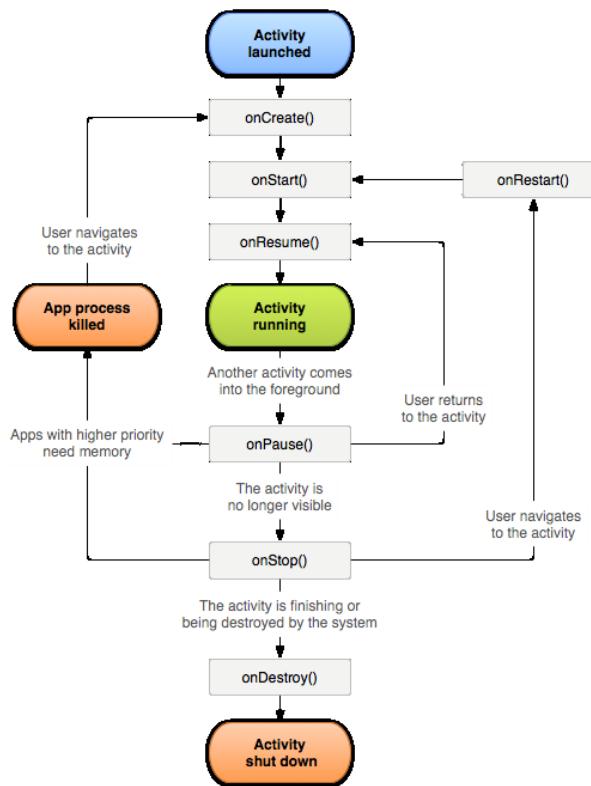
onStop()

Metoden kalles opp rett før man er ferdig med å bruke aktiviteten.

onDestroy()

Metoden kalles opp når man er ferdig å bruke aktiviteten, slik at den ikke kjører i bakgrunnen og tar opp ressurser.

I hver av disse metodekallene har man mulighet til å definere hva som skal skje, hver gang aktiviteten havner i en spesifisert tilstand. [37]



FIGUR 25. ANDROID, FLYTSKJEMA

2.7.4 INSTALLASJON AV APPLIKASJONER

Utvikling av applikasjoner foregår som regel på PC, derfor må man overføre den til smart-telefonen før man kan ta den i bruk. Dette kan gjøres på ulike måter. Dersom man selv har utviklet applikasjonen, har Android Studio en innebygd funksjon som lar deg installere applikasjonen på telefonen din. For at andre skal kunne bruke den finnes det to enkle måter man kan installere en applikasjon på. Man kan laste opp applikasjonen til Google Play, som er Google's eget applikasjonsmarked, hvor hvem som helst kan laste ned det du har laget. [38] Det koster riktignok 25 dollar å opprette en Google Play utviklerkonto. [39]

Et gratis alternativ er å laste over en .apk-fil via USB til smart-telefonen. Dette er en installasjonsfil som legger inn applikasjonen for deg.

2.8 BLUETOOTH SMART

2.8.1 KORT OPPSUMMERING

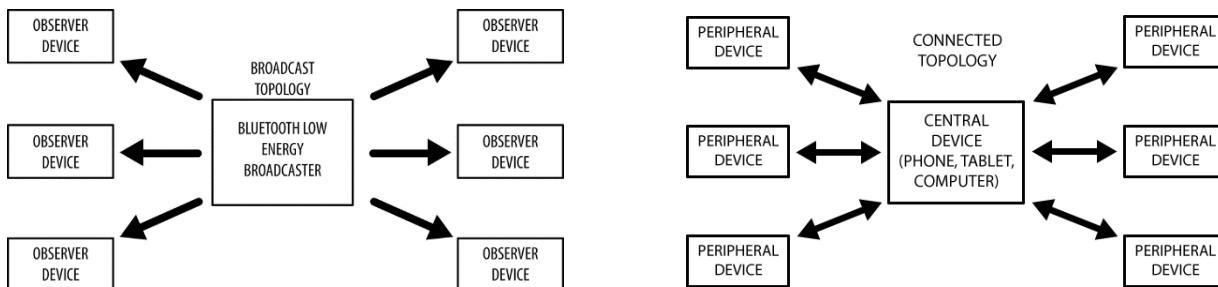
Bluetooth Smart, tidligere kjent som Bluetooth Low Energy (BLE), er en trådløs kommunikasjonsteknologi designet for bruk i personlige kortavstandsnettverk (WPAN) i 2,4 GHz-båndet. Teknologien er utviklet med tanke på energieffektivitet og egner seg best i applikasjoner hvor små datamengder sendes med pauser i mellom hver transmisjon.

Bluetooth Smart ble opprinnelig utviklet av Nokia under navnet Wibree i perioden 2001 til 2007. Etter samtaler med Bluetooth Special Interest Group (SIG) i 2007 ble det avgjort at teknologien skulle inkluderes i den offisielle Bluetooth-spesifikasjonen under navnet Bluetooth Smart. Tre år senere, i 2010, ble Bluetooth Smart lansert som en del av Bluetooth Core Specification 4.0.

Android (4.3 og nyere) og Apple iOS (7 og nyere), har begge full støtte for Bluetooth Smart. Dette kombinert med et forenklet rammeverket sammenlignet med andre typer Bluetooth, gjør at Bluetooth Smart har en unik posisjon i forhold til utvikling av systemer som skal samhandle med smarttelefoner og tablets.

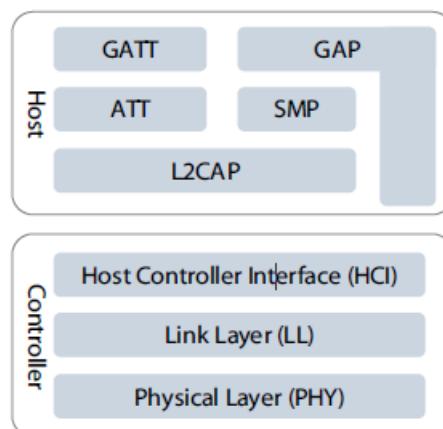
Kommunikasjon mellom BLE-kompatible enheter kan foregå i form av to nettverkstopologier:

- Broadcasting: Har støtte for to enhetsroller, Broadcaster og Observer, og brukes for å oppnå koblingsfri envegskommunikasjon mellom Broadcaster og samtlige Observere innenfor rekkevidde. Se Figur 27.
- Connected: Har støtte for to enhetsroller, Central og Peripheral, og brukes for å oppnå koblet tovegskommunikasjon mellom to - og bare to - enheter av gangen. Se Figur 26.



FIGUR 27. BLE, BROADCAST-TOPOLOGI

FIGUR 26. BLE, CONNECTED-TOPOLOGI



FIGUR 28. BLUETOOTH SMART, PROTOKOLLSTAKK

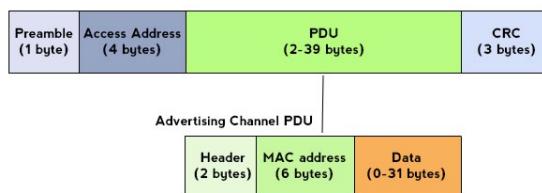
2.8.2 GENERIC ACCESS PROFILE (GAP)

Generic Access Profile (GAP) er et rammeverk som definerer og kontrollerer hvordan enheter oppdager hverandre, kobles sammen og samhandler. Enhetenes rolletyper (Central, Peripheral, Broadcaster og Observer) og kommunikasjonsmoduser er blant parameterne som defineres i GAP.

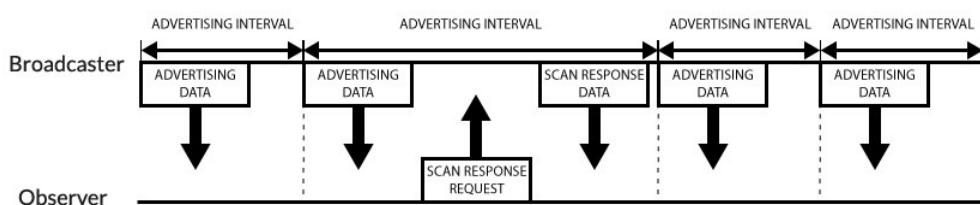
Rolletype	Forklaring
Broadcaster	Konfigureres som sender i en broadcast-topologi. Har i oppgave å sende periodiske advertising-pakker til Observere i nettverket. Må også svare på forespørslar om scan response med en scan response-pakke.
Observer	Konfigureres som mottaker i en broadcast-topologi. Har i oppgave å lytte etter advertising-pakker og kan i tillegg be om ytterligere informasjon ved å sende en scan response-forespørsel.
Central	Fungerer som masterenhet i en connected-topologi, og er ofte en smarttelefon eller lignende. Kan holde koblinger til flere peripheral-enheter, men har kun støtte for én aktiv kobling av gangen.
Peripheral	Fungerer som slaveenhet i en connected-topologi og er ofte en liten, spesialisert og energieffektiv enhet. Peripheral-enheter bruker advertising for å bli oppdaget av central-enheter.

TABELL 3. BLUETOOTH SMART, OVERSIKT OVER ENHETSROLLER

Advertising er den eneste måten enheter kan oppdage hverandre på og er derfor en helt elementær del av Bluetooth Smart. En advertising-pakke består av maksimalt 47 bytes, hvorav totalt 31 bytes kan brukes til nyttedata. Se Figur 29. I applikasjoner som utelukkende benytter broadcast-topologi, vil advertising- og scan response-pakker være eneste form for datatransmisjon. En typisk nyttedatapakke består av enhets navn i full- eller kortform, advertising-flagg, en service liste og manufacturer spesific data som f.eks. kan være sensordata eller lignende. Figur 30 illustrerer et typisk advertising-forløp. Scan response-pakker er formatert på samme måte som advertising-pakker og kan brukes til å sende ekstra datamengder, men sendes bare etter forespørsel fra en observer-enhet. Merk at advertising-pakker er obligatoriske, mens scan response-pakker er valgfrie.



FIGUR 29. BLUETOOTH SMART, PAKKESTRUKTUR FOR ADVERTISING



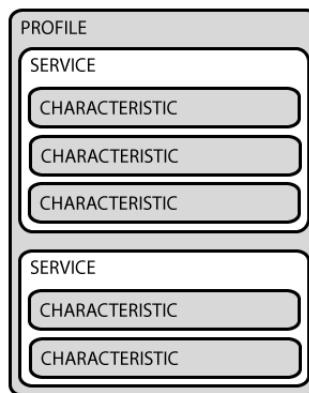
FIGUR 30. BLUETOOTH SMART, EKSEMPELFORLØP FOR ADVERTISING

2.8.3 GENERIC ATTRIBUTE PROFILE (GATT)

Når en applikasjon som benytter connected-topologi har fullført advertising, og en tilkobling har blitt etablert mellom Central og Peripheral, trer Generic Attribute Profile (GATT) i kraft. Dette protokollaget har i oppgave å definere hvordan brukerdata struktureres og sendes over den etablerte koblingen.

For at kommunikasjon mellom de to tilkoblede enhetene skal kunne forgå, kreves det at all data struktureres i et strengt hierarkisk system. Se Figur 31.

Characteristics betegner individuelle dataverdier som for eksempel kan være data fra en lyssensor, eller en verdi som kontrollerer et lys. Services er logiske grupperinger av en eller flere characteristics som naturlig hører sammen med hverandre. Profiles er fastsatte samlinger av services som enten Bluetooth SIG eller andre fabrikantér har laget. Heart Rate Profile er et eksempel på sistnevnte.



FIGUR 31. BLUETOOTH SMART, GATT-HIERARKI

Både Characteristics og Services refereres til ved hjelp av en 16-bits eller 128-bits Universally Unique Identifier (UUID), det vil si at hver enkel characteristic og hver enkel service blir knyttet opp til egne UUID-er. I Bluetooth Smart er alle 16-bits UUID-er reservert for standard-characteristics definert av Bluetooth SIG, men hvem som helst kan opprette egne 128-bits UUID-er.

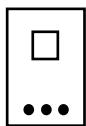
KAP.3 HARDWARE

3.1 SYSTEMLØSNING

Dette underkapitlet tar for seg det utviklingen av systemoppsettet for hardware, fra prosjektets oppstart til endelig design. Vi tar her for oss sentrale enheter i vår hardware-løsning, og gir begrunnelse på vesentlige valg ved løsningen.

3.1.1 ENDRINGER FRA OPPRINNELIG LØSNING

Prosjektgruppens forprosjektrapport har presentert et teknisk løsningskonsept for hardware, og bruk av løsningskonseptet i produkter og demonstrasjonsenheter. Etter hvert som utviklingsarbeidet har tatt til har vi etter rådføring med oppdragsgiver besluttet å endre elementer av hardwarestrukturen. Den endelige hardwarestrukturen er presentert her:



KONTROLLMODUL

Prosjektgruppen har valgt å gå bort fra utvikling av egen hardware for kontrollmodul, og benytter i stedet en nRF51 dongle som kontrollmodul. nRF51 donglen inneholder blant annet nRF51422 og en rekke andre støttekomponenter;

Donglen håndterer BLE-kommunikasjon og genererer styresignaler for lokal hovedkontroller i binærmodul eller Dimmermodul. Tjener også andre formål enn bare lysstyring ved at den opererer som en uavhengig enhet. Kan dermed benyttes i en rekke andre applikasjoner (bruksområder).



BINÆRMODUL

Prosjektgruppa har utviklet en modul med binær bryterkrets;

Mottar styresignal fra kontrollmodul for deretter å skru utgangen av/på ved hjelp av relé.



DIMMERMODUL

Prosjektgruppa har utviklet en modul som inneholder dimmerkrets;

Mottar styresignal fra kontrollmodul for deretter å utføre dimming eller skru av/på strøm på utgangen.



SWITCHMODUL

Prosjektgruppa har koblet opp en Switchmodul på breadboard og har brukt denne til testing av blant annet firmware, se Kap. 4.4. Vi har ikke prioritert å utvikle hardware for denne modulen i like stor grad som binær- og Dimmermodul og har derfor ingen konkrete resultater å vise til, annet enn at switchen har vært til stor hjelp under utvikling og testing av firmware.

FIGUR 32. KONTROLLMODUL

FIGUR 33. BINÆRMODUL

FIGUR 34. DIMMERMODUL

FIGUR 35. SWITCHMODUL

3.1.2 FYSISK GRENSESNIITT

Forprosjektrapporten har fastsatt at prosjektgruppen ønsker å konstruere veggplugg for henholdsvis binær- og dimmermodul som førsteprioritet. Veggpluggene som brukes til referanse er Nexa EYCR-2300 og Nexa EYCR-201, som opererer likt veggplugger med henholdsvis av/på-funksjon og dimmefunksjon. Ettersom de to enhetene har samme fysiske størrelse velger vi kun å referere til EYCR-2300 for videre bruk i prosjektet. Vår binær- og dimmemodul må ha en fysisk utforming som gjør at de passer inn i chassiset til Nexa EYCR-2300 [38]. Dette gir begrensninger med hensyn på den fysiske størrelsen til kretskortet, samtidig som man må ta hensyn til den fysiske størrelsen til komponentene som skal benyttes. Prosjektgruppen må benytte seg av både overflate- og hull-monterte komponenter, og kombinere bruken av disse i en form som gjør at modulene tar minst mulig fysisk plass.

Fysisk størrelse på kretskort: $45,1 * 48,1 \text{ [mm]}$

Høyde i Nexa-chassis: tilnærmet 23 [mm]

3.1.3 VALG AV RELÉ-TYPE

Den mest kjente topologien for å styre spenningstilførsel helt av eller på er ved hjelp av relé. Det anses derfor at reléstyring er den mest hensiktsmessige løsningen å benytte seg av for prosjektgruppen. Reléer eksisterer i flere konstellasjoner, hvor elektromekaniske, reed-reléer og Solid-State-reléer er de mest utbredte. Et elektromekanisk relé kan konstrueres for slå av og på en forholdsvis stor strøm, men da det inneholder bevegelige deler vil det være følsomt for eksternt påtrykte bevegelser. Hverken reed- eller solid-state-reléet inneholder bevegelige deler, men de er mer ømfintlige for å slå av og på store strømmer og vil kunne ta varig skade av variasjoner i spenningen når de slås av og på [1]. Vi velger å benytte et elektromekanisk relé fordi den type teknologi benyttes i lignende styresystemer, samtidig som lysstyringssystemet ikke vil utsettes for eksterne bevegelser.

3.1.4 VALG AV DIMMERTYPE

Dimming av lys for boliginstallasjon utføres i hovedsak ved hjelp av mosfet-transistorer eller triac. Under prosjektets studiefase har det blitt klart at den mest benyttede teknologien for dimming av lys i hovedsak er triac, og som en direkte konsekvens av dette er hovedvekten av all støttelitteratur for dimming av lys basert på triacsystemer. Ettersom en triac også muliggjør en bedre utnyttelse av den påførte vekselspenningen i forhold til en mosfet-transistor velger prosjektgruppen å benytte seg av triacstyring for det videre utviklingsarbeidet.

Som det ble nevnt i Kap. 2.2.2 vil man være avhengig av et styresystem ved dimming av lys, og prosjektgruppen har valgt et digitalt styresystem, både for Binær- og Dimmermodul. I og med at det digitale styresystemet opererer på en 5 volts spenningsforsyning er det ønskelig å isolere forsyningen fra vekselspenningstilførselen, en utfordring vi har løst ved å benytte optokoblere.

For nullgjennomgangsdeteksjon benyttes H11AA1, konfigurert med to likeretterdioder i antiparallellel på primærsiden og npn-transistor på sekundærside. Primærsiden vil emittere et optisk signal over til basen på transistoren når den tilføres spenning, og signalet til basen vil brytes hver gang spenningen krysser nullpunktet. Ved å tilføre en spenning til kollektor og jorde emitter kan man ta ut styresignalet for nullgjennomgangsdeteksjon over kollektor. Se Vedlegg 2.5 for skjemategning.

Vi velger å bruke MOC3023 optodiac for å styre triacen, bestående av en lysdiode på primærsiden og en diac på sekundær siden. Vi tilfører primærsiden et triggersignal fra lokal hovedkontroller, forsterket gjennom transistorkrets Q1. Ved påførsel av triggersignal vil lysdioden emittere et lys til sekundær siden, hvor diacen på sekundærkretsen starter å lede. Diacen står tilkoblet forsyningsspenningen og gir et styresignal til gate på triac når den starter å lede.

3.1.5 VALG AV INTERN STRØMFORSYNING

For å drive den interne styrekretsen i de forskjellige modulene, trengs det en intern strømforsyning. Ettersom vi har et begrenset fysisk grensesnitt å utfolde oss på er vi nødt til å konstruere strømforsyningen i så liten fysisk størrelse som overhodet mulig. Den mest benyttede typen strømforsyning i dag er switch-mode-forsyningen, som er konfigurerbar etter bruksområde og leverer en variabel mengde effekt i forhold til sin konfigurasjon. Switch-mode-forsyningen vil i mange tilfeller også inneholde en skilletransformator, og regnes derfor som sikker til bruk i forbrukerelektronikk da jordingstilkoblingen i det påkoblede utstyret skiller fra kraftverkets tilførte jordforbindelse. Fordi switch-mode-forsyningen inneholder en vesentlig mengde komponenter for å være sikker og effektiv i bruk vil den også opppta mye fysisk plass, i mange tilfeller for stor plass.

En kapasitiv strømforsyning vil på grunn av sin noe enklere konfigurasjon inneholde færre komponenter og på grunn av det miste funksjoner som tilbakekobling og galvanisk skille, samtidig som den ikke kan levere like mye effekt som en switch-mode-forsyning. En kapasitiv strømforsyning vil allikevel kunne operere stabilt og levere en anseelig mengde effekt, samtidig som den tar opp minst fysisk plass av de to. På dette grunnlaget velger vi å basere den interne strømforsyningen på en kapasitiv modell, da vi mener at den skal tjene formålet vårt.

3.1.6 VALG AV LOKAL HOVEDKONTROLLER

Siden systemløsningen vår i stor grad baserer seg på modularitet er vi avhengige av en sentral enhet som kan utøve intern databehandling og kontrollere lyset, samtidig som den skal utøve kommunikasjon med andre tilkoblede moduler. Vi har valgt en løsning basert på Atmels AVR mikrokontrollere, i hovedsak for å sikre interoperabilitet mellom de forskjellige modulene da det eneste som vil være nødvendig for å kommunisere mellom modulene er en I2C-buss. Vi har også noe erfaring med bruk av denne type mikrokontroller fra tidligere prosjekter, og synes det er spennende å bruke denne kunnskapen til å utforske bruk av mikrokontrollere i nye nyttefelt.

3.2 DIMENSJONERING AV KAPASITIV STRØMFORSYNING

3.2.1 EFFEKTBEREGNINGER

Den interne strømforsyningen skal forsyne både styrekretsen og nRF51-donglen, og må derfor dimensjoneres etter effektforbruket både for Binær- og Dimmermodul. Følgende effektberegnung er å anse som modulenes maksimale potensielle effektforbruk, og vil ikke gjenspeile deres faktiske effektforbruk ved normal drift.

Merk: Effektforbruket for nRF51-donglen er ikke å oppdrive i de databladene som ligger tilgjengelig på nåværende tidspunkt. Vi tar derfor utgangspunkt i effekttallene til nRF51422 ved radiotransmisjon, da vi anser dette som tilstrekkelig. [40] Dette dekker ikke nRF51-donglens effektforbruk i sin helhet, da forbruk for blant annet Segger J-link programmeringschip ikke er medregnet. Effektforbruk av denne art, i tillegg til resistanstap i kondensatorer vil medregnes i en sikkerhetsmargin som legges til det totale effektforbruket.

Komponent- navn	RefDes	Effektforbruk Pd [mW]	Kommentar
Atmel ATMEGA48PV-10AU	MCU	60	@ 8 Mhz crystal, VCC = 5V, ICC = 12 mA
nRF51422		35,4	Tx = 4 dBm, I = 11,8 mA VDD = 3V
TI UA78M05CDCYG3	VR1	60	V _{in} = 10 V, I _{bias} = 6 mA
Omron G5Q-1AEU 5DC	K1	200	V _{in} = 5 V, I = 40 mA
Total		355,4	
Sikkerhetsmargin		10 %	
Total effekt med sikkerhetsmargin		391	

TABELL 4. EFFEKTBEREGNING FOR BINÆRMOUL

Med en likeretterbro med effektivitet på 81,2 % trengs følgende effekt levert fra den kapasitive spenningsdeleren:

$$P_d \text{total} = 391 * \frac{1}{0,812} = 482 \text{ mW}$$

Komponent- navn	RefDes	Effektforbruk Pd [mW]	Kommentar
Atmel ATMEGA48PV-10AU	MCU	60	@ 8 Mhz crystal, VCC = %V, ICC = 12 mA
nRF51422 chip		35,4	Tx = 4 dBm , I = 11.8 mA, VDD = 3 V
TI UA78M05CDCYG3	VR1	60	V in = 10 V, Ibias = 6 mA
FS H11AA1M	OPC1	150	Ic = 50 mA
FS MOC3023SR2M	OPC2	100	Ic = 60 mA
Total		405,4	
Sikkerhetsmargin		10 %	
Total effekt med sikkerhetsmargin		446	

TABELL 5. EFFEKTBEREGNING FOR DIMMERMODUL

Med en likeretterbro med effektivitet på 81,2 % trengs følgende effekt levert fra den kapasitive spenningsdeleren:

$$P_d \text{total} = 446 * \frac{1}{0,812} = 549 \text{ mW}$$

3.2.2 DIMENSJONERING AV FILMKONDENSATOR

Strømgjennomgangen i den interne strømforsyningen begrenses i størst grad av den kapasitive spenningsdelen, representert med kondensator C1. Strømgjennomgangen i C1 er representert med formelen:

$$I = C * 2 * \pi * f * V$$

Løst med hensyn på kapasitansen for C1 blir formelen:

$$C = \frac{I}{2\pi f V} = \frac{\frac{P}{U}}{2\pi f V}$$

Hvor U representerer spenningen på utgangen av spenningsregulatoren og V representerer spenningen påført den kapasitive spenningsdelen. Vi kan nå beregne filmkondensatorens kapasitansverdi for henholdsvis Binær- og Dimmermodul:

$$C_{1\text{binær}} = \frac{\frac{482\text{ mW}}{5\text{ V}}}{2\pi f 50\text{ Hz} 230\text{ V}} = 1,33 * 10^{-6}\text{ F} \approx 1,3\text{ }\mu\text{F}$$

$$C_{1\text{dimmer}} = \frac{\frac{549\text{ mW}}{5\text{ V}}}{2\pi f 50\text{ Hz} 230\text{ V}} = 1,52 * 10^{-6}\text{ F} \approx 1,5\text{ }\mu\text{F}$$

Vi velger på grunnlag av beregningene å benytte filmkondensatorer med kapasitansverdi 1,5 uF for både Binær- og Dimmermodul. Grunnet filmkondensatorens konstruksjon (se Kap. 2.3.2) vil den også ta opp så stor fysisk plass i modulene at den ikke kan konstrueres men noen høyere kapasitansverdi.

3.2.3 VALG AV REGULATORTYPE

Som det ble nevnt i Kap. 2.3.6 eksisterer det tre hovedtyper lineære spenningsregulatorer, hver type med sine formål. Ettersom vår krets skal være drevet av nettspenning 230 Volt RMS er ikke den minste mulige inngangsspenningen til regulatoren en kritisk faktor, og vi velger derfor å benytte oss av en lineær standard regulator. For applikasjoner der regulatoren skulle blitt drevet av en batterikilde hadde for eksempel en low-dropout regulator vært mer egnet. [24]

Vår standardregulator må også dimensjoneres etter effektforbruket i kretsen, en parameter som kan anses å være en mer kritisk faktor. Vi ser fra Kap. 3.2.1 at kretsens maksimale potensielle effektforbruk vil være ca. 405 mW uten sikkerhetsmargin. Ved å se bort fra regulatorens interne effektforbruk kan vi finne effektforbruket for delen av kretsen som kun opererer med 5 volt spenningsnivå, og dimensjonerer regulatoren etter følgende effektforbruk:

$$\text{Total effekt - effekt regulator} = \text{Effekt levert fra regulator}$$

$$405,4\text{ mW} - 60\text{ mW} = 345\text{ mW} + 10\% \text{ sikkerhetsmargin} = 380\text{ mW}$$

Sett fra regulatorens ståsted tilsvarer dette en strøm pålydende:

$$I = \frac{P}{U} = \frac{380\text{ mW}}{5\text{ V}} = 76\text{ mA}$$

Regulatoren må derfor dimensjoneres for å leve en strøm på minst 76 mA.

3.2.4 DIMENSJONERING AV ZENERDIODE

Zenerdioden skal gi en tilnærmet konstant spenning for inngangen til spenningsregulatoren og må derfor dimensjoneres med hensyn på sin spenningsverdi og strømegenskaper, slik at regulatoren kan operere i sitt anbefalte arbeidsområde. For vår regulator er den anbefalte inngangsspenningen mellom 7 og 25 volt, og vi dimensjonerer derfor zenerdioden til å leve en spenningsverdi så tett opp mot den maksimalt anbefalte verdien som mulig. [41]

Vi velger å benytte zenerdiode 1N5932B for bruk i vårt prosjekt, da den gir en typisk spenningsverdi på 20 volt $\pm 5\%$, samtidig som den er kapabel til å forsyne regulatoren med opptil 3 W effekt.

3.2.5 DIMENSJONERING AV GLATTEKONDENSATOR

Glattekondensatorens kapasitansverdi dimensjoneres vanligvis etter hvor stor rippel som tillates i spenningen inn til påsatt last. For vårt tilfelle vil zenerdiode Z1 og motstand R3 operere som lasten til kondensatoren, og det er derfor vesentlig å se på hvilke spenningsnivåer som ligger over disse to komponentene for å beregne kapasitansverdien til C2.

$$V_{C2}(t) = \sqrt{2 * V_I} * \text{Cos}(\omega * t) + V_0 + 2 * V_D$$

Kondensator C2 danner også en kapasitiv spenningsdeler, noe som øker kompleksiteten i beregningen. Vi har derfor simulert oppsettet av den kapasitive spenningsdelen i Multisim (se Kap 3.3.1), og har valgt kapasitansverdi og maksimal operativ spenning for kondensatoren ved hjelp av resultatene i simuleringene.

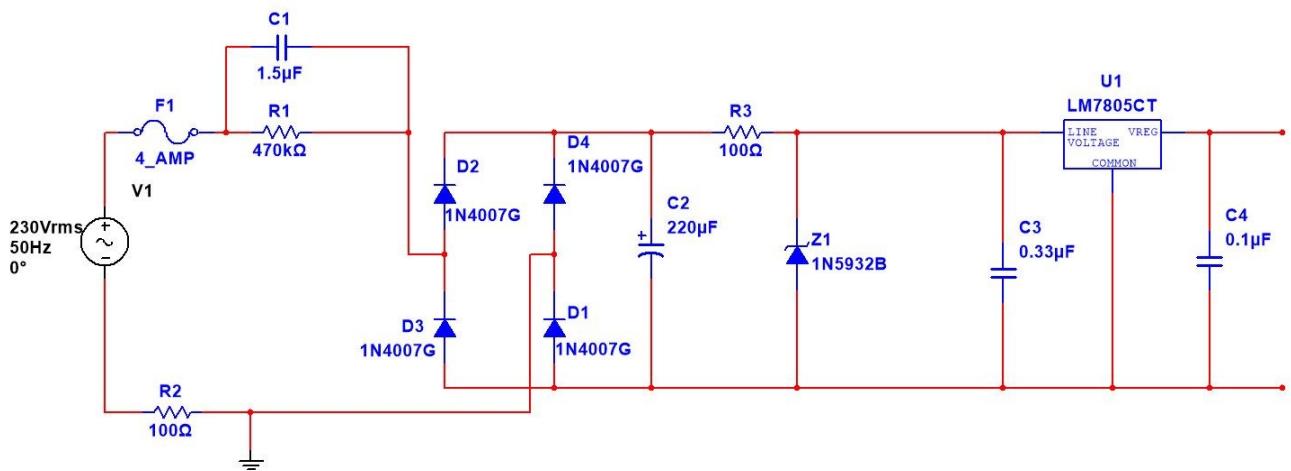
Zenerdioden skal gi en tilnærmet konstant spenningsverdi over inngangen til spenningsregulatoren, såfremt spenningen zenerdioden tilføres er over dens grenseverdi. Vi dimensjonerer derfor glattekondensator C2 med hensyn på å oppnå en spenningsdeling der den leverer en spenning over zenerdiodens grenseverdi, se Kap 2.3.2 for bakgrunn om kapasitiv spenningsdeling. Vi må også ta hensyn til kondensatorens fysiske mål. Vi ser det som mer kritisk å velge en kondensator som har en ønsket fysisk størrelse, og tolererer derfor at den ikke vil glatte spenningen inn til R3 og zenerdioden i like stor grad som en kondensator med større kapasitans ville gjort.

Vi benytter en kondensator med kapasitansverdi 220 uF, dimensjonert for å operere i spenningsområder opp til 50 volt.

3.3 SIMULERING OG TESTING

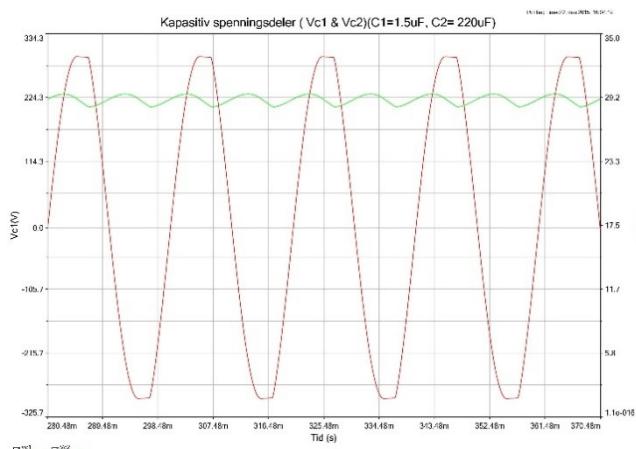
3.3.1 SIMULERING

Bruk av hardwaresimulering har vært en vesentlig del av den hardwarespesifikke delen av prosjektet, og har blant annet lagt grunnlaget for elementær forståelse av den kapasitive strømforsyningen, samt kunnskap om bruk av lysstyringssystemer. Den kapasitive strømforsyningen har vært vesentlig for gjennomføring av hele prosjektet, og prosjektgruppen har derfor benyttet programvaren Multisim 13.0 for å simulere en tiltenkt modell av strømforsyningen. Multisim muliggjør simulering av kretser både med ideelle og tilnærmet reelle modeller, hvor man benytter ideelle komponenter og tilegner dem ikke-ideelle egenskaper. Vi har i hovedsak simulert den kapasitive strømforsyningen som en ideell modell, da vi anser det som mest hensiktsmessig for den elementære forståelsen av den. Strømforsyningen har i hovedsak blitt simulert i denne modellen:

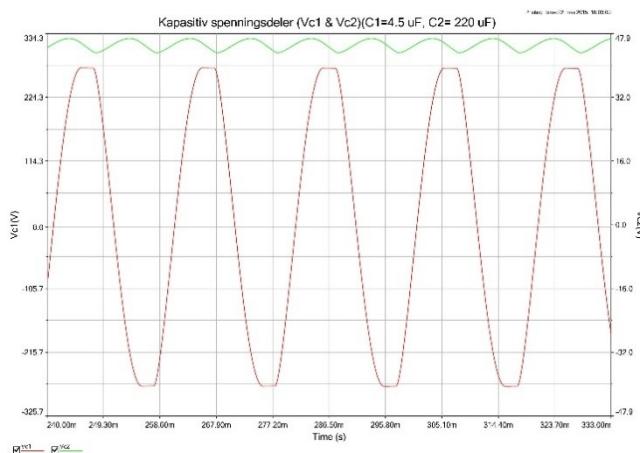


FIGUR 36. SIMULERINGSODELL AV KAPASITIV STRØMFORSYNING

Bruk av en simuleringsmodell i denne formen har gjort det mulig å se strømforsyningens elementære virkemåte, samt virkningen av endringer i komponentvalg i en forholdsvis enkel og ryddig form ved hjelp av tabeller og grafer. Simuleringsmodellen har blitt brukt til å hente ut data vi anser som sentrale for konstruksjonen av den reelle strømforsyningen, og vil ved flere tilfeller bli brukt som sammenligningsgrunnlag i rapporten.



FIGUR 37. SIMULERING, SPENNINGSDELING C1 OG C2, C1 = 1,5μF



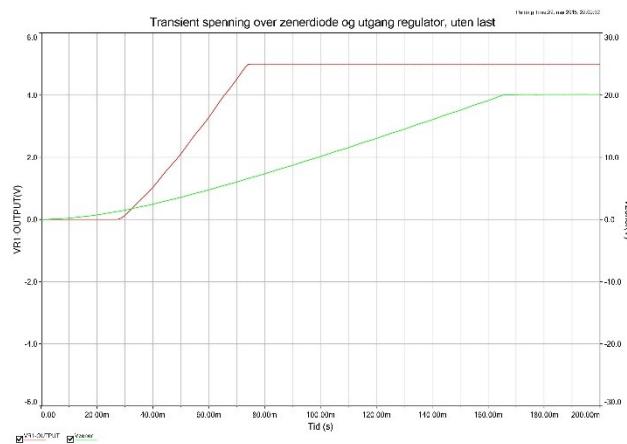
FIGUR 38. SIMULERING, SPENNINGSDELING C1 OG C2, C1 = 4,5 μ F

Figur 37 og Figur 38 viser at økt kapasitansverdi i kondensator C1 vil øke spenningen over kondensator C2.

Spenningen over C2 tilsvarer toppverdien spenningen over C1 skulle hatt for å opprettholde en ren sinus-kurve. Vi har dermed en operativ spenningsdeling mellom C1 og C2.

Transient og stasjonære spenningsforløp uten last: Transiente og stasjonære spenningsforløp uten last brukes i hovedsak som en referanse, og viser zenerdioden og spenningsregulatoren i deres enkleste operasjonsform.

Transient spenning over zenerdiode og utgang regulator uten last: Det transiente spenningsforløpet viser spenningsutviklingen over zenerdiode og spenningsregulator fra de blir påsatt spenning og fram til spenningen over dem er stasjonær.



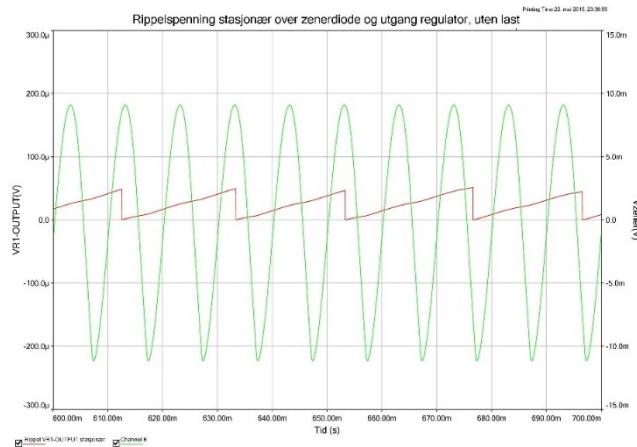
FIGUR 39. SIMULERING, TRANSIENTSPENNING UTE LAST

Vi ser her at spenningsregulatoren blir operativ når spenningen over zenerdioden har nådd ca 2 volt. Spenningen ut fra regulatoren øker sammen med spenningen over zenerdioden helt fram til zenerdioden krysser 7 volt ved ca 75 ms. Spenningen ut fra regulatoren holder seg tilnærmet konstant på 5 volt mens spenningen over zenerdioden øker til sin stasjonære verdi på 20 volt.

Stasjonær spenning over zenerdiode og utgang regulator uten last:

Det stasjonære spenningsplottet viser endringer i spenningen over henholdsvis zenerdiode og spenningsregulator ved

stasjonære forhold. En simulering av denne typen kan benyttes for å fastslå stabiliteten i spenningsleveransen ved stasjonære forhold. For vår del brukes denne simuleringen til å fastslå hvor stor andel av rippelspanningen regulatoren får tilført på inngangen som blir med videre til utgangen, med andre ord; regulatorens dempeevne.



FIGUR 40. SIMULERING, RIPPELSPENNING UTEN LAST

Vi ser at regulatoren får tilført en rippelspanning pålydende 16 mV topp-bunn fra zenerdioden, og at spenningen på utgangen av regulatoren på det meste varierer med 50 µV fra stasjonær verdi. Vi ser også at spenningsrippelen endrer form fra inngang til utgang på regulatoren; fra en ren sinusbølge til sagtannspanning. Ved å sammenligne topp-til-bunn-verdiene for inngangs- og utgangsspenning ser vi at regulatoren demper rippelen med en faktor på 320:

$$\frac{V_I}{V_O} = \frac{16 \text{ mV}}{50 \mu\text{V}} = 320$$

Innsvingningsforløp med last:

Innsvingningsforløpet ved full last skal simulere bruk av dongle, mikrokontroller og annen styrelogikk ved deres maksimale potensielle effektforbruk. Dette simuleres ved å sette en motstand som tilsvarer det totale effektforbruket med sikkerhetsmargin på utgangen av spenningsregulatoren.

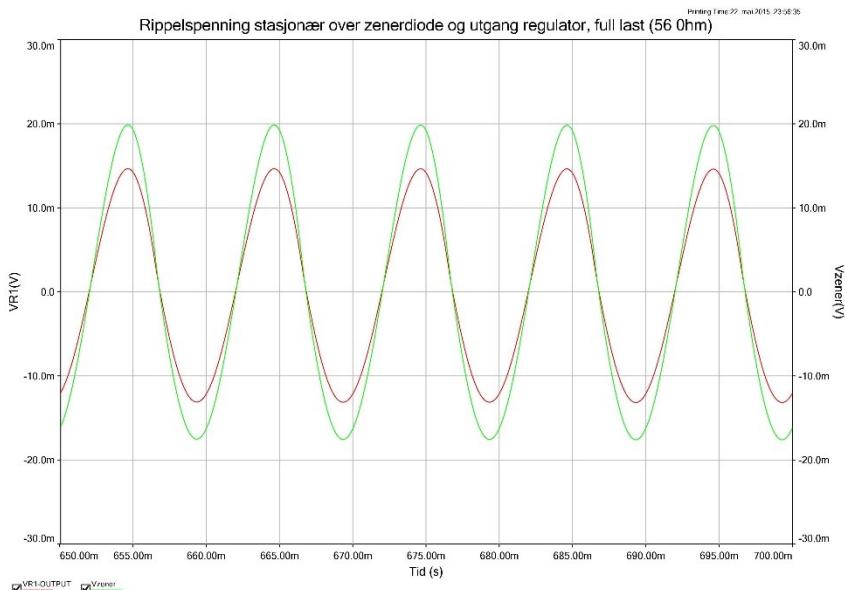
Full last på utgangen av spenningsregulatoren vil tilsvare en motstandsverdi pålydende:

$$R = \frac{U^2}{P} = \frac{5^2}{446 \text{ mW}} = 56 \Omega$$



FIGUR 41. SIMULERING, TRANSIENTSPENNING MED FULL LAST (56 OHM)

Vi ser at spenningsregulatoren blir operativ når zenerdioden har et spenningsnivå på 1.5 volt over seg, ved ca 30 ms. I motsetning til simuleringen uten last øker nå spenningen over både zenerdiode og utgangen av spenningsregulatoren med omtrent samme stigningstall. Spenningsregulatoren bruker vesentlig lengre tid på å bli stasjonær, omtrent 150 ms.



FIGUR 42. SIMULERING, RIPPELSPENNING MED FULL LAST (56 OHM)

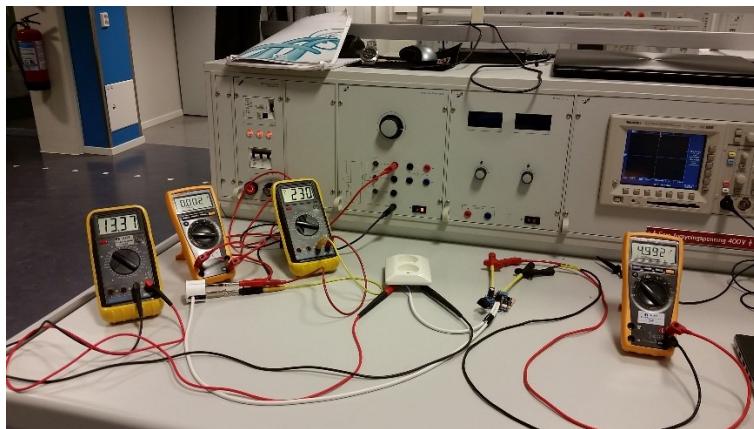
Ved stasjonære forhold får regulatoren tilført 35 mV rippelspanning fra zenerdioden, og at spenningen på utgangen av regulatoren på det meste varierer med 25 mV fra stasjonær verdi. Vi ser også at spenningsrippelen har samme form både på inngang og utgang ved full last, i motsetning til simuleringen uten last. Ved å sammenligne spenningsverdiene ser vi at regulatoren demper spenningsrippelen med en faktor på 1.4

$$\frac{V_I}{V_O} = \frac{35 \text{ mV}}{25 \mu\text{V}} = 1.4$$

Vi konkluderer med at regulatoren har vesentlig dårligere stasjonære egenskaper ved full last sammenlignet med uten last, og fremholder viktigheten av at regulatorens tilførselsspenning er i dens anbefalte område.

3.3.2 TESTING AV PROTOTYPER

Prosjektgruppen har i stor grad vært avhengig av testprosedyrer for å få kartlagt prototypenes egenskaper og virkemåte, samtidig som resultatet av testingen har lagt grunnlaget for endringer i nye versjoner av prototypene. Prosjektgruppen har benyttet den justerbare strømforsyningen Elabo 35-OL som spenningsforsyning under testing av prototypene. Den har også skilletransformator, noe som har vist seg å være viktig både for vår egen sikkerhet og utstyrets sikkerhet under testprosedyrene. Skilletransformatoren isolerer de lokale tilkoblingspunktene for spenning fra kraftleverandørens spenningstilførsel [48]. Dermed kan man trygt gjøre tester og målinger av prototypene med eksternt utstyr, uten å være bekymret for at det går store lekkasjestrømmer til nøytralleder eller jordingspunkt.

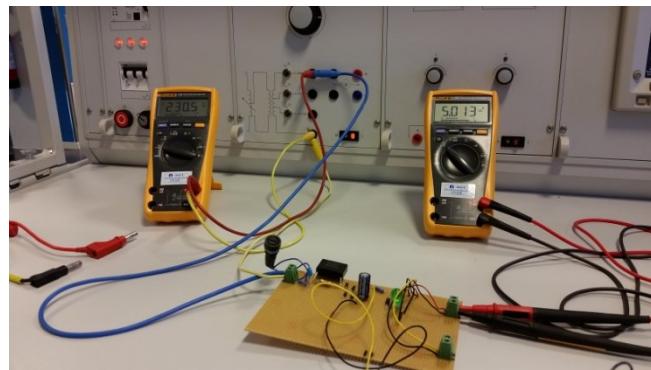


FIGUR 43. TESTING, TESTRIGG MED VARIABEL STRØMFORSYNING

Som en konsekvens av at vår kunnskap rundt prototypene har økt underveis i prosessen har også testprosedyrene utviklet seg til det bedre, og vi har kunnet arbeide mer målrettet for å finne den informasjonen vi mener har vært nødvendig for å verifisere forskjellige funksjoner i prototypene. Testprosedyrene har i stor grad omhandlet de samme elementene vi har simulert i Multisim, og har utviklet seg fra enkle strøm- og spenningsmålinger på de første prototypene til funksjonstester av operativ binær- og Dimmermodul, hvor ikke bare strøm- og spenningsmålinger men også I2C-kommunikasjon, nullgjennomgangsdeteksjon sendt til lokal hovedkontroller og triggersignal til sendt til henholdsvis relé og optokobler har vært sentrale elementer for testprosedyrene.

En stor del av test-arbeidet har også bestått av feilsøking, da ikke alle prototypene har fungert tilfredsstillende ved første forsøk. Feilsøkingsarbeidet har avdekket feil og mangler vi ikke hadde forutsett og/eller oppfattet under simulering og ved designprosessen, og har i så måte også bidratt til en positiv utvikling av prototypene.

En av de tidligste prototypene vi testet var en modell av den kapasitive strømforsyningen realisert på veroboard. Veroboard-prototypen har operert som vår faste referanse for testing av den kapasitive strømforsyningens egenskaper, og er i de fleste tilfeller den prototypen som det først har blitt utført endringer på ved utbedring av feil eller testing av nye konsept.



FIGUR 44. TESTING, KAPASITIV STRØMFORSYNING PÅ VEROBOARD

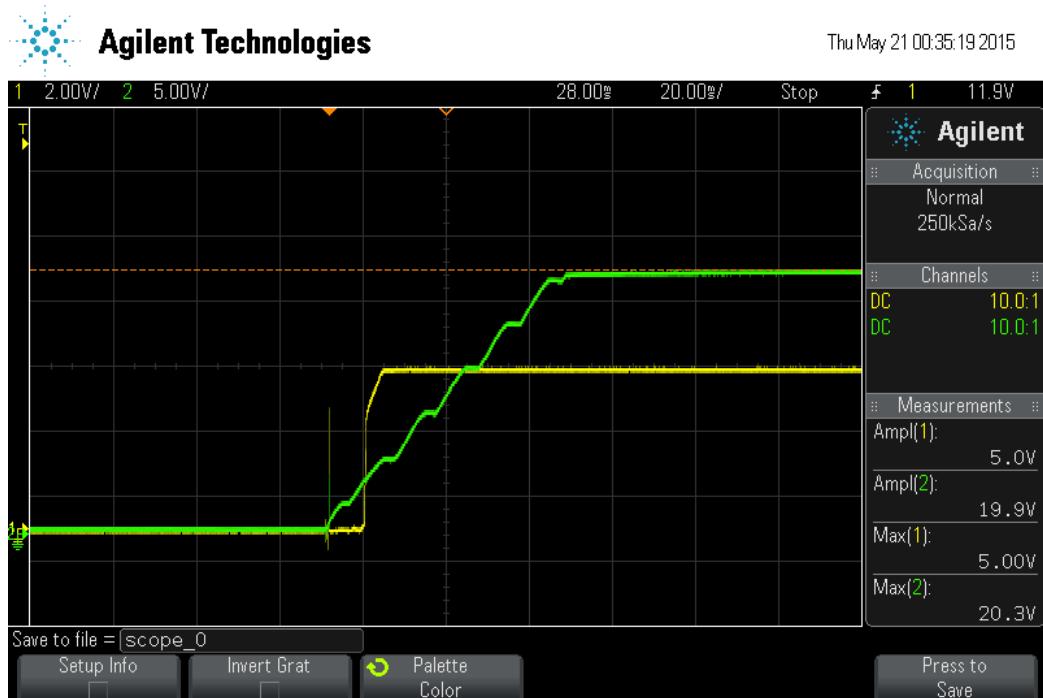
I og med at veroboard-prototypen er ment å operere mest mulig likt modellen i Multisim har vi kjørt den gjennom testmålinger i lik grad som Multisim-modellen for å sammenligne den teoretiske virkemåten i Multisim og den faktiske virkemåten.

For alle målingene i dette delkapittel gjelder:

Agilent MSO-X-2002A Oscilloscope, 10X måleprober

Kanal 1 (gul): spenning over utgang av spenningsregulator

Kanal 2 (grønn): spenning over zenerdiode

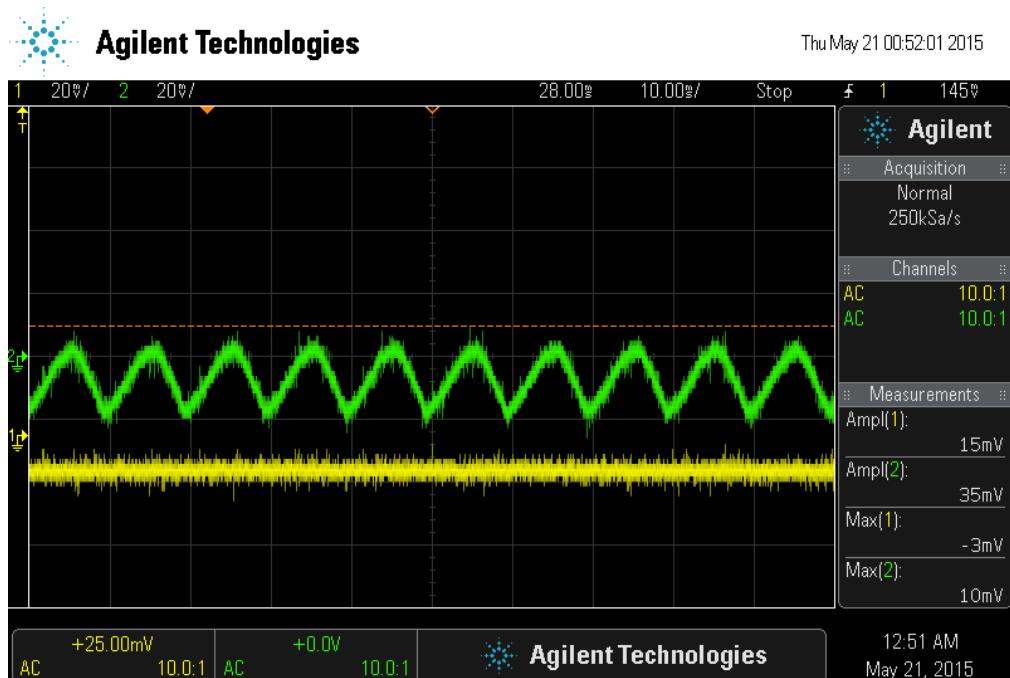


FIGUR 45. TESTING, TRANSIENTSPENNING OVER ZENERDIODE OG UTGANG REGULATOR, UTEK LAST

Av Figur 45 ser vi at det påtrykkes en spenning over zenerdioden, og at regulatoren ikke blir aktiv før spenningen over zenerdioden er tilnærmet 5 volt. Spenningen på utgangen av regulatoren går til omtrent 4 volt, før spenningen over zenerdioden øker ytterligere med det resultat at spenningen over utgangen av regulatoren øker til 5 volt. Regulatoren er stasjonær etter 10-15 ms, mens zenerdioden er stasjonær etter tilnærmet 55 ms.

Til sammenligning er regulatoren i Multisim-modellen stasjonær etter tilnærmet 75 ms, og zenerdioden etter 165 ms. Vi har ingen god teori på hvorfor det reelle innsvingningsforløpet er raskere enn det simulerte, men mistenker det kan ha en sammenheng med at Multisim bruker en generell modell for å simulere 7805-regulatorserien, og derfor ikke har eksakt de samme spesifikasjonene som vår regulator.

Stasjonær spenning over zenerdiode og utgang regulator uten last:

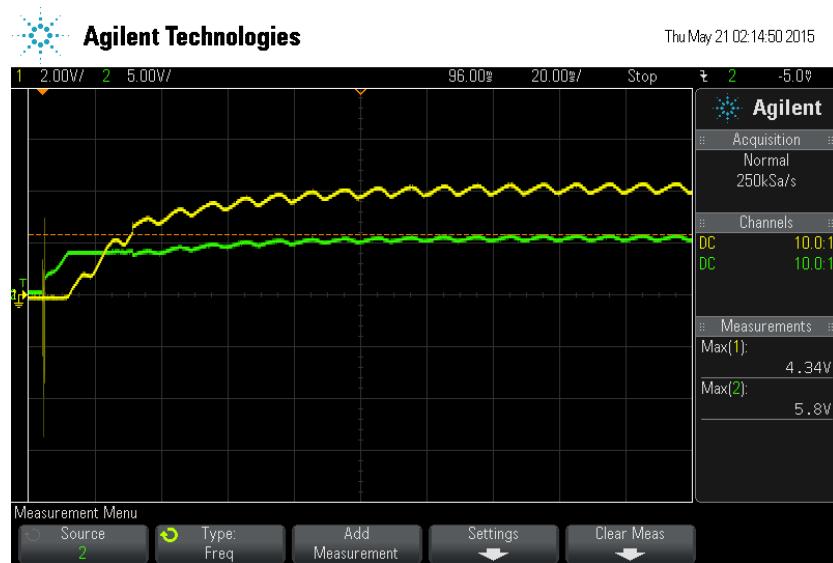


FIGUR 46. TESTING, STASJONÆR SPENNING OVER ZENERDIODE OG UTGANG, UTEN LAST

Vi ser av Figur 46 at den stasjonære spenningen over zenerdioden har en rippelverdi på 35 mV, mens spenningsrippelen på utgangen av regulatoren er 15 mV. Som det går fram av figuren måles det spesielt på kanal 2 en vesentlig mengde støy, og vi velger derfor å ikke benytte denne målingen for å beregne et spesifikt rippeldempningstall for regulatoren. Målingen gir derimot en god indikasjon på regulatorens rippeldempningsegenskaper, og vi kan ut fra målingen si at regulatoren gir en stabil utgangsspenning når vi måler etter rippel og for det meste detekterer støykomponenter.

Transient spenning over zenerdiode og utgang regulator ved full last (56 ohm):

Transientforløpet til regulatoren har blitt målt ved å plassere en 56 ohms motstand på utgangen av regulatoren.



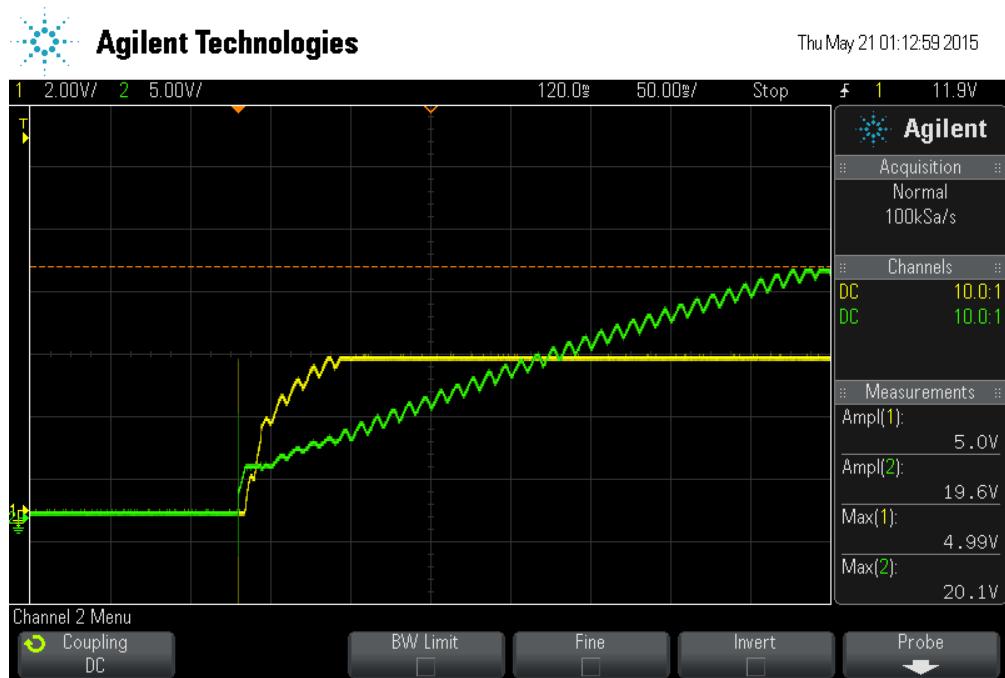
FIGUR 47. TESTING, TRANSIENTSPENNING OVER ZENERDIODE OG UTGANG REGULATOR, FULL LAST (56 OHM)

Figur 47 viser at spenningen over utgangen av regulatoren blir stasjonær på ca 4,3 volt, en verdi som ikke anses som tilfredsstillende. Vi ser også at den stasjonære spenningen inneholder en vesentlig mengde rippel, og konkluderer med at denne utgangsspenningen ikke vil være tilfredsstillende for å drive en last av denne størrelsen stasjonært. Ut fra effektberegningene vist i Kap 3.2.1 er det tydelig at det er filmkondensatoren som begrenser strøm gjennomgangen i stor grad. Samtidig vil en last på 56 ohm med stor sannsynlighet kun representere en andel av kretsens transiente forløp, og vi velger derfor å fortsette testprosedyren med en motstand som representerer et lavere effektforbruk.

Transient spenning over zenerdiode og utgang regulator ved påsatt last (82 ohm):

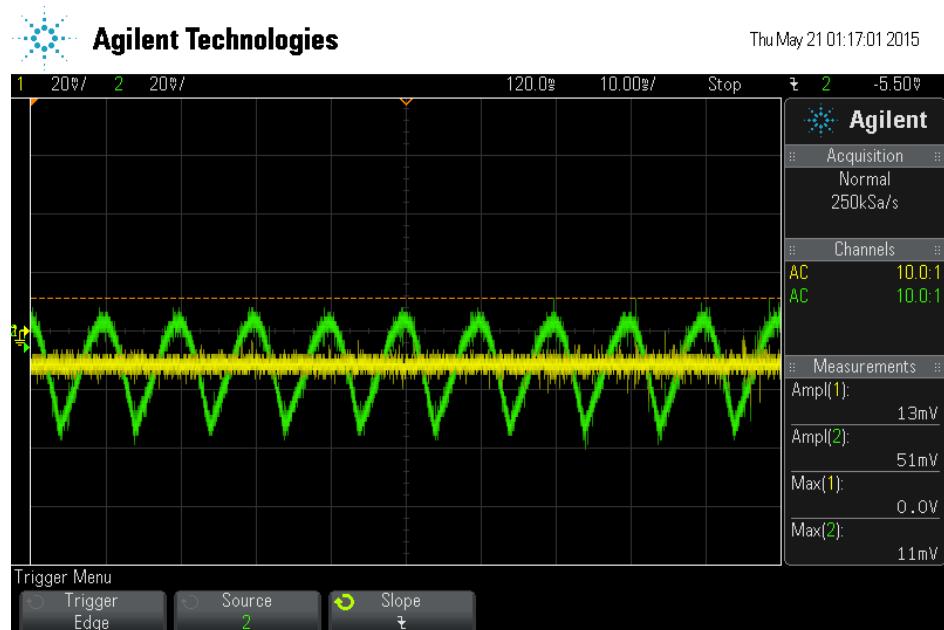
En last på 82 ohm representerer en effekt levert fra spenningsregulatoren pålydende:

$$P = \frac{U^2}{R} = \frac{5^2}{82} = 305 \text{ mW}$$



FIGUR 48. TESTING, TRANSIENTSPENNING OVER ZENERDIODE OG UTGANG REGULATOR. LAST = 82 OHM

Figur 48 viser at spenningen over utgangen av regulatoren øker til 5 volt stasjonært nivå samtidig som zenerdioden øker sitt spenningsnivå. Spenningsregulatoren er stasjonær etter omrent 60 ms, mens zenerdioden har nådd sin stasjonære verdi etter omrent 360 ms. Stasjonær spenning over zenerdiode og utgang regulator ved 82 ohm last:



FIGUR 49. TESTING, RIPPELSPENNING OVER ZENERDIODE OG UTGANG REGULATOR. LAST = 82 OHM

Den stasjonære spenningsverdien over zenerdiogen har en rippelverdi på omtrent 50 mV, mens rippelverdien på utgangen av spenningsregulatoren er tilnærmet 13 mV. Som det også går frem av den stasjonære målingen uten last i Figur 49 er også denne målingen representert med en større mengde støy i kanal 2, og vi mener ikke målingen representerer strømforsyningens reelle rippelundertrykkingsegenskaper. I og med at vi ser en markant endring i rippelspanningen på kanal 2 i forhold til testkjøring uten last, men ingen endring i rippelspanningen i kanal 1 konkluderer vi med at vi for det meste mäter støy i kanal 1.

3.3.3 ENDRINGER SOM FØLGE AV TESTING

Som nevnt i Kap 3.3.2 har flere av testprosedyrene ført til både store og små endringer i hardwarekonfigurasjonen. Noen av de mest vesentlige endringene i hardwarekonfigurasjonen er nevnt her:

Bytte av filmkondensator:

Under fullskalatesten for første prototyp av binærmodul ble det klart at den kapasitive strømforsyningen ikke var kapabel til å levere nok effekt. Dette viste seg ved at spenningen over zenerdiogen og utgangen av spenningsregulatoren ikke økte nok til å komme i operativt område, samtidig som nRF51-donglen ikke initialiserte. Vi ble under neste møte med oppdragsgiver gjort oppmerksomme på at donglens Segger J-link programmeringschip har en usedvanlig høy startstrøm, og at dette kunne være noe av årsaken til problemene vi opplevde. Vi valgte derfor å bytte til en filmkondensator som kombinerte to egenskaper; høyest mulig kapasitansverdi og minst mulig fysisk plass. Kondensatoren vi byttet til har en kapasitansverdi på 1.5 uF og var den eneste tilgjengelige modellen hos Farnell som oppfylte vårt krav om maksimal tillatt størrelse for å få plass i et nexa-chassis.

Bytte av relé:

Da det ble slått fast at den kapasitive strømforsyningen var underdimensjonert ble det utredet hvilke komponenter som kunne byttes ut og nedskaleres for å senke effektforbruket. Vi ble klar over at reléet som ble brukt for første prototype var en type med to sett kontakter, ett for normalt åpen og ett for normalt lukket. Denne konfigurasjonen medførte at reléet hadde et effektforbruk opp til 360 mW ved påslag, noe vi anså som uhensiktsmessig for vår bruk. Dermed ble reléet byttet til en modell bestående kun av ett normalt åpent kontaktsett og med et effektforbruk opp til 200 mW ved påslag.

Bytte av lokal hovedkontroller:

I startfasen av prosjektutførelsen forsøkte vi å bruke en ATtiny45V som lokal hovedkontroller. Etter innledende tester viste det seg derimot at denne kontrolleren ikke var like godt egnet som først antatt. Den hadde ikke god nok fleksibilitet, hadde ikke tilstrekkelige egenskaper og hadde dessuten ikke tilstrekkelig antall I/O-pinner. Vi valgte derfor å bytte til en litt mer avansert kontroller; Atmega48V. Denne kontrolleren er riktignok noe fysisk større, men gjør opp for det ved å inneholde mer funksjonalitet og rikelig med I/O-pinner.

Implementering av snubberkrets:

Under fullskalatest for første prototyp av Dimmermodul opplevde vi at lyset i lampelosten flimret ved flere anledninger. Flimringen er i de fleste tilfeller forårsaket av spenningstransienter på tilførselsspenningen, som skaper ujevnhet i bølgeformen tilført triacen [49]. For å løse denne problemstillingen ble neste versjon av prototypen dimensjonert med en snubberkrets, med spesifikasjoner fra databladet for triac-optokobleren. En snubberkrets består av en kondensator og en mostand, og opererer som et RC-filter. Filteret fjerner majoriteten av spenningstransientene inn til triacen og sørger dermed for at lasten som står tilkoblet triacen får en filtrert spenningstilførsel.

3.4 ENDELIG DESIGN

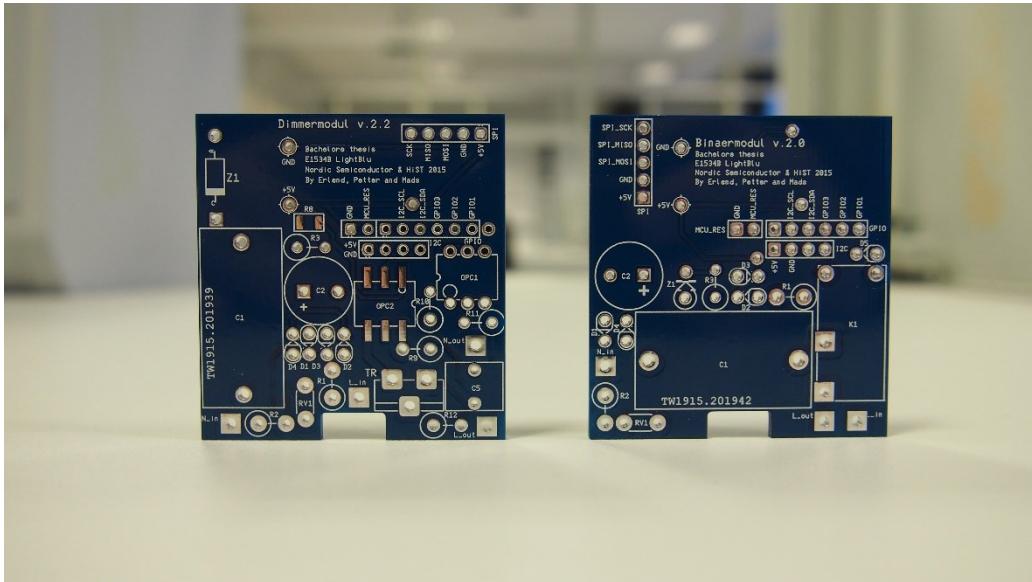
Binærmodul v.2.0:

Binærmodulen består av en kapasitiv strømforsyning, en lokal hovedkontroller, et relé og diverse styre- og kommunikasjonskretser. Modulen måler $45,1 * 48,1 \text{ mm}$ (B, L)

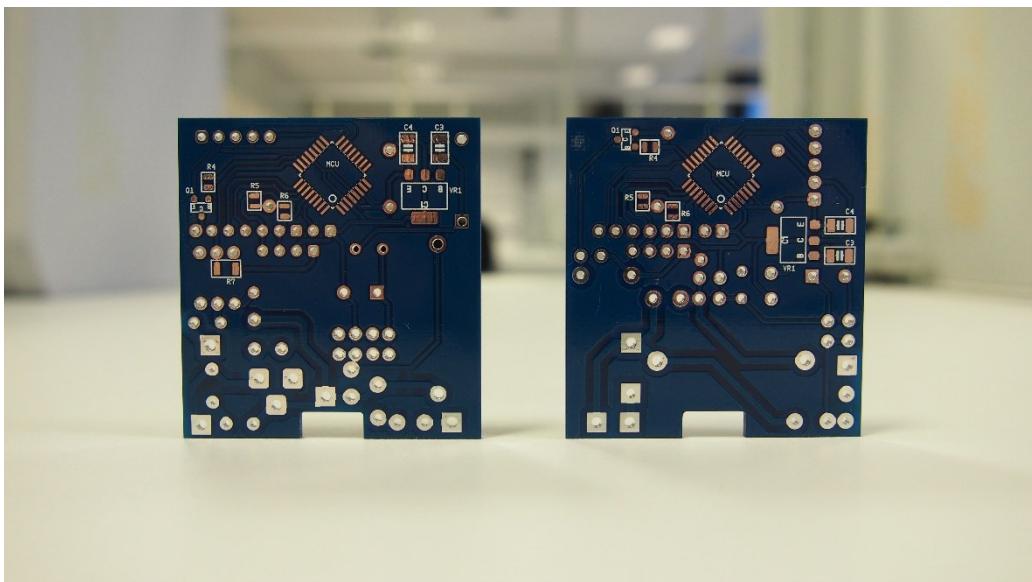
Dimmermodul v.2.2:

Dimmermodulen består av en kapasitiv strømforsyning, en lokal hovedkontroller, en TRIAC, to optokoblere og diverse styre- og kommunikasjonskretser. Modulen måler $45,1 * 48,1 \text{ mm}$ (B, L)

Se Vedlegg 2 for skjema og utlegg for Binær- og Dimmermodul.



FIGUR 50. ENDELIG DESIGN, BINÆR- OG DIMMERMODUL (TOP)



FIGUR 51. ENDELIG DESIGN, BINÆR OG DIMMERMODUL (BOTTOM)

KAP.4 FIRMWARE

4.1 LØSNING

Både Binær- og Dimmermodulen bruker ATmega48V som lokal hovedkontroller. Denne har som kjent i oppgave å styre selve lyskretsene basert på kommandoer som mottas over TWI-bussen.

Vi har utviklet følgende firmwareversjoner:

Firmware-versjon	SoftDevice	Modul	MCU	Kommentar
LDF1A				
-> _v.0.3	S110	Bluetooth kontrollmodul (nRF51 DK)	nRF51422	Kun brukt til testing
-> _v.0.3_Dongle	S110	Bluetooth kontrollmodul (nRF51 Dongle)	nRF51422	
LDFBIN1				
-> _MASTER	-	Switchmodul	ATmega48V	Kun brukt til testing.
-> _SLAVE_v.1.1	-	Binærmodul	ATmega48V	
LDFDIM1				
-> _SLAVE_v.1.3	-	Dimmermodul	ATmega48V	

TABELL 6. FIRMWARE, VERSJONSOVERSIKT

Som det fremgår av tabellen, så har vi tre forskjellige firmwareversjoner som er implementert i det ferdige produktet. En versjon hver for binærmodul, Dimmermodul og kontrollmodul. To versjoner til ble utviklet for switch og dev-kit, men ble kun anvendt til testing.

Poenget med firmware er at de ulike modulene skal kunne håndtere kommandoer, i form av tallverdier, de mottar. Under vises en liten tabell over betydningen av kommandoene som modulene er programmet til å kunne tolke:

Kommando:	Betydning:
0x05	Lys på
0x07	Lys av
0x0A	Lysnivå 0 %
0x0C	Lysnivå 20 %
0x0E	Lysnivå 40 %
0x10	Lysnivå 60 %
0x12	Lysnivå 80 %
0x14	Lysnivå 100 %

TABELL 7. FIRMWARE, KOMMANDOOVERSIKT

Tallene for kommandoene er helt vilkårlige verdier som vi har valgt selv. 0x05 og 0x07 gjelder for binærmodulen, mens de resterende gjelder for Dimmermodulen. Slik at når en modul mottar en av disse kommandoene iverksettes de nødvendige tiltakene for at lyset skal stilles inn til ønsket nivå.

4.2 FIRMWARE FOR LOKAL HOVEDKONTROLLER

4.2.1 KODE FOR TWI

Både Binær- og Dimmermodulen skal bruke I2C-kompatibel TWI som primær kommunikasjonsmåte for sending og mottak av hhv. statusmeldinger og styringskommandoer. Ettersom begge modulene benytter samme type lokal hovedkontroller, ATmega48V, har vi vurdert det som mest praktisk å utvikle én felles TWI-kode som kan implementeres i firmwareversjonene til begge modulene. TWI-koden vi benytter er en modifisert og videreutviklet versjon av en kode hentet fra engineersgarage.com [42] som opererer i henhold til Atmels TWI-spesifikasjon definert av ATmega48Vs datablad. [43]

Koden er skrevet i C og er lagret i form av en include-fil (.h).

Følgende kodesnipper er ment for å illustrere de viktigste funksjonalitetene til TWI-koden, slik at kodeforskninga i de påfølgende underkapitlene gir mer mening. Hele koden finnes i mappestrukturen til hver av firmwareversjonene LDFBIN1 og LDFDIM1 i det digitale arkivet (Vedlegg 4).

Kodesnipp som viser funksjonen «TWI_init_slave»:

Funksjon:	Setter opp hovedkontrolleren som slave.
Nødvendige variabler:	Slavens TWI-adresse.
Returnerer:	Ingenting.

```
void TWI_init_slave(uint8_t slave_addr)
{
    TWAR = (slave_addr<<1);           // Set own TWI slave address. Accept TWI general calls.
}
```

Kodesnipp som viser funksjonen «TWI_slave_reciever_transmitter»:

Funksjon:	Hovedkontrolleren settes i en sender/mottaker-modus som kontinuerlig overvåker TWI-bussen. Oppdages enten SLA+W eller SLA+R hopper programmet ut av while-løkka og går videre i utføringen.
Nødvendige variabler:	Ingen.
Returnerer:	Ingenting.

```
void TWI_slave_reciever_transmitter(void)
{
    while ( ((TWSR & 0xF8) != 0xA8) && ((TWSR & 0xF8) != 0x60) )
        // Loop until correct acknowledgment has been received
    {
        TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA); // Enable TWI, clear TWI interrupt flag
        while (!(TWCR &(1<<TWINT)));
            // Wait for TWINT flag
    }
}
```

Kodesnipp som viser funksjonen «TWI_slave_read_data»:

Funksjon:	Hovedkontrolleren leser én 8-bits datapakke fra kommuniserende masterenhet og svarer med ACK.
Nødvendige variabler:	Ingen.
Returnerer:	Én 8-bits datapakke.

```
uint8_t TWI_slave_read_data(void)
{
    uint8_t data = 0;

    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
    // Clear TWINT and set TWEN and TWEA to start receiving data

    while (!(TWCR &(1<<TWINT)));    // Wait for TWINT flag set
    while ((TWSR & 0xF8) != 0x80);   // Wait for ACK

    data = TWDR;      // Load First byte of data from data register into the variable 'data'
    return data;
}
```

Kodesnipp som viser funksjonen «TWI_slave_write_data»:

Funksjon:	Hovedkontrolleren skriver én 8-bits datapakke til kommuniserende masterenhet som svarer med NACK.
Nødvendige variabler:	Én 8-bits datapakke.
Returnerer:	Ingenting.

```
void TWI_slave_write_data(uint8_t data)
{
    TWDR = data;           // Copy Data into data register
    TWCR = (1<<TWINT)|(1<<TWEN); // Clear TWINT and set TWEN to start transmission

    while ((TWSR & 0xF8) != 0xC0); // Wait for ACK
}
```

4.2.2 FIRMWARE FOR BINÆRMODUL (LDFBIN1)

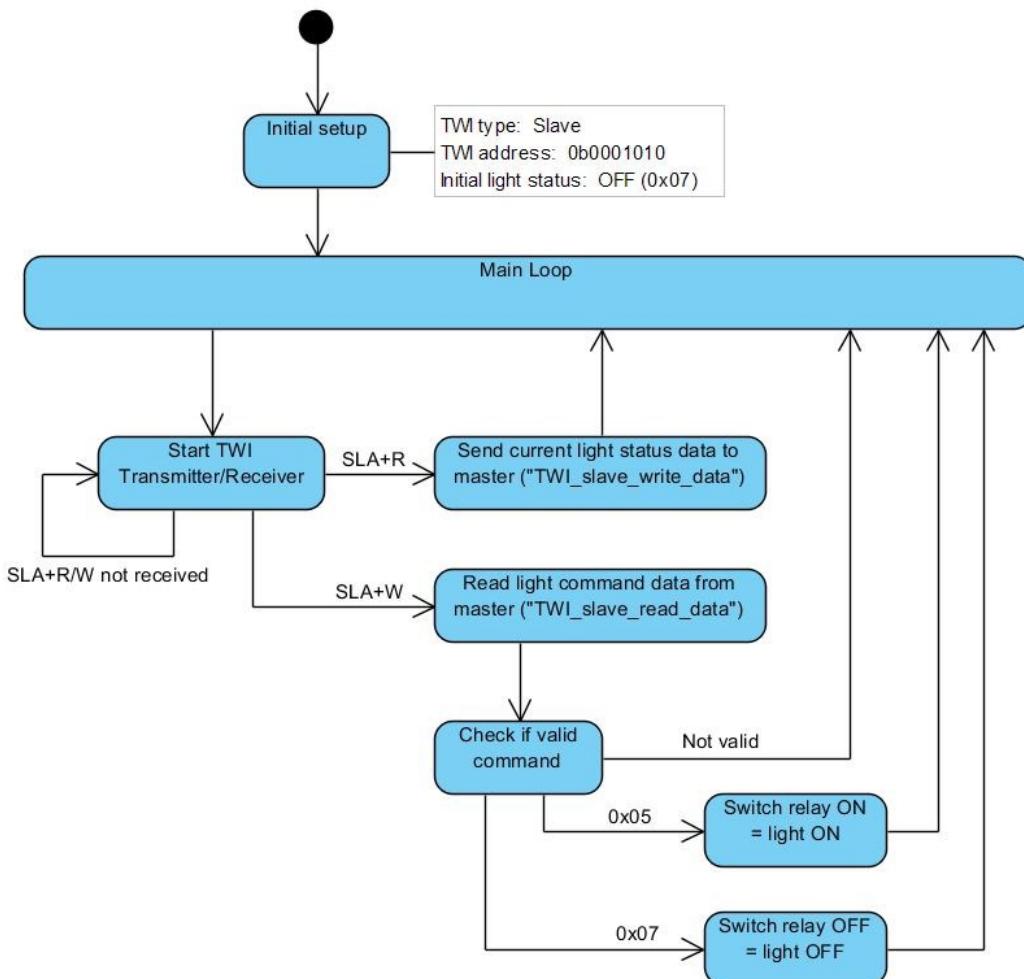
Binærmodulen har fått navnet sitt fordi den styrer lyset binært, det vil si enten er lyset av eller så er det på. I Kap 3.1 beskrives hardwareløsningen vi har valgt for denne modulen. Kort oppsummert er den relativt enkel og ukomplisert. Binærmodulens firmware er tilsvarende ukomplisert og vi har derfor ikke lagt spesielt mye sjel eller finesse i denne delen av kodeutviklingen.

Vi har utviklet to forskjellige versjoner av LDFBIN1; LDFBIN1_MASTER for Switchmodulen (kun brukt til testing) og LDFBIN1_SLAVE for selve Binærmodulen. Sistnevnte er helt klart mest relevant og vi velger derfor å fokusere på denne.

LDFBIN1_SLAVE skal ha følgende funksjonalitet:

- Den skal kunne sende lysstatusmeldinger over TWI-bussen.
- Den skal kunne motta lysstyringskommandoer over TWI-bussen.
- Den skal kunne vurdere om mottatt lysstyringskommando er gyldig.
- Den skal kunne ignorere gyldige kommandoer som ikke fører til endring i lysstatus.
- Den skal kunne generere et kontrollsignal som styrer reléet.

For å bedre forstå denne funksjonsbeskrivelsen har vi laget følgende tilstandsdiagram:



FIGUR 52. FIRMWARE, TILSTANDSDIAGRAM FOR LDFBIN1_SLAVE

LDFBIN1_SLAVE er skrevet i C ved hjelp av Atmel Studio 6.2 og finnes i det digitale arkivet (Vedlegg 4) sammen med LDFBIN1_MASTER. Figur 53 gjengir hovedkoden til LDFBIN1_SLAVE.

Merk at funksjonskall som refererer til funksjoner beskrevet i Kap 4.2.1 er skrevet med rød/oransje tekst.

```
#define F_CPU 1000000UL
#define TWI_ADDR_MODUL 0b0001010 // TWI address for BINAERMODUL

#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "config.h"
#include "TWI.h"

int main(void)
{
    uint8_t data_from_master = 0;
    uint8_t status_data_slave = 0x07;

    DDRC |= (1<<DDC0); // Set PC0 as output for relay control signal
    DDRC |= (1<<DDC1); // Set PC1 as output for test signal on GPIO1
    PORTC &= ~(1<<PC0); // Ensure RELAY is in off-state to begin with
    PORTC &= ~(1<<PC1);

    sei();

    TWI_init_slave(TWI_ADDR_MODUL); // Initialize TWI (I2C) communication as SLAVE

    while(1)
    {
        TWI_slave_reciever_transmitter(); // Start dual-mode transmitter + receiver

        if ((TWSR & 0xF8) == 0x60) // If SLA+W was received from master
        {
            data_from_master = TWI_slave_read_data(); // Get data from MASTER

            if (data_from_master == 0x05)
            {
                PORTC |= (1<<PC0); // Switch on RELAY
                PORTC |= (1<<PC1); // Set test signal on GPIO1
                status_data_slave = 0x05;
            }
            else if (data_from_master == 0x07)
            {
                PORTC &= ~(1<<PC0); // Switch off RELAY
                PORTC &= ~(1<<PC1); // Clear test signal on GPIO1
                status_data_slave = 0x07;
            }

            data_from_master = 0;
        }

        else if ((TWSR & 0xF8) == 0xA8) // If SLA+R was received from master
        {
            TWI_slave_write_data(status_data_slave);
        }
    }
}
```

FIGUR 53. FIRMWARE, HOVEDKODE LDFBIN1_SLAVE_v.1.1.c

4.2.3 FIRMWARE FOR DIMMERMODUL (LDFDIM1)

I motsetning til Binærmodulen er Dimmermodulen vesentlig mer kompleks. Se Kap 2.2 for teori om styring og dimming av lys og Kap 3.1 for beskrivelse av valgt hardwareløsning. Firmwareversjon LDFDIM1 må nødvendigvis være tilsvarende kompleks, men heldigvis deler Dimmermodulen mye av funksjonaliteten til Binærmodulen takket være designvalgene våre. LDFDIM1 kan derfor sies å være en utvidet form av LDFBIN1.

Vi har kun utviklet én utgave av LDFDIM1; LDFDIM1_SLAVE_v.1.3. Denne versjonen skal ha følgende funksjonalitet:

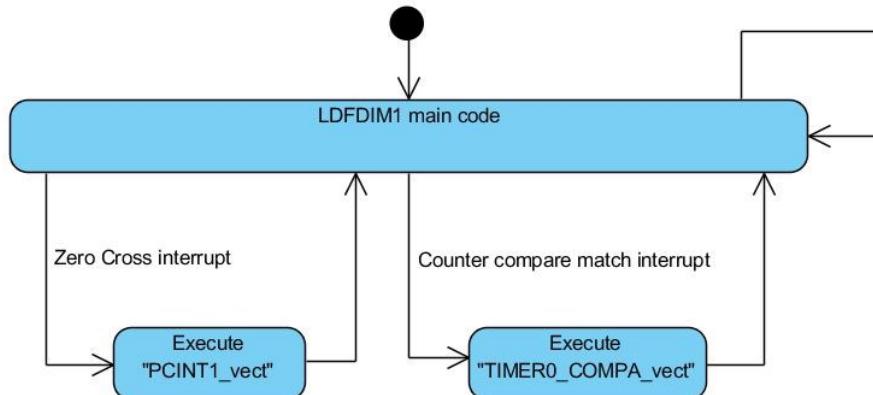
- Den skal kunne sende lysstatusmeldinger over TWI-bussen.
- Den skal kunne motta lysstyringskommandoer over TWI-bussen.
- Den skal kunne vurdere om mottatt lysstyringskommando er gyldig.
- Den skal kunne ignorere gyldige kommandoer som ikke fører til endring i lysstatus.
- Den skal kunne operere en separat teller (Counter) som kjører parallelt med hovedkoden.
- Den skal kunne sette opp og håndtere avbruddsrutiner, eller Interrupt Service Routine (ISR), som genereres ved nullkryssing (zero-cross) av linjespenning.
- Den skal kunne sette opp og håndtere avbruddsrutiner, eller Interrupt Service Routine (ISR), som genereres av tellere ved oppnådd tellerverdi (compare match).
- Den skal kunne generere en periodisk kontrollpuls som styrer trigg av TRIAC.

Den største forskjellen med LDFDIM1 kontra LDFBIN1 er tilleggsfunksjonaliteten for teller og avbruddsrutiner. Telleren er hardwarebasert, det vil si at tellingen foregår i en separat hardwaremodul i ATmega48V, og det er kun initialiseringen av denne som gjøres i koden. Avbruddsovervåkning er også hardwarebasert og hindrer dermed ikke utføringen av hovedkoden annet enn når selve avbruddsrutinen trigges. Når LDFDIM1 er i normal operasjon foregår det derfor flere ting samtidig.

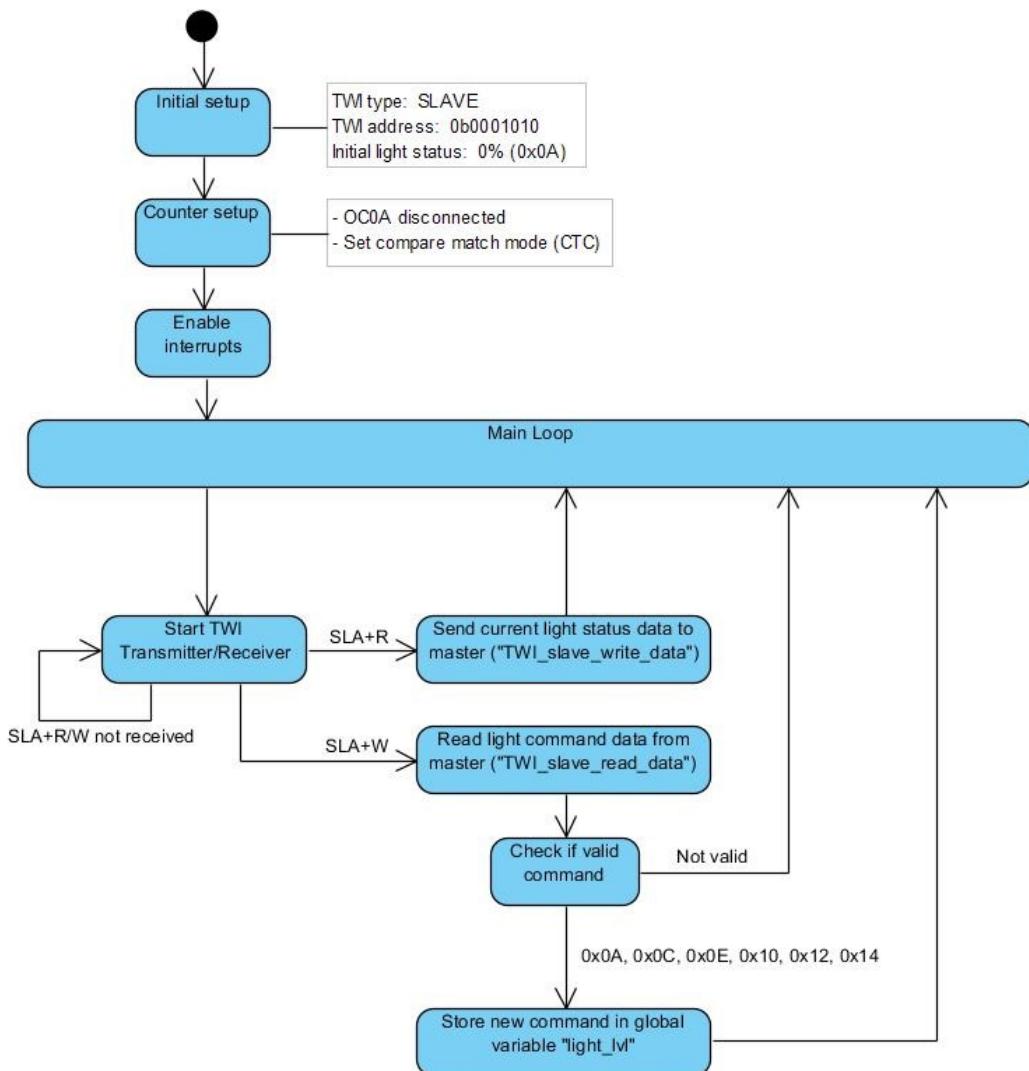
Det hele starter med hovedkoden. Figur 55 illustrerer hovedkoden i form av et tilstandsdiagram. Etter innledende konfigurasjon av TWI, inputs, outputs osv. aktiveres avbruddsovervåkningen og hovedsløyfa (main loop) starter. Avbruddsovervåkningen kjøres parallelt med hovedkoden og er satt opp til å agere ved følgende anledninger:

- Signalet fra deteksjonskretsen for nullkryssing går fra lavt til høyt (rising edge).
- Telleren er ferdig med en tellesyklus og compare match er oppnådd.

Hver gang en av disse hendelsene forekommer trigges et avbrudd. Dette fører til at hovedkoden stanses og den aktuelle avbruddsrutinen (ISR) startes; PCINT1_vect og TIMERO0_COMPA_vect for hhv. nullkryssing og teller. Når avbruddsrutinen er gjennomført fortsetter hovedkoden der den slapp. Figur 54 illustrerer dette forløpet.



FIGUR 54. FIRMWARE, SAMMENHENG MELLOM HOVEDKODE OG AVBRUDDSRUTINER FOR LDFDIM1



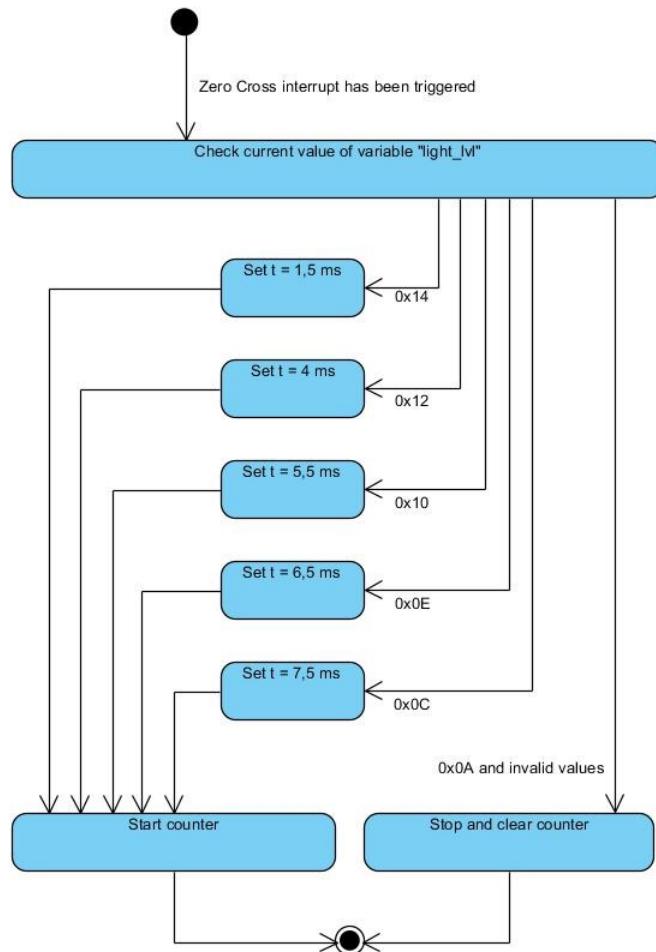
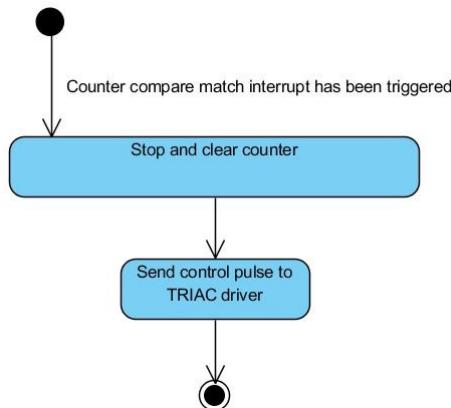
FIGUR 55. FIRMWARE, TILSTANDSDIAGRAM FOR HOVEDKODEN I LDFDIM1

Avbruddsrutinen for nullkryssing PCINT1_vect har kun to oppgaver. Når et nullkryssingsavbrudd har blitt trigget skal PCINT1_vect først sjekke hvilken lyskommando som er gjeldende. Deretter konfigureres telleren med en spesifikk tellerverdi som er assosiert med det aktuelle lysnivået og telleren startes. Dersom gjeldende kommando har verdien 0x0A, eller er ugyldig, skal telleren stanses og tellerverdiregisteret nullstilles. Dette illustreres i Figur 56.

Avbruddsrutinen som triggas når en kjørende teller oppnår compare match har også kun to oppgaver. Når avbruddet triggas skal TIMER0_COMPA_vect stanse telleren og nullstille tellerverdiregisteret og tellerinstillingene. Deretter genereres kontrollpulsen som styrer triggning av TRIAC. Se Figur 57.

LDFDIM1 er skrevet i C ved hjelp av Atmel Studio 6.2 og finnes i det digitale arkivet (Vedlegg 4).

Figur 59 og Figur 60 gjengir hhv. PCINT1_vect og TIMER0_COMPA_vect i kodeform.

**FIGUR 56. FIRMWARE, TILSTANDSDIAGRAM FOR AVBRUDDSRUTINE PCINT1_VECT****FIGUR 57. FIRMWARE, TILSTANDSDIAGRAM FOR AVBRUDDSRUTINE TIMER_COMPA_VECT**

```

#define F_CPU 1000000UL
#define TWI_ADDR_DIM_MODUL 0b0001010 // TWI address for DIMMERMODUL

#include <stdio.h>
#include <stdlib.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#include "config.h"
#include "TWI.h"

uint8_t light_lvl;

int main(void){
    uint8_t data_from_master = 0;
    uint8_t status_data_slave = 0xA; // Start value for light output, 0x0A = 0% Light.
    light_lvl = 0;

    DDRC |= (1<<DDC0); // Set PC0 as output
    DDRC |= (1<<DDC2); // Set PC2 as output
    DDRC &= ~(1<<DDC1); // Set PC1 as input, circuit already contains pull-up resistor

    PORTC &= ~(1<<PC0); // Ensure TRIAC is in off-state to begin with
    PORTC &= ~(1<<PC2); // For test

    // Initiate interrupts
    PCICR |= (1<<PCIE1); // Enable interrupt requests on pins PCINT14..8
    PCMSK1 |= (1<<PCINT9); // Enable interrupt request on PCINT9 (PC1)

    // Initiate timer
    TCCR0A &= ~((1<<COM0A1)|(1<<COM0A0)); // Set normal port operation, OC0A disconnected
    TCCR0A |= (1<<WGM01); // Enable CTC-mode
    TIMSK0 |= (1<<OCIE0A); // Enable output compare match interrupt

    sei();
    TWI_init_slave(TWI_ADDR_DIM_MODUL); // Initialize TWI (I2C) communication as SLAVE

    while(1){
        TWI_slave_reciever_transmitter(); // Start dual-mode transmitter + receiver

        if ((TWSR & 0xF8) == 0x60){ // If SLA+W was received from master
            data_from_master = TWI_slave_read_data(); // Get data from MASTER

            if (data_from_master == 0xA){
                light_lvl = data_from_master;
                status_data_slave = 0xA;
            }
            else if (data_from_master == 0xC){
                light_lvl = data_from_master;
                status_data_slave = 0xC;
            }
            else if (data_from_master == 0xE){
                light_lvl = data_from_master;
                status_data_slave = 0xE;
            }
            else if (data_from_master == 0x10){
                light_lvl = data_from_master;
                status_data_slave = 0x10;
            }
            else if (data_from_master == 0x12){
                light_lvl = data_from_master;
                status_data_slave = 0x12;
            }
            else if (data_from_master == 0x14){
                light_lvl = data_from_master;
                status_data_slave = 0x14;
            }
            data_from_master = 0;
        }
        else if ((TWSR & 0xF8) == 0xA8){ // If SLA+R was received from master
            TWI_slave_write_data(status_data_slave);
        }
    }
}

```

FIGUR 58. FIRMWARE, MAIN LOOP FOR LDFDIM1_SLAVE (UTDRAG FRA LDFDIM1_SLAVE_v1.3.c)

```

ISR(PCINT1_vect)
{
    if ((PINC & (1<<PC1))){


        // Timer frequencies calculated by using the formula:
        // f = F_CPU / ( N * (1 + OCR0A) )
        // t = 1 / f
        // Hence the value of OCR0A can be found for desired values of f:
        // OCR0A = (F_CPU / ( N * f )) - 1

        if (light_lvl == 0x14){
            OCR0A      = 186;           // Set compare value so that f = 666Hz -> t = 1,5ms
            TCCR0B     |= (1<<CS01);   // Start timer with prescaler N = 8
        }
        else if (light_lvl == 0x12){
            OCR0A      = 61;           // Set compare value so that f = 250Hz -> t = 4ms
            TCCR0B     |= ((1<<CS01)|(1<<CS00)); // start timer with prescaler N = 64
        }
        else if (light_lvl == 0x10){
            OCR0A      = 85;           // Set compare value so that f = 180Hz -> t = 5,5ms
            TCCR0B     |= ((1<<CS01)|(1<<CS00)); // start timer with prescaler N = 64
        }
        else if (light_lvl == 0x0E){
            OCR0A      = 101;          // Set compare value so that f = 153Hz -> t = 6,5ms
            TCCR0B     |= ((1<<CS01)|(1<<CS00)); // start timer with prescaler N = 64
        }
        else if (light_lvl == 0x0C){
            OCR0A      = 117;          // Set compare value so that f = 133Hz -> t = 7,5ms
            TCCR0B     |= ((1<<CS01)|(1<<CS00)); // start timer with prescaler N = 64
        }
        else{
            TCCR0B     = 0;            // Stop timer
            OCR0A      = 0;            // Clear compare value
            TCNT0     = 0;            // Clear counter value
            TIFR0     |= (1<<OCF0A); // Clear interrupt flag
        }
    }
}

```

FIGUR 59. FIRMWARE, AVBRUDDSRUTINE PCINT1_VECT (UTDRAG FRA LDFDIM1_SLAVE_v1.3.c)

```

ISR(TIMER0_COMPA_vect)
{
    TCCR0B     = 0;           // Stop timer
    OCR0A      = 0;           // Clear compare value
    TCNT0     = 0;           // Clear counter value
    TIFR0     |= (1<<OCF0A); // Clear interrupt flag

    PORTC |= (1<<PC0);       // Set output to triacdriver high
    PORTC |= (1<<PC2);       // For test

    _delay_us(200);

    PORTC &= ~(1<<PC0);     // Set output to triacdriver low
    PORTC &= ~(1<<PC2);     // for test
}

```

FIGUR 60. FIRMWARE, AVBRUDDSRUTINE TIMER0_COMPA_VECT (UTDRAG FRA LDFDIM1_SLAVE_v1.3.c)

4.3 BLUETOOTH SMART FIRMWARE

4.3.1 OVERORDNET FUNKSJONSFORKLARING

Uansett om kontrollmodulen er koblet til en Binær- eller Dimmermodul, skal den kun fungere som et bindeledd mellom den tilkoblede modulen og enheter på Bluetooth Smart-nettverket, og bryr seg derfor ikke om hvilke data som passerer gjennom den. Oppgaven dens er å påse at data som sendes over Bluetooth Smart finner vegen til Binær- eller Dimmermodulen gjennom TWI-bussen. Firmwareversjon LDF1A for kontrollmodulen komplementerer dermed de andre firmwareversjonene ved å opptre som TWI-master.

I advertising mode opererer LDF1A som broadcaster og skal ha følgende funksjonalitet:

- Den skal kunne sende advertising-pakker hvert 60. millisekund.
- Den skal kunne sende scan response-pakker bestående av «Light Control» service UUID.
- Den skal kunne lese nåværende lysstatus hvert sekund fra tilkoblet modul over TWI-bussen.
- Den skal kunne oppdatere manufacturer data med lysstatusverdien etter hver avlesning.

I connected mode opererer LDF1A som peripheral og skal ha følgende funksjonalitet:

- Den skal kunne motta lysstyringskommandoer fra central (mobilapplikasjon).
- Den skal kunne videreforside mottatt lysstyringskommando til Binær- eller Dimmermodul over TWI-bussen.
- Den skal kunne lese nåværende lysstatus fra tilkoblet modul over TWI-bussen.
- Den skal kunne håndtere leseforespørslar fra central (mobilapplikasjon) og svare med nåværende lysstatus ved gyldig forespørsel.

Vi har valgt å implementere følgende datastruktur for GATT-serveren:

- En Service kalt «Light Control» med UUID: 2f160001-e5dc-11e4-9cf7-0002a5d5c51b
- En write-only Characteristic kalt «Light Command» med UUID: 2f160002-e5dc-11e4-9cf7-0002a5d5c51b.
- En read-only Characteristic kalt «Light Status» med UUID: 2f160003-e5dc-11e4-9cf7-0002a5d5c51b.

LDF1A bygger på eksemplkoden «ble_app_template_s110_pca10028» som inkluderes i Nordic Semiconductors nRF51_SDK_v.8.0.0, og er laget for å samarbeide med Nordic Semiconductors S110 SoftDevice.

Vi har utviklet to svært like utgaver av LDF1A; «LDF1A_v.0.3» for nRF51 DK og «LDF1A_v.0.3_Dongle» for nRF51 Dongle. Eneste forskjell mellom de to er GPIO-konfigurasjon for TWI SCL og SDA.

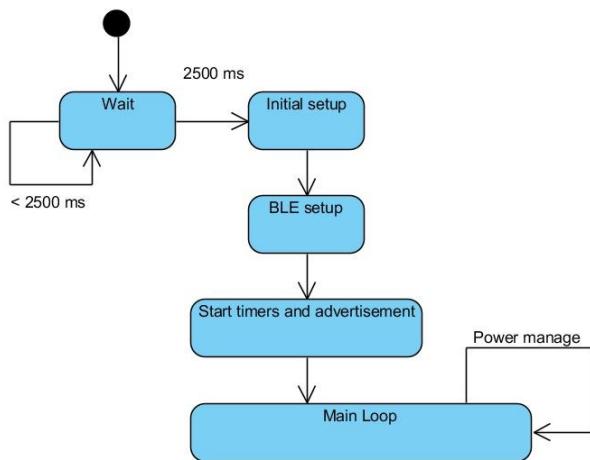
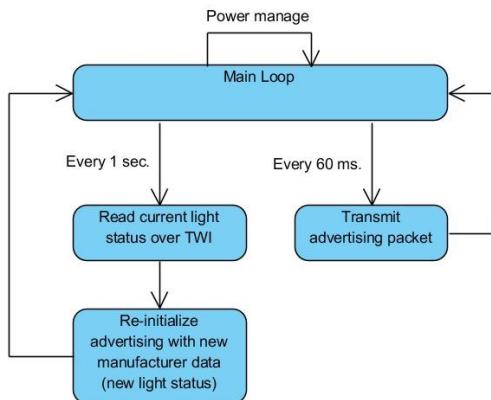
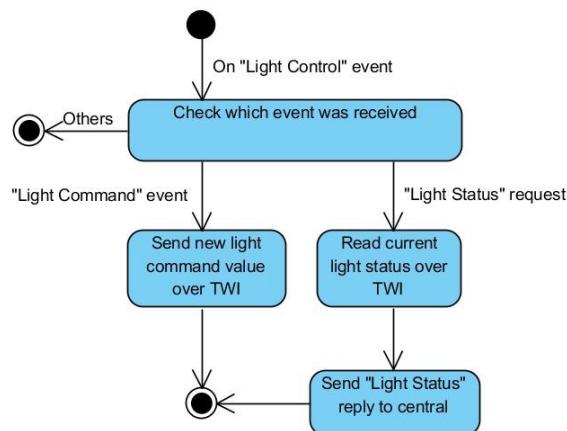
Sleve koden er skrevet i C ved hjelp av Nordic Semiconductors Bluetooth Smart API og Keils µVision, og finnes i det digitale arkivet (Vedlegg 4).

4.3.2 KODEFLYT

Figur 61 Illustrerer den overordnede flyten i hovedkoden. Utførelsen starter med en 2,5 sekunds ventefunksjon før den går videre med oppsett av TWI, timere, BLE-stack og advertising. Dette gjøres for å hindre potensielle oppstartsproblemer knyttet til transientforløp i den tilkoblede modulens strømforsyning. Deretter startes de forskjellige funksjonene og hovedsløya går i en power manage-syklus i påvente av Bluetooth-events.

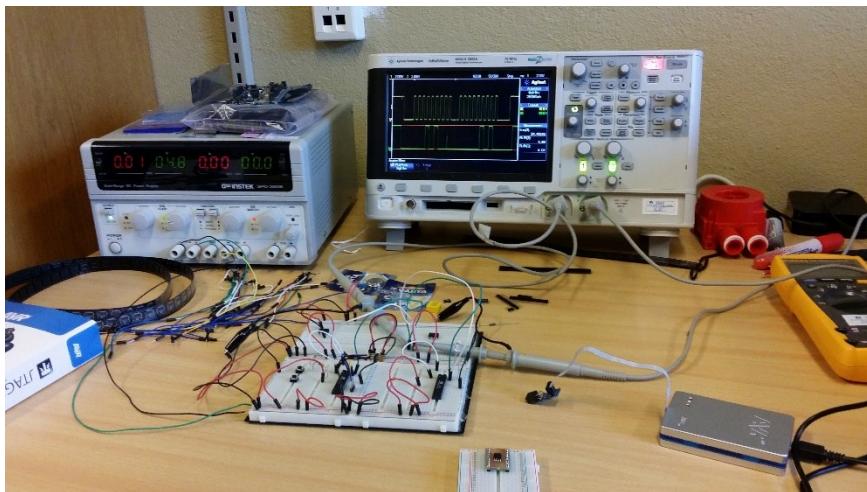
LDF1A opererer i broadcast-modus så lenge ingen central-enheter er tilkoblet. Figur 62 illustrerer hvordan advertisingen henger sammen med hovedsløya.

Når en central kobler seg til, stanses alle timere og advertisingen opphører. LDF1A går dermed over i connected-mode og avventer instrukser fra central. Figur 63 illustrerer hvordan LDF1A responderer på forespørslar og events.

**FIGUR 61. FIRMWARE, TILSTANDSDIAGRAM FOR LDF1As HOVEDSLØYFE****FIGUR 62. FIRMWARE, SAMMENHENGBRUK MELLOM MAIN LOOP OG ADVERTISING, LDF1A****FIGUR 63. FIRMWARE, TILSTANDSDIAGRAM FOR "LIGHT CONTROL" EVENTS, LDF1A**

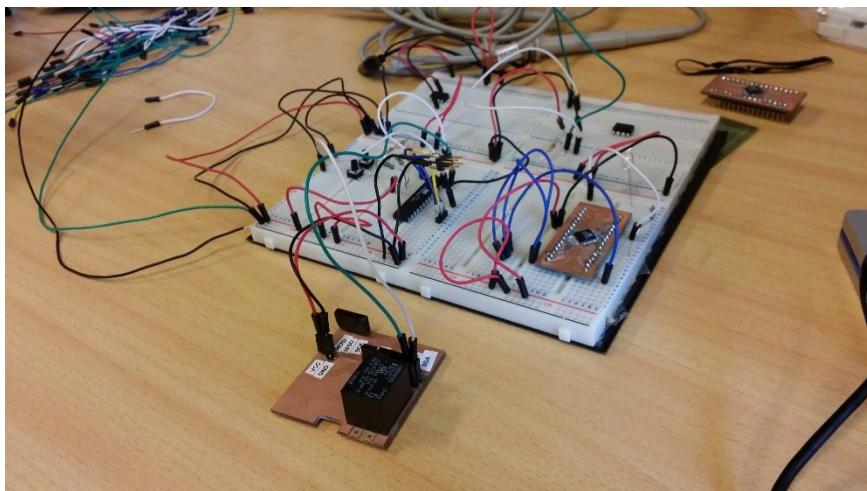
4.4 TESTING

Til å begynne med foregikk all firmwaretesting på midlertidige breadboard-prototyper. Figur 64 viser et tidlig testoppsett for TWI mellom to ATmega48V-kontrollere, hvor den ene var satt opp som master og den andre som slave. Masterenheten i dette tilfellet var koblet opp med to trykknapper og kjørte en tidlig versjon av LDFBIN1_MASTER. Den oppførte seg med andre ord som en Switchmodul. Slaven på sin side var programmert med en tidlig versjon av LDFBIN1_SLAVE og emulerte funksjonaliteten til en Binærmodul ved å skru av og på en LED.



FIGUR 64. TESTING, TESTOPPSETT FOR TWI-KOMMUNIKASJON MELLOM SWITCH- OG BINÆRMODUL

Etter hvert som utviklingen av hardware for Binær- og Dimmermodulene gikk videre, fulgte firmwaretestingen etter. Figur 65 viser en tidlig prototyp av Binærmodulen som testes i kombinasjon med den provisoriske Switchmodulen fra tidligere. Rent funksjonelt sett var ikke forskjellen stor fra det forrige eksempelet, men dette var første gang vår egenproduserte hardware og firmware ble testet som en integrert enhet.

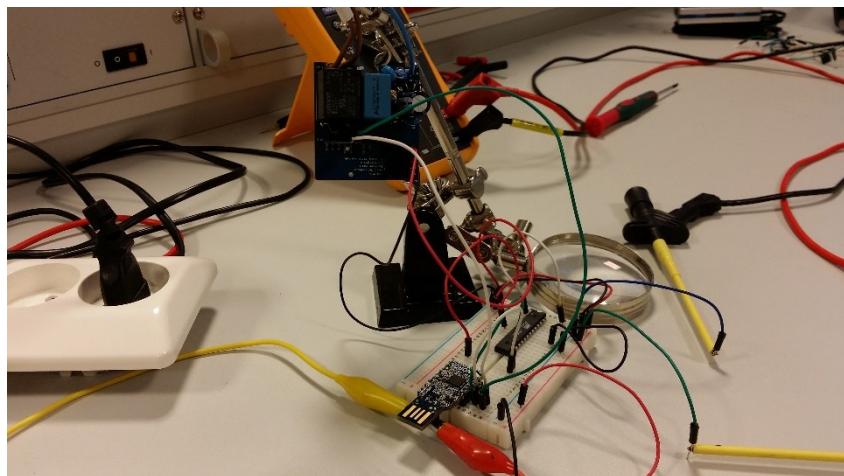


FIGUR 65. TESTING, OPPSETT FOR FIRMWARETESTING AV TIDLIG BINÆRPROTOTYP

Senere i prosjektutførelsen ble også firmware for Bluetooth Smart gjort klar til testing. Utviklingskortet nRF51 DK erstattet dermed Switchmodulen på breadboardet, men slaveenheten forble mer eller mindre den samme. Til å begynne med hadde den bare støtte for broadcast-mode, og kunne dermed bare lese av verdier fra slaven og legge disse til advertising-pakkene, men etter hvert fikk Bluetooth-firmwaren full funksjonalitet og den virkelige testingen kunne begynne.

Figur 66 viser oppkobling av en nRF51 Dongle i rollen som Kontrollmodul og en nesten ferdig Binærmodul. Dette oppsettet var fullt operativt og hadde støtte for alle de samme funksjonene som det ferdige produktet.

Kontrollmodulen fungerte i både broadcast- og connected-mode, og hadde ingen problemer med å ta imot og behandle lysstyringskommandoer og lysstatusmeldinger.



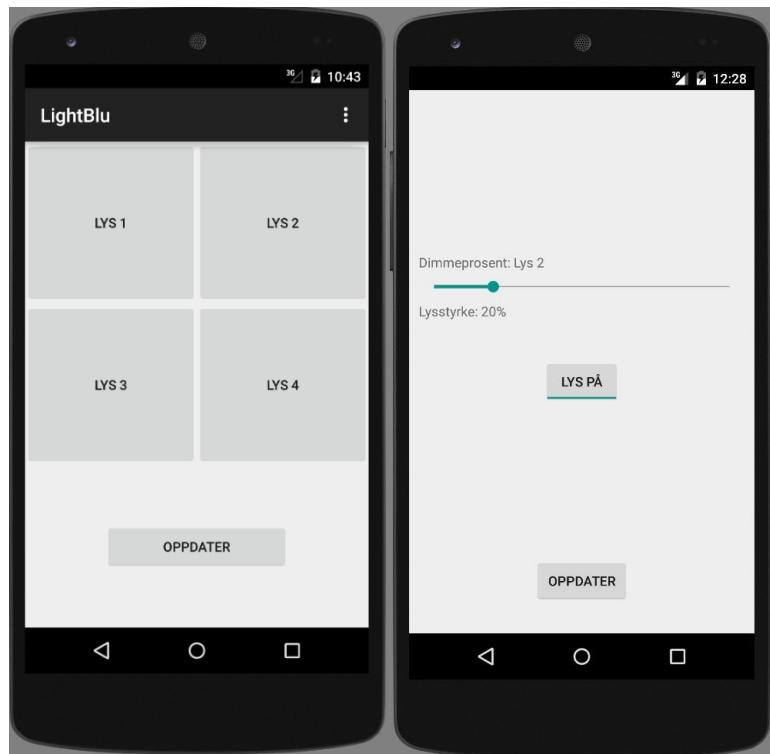
FIGUR 66. TESTING, TESTOPPSETT AV NÆRT FERDIGSTILT BINÆRMODUL OG BLUETOOTH SMART KONTROLLMODUL

KAP.5 MOBILAPPLIKASJON

5.1 OPPRINNELIG PLAN

Før Bluetooth ble implementert i applikasjonen ble det kun jobbet med det grafiske. Vi fikk til et grafisk resultat som vi syntes var tilfredsstillende, med smarte løsninger og et oversiktlig brukergrensesnitt. Planen var i utgangspunktet at vi skulle ha en scanneaktivitet hvor enhetene ville legge seg i små bokser på skjermen. På disse boksene skulle det stå navn på enheten, lysstatus og om lysenheten hadde mulighet for dimming eller ikke. Ved å trykke på en av disse boksene skulle det åpne seg en ny aktivitet. Dersom det var en binærmodul ville det være to knapper, en knapp for å slå lyset av og en knapp for å slå lyset på. For en Dimmermodul ville det vært en triggerknapp, altså en knapp med to forskjellige tilstander. Dersom knappen hadde vært i tilstanden på, ville en slider vises på skjermen som man kunne justere dimmingen av lyset med.

På grunn av problemer med å kombinere grafisk design med Bluetooth Smart-teknologien, ble vi nødt til å justere planene våre for hvordan applikasjonen skulle se ut.



FIGUR 67. OPPRINNELIG PLAN, SCANNEAKTIVITET OG AKTIVITET FOR STYRING AV LYS

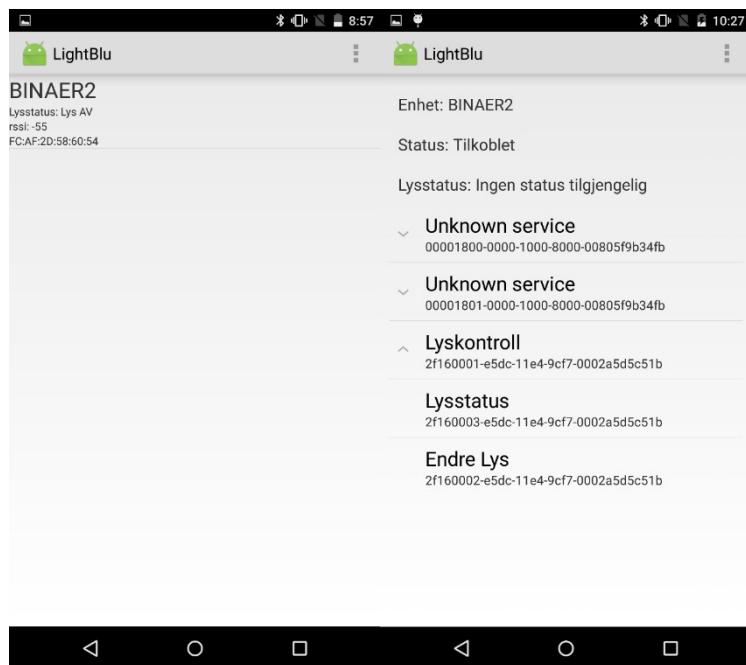
5.2 UTVIKLINGSVERKTØY

Til utvikling har vi benyttet oss av Android Studio v. 1.1.0 med Android SDK Tools rev 24.1.2. For at applikasjonen skal ha støtte for Bluetooth Smart, har vi brukt API versjon 18, som tilsvarer at man kan kun kjøre applikasjonen på enheter med Android versjon 4.3 eller nyere. Utover dette har vi brukt utviklingssidene til Android som ligger på nett [44] og Nordic Semiconductors GIT-hub [45] som støtte til arbeidet. Applikasjonen har blitt testet på en Motorola Nexus 6 som kjører Android versjon 5.1.

5.3 GRAFISK DESIGN

I vårt arbeid med applikasjonen har vi valgt å vektlegge funksjonalitet foran grafisk design. Derfor er det lite finesser og et veldig enkelt brukergrensesnitt. Ved oppstart av appen er det første man møter er main aktiviteten, som tar seg av scanning etter Bluetooth Smart enheter som annonser innenfor rekkevidde. Disse enhetene legger seg i en liste på skjermen. For hver enhet står enhetens navn, hvilken lysstatus som annonses samt rssi-verdi. Scanningen foregår over et bestemt tidsintervall, som vi har valgt å sette til ti sekunder. Dersom man ønsker å scanne på nytt etter disse ti sekundene må man inn i menyen som er plassert i høyre hjørne på toppen av skjermen, som har to menyvalg. Et valg for å starte en ny scan, og et for å stoppe scanningen dersom det skulle være ønskelig. Scanneskjermen har en minimalistisk utforming, som fører til god oversikt over de forskjellige elementene.

Når enheten man ønsker å regulere dukker opp på scanneskjermen, kan man trykke på den og det opprettes en tilkobling til enheten. Det vil samtidig åpnes en ny aktivitet hvor man øverst kan se navnet på enheten, om man er tilkoblet og lysstatus. Nedenfor vises servicene som enheten tilbyr i en liste. For vårt tilfelle vises servicen «Lyskontroll», med underkarakteristikkene «Lysstatus» og «Endre lys». I menyen finner man valgmuligheter for å koble til eller koble fra enheten.



FIGUR 68. APPLIKASJON, SCANNEAKTIVITET OG AKTIVITET FOR STYRING AV LYS

5.4 IMPLEMENTERING AV BLUETOOTH SMART

Vi har benyttet en allerede eksisterende Bluetooth Low Energy eksempelkode fra Android. Denne koden har blitt kraftig modifisert tilpasset vårt formål.

Når man starter appen gjennomgås rutiner for å gjøre den klar til bruk. Under ser vi et utdrag fra *MainActivity.java*. Ved oppstart (*onCreate*) skal innstillingar som har blitt lagret ved tidligere bruk lastes inn. Videre ser vi at navnet «LightBlu» settes på verktøylinjen. Til slutt sjekker applikasjonen om enheten som brukes har støtte for Bluetooth Smart. Dersom den ikke har det, lukkes applikasjonen.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mHandler = new Handler();
    getActionBar().setTitle("LightBlu");

    //Sjekker om enheten støtter Bluetooth Smart
    if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
        Toast.makeText(this, "Ingen støtte for Bluetooth Smart", Toast.LENGTH_SHORT).show();
        finish();
    }
}
```

For å kunne ta i bruk Bluetooth-komponenter i applikasjonen må det initialiseres et Bluetooth adapter

```
// Initialiserer Bluetooth adapter.
final BluetoothManager bluetoothManager =
    (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();

// Sjekker etter støtte for Bluetooth.
if (mBluetoothAdapter == null) {
    Toast.makeText(this, "Ingen støtte for Bluetooth", Toast.LENGTH_SHORT).show();
    finish();
    return;
}
```

Over vises koden for at man skulle kunne ta i bruk Bluetooth-objekter og standard Bluetooth funksjoner. Samtidig må man spesifisere tillatelse til bruk av Bluetooth i AndroidManifest-fila, slik:

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

MainActivity.java inneholder også kode for å håndtere Bluetooth Smart-enheter som har blitt oppdaget gjennom en scan. De blir lagt i en liste, og applikasjonen sørger for at navnene til enhetene vises, hva som advertises, adresse, samt rssi, som er en indikasjon på mottatt signalstyrke. Ved å trykke på en av disse enhetene åpnes en ny aktivitet. Innholdet i aktiviteten er definert i klasse-fila *DeviceControlActivity.java*. Som nevnt i kapittel 5.3, vil en liste med servicer og characteristics vises på skjermen etter å ha trykket på en enhet.

```
private final ExpandableListView.OnChildClickListener servicesListClickListner =  
new ExpandableListView.OnChildClickListener() {  
  
    @Override  
    public boolean onChildClick(ExpandableListView parent, View v, int groupPosition,  
                               int childPosition, long id) {  
  
        if (mGattCharacteristics != null) {  
            final BluetoothGattCharacteristic characteristic  
            mGattCharacteristics.get(groupPosition).get(childPosition)  
            final int charaProp = characteristic.getProperties();  
  
            if ((charaProp & BluetoothGattCharacteristic.PROPERTY_READ) > 0) {  
                mBluetoothLeService.readCharacteristic(characteristic);  
            }  
  
            if ((charaProp & BluetoothGattCharacteristic.PROPERTY_WRITE) > 0) {  
                mBluetoothLeService.writeCharacteristic(characteristic, write_data);  
            }  
  
            if ((charaProp & BluetoothGattCharacteristic.PROPERTY_NOTIFY) > 0) {  
                mNotifyCharacteristic = characteristic;  
                mBluetoothLeService.setCharacteristicNotification(  
                    characteristic, true);  
            }  
        }  
    }  
}
```

FIGUR 69. APPLIKASJON, KODE SOM VISER HENDELESFORLØPET VED Å TRYKKE PÅ EN CHARACTERISTIC

Figur 69 viser koden for hva som skjer dersom man trykker på en characteristic. En characteristic kan ha tre egenskaper; read, write og/eller notify. Dersom den har egenskapen read, kan man hente ut informasjon som er lagret i characteristicen i form av en tallverdi. Dersom den har egenskapen write, kan man endre verdien som allerede er lagret. Med egenskapen notify, vil brukeren få beskjed dersom verdien endrer seg. Så ved å trykke på en characteristic i listen sjekker applikasjonen hvilken egenskap den har, og utfører en operasjon deretter. I teorien kan en characteristic ha alle tre egenskapene samtidig, men det er ikke tilfellet for de vi har laget. Vi har laget to characteristicer som vi har valgt å kalle «Lysstatus» som er read, og «Endre lys» som er write.

For å styre en lysmodul slik at man endrer lysinstilling, trykker man på characteristicen «Endre Lys». Samtidig som applikasjonen sjekker hvilken egenskap characteristicen har, går den også gjennom en if-setning for å avgjøre hvilken kommando som skal sendes til lysmodulen. Dersom det er en dimmermodul man ønsker å styre, blir en verdi for den gjeldende lysstatusen, lastet inn i en global variabel. Det er innholdet i denne variablene som er betingelsen gjennom if-setninga.

```

if (Buffer.lysadv == 0x0A)
{
    write_data[0] = 0x0C;
    Buffer.lysadv = 0x0C;
}
else if (Buffer.lysadv == 0x0C)
{
    write_data[0] = 0x0E;
    Buffer.lysadv = 0x0E;
}

else if (Buffer.lysadv == 0x0E)
{
    write_data[0] = 0x10;
    Buffer.lysadv = 0x10;
}
else if (Buffer.lysadv == 0x10)
{
    write_data[0] = 0x12;
    Buffer.lysadv = 0x12;
}
else if (Buffer.lysadv == 0x12)
{
    write_data[0] = 0x14;
    Buffer.lysadv = 0x14;
}
else if (Buffer.lysadv == 0x14)
{
    write_data[0] = 0x0A;
    Buffer.lysadv = 0x0A;
}

```

FIGUR 70. APPLIKASJON, IF-SETNING SOM SØRGER FOR AT RIKTIG VERDI BLIR SKREVET TIL CHARACTERISTIC

I Figur 70 ser vi hva som skjer. «lysadv» er variabelen som lysstatusen lagres i idet man kobler seg til en Dimmermodul. Dersom lyset er skrudd helt av, vil lysstatusen være 0x0A. 0x0C er verdien for neste dimmetrinn, som lagres i variabelen write_data, før den sendes til dimmermodulen, og lyset endrer seg. Samtidig bytter «lysadv» verdi til 0x0C, slik at neste gang man trykker på «Endre lys» fortsetter if-setninga. Med dette har vi skapt en måte å gå gjennom dimmetrinnene kronologisk fra fullstendig dimmet, til minst mulig dimming. Den opprinnelige planen var at man skulle kunne velge dimmetrinn uten å måtte gå gjennom alle trinnene kronologisk, men på grunn av en stram fremdriftsplan, og at vi slet med å få dette til, ble vi nødt til å utelate det. Mer om dette i Kap 8.1.3.

Det samme prinsippet gjelder for binærmodulen. Forskjellen er at det kun er to statuser, enten av eller på. Dette fungerer akkurat slik som det var tiltenkt. Dersom lyset er av og man trykker «Endre lys», slår det seg på, og motsatt.

5.5 TESTING

Når vi har testet applikasjonen har vi benyttet oss av en Motorola Nexus 6, som vi har lånt av Nordic Semiconductor. Applikasjonen lastet vi over fra PC til Nexusen via USB. Android Studio har også en emulator som kunne brukes til testing, begrensningen er at den ikke støtter Bluetooth i det hele tatt, men før implementeringen av Bluetooth Smart i applikasjonen, ble emulatoren flittig anvendt til testing av grafisk design og under fasen når programmeringen skulle læres.

Siden lysmodulene ikke var ferdigstilt når prototyper av applikasjonen skulle testes, ble det satt opp en testrigg bestående av nRF51 development-kit levert av Nordic, som gjorde Bluetooth-kommunikasjon mulig. I tillegg til dev-kitet brukte vi en mikrokontroller, programmert med firmware som vi har utviklet selv, koblet til en LED som simulerer en helt vanlig lyspære. Vi hadde også et oscilloskop koblet til riggen, slik at det var mulig å se hvilke kommandoer mikrokontrolleren mottok.

I første omgang var hovedfokuset på å få til å slå på LEDen, som vi fikk til uten problemer. Deretter fikk vi til å både slå av og på LEDen, selv om dette var mer problematisk å få til. Dimming var ikke mulig å teste i praksis, men siden vi hadde muligheten til å overvåke hvilke kommandoer som ble sendt, kunne vi observere at kommandoene for de ulike dimmetrinnene ble sendt korrekt.

Rett før prosjektets slutt fikk vi mulighet til å teste applikasjonen på ferdige lysmoduler med en vanlig lampe koblet til. Det viste seg at det virket tilfredsstillende, slik som vi hadde forventet. Kun små justeringer og fjerning av unødvendig kode ble utført etter dette.

KAP.6 INTEGRERING OG RESULTAT

6.1 SYSTEMSAMMENSETNING

Systemet vi har utviklet består primært av tre moduler. Kontrollmodul, binærmodul og Dimmermodul. Kontrollmodulen hadde vi i utgangspunktet tenkt å utvikle selv, men siden Nordic har utviklet nRF-51 Bluetooth Smart dongle, som oppfyller de samme kravene som vi ønsket at kontrollmodulen skulle ha, valgte vi heller å implementere dongelen i systemet vårt. Kontrollmodulens oppgave er å håndtere Bluetooth Smart-kommunikasjon, og generere styresignaler til enten binærmodul eller Dimmermodul, alt etter hvilken av disse den er koblet til.

Binærmodul mottar styresignal fra kontrollmodul for deretter å skru utgangen av/på.

Dimmermodul mottar styresignal fra kontrollmodul for deretter å utføre dimming eller skru av/på strøm på utgangen

I tillegg til disse tre modulene inngår også en Android applikasjon i systemet, som installeres på hvilken som helst Android-enhet med støtte for Bluetooth Smart, slik at man har en smart-telefon som i praksis fungerer som en fjernkontroll for styring av lyset.

6.2 KOMMUNIKASJONSPROSESER

«LightBlu» er et sammensatt system bestående av ulike elementer som kommuniserer både gjennom trådløs og integrert teknologi. Under vises to tilfeller for hvordan kommunikasjonsflyten fungerer. I det første tilfellet beskrives hva som skjer dersom det ikke er noen styring, en form for «idle state». Det andre tilfellet beskriver overgangen fra idle state til det som skjer når man kobler seg til en lysmodul med mobilapplikasjonen, og forsøker å endre lyset.

Tilfelle 1: «Idle state», ingen styring fra applikasjon.

- 1) Lysstatusen er lagret i den lokale mikrokontrolleren, i form av en heksadesimal verdi.
- 2) Dongelen sender en gang i sekundet en read-forespørrelse til mikrokontrolleren over TWI, hvor den ber om å få ut lysstatus.
- 3) Mikrokontrolleren initialiserer en write-operasjon, hvor den sender lysstatus tilbake til dongelen.
- 4) Dongelen adviserer lysstatusen over Bluetooth Smart, slik at andre enheter med støtte for teknologien kan oppfatte lysmodulen, og se lystilstanden.

Tilfelle 2: Fra «Idle state» til «connected state», og endring av lys ved bruk av mobilapplikasjon.

- 1) Lysstatusen er lagret i den lokale mikrokontrolleren, i form av en heksadesimal verdi.
- 2) Dongelen sender en gang i sekundet en read-forespørrelse til mikrokontrolleren over TWI, hvor den ber om å få ut lysstatus.
- 3) Mikrokontrolleren initialiserer en write-operasjon, hvor den sender lysstatus tilbake til dongelen.
- 4) Dongelen adviserer lysstatusen over Bluetooth Smart, slik at andre enheter med støtte for teknologien kan oppfatte lysmodulen, og se lystilstanden.
- 5) På smart-telefonen gjennomfører man en scan ved bruk av applikasjonen, slik at lysmodulen vises. Deretter kan man sende en forespørrelse til dongelen om å opprette en tilkobling.
- 6) Dongelen svarer med en bekreftelse på at tilkoblingen har blitt opprettet, eventuelt mislykket. Kontrollmodulen går over i en «connected state» og slutter å advertise.
- 7) Når man er tilkoblet kan man endre lyset. Da sendes en write-kommando fra telefonen til dongelen.
- 8) Dongelen videreforsyner kommandoen til mikrokontrolleren.
- 9) Mikrokontrolleren sjekker opp mot firmware om den mottatte kommandoen er valid. Dersom den ikke er valid, forkastes kommandoen, og lysstatusen forblir uendret.
- 10) Dersom kommandoen er valid, sender mikrokontrolleren instrukser til releet/triac, og verdien for lysstatusen i mikrokontrolleren oppdateres.

11) Dongelen sender en kvittering på at write-kommandoen var vellykket til applikasjonen.

For å forstå hvordan kommunikasjonsflyten fungerer er det lurt å se for seg systemet i to separate nettverk. Det ene nettverket som består av smart-telefonen og Bluetooth-dongelen, tar seg av den trådløse kommunikasjonen via Bluetooth Smart. Her fungerer telefonen som master (også kalt central), mens dongelen fungerer som slave (også kalt peripheral).

På den andre siden finner man den interne kretsen som består av Bluetooth-dongelen, den lokale mikrokontrolleren samt ulike elektroniske komponenter. I dette nettverket har dongelen rolle som master, mens mikrokontrolleren er slave. Kommunikasjonen mellom disse foregår over TWI.

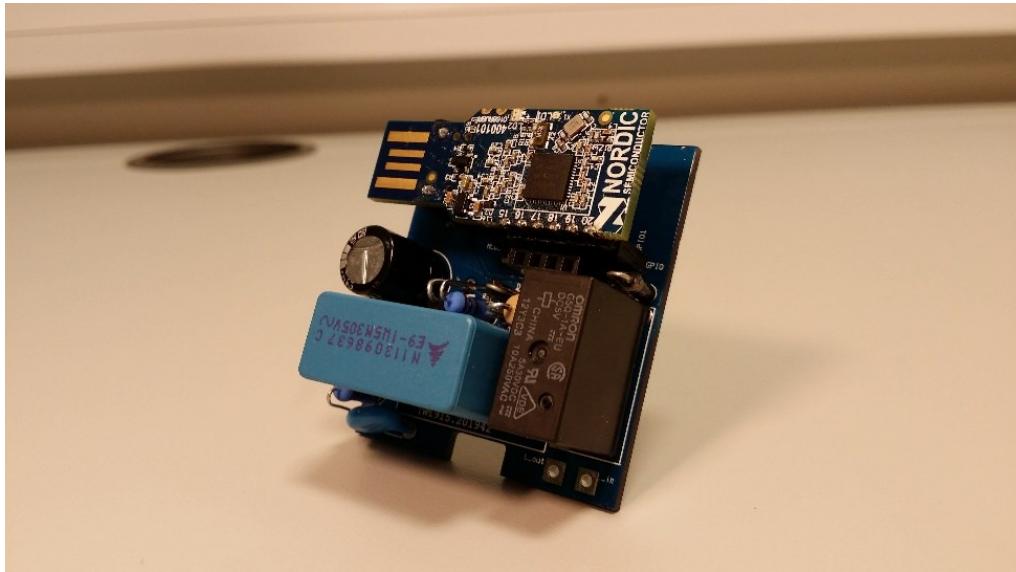
Som vi ser er Bluetooth-dongelen en del av begge nettverkene. Dette er fordi den fungerer som et bindeledd mellom smart-telefonen og mikrokontrolleren. Den henter ut informasjon fra mikrokontrolleren som den sender videre til smart-telefonen, og den mottar kommandoer fra smart-telefonen som den sender videre til mikrokontrolleren.

6.3 SLUTTRESULTAT

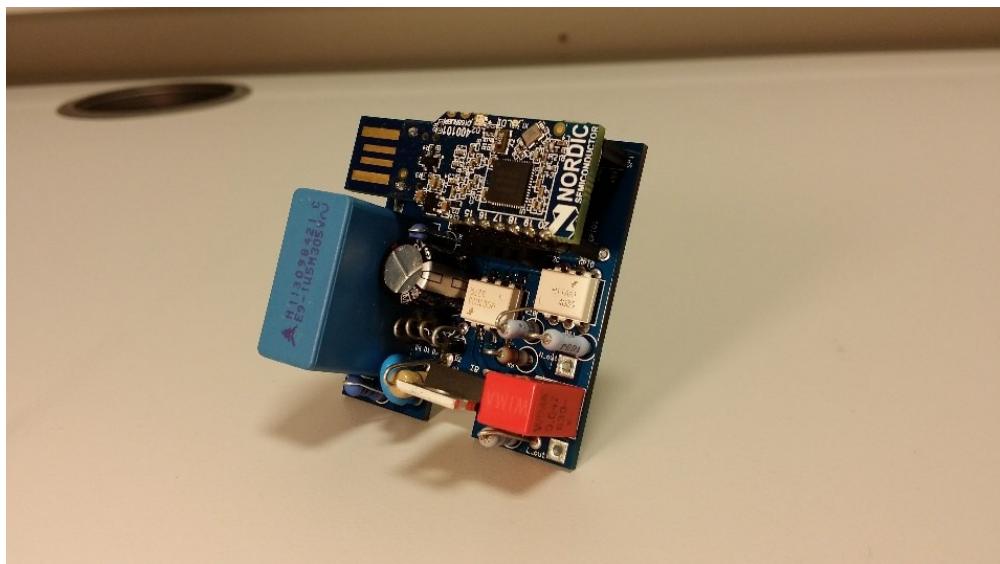
Det endelige fysiske resultatet er to produkt av typen «veggplugg»; den ene med Dimmerfunksjon og den andre med Binærfunksjon.

Figur 71 og Figur 72 viser de komplette funksjonsenheterne, som utgjør innmaten i disse to veggpluggene.

Figur 73 viser hvordan funksjonsenheten passer inn i nexa-chassiet.



FIGUR 71. SLUTTRESULTAT, BINÆR FUNKSJONHET



FIGUR 72. SLUTTRESULTAT, DIMMER FUNKSJONHET



FIGUR 73. SLUTTRESULTAT, DIMMERMODUL PÅSSERT I NEXA-CHASSIET

6.4 SLUTTMÅLINGER

Slutproduktenes egenskaper dokumenteres i dette delkapitlet, i form av en serie sluttmålinger. Denne måleserien danner sluttdokumentasjonen til sluttproduktene.

For alle målinger av den kapasitive strømforsyningen i Kap 6.4.1 og 6.4.2 gjelder:

Agilent MSO-X-2002A Oscilloscope

Kanal 1 (gul): spennin over utgang av spenningsregulator

Kanal 2 (grønn): spennin over zenerdiode

MERK: Strømgjennomgangen fra utgangen av spenningsregulatoren har blitt målt ved å plassere en 1,1 ohms motstand i serie med utgangen. Spenningsnivået over motstanden er så målt og omregnet til sin respektive strømverdi.

Grunnet problemer med støy og måleoppsett har vi ikke tilgjengelige strømmålinger for alle forløpene til binær og dimmermodul, men de målingene vi har tilfredsstiller våre forventninger om kretsenes spennings- og strømegenskaper.

6.4.1 BINÆRMODUL

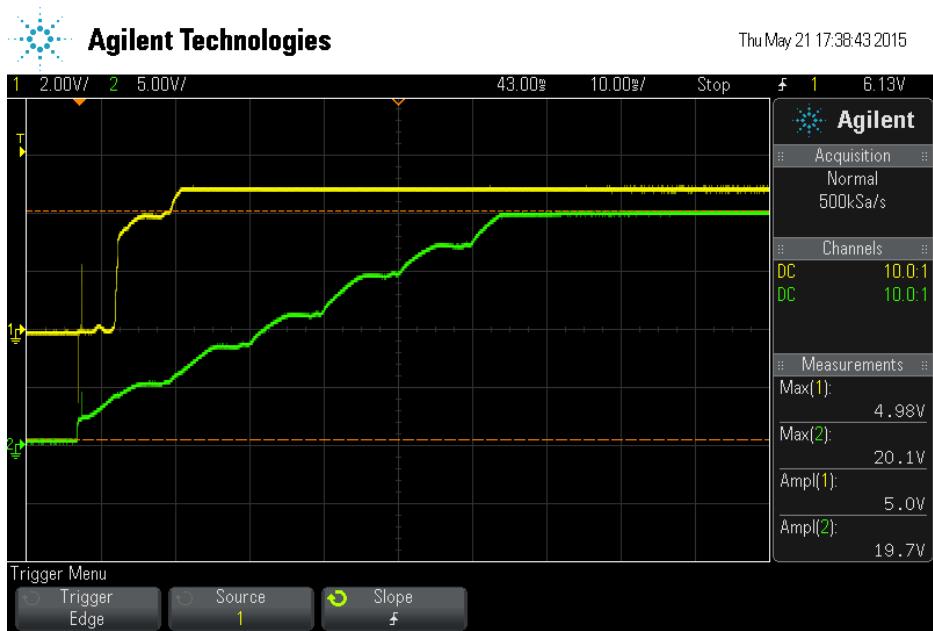
Kapasitiv strømforsyning – Transientforløp for spennin over zenerdiode og regulator med og uten tilkoblet kontrollmodul:

Vi ser en klar endring i binærmodulens innsvingningsforløp når den påmonteres sin kontrollmodul.

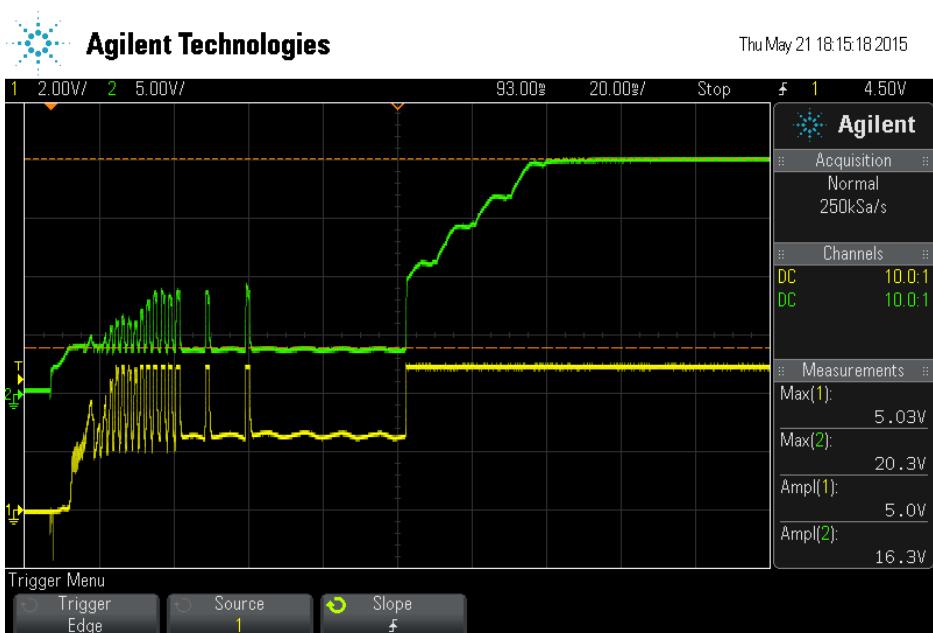
Innsvingningstiden for regulatoren øker fra 15 til 90 ms, mens zenerdioden øker fra 60 til 130 ms.

Det transiente spenningsforløpet med tilkoblet kontrollmodul viser at spenningen over zenerdioden er utenfor operativt område over lengre tid, før den klarer å øke spenningen sin nok til å drive spenningsregulatoren optimalt.

Dette skyldes med stor sannsynlighet at Segger programmeringschipen er aktiv under initialisering av kontrollmodulen.



FIGUR 74. SLUTTMÅLINGER, TRANSIENTSPENNING OVER ZENERDIODE OG UTGANG REGULATOR. KONTROLLMODUL FRAKOBLET



FIGUR 75. SLUTTMÅLINGER, TRANSIENTSPENNING OVER ZENERDIODE OG UTGANG REGULATOR. KONTROLLMODUL TILKOBLET

Kapasitiv strømforsyning – Transient- og stasjonærforløp for strømgjennomgang på utgang av regulator med og uten tilkoblet kontrollmodul:

Disse to målingene er dessverre preget av mye støy, men vi ser en klar forskjell i kretsens strømforbruk med og uten tilkoblet kontrollmodul.

Figur 76 viser det transiente forløpet til kretsens strømtrekk uten tilkoblet kontrollmodul. Bak alle støykomponentene ser man en strømkurve med middelverdi på omtrent 4,95 mV.

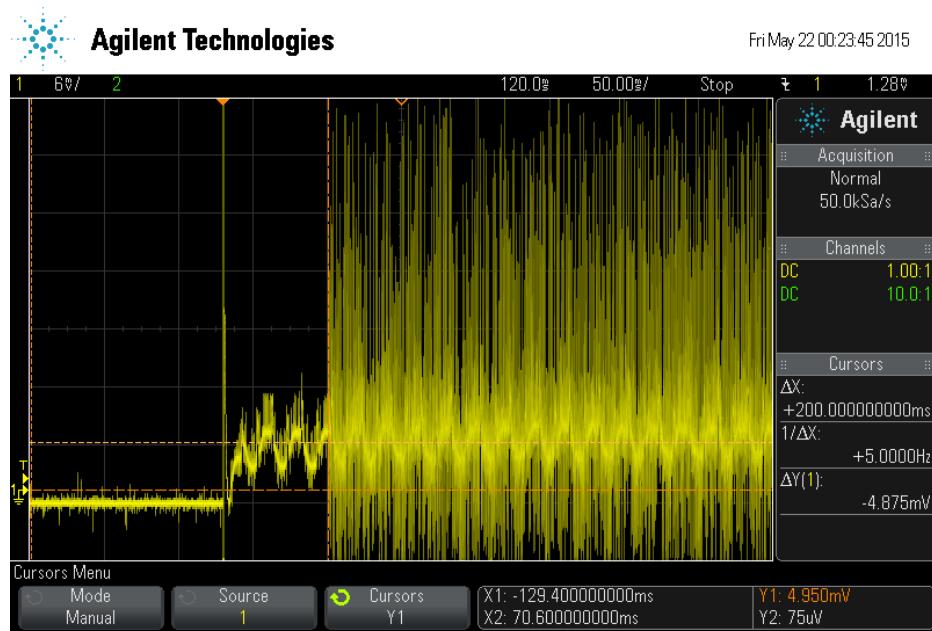
$$I_{stasjonær} = \frac{U}{R} = \frac{4,95 \cdot 10^{-3} V}{1,1 \Omega} = 4,5 \text{ mA} \quad P = U * I = 5 V * 4,5 \text{ mA} = 22,5 \text{ mW}$$

Dette tilsvarer et strømtrekk på tilnærmet 4,5 mA, noe som tilsvarer omtrent en tredjedel av det maksimale strømtrekket til den lokale hovedkontrolleren, som er den eneste tilkoblede enheten under målingen.

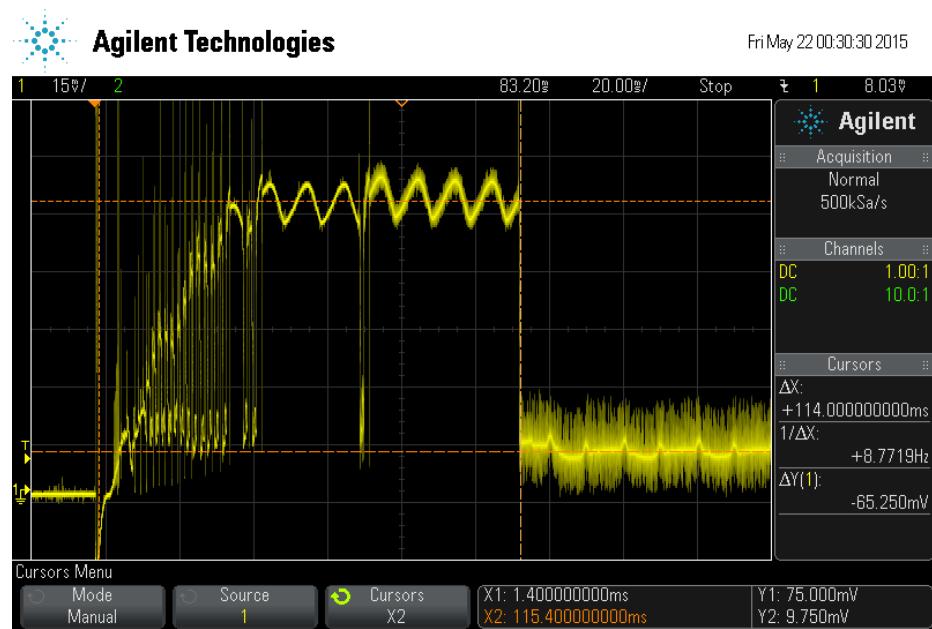
Vi ser fra Figur 77 at det transiente strømtrekket med tilkoblet kontrollmodul er vesentlig høyere, og at det med stor sannsynlighet er derfor zenerdioden bruker lang tid på å oppnå sin stasjonære verdi.

$$I_{transient} = \frac{U}{R} = \frac{75 \cdot 10^{-3} V}{1,1 \Omega} = 68 \text{ mA}$$

$$I_{stasjonær} = \frac{U}{R} = \frac{9,75 \cdot 10^{-3} V}{1,1 \Omega} = 8,9 \text{ mA}$$



FIGUR 76. SLUTTMÅLINGER, TRANSIENT OG STASJONÆRT STRØMFORBRUK FRA REGULATOR. KONTROLLMODUL FRAKOBLET



FIGUR 77. SLUTTMÅLINGER, TRANSIENT OG STASJONÆRT STRØMFORBRUK FRA REGULATOR. KONTROLLMODUL TILKOBLET

Kapasitiv strømforsyning - Stasjonær strømgjennomgang på utgang av regulator med tilkoblet kontrollmodul, advertisement og connected mode:

$$I_{stasjonær,rele\ inaktivt} = \frac{U}{R} = \frac{4,5*10^{-3}\ V}{1,1\ \Omega} = 4,1\ mA$$

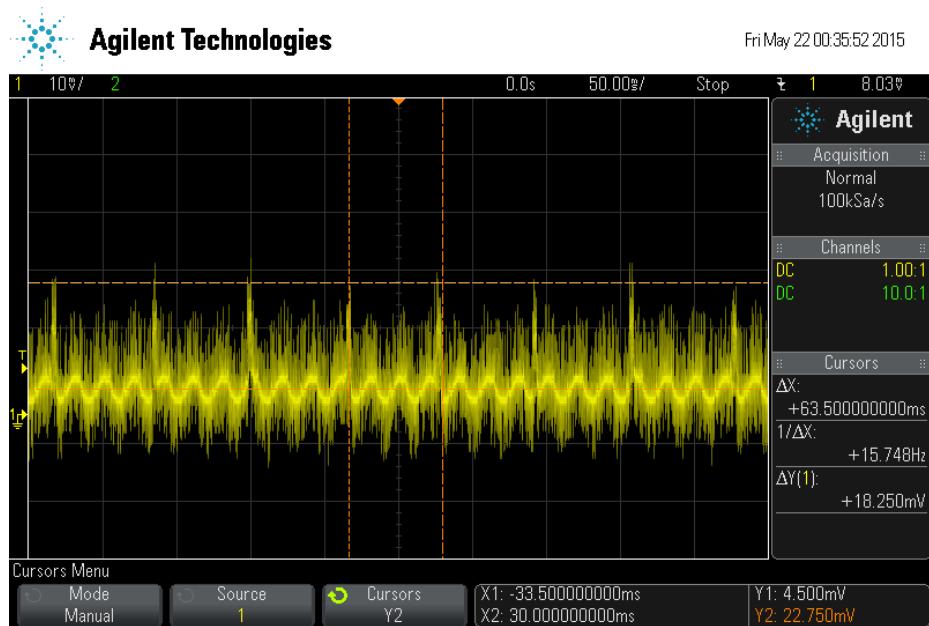
$$I_{peak,advertising} = \frac{U}{R} = \frac{22,75*10^{-3}\ V}{1,1\ \Omega} = 20,6\ mA$$

$$I_{stasjonær,rele\ aktivt} = \frac{U}{R} = \frac{55,475*10^{-3}\ V}{1,1\ \Omega} = 50,4\ mA$$

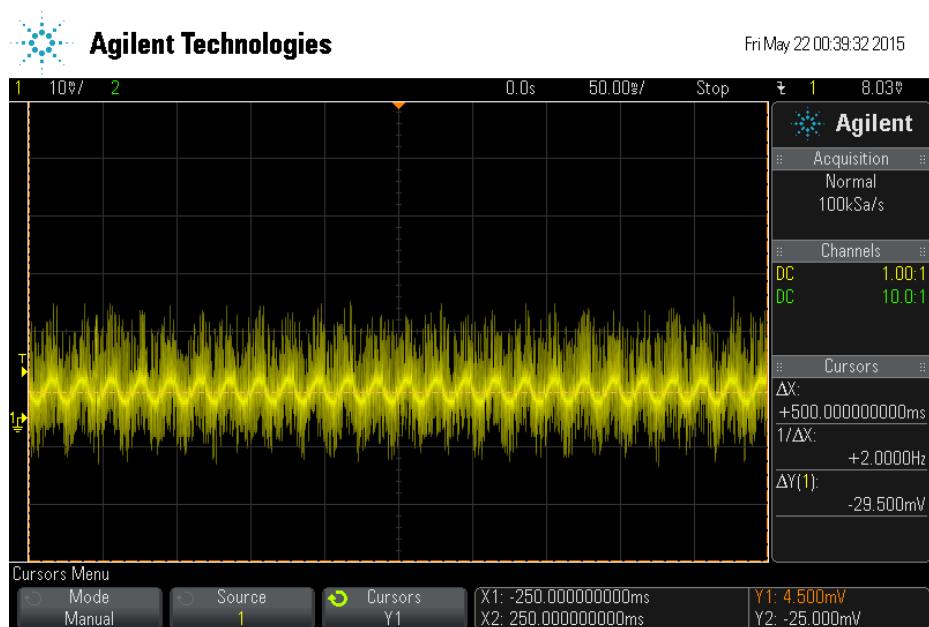
For å kontrollere lyset vil man måtte koble til kontrollmodulen og aktivere reléet, en operasjon som går gjennom tre stadier:

Figur 78 viser det stasjonære strømtrekket i kretsen når kontrollmodulen opererer i advertising mode. Man ser tydelig på kretsens strømtrekk at intervallet mellom hver topp er omtrent 60 ms, samme intervall som gjelder for sending av advertising-pakker. Kontrollmodulen får oppdatert sin lysstatus en gang pr sekund ved hjelp av TWI-kommunikasjon med den lokale hovedkontrolleren, vist i figur Figur 81.

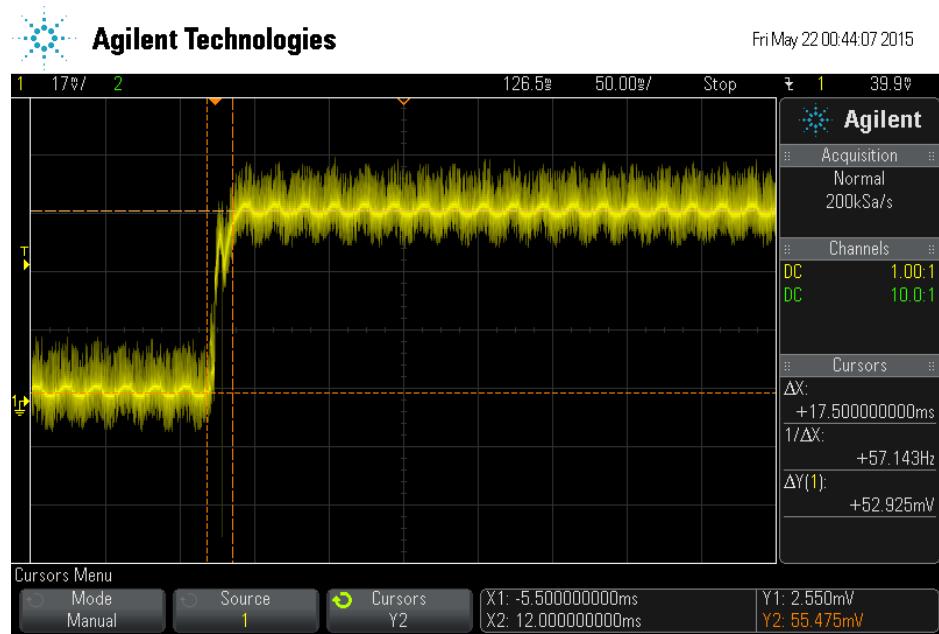
Fra Figur 79 ser vi at kontrollmodulen opererer i connected mode, og at den har avsluttet sending av advertising-pakker. Det stasjonære strømtrekket utenfor advertisement-intervallet er omtrent samme for begge modes; omtrent 4,1 mA. Ved å aktivere reléet øker strømtrekket til 50,4 mA stasjonært, som vist på figur Figur 80.



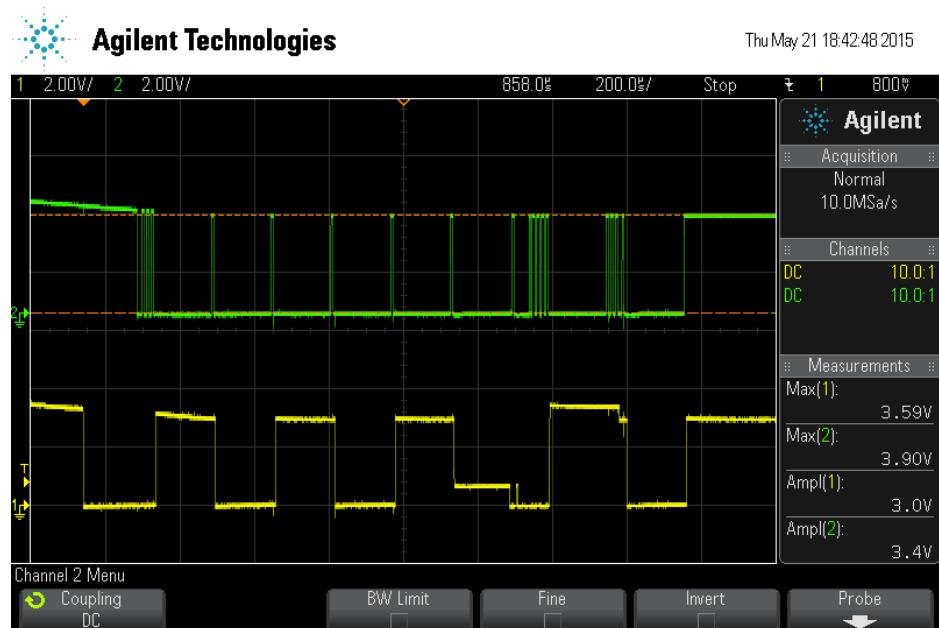
FIGUR 78. SLUTTMÅLINGER, STASJONÆRT STRØMFORBRUK FRA REGULATOR. ADVERTISEMENT MODE



FIGUR 79. SLUTTMÅLINGER, STASJONÆRT STRØMFORBRUK FRA REGULATOR. CONNECTED MODE



FIGUR 80. SLUTTMÅLINGER, TRANSIENT STRØMFORBRUK FRA REGULATOR. CONNECTED MODE. RELÉ AKTIVERES



FIGUR 81. SLUTTMÅLINGER, EKSEMPEL PÅ TWI-KOMMUNIKASJON

6.4.2 DIMMERMODUL

Dimming av lysnivå – Nullgjennomgangsdeteksjon, styresignal og spenningsbølge:

Se Vedlegg 2.5 for en skjematiske forståelse av dimmermodulens oppbygning.

Dimmermodulens optokobler OPC1 får tilført vekselspenning gjennom to seriemotstander på sin primærsiden, hvor vekselspenningen likerettes gjennom to dioder i anti-parallell. Den likerettede spenningen driver en lysdiode som gir energi til base-inngangen i npn-transistoren på sekundærskjemaet.

Kollektor-inngangen på sekundærskjemaet er tilkoblet den kapasitive strømforsyningen gjennom motstand R7 parallelt med GPIO1 på lokal hovedkontroller, mens emitter er tilkoblet jord. Når spenningen som tilføres primærsiden av optokobleren er over 0,7 volt for positiv og under -0,7 volt for negativ halvbølge vil diodene lede og drive lysdioden, som igjen aktiverer basis-inngangen på transistoren. Med basisinngangen aktivert vil transistoren lede strøm mellom kollektor og emitter, og spenningen på kollektor føres ned til jord via emitter.

Når forsyningsspenningen på primærsiden går mot nullpunktet vil den krysse spenningspunktet for henholdsvis 0,7 eller -0,7 volt, diodene vil sperre og basisinngangen på transistoren blir ikke aktivert. Dette fører igjen til at overgangen mellom kollektor og emitter stenger, og spenningen på kollektor går til lokal hovedkontroller. Signalet kan måles på pinne GPIO1

Den lokale hovedkontrolleren vil oppfatte dette som at en nullgjennomgang i vekselspenningen er i ferd med å skje, og må gjøre en beregning av hvor lenge den skal vente med å aktivere gaten på triacen, avhengig av hvilket lysnivå den har fått satt som referanse. Et typisk nullgjennomgangssignal er illustrert i figur nn7, hvor den faktiske nullgjennomgangen i spenningstilførselen skjer ved toppen av hver spenningsbølge på kanal 1.

Når den lokale hovedkontrolleren har ventet tilstrekkelig lenge for å oppfylle referanseverdien for lys sender den en puls lik figur nn7 kanal 2 på trigger-utgangen sin, som står tilkoblet basisinngangen på transistor Q1. Emitter på transistor Q1 står tilkoblet jord, og kollektor står tilkoblet optokobler OPC2s primærsiden. Primærsiden av OPC2 består av en diode, hvor anoden er tilkoblet den kapasitive strømforsyningen via en motstand og katoden er tilkoblet kollektor hos Q1.

Ved å føre et styresignal fra triggerutgangen hos lokal hovedkontroller til basis på transistor Q1 vil transistoren starte å lede, og lede en strøm fra den kapasitive strømforsyningen gjennom dioden i OPC2s primærside, emitter-kollektor overgangen i transistoren og ned til jord. Med en strømgjennomgang i OPC2s diode vil den på samme måte som OPC1 drive en lysdiode, som igjen aktiverer optokoblerens sekundærskjema. Optokobler OPC2s sekundærskjema består av en diac, som er tilkoblet vekselspenningen og gaten på triacen. Med diacen aktivert vil den lede strøm fra vekselspenningstilførselen og til gaten på triacen, som i sin tur aktiverer triacen og slipper gjennom vekselspenningen fra det punktet den er i halvperioden sin og ut til en last. Triacen vil være aktiv helt til vekselspenningen krysser nullpunktet, og må startes igjen med samme prosedyre for hver halvperiode av vekselspenningen.

Ved å regulere tidsintervallet fra nullgjennomgangsdeteksjonen skjer og til styresignalet fra den lokale hovedkontrolleren skal sendes til triacen kan vi regulere bølgeformen som sendes ut til lasten, og dermed regulere lysstyrken.

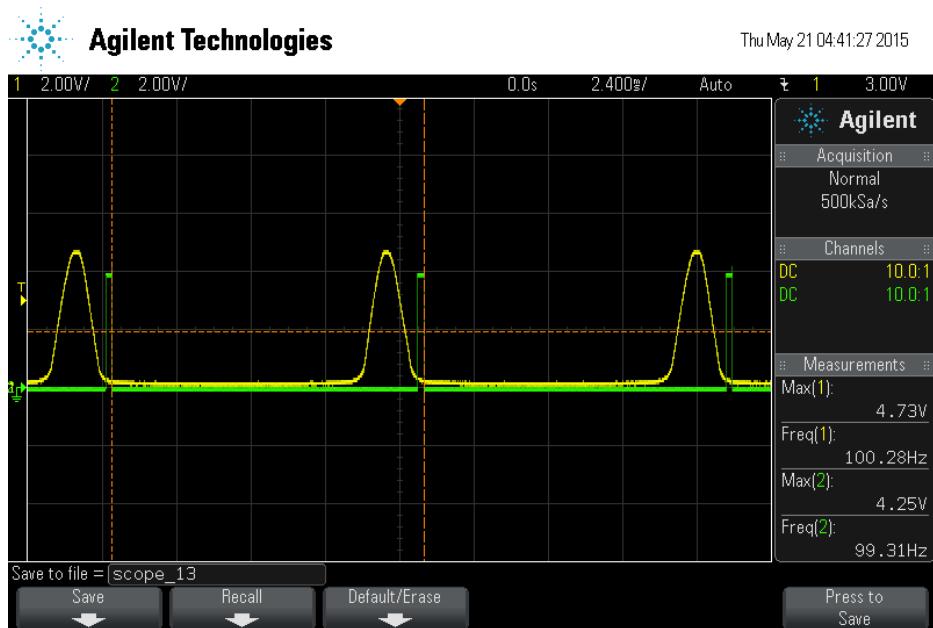
Tabell 8 viser en oversikt over nullgjennomgangsdeteksjon, styresignal fra lokal hovedkontroller og målt spenning over last. Figurtype A viser korrelasjonen mellom nullgjennomgangsdeteksjon og styresignal, mens figurtype B viser målt spenning over last.

MERK: Med last menes en 40 watt glødetrådpære

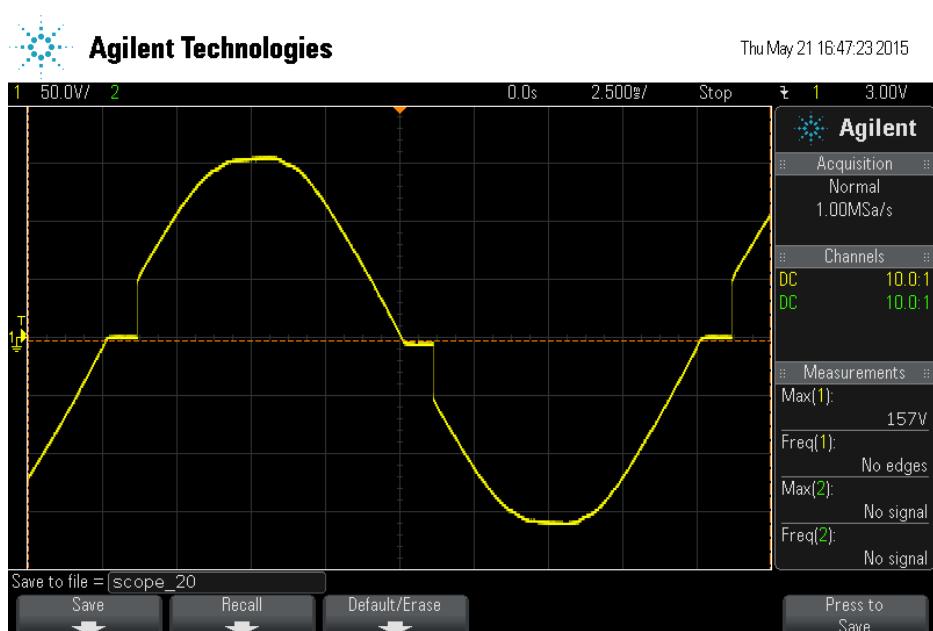
Spenningsnivået over lasten er målt med 20X måleprober, mens oscilloskopet er stilt inn med 10X forsterkning. Dette betyr at alle spenningsverdier som vises i figuren oppgis i $\frac{\text{spenningsnivå}}{2}$.

Gjeldende lysinnstilling	Lysnivå [%]	Figurtype A	Figurtype B
0x14	100	Figur 82	Figur 83
0x12	80	Figur 84	Figur 85
0x10	60	Figur 86	Figur 87
0x0E	40	Figur 88	Figur 89
0x0C	20	Figur 90	Figur 91
0x0A	0	Figur 92	Figur 93

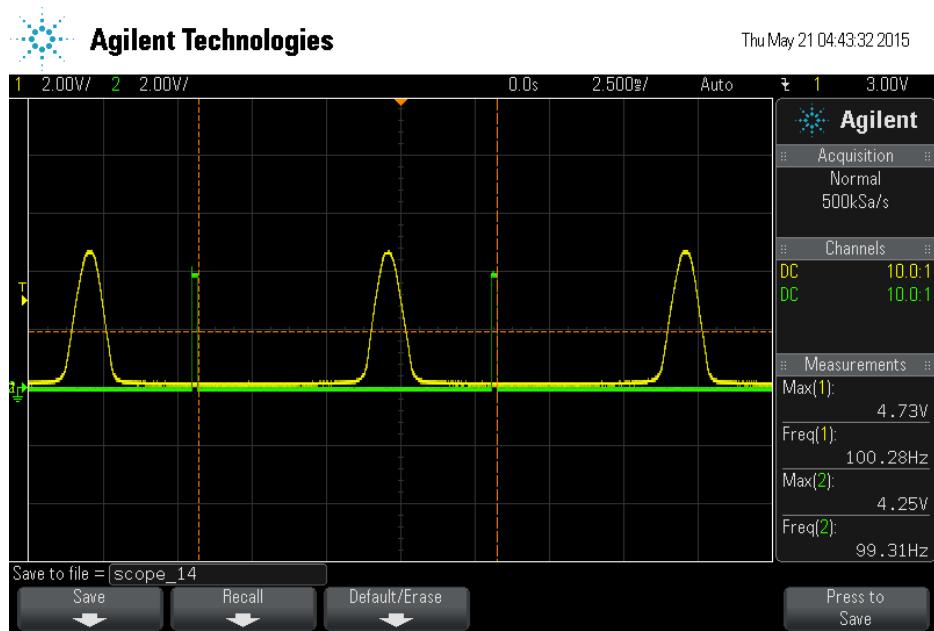
TABELL 8, SLUTTMÅLINGER, OVERSIKT OVER SLUTTMÅLINGER FOR DIMMERMODUL



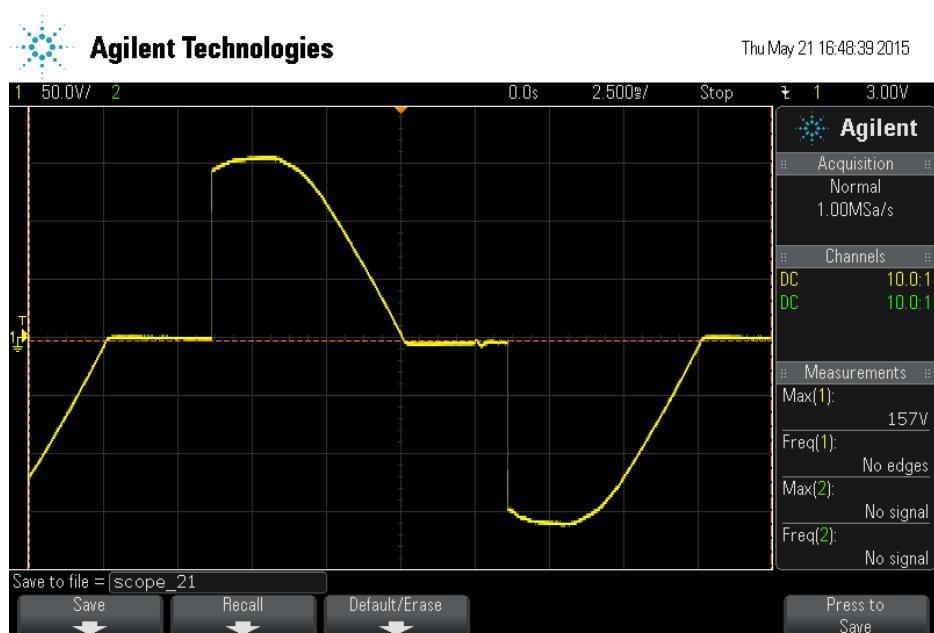
FIGUR 82. SLUTTMÅLINGER, NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 100 %



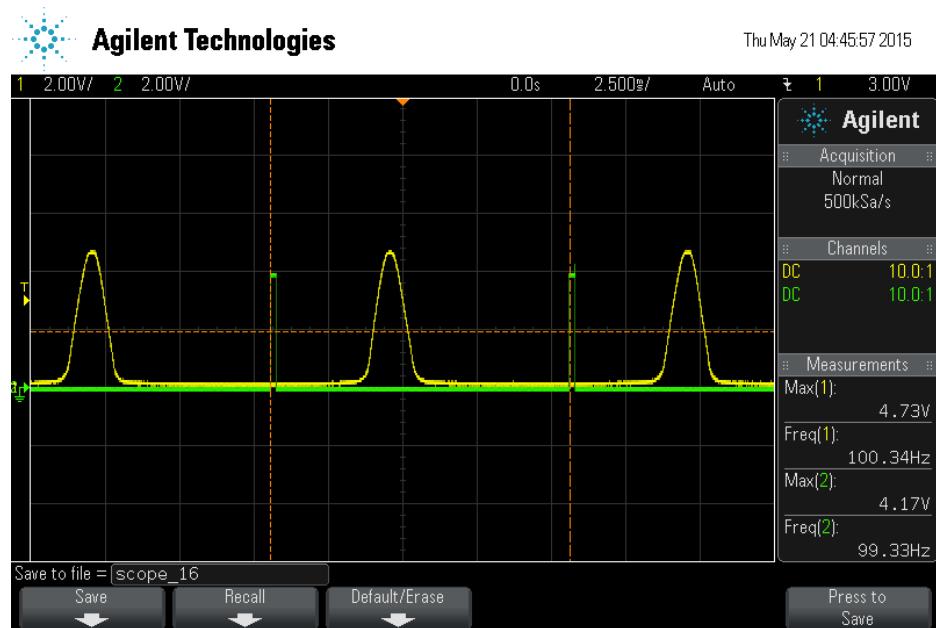
FIGUR 83. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 100%



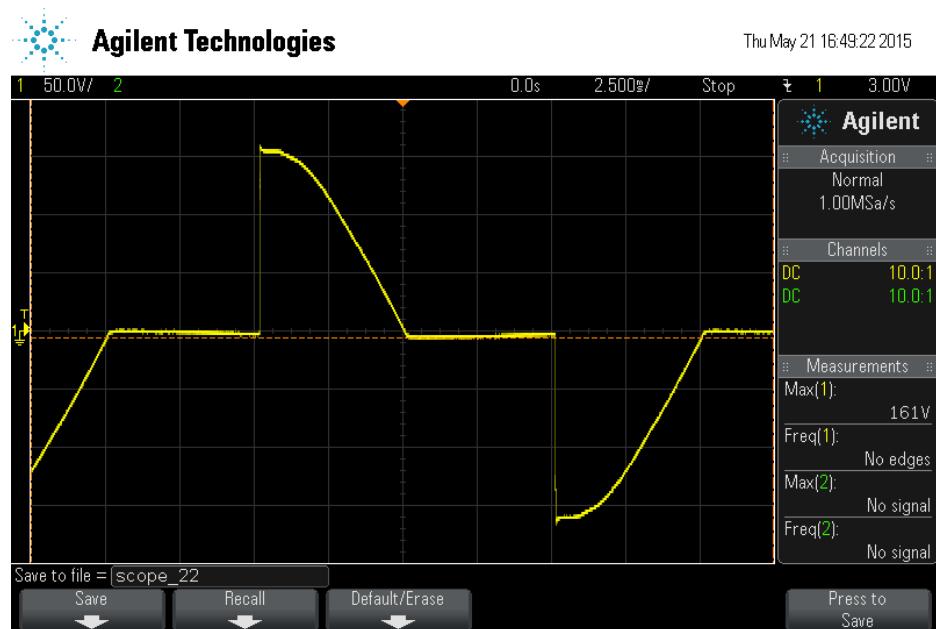
FIGUR 84. SLUTTMÅLINGER, NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 80 %



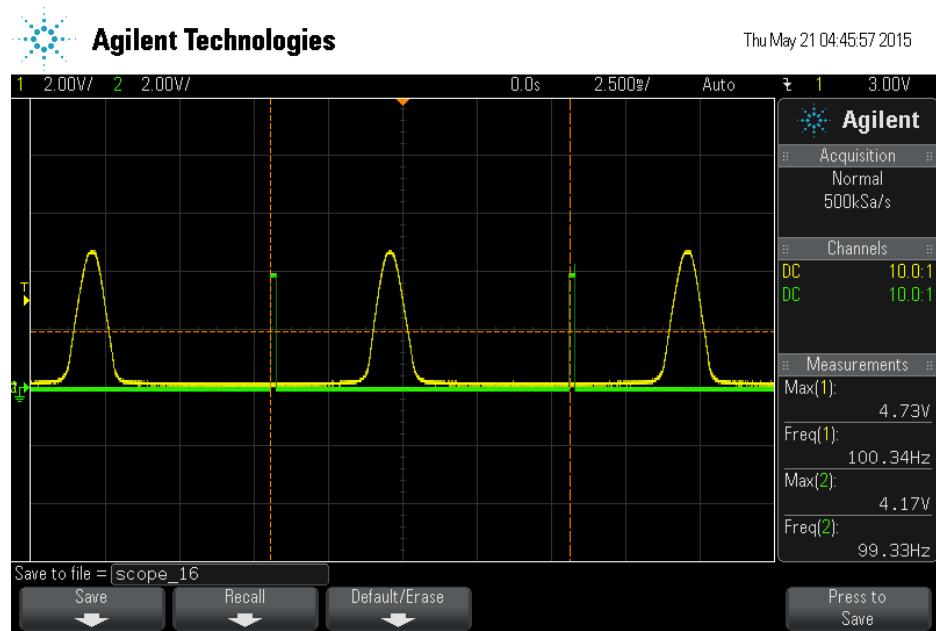
FIGUR 85. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 80 %



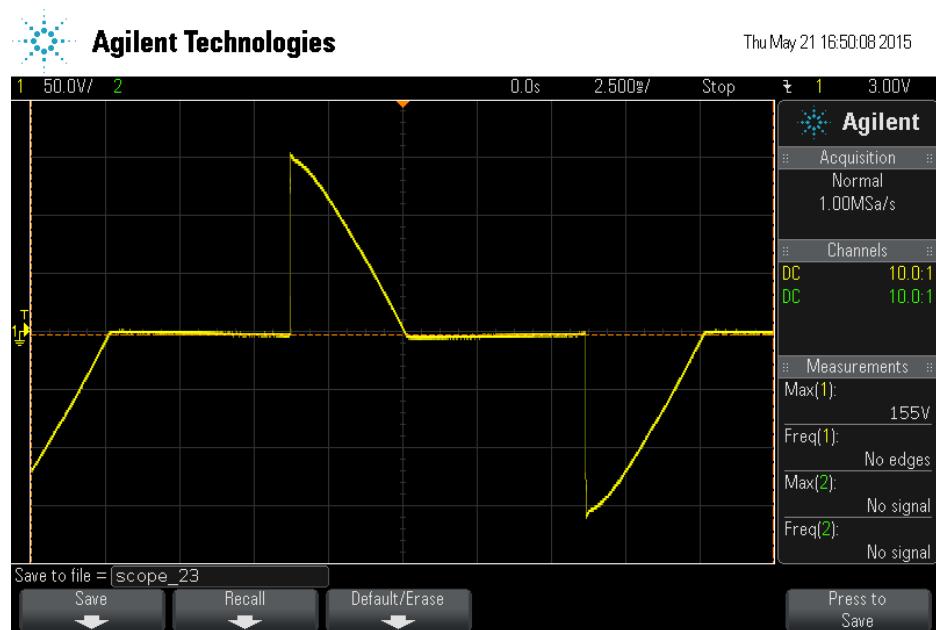
FIGUR 86. SLUTTMÅLINGER, NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 60 %



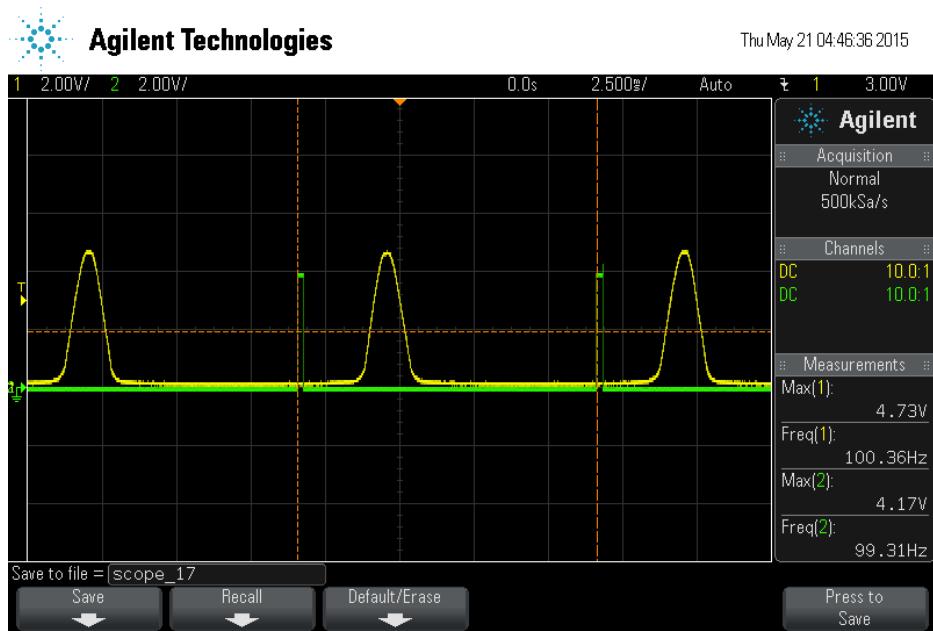
FIGUR 87. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 60 %



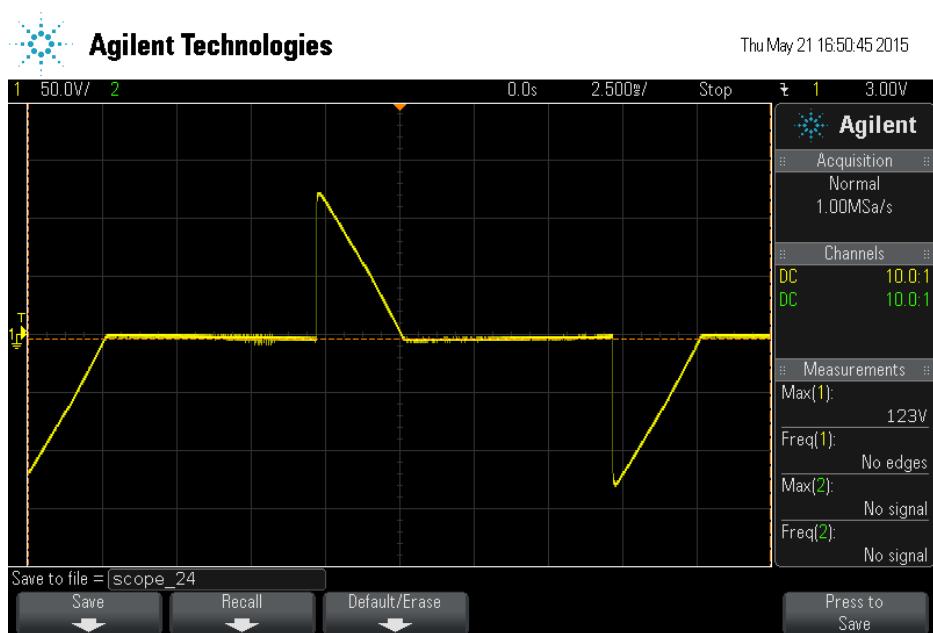
FIGUR 88. SLUTTMÅLINGER NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 40 %



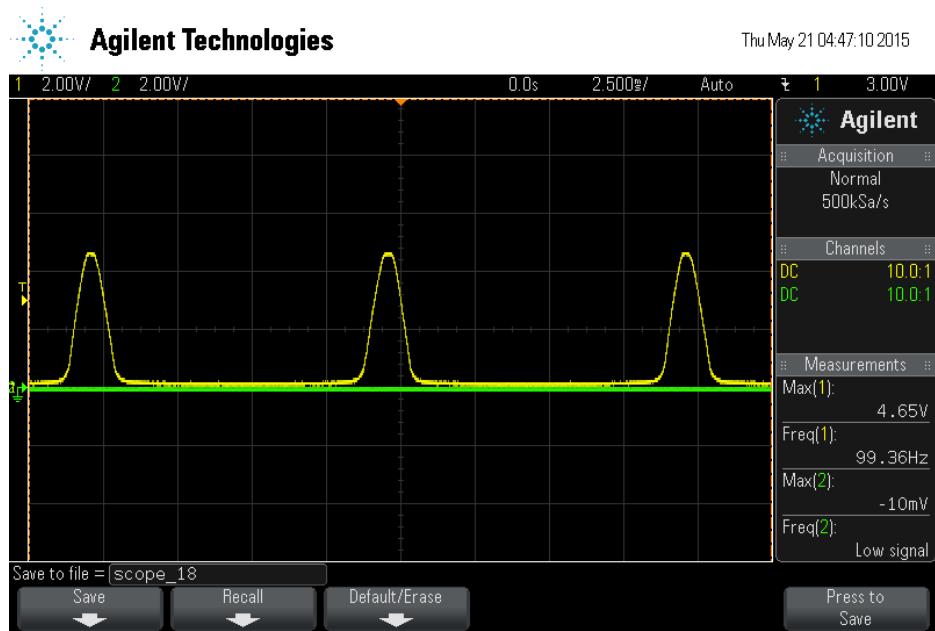
FIGUR 89. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 40 %



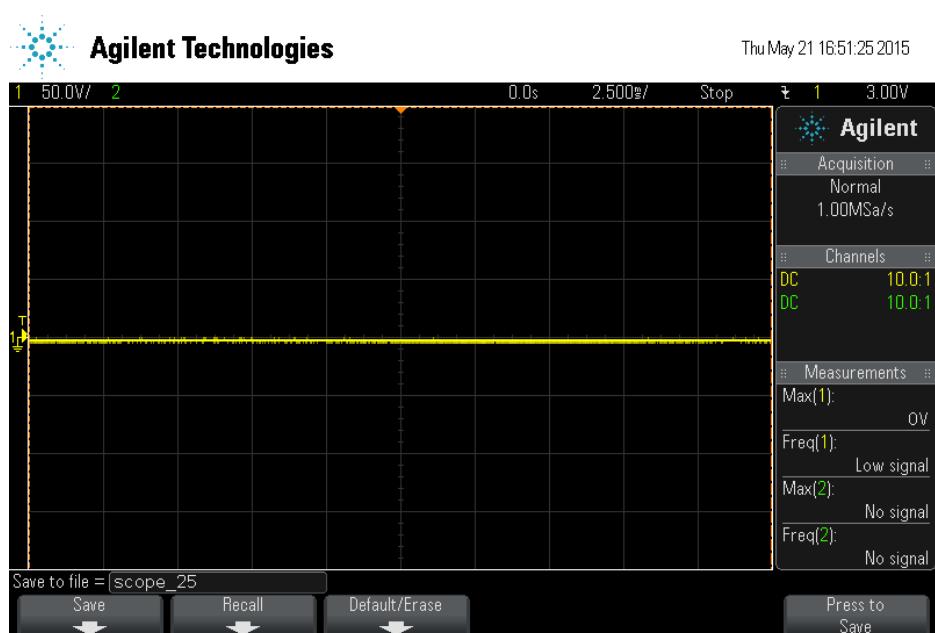
FIGUR 90. SLUTTMÅLINGER, NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 20 %



FIGUR 91. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 20 %



FIGUR 92. SLUTTMÅLINGER, NULLGJENNOMGANGSDETEKSJON OG STYRESIGNAL VED LYSNIVÅ 0 %



FIGUR 93. SLUTTMÅLINGER, SPENNINGSBØLGE OVER LAST VED LYSNIVÅ 0 %

Kapasitiv strømforsyning –

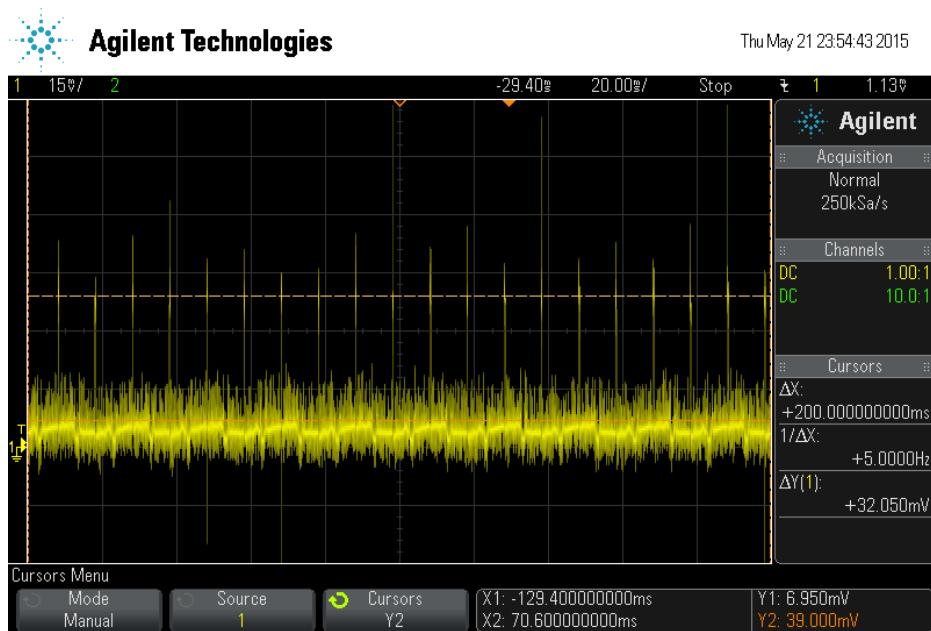
Vi ser av måleresultatene at den kapasitive strømforsyningen har tilnærmet lik oppførsel både for Binær- og Dimmermodul, og velger derfor å ikke legge vekt på presentasjon av resultatene til Dimmermodulen strømforsyning ettersom den er konfigurert likt som binærmodulens forsyning.

Den største forskjellen ligger i kretsens strømtrekk fra regulatoren når Dimmermodulen er aktiv. I motsetning til binærmodulen som har et stasjonært høyere strømtrekk når den driver reléet sitt opptrer Dimmermodulen mer likt kontrollmodulen i advertisement mode.

$$I_{stasjonær} = \frac{U}{R} = \frac{6,95 \cdot 10^{-3} V}{1,1 \Omega} = 6,32 mA$$

$$I_{peak} = \frac{U}{R} = \frac{39 \cdot 10^{-3} V}{1,1 \Omega} = 35,5 mA$$

Vi ser av Figur 94 at strømtrekket fra regulatoren har topper på omtrent 35,5 mA hvert 10. ms, noe som samsvarer med databeregningen den gjør ved nullgjennomgangsdeteksjonen hvert 10. ms.



FIGUR 94. SLUTTMÅLINGER, STRØMFORBRUK FRA REGULATOR. CONNECTED STATE. LYSINNSTILLING 0x10

KAP.7 PROSESSEN

7.1 PROSJEKTGRUPPA

Mads Ellingsen Stephansen – 01. Mai 1991

Mads er født og oppvokst i en liten bygd som heter Sigerfjord i Sortland kommune helt nord i Nordland. Her gikk han på realfagslinja ved Sortland Videregående Skole, før han deretter gjennomførte førstegangstjenesten i Hærens Ingeniørbataljon på Skjold i Indre Troms. Sterkt inspirert av enkelte familiemedlemmer gikk turen videre til Elektroingeniør-studiet ved NTNU i Trondheim, men etter et vanskelig første år med mye sykdom, flyttet han over til Høgskolen i Sør-Trøndelag. Her har han gjenopptatt elektroingeniør-studiet og går nå tredje og siste året med fordypning i elektronikk.

På fritiden interesser han seg for sport, og da særlig fotball, men hovedinteressen vil alltid være relatert til datamaskiner, herunder konstruksjon, modifisering, programmering, spilling og mye mer.

Petter Haugen – 14. November 1990

Petter er født og oppvokst i Trondheim. Studerte elektronikk ved Brundalen Videregående Skole, og tilegnet seg fagbrev som Dataelektroniker etter å ha vært den første lærlingen i NRK Trøndelags historie. Påbegynte så forkurs og senere ingeniørutdanning ved Høgskolen i Sør-Trøndelag, med stor inspirasjon gitt av familie samt kolleger i NRK Trøndelag. Har foruten lærlingejobben også relevant arbeidserfaring fra radio- og tv-verksted, samt nåværende deltidsjobb som drosjesjåfør. Er gjennom sin arbeidserfaring godt trent i problemløsning, og blir ofte kontaktet for råd forbundet med problemstillinger både av teknisk og annen art.

Interesserer seg i stor grad for elektronikk, musikk, trening og bil. Håper å kunne videreføre bilinteressen etter endt utdanning med fornyelse og utvidelse av bilparken.

Erlend Røed Myklebust – 15. September 1989

Erlend er født og oppvokst i en faglig sterk familie i Volda kommune på søre Sunnmøre, og har gjennom hele oppveksten vært lidenskapelig opptatt av teknologi og dens virkemåte.

Skolebakgrunnen hans består av studiekompetanse i realfag fra Volda Videregående Skole, samt en påbegynt bachelorgrad i elektronikk ved Høgskolen i Sør-Trøndelag. Fra 2008 til 2009 tjenestegjorde han i Hærens Artilleribataljon på Setermoen i Indre Troms, hvor han opparbeidet seg relevant ledererfaring og fikk svært gode skussmål. Arbeidserfaringen hans er i tillegg svært variert. Han har blant annet jobbet som drosjesjåfør, salgsmedarbeider og tekniker i en nettbutikk, og jobber nå som studentassistent i faget Anvendt Elektronikk ved Høgskolen i Sør-Trøndelag.

Erlend har erfaring innen programmering, mikrokontrollere og trådløs kommunikasjon, og har i tillegg god grunnleggende kunnskap om elektronikk og kretsdesign.

7.2 PROSJEKTPLAN

Ved prosjektets start delte vi inn arbeidet i fem faser.

Konseptfasen, hvor vi brukte tid på å faktisk finne ut hva vi ville oppnå med oppgaven. Ideer ble ytret og vi kom frem til flere forslag til hva vi skulle utvikle.

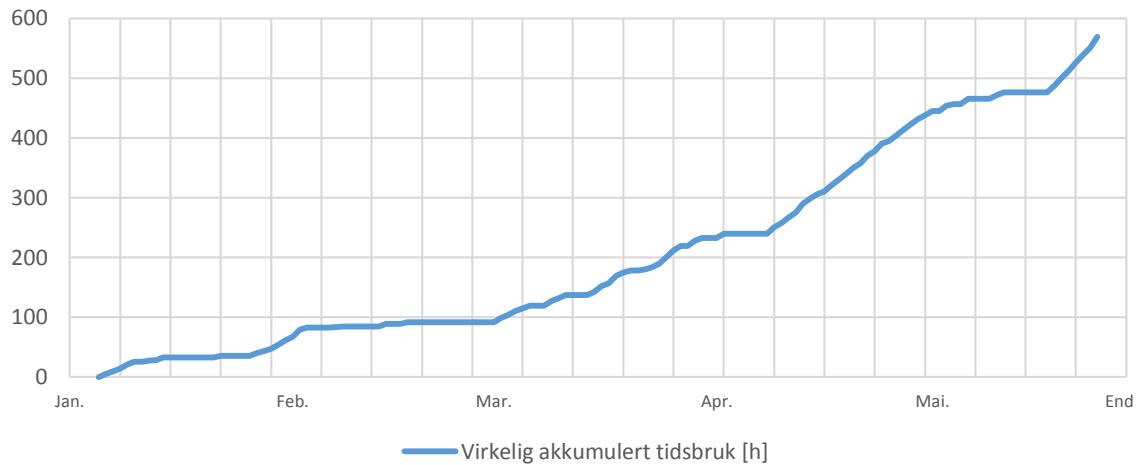
Planleggingsfasen. Når vi hadde bestemt oss for den løsningen som vi syntes var mest interessant og som samsvarer best til vår tolkning av oppgaven, satte vi i gang med planleggingen. Her ble det definert hvem som skulle ha ansvaret for hva. En fremdriftsplan ble også utformet, slik at vi skulle ha noe å forholde oss til underveis, og for å sikre progresjon.

Teorifasen/Teknisk studie. For at vi skulle ha mulighet til å utføre planene vi hadde sett for oss, var det nødvendig med en fase der vi leste oss opp på nødvendig teori. I denne perioden jobbet vi for det meste individuelt.

Utførelsesfasen. Etter at vi hadde opparbeidet oss nok kunnskap, satte vi i gang med arbeidet med selve lysstyringssystemet. Det var planlagt slik at arbeidet var delt i tre, hardware, firmware og applikasjon. Hver enkel person på gruppa hadde ansvaret for sitt «fagfelt», så også her ble det mye individuelt arbeid. Men for å lykkes arbeidet vi også på tvers av fagfeltene.

Avslutningsfasen. Vi satte en deadline for når vi skulle være ferdig med utførelsesfasen, og samtidig påbegynne den avsluttende fasen. Det var mye arbeid som skulle dokumenteres og forklares, og det er det vi ønsker å oppnå med denne rapporten. Mesteparten av avslutningsfasen gikk til å skrive en gjennomført rapport, men også forberedelser til muntlig forsvar av oppgaven.

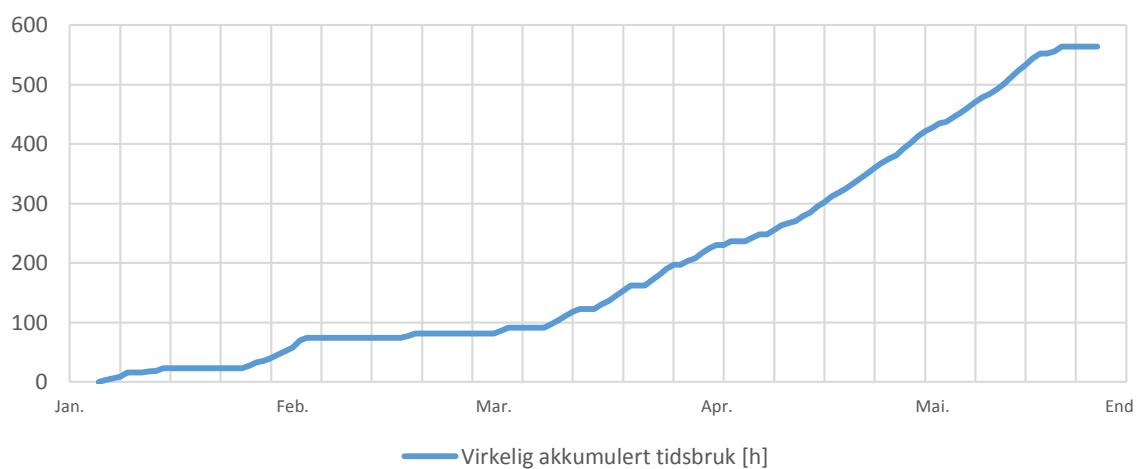
7.3 ARBEIDSFLYT



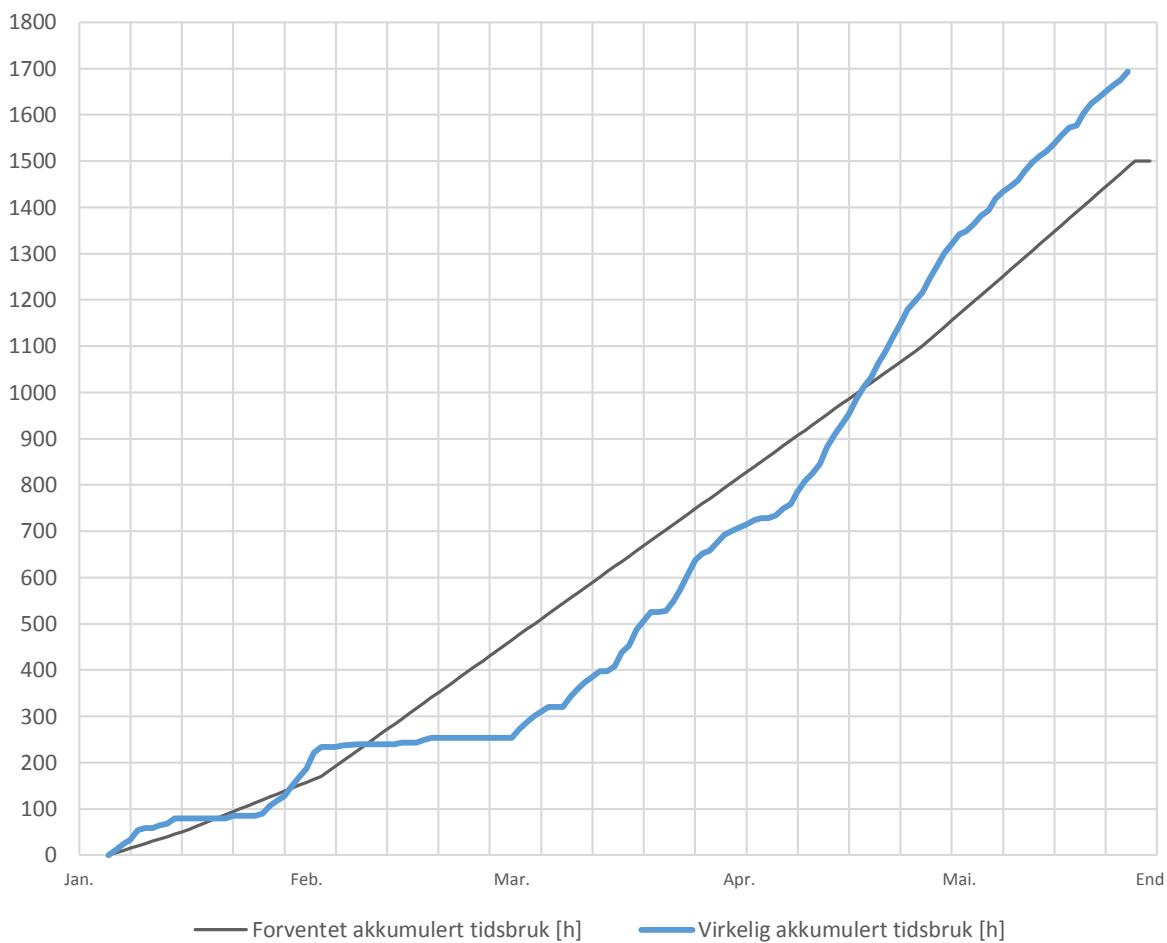
FIGUR 95. ARBEIDSFLYT, AKKUMULERT TIDSBRUK ERLEND



FIGUR 96. ARBEIDSFLYT, AKKUMULERT TIDSBRUK PETTER



FIGUR 97. ARBEIDSFLYT, AKKUMULERT TIDSBRUK MADS



FIGUR 98. ARBEIDSFLYT, TOTALT AKKUMULERT TIDSBRUK VS. FORVENTET TIDSBRUK.

7.4 AVVIK

Når vi gikk i gang med prosjekt i vinter var gruppeantallet fire. I planleggingsfasen ble det derfor tatt høyde for at vi skulle legge oss på et ambisjonsnivå som tilsvarte en passelig mengde arbeid for fire personer. Ved forprosjektets slutt ble vi redusert til tre personer, dermed måtte vi redusere våre ambisjoner. Vi satte prioriteringer på de ulike produktene vi skulle utvikle, som førte til at mye av det vi hadde sett for oss, ikke ble realisert.

Til tross for at vi reduserte ambisjonene for sluttresultatet etter at prosjektgruppa ble en person mindre, kan det tyde på at vi fortsatt la lista litt for høyt. Vi hadde lite oversikt over omfanget av oppgaven før vi faktisk gikk i gang med utviklingen, og det er lett å la seg rive med i idéfasen. Når man legger opp tidsplanen for prosjektet er det vanskelig å forutse hvilke problemer og hindringer man støter på underveis, derfor ble noen av planene ikke gjennomført.

Parallelt med forprosjektet, samt starten av utførelsesfasen, hadde vi et annet fag kalt ingeniørfaglig systemtenkning. Dette var et tidskrevende fag, som tok mye av fokuset vårt bort fra bacheloroppgaven. I februar brukte vi kun omtrentlig 25 timer hver til arbeid med lysstyringssystemet. Til sammenligning brukte vi nesten 200 timer hver i april. Vi har likevel brukt flere antall timer enn det som var kalkulert med for hver person, men progresjonen har ikke vært like jevnt fordelt utover semesteret som vi hadde håpet på. I perioden mellom forprosjektet og påske var det mye teori og nytt stoff som vi måtte sette oss inn i. Dette førte til at veldig lite ble gjennomført i praksis. Rett før utførelsesfasen var over økte progresjonen betraktelig, og vi har en følelse av at produktet vi har skapt kunne vært mer optimalisert dersom vi ikke måtte flytte fokuset vårt vekk fra bacheloroppgaven tidlig i semesteret.

7.5 ØKONOMI

Prosjektregnskap 2015, bacheloroppgave E1534B «LightBlu»

v/ Nordic Semiconductor ASA

Utgiftspost	Budsjett	Reelle utgifter
Nexus 6 «phablet»	3 000	0
Elektroniske komponenter	2 000	1706
Produksjon av kretskort	0	0
Div. lys, koblinger og annet	2 000	1561
Buffer	3 000	
Sum	10 000	3267
Overskudd		6733

TABELL 9. ØKONOMI, REGNSKAP

Navn	Dato	Artikkel	Kommentar	Pris
ERM	02/03/2015	El-artikler	Nexa-kit, elko-boks etc	413
ERM	30/03/2015	El-artikler	Støpsel, vegguttak	177
PH	24/03/2015	El-artikler	Strømbryter, sikring, sikringsholder	255
PH	20/04/2015	El-artikler	Støpsel, vegguttak	118
PH		El-artikler	2 * Nexa-kit	598
Nordic Semiconductor		Komponenter for prototyper		556,06
Nordic Semiconductor		Komponenter for ferdigstilling av hardware		1149,62
Nordic Semiconductor		Google Nexus 6 phablet		0,-
Total				3267

TABELL 10. ØKONOMI, SPESIFISERT REGNSKAP

KAP.8 DISKUSJON OG KONKLUSJON

8.1 DISKUSJON

8.1.1 HARDWARE

En av de største utfordringene i hardware-utviklingen har vært det fysiske grensesnittet. Siden de forskjellige funksjonenhetene skal kunne plasseres i et nexa-chassis har mye ressurser blitt brukt til å finne systemlösninger og komponenter som opptar minst mulig plass. Vi valgte i den tekniske studiefasen å benytte en kapasitiv strømforsyning på grunn av dens enkle design og fysiske størrelse, men ble i utførelsesfasen for alvor klar over dens effektbegrensninger, og utforsket derfor idéen om å benytte en switch-mode-forsyning. En switch-mode-forsyning ville vært kapabel til å levere mer effekt, samtidig som den anses som mer sikker for bruk i forbrukerapplikasjoner. Vi valgte å ikke bruke den type forsyning på grunn av dens fysiske utforming, samtidig som vi ikke var avhengig av økt personsikkerhet ettersom den skulle stå fastmontert i et lukket chassis.

I ettertid ser vi at den kapasitive strømforsyningen også kan oppfattes som uhensiktsmessig stor, da filmkondensatoren tar opp mye plass i chassiset. Det er med andre ord fordeler og ulemper for begge forsyningstypene, og vi kunne med fordel ha utforsket bruk av switch-mode-forsyningen ytterligere.

Under arbeidet med å utbedre den kapasitive strømforsyningen ble vi klar over at det eksisterer en relétype som opererer med vesentlig lavere effektegenskaper enn den vi benytter. Et latching relé er ypperlig for bruksområder hvor lavt strømtrekk fra styrekretsen er en kritisk faktor, men var fysisk større enn et non-latching relé, og ble derfor ikke valgt.

Dimmermodulen har vært den mest komplekse og utfordrende modulen vi har utviklet med tanke på at den inneholder langt flere og mer avanserte komponenter enn Binærmodulen. Det har dessuten også vært et problem å finne komponenter som både fungerer og er små nok.

Komponenten triac er sentral i Dimmermodulen. Ved strømgjennomgang i en triac vil den kunne utvikle varme, og påmonteres derfor i noen tilfeller en kjøleribbe. Grunnet plassmangel i det fysiske grensesnittet er ikke integrert kjøleløsning en prioritert del av det endelige designet. Vi har heller ikke opplevd utilsiktede varmetap fra triacen under test, men har ikke utført spesifikke testrutiner for varmetap som har gått over lengre tidsrom. Det kan være ønskelig å utrede triacens behov for kjøling ytterligere hvis den skal regulere lys over et lengre tidsrom.

8.1.2 FIRMWARE

I starten av prosjektet så vi for oss at vi skulle utvikle flere moduler og produkter enn de vi faktisk endte opp med å utvikle. Dette gjaldt også firmware. Vi hadde blant annet planer om å utvikle en såkalt trådløs fjernbryter som skulle kunne brukes som central-enhet i stedet for smarttelefon. En slik bryter hadde dermed krevet utvikling av en firmwareversjon bygget rundt Nordics S120 SoftDevice. Dette fikk vi som kjent ikke mulighet til å utforske.

Vi endte i stedet opp med å «bare» utvikle én S110-firmware, LDF1A, for Kontrollmodulen. Denne firmwarekoden bygger på en templatekode fra Nordics SDK som inneholder grunnleggende funksjonalitet for Bluetooth Smart-kommunikasjon. Vi hadde nok lært mye mer om oppsett og bruk av Nordics API dersom vi hadde utviklet denne koden fra bunnen av, men dette hadde tatt opp svært mye tid og ville antageligvis ikke medført noe bedre resultat med tanke på målene vi har hatt for prosjektet. Et annet viktig poeng for hvorfor vi valgte å bygge på en eksisterende kode, er at vi gikk inn i dette prosjektet uten forkunnskaper om kodeutvikling for Nordics nRF51 SoC-løsning. Både utviklingsverktøyet og API-en var ukjent for oss til å begynne med.

Kontrollmodulen er som kjent programmert til å bare fungere som et bindeledd. Enkelte kan nok mene at dette er løsing av potensiale. Nordics nRF51422 er tross alt en kraftig mikrokontroller, og kan i utgangspunktet utføre oppgavene til Binær- og Dimmermodulene i tillegg til å håndtere Bluetooth-kommunikasjonen. Grunnen til hvorfor vi ikke har valgt en slik løsning er fordi vi ønsker å ha muligheten til å utvikle en mer avansert firmware til Kontrollmodulen i fremtiden uten å måtte tenke på å inkludere ting som nullkryssing, timing av kontrollpulser osv. Når alt som behøves for å kontrollere lyset er å sende kommandoer over TWI-bussen, frigjøres det kapasitet som potensielt kan brukes til andre formål.

Lesere med falkeblikk har kanskje oppdaget at firmwareversjonene LDFBIN1 og LDFDIM1 er skrevet slik at Binærmodulen og Dimmermodulen får den samme TWI-adressen. Dette er en bevisst handling fra vår side, og er gjort for å simplifisere firmwareversjon LDF1A. Ved å bare måtte forholde seg til én TWI-adresse, kan Kontrollmoduler som er programmert med LDF1A simpelthen videresende kommandoer og data direkte uten å måtte ta hensyn til om disse dataene er ment for en Binær- eller Dimmermodul. På denne måten kan Kontrollmodulen kobles til hvilken som helst av de to modulene uten å måtte konfigureres på nytt.

Ulempen med dette er at alle kommandoer som Kontrollmodul mottar over Bluetooth Smart, sendes også over TWI-bussen uansett hvor vidt disse kommandoene er relevante for tilkoblet Binær- eller Dimmermodul. Dette kan medføre unødvendig aktivitet på TWI-bussen og øker dermed risikoen for låsing.

TWI-koden vi benytter i LDFBIN1 og LDFDIM1 er enkel og oversiktlig, men baseres tungt på bruk av while-løkker. Dette er også tilfellet for kontrollsløyfene som brukes i LDFBIN1 og LDFDIM1 for å oppdage aktivitet på TWI-bussen. Mikrokontrolleren er dermed ofte opptatt med å vente og kan ikke utføre annen kode i mellomtiden. Dette er ikke et så veldig stort problem for vår del siden vi har implementert avbruddsrutiner, men skal LDFBIN1 og LDFDIM1 utvides med ytterligere funksjonalitet bør det gjøres noe med. En potensielt løsning hadde vært å sette opp et avbrudd som trigges av endringer på TWI-bussen og som dermed erstatter de verste kontrollsløyfene. TWI-funksjonene ville likevel fortsatt måtte vært basert på while-løkker.

8.1.3 MOBILAPPLIKASJON

Under planleggingsfasen ble det bestemt at vi skulle utvikle en egen app til systemet vårt, til tross for at vi hadde muligheten til å benytte allerede eksisterende applikasjoner. En grunn til dette var at vi ønsket et komplett system som vi selv hadde produsert. Ingen av oss hadde erfaring med applikasjonsutvikling på forhånd, derfor var vi forberedt på at det kunne bli en utfordring.

Underveis oppstod problemer som følge av manglende forkunnskaper, men vi klarte allikevel å fullføre med et resultat som vi er forholdsvis fornøyd med. Hadde vi vært eksperter på området ville nok resultatet vært noe annerledes, men det er vi ikke, og det er litt av sjarmen. Vi har stått på, funnet kreative løsninger og lært veldig mye. Både når det gjelder det tekniske rundt applikasjonsutvikling, men også prosessrelatert. Å løse problemer som vi har stått fast på gir mestringsfølelse og innblikk i situasjoner som faktisk kan oppstå i arbeidslivet.

En viktig årsak til at arbeidet med applikasjonen hadde ujevn progresjon, var at fungerende firmware ikke var ferdig før etter påske. Dette førte til at testing i praksis var vanskelig å gjennomføre før den tid. Slik at hovedtyngden av utviklingsfasen for applikasjonen varte i kun få uker, noe som var i knappestre laget.

Samtidig som det ikke ble helt slik som vi hadde planlagt, føler vi at applikasjonen vår har et stort potensial. Det skal ikke store endringer til for at den kunne være optimalisert. Slik er det gjerne med utvikling av software. Tidlige versjoner inneholder ofte bugs og dårlige løsninger, derfor må man hele tiden tenke nytt og stadig komme med oppdateringer. Vi ser det slik at applikasjonen vi har laget er en prototype, eller en betaversjon som trenger litt oppussing før den eventuelt skulle vært gitt ut på markedet.

Valg av layout har blitt nedprioritert i forhold til å ende opp med et fungerende resultat. Vi var forberedt på at dette ville skje etter samtaler med oppdragsgiver, som verdsatte funksjonalitet over grafisk design.

Av endringer vi helst skulle sett gjennomført dersom utviklingen hadde vært uten problemer, er selve måten lysstyringen virker på verdt å nevne. For at brukeren skal kunne velge ønsket dimmenivå må personen trykke seg gjennom en loop, fra 0% til 100% med 20%-intervall. Dette syns vi er noe tungvint, og det skulle med fordel virket slik at man kunne stille inn med ett eller to trykk. Utover dette ligger forbedringspotensialet for det meste i det visuelle.

Dersom prosjektet tas opp i fremtiden vil det være interessant å undersøke muligheten for predefinerte lysprofiler. Et eksempel kan være «Lysnivå for film-visning», hvor lysene blir justert til et behagelig nivå for å se film. Samtidig kunne det vært greit med justering av flere lysmoduler samtidig. Under planleggingsfasen kom flere av disse idéene frem, men vi skjønte raskt at det ble for ambisiøst, gitt tidsmengden vi hadde til rådighet.

8.2 KONKLUSJON

På individuelt basis kan vi ikke påstå å ha skapt mye nytt innen utvikling av hardware, firmware og mobilapplikasjoner. Store deler av arbeidet vårt har tross alt vært basert på eksisterende løsninger og velkjent kunnskap, men dette betyr ikke at vi har funnet opp hjulet på nytt. Nyskapningen har vært i sammensetningen av disse elementene, og er tydelig til stede i sluttproduktene våre.

Sluttresultatet vi står igjen med etter utførelsen av prosjektet er to velfungerende lysstyringsenheter og en funksjonell mobilapplikasjon som sammen danner et enkelt og fleksibelt lysstyringssystem.

De siste versjonene av Binær- og Dimmermodulene fungerer nøyaktig som tiltenkt og danner «ryggraden» i systemet. Vi er spesielt fornøyd med Dimmermodulen som på elegant vis demonstrerer samspillet mellom hardware og firmware gjennom digital styring av TRIAC-basert lysdimming.

Utviklingen av Bluetooth Smart-firmware gikk veldig tregt i starten, og vi opplevde flere tekniske utfordringer med Keil og eksempelkodene fra SDK-et. Disse problemene løste seg imidlertid etter hvert, mye takket være utviklerportalen devzone.nordicsemi.com. Gjennombruddet kom like etter påske med den første testbare firmwareversjonen. Den endelige versjonen oppfyller alle forutsetningene våre og virker nøyaktig som tiltenkt.

Arbeidsprosessen har også stort sett gått bra, men ikke alt gikk helt som planlagt. Vi måtte blant annet nedskalere omfanget av prosjektet midtvegs i utførelsen på grunn av diverse forstyrrende elementer. Vi kom dessuten ikke helt i mål med mobilapplikasjonen.

Et resultat av prosessen er at vi sitter igjen med rikelig kunnskap innenfor temaer som vi ikke har undersøkt tidligere. Bluetooth Smart var ukjent for samtlige gruppemedlemmer ved prosjektets start, men etter å ha gjennomgått teori og anvendt denne teorien i praksis, føler vi oss trygge på hva Bluetooth Smart er og måten det virker på. Vi har dessuten tilegnet oss dypere innsikt i hvordan programmering av firmware og mobilapplikasjoner fungerer.

Vi mener derfor at vi har løst problemstillingen på en tilfredsstillende måte, og føler vi sitter igjen med mye nyttig kunnskap og gode erfaringer.

FIGURLISTE

Figur 1. Elektromekanisk relé.....	3
Figur 2. Regulering tyristor og triac.....	5
Figur 3. Triac, skjemasymbol og oppbygning.....	5
Figur 4. Triac, ekvivalent	5
Figur 5. Optokobler med diac Figur 6. Optokobler med transistor	6
Figur 7. Triac, moduser for triggering	7
Figur 8. Triac, strøm- og spenningskarakteristikk	8
Figur 9. Dimming, spenning over last	8
Figur 10. Dimming, analog eksempelkrets	9
Figur 11. Kapasitiv strømforsyning, eksempel	10
Figur 12. ATmega48V.....	14
Figur 13. ATTiny45V	14
Figur 14. Atmel JTAGICE v.3	15
Figur 15. 6-pin og 10-pin SPI-konnektor	15
Figur 16. Illustrasjon av Cypress PSoC4	15
Figur 17. Integrerte funksjonsblokker i nRF51422	16
Figur 18. I2C-buss med n antall enheter	17
Figur 19. Open Drain, transistorkobling	17
Figur 20. TWI, typisk transmisjonsstruktur.....	18
Figur 21. Single master, single slave SPI-buss	19
Figur 22. SPI-buss, kjedekobling	19
Figur 23. SPI-buss, uavhengig slave.....	19
Figur 24. Android studio, mappe- og filstruktur	24
Figur 25. Android, flytskjema	26
Figur 26. BLE, connected-topologi	27
Figur 27. BLE, broadcast-topologi	27
Figur 28. Bluetooth Smart, protokollstakk	27
Figur 29. Bluetooth Smart, pakkestruktur for advertising	28
Figur 30. Bluetooth Smart, eksempelforløp for advertising	28
Figur 31. Bluetooth Smart, GATT-hierarki	29
Figur 32. Kontrollmodul	30
Figur 33. Binærmodul	30
Figur 34. Dimmermodul	30
Figur 35. Switchmodul	30
Figur 36. Simuleringsmodell av kapasitiv strømforsyning	36
Figur 37. Simulering, spenningsdeling C1 og C2, C1 = 1,5uF	36
Figur 38. Simulering, spenningsdeling C1 og C2, C1 = 4,5 uF	37
Figur 39. Simulering, transientspenning uten last	37
Figur 40. Simulering, Rippelspenning uten last	38
Figur 41. Simulering, transientspenning med full last (56 ohm)	39
Figur 42. Simulering, rippelspenning med full last (56 ohm)	39
Figur 43. Testing, testrigg med variabel strømforsyning	40
Figur 44. Testing, Kapasitiv strømforsyning på veroboard	41
Figur 45. Testing, Transientspenning over zenerdiode og utgang regulator, uten last	41

Figur 46. Testing, Stasjonær spenning over zenerdiode og utgang, uten last	42
Figur 47. Testing, transientspenning over zenerdiode og utgang regulator, full last (56 ohm).....	43
Figur 48. Testing, transientspenning over zenerdiode og utgang regulator. Last = 82 ohm	44
Figur 49. Testing, rippelspanning over zenerdiode og utgang regulator. Last = 82 ohm	44
Figur 50. Endelig design, Binær- og Dimmermodul (top).....	46
Figur 51. Endelig design, Binær og Dimmermodul (bottom).....	46
Figur 52. Firmware, tilstandsdiagram for LDFBIN1_SLAVE	50
Figur 53. Firmware, hovedkode LDFBIN1_SLAVE_v.1.1.c	51
Figur 54. Firmware, sammenheng mellom hovedkode og avbruddsrutiner for LDFDIM1	52
Figur 55. Firmware, tilstandsdiagram for hovedkoden i LDFDIM1	53
Figur 56. Firmware, tilstandsdiagram for avbruddsrutine PCINT1_vect	54
Figur 57. Firmware, tilstandsdiagram for avbruddsrutine TIMER_COMPA_vect	54
Figur 58. Firmware, main loop for LDFDIM1_SLAVE (utdrag fra LDFDIM1_SLAVE_v1.3.c)	55
Figur 59. Firmware, avbruddsrutine PCINT1_vect (utdrag fra LDFDIM1_SLAVE_v1.3.c)	56
Figur 60. Firmware, avbruddsrutine TIMERO_COMPA_vect (utdrag fra LDFDIM1_SLAVE_v1.3.c)	56
Figur 61. Firmware, tilstandsdiagram for LDF1As hovedsløye	58
Figur 62. Firmware, sammenheng mellom main loop og advertising, LDF1A.....	58
Figur 63. Firmware, tilstandsdiagram for "Light Control" events, LDF1A	58
Figur 64. Testing, testoppsett for TWI-kommunikasjon mellom Switch- og Binærmodul.....	59
Figur 65. Testing, oppsett for firmwaretesting av tidlig Binærprototyp	59
Figur 66. Testing, testoppsett av nært ferdigstilt Binærmodul og Bluetooth Smart Kontrollmodul	60
Figur 67. Opprinnelig plan, scanneaktivitet og aktivitet for styring av lys	61
Figur 68. Applikasjon, scanneaktivitet og aktivitet for styring av lys	62
Figur 69. Applikasjon, kode som viser hendelsesforløpet ved å trykke på en characteristic	64
Figur 70. Applikasjon, if-setning som sørger for at riktig verdi blir skrevet til characteristic	65
Figur 71. Sluttnormalisert, Binær funksjonhet	69
Figur 72. Sluttnormalisert, Dimmer funksjonhet	69
Figur 73. Sluttnormalisert, Dimmermodul plassert i NEXA-chassiet	70
Figur 74. Sluttmålinger, transientspenning over zenerdiode og utgang regulator. Kontrollmodul frakoblet	72
Figur 75. Sluttmålinger, transientspenning over zenerdiode og utgang regulator. Kontrollmodul tilkoblet	72
Figur 76. Sluttmålinger, transient og stasjonært strømforbruk fra regulator. Kontrollmodul frakoblet	74
Figur 77. Sluttmålinger, transient og stasjonært strømforbruk fra regulator. Kontrollmodul tilkoblet	74
Figur 78. Sluttmålinger, stasjonært strømforbruk fra regulator. Advertisement mode	76
Figur 79. Sluttmålinger, stasjonært strømforbruk fra regulator. Connected mode	76
Figur 80. Sluttmålinger, transient strømforbruk fra regulator. Connected mode. Relé aktiveres	77
Figur 81. Sluttmålinger, eksempel på TWI-kommunikasjon	77
Figur 82. Sluttmålinger, Nullgjennomgangsdeteksjon og styresignal ved lysnivå 100 %	80
Figur 83. Sluttmålinger, Spenningsbølge over last ved lysnivå 100%	80
Figur 84. Sluttmålinger, Nullgjennomgangsdeteksjon og styresignal ved lysnivå 80 %	81
Figur 85. Sluttmålinger, Spenningsbølge over last ved lysnivå 80 %	81
Figur 86. Sluttmålinger, Nullgjennomgangsdeteksjon og styresignal ved lysnivå 60 %	82
Figur 87. Sluttmålinger, Spenningsbølge over last ved lysnivå 60 %	82
Figur 88. Sluttmålinger Nullgjennomgangsdeteksjon og styresignal ved lysnivå 40 %	83
Figur 89. Sluttmålinger, Spenningsbølge over last ved lysnivå 40 %	83
Figur 90. Sluttmålinger, Nullgjennomgangsdeteksjon og styresignal ved lysnivå 20 %	84
Figur 91. Sluttmålinger, Spenningsbølge over last ved lysnivå 20 %	84
Figur 92. Sluttmålinger, Nullgjennomgangsdeteksjon og styresignal ved lysnivå 0 %	85

Figur 93. Sluttmålinger, Spenningsbølge over last ved lysnivå 0 %	85
Figur 94. Sluttmålinger, Strømforbruk fra regulator. connected state. lysinnstilling 0x10	86
Figur 95. Arbeidsflyt, Akkumulert tidsbruk Erlend.....	89
Figur 96. Arbeidsflyt, Akkumulert tidsbruk Petter	89
Figur 97. Arbeidsflyt, akkumulert tidsbruk Mads	89
Figur 98. Arbeidsflyt, totalt akkumulert tidsbruk vs. forventet tidsbruk.	90

TABELLISTE

Tabell 1. Spesifikasjoner for ATtiny45V og ATmega48V	14
Tabell 2. Datatyper i Java	20
Tabell 3. Bluetooth Smart, oversikt over enhetsroller.....	28
Tabell 4. Effektberegning for Binærmoul	33
Tabell 5. Effektberegning for Dimmermodul	33
Tabell 6. Firmware, versjonsoversikt.....	47
Tabell 7. Firmware, kommandooversikt.....	47
Tabell 8, Sluttmålinger, oversikt over sluttmalinger for dimmermodul	79
Tabell 9. Økonomi, regnskap	92
Tabell 10. Økonomi, spesifisert regnskap.....	92

KILDELISTE

- [1] R. Alvestad, «Nasjonalbiblioteket,» [Internett]. Available: <http://www.nb.no/nbsok/nb/96d9f1b1620e05db7617dcc3238d0b3b.nbdigital?lang=no#19>.
- [2] «National Instruments,» [Internett]. Available: <http://www.ni.com/white-paper/3960/en/#toc2>.
- [3] «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/magnetisme>.
- [4] «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/halvledere>.
- [5] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/data/semicond/semiconductor/semiconductor-materials-types-list.php>.
- [6] «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/halvledere/komponenttyper>.
- [7] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/data/semicond/triac/what-is-a-triac-basics-tutorial.php>.
- [8] «Electronics Tutorials,» [Internett]. Available: <http://www.electronics-tutorials.ws/power/triac.html>.
- [9] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/data/semicond/diac/diac.php>.
- [10] «Future Electronics,» [Internett]. Available: <https://www.futureelectronics.com/en/optoelectronics/optocouplers.aspx>.
- [11] «Electronics Tutorials,» [Internett]. Available: <http://www.electronics-tutorials.ws/blog/optocoupler.html>.
- [12] «How Stuff Works,» [Internett]. Available: <http://home.howstuffworks.com/dimmer-switch1.htm>.
- [13] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/power-management/switching-mode-power-supply/basics-tutorial.php>.
- [14] «Capacitor Guide,» [Internett]. Available: <http://www.capacitorguide.com/film-capacitor/>.
- [15] «Capacitor Guide,» [Internett]. Available: <http://www.capacitorguide.com/applications/capacitors-in-series/>.
- [16] «Electronics Tutorials,» [Internett]. Available: http://www.electronics-tutorials.ws/diode/diode_5.html.
- [17] «Electronics Tutorials,» [Internett]. Available: http://www.electronics-tutorials.ws/diode/diode_6.html.
- [18] «Bright Hub Engineering,» [Internett]. Available: http://www.brighthubengineering.com/consumer-appliances-electronics/96645-efficiency-of-ac-rectifiers/#imgn_2.
- [19] «Capacitor Guide,» [Internett]. Available: <http://www.capacitorguide.com/electrolytic-capacitor/>.

- [20] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/data/capacitor/electrolytic-capacitor.php>.
- [21] «Electronics Tutorials,» [Internett]. Available: http://www.electronics-tutorials.ws/diode/diode_7.html.
- [22] «Evil Mad Scientist,» [Internett]. Available: <http://www.evilmadscientist.com/2012/basics-introduction-to-zener-diodes/>.
- [23] «Dimension Engineering,» [Internett]. Available: <https://www.dimensionengineering.com/info/switching-regulators>.
- [24] «Texas Instruments,» [Internett]. Available: <http://www.ti.com/lit/an/snva558/snva558.pdf>.
- [25] «Linear Technology,» [Internett]. Available: <http://cds.linear.com/docs/en/application-note/AN140fa.pdf>.
- [26] «Java Tpoint,» [Internett]. Available: <http://www.javatpoint.com/history-of-java>.
- [27] «Oracle,» [Internett]. Available: <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html>.
- [28] «Javabok,» [Internett]. Available: <http://javabok.no/pdf/kap5.pdf>.
- [29] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Android_%28operating_system%29.
- [30] «Developer Android,» [Internett]. Available: <http://developer.android.com/about/index.html>.
- [31] «Source Android,» [Internett]. Available: <https://source.android.com/>.
- [32] «Android Central,» [Internett]. Available: <http://www.androidcentral.com/android-z-what-kernel>.
- [33] «Developer Android,» [Internett]. Available: <https://developer.android.com/sdk/index.html>.
- [34] «Webopedia,» [Internett]. Available: http://www.webopedia.com/TERM/A/Android_SDK.html.
- [35] «Developer Android,» [Internett]. Available: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.
- [36] «Developer Android,» [Internett]. Available: <http://developer.android.com/guide/topics/resources/providing-resources.html>.
- [37] «Developer Android,» [Internett]. Available: <http://developer.android.com/guide/components/activities.html>.
- [38] «Google Play,» [Internett]. Available: https://play.google.com/intl/ALL_us/about/overview/index.html.
- [39] «Andromo,» [Internett]. Available: <http://support.andromo.com/kb/distributing/how-to-put-your-app-in-google-play>.
- [40] «Nordic Semiconductor,» [Internett]. Available: <http://www.nordicsemi.com/eng/Products/ANT/nRF51422/%28language%29/eng-GB>.

- [41] «Farnell,» [Internett]. Available: <http://www.farnell.com/datasheets/1849874.pdf>.
- [42] «EngineersGarage,» [Internett]. Available: <http://www.engineersgarage.com/embedded/avr-microcontroller-projects/atmega32-twi-two-wire-interface>.
- [43] «Atmel,» [Internett]. Available: <http://www.atmel.com/images/doc2545.pdf>.
- [44] «Android Developer,» [Internett]. Available: <http://developer.android.com/index.html>.
- [45] «Nordic Semiconductor GitHub,» [Internett]. Available: <https://github.com/NordicSemiconductor>.
- [46] «Radio-Electronics,» [Internett]. Available: <http://www.radio-electronics.com/info/data/semicond/triac/structure-fabrication.php>.
- [47] «Prentice Hall Companion Website,» [Internett]. Available: http://wps.prenhall.com/chet_paynter_introduct_6/6/1665/426386.cw/index.html.
- [48] «Circuits Today,» [Internett]. Available: <http://www.circuitstoday.com/diac-applications>.
- [49] «Texas Instruments,» [Internett]. Available: <http://www.ti.com/lit/an/snva559/snva559.pdf>.
- [50] «Osram,» [Internett]. Available: http://www.osram.no/osram_no/nyheter-og-kunnskap/lysstyringssystemer/produktkunnskap/dali-professional/index.jsp.
- [51] «Arduino Forum,» [Internett]. Available: <http://forum.arduino.cc/index.php?topic=125377.0>.
- [52] «Electronics Tutorials,» [Internett]. Available: http://www.electronics-tutorials.ws/transistor/tran_6.html.
- [53] «Electroschematics,» [Internett]. Available: <http://www.electroschematics.com/5678/capacitor-power-supply/>.
- [54] «Electronics Tutorials,» [Internett]. Available: <http://www.electronics-tutorials.ws/resistor/varistor.html>.
- [55] «Nexa,» [Internett]. Available: <http://www.nexa.se/EYCR-2300-Extra.htm>.
- [56] «Nasjonalbiblioteket,» [Internett]. Available: <http://www.nb.no/nbsok/nb/766eca7b2a2ced529c6687761b4599c3.nbdigital?lang=no#0>.
- [57] «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/nettsystem>.
- [58] «Elektroteknikk,» [Internett]. Available: <http://w3.elektrofag.info/elektroteknikk/fordelingssystemer>.
- [59] «Ken's Web Site,» [Internett]. Available: http://www.kennethkuhn.com/students/ee351/power_supplies_filter_capacitor.pdf.
- [60] «Learning About Electronics,» [Internett]. Available: <http://www.learningaboutelectronics.com/Articles/Capacitive-voltage-divider.php>.
- [61] «Hamill and Hamill,» [Internett]. Available: <http://www.hamill.co.uk/pdfs/acfvsdsp.pdf>.

- [62] «Cocoon Tech Forum,» [Internett]. Available: <http://cocoontech.com/forums/topic/25820-need-some-adviceupb-lights-pulsing/page-2>.
- [63] «Store Norske Leksikon,» [Internett]. Available: <https://snl.no/Skilletransformator>.
- [64] «Educypedia,» [Internett]. Available: <http://educypedia.karadimov.info/library/6785.pdf>.
- [65] «Flickriver,» [Internett]. Available: <http://www.flickriver.com/photos/5volt/tags/control/>.
- [66] «Electronics Stackexchange,» [Internett]. Available:
<http://electronics.stackexchange.com/questions/48462/optocoupler-with-phototransistor-base-lead>.
- [67] «Electronic Products,» [Internett]. Available:
http://www.electronicproducts.com/Digital_ICs/Communications_Interface/The_venerable_I2C-bus.aspx.
- [68] «Nordic Semiconductor,» [Internett]. Available:
http://www.nordicsemi.com/eng/content/download/13359/215006/file/nRF51422_PS%20v3.1.pdf.
- [69] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Open_collector#MOSFET.
- [70] «Wikipedia,» [Internett]. Available: <http://en.wikipedia.org/wiki/I%C2%B2C>.
- [71] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus.
- [72] «Sparkfun,» [Internett]. Available: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>.
- [73] «Wikipedia,» [Internett]. Available: <http://en.wikipedia.org/wiki/Atmel AVR>.
- [74] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Reduced_instruction_set_computing.
- [75] «Atmel,» [Internett]. Available: http://www.atmel.com/Images/Atmel-2586-AVR-8-bit-Microcontroller-ATTiny25-ATTiny45-ATTiny85_Datasheet.pdf.
- [76] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/ARM_architecture.
- [77] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Acorn_Computers.
- [78] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/ARM_Cortex-M.
- [79] «Nordic Semiconductor,» [Internett]. Available:
[http://www.nordicsemi.com/eng/Products/ANT/nRF51422/\(language\)/eng-GB](http://www.nordicsemi.com/eng/Products/ANT/nRF51422/(language)/eng-GB).
- [80] «Nordic Semiconductor,» [Internett]. Available:
http://www.nordicsemi.com/eng/content/download/13359/215006/file/nRF51422_PS%20v3.1.pdf.
- [81] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Intellectual_property.
- [82] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/ARM_Holdings.
- [83] «Nordic Semiconductor,» [Internett]. Available:
http://www.nordicsemi.com/eng/nordic/download_resource/20371/17/60152074.

- [84] «Wikipedia,» [Internett]. Available: http://en.wikipedia.org/wiki/Bluetooth_low_energy.
- [85] «Adafruit,» [Internett]. Available: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction>.
- [86] «Blogspot,» [Internett]. Available: <http://1.bp.blogspot.com/-b3-oj9-wFkU/UajKsooqsUI/AAAAAAAAD8/w9yUPQHuvnk/s1600/Primitive-Data-Types.png>.
- [87] «Høgskolen i Oslo og Akershus,» [Internett]. Available: <http://www.cs.hioa.no/~evav/uvstoff/intro/kap4.html>.

VEDLEGG

Vedlegg 1: Oppgaven

Vedlegg 2: Utlegg og skjema

Vedlegg 3: Komponentliste

Vedlegg 4: Digitalt Arkiv / Firmware

Vedlegg 5: Gantt-diagram

Vedlegg 6: Plakat

VEDLEGG 1
PROSJEKTOPPGAVEN



HØGSKOLEN I SØR-TRØNDELAG
Program for elektro- og datateknikk
7004 Trondheim

Oppgave nr: 34B

Reservert: Nei

Passer best for studieretning: Elektronikk, Instrumentering

Spesielle kommentarer:

Nordic Semiconductor's nRF51822 chip for Bluetooth® low energy communication is a powerful, highly flexible multi-protocol SoC ideally suited for Bluetooth® low energy and proprietary 2.4GHz ultra low-power wireless applications. The nRF51822 is built around a 32-bit ARM® Cortex™ M0 CPU with 256kB flash + 32kB RAM. The embedded 2.4GHz transceiver supports Bluetooth low energy as well as proprietary 2.4GHz operation, where the proprietary 2.4GHz mode is on air compatible with the nRF24L series products from Nordic Semiconductor.

The goal is to use our nRF51 Series chips to make a Bluetooth® low energy appcessory that connects either to an iPhone or Android Bluetooth® Low Energy enabled smartphone.

Light dimmer

The goal of this project is to use a Bluetooth® Low Energy enabled smartphone (and/or PC) to control the lights in your apartment.

On the device side, you should design and write firmware for an easily installable dimmer unit , e.g between socket and power plug or, if small enough, directly on cable.

On the host side, you should be capable of connecting to several units, controlling lights in several places from the same device.

Optional improvements

SUPPRESS noise from dimmer, decrease size of dimmer unit, make the unit dimmable from analog user input (traditional dimmer switch)

You can also contact us with your own appcessory idea. An appcessory is an app in a mobile device that activates something in the physical world.

Christian Wilhelmsen

R&D Applications



[Nordic Semiconductor](#)

Otto Nielsens veg 12, N-7052 Trondheim, Norway

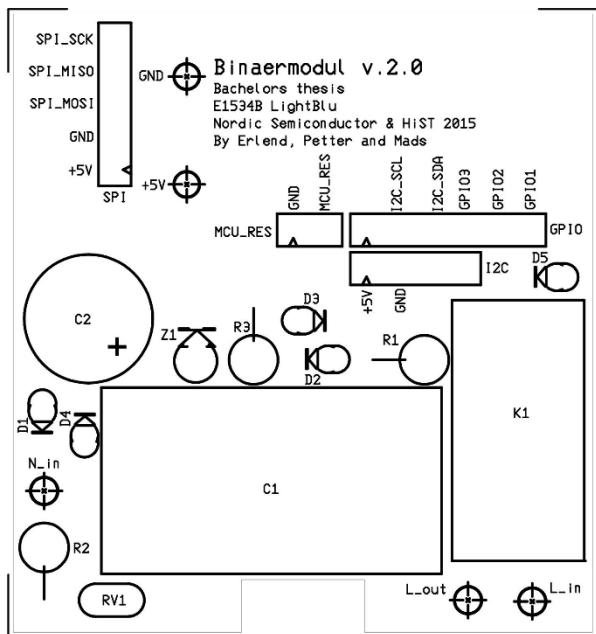
www.nordicsemi.com

VEDLEGG 2

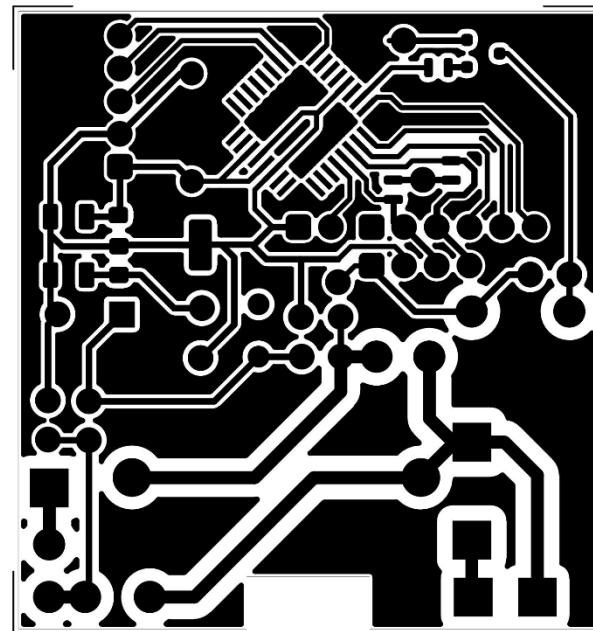
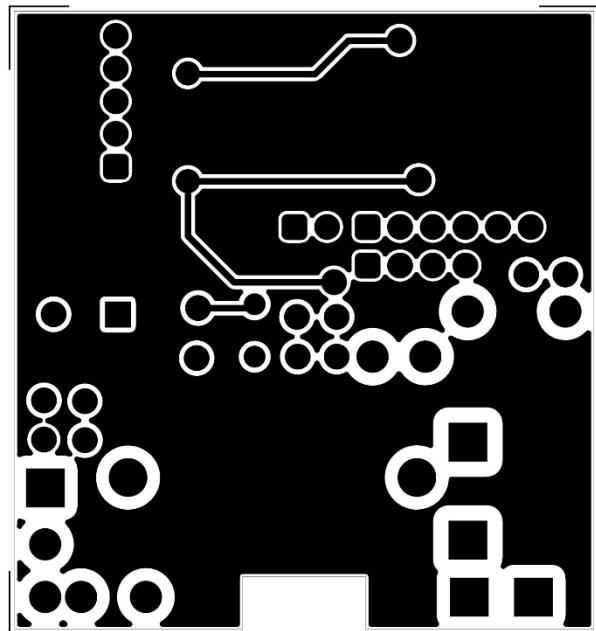
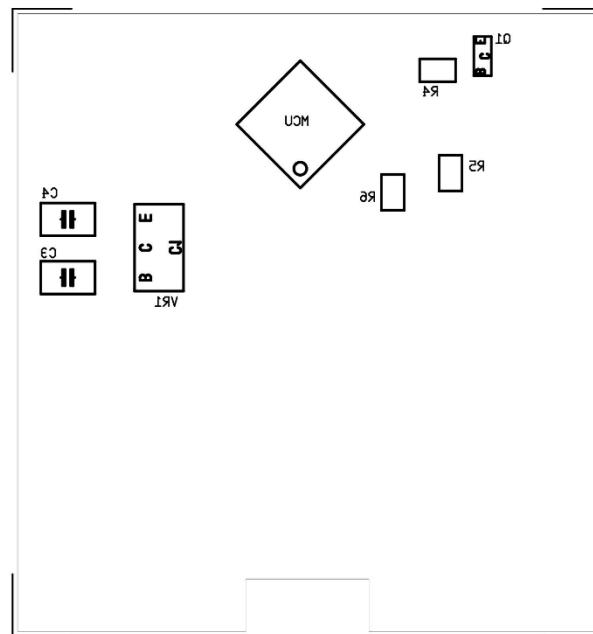
UTLEGG OG SKJEMA

VEDLEGG 2.1: UTLEGG FOR BINÆRMODUL (v.2.0)

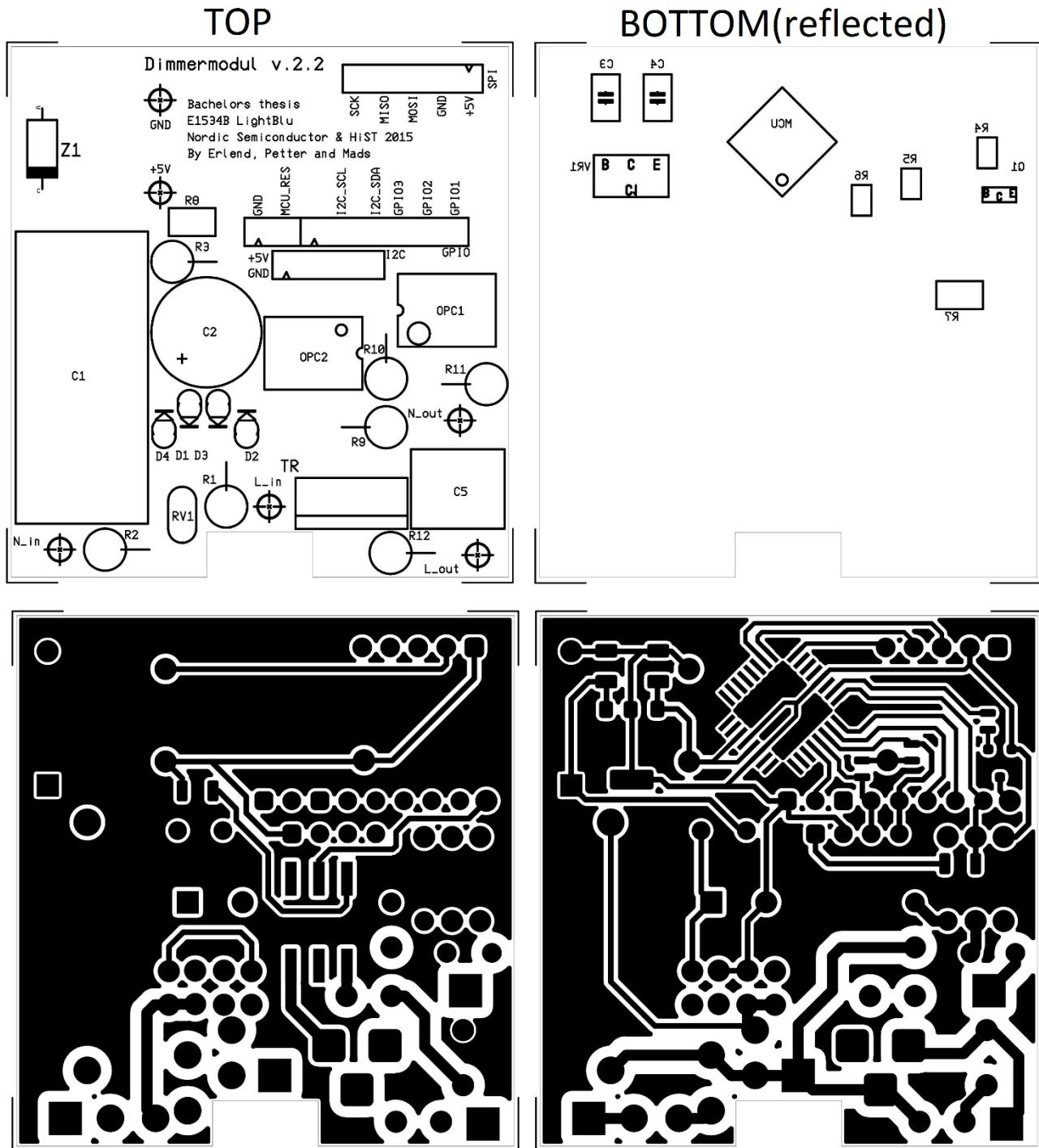
TOP



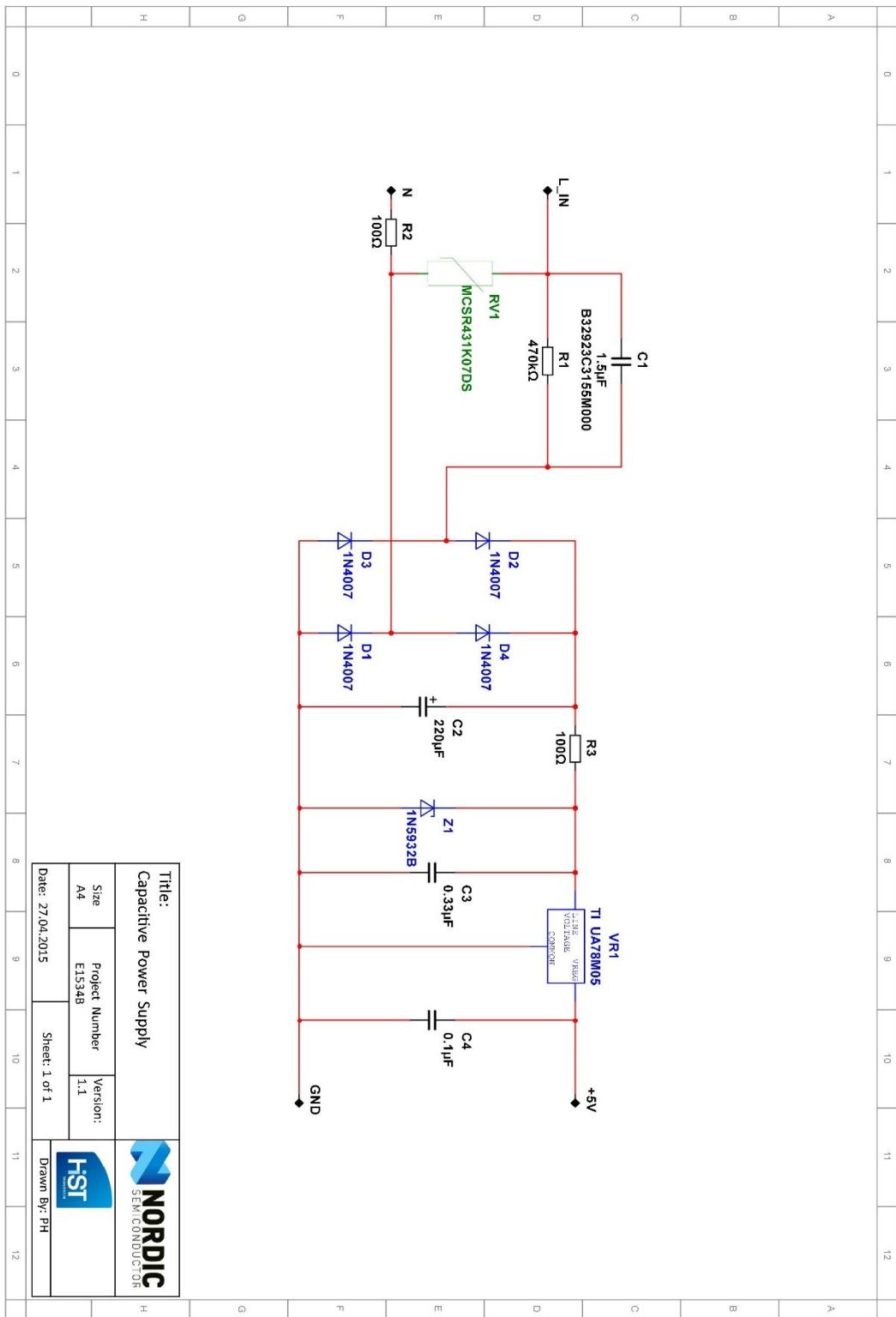
BOTTOM (reflected)



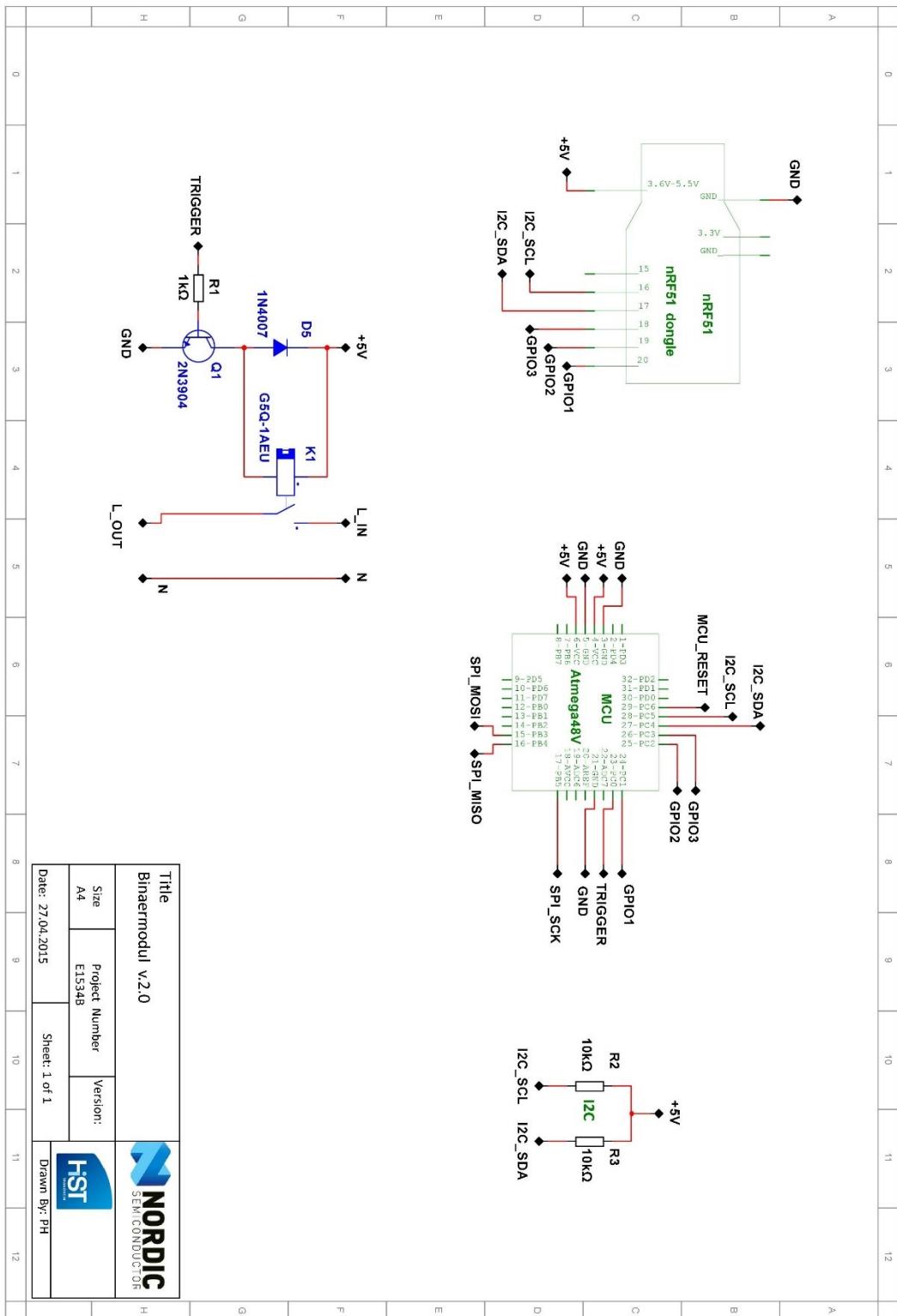
VEDLEGG 2.2: UTLEGG FOR DIMMERMODUL (V.2.2)



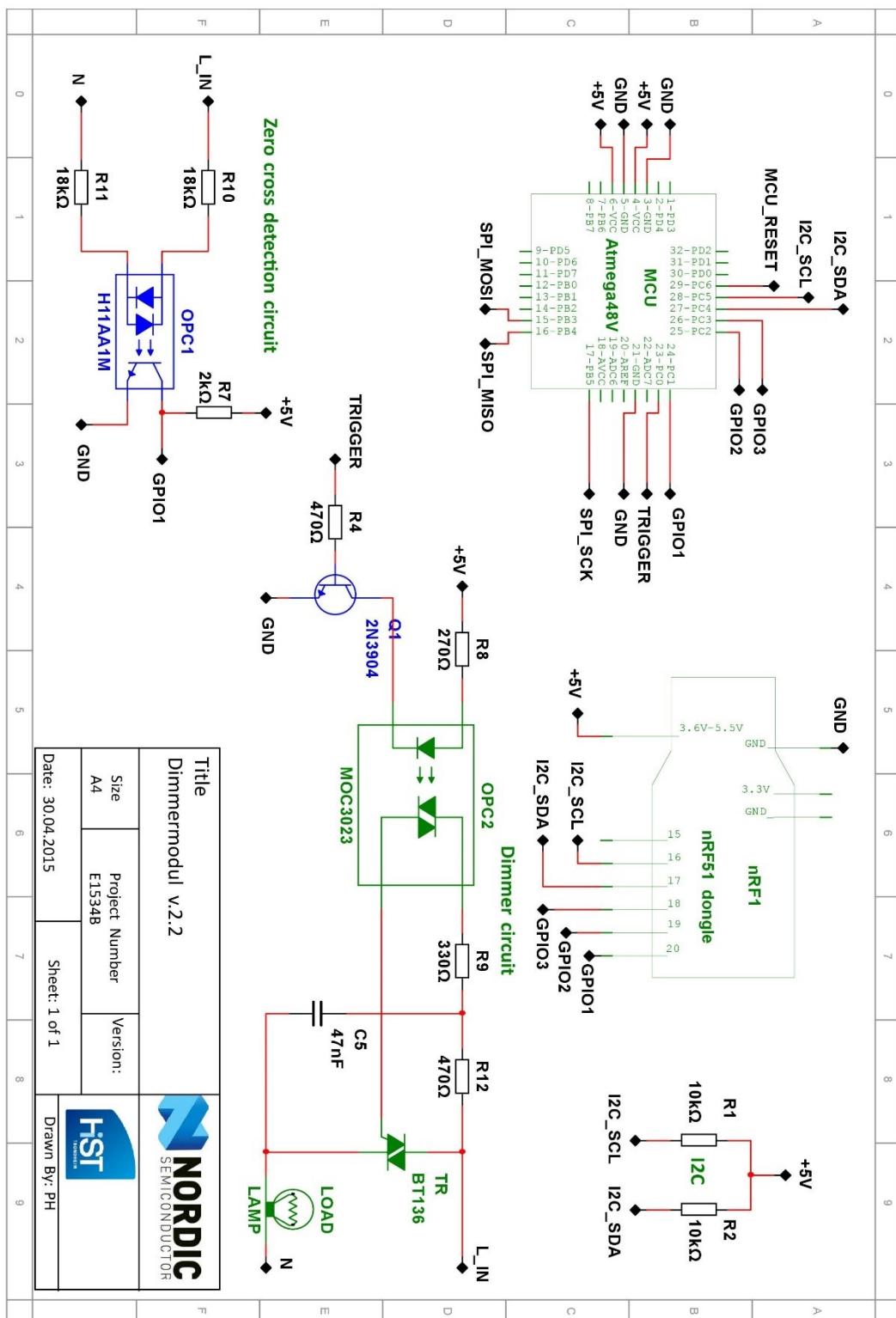
VEDLEGG 2.3: SKJEMA FOR KAPASITIV STRØMFORSYNING



VEDLEGG 2.4: SKJEMA FOR BINÆRMODUL



VEDLEGG 2.5: SKJEMA FOR DIMMERMODUL



VEDLEGG 3

KOMPONENTLISTER

VEDLEGG 3.1: KOMPONENTLISTE FOR BINÆRMODUL (v.2.0)

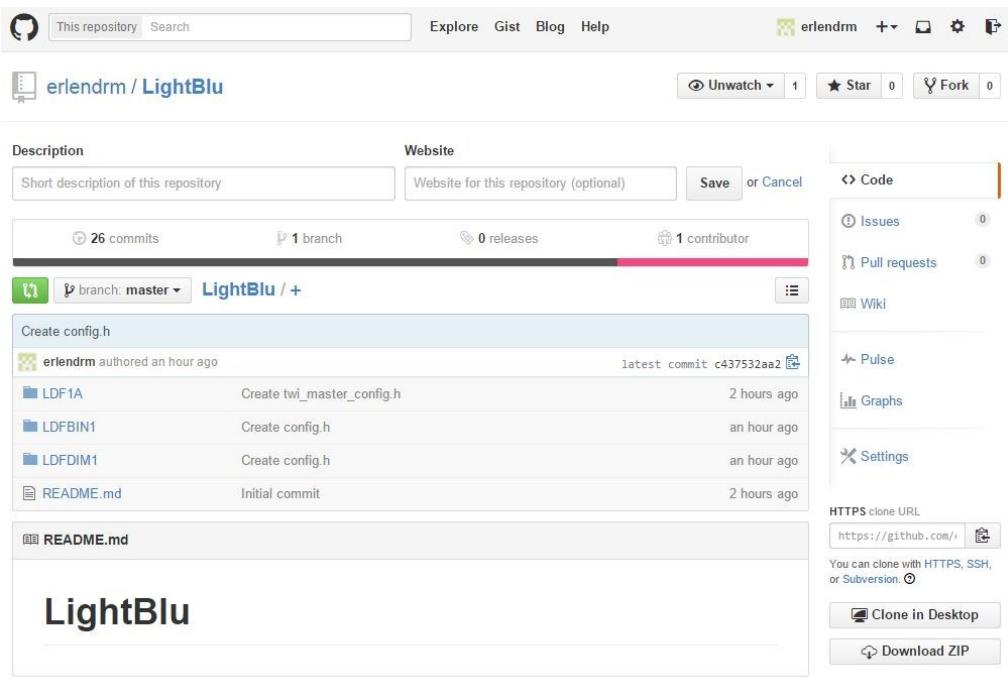
Komponentnavn	RefDes	Antall	Enhetspris (NOK)	Tot-pris	Kommentar
EPCOS B32923C3155M000	C1	1	11,95	11,95	Kondensator for spenningsdeler
Panasonic ECA-1HHG221	C2	1	3,95	3,95	Glattekondensator
AVX 12065C334KAT2A	C3	1	1,85	1,85	Glatting regulator
Multicomp MC1206B104K160CT	C4	1	0,305	0,305	Glatting regulator
Multicomp 1N4007	D1,D2,D3, D4, D5	5	0,227	1,135	Likeretterdiode, flyback diode for relé
Omron G5Q-1AEU 5DC	K1	1	22,90	22,9	Relè
Atmel ATMEGA48PV-10AU	MCU	1	42,70		Lokal mikrokontroller. Donert fra Elektra(Pris fra Elfa)
On Semiconductor MMBT3904LT1G	Q1	1	0,717	0,717	Transistor for styring av relé
Multicomp MCSR431K07DS	RV1	1	1,35	1,35	Varistor
Multicomp MCF 1W 470K	R1	1	0,1793	0,1793	Utladningsmotstand spenningsdeler
Multicomp MOR01SJ0101A10	R2,R3	2	1,95	3,9	Strømbegrensere
VISHAY DRALORIC CRCW0603470RFKEAHP	R4	1	0,501	0,501	Strømbegrenser transistor/MCU
PANASONIC ERJ3EKF1002V	R5, R6	2	0,455	0,91	I2C-motstand
NXP BT136-600	TR	1	4,15	4,15	Triac
Texas Instruments UA78M05CDCYG3	VR1	1	3,25	3,25	Spenningsregulator
Multicomp 1N5932B	Z1	1	0,87	0,87	Zenerdiode for konstant innspenning til regulator
SIL-Headers	-	6			Donert fra Elektra
Nexamodul		1	150	150	
nRF51 Dongle		1	397		Donert fra Nordic Semi(Pris fra Elfa)
Total				208	Ikke kalkulert for MCU, Headers og dongle

VEDLEGG 3.2: KOMPONENTLISTE DIMMERMODUL (V.2.2)

Komponentnavn	RefDes	Antall	Enhetspris (NOK)	Tot-pris	Kommentar
EPCOS B32923C3155M000	C1	1	11,95	11,95	Kondensator for spenningsdeler
Panasonic ECA-1HHG221	C2	1	3,95	3,95	Glattekondensator
AVX 12065C334KAT2A	C3	1	1,85	1,85	Glatting regulator
Multicomp MC1206B104K160CT	C4	1	0,305	0,305	Glatting regulator
WIMA MKP2J024701M00KSSD	C5	1	4,20	4,2	Snubber-kondensator
Multicomp 1N4007	D1,D2,D3, D4	4	0,227	0,908	Likeretterdiode
Atmel ATMEGA48PV-10AU	MCU	1	42,70		Lokal mikrokontroller. Donert fra Elektra (Pris fra Elfa)
Fairchild Semiconductor H11AA1M	OPC1	1	7,85	7,85	Optocoupler Zero Cross detection
Fairchild Semiconductor MOC3023SR2M	OPC2	1	2,30	2,3	Optocoupler Triac driver
On Semiconductor MMBT3904LT1G	Q1	1	0,717	0,717	Transistor for styring av optocoupler
Multicomp MCSR431K07DS	RV1	1	1,35	1,35	Varistor
Multicomp MCF 1W 470K	R1	1	0,1793	0,1793	Utladningsmotstand spenningsdeler
Multicomp MOR01SJ0101A10	R2,R3	2	1,95	3,9	Strømbegrenser
VISHAY DRALORIC CRCW0603470RFKEAHP	R4	1	0,501	0,501	Strømbegrenser transistor/MCU
PANASONIC ERJ3EKF1002V	R5, R6	2	0,455	0,91	I2C-motstand
BOURNES CR1206-FX-2001ELF	R7	1	0,582	0,582	Strømbegrenser Zero Cross input MCU
PANASONIC ERJ8ENF2700V	R8	1	0,66	0,66	Strømbegrenser Optocoupler Triac driver
WELWYN MFP1-330R	R9	1	0,793	0,793	Strømbegrenser Triac trigger
Panasonic ERG1SJ183	R10, R11	2	2,19	4,38	Strømbegrenser Zero Cross detection
PANASONIC ERG1SJ471	R12	1	2,19	2,19	Strømbegrenser snubber
NXP BT136-600	TR	1	4,15	4,15	Triac
Texas Instruments UA78M05CDCYG3	VR1	1	3,25	3,25	Spenningsregulator
Multicomp 1N5932B	Z1	1	0,87	0,87	Zenerdiode for konstant innspenning til regulator
Headers	-	6			Donert fra Elektra
Nexa-modul		1	150	150	
nRF51 Dongle		1	397		Donert fra Nordic Semi(conductor) Pris fra Elfa)
Total				208	Ikke kalkulert for MCU, headers og dongle

VEDLEGG 4

DIGITALT ARKIV / FIRMWARE

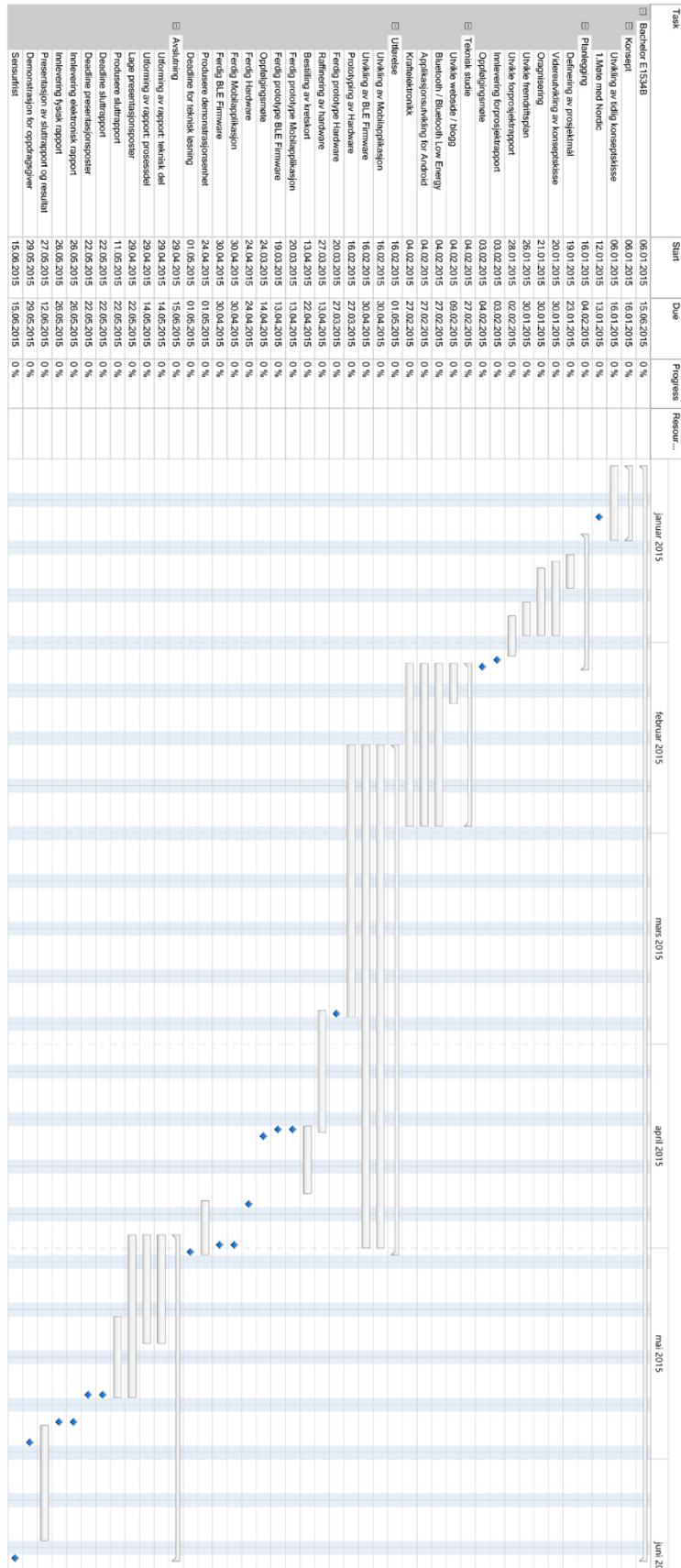


Det digitale arkivet inneholder samtlige firmwareversjoner og er tilgjengelig via nettjenesten GitHub fra følgende link:

https://github.com/erlendrm/LightBlu

VEDLEGG 5

GANTT-DIAGRAM



VEDLEGG 6

PLAKAT

LightBlu

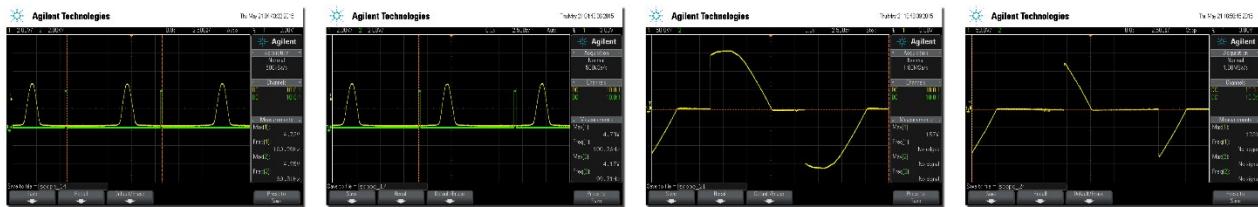
Bachelorprosjekt E1534B

Erlend R. Myklebust

Petter Haugen

Mads E. Stephansen

- Trådløs styring av lys gjennom Bluetooth Smart
- Sømløs integrering av firmware og hardware
- Laget for nRF51 Dongle
- Appcessory til Android (4.3++)
- Nordic S110 SoftDevice peripheral
- Modulær hardwareløsning



“Binærplugg” : Lys av og på

Lyskommando

0x07 0x05

“Dimmerplugg” : Lysnivå 0%, 20%, 40%, 60%, 80%, 100%

Lyskommando 0x0A 0x0C 0x0E 0x10 0x12 0x14



**HØGSKOLEN
ISØR-TRØNDELAG**

