

Even Johan Christiansen, Geir André Engesbak, Rasmus  
Espset, Ørjan Storås

Bacheloroppgave

BLE-kube

BLE-Cube

Trondheim, 09. juni , 2016

NTNU  
Norges teknisk-naturvitenskapelige  
universitet  
Fakultet for teknologi  
Institutt for elektrofag og fornybar energi



Institutt for elektrofag og fornybar energi

---

## Bacheloroppgave

---

<b>Oppgavens tittel:</b> BLE-kube– en leke for det fysiske nett  <b>Project tittel:</b> <i>BLE Cube – A Physical Web Toy</i>	<b>Gitt dato:</b> 11.12.2015
	<b>Innlevingsdato:</b> 09.06.2016
	<b>Antall sider/bilag</b> 84/16
<b>Gruppedeltakere:</b> Christiansen, Even Johan (EJC)      TLF: 90760377 <a href="mailto:evenjch@gmail.com">evenjch@gmail.com</a> Espset, Rasmus (RE)      TLF: 93053524 <a href="mailto:metrolf@hotmail.com">metrolf@hotmail.com</a> Engesbak, Geir Andre (GAE)      TLF:90069470 <a href="mailto:geir.andre.engesbak@gmail.com">geir.andre.engesbak@gmail.com</a> Storås, Ørjan (ØS)      TLF:92080266 <a href="mailto:orjansto@student.hist.no">orjansto@student.hist.no</a>	<b>Veileder:</b> Herman Ranes <a href="mailto:Herman.Ranes@ntnu.no">Herman.Ranes@ntnu.no</a> TLF: 73559606
<b>Studieretning:</b> Elektronikk & Industriell instrumentering	<b>Prosjektnummer:</b> E1627 01
<b>Oppdragsgiver:</b> Nordic Semiconductor ASA	<b>Kontaktperson hos oppdragsgiver:</b> Joakim Tønnessen <a href="mailto:joakim.tonnessen@nordicsemi.no">joakim.tonnessen@nordicsemi.no</a> TLF: 41140445

Fritt tilgjengelig ☒Tilgjengelig etter avtale med oppdragsgiver ☒Rapporten frigitt etter

# 1 Forord

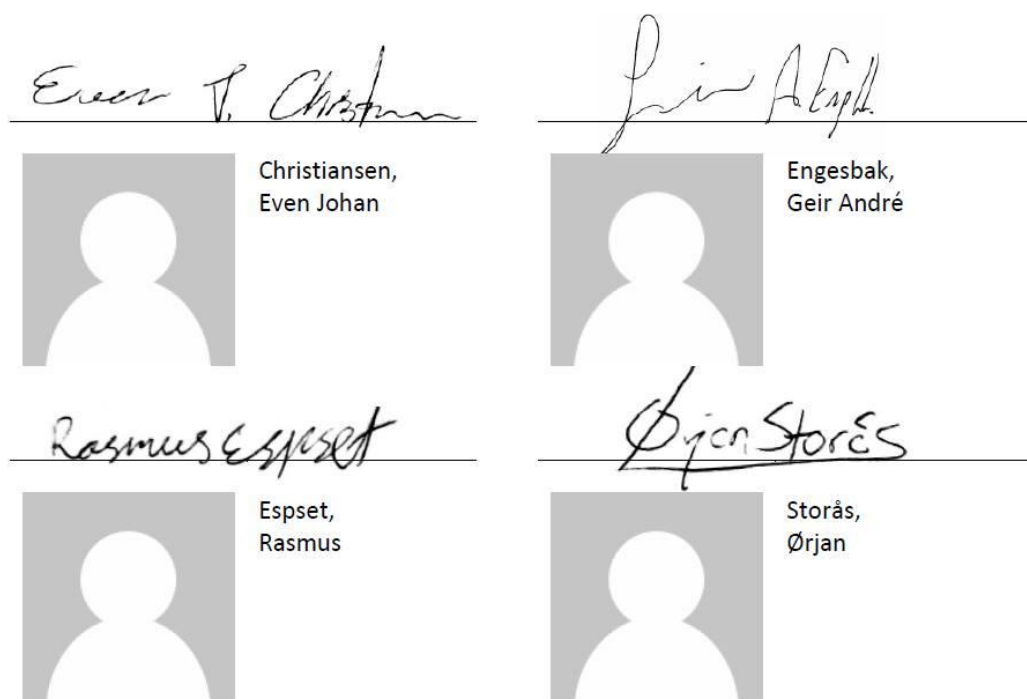
Dette er hovedrapporten for bachelorprosjektet «BLE-kube» ved NTNU, gjennomført i samarbeid med Nordic Semiconductor, i perioden januar-juni 2016. Prosjektet er gjennomført av en gruppe på tre bachelorstudenter fra studieretningen Industriell Instrumentering og én bachelorstudent fra studieretningen Elektronikk.

Denne rapporten er ment å gi leseren en detaljert innføring i hva gruppen satte seg som mål i prosjektets startfase, relevant fagteori, arbeidet som ble utført i prosjektperioden og hvordan resultatet ble til slutt. Fremgangsmetoder, avgjørelser og utfordringer vil også bli belyst.

Gruppen er særdeles fornøyd med arbeidet vi har gjennomført. Prosjektplanleggingen og ansvarsfordelingen gjorde det mulig for gruppen å håndtere et meget krevende prosjekt på en god måte, og vi føler sluttresultatet bærer preg av innsatsen gruppen har lagt ned i løpet av de siste seks månedene på en positiv måte.

Stor takk går til våre veiledere Håkon Grønning og Herman Raner fra NTNU for verdifulle innspill og deres engasjement i prosjektet, og til vår veileder fra Nordic Semiconductor, Joakim Tønnessen, for uvurderlig faglig støtte gjennom hele prosjektperioden. Ketil Erichsen og Rune Brandsegg fra Nordic Semiconductor fortjener også en ekstra takk. Rune bidro stort under utviklingen av kretskortet og Ketils hjelp var svært betydningsfull for utformingen av nettsiden. Vi vil også gjerne takke NTNU for tre utfordrende, men uforglemmelige år.

Særsilt takk går til Nordic Semiconductor for muligheten til å arbeide med deres teknologi, og for kurset som ble holdt i februar. Dette kurset ga oss viktig innsikt i virkemåten til deres utviklingsbrett, noe som var til stor hjelp.



Trondheim, våren 2016

## 2 Sammendrag

### *Norsk:*

Hensikten med denne oppgaven har vært å lage en leke for Nordic Semiconductor. Denne leken skulle benytte seg av *Physical Web*, og styres fra en mobiltelefon. Eksempelvis kunne dette vært en radiobil. Dette var noe som ble videreutviklet til en ny oppgave, som fikk tilnavnet «*BLE Cube – A Physical Web Toy*». Denne abstraksjonen av oppgaven baserte seg på å opprette toveiskommunikasjon med en mobiltelefon, i stedet for enveiskommunikasjonen oppgaveteksten foreslo opprinnelig.

Dette prosjektet har i stor grad basert seg på 3D-modellering, kretsdesign og kretskortproduksjon, utforming av nettsted og utvikling av programvare, noe denne rapporten vil representere.

Kubens fysiske skjelett ble 3D-modellert, og utskrevet i ABS-plast ved hjelp av en 3D-printer. Dette er plastmaterialet som benyttes i Lego. Det ble i tillegg produsert et skreddersydd kretsbrett for kuben. Størrelsen og signal-utgangene og -inngangene til kretsbrettet ble spesialtilpasset for konseptets behov og er dimensjonert etter kubens 3D-designede rammeverk. Kuben ble utstyrt med lysende trykkbare sider. Sidene ble lyst opp av rød/grønn/blå lysdioder, og registreringen av trykk skjedde via mekaniske knapper. Kuben ble også utstyrt med et internt litium ion polymer-batteri og en tilhørende ladekrets.

*Soft-* og *firmware* ble programmert på Nordic Semiconductor sin nRF52832 systemchip. Det ble skrevet et bibliotek for å kunne styre lysdiodene via skiftregistre. Det ble også skrevet kode for registrering av trykk på de trykkbare sidene, og deteksjon av risting av kuben. Risting blir detektert ved hjelp av et internt akselerometer.

Kuben kringkaster en URL, og kan kobles til en smarttelefon over *Bluetooth Low Energy*. Denne kringkastede URL-en markerer kubens relevans opp mot konseptet *The Physical Web* og veileder brukeren til en nettside. Denne nettsiden er egenprodusert og inneholder et spill som utnytter toveiskommunikasjon med kuben over *Bluetooth Low Energy*. Dette spillet går ut på at spilleren blir vist et bilde av en utbrettet kube, hvor hver side er tildelt én av seks mulige farger. Spilleren skal rekonstruere den utbrettede kuben på den fysiske kuben, ved at sidene skifter farge når de trykkes inn. Når spilleren tror at oppgaven er løst, skal kubens risting avleses for å avgis svar. En vibrasjonsmotor er benyttet for å gi tilbakemelding på hvorvidt dette svaret er korrekt.

Med høy dedikasjon og god innsats ble alle prosjektmålene nådd, og har resultert i en fungerende prototype. Denne prototypen vil fortsette sin reise med Nordic Semiconductor. Forhåpentligvis vil prototypen inspirere til videreutvikling av konseptet, samtidig som Nordic Semiconductor kan demonstrere potensialet til deres nyeste SoC, nRF52.

*English:*

The purpose of this assignment was to create a toy for Nordic Semiconductor. This toy is supposed to be a part of The Physical Web. In our case, this implies the ability to control the toy from a smart phone. This assignment was developed further, and given the name “BLE Cube — A Physical Web Toy”. This abstraction of the assignment is based upon establishing a two-way communication, instead of the one-way communication the assignment originally suggested.

This project leans heavily on 3D-modeling, manufacturing and design of circuit boards, development of a webpage and programming. This report will delve deeper into these things.

The physical structure of the cube was achieved by utilizing 3D modeling and consists of ABS-plastic. This is the same type of plastic that can be found in Lego. In addition to this, a tailor-made circuit board were produced for the cube. Its size, inputs and outputs where designed with the requirements of the concept, and the cubes inner frame, in mind. The cube is equipped with light-emitting sides which can be pushed. The sides of the cube lights up with an RGB LED, and utilizes mechanical buttons. It also features a lithium-ion polymer battery with a complimentary charge circuit.

Soft- and firmware was programmed on Nordic Semiconductors nRF52832 system chip. A library was written to interface the RGB LEDs, using shift registers. Code was also written for detection of button-presses and shaking. To detect shaking the cube is equipped with an internal accelerometer.

The cube will broadcast a URL and can connect to a smart phone via Bluetooth Low Energy. This URL marks the cubes relevance against The Physical Web and notifies the user of a web site. This web site is produced entirely by the group and features a game that utilizes two-way communication with the cube. This game shows the player a picture of an unfolded cube, where each side is assigned one of the six possible colors. The players task is to figure out how this unfolded cube could be “reassembled” into an actual cube by pressing the sides, and reconfiguring the color on each side of the physical cube. When the player believes that the physical cube represents the same layout as the webpage, the cube must be shaken to verify the solution. A rumble motor is utilized to indicate whether this solution is correct or not.

With dedication and effort, all of these goals where met, and the project has yielded a functional prototype. This prototype will continue its journey alongside Nordic Semiconductor. Hopefully, this prototype will spark further development of the concept, while allowing Nordic Semiconductor to demonstrate their versatile nRF52 Soc.

### 3 Innholdsfortegnelse

1	Forord .....	II
2	Sammendrag .....	III
3	Innholdsfortegnelse .....	V
4	Nomenklatur .....	1
4.1	Definisjoner .....	1
4.2	Forklaringer .....	2
5	Innledning .....	3
5.1	Bakgrunn .....	3
5.2	Oppgavetekst og problemstilling .....	3
5.3	Løsningskonsept .....	4
5.4	Videreutvikling og forkastede idéer .....	5
5.5	Arbeidsfordeling .....	6
5.6	Orientering og begrensninger .....	7
5.7	Hensikt og struktur .....	7
6	Interne rutiner .....	8
6.1	Samhandlings- og prosjektstyringsverktøy .....	8
6.2	Møter .....	8
6.3	Toukersrapport og timetelling .....	8
6.4	Stillingsorganisering .....	9
6.5	Utstys- og programvarebehov .....	10
7	Prosjektmål .....	11
7.1	Resultatmål .....	11
7.2	Effektmål .....	11
7.3	Prosessmål .....	11
7.4	Læringsmål .....	12
8	Teori .....	13
8.1	Physical web .....	13
8.2	Eddystone-beacon .....	13
8.3	NRF52 .....	14
8.4	nRF SoftDevice .....	14
8.4.1	S132 SoftDevice .....	14
8.5	BLE .....	15
8.6	GAP .....	16
8.7	GATT .....	16

8.8	Akselerometer.....	16
8.8.1	LIS2DH12 .....	16
8.8.2	ADXL335 .....	16
8.9	SAADC .....	17
8.10	Litiumbatteri .....	18
8.11	HTML/CSS & JavaScript.....	18
9	Spesifikasjoner og egenskaper .....	19
10	Prosess og gjennomført arbeid.....	20
10.1	Innledning .....	20
10.2	Testbrett og demokort.....	20
10.3	Kretsdesign [RE].....	22
10.3.1	Innledning .....	22
10.3.2	Hva har blitt gjort?.....	22
10.3.3	Resultat .....	27
10.3.4	Diskusjon .....	28
10.4	Software [EJC & ØS] .....	31
10.4.1	Innledning [EJC] .....	31
10.4.2	Hva har blitt gjort.....	31
10.4.3	Resultat [EJC].....	40
10.4.4	Diskusjon .....	40
10.5	3D-Modellering [GAE] .....	43
10.5.1	Innledning .....	43
10.5.2	Hva har blitt gjort?.....	43
10.5.3	Resultater .....	45
10.5.4	Diskusjon .....	46
10.6	Sammensetning av prototyp .....	49
10.7	Samlede Resultater .....	51
10.8	Samlet diskusjon.....	52
10.8.1	Beslutninger og veivalg.....	52
10.8.2	Andre utfordringer.....	53
10.8.3	Forbedringspotensiale .....	54
10.9	Samlet Konklusjon .....	55
11	Referanser/Litteratur.....	56
11.1	Kildeliste .....	56
11.2	Vedleggsliste .....	58
11.3	Figurliste .....	59

11.4	Tabelliste .....	60
12	Vedlegg .....	61
12.1	Kretsskjema .....	61
12.2	Pinmap .....	62
12.3	Pinne-definisjoner - BLE-cube circuit board (nRF52) .....	63
12.4	BOM / komponentliste.....	66
12.5	Hovedkretsbrett - Øvrige bilder .....	68
12.6	Tilpasningskretsbrett – Øvrige bilder .....	73
12.7	Flytskjema for blecube.github.io.....	76



## 4 Nomenklatur

### 4.1 Definisjoner

<b>ADC</b>	Analog-to-Digital Converter
<b>API</b>	Application Programming Interface (Forklart under forklaringer)
<b>BLE</b>	Bluetooth Low Energy (Bluetooth Smart/Bluetooth Version 4+)
<b>BOB</b>	Breakout Board
<b>CSS</b>	Cascading Style Sheets
<b>FIFO</b>	First in, first out
<b>GAP</b>	Generic Access Profile
<b>GATT</b>	Generic Attribute Profile
<b>GPIO</b>	General Purpose Input/Output
<b>GPIOE</b>	General Purpose Input/Output Tasks/Events
<b>HiST</b>	Høgskolen i Sør-Trøndelag
<b>HTML</b>	HyperText Markup Language
<b>HW</b>	Hardware
<b>JS</b>	JavaScript
<b>LED</b>	Light Emitting Diode
<b>NS</b>	Nordic Semiconductor
<b>NTNU</b>	Norges Teknisk- og Naturvitenskapelige Universitet
<b>PCB</b>	Printed Circuit Board
<b>PPI</b>	Programmable Peripheral Interconnect
<b>RF</b>	Radio Frequency
<b>RGBL</b>	Red, Green, Blue LED
<b>SAADC</b>	Successive Approximation Analog-to-Digital Converter
<b>SMD</b>	Surface Mount Device
<b>SoC</b>	System on Chip
<b>SW/FW</b>	Software/Firmware
<b>URL</b>	Uniform Resource Locator
<b>VPN</b>	Virtual Private Network
<b>EJC</b>	Even Johan Christiansen
<b>GAE</b>	Geir André Engesbak
<b>ØS</b>	Ørjan Storås
<b>RE</b>	Rasmus Espset
<b>HR</b>	Herman Ranæs
<b>JT</b>	Joakim Tønnesen
<b>KE</b>	Ketil Erichsen
<b>RB</b>	Rune Brandsegg

## 4.2 Forklaringer

API	<p>Application Programming Interface</p> <p><i>Applikasjonsprogrammeringsgrensesnitt</i>, betegner et grensesnitt i en programvare slik at spesifikke deler av denne kan aktiveres (kjøres) fra en annen programvare. API-et beskriver de metoder som en gitt programvare eller et bibliotek kan kommunisere med. En god API gjør det lettere å utvikle et program, ved å tilby alle byggeblokkene. En programmerer vil så sette disse blokkene i sammen for å få det ønskede programmet.</p>
GPIOTE	<p>General Purpose Input/Output Tasks/Events</p> <p>System for å enkelt bruke avbrudd for å endre programflyt.</p>
Google Drive	<p>Lagringsmedie</p> <p>Brukes for lagring av filer, slik at de enkelt er tilgjengelig for hele gruppa, uansett lokasjon.</p>
ItsLearning	<p>ItsLearning</p> <p>Læringsplattform som er brukt gjennom hele studiet. Bacheloroppgaven har stilt krav til at visse dokumenter skal leveres på ItsLearning. Eksempelvis toukersrapporter og møttereferat.</p>

## 5 Innledning

Denne rapporten vil oppsummere prosjektarbeidet som har blitt gjennomført i perioden januar-juni 2016 i sammenheng med bacheloroppgaven «*BLE-kube – en leke for det fysiske nett.*» Rapporten er i hovedsak skrevet slik at en elektroingeniør vil kunne lese og forstå innholdet, men burde til dels være forståelig for personer med interesse for enten elektronikk, mikrokontrollere eller produktutvikling.

### 5.1 Bakgrunn

Dette prosjektet har sitt utsprang i én av flere oppgaver Nordic Semiconductor hadde lagt frem som oppgaveforslag for studenter ved NTNU. Denne oppgaven gikk ut på å modifisere eksisterende leketøy for bruk med det fysiske nettet. Dette inspirerte deretter gruppens medlemmer til å få sine egne tanker og idéer rundt temaet. Ikke lenge etter tok gruppen kontakt med NS og presenterte forslaget om å produsere en helt ny 'leke', skreddersydd for det fysiske nettet.

### 5.2 Oppgavetekst og problemstilling

Som nevnt hadde dette bachelorprosjektet sitt utsprang som en oppgave Nordic Semiconductor hadde tilbudt NTNU. De presenterte følgende oppgavetekst:

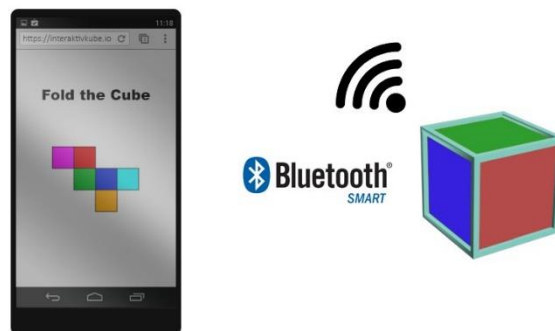
*«The Physical Web enables Bluetooth Smart devices to be controlled by a mobile phone without downloading an app. The device is controlled directly from a web page in the browser. A device can be a toy car, plane, quadcopter, boat, candy machine or whatever you can imagine. The task is to create a Physical Web enabled device and to control this with a mobile phone. This can be achieved by either creating custom hardware or by hacking an existing toy. It is up to you! The task involves writing firmware in the C programming language for a Nordic Semiconductor chip (ARM-based) and software for the browser in HTML, CSS and JavaScript. Nordic Semiconductor will provide necessary development kits for free as well as help with firmware/software/hardware»*

Med grunnlag i denne oppgaveteksten bestemte gruppen seg for å skape et eget leketøy og formulerte følgende problemstilling:

*«Klarer vi å skape et fungerende produkt som vekker interesse og benytter Nordics nRF52, Bluetooth Low Energy/Bluetooth Smart og Physical Web på en god måte? Klarer vi å gjennomføre arbeidet med dette konseptet på en slik måte at det inspirerer og legger opp til videreutvikling?»*

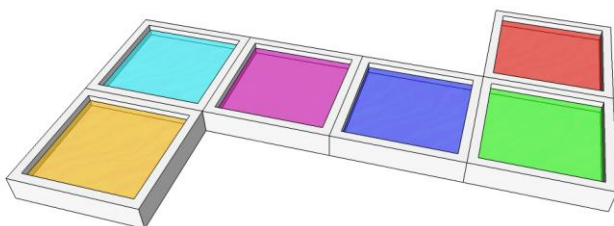
## 5.3 Løsningskonsept

Den opprinnelige visjonen gikk ut på å konstruere en kubeformet elektronisk leke. Denne kuben ble originalt tildelt tittelen «Interaktiv Kube», men dette ble senere endret til «BLE-kube» for å reflektere det høye fokuset på BLE-delen av prosjektet. Den originale tittelen beskriver derimot produktet godt; det skulle være en kube man kan plukke opp hvor hver av de seks sidene skulle være trykkbare og lyse opp i forskjellige farger.



Figur 5-1 Konseptskisse

Et krav for oppgaven var at kuben skulle benytte seg av konseptet *Physical Web*<sup>[19]</sup>. I tillegg var det ønskelig for gruppens medlemmer å inkludere *Eddystone beacon*-teknologi<sup>[6]</sup>. Dette innebærer at kuben skal være i stand til å kringkaste en URL. Det vil da dukke opp en nettside på nærliggende smarttelefoner med støtte for dette. Fra denne nettsiden skal man kunne initiere kommunikasjon mellom kuben og smarttelefonen. Deretter kan man starte et spill som krever at brukeren benytter begge enhetene for å løse spillet. Toveiskommunikasjonen mellom disse to enhetene er essensiell for virkemåten til spillet. Når det kommer til spillets virkemåte, falt valget på spillkonseptet «*Fold It*». Spillet baserer seg på at nettsiden indirekte indikerer en spesifikk fargekonfigurasjon i form av «utbrettede» kuber som brukeren skal forsøke å gjenskape på den fysiske kuben. I tillegg skulle kuben inneholde en enkel demonstrasjonsmodus, «*Light Show*», hvor kuben lyser opp i forskjellige farger.



Figur 5-2 Utbrettet kube

Utover disse egenskapene ble det også fantasert om diverse andre funksjonaliteter og anvendelsesområder. Flesteparten av disse ble forkastet grunnet tidsbegrensninger. Derav kom ønsket om at produktet skulle være så enkelt som mulig å videreutvikle, noe som ble et viktig fokus i prosjektarbeidet.

## 5.4 Videreutvikling og forkastede idéer

Det ville vært til stor glede for gruppen om noen tok dette konseptet videre, fant inspirasjon i arbeidet som er blitt nedlagt, eller om noen lagde sin egen spesielle vri på prosjektet. Potensialet for videreutvikling er derfor åpenbart, og gruppen valgte å gjennomføre arbeidet på en måte som legger grunnlaget for dette.

Mange av idéene gruppen hadde innledningsvis har som nevnt blitt forkastet på grunn av manglende tid eller kapasitet til å gjennomføre de, men det er verdt å nevne noen av disse.

- Kuben kunne ha benyttet en induktiv ladeplattform.
- Alle sidene kunne bestått av berøringssensitive LCD-skjermer eller kapasitive trykknapper.
- Kuben kunne vært en god ressurs innen programmeringsundervisning.
- GPS-avhengig samspill mellom flere spillere.
- Fjernkontroll til huset der hver side har sin funksjon og aktiverer forskjellige ting i huset, for eksempel ringeklokke, fjernsyn, kaffetrakter og lignende.

Opprinnelig ønsket gruppen at nettstedet skulle inneholde flere spillmuligheter enn kun «*Fold It*» og «*Light Show*». I den innledende idémyldringsfasen av prosjektperioden diskuterte vi idéer som «*Escape the Room*» og «*Simon Says*». Idéen med «*Escape the Room*» er at kubens må brukes for å løse gåter og oppgaver for å komme seg ut av et fiktivt rom, som vises på nettstedet. «*Simon Says*» skulle fungert slik spillet er kjent. En tilfeldig sekvens lyser opp som bruker skal gjenta. Vanskelighetsgrad øker mellom hvert vellykkede forsøk etter som sekvensen blir lengre.

## 5.5 Arbeidsfordeling

Gruppen innså tidlig at dette prosjektet ville bli omfattende og sette store krav til både arbeidsinnsats, så vel som hvert enkelt gruppemedlems kompetanse. Derfor ble det nødvendig å delegere arbeidsoppgaver i form av fokusområder. Fordelingen ble foretatt med hensyn på en kombinasjon av ønsket arbeidsområde og medlemmets kunnskapsnivå innen det aktuelle temaet. RE fikk ansvaret for hardware, kretsdesign og kretskort. GAE fikk ansvaret for 3D-modellering og fysisk utforming. ØS fikk ansvaret for utvikling av nettsted. EJC fikk ansvaret for utvikling av programvare. Endringer i arbeidsfordelingen ble etter hvert nødvendig ettersom fremgang på enkelte områder uteble. EJC tok derfor over arbeidet med nettstedet. Arbeidet med programvare ble delt mellom EJC, som stod for samspill mellom nettside og programvare, og ØS som fikk ansvaret for akselerometer og *beacon*-funksjonalitet. EJC og ØS samarbeidet om skiftregisterbiblioteket. Dette gjenspeiler seg i innholdsfortegnelsen, da denne beskriver hvilke deler hvert gruppemedlem har arbeidet med. Hvert av disse ansvarsområdene blir beskrevet nærmere i kapittel 10. Beskrivelse av hva som er gjort. Interne rutiner, for eksempel møter og fremdriftsrapporter, blir nærmere beskrevet i kapittel 6 - Interne Rutiner.

Ulempen med denne fremgangsmåten er at hvert enkelt gruppemedlem ender opp med begrenset innsikt i enkelte av prosjektets forskjellige aspekter. Til tross har dette vist seg å være et meget godt valg etter som tiden ble svært knapp mot slutten av prosjektperioden. Hvert enkelt gruppemedlem ville ikke hatt tid til å fordype seg i flere fokusområder og samtidig vedlikeholdt samme resultatet. På sin side ble derfor sluttresultatet svært godt.

Nedenfor ser man de opprinnelige milepælene, hovedansvarlige og frister.

Milepæl	Hovedansvar	Dato
Utlegg av demoversjon	EJC & RE	15. Februar
Ferdigstilt demoversjon	EJC	15. Mars
Skjelett for nettsted/spill	EJC & ØS	15. Mars
Utviklet nettsted/spill	EJC & ØS	02. Mai
Utlegg av hovedkort	RE	04. April
3D-modell	GAE	04. April
Ferdigstilling av prototyp	Alle	09. Mai
Sluttrapport	Alle	24. Mai

Tabell 5-1 Opprinnelig ansvarsfordeling

## 5.6 Orientering og begrensninger

Produktet var i hovedsak tiltenkt som et demonstrasjonsobjekt for teknologien Nordic Semiconductor tilbyr. Dette medførte at gruppen var delvis fritatt fra enkelte krav som gjelder for produktutvikling generelt. Dette innebærer at utredning av spesifikke målgrupper, endelig anvendelsesområde og slike ting ikke har blitt belyst i særlig grad i løpet av denne prosjektperioden. Det er verdt å nevne at aspekter ved produktet likevel blir belyst i kapittel 9 – Spesifikasjoner og egenskaper.

Gruppen valgte å søke om utsettelse for å sikre at prototypen ble ferdig. Dette skyldtes blant annet at flere små forsinkelse i løpet av prosjektet til sammen førte til at den originale tidsfristen ble ubehagelig kort. Årsaken til disse forsinkelsene vil bli belyst i løpet av rapporten. Denne utsettelsen resulterer i at denne rapporten har blitt ferdigstilt før alt arbeidet med prototypen ble gjennomført. Det gjenstår derimot svært lite og alle aspektene og funksjonene ved prototypen er allerede testet og verifisert. Dette går derfor ikke utover gruppens evne til å diskutere, eller trekke konklusjoner angående resultatet.

Gruppen så tidlig et stort potensiale i produktet og ønsket derfor å legge vekt på mulighetene for videreutvikling og tilrettelegging for et bredt spekter forskjellige anvendelsesområder. Det ble derfor bestemt å gjennomføre arbeidet med dette i fokus, samt at alle aspekter ved prosjektet skulle være fritt tilgjengelig for alle.

## 5.7 Hensikt og struktur

Denne rapporten er ment å gi leseren god innsikt i metodene som ble benyttet under prosjektperioden, slik at vedkommende er i stand til å gjenskape prosjektet. Rapporten vil også presentere hvordan prosjektgruppen gikk frem for å gjennomføre prosjektet.

Innledningsvis vil rapporten bestå av et kapittel om gruppens interne rutiner. Deretter presenteres gruppens prosjektmål. Videre følger en teoridel for å gi leseren en kjapp innføring i den mest relevante bakgrunnskunnskapen. Så blir det gitt en kort beskrivelse av de viktigste aspektene ved prototypen prosjektgruppen har produsert. Denne enkle beskrivelsen blir presentert i forkant av de mer dyptgående delene av rapporten, hvor det blir gått gjennom i detalj hvordan arbeidet har blitt gjennomført. Det neste kapittelet omhandler det faktiske arbeidet som er utført. Dette kapittelet deles opp etter hvert enkelt gruppemedlems fokusområde og er i hovedsak utformet av vedkommende som har ansvar for den aktuelle delen. Kapittelet er også svært omfattende og hadde vært utfordrende å skrive på en annen måte grunnet arbeidsfordelingen. Som nevnt inneholder innholdsfortegnelsen informasjon om hvem som har gjort hva. Deretter beskrives arbeidet med å spleise sammen de forskjellige fokusområdene til ett helhetlig produkt. Til slutt følger en stor diskusjons- og konklusjonsdel, etterfulgt av de viktigste vedleggene. Øvrige vedlegg er tilgjengelig digitalt.

## 6 Interne rutiner

### 6.1 Samhandlings- og prosjektstyringsverktøy

Gruppen har benyttet seg av kommunikasjonskanalen Slack for generell diskusjon og planlegging av trivielle ting. Google Drive ble benyttet som lagringsmedie slik at alle dokumenter, bilder, avtaler og lignende er tilgjengelig for hvert enkelt medlem. I tillegg ble det opprettet en egen Gmail-konto for prosjektet. Alle medlemmer av gruppen hadde full tilgang til denne kontoen og all kommunikasjon fra gruppen til andre kom fra denne kontoen. Videre ble kalenderen til denne kontoen koblet sammen med Slack, slik at all info om møter automatisk dukket opp på Slack. Førsteutkastet av prosjektrapporten ble skrevet i Google Docs, da samskrivningsegenskapene til dette systemet fungerte godt. Renskriving og finpuss av rapportens utseende ble likevel gjennomført i Word.

Gruppen forsøkte innledningsvis å benytte seg av Gantt-skjema for å holde styr på prosjektets fremgang. Det viste seg derimot at god bruk av toukersrapporter ga en tilstrekkelig oversikt over tidsfrister og delmål. Gantt-skjemaet ga likevel verdifull oversikt over de største og viktigste milepælene og planlagt rekkefølge av diverse oppgaver.

### 6.2 Møter

I løpet av prosjektperioden forsøkte gruppen å avholde minst ett organisert møte ukentlig. Gruppen har stort sett jobbet sammen i løpet av hele prosjektperioden, slik at mange avgjørelser ble gjort uten at et møte var nødvendig. Annenhver fredag ble det også gjennomført møter med gruppens veiledere fra NTNU og NS slik at gruppen kunne spørre om assistanse i tilfeller hvor arbeidet stopper opp eller går tregere enn planlagt. Alle innkallinger og referater ble fortløpende opplastet til både Google Drive og ItsLearning.

### 6.3 Toukersrapport og timetelling

I løpet av prosjektperioden har gruppen gjennomført rapportering av fremdrift i form av toukersrapporter. Formålet med denne rapporteringen var å gi veileder og gruppens medlemmer muligheten til holde oversikt over prosjektets helhetlige fremgang, samt å gi medlemmene evnen til å dokumentere og stå for egen innsats. Alle mål, milepæler, avvik og individuelt tidsbruk har blitt godt dokumentert. En annen fordel med toukersrapportene er at gruppen har vært i stand til å definere delmål underveis i arbeidet med de store milepælene fra Gantt-skjemaet.



## 6.4 Stillingsorganisering

Som nevnt i innledningen har hver enkelt på gruppen fått tildelt sitt område vedkommende har hovedansvaret for. Utenom dette ble administrative oppgaver fordelt på følgende måte:

Dato (Fra)	Dato (Til)	Møteinnkaller	Møteleder	Møtereferent	Toukersrapport
15.02.16	28.02.16	Even	Ørjan	Geir	Ørjan
29.02.16	13.03.16	Even	Ørjan	Geir	Ørjan
14.03.16	27.03.16	Even	Rasmus	Even	Even
28.03.16	10.04.16	Even	Rasmus	Even	Even
11.04.16	24.04.16	Even	Even	Ørjan	Geir
25.04.16	08.05.16	Even	Even	Ørjan	Geir
09.05.16	22.05.16	Even	Geir	Rasmus	Rasmus
23.05.16	05.06.16	Even	Geir	Rasmus	Rasmus

Tabell 6-1 Administrativ fordeling

## 6.5 Utstyrs- og programvarebehov

I løpet av dette prosjektet har det blitt benyttet mange forskjellige verktøy, apparater og programmer.

Verktøy	Programmer	Øvrige ressurser
Multimeter	Keil $\mu$ Vision	Utviklingsbrett for nRF52
Loddebolt	Altium Designer	3D-printer (Stratasys Mojo)
Loddesuger	nRFGO Studio	Loddestasjon for SMD
Loddelisse	Segger RTT Viewer / Putty	Loddeovn
Loddetinn	Notepad++	Infrarød loddeovn
Flussmiddel	3Ds Max	
Varmluftspistol	Google SketchUp	
Limpistol	Nordic Master Control Panel	
Superlim		
Skoletyggis		
Epoxykitt		
Sandpapir		
Mikroskop		
Multiverktøy (Dremel)		
Ledningsstripper		
Diverse pinsetter og tenger		
Kniver (X-acto)		

Tabell 6-2 Benyttede verktøy og ressurser

## 7 Prosjektmål

Målet med prosjektet er å utvikle leken BLE kube som skal brukes sammen med *Physical Web*. Leken skal kunne brukes av NS som ett demonstrasjonsobjekt for deres teknologi, ved diverse messer eller presentasjoner. Det vil også bli laget en forumpost som er å finne på blant annet GitHub som skal reklamere for prototypen og gi lesere en veiledning til hvordan de selv kan bygge videre på konseptet.

### 7.1 Resultatmål

- Lage et leketøy NS kan bruke til å promotere deres nyeste SoC, nRF52832, og noen av dens funksjoner.
- Lage en nettside som kan styre/utveksle informasjon med leketøyet via BLE.
- Leketøyet skal kringkaste en link til nettsiden i punktet over, via *Physical Web*.
- Vi skal ha tilrettelagt for at andre skal kunne videreutvikle produktet. Dette omhandler produktets dokumentasjon og forumpost nevnt ovenfor.
- Produktet skal være ferdig 15. juni og sluttrapport skal være innlevert innen 09. juni 2016.

### 7.2 Effektmål

- Fullføre treårig bachelorutdanning innen elektrofag.
- Gi NS et produkt de kan bruke for å vise frem sin teknologi.
- Gi gruppens medlemmer muligheten til å tilegne seg yrkesspesifikke ferdigheter og bygge kompetanse innen disse områdene.

### 7.3 Prosessmål

- Gjennomføre prosjektet effektivt ved å nå gitte milepæler.
- Holde tidsfrister.
- Danne gode rutiner for å lettere kunne gjennomføre prosjektet.
- Bygge ny kompetanse innen tidligere ukjente fagområder, som for eksempel:
  - ◆ Altium
  - ◆ 3ds-Max
  - ◆ nRF52-serien

- Bli bedre kjent med prosjektdeltakerne.
- Ha et sosialt og trygt arbeidsmiljø innad i gruppen.
- Holde prosjektet innenfor gitte budsjetttrammer, 5000kr.

## 7.4 Læringsmål

- Nordic Semiconductors system
- Altium Designer
- 3Ds Max
- JavaScript

## 8 Teori

Her kommer nødvendig teori for å forstå en del konsepter senere i rapporten. Den gir også innføring i teknologien oppgaven baserer seg på.

### 8.1 Physical web

Tanken bak *Physical Web*-prosjektet<sup>[14]</sup> er at fysiske objekter i hverdagen skal kunne gi informasjon til smarttelefoner eller nettbrett man har med seg. Dette skjer via kringkasting av en URL, en lenke til et nettsted. For å nyttiggjøre seg av URL-en er mottaker avhengig av tilgang til internett via mobilnett/WiFi. Disse fysiske objektene kan være:

- Halsbånd for kjæledyr som sender informasjon om dyr/eier.
- Bussholdeplasser som sender lenke til nettside med rutetider.
- Severdigheter som sender informasjon om stedet man er på.

Om denne URL-en sendes via BLE kan en forlengelse av bruken være å la nettsida som den mottatte URL-en lenker til samhandle med enheten som sender ut URL-en via BLE integrert på smarttelefonen eller nettbrettet som viser nettsida.

### 8.2 Eddystone-beacon

*Eddystone* er en protokollspesifikasjon utviklet av Google<sup>[6]</sup>. Denne spesifikasjonen definerer et BLE format for *beacon*-er, som ble sluppet i juli 2015. *Eddystone*-formatet er oppkalt etter et britisk fyrtårn med samme navn<sup>[8]</sup>, for å symbolisere protokollens form for enveiskommunikasjon. *Eddystone*-protokollen beskriver en rekke ulike rammeverk som kan brukes individuelt, eller i kombinasjon med hverandre. Disse kan brukes for å opprette *beacon*-er for en rekke ulike formål. Rammeverket som er viktigst i denne oppgaven er *Eddystone*-URL<sup>[5]</sup>. Dette er en underprofil, som brukes som del av *Physical Web*-prosjektet, hvor enheten kringkaster en URL. Dette tolkes deretter av mottakerapplikasjonen som en lenke, som igjen sender brukeren til en nettside.

*Eddystone* har i tillegg en ramme som heter *Eddystone*-UID, denne brukes typisk til å sende informasjon til skreddersydde applikasjoner, som er forhåndkonfigurert til å lese dataen fra den gitte *beacon*-en. Selv om *Eddystone*-protokollen, da spesielt *Eddystone*-UID ligner på *Apple* sin *iBeacon*-profil lansert desember 2013, kan *Eddystone* implementeres uten restriksjoner<sup>[4]</sup>. Den inneholder også en telemetri-ramme (*Eddystone*-TLM) som kan rapportere enhetens helse. Dette kan eksempelvis være enhetens batterinivå, temperatur eller antall pakker sendt. URL-rammeverket og *Physical Web* er også noe som skiller *Eddystone* fra *iBeacon*. Ettersom *Physical Web* er en viktig del av denne oppgaven, falt protokollvalg på Google sin *Eddystone* protokollspesifikasjon, med *Eddystone*-URL.

Siden *beacon*-teknologien er satt opp slik at den bare sender ut data, kan den ikke identifisere om det er andre enheter i nærheten. Foreløpig fungerer det slik at om man vil lese og tolke et *beacon*-signal, trenger man en applikasjon eller en nettleser med støtte for *The Physical Web*. Av de vanligste nettleserne, er det foreløpig Google Chrome og Opera som er tilrettelagt for *Physical Web*. Det finnes også en rekke andre, mindre kjente nettlesere med denne tjenesten<sup>[11]</sup>.

Teknologien er åpen, slik at andre selskaper kan tilpasse sine applikasjoner til å lese *Eddystone beacon*-signaler. Dette forutsetter at enheten har en applikasjon med muligheten til å lese BLE-signaler. Protokollen er fleksibel og brukervennlig, noe som har bidratt til å gjøre dette til en foretrukket standard hos utviklere. Siden *beacon*-et bare sender signaler blir personvernet tatt vare på, fordi den ikke samler inn informasjon fra forbipasserende. Problemet rundt personvernet blir ikke berørt før brukeren eventuelt bruker *beacon*-et til å koble seg opp mot en nettside.

I april 2016 slapp Google sitt nye *Eddystone*-EID (*Ephemeral ID*) som et rammeverk. Rammen definerer en kryptografisk og sikker metode for å konfigurere *beacon*-en til å kringkaste informasjon som kun autoriserte mottakere kan dekryptere og lese. Den inneholder også støtte for sikker overføring av TLM (telemetri) informasjonen. Et av designmålene bak *Eddystone* er at det skal fungere godt med Android og iOS sine utvikler-API-er. Fleksibel arkitektur slik at man lett kan designe og implementere nye rammeverk. Og at man lett kan implementere *Eddystone* i eksisterende enheter.

*Eddystone* definerer en GATT-konfigurasjonstjeneste som tilrettelegger for interoperabilitet mellom HW-produsenter og applikasjonsutviklere. Den lar *beacon*-en rapportere sine evner til applikasjoner, og lar *beacon*-ets kringkastede informasjon være mulig å omkonfigurere. Denne tjenesten er også nødvendig for å konfigurere og registrere seg som en *Eddystone*-EID *beacon*.

## 8.3 NRF52

nRF52832 er en ultra lav-effekt 2.4GHz trådløs SoC, som integrerer nRF52 seriens 2.4 GHz transceiver og en ARM® Cortex®-M4 CPU med flash-minne<sup>[9]</sup>. Komplette BLE-stacks for nRF52832 er implementert i SoftDevice-er av S100-serien. Disse SoftDevice-ene er gratis og tilgjengelige på NS sine nettsider. Den har også kodekompatibilitet med enheter fra nRF52 produktfamilien.

## 8.4 nRF SoftDevice

Et SoftDevice er en alenestående, forhåndskompilert FW-blokk laget av NS for trådløs kommunikasjon via deres System on Chip (SoC) løsninger. Disse kan være spesialdesignet for kommunikasjon via ordinær Bluetooth, BLE eller ANT. Fordelen med en slik løsning, er at SW-utvikler kun trenger tenke på utvikling av applikasjonen, da protokoller for trådløs kommunikasjon er forhåndstestet og garantert å fungere. SoftDevicet sitt HW- og SW-rammeverk gir kjøretid, minnebeskyttelse, trådsikkerhet og deterministisk sanntidoppførsel. APIet er deklart i headerfiler tilhørende C programmeringsspråket. Dette gjør det lett for applikasjonsprogrammerer å finne virkemåte og mulige funksjonsskall. Disse karakteristikene gjør at grensesnittet kan ligne på det til en HW driverabstraksjon. I dette tilfellet en komplett trådløs protokoll.

### 8.4.1 S132 SoftDevice

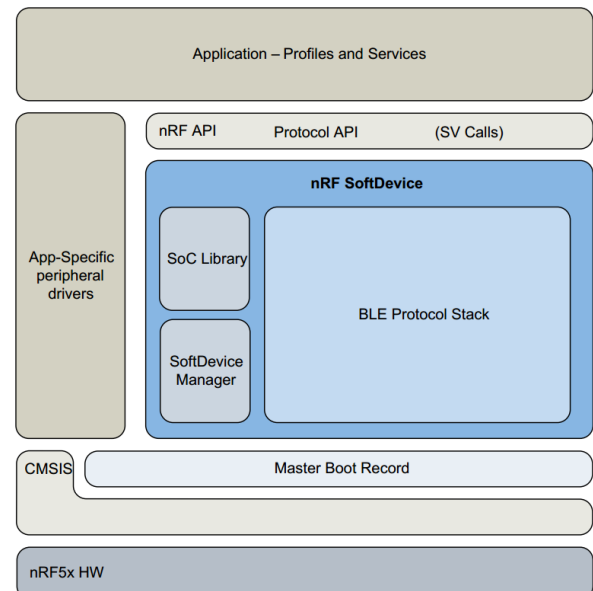
S132 SoftDevice-et er en protokollprogramløsning for sentral og periferi<sup>[13]</sup>. SoftDevice-et har et API spesielt tilpasset BLE på NS sin nRF52 SoC serie. Det støtter opptil åtte tilkoblinger med en ekstra observasjon og kringkasterrolle som kjøres samtidig. I starten ble S132 versjon 2.0.0-7. Alpha ble benyttet siden den ble presentert sammen med SDKet, som ble utdelt i starten av

prosjektperioden. Da nRF52832 ble montert på prosjektets egenproduserte PCB, ble SoftDevice-et oppdatert til S132 versjon 2.0.0. for å håndtere problemene som var til stede i alphaversjonen.

S132 kan ha roller som sentral (*central*), kringkaster (*broadcaster*), observator (*observer*), og periferi (*peripheral*). SoftDevice-et har mulighet til å ha en eller flere av disse rollene samtidig. Den kan ha opptil åtte parallelle tilkoblinger til en observator og en kringkaster. For dette prosjektet er denne egenskapen nyttig, da det er ønskelig å både kunne være kringkaster og periferi. Kringkaster for å sende *Eddystone*-URL-en og periferi for å kunne koble seg til enheten nettspillet utføres på. S132 har minneisolasjon mellom applikasjonen og protokoll-*stack*-en, dette øker sikkerheten og gjør systemet mer robust.

Et SoftDevice består av tre hovedkonsept:

- SoC Bibliotek - Dette inneholder et nRF API og implementasjon for delt ressursstyring av HW (Application Coexistence)
- SoftDevice Manager - Inneholder et API og implementasjon for styring av SoftDevicet, dette innebærer blant annet å kunne skru det på og av.
- Bluetooth® 4.2 low energy protokoll *stack* - Implementasjon av protokoll *stack* og API.



Figur 8-1 Oppbygning av SoftDevice

SoftDevice-et har innebygget API for GATT og for GAP; begge disse protokollene er forklart nærmere i teorien. Disse API-ene gjør jobben med å tilpasse protokollene lettere for programmerer. SoftDevice-et har et 100% event-basert API. Kort forklart innebærer dette at programkoden ikke trenger kjøre kode før et event inntreffer. Man kan da eksempelvis sette SoC i strømsparemodus, mens man venter på nytt event. At SoftDevice-et er event-basert er også viktig å vite når applikasjonskoden skrives. Hvis programmet gjør en oppgave inne i en løkke, kan dette hindre programmet å detektere hendelser som hindrer programmet å gjøre rett oppgave til rett tid.

## 8.5 BLE

Bluetooth low energy (BLE), er fjerde generasjons Bluetooth. Tanken er at man reduserer pakkeoverføringshastigheten fra tidligere versjon, for å samtidig redusere strømtrekket. BLE har ikke særlig lang rekkevidde, noe som gjør det spesielt godt tilpasset applikasjoner hvor man vil sende enkel informasjon over korte distanser. Disse spesifikasjonene, spesielt med det lave strømtrekket, gjør at den egner seg godt til Internet of Things (IoT). BLE er også offisielt støttet i iOS og Android mobilplattformer, i tillegg til de fleste *desktop*-plattformer inkludert OSX, Linux og Windows 8.

## 8.6 GAP

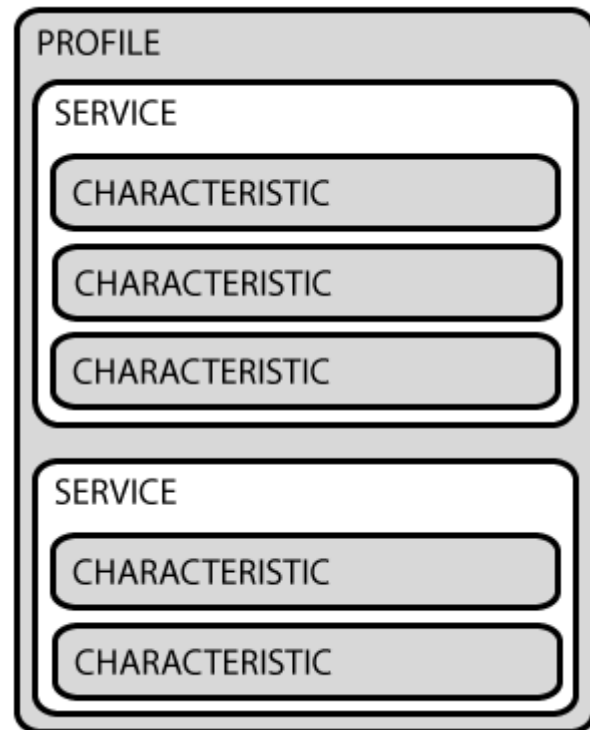
*Generic Access Profile* (GAP), definerer hvordan BLE-aktiverte enheter kan gjøre seg selv tilgjengelig, og hvordan to enheter kan kommunisere med hverandre.

## 8.7 GATT

GATT er et akronym for *Generic Attribute Profile*, som definerer måten to BLE-enheter kommuniserer på. GATT fokuserer spesielt på at data er formatert, pakket og sendt i henhold til dets forklarte regler. Innebygd i GATT er tre forskjellige protokoller. Disse kalles profil, tjeneste og karakteristikk, og kan se slik ut.

For å klargjøre hva dette innebærer, kan vi se på en profil som allerede eksisterer. Denne kalles «*Heart Rate Profile*»<sup>[14]</sup>.

Denne profilen definerer to tjenester, «*Heart Rate Sensor*» og «*Collector*». Altså en sensor som måler hjerterytme og en enhet som mottar denne data. Disse tjenestene består av karakteristikker. «*Heart Rate Sensor*» har for eksempel en egen karakteristikk for sending av hjerterytme.



Figur 8-2 GATT profil

## 8.8 Akselerometer

### 8.8.1 LIS2DH12

LIS2DH12 er et ultra laveffekt, høy-ytelses tre-akse lineært akselerometer, med I2C/SPI seriell interface standard output. Det klarer å måle akselerasjon, og sende ut med senderate på 1 Hz til 5.3 kHz. Akselerometeret har også en innebygget selvtestfunksjon, som lar bruker sjekke funksjonalitet i den endelig applikasjonen. Enheten kan konfigureres til å generere interrupts både på fritt fall, og på *wake-up*. Akselerometeret har et veldig lite fotavtrykk (2.0 × 2.0 × 1.0 mm), noe som er gunstig når man skal finne plass på et PCB. Akselerometeret er garantert til å operere innenfor et utvidet temperaturspekter, -40 °C til +85 °C. Akselerometeret kan forsynes med en spenning fra 1.71 V til 3.6 V. Har innebygde «*Sleep to wake*» og «*return to sleep*» funksjoner. Innebygd temperatursensor og innebygd FIFO-filter. <sup>[6]</sup>

### 8.8.2 ADXL335

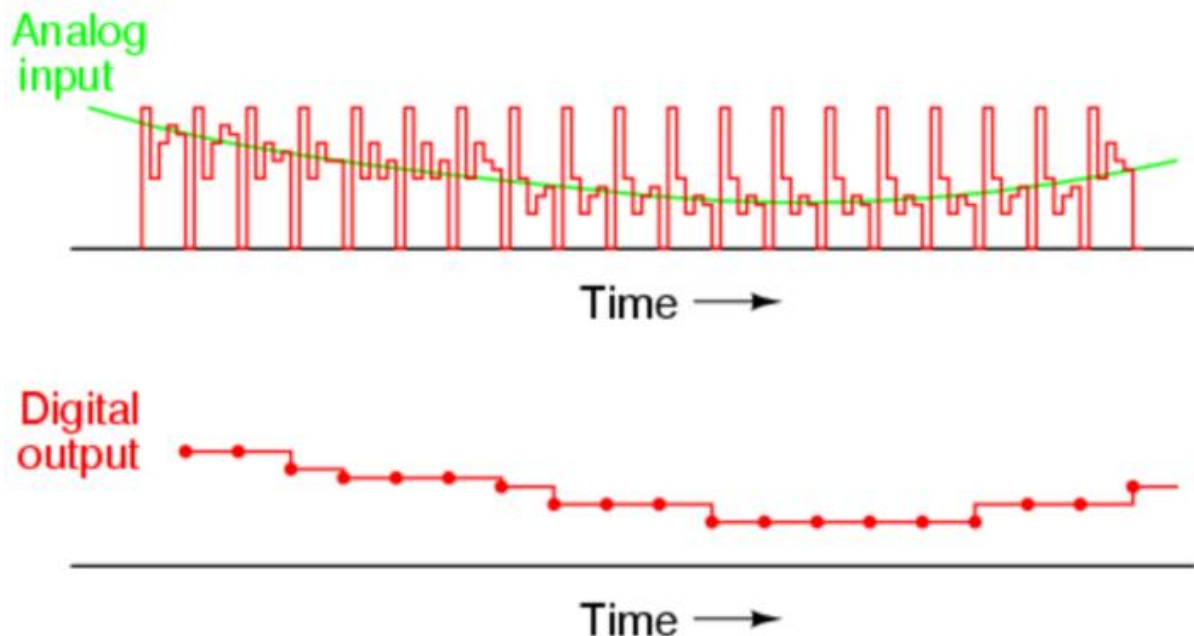
ADXL335 er et tre-akses analogt akselerometer<sup>[17]</sup>. Båndbredden til målingene reguleres med en kondensator ved akseutgangene. Denne kan settes fra 0.5 Hz til 1600 Hz for X og Y aksene, og 0.5 Hz til 550 Hz for Z aksen. Siden BOB-et ble brukt, ble det ikke rom for justering av denne



båndbredden, da kondensatoren på BOB-et på forhånd var satt for å gi en båndbredde på 50 Hz. Referert datablad ligger som vedlegg.

## 8.9 SAADC

nRF52832s innebygde ADC, er en *Successive Approximation Analog-to-Digital Converter* (SAADC) med tilhørende API<sup>[13]</sup>. Suksessiv approksimasjon er en metode for å omvandle analoge signaler til digitale. Dette gjøres ved å konvertere det analoge signalet til en diskret digital representasjon via et binærsøk gjennom alle de mulige kvantiseringsnivåene før den konvergerer til den digitale outputen. En fordel med suksessiv approksimasjon er at man kan ha et fast tidsintervall mellom hver konvertering.



Figur 8-3, Viser hvordan SAADC-en tilnærmer seg rett verdi vha. binærsøk.

SAADC brukes gjerne sammen med PPI-API-et, dette lar periferier samhandle selvstendig med hverandre uten å involvere prosessoren. Dette sparer strøm, ved å minimalisere tiden prosessoren er aktiv.

## 8.10 Litiumbatteri

Når det kommer til bærbar elektronikk er litiumbatterier, også kjent som litium-ionbatterier, blant de mest populære oppladbare batteritypene. Disse batteriene har meget høy energitetthet, noe som medfører høy kapasitet til tross for liten størrelse og lav vekt. Bruk av litiumbatterier setter derimot høye krav til at det benyttes spesielle ladekretser. Feillading, både over- og underlading, kan medføre enten brann eller reduksjon av batteriets kapasitet.

Litiumbatterier finnes i flere varianter. En av disse variantene er litium ion polymer, ofte forkortet til LiPo. Betegnelsen «polymer» betyr som oftest at batteriet er innkapslet i et mykt materiale, for eksempel en pose, noe som bidrar til å minke batteriets vekt og størrelse i forhold til andre batterityper.



Figur 8-4 Litium ion polymer-batteri

## 8.11 HTML/CSS & JavaScript

HTML, altså *HyperText Markup Language*, er et markeringsspråk for oppbygning av nettsider. Komplimentært til dette brukes som regel noe som kalles CSS, *Cascading Style Sheets*. Oppgaven til CSS er å definere hvordan HTML skal se ut og oppføre seg. CSS og HTML har to hovedmåter å kommunisere på, klasser og ID-er. Alle knapper på et nettstedet har en unik ID som kan refereres til både i CSS og JS. Under er et enkelt utdrag fra nettstedet, hvor en ID brukes på tvers av de tre språkene.

```
//HTML. Definerer en preformatert tekst med ID "log"
<pre id="log"></pre>

//CSS. Fjerner overflyt på alle HTML-tagger med ID "log"
#log {
    overflow: auto;
}

//JS. Sender et tekstinnhold til HTML-tagger med ID "log"
function log(text) {
    document.querySelector('#log').textContent += text + '\n';
}
//Javascript setter altså inn informasjon i HTML-filen.
```

## 9 Spesifikasjoner og egenskaper

- BLE-kube
  - microUSB for lading
  - 10-pins debug-inngang for programmering
  - Av-/på bryter
  - Batteri
    - Litium ion polymer-batteri
      - Én celle - 3,7 V, 2500 mAh<sup>[7]</sup>
    - Tilhørende regulator
      - Lastspenning 4,2 V<sup>[22]</sup>
  - Seks trykkflater med mulighet for fargeopplysning
    - Trykknapp - 320.01 E1-1 WHT
    - RGBL - LRTB GRTG
  - Vibrasjonsmotor
  - SoC
    - nRF52832
      - S132 SoftDevice v2.0.0<sup>[18]</sup>
  - Eddystone Beacon<sup>[5]</sup>
  - Akselerometer
    - LIS2DH12<sup>[9]</sup>
    - ADXL335<sup>[17]</sup>
  - Tre skiftregister
    - 8-bits - 74HC595<sup>[16]</sup>
- Nettsted
  - Til- og frakobling over BLE
  - Mottak av data over BLE
  - Sending av data over BLE
  - Tilfeldiggjøring av oppgave
  - Løsningsmekanisme
  - Navigasjonssystem
  - Informasjonsside

## 10 Prosess og gjennomført arbeid

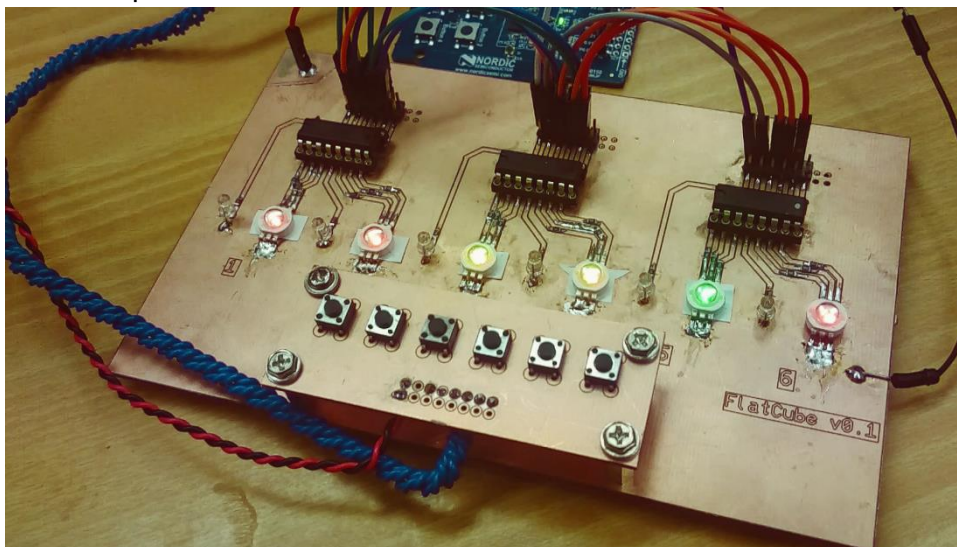
### 10.1 Innledning

I dette kapitlet skal hvert enkelt hovedfokusområde utbroderes nærmere. Det er laget en del for felles beslutninger som fikk innvirkning på hvert enkelt fokusområde. Hvert enkelt fokusområde har sin innledning, en beskrivelse av hva som er gjort, resultatet, og avsluttes med en diskusjon. Diverse testbrett ble benyttet underveis og vil bli belyst i dette kapitlet. Disse testbrettene ga oss muligheten til å ta avgjørelser fortløpende. Det hele avsluttes med felles resultat, diskusjon og konklusjon.

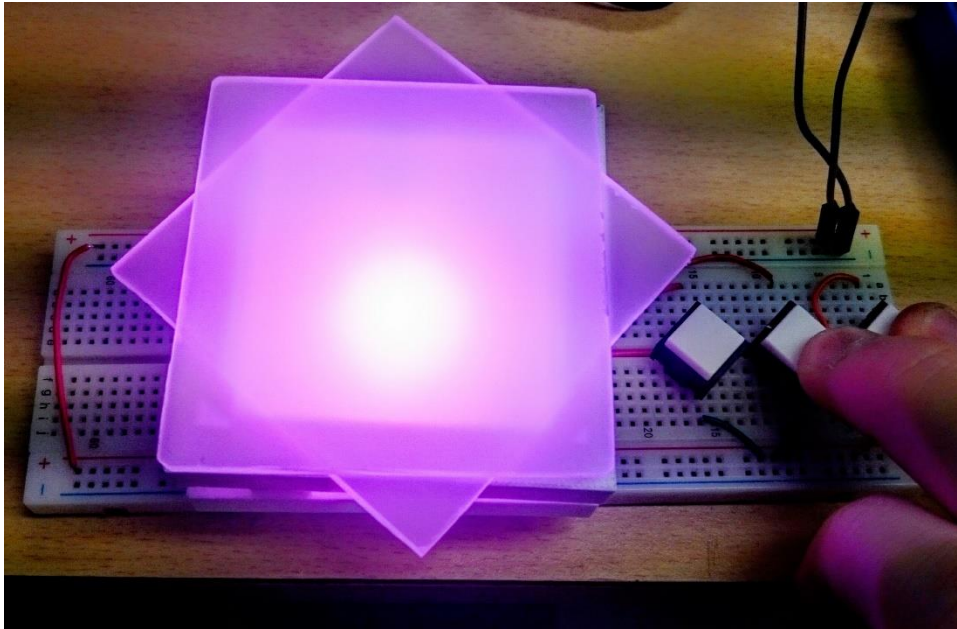
### 10.2 Testbrett og demokort

I løpet av dette prosjektet har det ved flere anledninger vært nødvendig å teste en idé eller oppkobling ved hjelp av diverse testbrett.

Gruppen fant det nødvendig å lage et testbrett for å se om LED-ene og trykknappene fungerte som det var tiltenkt. Dette testbrettet var nødvendig for å begynne arbeidet på programmeringsdelen så tidlig som mulig. Spesielt nyttig var dette for utvikling av løsningsmekanisme, funksjon for telling av knappetrykk, og generell funksjon på skiftregistrene. For å drive funksjonene ble nRF52 *development kit* fra NS brukt sammen med tre skiftregister og tolv LED-er. Selve skiftregistrene, LED-ene og trykknappene ble festet til et PCB som ble produsert på skolen. Videre ga det en mulighet til å teste flere LED-er og se hvordan de oppførte seg med forskjellige motstandsverdier, materiale de skulle lyse gjennom og avstander fra materialet. Demokortet hadde stor betydning for videre arbeid med programmering og utforskning av lysbrytning. Lysspredning, fargeblanding og motstandsveking ble utforsket nærmere på et dedikert testbrett.



Figur 10-1 Demokort



Figur 10-2 Testbrett for lysbrytning med korrekt motstandsvektning

Det ble også forsøkt å konstruere et testbrett for akselerometeret LIS2DH12, noe som viste seg å være utfordrende med tanke på at loddepunktene er  $0,25 \times 0,25$  mm store. Dette ble derfor ikke realisert. Det analoge akselerometeret ADXL335 var derimot tilgjengelig på et BOB og var kjapt og enkelt å benytte i et testbrett.

## 10.3 Kretsdesign [RE]

### 10.3.1 Innledning

Det ble tidlig besluttet at det skulle designes et skreddersydd kretskort for bruk i dette prosjektet. Kretskortet skulle være relativt enkelt, men likevel inneha mange funksjoner og egenskaper. De essensielle egenskapene innebærer evnen til å kommunisere med enheter i nærheten via BLE, kringkaste en URL, gi tilbakemeldinger i form av lyd eller vibrasjon, benytte tjuefire forskjellige LED-er og seks trykknapper på sidene, samt detektere når kuben er i bevegelse. Denne delen av rapporten vil forsøke å belyse fremgangsmetoden som ble brukt for å skape et kretskort som innfrir alle disse kravene og hvordan prosessen med å nå dette målet har gått for seg.

Digitale vedlegg inneholder den komplette prosjektmappen for alt utleggsrelatert som har blitt produsert i løpet av dette prosjektet. Altium bør benyttes for å åpne eller endre på disse filene.

### 10.3.2 Hva har blitt gjort?

#### 10.3.2.1 Anskaffelse av utviklingsverktøy

For å komme i gang med design av kretskort falt valget på Altium Designer som utviklingsmiljø. Dette førte til en del problemer med anskaffelse av lisens for å bruke programmet. Det ble nødvendig å dra innom skolens IT-avdeling, både på Kalvskinnet og Gløshaugen, mange ganger for å få hjelp med dette. Løsningen var å benytte seg av en VPN for å koble seg opp mot NTNUs lisensservere. Etter hvert som datasystemene til HiST og NTNU ble slått sammen fikk alle tidligere studenter ved HiST tilgang til lisensserverne til NTNU, noe som tillot bruk av Altium Designer. I tiden før dette skjedde måtte en prøvelisens benyttes.

Altium viste seg å være et særdeles omfattende verktøy med utallige valgmuligheter. De to første ukene i mars ble derfor i hovedsak benyttet til å anskaffe Altium og til å tilegne seg de essensielle grunnkunnskapene for å bruke programmet på en god nok måte. Instruksjonsvideoer ble også benyttet. Disse kan man få tilgang til ved å opprette en konto på Altiums nettsted<sup>[21]</sup>.



#### 10.3.2.2 Symboler, fotavtrykk og kretsskjema

Etter dette begynte arbeidet med å sette sammen et bibliotek med skjemasymboler og fotavtrykk for de komponentene som skulle inngå i designet. Utleggsfiler og skjema for utviklingskortet til NS var tilgjengelige på nett, noe som gjorde arbeidet med utlegget enklere<sup>[11]</sup>. Data for de fleste komponentene som skulle inngå i kretskortet for BLE-kuben var allerede tilgjengelige i filene for utviklingskortet. Skiftregister og akselerometer måtte bli konstruert separat.

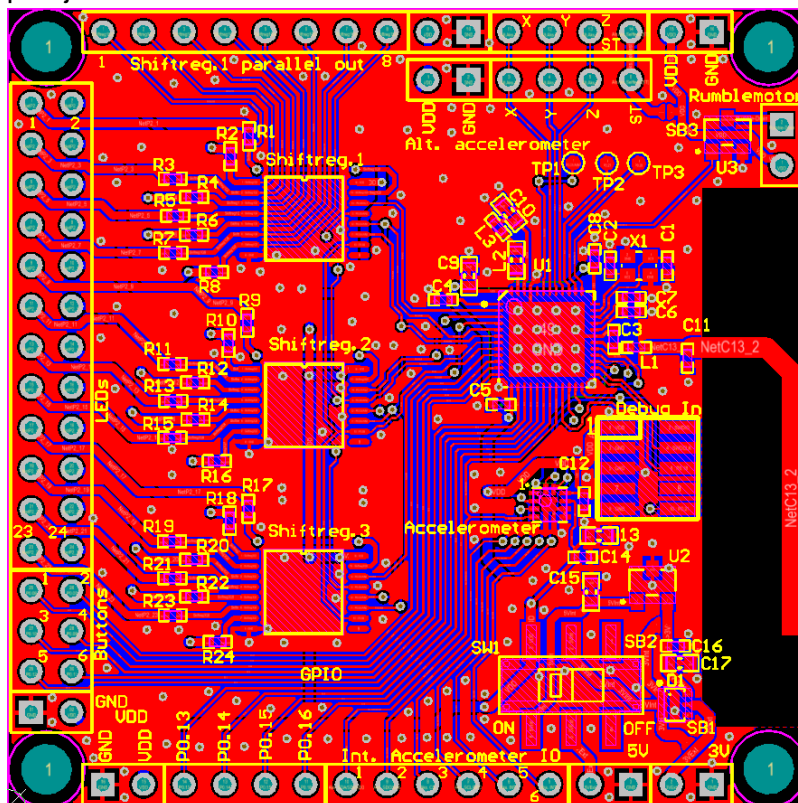
Med dette unnagjort kunne arbeidet med kretsskjemaet begynne. Nok en gang ble mye tid spart ved at referanseskjemaet fra NS kunne benyttes. Skjemaet er relativt enkelt og tok ikke lang tid å tegne opp. Skjemaet måtte senere oppdateres grunnet problemer med å forstå hvordan programmet knytter sammen skjema og utlegg. Skjemaet ble samtidig ryddet og gjort mer oversiktlig. Det endelige kretsskjemaet kan finnes i vedlegg 12.1.

Kretsen inneholder referansekretsen fra NS bestående av en nRF52 SoC, en 32MHz krystall, spoler og kondensatorer. Øvrige komponenter på kretsbrettet er tre skiftregistre for å drive tilkoblede LED-er, motstander, et akselerometer, en tilkobling for debug for programmering av kortet, en transistor for å drive en vibrasjonsmotor, en av/på bryter og en krets som vha. en lineærregulator legger til rette for bruk av forsyningspenninger på enten 3 V eller 5 V.

### 10.3.2.3 Utlegg

Som nevnt er Altium et komplisert verktøy med høye kompetansekrav for å kunne bruke det effektivt. Dette førte til at arbeidet gikk litt tregt til å begynne med. Rune Brandsegg fra NS er godt kjent med Altium og var en god ressurs å ha tilgjengelig. Med noen små tips så ble arbeidet med utlegget mye enklere. Med et akseptabelt kretsskjema kan det opprettes en utleggsfil, altså et PCBDoc. Utleggsdelen av programmet benytter skjemafilen til å tegne opp nyttige hjelpestreker mellom punkter i kretsen som skal kobles sammen. Altium er svært regelbasert, noe som vil si at avstand mellom ledningsbaner og jordplan, tykkelsen på ledningsbaner, klaring mellom komponenter, viahullstørrelser og lignende bør stilles inn før man begynner konstruksjon av utlegget. Hvis disse på forhånd er konfigurert så blir disse reglene automatisk overholdt under konstruksjon av utlegget. Hvis man ikke er kjent med programmet fra før er det ikke alltid godt å vite hvordan alle disse innstillingene burde konfigureres. På en annen side kan dette være en stor fordel da design av utlegg går veldig kjapt hvis disse er innstilt på forhånd.

Utlegsfilen for utviklingsbrettet var en viktig kilde under arbeidet med utlegget<sup>[11]</sup>. Dette designet ble studert nøye, og ved å benytte Altium ble det mulig å spore ledningsbaner og sammenkoblede komponenter på en god måte. Dette ga god innsikt i hvordan man kan implementere nRF52 SoC og tilhørende hardware. Ettersom utviklingskortet hadde blitt benyttet i stor grad i tiden før gruppens egne utlegg ble designet, virket det som et godt valg å følge de samme designvalgene. Utlegsfilen for utviklingsbrettet ble derfor blant annet brukt som utgangspunkt for innstilling av de fleste av de overnevnte reglene. Utleget for utviklingsbrettet inneholdt også en forsyningskrets med en lineærregulator, noe som gir kortet støtte for både 3V fra knappenålsbatteri, og for 5 V fra en USB-tilkobling. Kretsen inneholdt også en diodebro for å hindre skader på kortet dersom spenning med feil polaritet skulle bli påtrykt. Dette ble ansett som nyttige egenskaper, derfor ble denne kretsen tilpasset og anvendt i utlegget for dette prosjektet.



Figur 10-3 Utlegg i Altium



Viktige fokusområder under utleggsarbeidet:

- Holde utlegget mindre enn, eller nøyaktig 5\*5cm
- Overholde referanseutlegget
- Skruehull i hvert hjørne
- Godt jordplan, mange viahull
- Komponentplassering med hensyn på montering og lodding
- Teksting og nummerering
- GPIO tilgjengelig ved kantene
- Korte ledningsbaner

NS har mange ressurser tilgjengelig for å bistå utvikling av hardware på deres infocenter<sup>[10]</sup>, og i den sammenheng stilles det krav til at deres referanseutlegg blir implementert uten endringer. Dette utlegget er kvalitetssikret, og direkte implementasjon av dette utlegget er et god steg på veien mot god RF-ytelse. Bakgrunnen for dette kravet er ønsket om at deres teknologi skal være så enkel som mulig å anvende. Det ble også benyttet en guide med retningslinjer for kretsdesign<sup>[20]</sup>.

Det ble lagt stort fokus på nødvendigheten av å ha et godt jordplan på kortet. I den forbindelse ble utlegget populært med store mengder viahull. Dette er et godt tiltak for å unngå at deler av jordplanen fungerer som antenner og forstyrrer radiokommunikasjonen. RB hjalp til med kvalitetssikring av utlegget etter at det var ferdigdesignet og klart til å bestilles.

#### 10.3.2.4 Komponentlist og lodding

Med kretskortene bestilt måtte komponentlisten skrives og verifiseres. Å bestille riktige komponenter har stor betydning; de må passe med fotavtrykkene på kretsbrettet og de må ha riktige parametere i henhold til bruksområde. Det tok lang tid å dobbeltsjekke at alle komponentene stemte. Med unntak av klokkekrytallen, som ble bestilt fra Farnell, og LEDs, som ble bestilt fra RS ble alle komponentene bestilt fra samme varehus, Digi-Key. Komponentlisten er å finne i vedlegg 12.4.

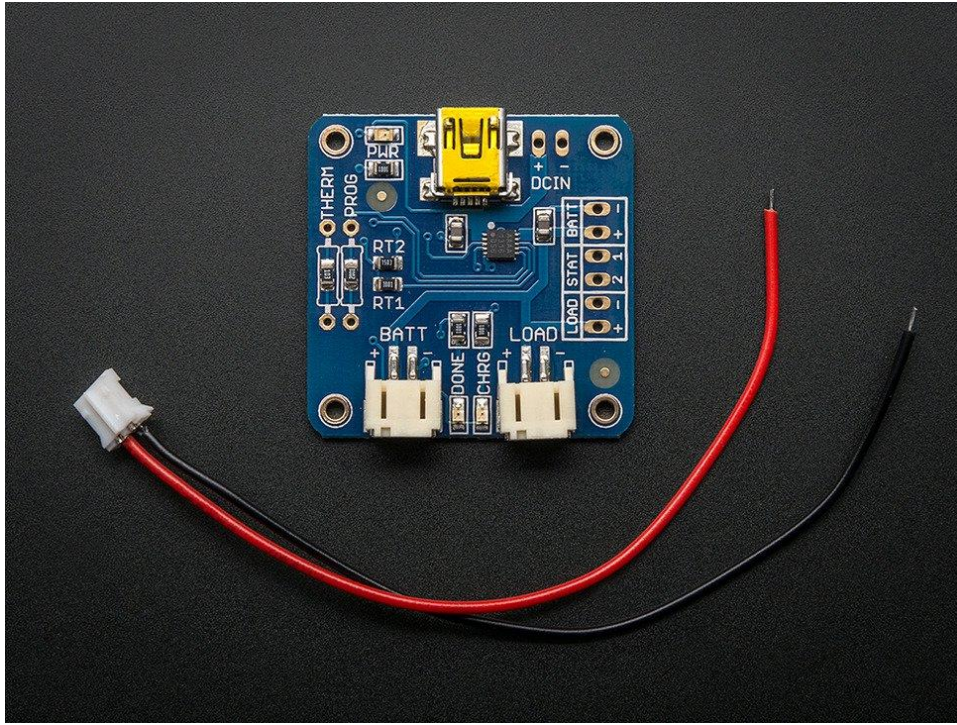
Så snart komponentene ankom kunne lodding påbegynnes. Dette ble gjennomført på labben hos NS. RB bistod med innføring i nyttige loddeteknikker. Lodding av så små komponenter var utfordrende, men takket være godt utstyr ble resultatet akseptabelt. To komplette kretsbrett ble loddet ferdig. Det ble loddet to kort for å øke sannsynligheten for at minst ett kort kom til å fungere.

#### 10.3.2.5 Testing av produsert kretsbrett

Elementære tester ble utført for å sikre at det ikke var noen katastrofale kortslutninger. Deretter ble spenning påtrykt, både 3V og 5V. Regulatoren fungerte som tenkt og nedregulerte 5V til 3,2V og kortet fungerte med begge forsyningsspenningene. Det ble oppdaget en kortslutning på ett av kortene ved klokkekrytallen. Dette er en komponent med fire tilkoblingspunkter og måtte fikses med en varmluftspistol. Øvrige loddepunkter ble også varmet opp samtidig for å få mer sentrerte komponenter med bedre tilkobling. Dette førte til to fungerende kretskort som var i stand til å kjøre software. Videre informasjon om hvordan denne testprosessen ble gjennomført er å finne under kapittelet om software, kapittel 10.4.2.4 – Testing av produsert kretsbrett, side 45.

### 10.3.2.6 Ladekrets og adapterbrett

Grunnet tidspress ble det besluttet å ikke designe en ladekrets til bruk med et Lithium-Ion batteri fra bunnen av på egen hånd. Dette førte til beslutningen om å bestille en ferdigprodusert batteriløsning. Valget falt på laderegulatoren *MCP73833 lithium ion polymer charger*<sup>[22]</sup>, og et litium ion polymerbatteri<sup>[7]</sup> fra Adafruit.



Figur 10-4 Laderegulator

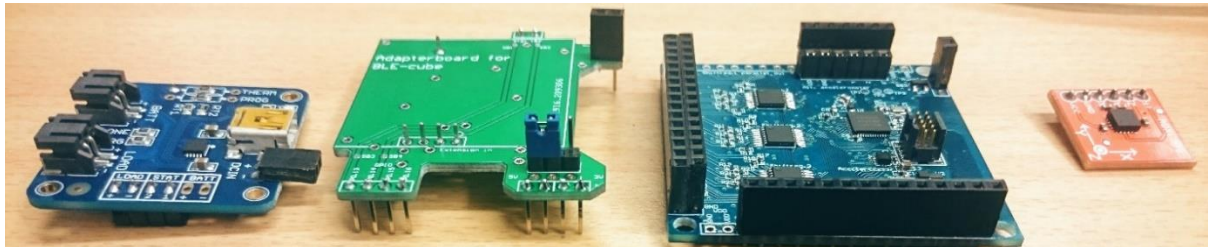
Denne regulatoren er designet for å fungere med alle litiumbatteriene Adafruit tilbyr. For å være på den sikre siden ble det bestilt et batteri med så høy kapasitet som mulig, 2500 mAh, uten at batteriets størrelse overskred plassbegrensningene innad i kuben.

Det ville også være nødvendig at denne ladekretsen skulle kunne kobles til hovedkretskortet og fremdeles inneha god mekanisk stabilitet. Dette fordi kubens vil måtte kunne tåle en del fysisk påkjenning. Derfor ble det designet et tilpasningsbrett hvor man kan koble til både kubens hovedkort og ladekrets. Dette ble frest ut på skolens kretskortfres for å verifisere designet, men ble også innsendt til NS for bestilling ettersom produsenten de benytter seg av gir meget gode resultater.

Dette tilpasningsbrettet bringer med seg lastspenningen 4,2 V fra laderegulatoren og tilfører den til hovedkortets inngang for 5 V. Denne inngangen leder til en regulator som regulerer ned inngangspenningen med ca. 1 V. Dette er nødvendig ettersom 4,2 V overskrider maksimal driverspenning for nRF52-SoCen og kan derfor ikke kobles til inngangen for 3 V. I følge datablad<sup>[13]</sup>, er maksimal driverspenning for nRF52 oppgitt til å være 3,6 V.

### 10.3.3 Resultat

Resultatet av denne prosessen er to kretskort som er i stand til å kjøre kode og fungerer godt. Et batteri med tilhørende ladekrets ble også anskaffet og dette fungerer godt sammen med det produserte hovedkortet. Altså ble til og med kompromisser, som valget med å utelate ladekrets, løst på en eller annen måte og alle mål som ble satt før arbeidet med kretskortet begynte ble til slutt oppfylt. Kretskortet fungerer godt med all tilkoblet hardware og innehar alle egenskapene som opprinnelig var tiltenkt.

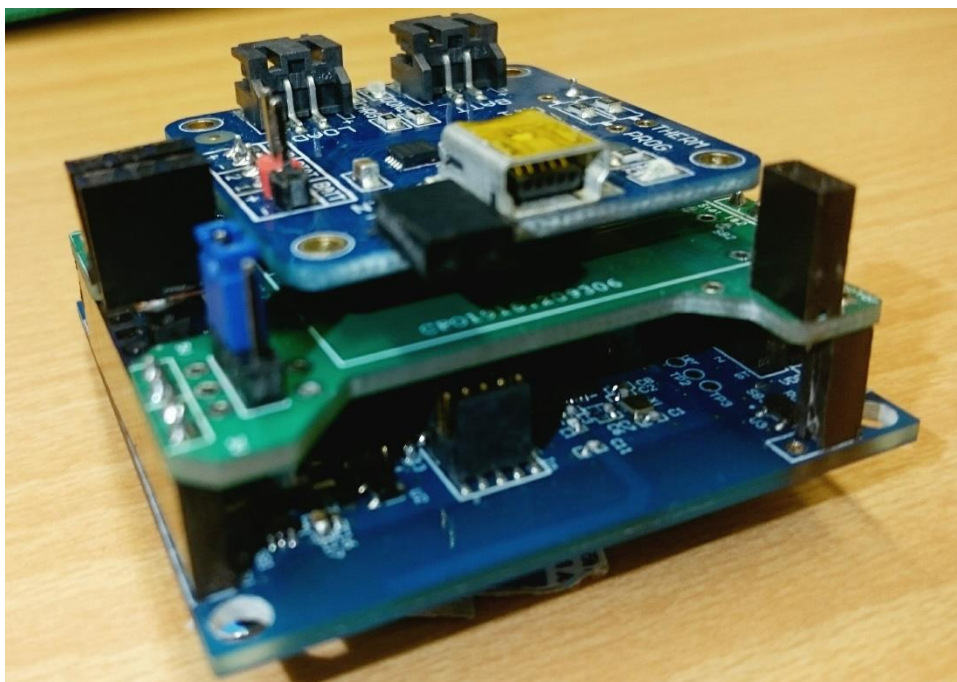


Figur 10-5 All benyttet HW

Ved å legge til grunn for videreutvikling i form av GPIO-er på kretsbrettets sider ble også målet om å tillate videreutvikling oppnådd.

Dokumentasjon og vedlegg som har blitt produsert i løpet av arbeidet skal være av god nok kvalitet til at reproduksjon eller videreutvikling av arbeidet kan gjennomføres.

Den ansvarlige for dette arbeidet har i tillegg fått tilegnet seg verdifull kunnskap om kretsdesign, komponentvalg, Altium Designer, loddeteknikk for SMD, samt generell innsikt i prosessen ved å skape egen hardware.



Figur 10-6 Komplet kretskort med ladekrets og tilpasningsbrett



## 10.3.4 Diskusjon

### 10.3.4.1 Anskaffelse av utviklingsmiljø

Det viste seg at Altium var et godt valg til tross for at programmet var utfordrende å lære seg. Mye tid ble derimot tapt under prosessen ved å skaffe tilgang til programmet.

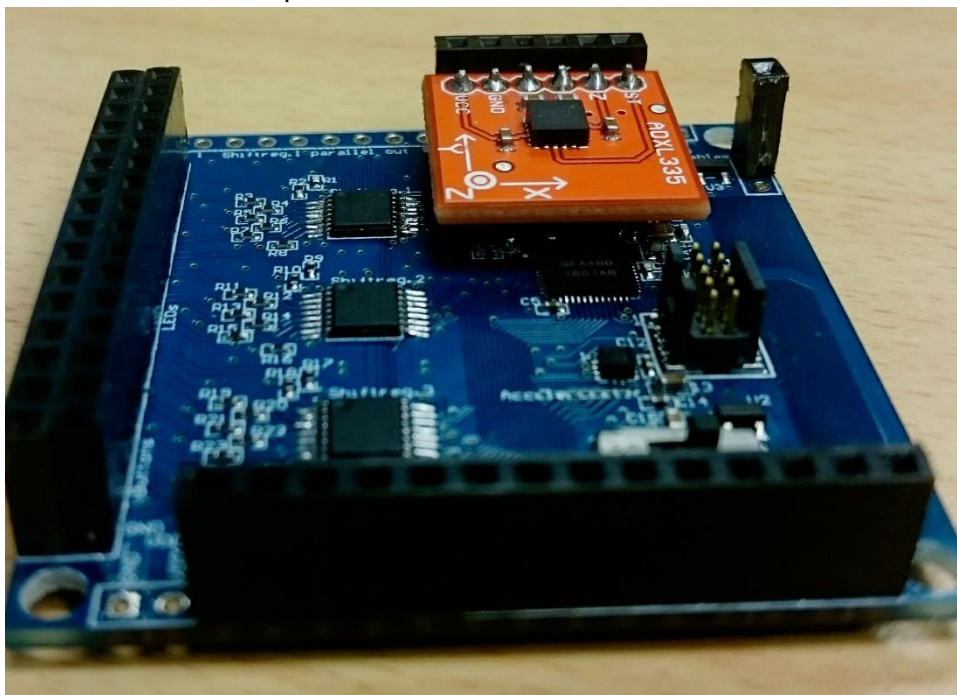
### 10.3.4.2 Symboler, fotavtrykk og kretsskjema

Utfordringene ved denne delen av arbeidet kommer i stor grad fra mangel på erfaring ved bruk av programmet. Altium benytter seg av biblioteker for komponentsymboler- og fotavtrykk. Disse kan enten lages fra grunnen av selv, hentes inn eksternt, eller genereres fra andre kretsdesign. Dette gjelder også for de faktiske symbolene og fotavtrykkene disse bibliotekene består av. Dette var i begynnelsen forvirrende og en kombinasjon av alle disse fremgangsmåtene ble benyttet for å konstruere kretsskjemaet og utlegget.

Det er vanlig å dele opp et kretsdesign i separate skjema. Dette ble vurdert, men til sist ansett som overflødig.

### 10.3.4.3 Utlegg

Stort fokus ble satt på tilrettelegging av potensielle situasjoner og eventualiteter senere i prosjektarbeidet. Eksempelvis ble forsyningskretsen implementert med hensyn på at valget om hvilken spenningskilde som skulle brukes enda ikke var tatt. Når valget senere falt på en bedre batteriløsning med høyere forsyningspenning viste dette seg å være et godt valg. Hovedkortet vårt er utstyrt med mange tilkoblingspunkter på tre av de fire sidene. Dette tillater tilkobling av annen hardware eller diverse utvidelsesbrett som kan inneholde nesten hva som helst. I prototypen benyttes nettopp denne egenskapen for tilpasningsbrettet og ladekretsen. Ettersom kuben har en høyde på fem centimeter er det derfor i teorien plass til fem forskjellige kort ved bruk av korte header-pinner.



Figur 10-7 Kretskortets tilkobling for akselerometer

Kretsen er også utstyrt med tilkoblinger for et alternativt akselerometer grunnet at det tiltenkte akselerometeret ikke var mulig å teste før kretskortet var ferdigprodusert og ferdigloddet. Ved å tilrettelegge for et akselerometer som allerede var tilgjengelig og testbart kunne denne delen av prosjektet påbegynnes mye tidligere. Tilkoblingen for dette akselerometeret ble plassert innenfor den primære raden med tilkoblingspunkter. Slik kan enten akselerometeret fysisk plasseres på eventuelle utvidelsesbrett, eller tillate annen hardware på eventuelle utvidelsesbrett å benytte signalene fra akselerometeret til andre anvendelser. Dersom dette alternative akselerometeret ikke benyttes fungerer disse tilkoblingene som vanlige GPIO-er. Den samme utleggsfilosofien gjelder også for det innebygde akselerometeret, LIS2DH12.

Med header-pinner loddet på blir det ganske dårlig med plass til skruene i hjørnene. Ved å ha sløffet noen av tilkoblingene og omorganisert plasseringene ville dette vært mulig å unngå.

De forskjellige diodene inne i en RGBL fungerer optimalt på forskjellige spenninger, noe som medfører at man egentlig burde benytte forskjellige motstandsverdier for de forskjellige fargene. Å ta hensyn til dette ble nedprioritert etter som det tross alt er snakk om en prototype, men dette kunne ha ført til mer definerte tertiærfarger. Motstandsverdiene som benyttes er 390 Ohm og fungerer godt med de LED-ene som ble benyttet til slutt.

Plassering av komponenter med tanke på å forenkle loddningen ble gjennomtenkt, men i området rundt akselerometeret og debug tilkoblingen ble det likevel en anelse trangt.

Teksten og merkingen kunne også ha vært gjennomført på en bedre måte. Etter som *header*-ene ble loddet på ble det ikke enkelt å differensiere jordtilkoblinger fra forsyningstilkoblinger uten å vri og vende på kortet.

I retningslinjene for PCB-design<sup>[16]</sup> blir det nevnt at *tuning*-kondensatoren på antennen burde dimensjoneres spesifikt for kortet. Ettersom det viste seg at den trådløse kommunikasjon fungerte greit nok med en *tuning*-kondensator på 1 pF ble det ansett som unødvendig å vie mer tid til dette.

#### 10.3.4.4 Komponentliste og loddning

Altium har i utgangspunktet støtte for å generere BOM, Bill Of Material, men det ble lettere å skape denne på egenhånd ettersom varehus, pakke type og lignende kun kan stilles inn i hver enkelt komponents egenskaper. Dette ville vært tidkrevende. I tillegg inneholder dette prosjektet andre komponenter og deler som bør stå i en overordnet komponentliste, men som ikke er direkte knyttet opp mot kretskortet. LED-er og knapper er gode eksempler på slike komponenter. Loddningen, som ble gjennomført av RE og EJC, gikk bra, men kunne vært enda enklere om plasseringen av komponentene hadde vært annerledes. Plassering av komponenter, med tanke på å forenkle loddningen, ble gjennomtenkt under arbeidet med utlegget, men i området rundt akselerometeret og debug tilkoblingen ble det likevel en anelse trangt.

#### 10.3.4.5 Testing av produsert krets Brett

De fleste problemene som ble åpenbare under testing stammet hovedsakelig fra loddningen. Kortslutninger ble heldigvis oppdaget og utbedret før kortet ble tilført spenning.

#### 10.3.4.6 Ladekrets og adapterbrett

Ved beslutningen om å kjøpe inn et LiPo-batteri<sup>[7]</sup> med tilhørende ladekrets<sup>[22]</sup> fulgte også behovet for et tilpasningsbrett for å bruke disse nye tilskuddene med det produserte kretskortet. Dette tilpasningsbrettet ble også benyttet til å tilføre 4,2 V direkte til vibrasjonsmotoren, noe som gir en mer definert ristelyd i motsetning til å drive den med 3 V.

Laderegulatoren har også to tilkoblingspunkter for indikering av ladestatus. Disse tilkoblingene ble dirigert gjennom tilpasningsbrettet ned til GPIO på hovedkortet. Dette vil tillate SoC-en å lese av om batteriet behøver lading, blir ladet eller er fulladet. Dette kan deretter benyttes til å programmere kuben til å indikere disse forskjellige situasjonene på sin egen måte og med sine egne LED-er.

## 10.4 Software [EJC & ØS]

### 10.4.1 Innledning [EJC]

Denne delen av prosjektet består av programvare og nettsted. Med programvare menes all programkoden som skal lastes opp på nRF52 på vårt kretskort. Nettstedet er stedet hvor man kobler seg på kuben og får oppgaver å løse. Programvaren er skrevet i C og nettstedet er en kombinasjon av HTML, CSS og JS. HTML og CSS står for design og form, mens JS brukes for BLE-kobling, generering av oppgave, fremvisning av nettsted og håndtering av alle knapper. Nettstedet og programvaren skal kunne kommunisere begge veier. Eksempelvis skal kubens kunne fortelle nettsted hvilke farger som lyser hvor, og nettstedet skal kunne plassere kubens i demomodus.

Det er særdeles viktig at programvaren og nettstedet kommuniserer på en god måte. Dette er løst ved å tydelig definere en datamatrix, hvor informasjon sendes. Utdypet informasjon om dette finnes under kapittel 10.4.2.3 Samspill - C og JavaScript, side 39.

Kommentarer i kodesnutter som er inkludert i rapporten vil noteres med følgende syntaks:

```
//Kommentar angående neste kodelinje skrives her. x settes lik 2
uint8_t x = 2;
```

### 10.4.2 Hva har blitt gjort

#### 10.4.2.1 Firmware – C [ØS & EJC]

Store deler av prosjektets startfase ble brukt på å lære seg systemet til NS. Dette innebærer deres tilgjengelige programkode og API, programvaren som brukes, og generelt om BLE. Det er tatt utgangspunkt i prosjektet «ble\_app\_uart». Dette eksempelet finnes i SDK.

For god kommunikasjon mellom to enheter har vi konfigurert en egen GATT-profil. Denne består av en primærtjener og to karakteristikk.

Dette gjøres i filen ble\_nus.c og ble\_nus.h, som finnes i «ble\_app\_uart». Under vises et utdrag av nettopp dette:

```
ble_nus.c
//The UUID of the TX Characteristic
#define BLE_UUID_NUS_TX_CHARACTERISTIC 0x2000

//The UUID of the RX Characteristic
#define BLE_UUID_NUS_RX_CHARACTERISTIC 0x2001

ble_nus.h
//The UUID of the Nordic UART Service
#define BLE_UUID_NUS_SERVICE 0x1338
```

#### 10.4.2.1.1 Skiftregisterbibliotek [EJC]

For å styre tjuefire LED-er på en effektiv måte, uten å sløse GPIO-er, har vi valgt en løsning som innebærer tre skiftregister. Demoversjonen bestod av fire LED-er på hver side av kuben, som alle ble styrt separat. Hvert skiftregister har en oppløsning på åtte bit, og går utmerket opp med antall LED-er vi ønsker å styre. Dette valgte vi å skrive et dedikert bibliotek for, slik at LED-er enkelt kan styres når ønskelig. For å utvikle dette biblioteket ble det brukt elementer fra en guide<sup>[23]</sup>. Biblioteket består av fargedefinisjoner og fire funksjoner:

```
#define redA      8
#define greenA    4
#define blueA     2
#define whiteA    1
#define redB    128
#define greenB   64
#define blueB   32
#define whiteB   16
```

Hvert skiftregister får altså en åttebits matrise hvor de fire minst signifikante bits går til en side, og de fire mest signifikante bits går til en annen. Bitene i masken er definert slik:

RedA	GreenA	BlueA	WhiteA	RedB	GreenB	BlueB	WhiteB	regN
0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	Mulig verdi

Under vises et par eksempler for å klargjøre virkemåten:

$$\text{Eksempel A: } redA + redB = 128 + 8 = 136 = 1000\ 1000$$

$$\text{Eksempel B: } redA + blueA + greenB + blueB = 8 + 2 + 64 + 32 = 106 = 0110\ 1010$$

Eksempel A resulterer i at de to sidene som er tilkopleet dette skiftregisteret, vil lyse rødt. Eksempel B resulterer i at en side lyser lilla og den andre cyan.

Dette systemet ble utviklet for demobrettet, som hadde egen LED for hvit farge, noe som ikke befinner seg på kuben. For å få hvit farge kreves det at alle de tre RGBL-ene er påskrudd. Dette resulterer i følgende fargedefinisjoner:

```
#define redA      8
#define greenA    4
#define blueA     2
#define whiteA    14
#define redB    128
#define greenB   64
#define blueB   32
#define whiteB   224
```



Funksjonene som inngår i biblioteket er følgende:

- {1} `shiftRegInit()`
- {2} `shiftRegPulse()`
- {3} `shiftRegLatch()`
- {4} `shiftRegWrite()`

Funksjon {4} og {1} er de eneste høynivåfunksjonen av disse fire. Det er de eneste som kalles fra programløkka. De resterende funksjonene brukes av {4}.

{1} initialiserer skiftregistrene, dette innebærer bestemmelse av retning på alle GPIO som brukes av registrene. I tillegg setter {1} OE lav og SRCLR høy. OE står for *Output Enable* og avgjør om skiftregistrenes utganger får være aktiv, denne vil vi altså alltid ha lav. Det er fint å merke seg at OE vises som invertert på skjemategningen, og er grunnen til at vi ønsker lav verdi. SRCLR står for Shift Register Clear. Denne inngangen lar oss klarere lagret data på skiftregisterene. Denne settes derfor høy i initialiseringen.

{2} sender en puls til SRCLK, *Shift Register Clock*. Denne håndterer bitvis seriell input til skiftregistrene. Denne kalles mellom hvert venstreskift.

{3} skifter data inn i skiftregistrene ved puls på RCLK, *Storage Clock*. Denne kalles etter alle åtte bit har blitt sendt inn i skiftregistrene.

{4} er funksjonen som gjør den faktiske jobben. Denne funksjonen får innsendt tre argumenter, respektivt de seks tellerverdiene på knappene våre.

Det neste som ble utviklet var systemet for å bruke knappetrykk for å bla gjennom fargene. Både kube og nettsted trenger informasjon om hvilken farge som vises på hver side. Dette er synonymt med hvor mange trykk som har inntruffet. For å kunne relatere antall trykk til en gitt farge, ble det definert en fargematrise:

```
//Fargematrise for side A. ledMatriseA[3] = redA + greenA.  
uint8_t ledMatriseA[] = {whiteA, redA, greenA, redA+greenA,  
blueA+greenA, blueA+redA, blueA};
```

Metoden for å telle tastetrykk ser slik ut:

```
//Fra shiftLib.c. Funksjonen mottar hvilken knapp som har blitt
trykket, og antall ganger denne knappen har blitt trykket.
uint8_t btnCounter(uint8_t btn, uint8_t btncounter) {

//Ved registrert trykk sitter programkoden fast her.
    while(nrf_gpio_pin_read(btn));

//Ved fallende flanke øker vi btncounter med én, og dersom verdien er
større enn antall fargedefinisjoner, tilbakestilles verdien.
    btncounter++;
    if(btncounter > 6){
        btncounter = 1;
    }

//Antall trykk returneres
    return btncounter;
}

//Fra main.c. Her lyttes det etter knappetrykk
if(nrf_gpio_pin_read(btn1)) {

//Her lagrer vi den returnerte verdien, altså antall trykk. Samtidig
sender vi inn informasjon om hvilken knapp.
    btn1_counter = btnCounter(btn1, btn1_counter);

//Her lagres tellerverdi i løsningsmatrisen
    solution[0] = btn1_counter;
}
```

Dette gir muligheten til å bruke {4} på følgende vis:

```
shiftRegWrite(ledMatriseA[btn1_counter],...);
```

Det bør merkes her at når bruker holder en knapp inne, sitter programløkken fast i en *while*-løkke som ikke har noen funksjon. Det er på fallende flanke at tastetrykket blir registrert. Grunnen til dette er at tastetelleren ikke skal stige ukontrollert.

I tillegg kan det være fint å merke seg at `btn1_counter` lagres i `solution[0]`. Dette gjøres for å lagre alle seks knappeverdier i en løsningsmatrise. Denne vil være identisk med nettstedets løsningsmatrise, dersom bruker har valgt riktige fargekombinasjoner.

NS har innebygd et system for håndtering av GPIO-er, eksempelvis Arduinopinner i forhold til *Development Kit-et*. Dette systemet har blitt tatt i bruk og det har blitt skapt en egen *header*-fil for vårt kort. Denne filen har vi valgt å kalle *ble\_card*. Her defineres pinnene til skiftregistre, knappene og krystallvalg.

#### 10.4.2.1.2 Akselerometer [ØS]

Tidlig i prosessen ble det uttrykt et ønske om å ha et innebygd akselerometer i kuben. Dette skulle ha i oppgave å detektere ristebevegelse som inndata. Samtidig var det et ønske om å bruke et akselerometer for å detektere om kuben hadde vært liggende i ro over en lengre periode, slik at den kunne gå i dvalemodus. For så å våkne når kuben igjen fikk bevegelse. Hensikten med dette skulle være å spare strøm, da kuben drives av et internt batteri.

Valget av akselerometer havnet til slutt på ADXL335. Dette er et analogt akselerometer, man trenger derfor en ADC for å kunne lese måleverdiene. nRF52832 har en innebygget SAADC, valget for sampling av de analoge signalene falt da naturlig på denne løsningen, siden man da slipper noen ekstra hardware. Man kan også benytte seg av forhåndsdefiniserte API.

Først må man kalle:

```
void saadc_sampling_event_init(void);
```

Dette er en funksjon som initialiserer når SAADC skal hente samples. Funksjonen setter opp `m_timer` til å lage et sammenligningshendelse hvert `N` millisekund, definert av

```
#define ADC_COMPARE_RATE 100 //Compare ADC event every N milisec
```

Den setter også opp PPI-kanalen, sånn at sammenligningsevent fra `m_timer` triggerer sampling i SAADC.

Når man har definert hvilke *events* som skal triggere sampling, vil man initialisere SAADC-en.

Dette gjøres ved å kalle:

```
void saadc_init(void); //Funksjon for å initialisere SAADC
```

Denne funksjonen initialiserer SAADC-en, og den sier hvilke AI-porter(Analog Input) det skal lyttes til.

```
NRF_SAADC_INPUT_AIN6 //Her leses X-aksen fra ADXL335
NRF_SAADC_INPUT_AIN5 //Her leses Y-aksen fra ADXL335
NRF_SAADC_INPUT_AIN4 //Her leses Z-aksen fra ADXL335
```

Funksjonen benytter seg av forhåndsdeklarte funksjoner fra SAADC API-et, disse finnes i `nrf_drv_saadc.h`. Først kalles `nrf_drv_saadc_init(..)` denne initialiserer SAADC i henhold til en konfigurasjons-*struct* som brukes som argument. Dersom det ikke blir oppgitt noen konfigurasjons-*struct*, blir en standard *struct* brukt. `nrf_drv_saadc_init(..)` bruker også `saadc_event_handler()` som argument, denne vil hente ADC-resultatene via ADC-interrupts. Disse resultatene sammenlignes for å se etter endring i akselerasjon. Dersom endring registreres, inkrementeres en teller. Har man hatt nok akselerasjonforandring innenfor et tidsrom, vil bevegelsene tolkes som et rist. Når `nrf_drv_saadc_init(..)` er ferdigkjørt kalles `nrf_drv_saadc_channel_init(..)`. Denne funksjonen konfigurerer en SAADC-kanal basert på hvilke AIN-porter man setter som *input*. Tilslutt i `saadc_init(..)` kjører man `nrf_drv_saadc_buffer_convert(..)`, denne funksjonen utsteder omdannelse av data til

en buffer-matrise den får som argument. Funksjonen er ikkeblokkerende, og applikasjonen får beskjed om at bufferen skal fylles av *event*-håndtereren.

Til slutt må man kalle en funksjon for å aktivere SAADC-en sin PPI-kanal.

```
void saadc_sampling_event_enable(void); //aktiverer SAADC sin PPI-kanal.
```

Når dette er satt opp, vil man sjekke akselerometerdata hvert N-te millisekund, som definert i ADC\_COMPARE\_RATE. Disse dataene legges i en buffer, hvor dataen sjekkes for å se etter endringer i akselerasjonen, slik at rist blir detektert.

#### 10.4.2.1.3 Eddystone Beacon [ØS]

Når kuben er aktiv, er det tiltent at den skal kringkaste en URL som kan plukkes opp av telefoner med *Physical Web*. På denne måten får kuben informert nærliggende enheter om at den finnes, og den får sendt bruker til netspillet dersom de velger følge URL-en. Dette er spesielt ønskelig dersom NS velger å bruke vårt bachelorprosjekt til demonstrasjon. Da vil det være lett for brukere å koble seg til kuben via den kringkastede URL-en som sender dem til netspillet.

For å kunne veksle mellom å bruke radioen til å være kringkaster og å være et periferi som skal kunne tilkobles mobiltelefoner, ble det nødvendig med en funksjon som bytter mellom disse rollene med gitte tidsintervall. Her vil man at periferirollen skal ha prioritet dersom begge vil ha tilgang til radioen samtidig.

Løsningen ble å bruke `timeslot_init(void)` sammen med `advertiser_beacon_timeslot` API-et. Dette er forklart i detalj under, med kodereferanse og kommentarer.

```
static uint32_t timeslot_init(void)
{
    static ble_beacon_init_t beacon_init;

    //Setter intervall for kringkasting, gitt i ms.
    beacon_init.adv_interval = 360;

    //Peker til URL datamatrise.
    beacon_init.p_data = (uint8_t *) eddystone_url_data;

    //Informasjon kringkastet av Eddystone-URL
    static uint8_t eddystone_url_data[] =
    {
        APP_EDDYSTONE_URL_SCHEME,

        #define APP_EDDYSTONE_URL_SCHEME 0x03
        // 0x03 = "https://" URL prefiksordning i henhold til
        // Eddystone spesifikasjoner

        APP_EDDYSTONE_URL_URL
    }
```

```

#define APP_EDDYSTONE_URL_URL
                                0x67, 0x6f, 0x6f, 0x2e, \
                                0x67, 0x6c, 0x2f, 0x35, \
                                0x70, 0x6b, 0x73, 0x48, \
                                0x52

// URL with a maximum length of 17 bytes. Last byte is
// suffix (".com", ".org", etc.)

};

//Antall bytes i url datamatriksen.
beacon_init.data_size      = sizeof(eddystone_url_data);
beacon_init.error_handler = beacon_advertiser_error_handler;
//Henter ble-cube sin egne Bluetooth adresse
uint32_t err_code =
sd_ble_gap_address_get(&beacon_init.beacon_addr);
if (err_code != NRF_SUCCESS)
{
    return err_code;
}
//Inkrementerer adressen slik at vanlig kringkasting, og eddystone
har ulik adresse.
beacon_init.beacon_addr.addr[0]++;

//Denne funksjonen bruker dataen i beacon_init legger dem inn i
m_beacon structen
app_beacon_init(&beacon_init);

m beacon er en static struct definert i
advertiser_beacon_timeslot.c.

//Denne funksjonen starter beacon-en, da ved hjelp av variabler satt
i m_beacon struct-en. Disse sendes til SoftDevicet. Først åpnes en økt
for å kunne forespørre radio-timeslot, dernest spør man om en radio
timeslot å kringkaste på.
app_beacon_start();

return NRF_SUCCESS;
}

```

#### 10.4.2.1.4 Testing av produsert kretsbrett [EJC, ØS & RE]

For å forsikre at kretskortet fungerte som planlagt, ble alle grunnfunksjoner testet hver for seg. Dette ble gjort via eksempelet «*ble\_app\_blinky*», da dette prosjektet inneholder svært lite fra før. De første testene som ble gjennomført var grunnleggende GPIO-funksjonaliteter. Det ble programmert et enkelt program for å sette en pinne høy og lav. At dette fungerte, var første indikasjon på at kortet var i fungerende stand. Det neste som måtte testes var skiftregistrene. Disse ble gjort med biblioteket vi har skrevet. Dette gikk ikke helt smertefritt, da det viste seg at LED-ene som ble brukt på demokortet var av typen felles katode, mens de nye LED-ene hadde felles anode. Dette er nærmere beskrevet i diskusjon.

### 10.4.2.2 Nettsted – JS [EJC]

Denne delen av prosjektet fungerer som spillets brukergrensesnitt. Det er her brukeren må inn for å koble seg på kuben for å spille. Etter vellykket tilkobling vil bruker få valget mellom å starte et nytt spill, eller sette kuben i demovisning. Denne demovisningen har vi valgt å kalle «*Light Show*». Velger bruker å oppdatere nettstedet samtidig som kuben er tilkople, vil tilkoplingen forsvinne. Dette er grunnlaget for et designvalg som ble tatt; å bruke JS til å generere nettstedet. Ulempen med dette designet, er at posisjonen til bruker ikke blir ivaretatt. Dette passer utmerket i vårt tilfelle, da en oppdatering av nettstedet, uansett, vil resultere i frakopling. En komplett oversikt over programflyten til nettstedet kan finnes i vedlegg 12.7



Figur 10-8 Startside



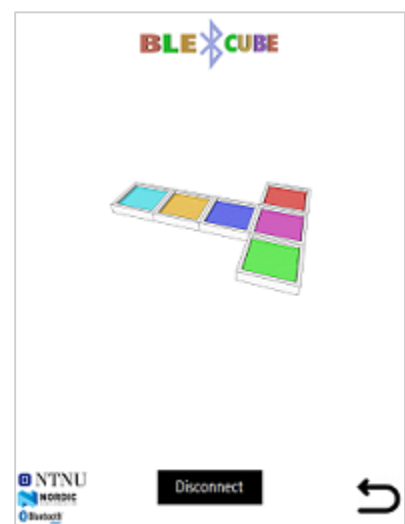
Figur 10-9 Valgmuligheter

For å kunne bruke BLE for å koble nettsted opp mot kube, må nettstedet benytte seg av et API kalt *Web Bluetooth*<sup>[24]</sup>. Dette API-et gir muligheter for å oppdage og kommunisere over BLE-standarden ved å bruke GATT-profiler. API-et finnes i sin helhet i kildelisten<sup>[24]</sup>.

Alt innhold på nettstedet er fritt tilgjengelig for alle som ønsker å bruke dette selv, enten til inspirasjon, læring eller ren kopiering. Alt dette kan finnes på vår GitHub<sup>[3]</sup>

Viktig for nettstedet er delen som velger ut en oppgave

bruker skal løse. Her ble det tatt noen valg for å gjøre det mulig å gjennomføre prosjektet innenfor gitte tidsrammer. Oppgaven som gies til bruker er ikke genuint tilfeldig. Når oppgaven tilfeldiggjøres velges ett av ti forhåndsskapte bilder. Denne funksjonen kalles for eksempel når bruker trykker på knappen som leder til «*Fold it*-spillet». Under konstruksjonen av løsningsmekanismen åpenbarte et par problemer seg. Hvordan disse er løst står i detalj i diskusjonsdelen.



Figur 10-10 Utfoldet kube

Nettstedet har i tillegg en egen JS-fil som sjekker hvilken nettleser som brukes. Grunnen til dette er den store forskjellen i oppløsning fra PC til mobil. Det er ønskelig at nettstedet skal se pent ut på begge deler, spesielt da det finnes bærbare datamaskiner med BLE innebygd, for eksempel Chromebook. Dette foregår i filen `browserCheck.js`. Denne funksjonen benytter seg også av teknikken med å sette inn HTML/CSS via JS. I dette tilfellet setter vi inn kilden til CSS-filen.

```
//Apply styles for mobile version
if (isMobile.iOS() || isMobile.Android() || isMobile.Opera()) {
    document.getElementById('styleSheet').href =
        "styles/stylesMobile.css";
}
//Apply styles for PC version
else {
    document.getElementById('styleSheet').href =
        "styles/stylesPC.css";
}
```

#### 10.4.2.3 Samspill - C og JavaScript [EJC]

For å kunne sende nødvendig informasjon begge veier mellom kube og nettsted, har datamatriksen blitt utvidet med to bit-verdier. Disse er døpt RX-flagg og TX-flagg, og har forhåndsdefinerte verdier.

Bit	7	6	5	4	3	2	1	0
	BTN1	BTN2	BTN3	BTN4	BTN5	BTN6	RX	TX
Mulig verdi	0/1/2/3/4/5/6	0/1/2/3/4/5/6	0/1/2/3/4/5/6	0/1/2/3/4/5/6	0/1/2/3/4/5/6	0/1/2/3/4/5/6	NOP(0) / RNG(1) / ANS(2)	GALT(0) / RETT(1)
Start-verdi	0	0	0	0	0	0	0	0

Tabell 10-1 Mulige verdier

Denne matrisen ligger konstant som en notifikasjon som kan leses av nettstedet. Det er RX-flagget som avgjør om noe skal gjøres med informasjonen som er blitt sendt. Et ett-tall på RX-flagget vil resultere i at funksjonen som tilfeldiggjør oppgaven kalles. Et to-tall vil kalle funksjonen som vurderer avgitt svar. Det siste denne vurderingsfunksjonen foretar seg er å plassere en eller null i TX-flagget. Dermed får kubene informasjon om svaret som ble avgitt var rett eller galt. I tillegg sendes kontinuerlig informasjon om hvilken «verdi» hver side har. Med verdi menes farge/antall trykk.

### 10.4.3 Resultat [EJC]

Dette arbeidet har resultert i et meget velfungerende nettsted som gjennomfører alle oppgavene som var tiltenkt. Funksjonen som står for å sjekke det avgitte svaret har blitt nedgradert til å selv definere en av sidene på kubens som «opp». Dette skjer ved å vise riktig farge ved valg av oppgave. Grunnlaget for at dette ble løsningen er at akselerometerfunksjonaliteten ikke ble utviklet i tide.

Biblioteket for drift av skiftregistre fungerer utmerket og uten problemer. *Beacon*-funksjonaliteten med kringkastet URL er innebygd, og vil kringkaste så lenge kubens radio ikke er opptatt med å snakke til telefonen med nettspill. Kubens analoge akselerometer detekterer rist, og formidler til nettsted at bruker vil avgi løsning. I tillegg har kretskortet vårt en egen *header*-fil for pin-konfigurasjon og krystallvalg.

### 10.4.4 Diskusjon

#### 10.4.4.1 Firmware – C [EJC & ØS]

Funksjonen som står for å vurdere om avgitt svar er korrekt eller ikke, viste seg å være en av de mer utfordrende delene av nettstedet. Utfordringen som oppstod består av at kubens ikke har en definert «opp» eller «ned». Altså er det seks mulige startpunkt en bruker kan velge. Dermed ble det tatt et valg med hensikt å minimere arbeidsmengden dette medfører. Når en ny oppgave blir valgt, vil en av sidene på kubens vise fargen den skal. I vår datamatrise er dette indeks null. Dette bestemmer indirekte hvilken verdi motsatt side er nødt å ha. Dermed er det bare fire forskjellige mulige løsninger som er korrekte, de fire mulige rotasjonene.

Ved at en av sidene blir definert automatisk, blir den motsatte også indirekte definert; disse sidene er uavhengige av orientering. Det er de fire resterende sidene som fortsatt ikke har fått en bestemmelse for «hvor» de befinner seg. Disse fire sidene vil ha fire ulike kombinasjoner, alle disse vil være riktige.

#### 10.4.4.2 Skiftregisterbibliotek [EJC & ØS]

Skiftregistre ble brukt til å styre LED-er på grunnlag av at det var dette som var tilgjengelig. Det ble diskutert for og imot både skiftregister, *charlieplexing* og *multiplexing*. Igjen var det tidsbegrensningene som ble grunnlaget for avgjørelsen. EJC har programmert disse før, og følte seg trygg på å raskt finne en løsning.

En bedre løsning for registrering av knappetrykk hadde vært om GPIOTE-systemet til NS ble brukt. Hadde dette vært implementert, kunne programstrukturen blitt skrevet på en mer effektiv måte. I stedet for at programmet kontinuerlig sjekker etter trykk, kunne heller selve trykket avbrutt programmet. Hadde gruppen hatt mer tid, er dette noe av det første som ville blitt endret. Fordelene med dette er at det kunne eksempelvis bli brukt til å tvinge kubens i søvnmodus etter en viss tid dersom ingen har anvendt kubens. I tillegg kunne kontrolleren gjort helt andre oppgaver for så å bli avbrutt av knappetrykket. Slik det fungerer nå lyttes det kontinuerlig etter knappetrykk.

Funksjonalitet for simultane trykk er noe som gjerne skulle blitt inkorporert. Store deler av jobben for å implementere dette er allerede gjort, da programløkken sitter fast i en *while*-løkke ved



knappetrykk. I denne *while*-løkken kunne det eksempelvis blitt introdusert et system som leter etter trykk på andre knapper med samme tellerverdi. Altså, to av sidene lyser likt og blir i tillegg trykket inn. Dette kunne for eksempel aktivert «*Light Show*-modusen». Problemet med å implementere dette systemet, vil være dersom bruker forsøker å trykke inn flere sider samtidig, mens bruker prøver å løse oppgave. Dette vil da resultere i at kubens går inn i «*Light Show*» og tilbakestill de fargene bruker har kommet frem til.

#### 10.4.4.3 Akselerometer [ØS]

Gruppen hadde på forhånd tilgang til et BOB for ADXL335. Det ble derfor tilrettelagt for bruk av dette i påvente på at PCB skulle bli ferdigstilt, slik at LIS2DH12 kunne programmeres. ADXL335 var kun ment som en nødløsning for LIS2DH12. Dette fordi det trekke vesentlig mer strøm, da ADXL335 trekker 320µA, sammenlignet med LIS2DH12 som kun trekker 11 µA med samme båndbredde. Ved å ikke få implementert LIS2DH12 mistet vi også muligheten til å inkorporere LIS2DH12s innebygde «*sleep to wake*» og «*return to sleep*» funksjoner. Fikk heller ikke brukt LIS2DH12s avbrudd på *wake-up*. En slik «sovefunksjon» ville nok ikke fungert i det eksisterende systemet, siden knappetrykk som nevnt leses av GPIO-API-et i stedet for GPIOTE-API-et. En annen ulempe med at ADXL335 ble benyttet, er at LIS2DH12 er digitalt. Konsekvensen av dette, at LIS2DH12 må programmeres via SPI, og man kan da ikke bruke noe av SAADC-koden, bortsett fra ristedeteksjonen. En bedre løsning enn å utvikle for ADXL335, hadde nok vært å heller designet et testbrett for LIS2DH12 tidlig i prosjektperioden.

#### 10.4.4.4 Eddystone Beacon [ØS]

Det ble nedlagt mye tid i å få multiaktivitet, det vil si at man kan bytte på å bruke radioen som kringkaster og som periferi slik at det for bruker ser ut som om begge er aktivert samtidig. JT foreslo bruk av multiaktivitet via en *timeslot*-funksjon, og et tilhørende *advertiser\_beacon\_timeslot* API. Dette ville ikke fungere som tiltenkt, og mye feilsøking ble gjennomført uten å finne noen direkte feil. JT foreslo oppgradering av hele programmet for å tilpasse det til et nyere SoftDevice. Oppgradering av SoftDevice løste dette problemet, da vi på forhånd hadde brukt en alfaversjon med noen feil. Grunnen til at denne alfaversjonen hadde vært brukt, var at den endelige versjonen ikke var lansert i starten av prosjektperioden.

#### 10.4.4.5 Nettsted – JS [EJC]

Utviklingen av toveiskommunikasjon tok en god del lengre tid enn ønsket. Grunnen til tidsproblemet på dette området var kunnskap om JS. Enveiskommunikasjon gikk raskt å implementere, da eksempler på dette var fritt tilgjengelig. Toveiskommunikasjon lyktes ikke gruppen å finne eksempler på, og det ble dermed et par uker med prøving og feiling. Etter et par møter med JT viste det seg at gruppen var svært nær en god løsning, sett bort fra et par syntaksfeil.

Det største kompromisset som ble tatt vedrørende nettstedet var den begrensede løsningsfunksjonen. Optimalt sett skulle kubens selv ha vært klar over orienteringen sin og forstått hvordan bruker prøver å løse oppgaven. Dette ble utfordrende da akselerometerfunksjonaliteten ikke falt på plass før sent i prosjektperioden.

Problemene som oppstod av at akselerometerfunksjonaliteten ikke ble implementert i tide, kunne blitt løst på andre måter. Eksempelvis kunne *solver*-en funnet ut om en av sidene i løsningen hadde en unik farge. Denne fargen kunne blitt brukt til å orientere seg i

løsningsmatrisen sendt av kuben, om denne fargen eksistere kun én gang der også, kan nettsiden løse med det som utgangspunkt. Hvis denne fargen ikke eksisterer kun én gang, vet man at kuben har feil løsning.

Funksjonen som står for å vise innholdet på nettstedet er noe mangelfull. Det hadde vært ønskelig med en viss rekursivitet, i form av at funksjonen visste hvor bruker kom fra. Dette har medført et par kompromisser ved utvikling av nettsted. For eksempel var det utfordrende å avslutte kubens «*Light Show*», når bruker trykket på tilbake-knappen på nettstedet uten rekursivitet implementert. Dette fordi tilbake-knappen er identisk for «*Fold It*» og «*Light Show*». I tillegg er det mye redundans i denne funksjonsblokken, da den skriver *display*-attributter til alle elementer, uansett om attributten er endret eller ikke.

## 10.5 3D-Modellering [GAE]

### 10.5.1 Innledning

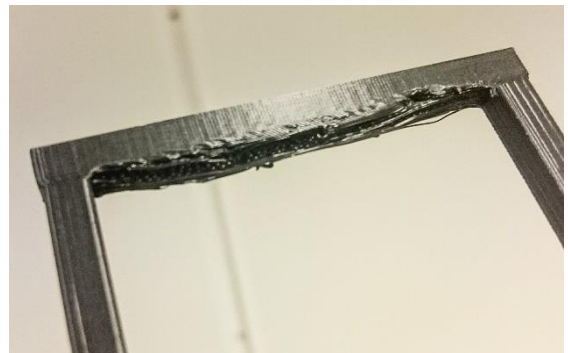
For å kunne realisere denne prototypen var det behov for noe fysisk som holder på plass alle komponentene. Modelleringen begynte med å først vurdere hvilke behov modellen måtte oppfylle. Noen av disse behovene går på tvers av fokusområdene. Eksempelvis ble det tidlig bestemt at kubens skulle romme et kretskort på  $5 \times 5 \text{ cm}$ . Videre ble det bestemt at kubens skulle bruke mekaniske knapper for å registrere et trykk på kubens side. I tillegg er det tiltenkt at et større område på hver side skulle skifte farge. For å få til dette var det også ønskelig å bruke en 3D-printer til å lage kubens.

I dette avsnittet skal vi derfor se nærmere på hvordan denne prosessen ble gjennomført, og hvordan den endelige modellen endte opp.

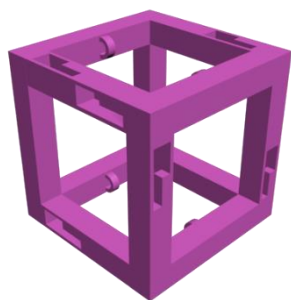
### 10.5.2 Hva har blitt gjort?

#### 10.5.2.1 Valg av 3D-printer

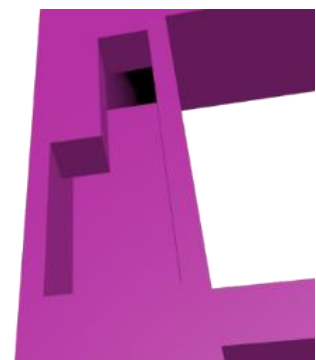
På skolen hadde gruppen tilgang på flere 3D-printere som kunne benyttes. Det ble raskt oppdaget stor kvalitetsforskjell på 3D-printerene, derfor ville det være viktig å benytte seg av den printeren som ga best resultat. Det ble først utført noen tester for å lære hvordan skolens Makerbot og Ultimaker behandlet print. Dette ga innsikt i hvordan de utførte utskriften og kvalitetsforskjeller mellom de forskjellige printerne. Den umiddelbare forskjellen som åpenbarte seg var tiden det tok å printe modellene. Makerbot var desidert raskest, men resultatet var dessverre av utilstrekkelig kvalitet i begge printerne. Printerne var ikke i stand til å skape støttemateriale på en god måte for funksjoner som går inn i objektet, men uten støttemateriale kan ikke overhengende konstruksjoner skrives ut på en god måte. Dette medførte at printeren som ble brukt var NS egen 3D-printer, Stratasys Mojo. Denne printeren bruker to materialer under utskrift, hvor det ene er selve byggematerialet og det andre er støttematerialet. Støttematerialet til denne printeren kan vaskes bort og gir derfor et godt resultat ved overhengende konstruksjoner og funksjoner som går inn i objektene.



Figur 10-13 Feil ved overhengende konstruksjon



Figur 10-12 Indre ramme



Figur 10-11 Innovervendt feste

#### 10.5.2.2 Programvare

For å få en utskrift på 3D-printer var det nødvendig å finne et passende program til oppgaven. Fra skolen var det gitt tilgang til SolidWorks og 3DS Max. Begge ble anbefalt av medstudenter. Av disse programmene ble 3DS Max valgt på grunn av flere anbefalinger fra brukere, og sammenligninger mellom disse to programmene. 3DS Max er et meget avansert modelleringsprogram som tilrettelegger for mange forskjellige modelleringsteknikker. Prosessen med å lære seg programmet, og å finne alle mulige innstillinger, var en stor utfordring på grunn av alle valgmulighetene ved hvert trinn i modelleringsprosessen.

#### 10.5.2.3 Modellering

Prosessen med å lage en god modell starter med å orientere seg om begrensninger og modellens tiltenkte egenskaper. Å kunne se for seg modellen som skal kunne brukes i praksis, er et avgjørende aspekt for hvordan modelleringen bør gjennomføres. I dette tilfellet var det et behov for en prototype som måtte tåle en god del montering og demontering.

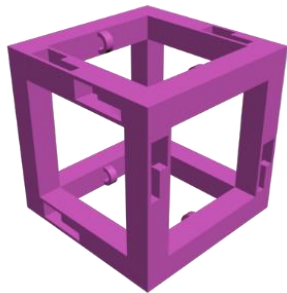
En indre ramme ble laget for at PCB skulle kunne monteres. Inspirasjon for festemekanismen på den indre rammen ble hentet fra en monteringsmekanisme brukt på røykvarslere. På denne mekanismen festes en festebrakett som holder det ytre dekselet på plass. Mellom dekselet og festebraketten ble plattform for LED og trykknapp plassert. Videre skulle fjærer benyttes for å holde på plass en plexiglassplate mot det ytre dekselet.

Det ble det også gjennomført et eksperimentert med forskjellige materialer som skulle lyses gjennom, og ga innsikt i avstander mellom disse som ga en mer optimal løsning. Dette måtte det tas hensyn til i produksjonen av en fungerende modell.

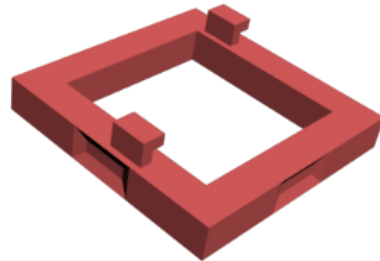
### 10.5.3 Resultater

Resultatet av modellen i 3DS Max består av seks deler som passer godt sammen. Disse hadde sin egen funksjon som var tilfredsstillende for oppgaven de skulle utføre.

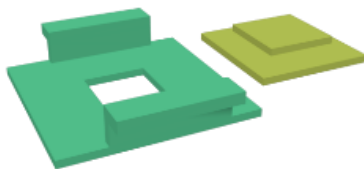
Modellene vises i figurene nedenfor:



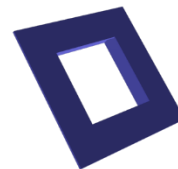
Figur 10-14 Indre ramme



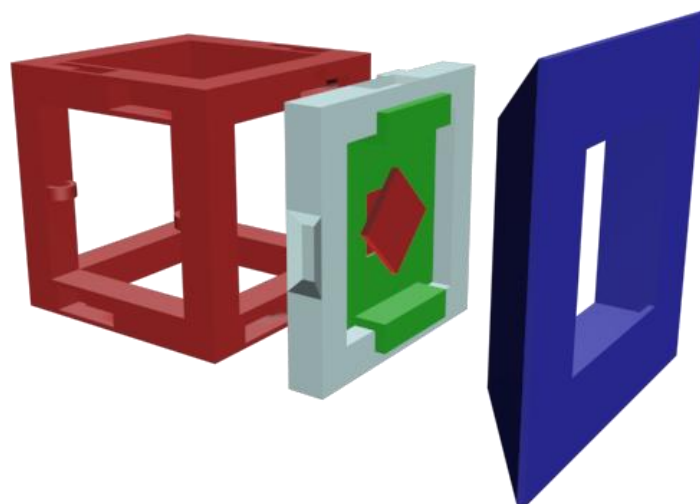
Figur 10-15 Festebrakett



Figur 10-16 Plattform for LED



Figur 10-17 Ytre deksel



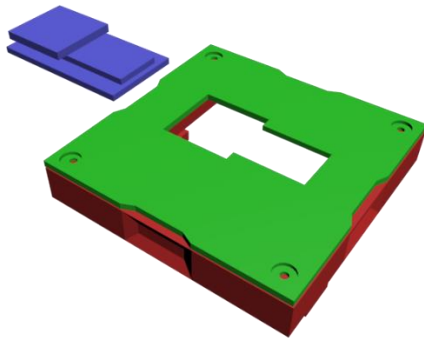
Figur 10-18 Delene samlet

Utskriften fungerte som forventet, bortsett fra LED-modulen som ble for liten til oppgaven. Festemetodene virket som forventet og de tålte stresset vi påførte systemet.

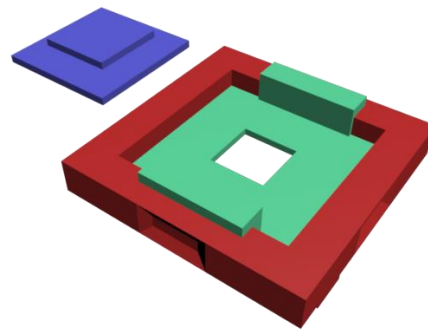
## 10.5.4 Diskusjon

### 10.5.4.1 Multimodularitet

Gruppen valgte multimodularitet fordi det ga muligheten til å se kubens fremgang. Samtidig var det muligheter for å innspill fra gruppen angående endringer, eller forbedringer for neste trinn. Det fysiske knappetrykket representerte et behov for noe som skulle bevege seg inne i kuben. For å realisere dette ble det laget rom for en plate som skulle kunne bevege seg og treffe en trykknapp. Selve registreringen av inntasting kunne blitt løst ved hjelp av andre mekanismer, eksempelvis en konduktiv trykkplate eller lignende. Ett ønske fra gruppen gjorde at valget falt på å bruke en fysisk trykknapp, noe som ble en utfordring under modelleringen. Løsningen som ble introdusert gjorde det lettere å bytte ut deler ved behov i tilfellet gruppen kom med andre innspill. Eksempelvis ble LED-modulen og monteringsbraketten endret i løpet av en av de siste dagene på grunn av plassmangel. Komponentene er også like for alle sidene, noe som gjør det lettere å lage nye deler, eller erstatte de med nye, forbedrede versjoner.



Figur 10-19 Gammel plattform for LED-er

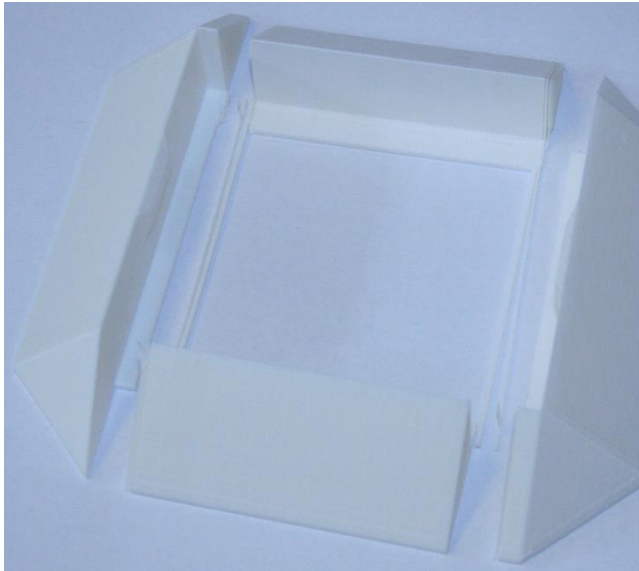


Figur 10-20 Ny plattform for LED-er

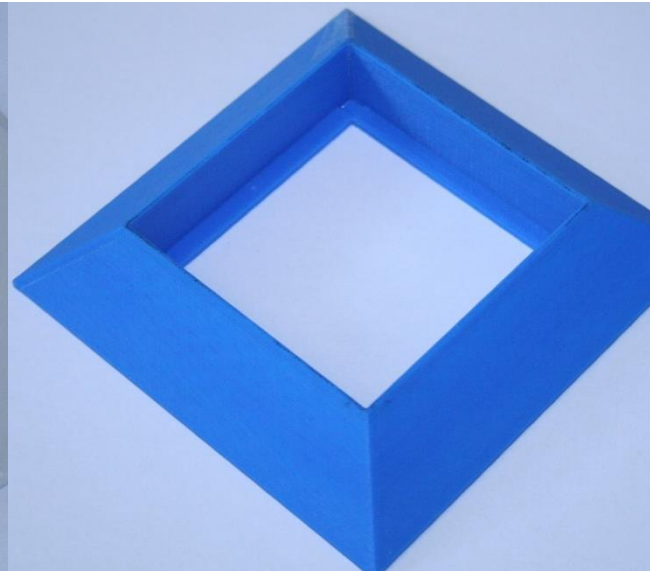
### 10.5.4.2 Veien til et godt resultat

Modellen som først ble vurdert var en tynn, indre ramme. Modellen var for svak, og etter få demonteringer ble festene ødelagt. En avgjørelse om å beholde noe av ideene ble tatt og en større modell ble laget. Videre ble det fort klart at festemekanismen ikke kunne skrives ut på skolen fordi skriverne brukte samme materiale som byggemateriale og som støttemateriale. Dette ville lede til at en del støttemateriale ville bli brukt på områder som det ikke ville være mulig å fjerne. Modellen ville derfor ikke kunne brukes. Med utskrift fra Mojo Stratasys ble dette problemet løst. Det ble fort oppdaget at 3D-utskrift fra skolens printere var meget tidkrevende og resultatet ikke alltid var like imponerende som modellen skulle tilsi.

Det er en del fallgruver ved 3D-modellering som ikke er helt enkle å se før de er erfart. Modellene kan se meget fine ut på skjermen, men når det skal skrives ut kan det virke som om konverteringen til utskriftsprogrammet fjerner enkelte vegger. Dette skyldes at i modelleringen skilles det mellom bakside og framside på en flate. Baksiden av en flate blir ikke regnet som en del av modellen og derfor fjernes de under utskrift. Videre kan slike orienteringsfeil føre til at programmet jukser ved sammenføyninger av flere objekter og fører til dannelse av flere vegger på samme plass. Disse veggene blir tolket av utskriften som bevisste oppdelinger av objektet og et tynt lag med støttemateriale blir tilført. Dette vaskes deretter vekk ved utskrift.



Figur 10-21 Feil ved utskrift



Figur 10-22 Vellykket utskrift

Utskriften i seg selv var meget tidkrevende. Det tok tjuetre timer å skrive ut fem av de ytre dekslene og selve vaskingen av de tok over en dag. Siden skriveren ikke var i umiddelbar nærhet, ble det også tapt mye tid når skriveren stanset under utskrift.

#### 10.5.4.3 Valg av festemetode

Gruppen var innstilt på å lage en prototype. Ved konstruksjon av prototyper blir det en del montering og demontering. Dette gjorde at delene som skulle lages måtte kunne tåle påkjenningen av slik behandling uten at dette fokuset går ut over enkelheten ved demontering og montering. Slike faktorer er noe som får stor betydning for konstruksjonen av objektet og de diverse delene objektet kommer til å bestå av. For å sammenlikne dette med noe kan man se for seg en fjernkontroll. Skallet til fjernkontrollen trenger bare å monteres en gang og ytre festemekanismer som låser seg ved sammensetting er en god løsning på dette. Dersom man skal demontere en slik kontroll er faren for at festene ødelegges stor og denne metoden kan derfor ikke brukes på dette prosjektet.

Videre finnes det festemekanismer designet for å brukes ofte, for eksempel i røykvarslere. Dette gjør de enkle å demontere fra en eventuell monteringsbrakett. Disse festene er små, solide deler som kommer fra en støypeprosess og holder varsleren godt på plass. Når dette blir skrevet ut på en 3D-printer blir plasten fordelt i lag over hverandre, dette gjør modellene sårbare for press eller spenn i visse retninger. Løsningen blir å lage mer solide deler som en kompensasjon for materialets litt svakere konstruksjonsmetode.

Dekselet var også ment til å festes godt. Det ble derfor til å begynne med laget en fordypning i festebraketten, der motstykket skulle sitte på dekselet. Det ble fort oppdaget at dette ble for dypt og faren for at dekselet ikke tålte påkjenningen ved montering var stor. Denne ideen ble derfor forkastet og det ble tilstrekkelig med å bare bruke friksjonen mellom festebraketten og dekselet.

Utbytning av eventuelle trykknapper eller LED-er skulle løses ved å bytte en liten del fra baksiden av monteringsbraketten. Det viste seg at trykknappen ble for stor til å passe inne i



LED-braketten og derfor ble det valgt å lime den direkte på monteringsbraketten. Ved bruk av andre LED-er og trykknapper ville dette fungert.

Det er en stor forskjell mellom å se hvordan modellen passer sammen på en skjerm, og det å faktisk holde i en ferdig modell. Det ble tidlig laget forsøk på festemekanismer som så meget lovende ut i 3DS Max, men som viste seg å være for svake, eller at mekanismen ikke fungerte etter utskrift. En av disse festemekanismene ble som nevnt rett og slett byttet ut med ren friksjon mellom flatene, noe som var tilstrekkelig for en prototype.

#### 10.5.4.4 Arbeidsprosess

Det er til stor hjelp å lage hjelpelinjer som bidrar til å finne gode plasser for de forskjellige funksjonene. Med dette blir det lettere å skape symmetri i modellen og det letter arbeidet med å videreutvikle neste trinn. En ryddig fremgangsmetode under dette arbeidet er å prøve så godt det lar seg gjøre å fjerne hjelpelinjer fortløpende, slik at de ikke blir brukt til å lage vegger inne i modellen som man ikke er klar over. Hvis dette forekommer, resulterer det i at utskriftsprogrammet ser disse indre veggene og tolker det som vegger med feil orientering. Som et resultat av dette blir utskriften ødelagt av et lag med fyllmasse og utskrevet modell faller fra hverandre. Se figur 10-21.

Ved en eventuell masseproduksjon av denne leken ville en støpeprosess egnet seg bedre. Med det i tankene burde modellen omgjøres slik at det ikke blir brukt så mye materiale for å produsere den. Både festemekanismen og tykkelsen på modellenes vegger kan med fordel forminskes betydelig, og en bedre metode for implementering av trykknapper og LED-er burde utvikles. Den opprinnelig tiltenkte låsen for dekslet burde også implementeres slik at kubene holder seg bedre sammen.

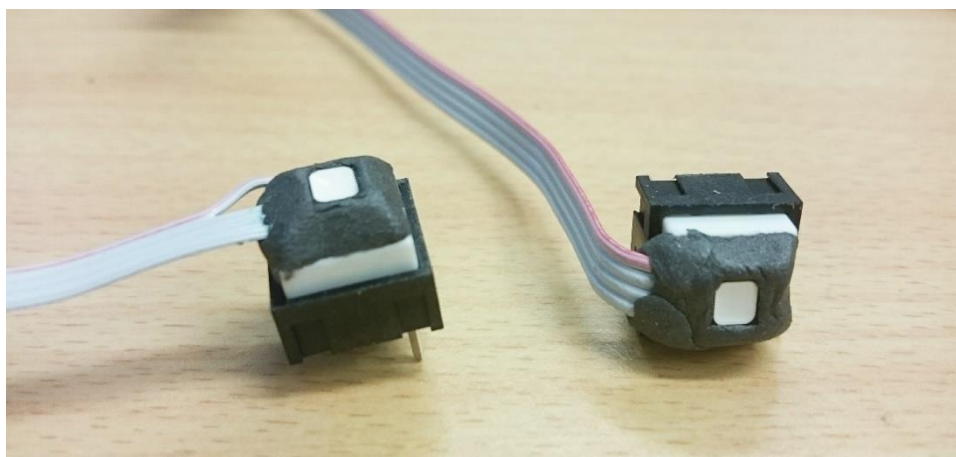


## 10.6 Sammensetning av prototyp

Etter arbeidet med alle fokusområdene var gjennomført var det på tide å forsøke å sette sammen alle delene. Dette viste seg å by på flere utfordringer. Dette kapittelet vil ta for seg disse utfordringene og hvordan de ble løst.

Ved at batteriet måtte implementeres inne i kuben etter at den indre delen av kuben hadde blitt 3D-printet, ble det besluttet at det ville være lettere å feste batteriet med skoletyggis til fordel for å modellere og skrive ut denne delen av kuben på nytt. Til tross for at dette kanskje ikke høres ut som en fantastisk god idé, viste dette seg å være mer enn tilstrekkelig og fungerer godt. Skoletyggisen holder også på plass alle festemekanismene og skruehullene for kubens elektronikk.

Det største usikkerhetsmomentet ved dette arbeidet var de ytterste delene som skulle inneholde knapper og LED. Den endelige beslutning angående hvilke knapper og LED-er som skulle benyttes ble tatt for sent i prosjektperioden til at 3D-modellen kunne endres. Dette aspektet ved prototypen ble derfor sterkt preget av improvisasjon. For at disse sidene skulle kunne lyse opp på en god måte måtte LED-ene plasseres så nærme plexiglassplatene som mulig, uten at noe kommer i veien for lyset. På en annen side må disse platene være festet til trykknappene. Med andre ord måtte LED-ene plasseres over trykknappene. Dette ble løst ved å først lodde ledninger til LED-ene, lime de fast til toppen av knappen, for deretter å stabilisere LED-ene med stålepoxykitt. Stålepoxymiddel er et to-komponentbasert reparasjonsmiddel som kan blandes sammen og formes. Dette materialet herdes fort og blir meget solid. Dette materialet ble formet slik at trykkplatene ikke treffer selve LED-ene. Dette bidrar til å minimere risikoen for at LED-ene eller loddepunktene på LED-ene blir påtrykt all kraften ved knappetrykk. Plexiglassplatene i seg selv måtte også modifiseres. En fiffig kombinasjon av lim fra en limpistol og varmen fra en varmluftspistol resulterte i en konveks form på trykkplatenes underside, noe som bidrar til å øke avstand mellom trykkplatene og LED-ene. Dette fører til at LED-ene lyser opp en større del av platene samtidig som at lyset spres gjennom limet. Disse trykkflatene stabiliseres ytterligere av fire fjærer i hvert hjørne. Disse er klippet slik at de gir akkurat passe motstand når flatene trykkes inn.



Figur 10-23 LED og trykknapp montert sammen



*Figur 10-24 Plastglass etter modifisering*

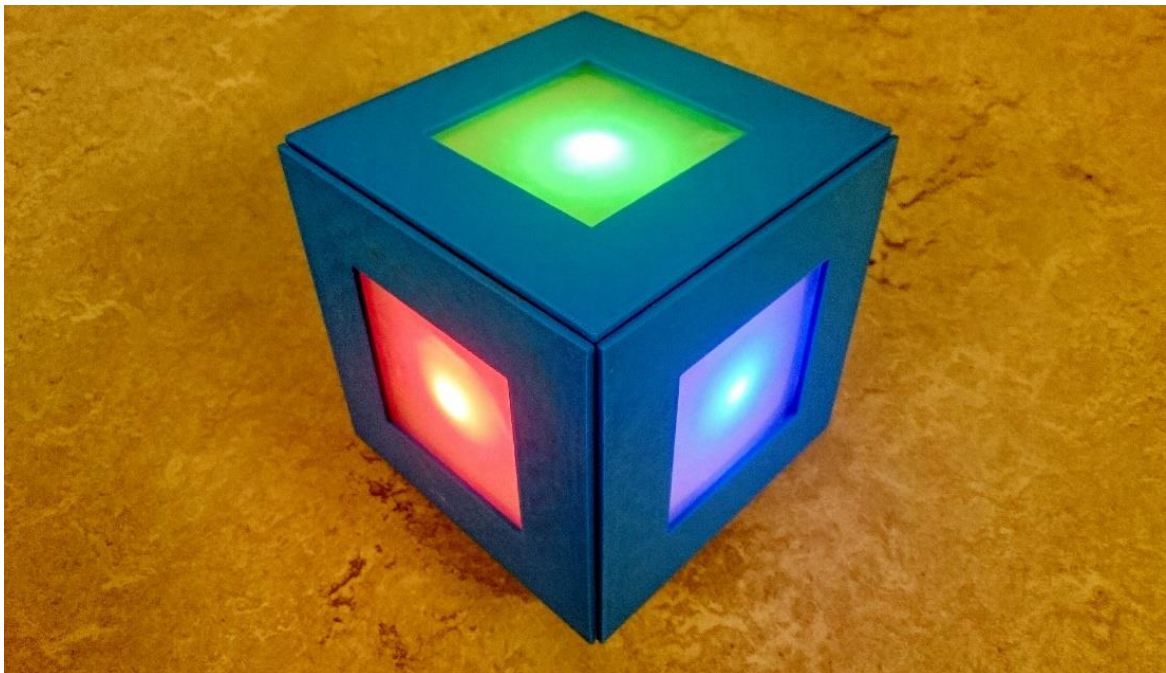
En annen ettertanke som ble implementert er en av-/påknapp, en microUSB for lading og en debug for programmering på utsiden av kuben.

## 10.7 Samlede Resultater

Den opprinnelige oppgaven ble løst ved at det ble produsert et fungerende leketøy som kringkaster en URL via *Physical Web*. Denne URL-en leder til en nettside med et spill. Nettsiden oppretter toveiskommunikasjon over BLE med kuben. Man blir deretter gitt en oppgave fra nettstedet som må løses fysisk på kuben.

Det ble vektlagt at prosjektet skulle gjennomføres med et egetprodusert PCB som passer i gruppens selvlagde interaktive kube. Prosjektet har derfor resultert i et fungerende produkt som kan brukes slik det er. Det finnes selvfølgelig forbedringspotensialer, men at prosjektet resulterte i noe fungerende reflekterer gruppens grunnønske.

Som et resultat av prosjektperioden ble det laget en kube med yttermål på  $11 \times 11 \times 11$  cm. Det ble i tillegg produsert et eget PCB og programkode for å få dette til å fungere som tiltenkt.



Figur 10-25 Ferdig kube

## 10.8 Samlet diskusjon

### 10.8.1 Beslutninger og veivalg

Som en del av å lage et produkt var det innledningsvis en idémyldring rundt hvordan produktet skulle fungere. Ett av punktene som ble vurdert var å lage en induktiv ladeplattform. Ingen av medlemmene i gruppen hadde noen erfaring med dette. Etter en del undersøkelser om induktiv lading ble det klart at dette ville bli unødvendig komplisert og tidkrevende. Som en løsning på å skape farger på alle sidene ble ideen om berøringsfølsomme LCD-skjermer introdusert. Med denne løsningen ville det gi muligheter for tekst eller grafikk på sidene og gi programmeringen en god utfordring. LCD ideen virket spennende, men dette ville bruke mange GPIO-er eller skape et behov for ett eget PCB for å håndtere det. Dette ville derfor ta for mye tid og bruke mye plass inne i kuben som allerede var presset for plass. Dette ville også spise opp nesten hele budsjettet.

Siden gruppen ønsket en fysisk tilbakemelding og en mer utfordring inne modelleringen, ble ideen om kapasitiv følsomhet forkastet.

Under gjennomføring av prosjektet ble det gjort en del avgjørelser for veien videre. En av punktene var hvordan LED-ene skulle lyses opp. Det ble vurdert om vi skulle bruke GPIO fra PCB-en, men siden det var et ønske å tilrettelegge for videre utvikling og tilgang til mest mulige GPIO-er for andre utviklere ble dette forkastet. Andre mulige løsninger er å bruke Charlieplexing eller skiftregister. Etter en vurdering av hva som ville bli mest effektivt tidsmessing ble det valgt å bruke skiftregister. Gruppen hadde erfaring med 74HC595. Valget falt derfor på å bruke tre skiftregister for å løse oppgaven.

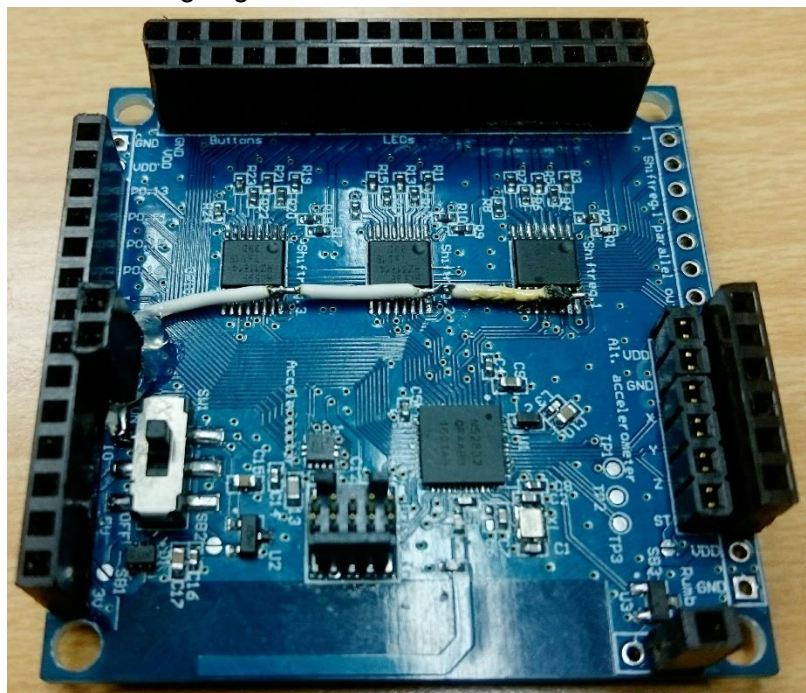
Det ble tidlig avgjort at kubens skulle inneholde en PCB på 5x5 cm, men en ytre ramme ble ikke definert. Begrensningen for kubens ytre ramme ble satt til under 12,7 cm fordi skriveren som skulle brukes ikke kunne lage større deler. Videre ble rammen satt til 11 cm fordi dette ga de best estetiske utseende, samtidig som det ga tilstrekkelig med plass for LED-er og trykknapper.



### 10.8.2 Andre utfordringer

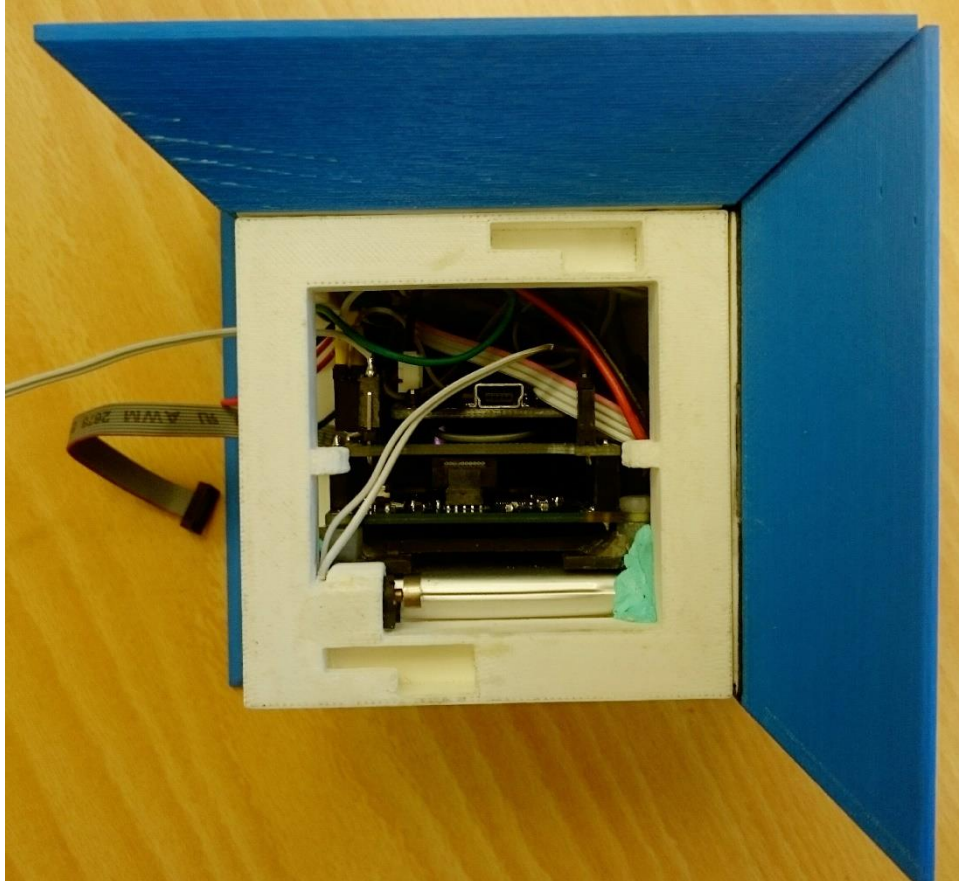
LED-ene som til slutt ble benyttet viste seg å være av typen felles anode, i motsetning til LED-ene som hadde blitt benyttet tidligere. Dette innebærer at skiftregistrene ikke kunne benyttes til å forsyne LED-ene med spenning, men heller til å jorde LED-ene. Dette medførte at programvaren måtte endres, samt at hver enkelt LED måtte tilføres spenning fra en annen kilde.

Det viste seg også at kretsspenningen på 3,2 V ikke var tilstrekkelig til å drive LED-ene. Dette kunne blitt løst ved å bytte ut alle motstandene på kretskortet, men dette ville vært tidkrevende. Det ble derfor besluttet å modifisere kretskortet. 4,2 V ble hentet fra av-/påbryteren og ført til en header hvor LED-ene kunne kobles til. Å hente spenningen fra dette punktet tillot at av-/påbryteren også styrer LED-ene. En uforutsett utfordring ved denne beslutningen var at skiftregistrene ikke var i stand til å sette spenningen på utgangene høye nok til å stoppe strømmen og slukke LED-ene. Skiftregistrene benytter tilført spenning som en referanse for å bestemme hvilke spenninger som medfører at en pin er «høy». 3 V blir derfor utilstrekkelig. I følge databladet<sup>[16]</sup> for skiftregistrene, ville 4,2 V være mulig å benytte som forsyningsspenning. VDD-pinnene på skiftregistrene ble derfor løftet fra kretsbrettet og tilført 4,2 V fra samme sted som LED-ene. Se hvit ledning i figur nedenfor.



Figur 10-26 Modifisert PCB

En annen utfordring er plassmangel. Batteri, ladekrets og adapterbrett tok også opp mye plass. Til tross for at de ovennevnte problemene ble løst, resulterte det i flere ledninger inne i kuben enn ønskelig. Ved foreløpig montering er tre av de til sammen seks sidene montert og det begynner å bli urovekkende trangt. I tillegg vil en av disse sidene inneholde ytterligere ti ledninger for microUSB, debug, og av-/påknapp. Som nevnt har ikke prototypen per dags dato blitt ferdigstilt enda. Derfor gjenstår det enda å se hvor betydningsfullt dette problemet vil vise seg å være. Det er trolig overkommelig, men vil sannsynligvis gjøre den endelige sammensetningen utfordrende.



Figur 10-27 Tettpakket kube

### 10.8.3 Forbedringspotensiale

Etter at arbeidet var avsluttet, så gruppen nærmere på det som var utført og innså at det var noen områder som kunne forbedres. Dersom det hadde vært tid til dette, ville gruppen forsøkt å utbedre følgende punkter:

- ➔ Løsning for montering av LED-er og knapper
- ➔ Spredning av farger
- ➔ Montering av PCB, batteri og nivåene
- ➔ Kabelbaner inne i rammen
- ➔ Spillforbedringer og utvidelser
- ➔ Rekursivitet på nettsted i form av navigasjonssystem
- ➔ Deteksjon av retning. Kan bruke orientering som input
- ➔ Implantasjon av LIS2DH12 via SPI
- ➔ Tilrettelagt ytterligere for videreutvikling

## 10.9 Samlet Konklusjon

Prosjektgruppen har gjennomført sitt bachelorprosjekt, «BLE-kube». Dette prosjektet har vært svært krevende og utfordrende, men de viktigste målene ble likevel nådd. Prosjektgruppen har fått muligheten til å tilegne seg god kunnskap innen fagområdene som ble utforsket. Samspillet mellom 3D-modelleringen, kretskortproduksjonen, utviklingen av nettstedet og programvare har fungert svært godt under fremstillingen av gruppens konsept.






Gruppen sitter igjen med en velfungerende prototype av god kvalitet, supplementert av en stilren nettside. Denne første prototypen vil bli gitt til Nordic ved slutten av prosjektperioden. Gruppen har lagt både hjerte og sjel i denne prototypen, og er stolte over å kunne levere den som et svar på oppgaven vi mottok i desember.

For konseptets del, har arbeidet blitt gjennomført på en slik måte at produsert stoff og tilhørende dokumentasjon burde bidra til et godt utgangspunkt for videreutvikling. Gruppen ser ikke for seg å jobbe videre med konseptet per dags dato. Det vil derimot oppfordres til videreutvikling i form av forumposter. Gruppens medlemmer vil være tilgjengelige for spørsmål fra interesserte i tilfellet noen vil benytte seg av vårt konsept og arbeid.





Med dette kan gruppen omsider anse effektmålet om å gjennomføre en treårig bachelorutdanning innen elektrofag som oppfylt. Kunnskapene som har blitt tilegnet i løpet av disse tre årene har vært uunnværlige under dette prosjektarbeidet.

# 11 Referanser/Litteratur

## 11.1 Kildeliste

[1]	Autodesk 3ds Max Learning Channel - YouTube
	<a href="https://www.youtube.com/user/3dsMaxHowTos">https://www.youtube.com/user/3dsMaxHowTos</a>
[2]	Beacons: 101: Simplified Explanation of Beacons
	<a href="https://bkon.com/blog/simple-explanation-of-ibeacon-and-eddystone-beacons/">https://bkon.com/blog/simple-explanation-of-ibeacon-and-eddystone-beacons/</a>
[3]	blecube (BLE Cube)
	<a href="https://github.com/blecube">https://github.com/blecube</a>
[4]	Differences Between iBeacon and Eddystone
	<a href="https://bkon.com/beacons/differences/">https://bkon.com/beacons/differences/</a>
[5]	eddystone/eddystone-url at master · google/eddystone
	<a href="https://github.com/google/eddystone/tree/master/eddystone-url">https://github.com/google/eddystone/tree/master/eddystone-url</a>
[6]	google/eddystone: Specification for Eddystone, an open beacon format from Google
	<a href="https://github.com/google/eddystone">https://github.com/google/eddystone</a>
[7]	Lithium Ion Polymer Battery - 3.7v 2500mAh ID: 328 - \$14.95 : Adafruit Industries, Unique & fun DIY electronics and kits
	<a href="https://www.adafruit.com/products/328">https://www.adafruit.com/products/328</a>
[8]	Meet Google's "Eddystone"—a flexible, open source iBeacon fighter   Ars Technica
	<a href="http://arstechnica.com/gadgets/2015/07/meet-googles-eddystone-a-flexible-open-source-ibeacon-fighter/">http://arstechnica.com/gadgets/2015/07/meet-googles-eddystone-a-flexible-open-source-ibeacon-fighter/</a>
[9]	MEMS digital output motion sensor: ultra-low-power high-performance 3-axis "femto" accelerometer (LIS2DH12)
	<a href="http://www.st.com/content/ccc/resource/technical/document/datasheet/12/c0/5c/36/b9/58/46/f2/DM00091513.pdf/files/DM00091513.pdf/jcr:content/translations/en.DM00091513.pdf">http://www.st.com/content/ccc/resource/technical/document/datasheet/12/c0/5c/36/b9/58/46/f2/DM00091513.pdf/files/DM00091513.pdf/jcr:content/translations/en.DM00091513.pdf</a>
[10]	Nordic Semiconductor Infocenter
	<a href="http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52.v1.0.0%2Fdevelopment%2Fnrf52_development.html&amp;cp=1_0">http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52.v1.0.0%2Fdevelopment%2Fnrf52_development.html&amp;cp=1_0</a>
[11]	nRF52 DK / Bluetooth Low Energy / Products / Home - Ultra Low Power Wireless Solutions from NORDIC SEMICONDUCTOR
	<a href="http://www.nordicsemi.com/eng/nordic/download_resource/50980/3/18936650">http://www.nordicsemi.com/eng/nordic/download_resource/50980/3/18936650</a>



[12]	nRF52832 - Product Specification v1.0
	<a href="http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf">http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf</a>
[13]	nRF52832 - Product Specification v1.0 – side 352
	<a href="http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf">http://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.0.pdf</a>
[14]	Pages - Profile Viewer (Heart Rate)
	<a href="https://developer.bluetooth.org/gatt/profiles/Pages/ProfileViewer.aspx?u=org.bluetooth.profile.heart_rate.xml">https://developer.bluetooth.org/gatt/profiles/Pages/ProfileViewer.aspx?u=org.bluetooth.profile.heart_rate.xml</a>
[15]	Physical Web Browser   Physical Web Scanners
	<a href="https://www.phy.net/physical-web-browsers/">https://www.phy.net/physical-web-browsers/</a>
[16]	Shift Register 74HC595
	<a href="https://www.sparkfun.com/datasheets/IC/SN74HC595.pdf">https://www.sparkfun.com/datasheets/IC/SN74HC595.pdf</a>
[17]	Small, Low Power, 3-Axis $\pm 3$ g Accelerometer (ADXL335)
	<a href="https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf">https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf</a>
[18]	SoftDevice Specifications - S132 SoftDevice v2.0
	<a href="http://infocenter.nordicsemi.com/pdf/S132_SDS_v2.0.pdf">http://infocenter.nordicsemi.com/pdf/S132_SDS_v2.0.pdf</a>
[19]	The Physical Web
	<a href="https://google.github.io/physical-web/">https://google.github.io/physical-web/</a>
[20]	Tutorials - Nordic Developer Zone
	<a href="https://devzone.nordicsemi.com/tutorials/25/">https://devzone.nordicsemi.com/tutorials/25/</a>
[21]	Video Library
	<a href="https://altiumvideos.live.altium.com/">https://altiumvideos.live.altium.com/</a>
[22]	USB Lilon/LiPoly charger [v1.2] ID: 259 - \$12.50 : Adafruit Industries, Unique & fun DIY electronics and kits
	<a href="https://www.adafruit.com/product/259">https://www.adafruit.com/product/259</a>
[23]	Using Shift Registers with AVR Micro - AVR Tutorial
	<a href="http://extremeelectronics.co.in/avr-tutorials/using-shift-registers-with-avr-micro-avr-tutorial/">http://extremeelectronics.co.in/avr-tutorials/using-shift-registers-with-avr-micro-avr-tutorial/</a>
[24]	Web Bluetooth
	<a href="https://webbluetoothcg.github.io/web-bluetooth/">https://webbluetoothcg.github.io/web-bluetooth/</a>

## 11.2 Vedleggsliste

### Vedlegg i rapport

- 12.1 Kretsskjema
- 12.2 Pinmap
- 12.3 Pinne-definisjoner
- 12.4 BOM / Komponentliste
- 12.5 Hovedkrets Brett – Øvrige bilder
- 12.6 Tilpasningskrets Brett – Øvrige bilder
- 12.7 Flytskjema for blecube.github.io

### Vedlegg digitalt:

- Firmwarestack (Github repository)
- Blecube.github.io (Github repository)
- PCB-Design (Github repository)
- 3D-modellering.rar
- Datasheets.rar
- Blecube.github.io.rar

## 11.3 Figurliste

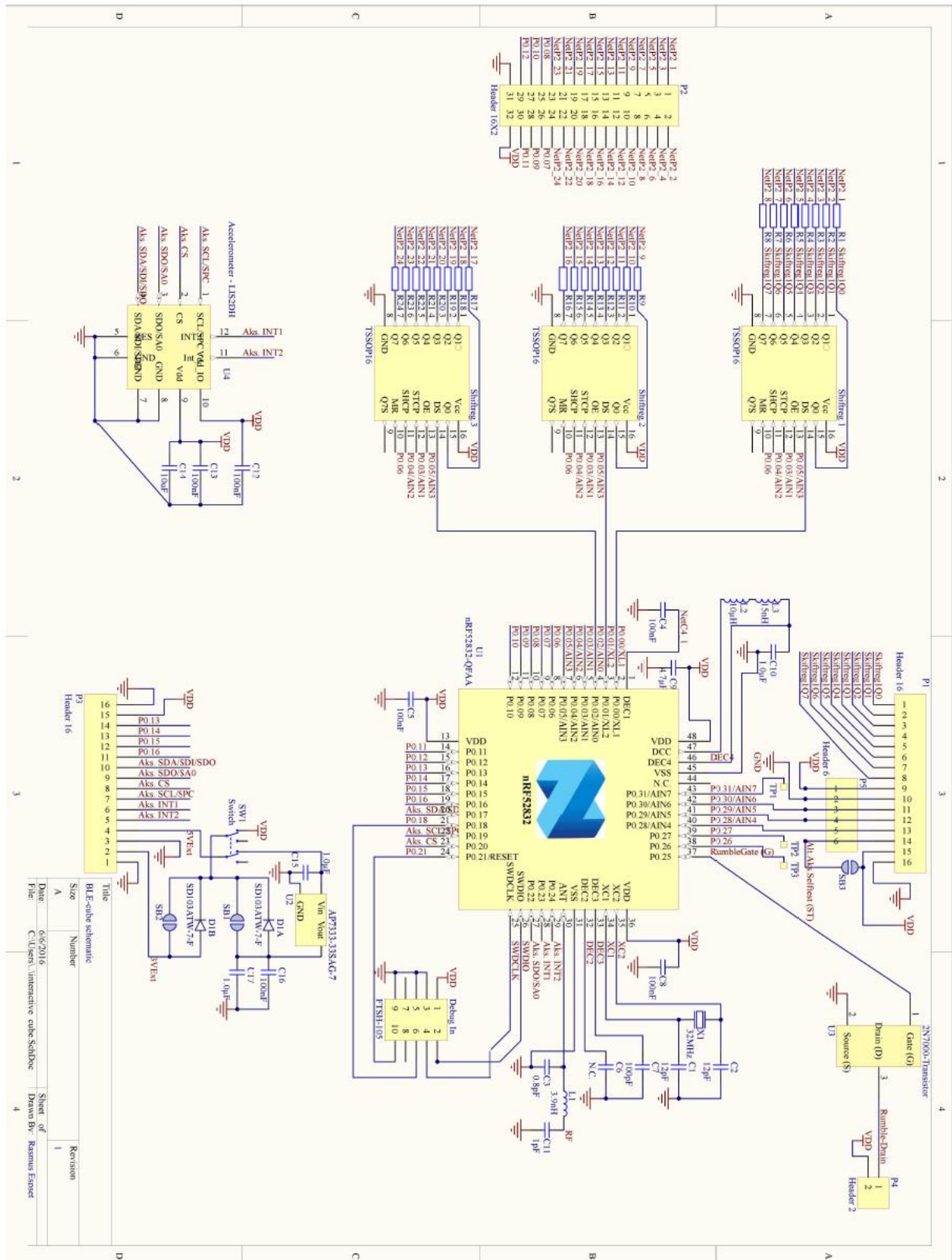
Figur 5-1 Konseptskisse	4
Figur 5-2 Utbrettet kube	4
Figur 8-1 Oppbygning av SoftDevice	15
Figur 8-2 GATT profil	16
Figur 8-3, Viser hvordan SAADC-en tilnærmer seg rett verdi vha. binærsøk.	17
Figur 8-4 Litium ion polymer-batteri	18
Figur 10-1 Demokort	20
Figur 10-2 Testbrett for lysbrytning med korrekt motstandsvektning	21
Figur 10-3 Utlegg i Altium	24
Figur 10-4 Laderegulator	26
Figur 10-5 All benyttet HW	27
Figur 10-6 Komplet kretskort med ladekrets og tilpasningsbrett	27
Figur 10-7 Kretskortets tilkobling for akselerometer	28
Figur 10-8 Startside	38
Figur 10-9 Valgmuligheter	38
Figur 10-10 Utfoldet kube	38
Figur 10-11 Innovervendt feste	43
Figur 10-12 Indre ramme	43
Figur 10-13 Feil ved overhengende konstruksjon	43
Figur 10-14 Indre ramme	45
Figur 10-15 Festebrakett	45
Figur 10-16 Plattform for LED	45
Figur 10-17 Ytre deksel	45
Figur 10-18 Delene samlet	45
Figur 10-19 Gammel plattform for LED-er	46
Figur 10-20 Ny plattform for LED-er	46
Figur 10-21 Feil ved utskrift	47
Figur 10-22 Vellykket utskrift	47
Figur 10-23 LED og trykknapp montert sammen	49
Figur 10-24 Plastglass etter modifisering	50
Figur 10-25 Ferdig kube	51
Figur 10-26 Modifisert PCB	53
Figur 10-27 Tettpakket kube	54
Figur 12-1 Flytskjema for blecube.github.io	76

## 11.4 Tabelliste

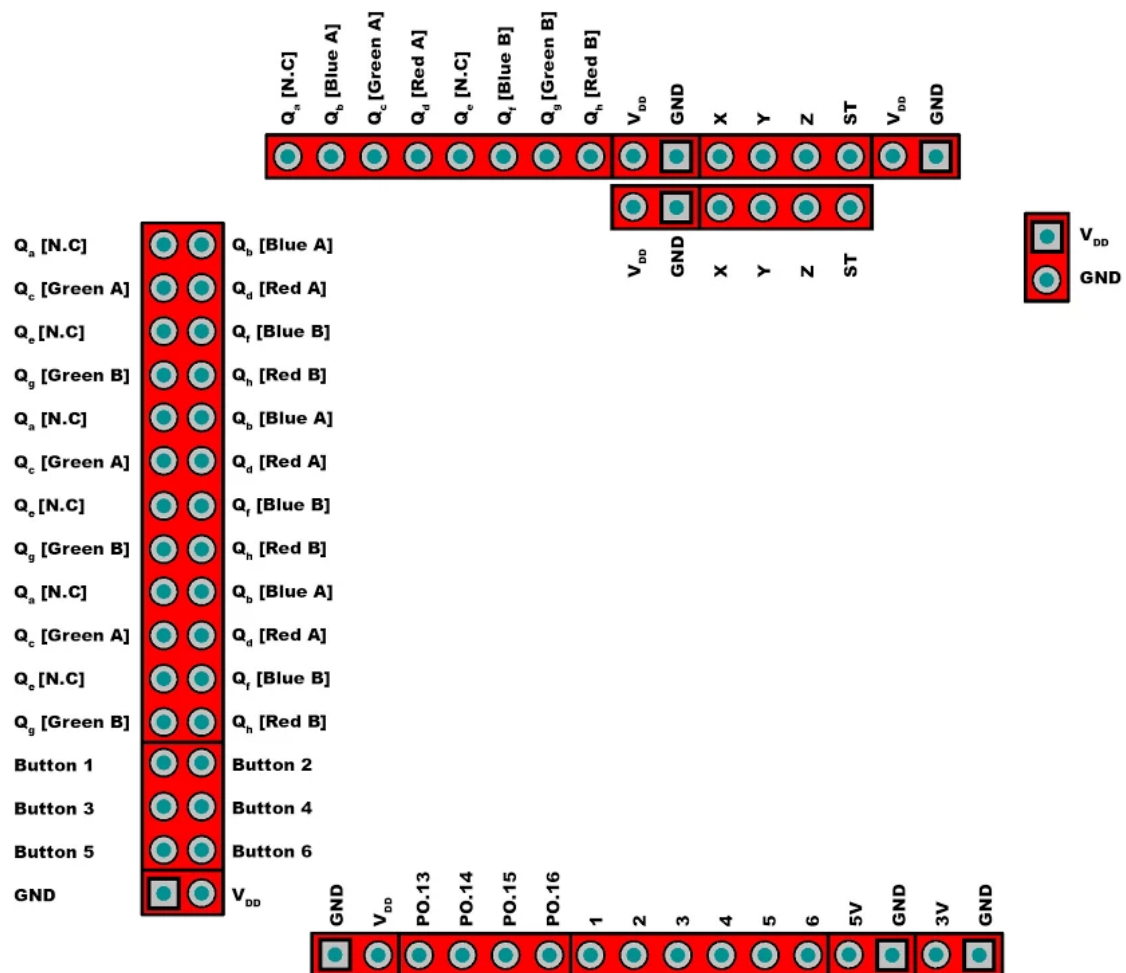
Tabell 5-1 Opprinnelig ansvarsfordeling	6
Tabell 6-1 Administrativ fordeling	9
Tabell 6-2 Benyttede verktøy og ressurser	10
Tabell 10-1 Mulige verdier	39

## 12 Vedlegg

## 12.1 Kretsskjema



## 12.2 Pinmap



Button 1 - P0.08      Button 2 - P0.07  
 Button 3 - P0.010      Button 4 - P0.09  
 Button 6 - P0.12      Button 6 - P0.11

X, Y, Z er henholdsvis P0.30, P0.29 og P0.28

1-6 (internt akselerometer)

Testpunkt 1-3 (TP1, TP2, TP3) er ikke oppført, men er henholdsvis P0.31, P0.27 og P0.26

## 12.3 Pinne-definisjoner - BLE-cube circuit board (nRF52)

<http://bit.ly/1XA3fwh>

Name	Pin	Type / Function	Comment
DEC1	1	Decoupling cap. 1	0V9 regulator digital supply decoupling.
P0.00	2	Serial data 1	Shiftreg.1: serial data input.
P0.01	3	Serial data 2	Shiftreg.2: serial data input.
P0.02	4	Serial data 3	Shiftreg.3: serial data input.
P0.03	5	RCLK	Shiftreg.1, 2, 3: storage register clock.
P0.04	6	SRCLK	Shiftreg.1, 2, 3: shift register clock.
P0.05	7	OE (inv.)	Shiftreg.1, 2, 3: Output Enable.
P0.06	8	SRCLK (inv.)	Shiftreg.1, 2, 3: direct overriding clear.
P0.07	9	Button 2	GPIO, used for button inputs.
P0.08	10	Button 1	GPIO, used for button inputs.
P0.09	11	Button 4	GPIO, used for button inputs.
P0.10	12	Button 3	GPIO, used for button inputs.
VDD	13	Power	Supply Voltage.
P0.11	14	Button 6	GPIO, used for button inputs.
P0.12	15	Button 5	GPIO, used for button inputs.
P0.13	16	P0.13	GPIO.
P0.14	17	P0.14	GPIO.
P0.15	18	P0.15	GPIO.
P0.16	19	P0.16	GPIO.
P0.17	20	Accel. SDA/SDI/SDO	Internal accelerometer pin (GPIO if unused).
P0.18	21	Debug pin 6: TRACECTL	Single Wire Output.



P0.19	22	Accel. SCL/SPC	Internal accelerometer (GPIO if unused).
P0.20	23	Accel. CS	Internal accelerometer (GPIO if unused).
P0.21/RESET	24	Reset	Debug pin 10.
SWDCLK	25	Debug pin 4	Serial Wire Debug clock input for debug and programming.
SWDIO	26	Debug pin 2	Serial Wire Debug I/O for debug and programming.
P0.22	27	Accel. SDO/SA0	Internal accelerometer pin (GPIO if unused).
P0.23	28	Accel. INT1	Internal accelerometer pin: Interrupt signal 1 (GPIO if unused).
P0.24	29	Accel. INT2	Internal accelerometer pin: Interrupt signal 2 (GPIO if unused).
ANT	30	RF	Single-ended radio antenna connection.
VSS	31	Power	Ground pin (Radio supply).
DEC2	32	Decoupling cap. 2	1V3 regulator supply decoupling (Radio supply).
DEC3	33	Decoupling cap. 3	Power supply decoupling.
XC1	34	Analog Input	Connection for 32 MHz crystal.
XC2	35	Analog Input	Connection for 32 MHz crystal.
VDD	36	Power	Supply Voltage.
P0.25	37	RumbleGate (G)	Gate on transistor U3.
P0.26	38	TP3	Testpoint 3.
P0.27	39	TP2	Testpoint 2.
P0.28	40	Z-signal - (AIN4 - Analog input)	External accelerometer: Z-values.
P0.29	41	Y-signal - (AIN5 - Analog input)	External accelerometer: Y-values.
P0.30	42	X-signal - (AIN6 - Analog input)	External accelerometer: X-values.
P0.31	43	TP1 (AIN7 - Analog input)	Testpoint 1.
N.C	44		Not Connected.
VSS	45	Power	Ground Pin.

DEC4	46	Decoupling cap. 4	1V3 regulator supply decoupling. Input from DC/DC regulator. Output from 1.3 V LDO.
DCC	47	Power	DC/DC regulator output pin.
VDD	48	Power	Supply Voltage.

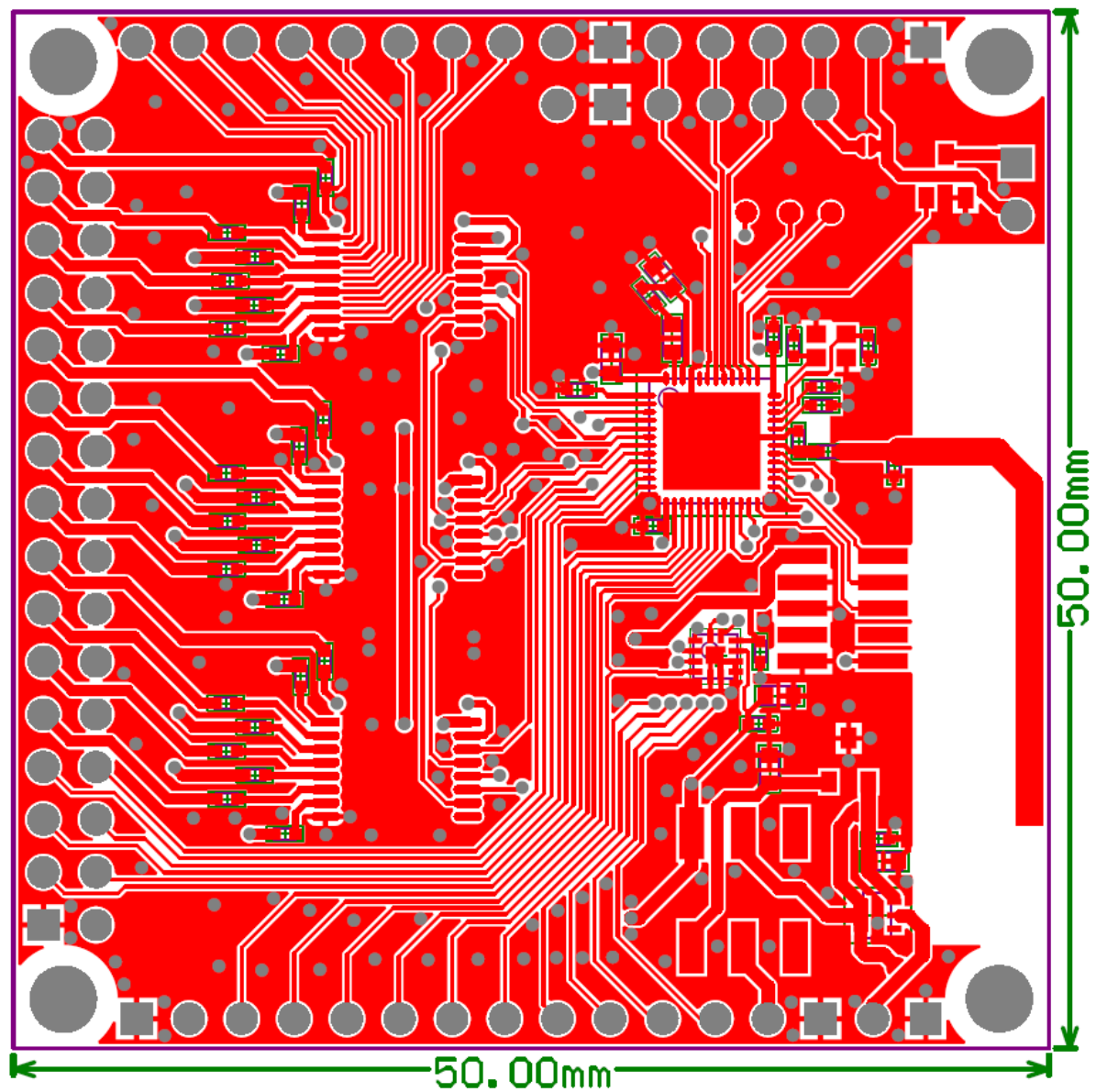
## 12.4 BOM / komponentliste

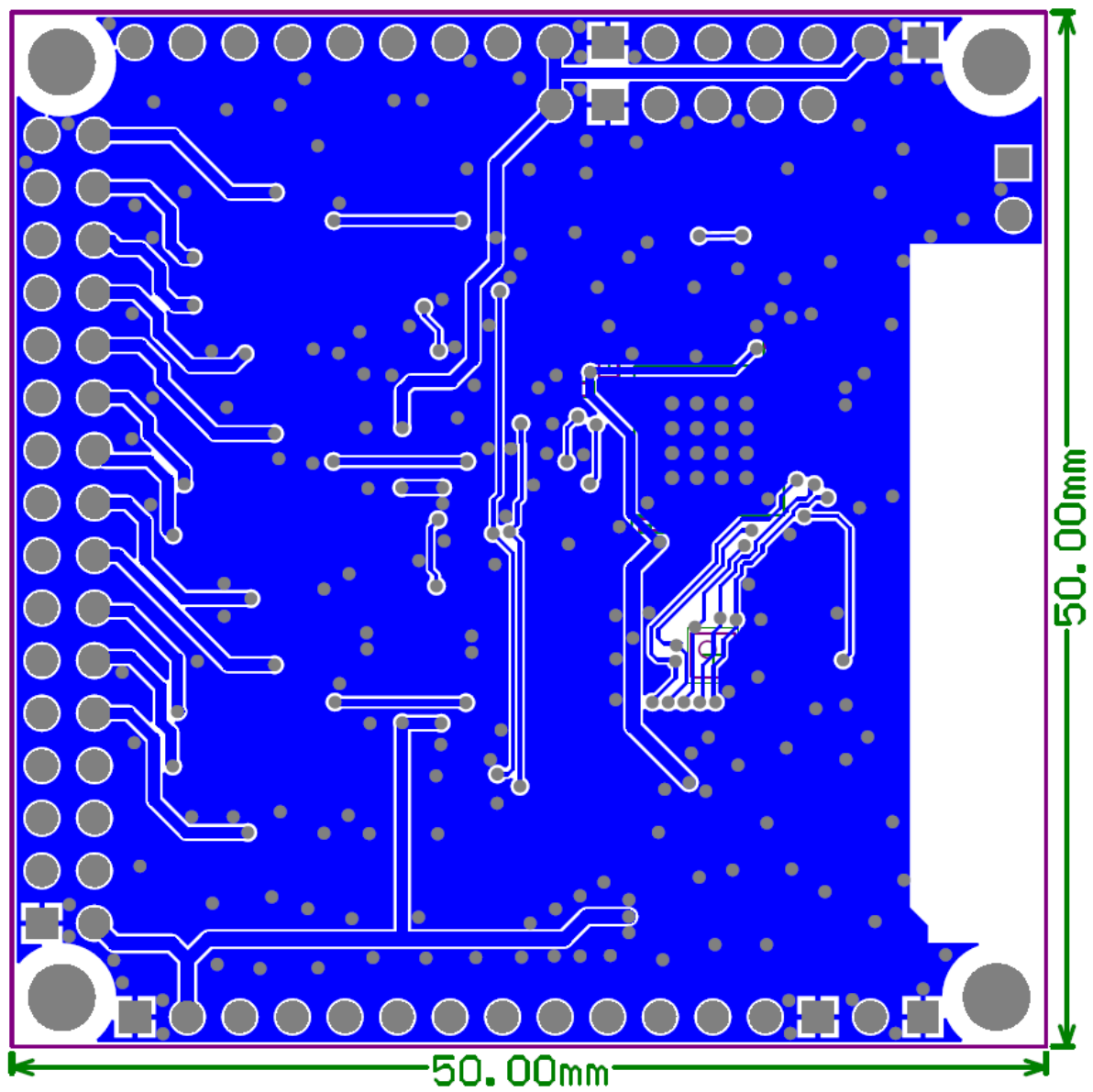
Part name	Package	Quantity	Description	Supplier	Supplier part number
NRF52832-QFAA	QFN40P600X600X90-48N	1	Multi-protocol Bluetooth Low Energy and 2.4GHz proprietary system-on-chip	Nordic Semiconductor	
74HC595	SOP16	3	Shift register	Digi-Key	568-2263-1-ND
SW-DPDT	JS202011SCQN	1	SMD slide Switch (ON-ON)	Digi-Key	401-2002-1-ND
SD103ATW-7-F	SOT-363	1	Surface Mount Schottky Barrier Diode Array	Digi-Key	SD103ATW-FDICT-ND
AP7333-33SAG-7	SOT-23	1	300mA, Low Quiescent Current, Fast Transient Low Dropout Linear Regulator	Digi-Key	AP7333-33SAG-7DICT-ND
LIS2DH12	LGA-12	1	Ultra-low-power highperformance three-axis linear accelerometer	Digi-Key	497-14851-1-ND
Samtec FTSH-105 Keying Shroud	FTSH-105-K	1	Debug In - Samtec - Conn Header 10pos dual .05" SMD Keying Shroud	Digi-Key	SAM8796-ND
2N7002	SOT-23	1	N-Channel Enhancement Mode Field Effect Transistor	Digi-Key	2N7002LT1GOSCT-ND
RESC1005X04L	0402	24	Chip Resistor, 2-Leads, Body 1.0x0.5mm, IPC High Density	Digi-Key	P390LCT-ND
XTAL_SMD2016_32MHz_8pF_Epson	BT-XTAL_2016	1	XTAL SMD 2016, 32MHz, Cl=8pF, Total Tol: ±40ppm	Farnell	171-2806
Capacitor	CAPC1005X04L	6	100nF - Capacitor, X7R, ±10%	Digi-Key	1276-1001-1-ND
Capacitor	CAPC1005X04L	2	12pF - Capacitor, NP0, ±2%	Digi-Key	712-1256-1-ND
Capacitor	CAPC1005X04L	1	100pF - Capacitor, NP0, ±5%	Digi-Key	709-1123-1-ND
Capacitor	CAPC1005X04L	1	1pF - Capacitor, NP0, ±0.1pF	Digi-Key	490-6073-1-ND
Capacitor	CAPC1005X04L	1	0,8pF - Capacitor, NP0, ±5%	Digi-Key	490-11253-1-ND
Capacitor	CAPC1608X06L	3	1uF - Capacitor, X7R, ±10%	Digi-Key	1276-1946-1-ND
Capacitor	CAPC1608X06L	1	10uF - Capacitor, NP0, ±2%	Digi-Key	445-11185-1-ND
Capacitor - Nichicon UKL1C100KDD1TD	UKL1C100KDD1TD	1	Cap Alum 10uF 10% 16V Radial	Digi-Key	493-14344-1-ND

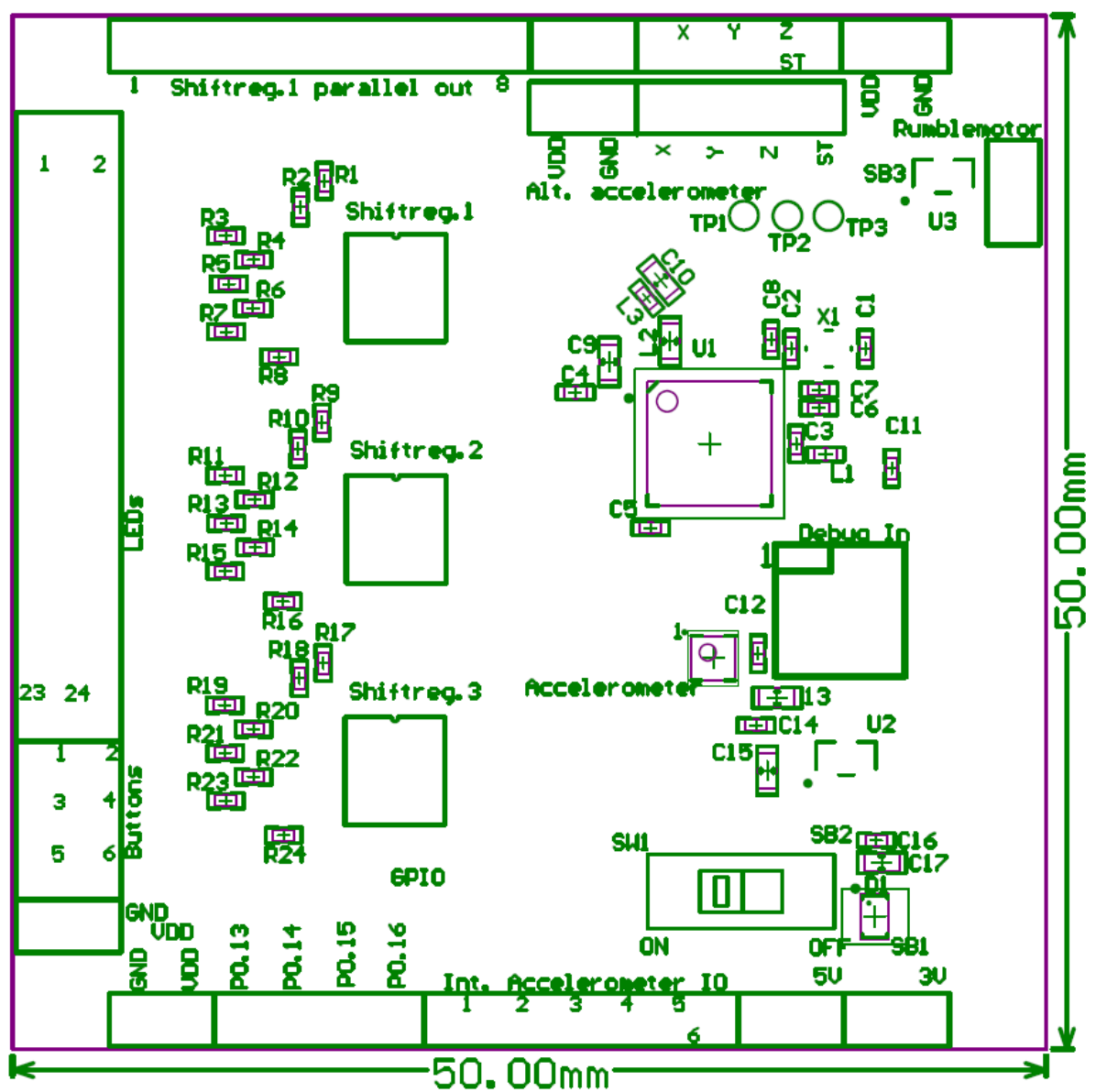
Capacitor	CAPC1608X06L	1	4,7uF - Capacitor, X5R, $\pm 10\%$	Digi-Key	1276-1045-1-ND
Inductor	INDC1608X06L	1	10uH - Chip inductor, IDC,min = 50 mA, $\pm 20\%$	Digi-Key	587-1714-1-ND
Inductor	INDC1005X04L	1	15nH - High frequency chip inductor $\pm 10\%$	Digi-Key	587-1521-1-ND
Inductor	INDC1005X04L	1	3,9nH - High frequency chip inductor $\pm 5\%$	Digi-Key	490-1131-1-ND
RGB LED		6	Lightdiode - Red, Green, Blue	RS	835-2891
Switch		6	SWITCH TACT SPST-NO 0.025A 50V	Digi-Key	EG1328-ND
Switch		1	SLIDE SWITCH DPDT GULLWING 6V	Digi-Key	CAS220GCT-ND
Battery		1			<a href="https://www.adafruit.com/products/328">https://www.adafruit.com/products/328</a>
Charge circuit		1			<a href="https://www.adafruit.com/product/259">https://www.adafruit.com/product/259</a>

Note: Some components, such as the rumble engine, or the microUSB-input and headers has been scavenged from other sources and is thus not included in this BOM.

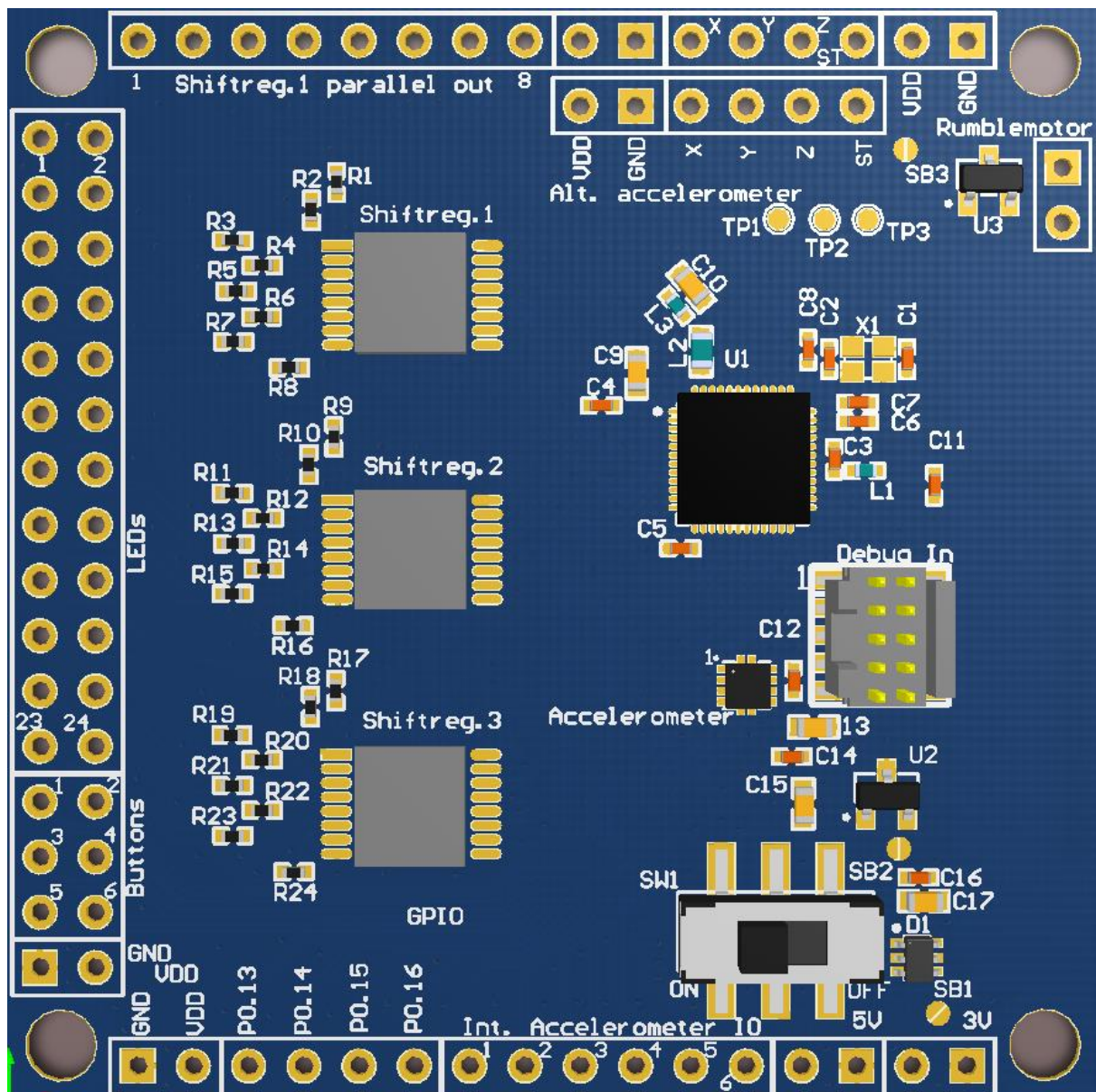
## 12.5 Hovedkrets Brett - Øvrige bilder

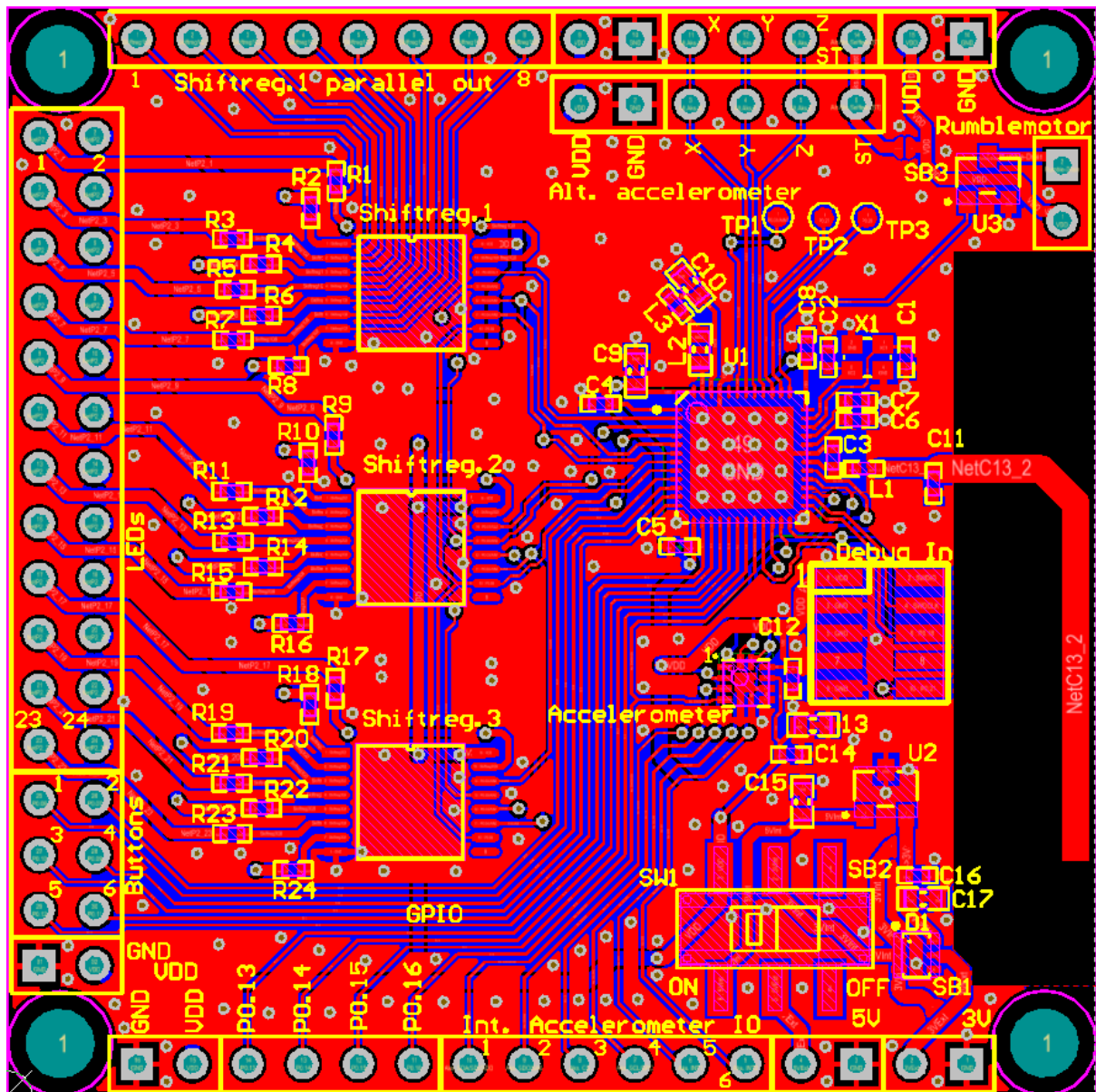






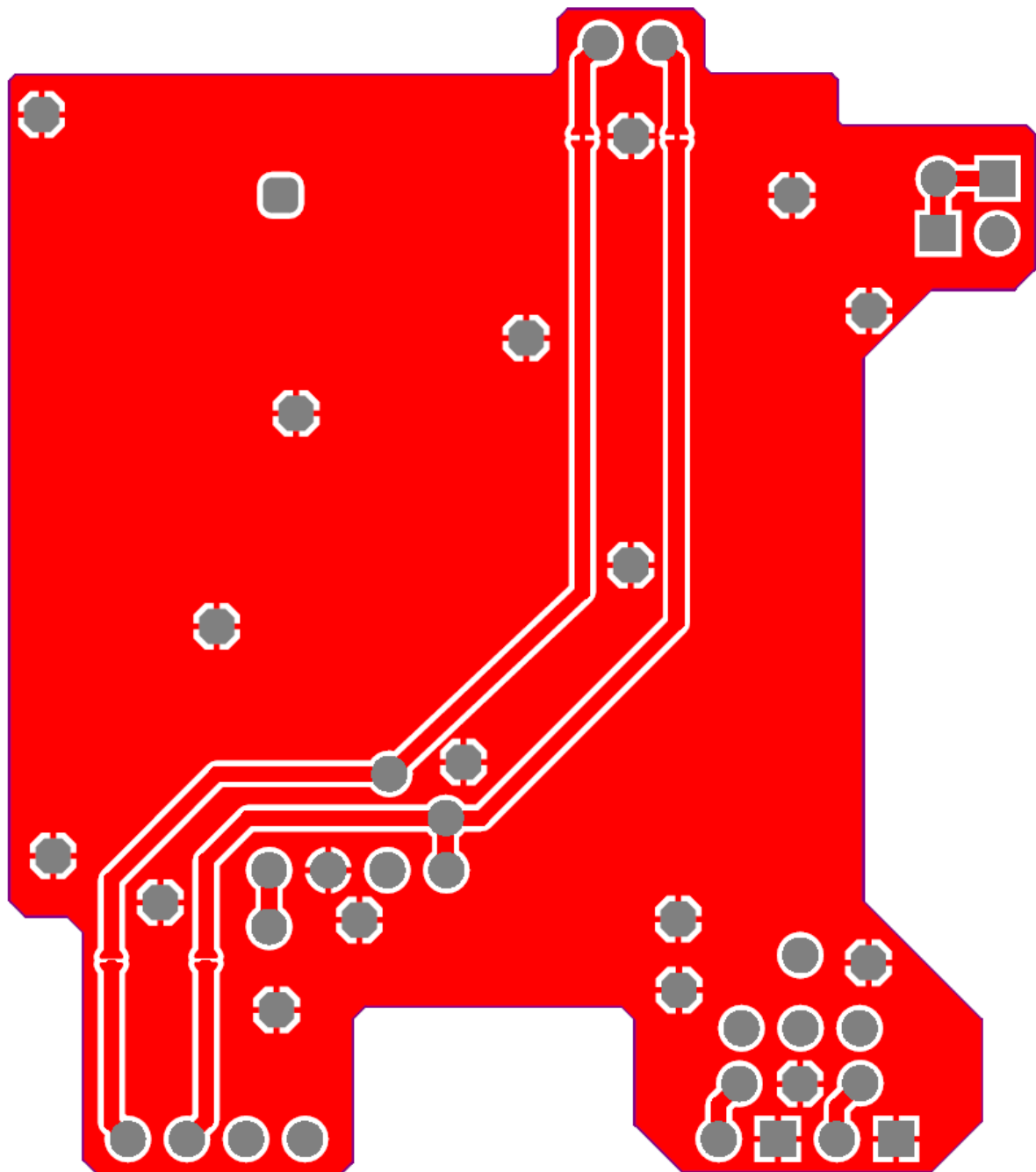


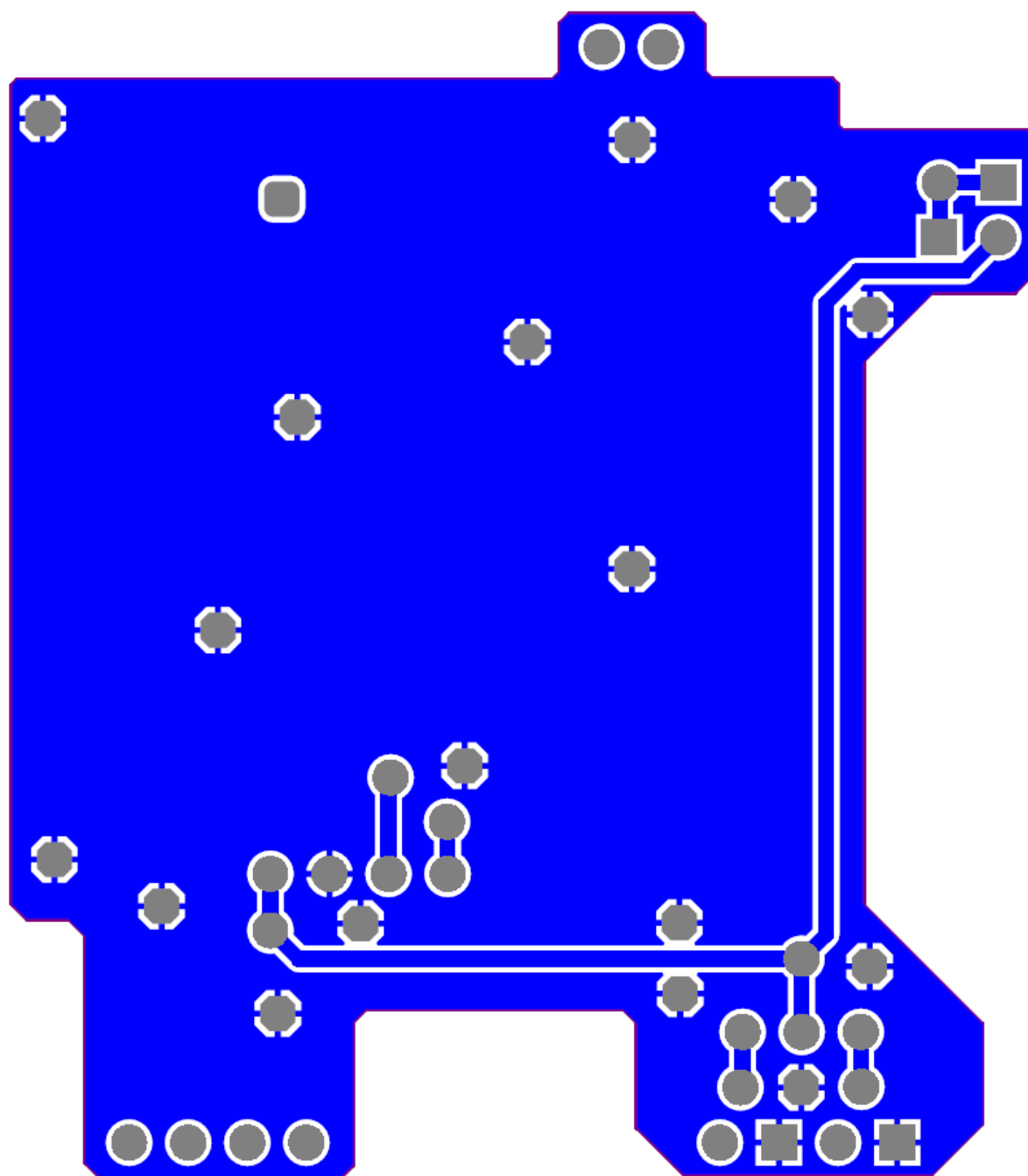


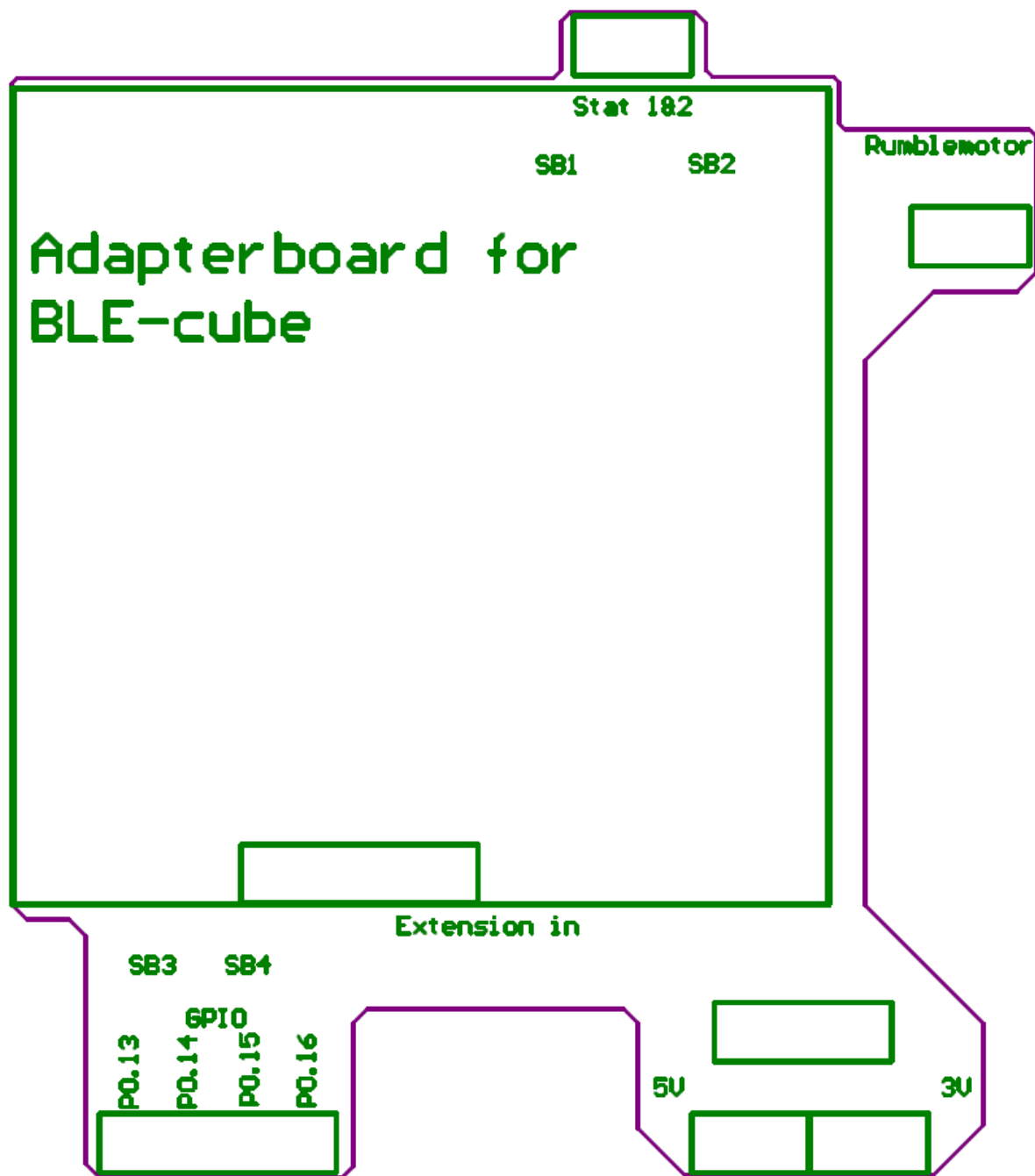


## 12.6 Tilpasningskrets Brett – Øvrige bilder

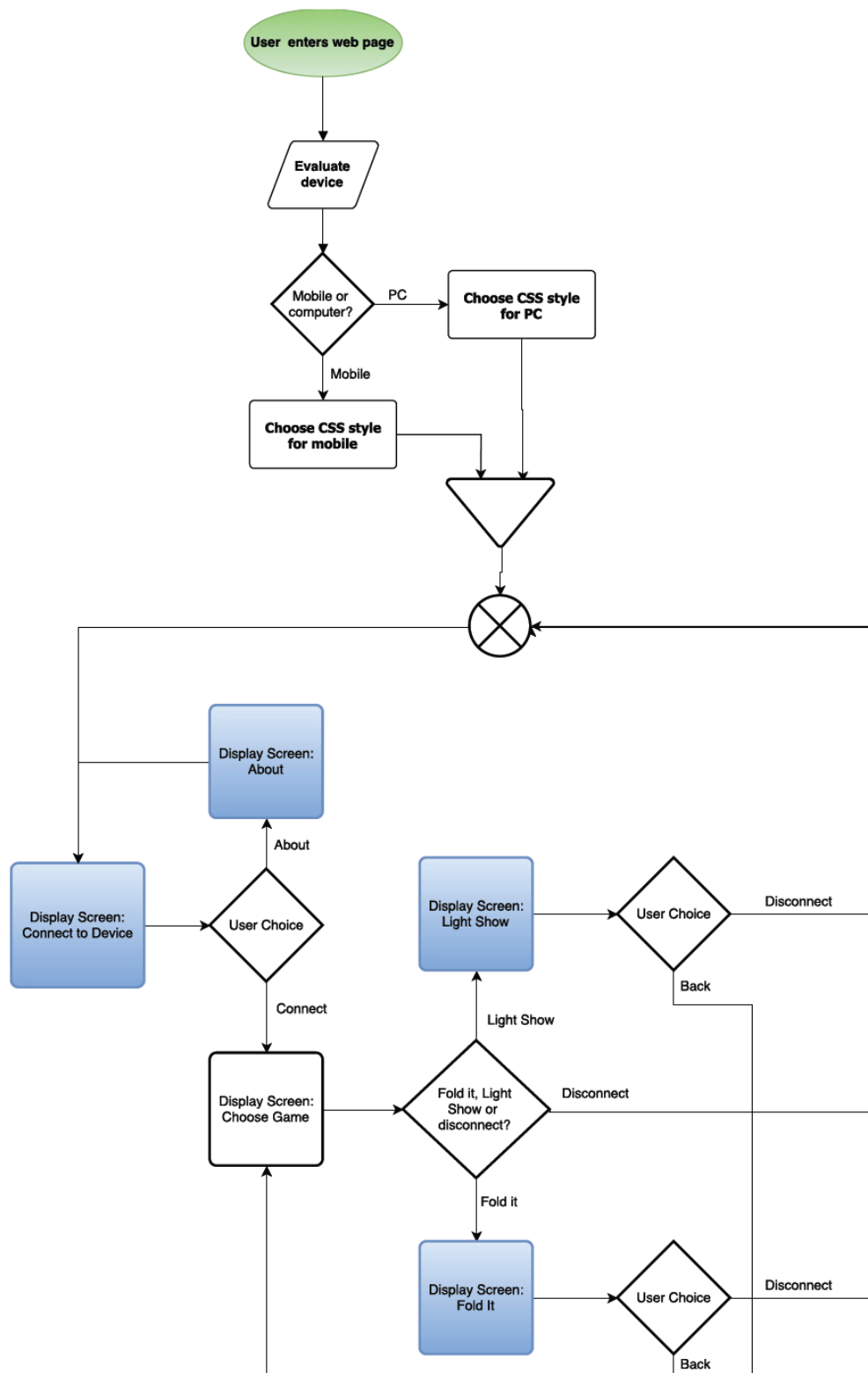
---







## 12.7 Flytskjema for blecube.github.io



Figur 12-1 Flytskjema for blecube.github.io