

Anders Ziener Kristensen, Filip August Walla  
Saasen, Håvard Olai Kopperstad, Khuong Huynh

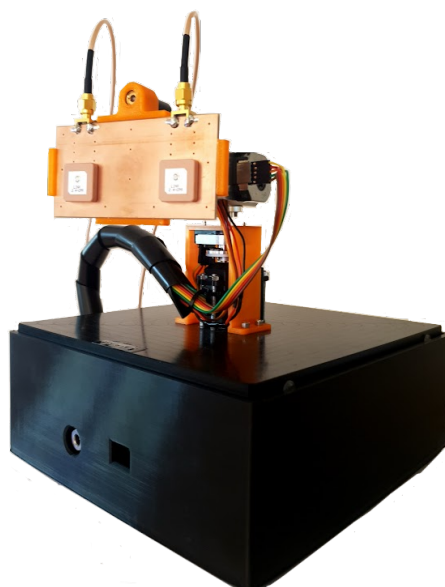
## Automatisk radiopeiling for 2.4 GHz

Bacheloroppgave i Elektroingeniør - BIELEKTRO

Veileder: Arne Midjo

Medveileder: Torolv skjølsvik

Mai 2022

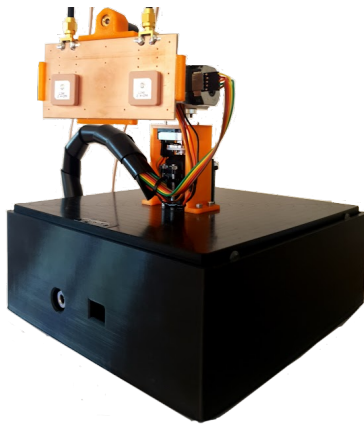


Khuong Huynh



Anders Ziener Kristensen, Filip August Walla Saasen,  
Håvard Olai Kopperstad, Khuong Huynh

## Automatisk radiopeiling for 2.4 GHz



Bacheloroppgave i Elektroingeniør - BIELEKTRO  
Veileder: Arne Midjo  
Medveileder: Torolv skjølsvik  
Mai 2022

Norges teknisk-naturvitenskapelige universitet  
Fakultet for informasjonsteknologi og elektroteknikk  
Institutt for elektroniske systemer





## Tittelside Bacheloroppgave BIELEKTRO

<b>Oppgavetittel (norsk og engelsk):</b> Automatisk radiopeiling for 2.4 GHz Automatic direction finding for 2.4 GHz	
<b>Forfattere:</b> Anders Ziener Kristensen Filip August Walla Saasen Håvard Olai Kopperstad Khuong Huynh	<b>Prosjektnummer:</b> E2226
	<b>Innleveringsdato:</b> 20.05.2022
	<b>Gradering:</b> [ x ] åpen [ ] lukket
<b>Studium:</b>	Elektroingeniør - BIELEKTRO
<b>Studieretning:</b>	Automasjon og robotikk Elektronikk og sensorsystemer
<b>Veileder internt:</b>	Arne Midjø
<b>Institutt:</b>	Institutt for elektroniske systemer
<b>Oppdragsgiver:</b>	Nordic Semiconductor
<b>Kontaktperson:</b>	Torolv Skjølvsvik, torolv.skjolsvik@nordicsemi.no, 48067071
<b>Sammendrag (norsk og engelsk):</b>	
<p>Radiopeiling er en teknikk innenfor radiokommunikasjon som tar i bruk ulike applikasjoner for mottakelse av radiobølger til å orientere seg mot en senderkilde. Ved å ta i bruk et antennearray kan målinger i fase- og motfase brukes for å detektere endringer i signalstyrken ved orientering mot senderantennen. Prosjektet går ut på å designe og konstruere en passende mottakerantenne for systemet, designe og dimensjonere robotarmen som skal styre antennen og utvikle en algoritme for styring av robotarmen og databehandling. Innretningen anvender Bluetooth Low Energy som nettverksprotokoll og opererer på 2.4 GHz. Ved undersøkelser av eksisterende teknologi innenfor fagområdene utledes det en fungerende prototype av et radiopeilingssystem, som gjennom empirisk testing kan danne et grunnlag for en estimator for nøyaktigheten til radiopeilingen.</p> <p>Direction finding is a technique within RF-communication that uses different applications for reception of radio waves to determine the direction of the beacon. By measuring the signals in an antenna array in phase and anti-phase, the sum-difference between the respective signals can be used to detect beacons by rotating the antenna in the azimuth- and elevation plane. The goal for the project is to design and construct an antenna array with the applicable characteristics, design and dimension the robot arm holding and orienting the receiver antenna and develop a program code for the search algorithm and data processing. The application utilizes the Bluetooth Low Energy as the network protocol and operates on 2.4 GHz. By researching the existing literature within the field of science, a functioning direction finding prototype was developed. Through empirical testing and observation, an estimator for the accuracy of the direction finding is derived.</p>	
<b>Stikkord norsk:</b> Bluetooth Low Energy, radiopeiling, programmering, nRF52832, radiokommunikasjon	<b>Stikkord engelsk:</b> Bluetooth Low Energy, direction finding, programming, nRF52832, RF-communication

---

## Anerkjennelser

Denne oppgaven markerer avslutningen på et 3 år langt elektroingeniørstudium, og gruppen vil med dette takke alle forelesere og involverte som har lagt til rette for at vi har tilegnet oss kunnskap og ferdigheter som gir oss rett til å kalle oss ingeniører.

Gruppen vil takke Torolv Skjølvsvik ved Nordic Semiconductor ASA og internveileder Arne Midjo ved NTNU for deres engasjement og veiledning gjennom hele prosjektet. Det rettes også en takk til senioringeniør Terje Mathiesen og førsteamanuensis Egil Eide ved NTNU for assistanse med antenneløsningen som er fremstilt i prosjektet. Avdelingsingeniør Glenn Angell ved det mekaniske verkstedet for ITK skal også takkes for sin imøtekommende holdning og hjelp til utformingen av aluminiumsdeler til robotarmen. Til slutt takkes også de frivillige ved MakeNTNU for hjelp til 3D-printing av kontrollboksen til robotarmen.

## Signaturer



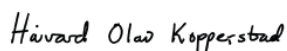
---

Anders Ziener Kristensen  
<anderszk@stud.ntnu.no>



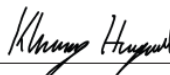
---

Filip August Walla Saasen  
<fasaasen@stud.ntnu.no>



---

Håvard Olav Kopperstad  
<haavarok@stud.ntnu.no>



---

Khuong Huynh  
<khuongh@stud.ntnu.no>

*Innholdet i denne oppgaven er fritt tilgjengelig, men publisering (med referanse) kan kun gjøres etter avtale med forfatterne.*

---

# Innholdsfortegnelse

<b>Figurer</b>	<b>iv</b>
<b>Tabeller</b>	<b>vi</b>
<b>1 Innledning</b>	<b>1</b>
1.1 Bakgrunn . . . . .	1
1.2 Oppgaven . . . . .	1
1.3 Mål . . . . .	2
1.4 Utforming av rapporten . . . . .	3
1.5 Forkortelser . . . . .	4
1.6 Definisjoner . . . . .	4
<b>2 Teoretisk rammeverk</b>	<b>5</b>
2.1 Robotmanipulatoren . . . . .	5
2.1.1 Robotgeometri . . . . .	5
2.1.2 Dreiemoment . . . . .	5
2.1.3 Treghetsmoment . . . . .	6
2.1.4 3D-printing . . . . .	6
2.1.5 Polyoxymetylen (POM) . . . . .	7
2.1.6 Dataassistert konstruksjon (CAD) . . . . .	7
2.2 Antennearray . . . . .	8
2.2.1 Radiokommunikasjon . . . . .	8
2.2.2 Elektriske ledere i radiokommunikasjon . . . . .	9
2.2.3 Linkbudsjett . . . . .	10
2.2.4 Nær- og fjernfelt . . . . .	11
2.2.5 Flerveisinterferens . . . . .	12
2.2.6 Effektnivå i antennesystemer . . . . .	12
2.2.7 Asimut- og elevasjonsplan . . . . .	14
2.2.8 Antennetyper . . . . .	14
2.2.9 Søkemetoder . . . . .	16
2.2.10 Hybridkobler . . . . .	17
2.3 Integrert system . . . . .	18
2.3.1 Komponenter . . . . .	18
2.3.2 nRF Connect for Desktop . . . . .	20
2.3.3 nRF Connect SDK periferaler . . . . .	20
2.3.4 Real-time operating system (RTOS) . . . . .	22
2.3.5 PWM-signal . . . . .	22
2.3.6 Elektronikk . . . . .	23
2.4 Algoritme og programmering . . . . .	25
2.4.1 Datatyper . . . . .	25
2.4.2 Egendefinerte datatyper . . . . .	26
2.4.3 Array . . . . .	26
2.4.4 Operatorer . . . . .	27
2.4.5 Funksjoner . . . . .	28
2.4.6 Behandlere . . . . .	28
2.4.7 Prosessynkronisering . . . . .	28
2.4.8 Databehandling . . . . .	29
2.5 Bluetooth . . . . .	29
2.5.1 Bluetooth . . . . .	29

2.5.2	Bluetooth Low Energy . . . . .	30
2.5.3	Generic Attribute Profile (GATT) . . . . .	30
2.5.4	Generic Access Profile (GAP) . . . . .	30
2.6	Statistikk . . . . .	31
<b>3</b>	<b>Metode</b>	<b>32</b>
3.1	Utstyrliste . . . . .	32
3.2	Programvare . . . . .	32
3.3	Robotikk . . . . .	33
3.3.1	Kraftberegninger . . . . .	33
3.4	Design av robotarm . . . . .	36
3.4.1	Digitale verktøy . . . . .	36
3.4.2	Designprosessen . . . . .	37
3.4.3	Produksjon av robotarmdeler . . . . .	40
3.5	Antennearray . . . . .	41
3.5.1	Kravspesifikasjoner . . . . .	41
3.5.2	Valg av antenne . . . . .	42
3.6	Design av PCB-utlegg . . . . .	44
3.6.1	1x2 Antenne-patch array . . . . .	44
3.6.2	Impedansmatching . . . . .	45
3.6.3	Karakterisering av antennearray . . . . .	46
3.7	Integrert system . . . . .	48
3.7.1	Mikrokontroller . . . . .	48
3.7.2	Komponenter . . . . .	49
3.7.3	Tilkoblinger til utviklingskort . . . . .	51
3.7.4	Biblioteker og drivere (kode) . . . . .	52
3.8	Algoritme og programmering . . . . .	53
3.8.1	Initialisering og konfigurering . . . . .	54
3.8.2	Bluetoothkommunikasjon . . . . .	54
3.8.3	Styring av servomotorer . . . . .	54
3.8.4	Søkealgoritme . . . . .	55
3.8.5	Databehandling . . . . .	55
3.9	Bluetooth . . . . .	56
3.9.1	Senderantennen . . . . .	56
3.9.2	Anvendning av BLE i applikasjonen . . . . .	56
3.10	Empirisk testing av fullstendig system . . . . .	56
<b>4</b>	<b>Resultater</b>	<b>58</b>
4.1	Design av robotarm . . . . .	58
4.2	Integrerte systemer . . . . .	69
4.2.1	Koblingsskjema . . . . .	69
4.2.2	Ferdig oppkobling . . . . .	69
4.3	Algoritme og programmering . . . . .	69
4.4	Antennearray . . . . .	70
4.4.1	PCB-utlegg for 1x2 patch-antennearray . . . . .	70
4.4.2	Resultater av antennelaben . . . . .	70
4.4.3	Empirisk testing av antennearrayet . . . . .	76
4.4.4	Linkbudsjett . . . . .	76
4.5	Radiopeiling . . . . .	79
<b>5</b>	<b>Drøfting</b>	<b>82</b>

5.1	Antennearray . . . . .	82
5.1.1	Valg av antenntype . . . . .	82
5.1.2	PCB-utlegget . . . . .	84
5.1.3	Strålingsdiagram og antennekarakteristikk . . . . .	85
5.1.4	Linkbudsjett . . . . .	86
5.2	Integrerte systemer . . . . .	86
5.2.1	Valg og styring av motor . . . . .	86
5.2.2	Valg av enkoder . . . . .	87
5.2.3	Bruk av knapp eller hall-effect-sensor . . . . .	88
5.3	Algoritme og programmering . . . . .	89
5.3.1	Bruk av C, NRF desktop, Git og headerfiler . . . . .	89
5.3.2	Begrensninger . . . . .	89
5.3.3	Valg av søkealgoritme . . . . .	89
5.3.4	Utvikling av algoritmer . . . . .	90
5.4	Bluetooth . . . . .	91
5.5	Radiopeiling . . . . .	91
<b>6</b>	<b>Konklusjon</b>	<b>94</b>
<b>A</b>	<b>Gantt</b>	<b>100</b>
<b>B</b>	<b>Budsjett</b>	<b>101</b>
<b>C</b>	<b>Fullstendig delliste</b>	<b>102</b>
<b>D</b>	<b>Møtereferat</b>	<b>103</b>
<b>E</b>	<b>Risikoanalyse</b>	<b>104</b>
<b>F</b>	<b>Koblings skjema</b>	<b>105</b>
<b>G</b>	<b>RF Koblings skjema</b>	<b>106</b>
<b>H</b>	<b>Testskjema</b>	<b>107</b>
<b>I</b>	<b>C kode</b>	<b>117</b>
<b>J</b>	<b>Plakat</b>	<b>143</b>

## Figurer

1	Illustrasjon av Bluetooth-AoA-prinsippet . . . . .	16
2	Skjema for en 180°-kobler . . . . .	18
3	Super-loop . . . . .	22
4	RTOS . . . . .	22
5	Eksempel på et PWM-signal med en arbeidssyklus på 50 % (Lambert 2021) . . . . .	23
6	Pull-up-krets . . . . .	24
7	Pull-down-krets . . . . .	24
8	Enkel spenningsdeler . . . . .	24
9	Illustrasjon av robotkonfigurasjonen ved maksimalt statistisk dreiemoment for elevasjonsservoen . . . . .	33
10	Illustrasjon av robotkonfigurasjonen sett ovenfra med elevasjonsservo i horisontal retning . . . . .	34
11	Utsnitt av egendefinerte parametre i Autodesk Fusion 360 . . . . .	37
12	3D-modell av toppen av akslingen mellom asimut- og elevasjonsservoen . . . . .	39
13	3D-modell av akslingen mellom asimut- og elevasjonsservoen . . . . .	39
14	Linx Technologies, 2.4 GHz Keramisk patch-antenne . . . . .	43
15	Anaren 3A0200 180°hybridkobler . . . . .	43
16	Linx Technologies, 2.4 GHz Keramisk patch-antenne . . . . .	44
17	Schematic-filen til PCB-utlegget der signalplanet er representert til venstre i rødt og jordplanet til høyre i blått . . . . .	45
18	Miljø hvor innendørs testing ble gjennomført . . . . .	48
19	PE42442 Evalueringkort . . . . .	50
20	3D-modell av robotarmen med komponenter og deler . . . . .	58
21	3D-modell av kontrollboksen til robotarmen . . . . .	59
22	3D-modell av baseplaten til robotarmen . . . . .	60
23	3D-modell av servomotoren DFRobots SER0038 . . . . .	61
24	3D-modell av akslingen mellom de to roterende leddene til robotarmen . . . . .	62
25	3D-modell av festeanordningen for enkoder i asimutplanet . . . . .	63
26	3D-modell av servomotorfestet til elevasjonsservoen til robotarmen . . . . .	64
27	3D-modell av nedre del av elevasjonsservoen til robotarmen . . . . .	65
28	3D-modell av det roterende festet til elevasjonsservoen til robotarmen . . . . .	66
29	3D-modell av festeanordningen for servomotoren DFRobots SER0049 . . . . .	67
30	3D-modell av antennefestet til robotarmen med forenklet illustrasjon for antennearray . . . . .	68
31	Ferdig oppkobling av integrert system . . . . .	69
32	PCB-utlegg med på-loddede patch-antenner: Antenne 1 (venstre) og Antenne 2 (høyre) . . . . .	70
33	Strålingsdiagram Antenne 1 (dB) . . . . .	71
34	Strålingsdiagram Antenne 2 (dB) . . . . .	71
35	Strålingsdiagram Antenne 1 (Smith) . . . . .	72
36	Strålingsdiagram Antenne 2 (Smith) . . . . .	72
37	Strålingsdiagram for måling i fase ( $\Sigma$ ) der x-aksen er grader og y-aksen er dBm . . . . .	73
38	Strålingsdiagram for måling i fase ( $\Sigma$ ) . . . . .	74
39	Strålingsdiagram for måling i motfase ( $\Delta$ ) der x-aksen er grader og y-aksen er dBm . . . . .	74
40	Strålingsdiagram for måling i motfase ( $\Delta$ ) . . . . .	75

41	Måling i motfase ( $\Delta$ ) i asimutplanet innendørs ved 5 meter avstand .	76
42	Måling i fase ( $\Sigma$ ) i asimutplanet innendørs ved 5 meter avstand . . .	76
43	Plott av distanseavvik ved ti radiopeilinger på 5.0 m . . . . .	79
44	Polardiagram for resultatene fra ti radiopeilinger på 1.0 m, 5.0 m og 10.0 m respektivt . . . . .	81

## Tabeller

1	Oversikt over logiske operatører og relasjonsoperatører . . . . .	27
2	Bluetooth-klasser for signalstyrke . . . . .	30
3	En liste over komponentene som inngår i systemet . . . . .	32
4	Programvaren som ble benyttet under prosjektet . . . . .	32
5	Kravspesifikasjoner for antennearrayet . . . . .	41
6	Kravspesifikasjoner for dimensjonering av antennearrayet . . . . .	42
7	Kravspesifikasjoner for drift av antennearrayet . . . . .	42
8	Sannhetstabell for svitsjen på evalueringkortet (pSemi 2021) . . . . .	50
9	GPIO-tilkoblinger . . . . .	51
10	Systemvariabler i linkbudsjettet . . . . .	76
11	Passive tap i linkbudsjettet . . . . .	77
12	Flerveisinterferens ved 1.0 m, 5.0 m og 10.0 m . . . . .	77
13	Linkbudsjett for antennesystemet . . . . .	78



# 1 Innledning

## 1.1 Bakgrunn

I et samfunn med stadig økende trådløs informasjonsflyt blir mer informasjon enn noen gang overført i form av radiobølger. Overføring av data ved bruk av retningsstyrte antenner gjør det mulig å effektivt sende store mengder data over lange distanser samtidig som det sparer energi ved å unngå å sende overflødige signaler der hvor det ikke er nødvendig. Retningsstyrte antenner reduserer også støy og flerveisinterferens mellom sender og mottaker (NetXL 2019). Ulempen derimot, er antenneoppsettet krever riktig orientering av den retningsstyrte antennen, der en liten feilmargin gjør signalet fort faller av. Ved å montere en antenne på en robotinnretning blir det mulig å kontrollere orienteringen til den retningsstyrte antennen. På denne måten kan antennen styres etter hvor signalstyrken er sterkest.

Det er ulike måter å utføre peiling av retningsstyrte antenner på, men den mest kjente og brukte metoden er ved bruk av patch-antennearray. Ved å endre på faseforskyvningen i de ulike antenneelementene kan hovedloben endres, som vil si at retningen til det utstrålte signalet endres. En annen løsning er å montere en retningsstyrt antenne på en robotinnretning slik at nettopp orienteringen til antennen kan endres.

Det å montere antenner på en robotinnretning er ikke et nytt konsept. Slike løsninger blir brukt ved satellittkommunikasjon for å orientere en bakkemontert antenne i asimut- og elevasjonsplanet mot satellitter som befinner seg i verdensrommet. Løsningen blir også brukt for å motvirke jordens rotasjon for å holde en bestemt posisjon på himmelen. Når dette er sagt, er det ikke funnet artikler på noe liknende for en mindre skala, slik som det er gjort i dette prosjektet.

## 1.2 Oppgaven

I oppgavebeskrivelsen gitt i NTNU (2021) står det blant annet at “Studentene skal utvikle en robotarm eller en innretning som skal kunne orientere en retningsstyrt antenne for 2.4 GHz” og at “[...] Studentene skal måle og vurdere hvor nøyaktig man kan finne lokasjonen til en eller flere stasjonære radiosendere på 2.4 GHz.”.

For å kunne besvare nøyaktighetsspørsmålet er det først nødvendig å utvikle en robotinnretning med mulighet for å orientere i asimut- og elevasjonsplanene. Det må også velges en antenne som har en karakteristikk som tillater radiopeiling etter en 2.4 GHz sendeantenne. Det må også utvikles en algoritme som tolker data samlet inn fra mottaksantennen og utfører nødvendige rotasjoner på robotarmen.

På grunn av at oppgaven har mange utfordringer som må løses før nøyaktighetsspørsmålet kan besvares, kan en oppsummering av arbeidsoppgavene se slik ut:

- Utforme et design av en robotarm med mulighet for å orientere den retnings-

styrte antennen i rommet. Det antas at antennen har behov for å orienteres  $180^\circ$  i asimutplanet og  $50^\circ$  i elevasjonsplanet

- Det må gjøres et valg av hvilke komponenter som egner seg i robotarmen og det må utvikles drivere og bibliotek for robotstyringen. I tillegg må komponentvalgene ses i lys av budsjettbegrensninger
- Det må skaffes innsikt i hvordan utvikling med nRF Connect SDK-et gjøres i tillegg til å bli bedre kjent med programmering i C
- En antenne med ønskelig karakteristikk må finnes eller produseres for å gjøre det mulig å peile etter signalkilden
- Det må programmeres en algoritme som analyserer innsamlet data fra antennen og bestemmer posisjonen til signalkilden basert på det innsamlede data-grunnlaget
- Gjøre empiriske tester og vurdere hvor nøyaktig posisjonen til en stasjonær radiosender på 2.4 GHz kan lokaliseres

### 1.3 Mål

Det ble satt noen mål for radiopeilingsløsningen i prosjektet, der disse blant annet er:

- Ved å benytte 3D-modellering og 3D-printing muliggjør det billig og effektiv testing og produksjon av deler til robotarmen, samtidig som det gjør produktet enkelt å gjenskape.
- Det må etterstrebes informasjon om komponentene som inngår i robotarmen for å gi tilstrekkelig nøyaktighet og presisjon som er nødvendig for peilingen etter signalkilde
- Funksjonalitet som allerede er tilgjengelig i nRF Connect SDK-et må brukes mest mulig for å unngå behov for å utvikle egne drivere og bibliotek
- Det må innhentes informasjon om forskjellige antenner for å bestemme hvilken antenntype som passer best for formålet i dette prosjektet
- Det må innhentes informasjon om søkealgoritmer for å finne ut hva som fungerer best for robotarmen og antennen som benyttes i prosjektet
- Utføre empiriske tester for å kunne svare på hvor nøyaktig lokasjonen til radiosender på 2.4 GHz kan bestemmes ved hjelp av innretningen som blir utviklet
- Fordeler og ulemper med alle komponenter i innretningen må veies opp mot hverandre for å holde prosjektet innenfor budsjettet og oppnå et godt resultat

## 1.4 Utforming av rapporten

Rapporten er strukturert slik at leseren får kunnskap som er nødvendig for å kunne forstå innholdet i rapporten og gjenskape resultatene, med forbehold om at grunnleggende kunnskap om elektroteknikk, robotikk og antennteknikk på bachelornivå er tilegnet fra før.

Teorikapittelet inneholder alt av teori bak prinsippene som blir omtalt senere i rapporten. Det er tatt utgangspunkt i at grunnleggende kunnskap om fagfeltene allerede er kjent for å forstå innholdet i teori, men om leseren har behov for videre utdypning er det anbefalt å tilegne seg ekstra kunnskap på egen hånd.

Metodekapittelet tar for seg hvordan selve prosjektet er utført og omfatter hvordan de ulike problemstillingene i prosjektet ble løst. Det blir også presentert hvilket utstyr og komponenter som er tatt i bruk, og begrunnelser for hvorfor akkurat disse er valgt.

Kapitlet om resultater dekker endelige resultat i form av figurer, tabeller og vedlegg. Her presenteres konkrete funn som kommer av arbeidet som ble utført.

Drøftekapitlet diskuterer resultatene presentert i kapitlet over og hva disse faktisk betyr ved å sette forskjellige synspunkt opp mot hverandre. I tillegg blir det drøftet hvordan resultatene eventuelt kan forbedres og hvordan prosjektet kan videreutvikles.

Til slutt blir det gitt en konklusjon for rapporten. Konklusjonen gir en kort oppsummering av hva som er blitt gått igjennom i rapporten i tillegg til å gi endelig svar på problemstillingen. Det blir også pekt på mulige utvidelser av systemet.

## 1.5 Forkortelser

---

Forkortelse	Definisjon
ADC	Analog-til-digital omformer
ASCII	American Standard Code for Information Interchange
BLE	Bluetooth Low Energy
C89	ANSI Standard in 89
C99	Videreutviklet standard for C
CAD	Computer Aided Design (Dataassistert konstruksjon)
CPU	Central Processing Unit
DOS	Disc Operating System
HPBW	Half Power Beam Width
VSWR	Voltage Standing Wave Ratio
ISM	Industrial, Scientific and Medical
ISR	Interrupt Service Routine
MCU	Microcontroller unit
PCB	Printed Circuit Board
PoF	Proof of Concept
PPR	Pulses Per Revolution
RSSI	Received Signal Strength Indicator
SoC	System on Chip
UHF	Ultra High Frequency
VS Code	Visual Studio Code

---

## 1.6 Definisjoner

---

Definisjon	Forklaring
char	Tegn
double	Dobbelt flyttall
float	Flyttall
int	Heltall
notch	Tegn
void	Verdiløs

---

## 2 Teoretisk rammeverk

### 2.1 Robotmanipulatoren

#### 2.1.1 Robotgeometri

Sammensetningen av en robotmanipulator (robot) består av en serie rigide lenker koblet sammen med ledd i enten en åpen eller en lukket kinematisk kjede. En åpen kinematisk kjede har lenker koblet sammen i unike endepunkter av kjeden og en lukket kinematisk kjede har lenker koblet sammen i ikke-unike endepunkter, altså en løkke. Roterende ledd betegnes ofte med “R” og prismatiske (lineære) ledd betegnes med “P”. Roboter blir ofte betegnet avhengig av hvilke typer ledd som utgjør roboten, og kan ta form som for eksempel en “RRR”-robot dersom roboten har tre roterende ledd (Spong, Hutchinson og Vidyasagar 2020, s. 5).

Nøyaktigheten i en robotmanipulator er et mål på hvor nært robotens ytterste del (endeffektoren) kan komme et gitt punkt i arbeidsområdet, mens repeterbarheten til roboten er hvor nært endeffektoren kan komme en tidligere besøkt robotkonfigurasjon (Spong, Hutchinson og Vidyasagar 2020, s. 10). Avvikene til endeffektoren måles vanligvis med posisjonsenkodere montert på leddene på roboten og er typisk ikke et direkte mål på hvor endeffektoren er i posisjon og orientering, men kan beregnes basert på geometrien til en matematisk modell av robotarmen. Nøyaktigheten til robotmanipulatoren kan være påvirket av beregningsfeil i mikrokontrollere som styrer roboten, maskineringsavvik ved produksjon av delene til roboten, fleksibilitet i lenkene og tilbakeslag i eventuelle gir i leddene (Spong, Hutchinson og Vidyasagar 2020, s. 10). På bakgrunn av avvikskildene over, ønskes en så høy stivhet i robotmanipulatorer som mulig.

#### 2.1.2 Dreiemoment

I en robotmanipulator er det ulike krefter som påvirker leddene, lenkene og de øvrige komponentene av roboten, avhengig av utformingen til roboten. Når en robot står i ro, kan de statiske kreftene som påvirker roboten beregnes. Dreiemomentet om en akse som følge av påvirkning av en kraft, kan finnes som (Ling, Sanny og Moebis 2021, s. 497):

$$\vec{\tau} = \vec{r} \times \vec{F}$$
$$\tau = |\vec{\tau}| = |\vec{r} \times \vec{F}| = r \cdot F \cdot \sin \phi$$

hvor

- $\vec{\tau}$  er dreiemomentet [Nm]
- $\vec{F}$  er kraftvektoren [N]

- $\vec{r}$  er posisjonsvektoren [m]

I tilfeller hvor flere momenter virker på et stivt legeme med en fast akse, kan likningen for rotasjonsdynamikk for skalare tilfeller defineres slik (Ling, Sanny og Moebs 2021, s. 502):

$$\sum_i \tau_i = I \cdot \alpha$$

hvor

- $\tau$  er dreiemomentet [Nm]
- $I$  er treghetsmomentet [kgm<sup>2</sup>]
- $\alpha$  er vinkelakselerasjonen [1/s<sup>2</sup>]

### 2.1.3 Treghetsmoment

Et treghetsmoment er i et rotasjonsdynamisk system et mål på rotasjonstreggheten til et stivt legeme, og noteres  $I$ . For å produsere en rotasjonsbevegelse trenger man en kraft  $F$  som virker på legemet, siden legemet med masse  $m$  ikke spontant av seg selv kan begynne å rotere.

Ulike stive legemer har ulike treghetsmomenter avhengig av legemets masse, utforming og plassering av rotasjonsaksen. Treghetsmomentet for en punktpartikkel uten utstrekning med masse  $m$  en avstand  $r$  fra rotasjonsaksen, er gitt som:

$$I = mr^2 \tag{1}$$

En tynn stang med masse  $m$ , lengde  $l$  og gjennomgående rotasjonsakse på den ene enden av stanga, har treghetsmoment gitt som:

$$I = \frac{1}{3}ml^2 \tag{2}$$

Et rektangulært legeme med rotasjonsakse gjennom senter av legemet, lengde  $l$  og bredde  $b$ , har treghetsmoment gitt som:

$$I = \frac{1}{12}m(l^2 + b^2) \tag{3}$$

### 2.1.4 3D-printing

Mæhlum (2021) definerer 3D-printing som: “3D-printing, eller additiv produksjon, er en fellesbetegnelse for teknikker som bygger objekter i fast materiale med utgangspunkt i en tredimensjonal digital modell. Arbeidet utføres av en 3D-printer.”.

Utførelse av 3D-printing kan variere noe basert på hvilket materiale som benyttes. Materiale på en “Fused deposition modeling”-printer (FDM-printer) kan være kunstige polymer som “polylactic acid” (PLA) eller “polyethylene terephthalate glycol” (PETG). I en FDM-3D-printer for plastmaterialer blir materialet varmet opp og ekstrudert gjennom en dyse der det oppvarmede materialet fester seg på en overflate lag for lag som til sammen utgjør den digitale modellen en ønsker å printe (Mæhlum 2021).

PLA-materialet er et prisgunstig og ikke-giftig material som kommer i et mangfold av farger- og materialegenskaper, har god tilgjengelighet på markedet og gir minimale utslipp av skadelige gasser og partikler under printing (All3DP 2022). I tillegg fremstilles PLA-materialet av fornybare råvarer som mais, som gir relativt hurtig nedbrytningstid (Klepp 2020). Ulempene med PLA er derimot at materialet er sprøtt med forholdsvis lav bestandighet for mekanisk belastning og tåler ikke store bøyninger eller vridninger (Klepp 2020). Materialet er ikke varmebestandig og kan deformeres i nærheten av varmekilder.

PETG-materialet skiller seg derimot fra PLA-materialet ved å ha høyere bestandighet mot mekanisk belastning og temperatur, er mer fleksibelt og har høyere varighet (Frey og Locker 2021). Ulempene med PETG er at materialet har ofte høyere pris enn PLA, kan oftere feile underveis i printingen, frigir flere gasser og partikler under printing, tiltrekker seg fuktighet fra luften som degraderer materialet og det har svært lang nedbrytningstid på grunn av fremstillingen fra ikke-fornybare råvarer (Frey og Locker 2021).

### **2.1.5 Polyoxymetylen (POM)**

POM er en type plast som egner seg godt til maskinbearbeiding på grunn av sin høye mekaniske styrke, stivhet og overflatehardhet (VINK 2022). Egenskapene til POM gjør den enkel å behandle på en CNC og det kan for eksempel lett maskineres ut gjenger i materialet. Andre fordeler med POM er store utmattelsesegenskaper og god elektrisk isoleringsevne. Ulempene med POM er blant annet at UV-stråling, varmt vann og damp vil endre egenskapene til materialet over tid (VINK 2022).

### **2.1.6 Dataassistert konstruksjon (CAD)**

Dataassistert konstruksjon er en metode for å designe 3D-modeller som kan brukes til å produsere deler, for eksempel ved hjelp av en 3D-printer. Autodesk Fusion 360 er en skybasert CAD-programvare med mulighet for eksportering av modeller til “stl”- eller “3mf”-format. Filformatet “stl” (Stereolithography) er et format for å beskrive overflaten til en modell ved hjelp av et maskenett bestående av trekkanter, som kan importeres i en programvare for redegjøring til printing på en 3D-printer (Library of Congress 2019). “3mf”-formatet kan i tillegg til å beskrive modellen også gi informasjon om materialtype, farger, tekstur og enhetstypen som er brukt i modellen (Kauppila 2021).

## 2.2 Antennearray

### 2.2.1 Radiokommunikasjon

Radiokommunikasjon er en form for trådløs kommunikasjon som tar i bruk modulerte elektromagnetiske bølger til å overføre signaler. Elektromagnetiske bølger trenger ikke et medium for å utbre seg, men ofte er luft propogasjonsmediumet for signalene. Det vil oppstå et magnetisk felt i en antenne i respons med den påtrykte spenningen.

Bølgelengden er en viktig parameter som avgjør frekvensbåndet som signalet sendes i, og er definert ved ( $\lambda = \frac{c}{f}$ ). Bølgelengden er 12.5 cm i luft for  $f = 2.4$  GHz, med de overharmoniske komponentene ved 6.25 cm ( $\frac{\lambda}{2}$ ) og 3.125 cm ( $\frac{\lambda}{4}$ ). Bluetooth jobber derfor i UHF-området etter ITU standarden (ITU Radiocommunication Sector 2015b), og S-båndet etter IEEE-standarden IEEE 1984.

Elektromagnetiske bølger er transversale bølger som består av et elektrisk felt og et magnetisk felt. Orienteringen til det elektriske og magnetiske feltet betinger polariseringen til signalet. Dersom det elektriske feltet ligger parallelt med jordoverflaten er signalet horisontalpolarisert, men dersom feltet står vinkelrett på jordoverflaten er det vertikalpolarisert. Noen antenner har sirkulærpolarisering, der det elektromagnetiske feltet roterer ved propagering av signalet. Disse kan rotere mot høyre og venstre og benevnes RHCP og LHCP respektivt (Frenzel 2015, s. 510). Det vil oppstå tap i signalstyrke dersom antennene har feil polarisering relativt til hverandre. Ved bruk av lineære antenner vil antenner med samme polarisering, ideelt sett, ha 0 dB polariseringstap. Dersom det brukes en vertikalpolarisert antenne og en horisontalpolarisert antenne, vil det elektriske og magnetiske feltet være rotert 90°, og mottakerantennen vil derfor ikke motta signaler. I et system med en sirkulær- og en lineærpolarisert antenne vil man derfor kunne overføre signaler for alle orienteringer av antennene, rundt samme akse. Det vil oppstå et tap på -3 dB på grunn av endringen av det elektromagnetiske feltet utstrålt fra den sirkulærpolariserte antennen (Galuscak og Hazdra 2005).

For bruk av antenner i praksis vil det oppstå støy ved målinger av signaler. Det vil derfor være en nedre grense for det mottatte signalet. For signalnivåer under den nedre grensen, vil ikke systemet skjelne mellom støy og radiosignaler. I signalteori benevnes dette som støygulvet, og vil være en sum av alle støykildene i et gitt målesystem.

Båndbredden er frekvensspekteret som antennen kan operere i. Frekvensspekteret er derfor differansen mellom den høyeste og den laveste grensefrekvensen (Frenzel 2015, s. 18). Båndbredden til antennen kan finnes ved å pare returtapet med en kjent VSWR-verdi. Ved å lese av frekvensen på punktene for VSWR, kan båndbredden utregnes ved hjelp av følgende formel:

$$BW = f_{upper} - f_{lower} \quad (4)$$

hvor:



- $BW$  er båndbredden [Hz]
- $f_{upper}$  er øvre grensefrekvens [Hz]
- $f_{lower}$  er nedre grensefrekvens [Hz]

## 2.2.2 Elektriske ledere i radiokommunikasjon

I ledere kan det oppstå ohmsk tap, da en elektrisk leder ved vekselstrøm vil ha en tendens til å samle seg i det ytre laget av lederen. Dette kalles “skin depth” (Wheeler 1977) og er avhengig av frekvensen til vekselspenningen. Strømtettheten vil minke eksponentielt mot senteret av lederen, som reduserer det totale tverrsnittet til det ledende materialet som strømmen ledes i. Dette vil bidra med dielektrisk tap som vil påvirke impedansen gitt ved konstanten  $\epsilon_r$ .

Impedansmatchingen for PCB-utlegget er betinget av tykkelsen på microstrip-linjene. For å minimere diskontinuitet er det kritisk at kontaktene og microstrip-linjene på PCB-utlegget har lik impedans. Den nominelle impedansen til microstrip-linjene påvirkes av flere parametere og er gitt ved:

$$Z = \frac{87}{\sqrt{\epsilon_r + 1.41}} \cdot \ln \left( \frac{5.98 \cdot H}{0.8 \cdot W + T} \right) \quad (5)$$

hvor:

- $Z$  er impedansen til microstrip-linjen [ $\Omega$ ]
- $\epsilon_r$  er den dielektriske konstanten til FR4
- $H$  er tykkelsen til substratet [mm]
- $T$  er tykkelsen til kobberlaget på PCB-utlegget [mm]
- $W$  er bredden på microstrip-linjen [mm]

Refleksjonskoeffisienten beskriver det reflekterte signalet på grunn av avvik i impedansmatching (Nikitin mfl. 2005). Refleksjonskoeffisienten vil derfor beskrive hvor mye av spenningen som blir reflektert i forhold til inngangsspenningen som blir påtrykt overføringslinjen. Refleksjonskoeffisienten finnes ved å se på forholdet mellom det inngående signalet og utgående signalet i antennen. Det vil dermed være mulig å analysere antennens evne til å motta og sende signaler, med hensyn på impedansmatchingen (Nikitin mfl. 2005). Refleksjonskoeffisienten  $\Gamma$  er gitt ved:

$$\Gamma = \left( \frac{Z - Z_0}{Z + Z_0} \right) \quad (6)$$

hvor:

- $\Gamma$  er refleksjonskoeffisienten

- $Z$  er lastimpedansen [ $\Omega$ ]
- $Z_0$  er den karakteristiske impedansen [ $\Omega$ ]

### 2.2.3 Linkbudsjett

Et linkbudsjett blir ofte brukt for å kartlegge tap og vinninger i antennesystemer. Et linkbudsjett inneholder parameterene som påvirker signalet under sending og mot-taking, der de viktigste parameterene er antenneforsterkning, tap ved propagering i fritt rom og passive tap i kontakter og ledninger (Yaacob mfl. 2017). Ved å regne ut alle tap og vinninger i systemet kan man estimere signalstyrken på det mottatte signalet. Den forventede signalstyrken kan videre brukes for å validere systemet for bruk med de gitte komponentene og omgivelsene.

Den generelle ligningen for et linkbudsjett er gitt ved:

$$P_{RX} = P_{TX} + G_{TX} - L_{TX} - L_{FSPL} - L_M + G_{RX} - L_{RX} \quad (7)$$

hvor:

- $P_{RX}$  er mottatt signalstyrke [dBm]
- $P_{TX}$  er utstrålet signalstyrke [dBm]
- $G_{TX}$  er gain for senderantennene [dBi]
- $L_{TX}$  er tap i senderantennen, som kontakt- og kabeltap [dB]
- $L_{FSPL}$  er friromstap [dB]
- $L_M$  er diverse tap som polariseringstap og fading [dB]
- $G_{RX}$  er gain for mottakerantennen [dBi]
- $L_{RX}$  er tap i mottakerantennen, som kontakt- og kabeltap [dB]

FSPL er et akronym for “Free Space Path Loss”, som beskriver friromstapet ved propagering i fritt rom. Dette skjer på grunn av spredningen av det elektromagnetiske feltet, der avstanden betinger styrken på feltet ved mottakelse av signalene (ITU Radiocommunication Sector 2015a). FSPL vil øke kvadratisk med distansen og er definert ved:

$$FSPL = 20 \cdot \log(d) + 20 \cdot \log(f) - 147.55 = 20 \cdot \log(d_{\text{km}}) + 20 \cdot \log(f_{\text{GHz}}) + 92.45 \quad (8)$$

hvor:

- $FSPL$  er “Free Space Path Loss” [dB]

- $d$  er distansen [m]
- $f$  er frekvensen [Hz]
- $d_{\text{km}}$  er distansen [km]
- $f_{\text{GHz}}$  er frekvensen [GHz]
- $Gr$  er gain for senderantennen [dBi]
- $Gt$  er gain for mottakerantennen [dBi]

#### 2.2.4 Nær- og fjernfelt

En antenne vil omgjøre et elektrisk signal til elektromagnetiske bølger i rommet. Området rundt antennen består av de tre feltene fjern-, strålings- og nærfelt, der feltstyrkene vil avta forskjellig med en rate på  $\frac{1}{r}$ ,  $\frac{1}{r^2}$ ,  $\frac{1}{r^3}$ , for de respektive feltene, hvor  $r$  er avstanden fra antennen. Feltene avtar på grunn av akselerasjon og de-akselerasjon av elektriske ladninger som blir utstrålt fra antennen (Johnson, Ecker og Hollis 1973). I fjernfeltet vil den elektriske og magnetiske komponenten i signalet være avhengige av hverandre, der en endring i det ene signalet vil resultere i en endring i det andre signalet.

Distansen fra senderkilden og de elektromagnetiske regionene (Nikravan og Schantz 2013) vil være definert ved:

$$R_{nearfield} = 0.62 \cdot \sqrt{\frac{D^3}{\lambda}} \quad (9)$$

$$R_{farfield} = \frac{2D^2}{\lambda} \quad (10)$$

$$0.62 \cdot \sqrt{\frac{D^3}{\lambda}} < R_{transitionfield} < \frac{2D^2}{\lambda} \quad (11)$$

hvor:

- $R_{nearfield}$  er distansen for nærfeltet [m]
- $R_{transitionfield}$  er distansen for overgangsfeltet [m]
- $R_{farfield}$  er distansen for fjernfeltet [m]
- $D$  er diameteren til antennen [m]
- $\lambda$  er bølglengden [m]

Regionene i det elektromagnetiske feltene vil ha en innvirkning på målingene som blir utført på en gitt antenne. Det er derfor hensiktsmessig å utføre målinger i fjernfeltet, da kommunikasjonen primært foregår i dette feltet.

### 2.2.5 Flerveisinterferens

Interferens i en mottakerantenne kan skyldes refleksjoner av signalet på overflater som vil oppstå under overføring av de elektromagnetiske signalene. Interferens vil enten resultere i forsterkning eller demping av det mottatte signalet, avhengig av fasen til det reflekterte signalet. En slik type interferens kalles flerveisinterferens, og vil i rom som ikke er ekkofrie være et uunngåelig fenomen (Wen og J. 1990).

Ved å beregne Fresnel-soner for den gitte sendefrekvensen, kan man gjøre et estimat på kritiske soner som vil føre til interferens til det mottatte signalet. Det finnes flere Fresnel-soner for et gitt system, med vekslende destruktive og konstruktive soner avhengig av fasen og perioden til signalet (Würth 2019).

Radiusen til Fresnel-sonen er gitt ved :

$$r = 17.32 \cdot \sqrt{\frac{d}{4 \cdot f}} \quad (12)$$

hvor:

- $r$  er radiusen til Fresnel-sonen [m]
- $d$  er avstanden mellom antennen [km]
- $f$  er frekvensen [GHz]

### 2.2.6 Effektnivå i antennesystemer

En desibel er en tiendedels Bel og beskriver endringen, eller forholdet, mellom en verdi og en referanse. Desibel er en dimensjonsløs enhet og er i antennteknikk brukt til å beskrive tap og vinnings til signalet gjennom et antennesystem. En desibel er gitt ved:

$$dB = 10 \cdot \log_{10} \left( \frac{P_{ut}}{P_{inn}} \right)$$

hvor:

- $dB$  er desibel [dB]
- $P_{ut}$  er målt utgangseffekt [W]
- $P_{inn}$  er påtrykt effekt [W]

Måleenheten dBm brukes til å beskrive forholdet mellom en signaleffekt og en referanseffekt på 1 mW. Dette brukes for å måle signalstyrke, der fortegnet til den

målte signalstyrken er, for de fleste applikasjoner, negativ og signalstyrken er bedre dersom den blir målt nærtliggende 0 dBm (1mW). Verdier kan også være positive ved bruk av antenner med høy forsterkning, der den målte effekten er over 1 mW (Bardwell 2002). Denne enheten blir brukt i antennesystemer fordi dBm er en praktisk måte å uttrykke absolutt effekt for lave og høye verdier og er gitt ved:

$$dBm = 10 \cdot \log_{10} \left( \frac{P}{1 \text{ mW}} \right)$$

hvor:

- $dBm$  er desibel-milliwatt [dBm]
- $P$  er målt effekt [W]

For å beskrive forsterkningen brukes måleenheten dBi. Denne måleenheten brukes til å beskrive forholdet til antennegainen i forhold til en isotropisk antenne med 0 dBi (Cisco 2007). Antenneforsterkningen brukes til å beskrive forsterkningen til signalet som blir propagert, men også direktiviteten til antennen. dBi er gitt ved:

$$dBi = 10 \cdot \log_{10}(G)$$

hvor:

- $dBi$  er desibel-isotropic [dBi]
- $G$  er antennens forsterkningsfaktor

Antenneforsterkningen kan også utledes fra HPBW, der HPBW kan avleses fra strålingsdiagrammet til antennen ved -3 dB (Cisco 2007). Antennegainen kan videre utledes fra følgende formel:

$$B = \frac{203}{(\sqrt{10})^x} \tag{13}$$

hvor:

- $B$  er HPBW for planet [°]
- $x$  er antenneforsterkningen i dB delt på 10 [ $x = \frac{dB}{10}$ ]

Ved å løse for  $x$ , kan det utledes en formel for antenneforsterkningen ( $G_{TX}$ ) uttrykt i dBi:

$$G_{TX} = 20 \cdot \log \left( \frac{203}{B} \right) \tag{14}$$

### 2.2.7 Asimut- og elevasjonsplan

Antenner propagerer elektromagnetiske bølger i det tredimensjonale rommet. Det er derfor hensiktsmessig å definere det horisontale og det vertikale planet. Det horisontale planet er definert som det asimutale planet og vil danne asimutvinkelen østover fra nordpunktet ( $\theta$ ) fra en toposentrisk origo. Det vertikale planet benevnes som elevasjonsplanet, som vil gi vinkelen mellom det asimutale planet og et objekt ( $\phi$ ) (Cisco 2007).

For antennteknikk vil asimut- og elevasjonsplanene bli brukt til å beskrive orienteringen til antennen i et sfærisk koordinatsystem. Et eksempel på beskrivelser av orientering i et sfærisk koordinatsystem, er et strålingsdiagram. Strålingsdiagrammet kan visualisere utstrålingen til antennen for de ulike planene (Cisco 2007).

### 2.2.8 Antennetyper

Strålingsdiagrammer brukes for å beskrive formen til det elektromagnetiske feltet som blir utstrålt eller mottatt fra antennen. Karakteristikken til antennen vil avgjøre formen på det elektromagnetiske feltet, der direktiviteten og antenneforsterkningen vil beskrive hvor retningstyrt antennen er (Cisco 2007). I radiokommunikasjon brukes det to typer antenner: omnidireksjonelle og retningsstyrte antenner. Disse har forskjellige bruksområder og karakteristikk og defineres ut i fra strålingsmønsteret.

En omnidireksjonell antenne vil ha en isotropisk utstråling i asimutplanet (Cisco 2007). Disse antennene blir anvendt i systemer som kringkasting av radio og tv-signaler, og vil utstråle effekten uniformt i asimutplanet. Mono- og dipolantenner er eksempler på omnidireksjonelle antenner som utstråler lik effekt i asimutplanet.

Direksjonelle antenner vil sende og motta signaler med høyere effekt i spesifikke vinkler, avhengig av utformingen på antennen (Cisco 2007). De vil derfor ha bedre ytelse i disse retningene, fremfor omnidireksjonelle antenner som sender og mottar signalene likt i asimutplanet. Yagi-Uda- og parabolantenner er eksempler på retningsstyrte antenner som vil utstråle høyere effekt i spesifikke retninger. Antenneforsterkningen til disse antennene er proporsjonal med størrelsen av konstruksjonen (Sophocles 2016, s. 1067).

Ved beregninger av direktiviteten til en antenne brukes det en isotropantenne som referanse. Dette er en teoretisk antenne som propagerer radiobølgene uniformt i rommet, som er beskrevet i 2.2.6. Dette impliserer at den isotropiske antennen vil ha en teoretisk antenneforsterkning på 0 dBi. I praksis vil alle antenner være direkte i forskjellig grad, fordi en isotropisk antenne kun er en teoretisk antenne (Cisco 2007). De direksjonelle antennene kjennetegnes med høy antenneforsterkning med en tilspisset hovedlobe der signalet vil påvirkes mindre av støy og andre innvirkninger. Den tilspissede hovedloben gjør direksjonelle antenner mer presise og egnet til trådløs kommunikasjon over lengre distanser, så de kan derfor konsentrere signalet som enten blir sendt eller mottatt i retningen som antennen er orientert. Grunner gjensidighet i strålingsdiagrammene for mottaking og sending av signaler,

vil en direksjonell antenne forsterke både det sendte og det mottatte signalet likt. Direksjonelle antenner blir ofte brukt der retning eller vinkel er en viktig parameter for brukbarheten.

Ved å kombinere to eller flere antenner, kan man konstruere et antennearray som gir ulike egenskaper basert på antenneytten som er brukt, og posisjonering av antennene i forhold til hverandre. De individuelle antennene, også kalt elementer, vil dermed fungere som én enkel antenne. Ved å kombinere flere enkeltantenner kan høyere grad av direktivitet oppnås uten de dimensjonale ulempene en høydirektiv antenne vil medbringe.

Nyquistfrekvensen  $f_s$  for en gitt frekvens  $f$  vil være gitt ved  $f_s = 2f$ . Aliasing kan oppstå dersom avstanden mellom elementene blir dimensjonert på ukorrekt vis med hensyn på  $f_s$  (Sophocles 2016, s. 1104). Ukorrekt avstand mellom elementene vil forårsake store sidelobes på størrelse med hovedloben og er en egenskap som er uønskelig ved design av antennearray. det vil derfor være hensiktsmessig å bruke en avstand mellom elementene lik bølgelengden til  $f_s$ . Avstanden kan finnes ved:

$$d = \frac{c}{f_s} = \frac{c}{2f} = \frac{\lambda_f}{2} \quad (15)$$

hvor:

- $d$  er avstanden mellom antennene i antennearrayet [m]
- $c$  er lysets hastighet [ $\frac{m}{s}$ ]
- $f_s$  er Nyquistfrekvensen [Hz]
- $f$  er den påtrykte frekvensen [Hz]
- $\lambda_f$  er bølgelengden til den påtrykte frekvensen [m]

Den synlige regionen til et lineært antennearray vil være  $180^\circ$  (Sophocles 2016, s. 1103), dette vil gi et bølgetall  $\psi$  på  $-\frac{2\pi d}{\lambda} \geq \psi \geq \frac{2\pi d}{\lambda}$ , der den totale variasjonen av  $\psi$  vil være gitt ved:

$$\psi_{visible} = 2 \left( \frac{2\pi d}{\lambda_f} \right) \quad (16)$$

- $\psi_{visible}$  er variasjonen av bølgetall [ $m^{-1}$ ]
- $\lambda_f$  er bølgelengden [m]

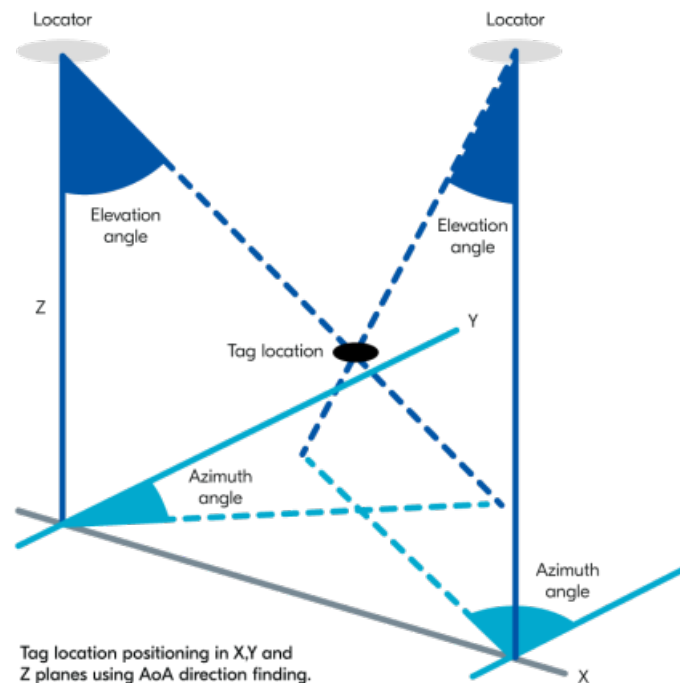
Ved sette inn en avstand  $d = \frac{\lambda_f}{2}$  i formel (16), kan endringen av bølgetall uttrykkes i  $\pi$ :

$$\psi_{visible} = 2 \left( \frac{2\pi}{\lambda} \right) \cdot \frac{\lambda_f}{2} = 2\pi \quad (17)$$

En variasjon av  $\psi$  på  $2\pi$  tilsvarer ett Nyquistintervall ( $-\pi \geq \omega \geq \pi$ ), og det vil derfor ikke oppstå sidelobes eller aliasing av signalet. Det er derfor vanlig å ta i bruk en avstand på  $\frac{\lambda}{2}$  ved konstruksjon av et antennearray (Sophocles 2016, s. 1103).

### 2.2.9 Søkemetoder

Ved bruk av et fasestyrt antennesystem vil det være mulig å anvende systemet til radiopeiling ved hjelp av fasemålinger og tid for de individuelle antennene. Disse metodene heter “Angle of Arrival” (AoA) og “Angle of Departure” (AOD) og måler differansen av tid og fase mellom signalene.. Bluetooth AoA er en presis lokalisering-metode for innendørs bruk på grunn sanntidslokalisering av Bluetooth-enheter som kan gjøres (Hollander 2019). Bluetooth AoA-metoden er illustrert i figur 1(Nordic Semiconductor 2022):



Figur 1: Illustrasjon av Bluetooth-AoA-prinsippet

RSSI er et akronym for “Received Signal Strength Indicator” og uttrykker signalstyrken for mottatt signalstyrke i dBm, som tidligere beskrevet i 2.2.6. Ved bruk av en retningsstyrt antenne vil målinger ved hovedloben ha en høyere RSSI når antennen er på linje med senderen. Maksimalpunktene som blir målt vil derfor korrespondere med vinklen senderantennen er orientert. Nøyaktigheten man kan forvente er derfor betinget av HPBW til hovedloben.

I motsetning til et maksimum RSSI-søk, er det mulig å bruke nullpunkter for radiopeiling. Nullpunktene i et strålingsdiagram er ofte mer veldefinerte enn hovedloben for en gitt antenne, og vil derfor kunne gi et mer nøyaktig anslag på retningen til sendeantennen. Ulempen med denne metoden er at det ikke vil være mulig å skille



mellom sendekilder og reelle nullpunkt. Dersom en detekterer falske nullpunkt øker sannsynligheten for å peile etter reelle nullpunkt.

Ved å kombinere de overnevnte RSSI-metodene kan man få en mer nøyaktig peiling. Denne metoden kalles sumforskjell-måling (Bonafacic, Janculan og Majurec 2007). Dersom man sammenligner høy RSSI-verdi med et målt nullpunkt, kan man verifisere at et reelt nullpunkt er funnet. Dette kan utføres ved hjelp av et antennearray der signalet blir forgrenet til en linje i fase og en i motfase. Forgreningen kan gjøres ved hjelp av et PCB-utlegg, hybridkobler eller ved bruk av splitter og faseskiftere. Med et slikt system kan det søkes etter høye RSSI-verdier for målinger i fase ( $\Sigma$ ) og samtidig etter et veldefinert nullpunkt ved måling i motfase ( $\Delta$ ). Ved måling i motfase vil et ideelt signal gi målt verdi lik 0, dersom de er orientert direkte mot senderen, på grunn av faseforskjellen på  $180^\circ$ . Ved måling i fase vil RSSI-verdien være den høyeste når antennen er orientert direkte mot sendeantennen. Dette er fordi det ikke vil være en faseforskjell mellom signalene. Følgende to formler viser utgangssignalet fra antennesystemet ved addering sinusformede signaler Lyons 2011 i fase ( $\Sigma$ ) og motfase ( $\Delta$ ):

$$y(t)_\Sigma = f_1(t) + f_2(t) = A \cos(\omega t + \phi) + B \cos(\omega t + \phi) \quad (18)$$

$$y(t)_\Delta = f_1(t) - f_2(t) = A \cos(\omega t + \phi) + B \cos(\omega t + (\phi + \pi)) \quad (19)$$

hvor:

- $A$  er amplituden til signal 1 [V]
- $B$  er amplituden til signal 2 [V]
- $\omega$  er vinkelfrekvensen til signalet [ $\frac{\pi}{\text{rad}}$ ]
- $t$  er tid [s]
- $\phi$  faseforskyvning [ $^\circ$ ]

I et sigma-delta-søk vil det hovedsaklig være målingene i motfase som brukes for å orientere seg mot sendeantennen. Ved å sammenligne et målt nullpunkt i motfase og fase, kan nullpunktet lokaliseres ved å utføre et søk i rommet. Dersom nullpunktet er et reelt nullpunkt, vil målingene i fase gi en lav RSSI.

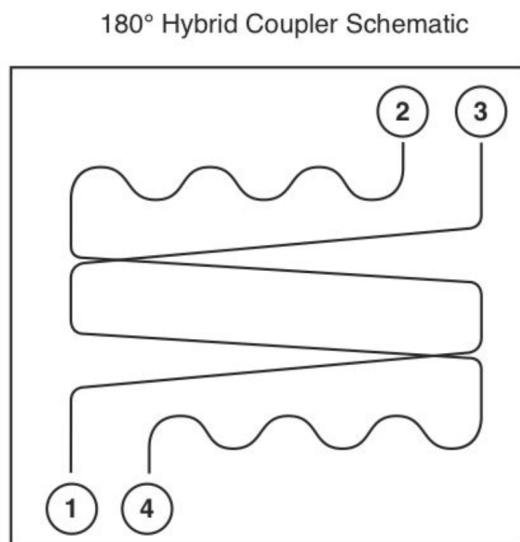
Dersom det innsettes verdier i formel (19), vil dette gi en null dersom antennen står normalt på senderantennen ( $\phi = 0^\circ$ ):

$$y(t) = A \cos(\omega t) + B \cos(\omega t + (\phi)) = 0 \quad (20)$$

### 2.2.10 Hybridkobler

En hybridkobler er en passiv komponent og kan utføre en rekke operasjoner som splitting, sammenslåing og faseforskyvning av signaler (Krytar 2022). Hybridkoblere er

ofte designet for et gitt frekvensbånd og oppnåes ved å modifisere overføringslinjene til signalene. Ved  $180^\circ$  faseforskyvning vil den ene overføringslinjen forlenges med  $\frac{\lambda}{2}$  for å oppnå faseforskyvningen. En  $90^\circ$ -kobler vil gjøre tilsvarende, men med en forlengelse av den ene overføringslinjen på  $\frac{\lambda}{4}$ . Figur 2 (Krytar 2022) viser skjematikken til en  $180^\circ$  hybridkobler.



Figur 2: Skjema for en  $180^\circ$ -kobler

## 2.3 Integrert system

### 2.3.1 Komponenter

#### Hybrid steppermotor

Hybride steppermotorer er en type synkronmotorer med innlagte tenner på rotoren som roterer i diskrete trinn som en reaksjon på en elektrisk puls påtrykt kontrollenheten til steppermotoren (Kothari og Nagrath 2018). Antallet trinn per omdreining er gitt av antallet tenner på rotoren til motoren. Typiske trinn per omdreining for en hybrid steppermotor er 200 og 400, med en trinnvinkel på henholdsvis  $1.8^\circ$  og  $0.9^\circ$ . Hybride steppermotorer er ofte anvendt i 3D-printere, kopimaskiner og numerisk kontrollerte verktøy som CNC-er (Kothari og Nagrath 2018).

Fordelene med en hybrid steppermotor er muligheten for den veldefinerte diskrete trinnvinkelrotasjonen og fraværet for behovet om en tilbakemeldingsløkke for å vite posisjonen til motorakslingen. Den diskrete trinnvinkelen gjør det mulig å kalkulere motorakslingens posisjon gjennom å telle pulser som er påtrykt motorviklingene, så lenge en holder seg innenfor motorens oppgitte grenseverdier. Ulempene med den hybride steppermotoren er at den veier mye i forhold til dimensjonene, har behov for en digital kontrollenhet og har lavt dreie- og holdemoment. I tillegg kan motoren miste trinn i rotasjonen dersom en operer med høy pulsrate på kontrollenheten til steppermotoren (Kothari og Nagrath 2018).

## Servomotor

En industriell servomotor består ofte av en veksel- eller en likestrømsmotor med børste eller børsteløs utforming. I tillegg er servomotoren ofte integrert med en enkoder for å angi posisjonen til motorakslingen og eventuelt giring (Gastreich 2018). En annen type servomotorer er sammensatte løsninger bestående av en børste- eller en børsteløs DC-motor, en kontrollkrets, giring, et potensiometer for innebygd tilbakekoblingsløyfe og regulator. Den siste løsningen er etablert i miljø der vinkelregulering blir gjort basert på et tilført PWM-signal.

Fordelene med en ikke-industriell DC servomotor er at den er forholdsvis sterk sammenlignet med størrelsen og vekten, kan ha høyt dreiemoment på grunn av intern giring og er ikke like kompleks å styre som steppermotoren. Ulempene med en slik servomotor er at den ikke har mulighet for kontinuerlig rotasjon, så en innfører en vinkelbegrensning i arbeidsområdet til motoren. Det er heller ikke tilbakekobling for vinkelposisjon slik at motoren kan kommunisere med mikrokontrolleren om den har nådd ønsket posisjon eller ikke.

## Enkoder

En inkrementell enkoder kan brukes til å gi informasjon om posisjon og hastighet til en roterende akse, ved å telle antall steg under rotasjon. En inkrementell enkoder kan for eksempel ha tre utganger: A-, B- og indekskanalen. Utgang A og B har en offset på  $90^\circ$  mellom hverandre, der antall firkantpulser som avgis per rotasjon oppgis i pulser per omdreining (PPR). Ved å se på forskjellen mellom kanal A og B kan rotasjonsretningen bestemmes og relativ posisjon kan utledes. Hastigheten til den roterende akselen kan bestemmes ved å se på frekvensen til pulsene fra enkoderen (Craig udatert). Indekspulsen, avhengig av utformingen på enkoderen, kan gi en puls per omdreining og benyttes som referansepunkt på en  $360^\circ$ -rotasjon.

Et alternativ til den inkrementelle enkoderen er den absolutte enkoderen. En absolutt enkoder måler posisjon på en roterende akse ved hjelp av binære verdier inngravert på en ring i form av Gray-kode. Gray-koden benyttes for å sikre at ikke oppstår feilavlesninger ved overgangene mellom to posisjoner (DYNAPAR udatert). Absolutte enkodere kan kjøpes med ulike oppløsninger, men til høyere oppløsning en absolutt enkoder skal ha, til høyere kostnad har den. En absolutt enkoder med tilsvarende oppløsning som en inkrementell enkoder er oftest betydelig dyrere.

## Lasere i laserklasse 3R

En laser i laserklasse 3R har maksimal effekt på 5 mW og er i følge Direktoratet for strålevern og atomsikkerhet (2019): “[...] sikker å sjå på berre i korte tidsintervall fordi naturlege reaksjonar for å vike unna som å blunke eller å sjå bort forhindrar eksponeringstid over ca.  $\frac{1}{4}$  sekund.”

Ved bruk av lasere i laserklasse 3R skal laseren være utstyrt med gul/sort advarselsmerking, samt opplysninger om klassifisering og fysiske egenskaper til laseren (Direktoratet for strålevern og atomsikkerhet 2019). I tillegg skal det gjøres en skriftlig risikovurdering knyttet til strålebruken og nye aktiviteter skal ikke settes i gang før risikovurderingen er gjennomført og forebyggende tiltak er iverksatt (Direktoratet for strålevern og atomsikkerhet 2019). Dersom risikovurderingen viser at det finnes en uakseptabel risiko for brukerne, skal det iverksettes forebyggende tiltak for å redusere risikoen. De forebyggende tiltakene kan være å utforme arbeidsrutiner, gi nødvendig informasjon om bruk og å bruke tilstrekkelig verneutstyr og materialer.

## **Svitsj**

En svitsj for radiosignaler (RF) tar imot flere RF-signaler og har evnen til å svitsje mellom hvilket av disse signalene som sendes ut fra en utgang. For å forenkle utviklingsprosessen kan svitsjen kjøpes ferdig montert på et evalueringskort, med tilhørende egenskaper for tilkobling for kontroll og inn- og utgangsporter. Ved å tilføre logisk høy eller lav på pinnene til evalueringskortet, endres hvilken RF-inngang som sendes på utgangen til svitsjen.

## **Utviklingskort for SoC**

Et utviklingskort består av en mikrokontroller montert på et kretskort med muligheter for oppkobling og bruk av funksjoner på mikrokontrolleren. Den har som hensikt å forenkle utviklingsprosessen ved å tilby mange funksjoner på et praktisk kort. Utviklingskort egner seg for prototyping siden det gir mulighet for å teste alle funksjonene tilgjengelig i et SoC uten å måtte utvikle eget kretskort.

### **2.3.2 nRF Connect for Desktop**

nRF Connect for Desktop er en samling programvare fra Nordic Semiconductor som gjør det enklere å utvikle programkode for nRF-enheter (Nordic Semiconductor udatert). Den består av flere verktøy utformet for å gjøre utvikling, testing og optimalisering av nRF-enheter enklere. Blant programvarene i nRF Connect for Desktop finnes “toolchain manager”, som forenkler installering og vedlikehold av nRF Connect SDK-et som brukes ved utvikling av programvare for nRF-enheter.

### **2.3.3 nRF Connect SDK periferaler**

## **Interrupt**

En “interrupt” på en mikrokontroller er en forespørsel fra hardware eller software som midlertidig stopper kjøring av hovedprogrammet for å kjøre en kort operasjon,

for så å gå tilbake til hovedprogrammet igjen. Når et interrupt-signal blir generert, vil systemet håndtere dette etter en definert “Interrupt Service Routine” (ISR). Interrupt-er har høy prioritet i programkoden til mikrokontrolleren, slik at når et signal kommer inn, stanses andre prosesser på mikrokontrolleren slik at ISR-en kan prosessere interrupt-en og utføre nødvendige handlinger. Etter utførelsen av ISR går prosessoren tilbake til å kjøre hovedprogrammet.

## **GPIO**

“General-purpose input/output” er en pinne på utviklingskortet som kan konfigureres slik som programmereren ønsker. Ved programmering kan funksjonen til disse pinnene bestemmes etter hva som er nødvendig å bruke den til, og kan ofte også endres underveis i programmet. GPIO-er gjør det mulig for utviklingskortet å enkelt kommunisere med andre komponenter og enheter, som sensorer eller enkodere. Det forenkler testing og prototyping siden oppkobling er raskt og enkelt, og protokoller som behøves for kontroll kan programmeres i software (Nordic Semiconductor 2021b).

## **PPI**

“Programmable peripheral interconnect” (PPI) er en måte å automatisere kontroll av ulike periferale. Ved å forhåndskonfigurere ønsket oppførsel mellom periferale kan handlinger bli utført uavhengig av CPUen. På denne måten unngås det at CPU må involveres hver gang en hendelse inntreffer, og det forhindrer feil som kan oppstå om CPU er opptatt med annet arbeid når det ønskes at en handling skal utføres (Nordic Semiconductor 2021e).

## **GPIOE**

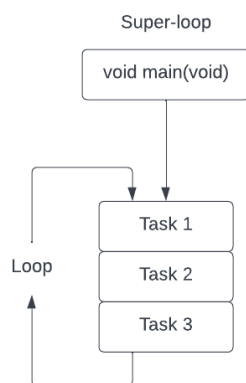
GPIO Tasks and Events (GPIOE) er en modul i nRF Connect SDK-et som gjør det mulig å få tilgang til GPIO-pinner gjennom bruk av “tasks” og “events”. Ved å kombinere GPIOE med PPI-systemet kan handlinger bli opprettet når pinner endrer status og handlinger kan bli utført gjennom tasks og events (Nordic Semiconductor 2021c).

## **Timer**

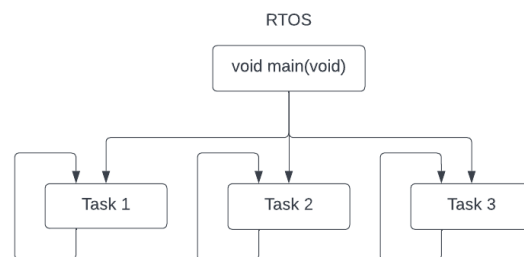
En timer på mikrokontrolleren gjør det mulig å holde styr på tiden. Dette er hjelpsomt ved oppgaver som er avhengig av tidsfrister eller for handlinger som utføres periodisk. Timeren teller etter en frekvens som er definert for timeren, som er avhengig av mikrokontrolleren. Hvert intervall i denne tellingen kalles “tics”. Ved å bruke timere sammen med andre periferale kan handlinger, som for eksempel å generere et PWM-signal, bli utført med bestemte tidsintervaller.

### 2.3.4 Real-time operating system (RTOS)

De fleste enkle programmer utviklet for mikrokontrollere vil følge en super-loop struktur. Strukturen går ut på at etter initialisering, kjører programmet i en løkke som ofte består av innhenting av data, gjøre beregninger og utføre en handling basert på resultat (se figur 3). For å håndtere uventede handlinger brukes en ISR for å midlertidig bryte løkken. Denne metoden krever ofte lite ressurser, kjører raskt og egner seg bra for enkle programmer (DevAcademy udatert). For mer komplekse prosjekter blir denne metoden svært upraktisk, derfor kan det egne seg å ta i bruk et RTOS.



Figur 3: Super-loop



Figur 4: RTOS

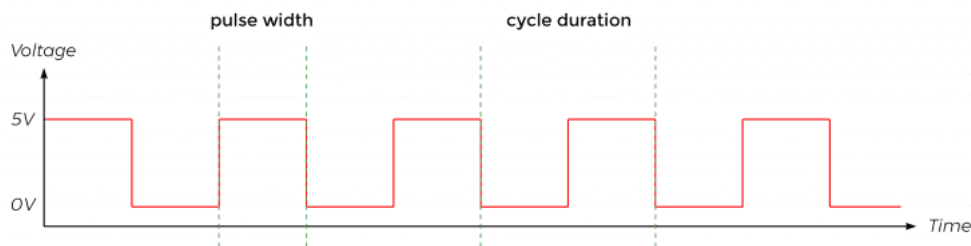
Et RTOS er en essensiell del av moderne innebygde systemer. Til forskjell fra generelle operativsystem som har til formål å gi brukeren en god opplevelse, er et RTOS utformet for å opprettholde og respektere strenge tidsfrister samtidig som det må ta minimalt med plass. RTOS er designet slik at det er deterministisk, som vil si at utfallet av en hendelse alltid er som forventet hver gang den samme hendelsen oppstår. Determinismen oppnås ved å opprettholde strenge tidsfrister og RTOS-et prioriterer oppgaver etter brukerdefinerte prioriteter, slik at disse blir utført i tide. Konsekvensen ved bruk av RTOS er at det vil alltid kreve ekstra prosessorkraft for å kjøre operativsystemet, men dette vil ofte veies opp for gjennom enklere determinisme, utvikling og debugging (freeRTOS udatert). En annen fordel ved bruk av RTOS er mulighet for flertrådsprogrammering som åpner for utføring av flere oppgaver simultant. Det betyr at tråder i programmet ikke behøver å vente på utføring av operasjoner som er ikke er relevante, og gjør det enklere å holde systemet deterministisk og raskt (Skøien 2019).

### 2.3.5 PWM-signal

Pulsbreddemodulasjon (PWM) er et digitalt signal som veksler mellom høy og lav tilstand over en fast periode. Et PWM-pulstog er illustrert i figur 5 med tidsintervaller på en konstant periode hvor den aktive tiden til signalet kan variere for å gi ulike utgangsspenning. Pulsbredden i forhold til perioden blir kalt arbeidssyklusen

(duty-cycle) og er oppgitt i prosent. En arbeidssyklus på 100 % tilsvarer alltid på, 0 % tilsvarer alltid av, og 50 % vil være halvparten av og på som i figur 5. Frekvensen til PWM-signalet vil variere avhengig av bruksområdet, men for styring av servomotor er det vanlig med en frekvens på 50 Hz. Dette gir en periodetid på:

$$T = \frac{1}{f} = \frac{1}{50 \text{ Hz}} = 20 \text{ ms} \quad (21)$$



Figur 5: Eksempel på et PWM-signal med en arbeidssyklus på 50 % (Lambert 2021)

### 2.3.6 Elektronikk

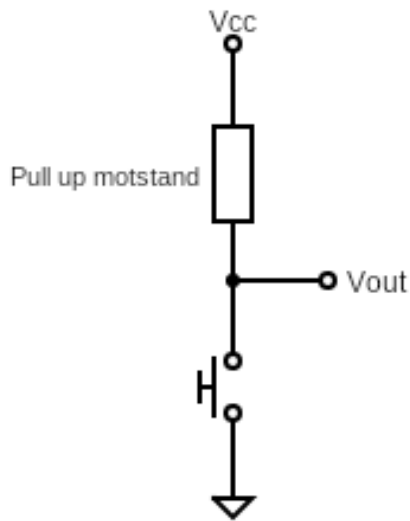
#### NPN-transistor

En NPN-transistor er en type bipolar transistor som består av to halvledere koblet i serie. Den har tre innganger som ofte kalles C for kollektor, B for base og E for emitter. Transistoren endrer karakteristikk ettersom hvordan strøm påføres disse inngangene. Om en transistor har null spenning på basen er den i cut-off modus, og det vil ikke gå strøm gjennom transistoren. Det tilsvarer at den oppføre seg som en åpen bryter hvor strømgjennomgangen er neglisjerbar. Om spenning påføres basen til transistoren går den over i metning. Da vil den oppføre seg som en lukket bryter, og strømmen vil bevege seg gjennom transistoren. Ved å koble til en LED før kollektoren kan denne egenskapen utnyttes ved at når strømmen sperres ved transistoren, flyter den gjennom LED-lyset. Når transistoren er åpen flyter heller strømmen gjennom transistoren. På denne måten kan en liten styrestrøm på basen til transistoren kontrollere en større strøm (ElectronicsTutorials udatert[b]).

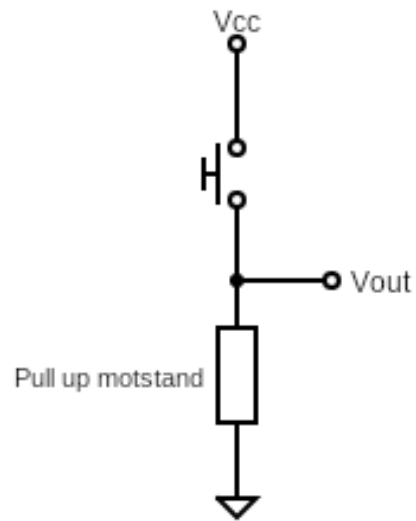
#### Pull-up og pull-down motstand

Når en mekanisk trykknapp blir koblet til et utviklingskort, er det vanlig å inkludere en pull-up- eller pull-down-krets. Disse kretsene forhindrer at trykknappen ligger i en flytende tilstand. Flytende tilstand betyr at det verken er definitivt høyt eller lavt nivå, slik at spontan endring i oppfattet logisk tilstand kan oppstå (ElectronicsTutorials udatert[a]). Pull-up-motstanden fungerer ved at signallinjen alltid er koblet til spenning gjennom en motstand (se figur 6). På denne måten er signalet alltid høyt når knappen ikke er trykket inn. Pull-down-motstanden fungerer ved at

signalinngangen kobles til jord gjennom en motstand (se figur 7), slik at når knappen ikke er trykket inn blir det logisk lavt nivå med referanse til jord.



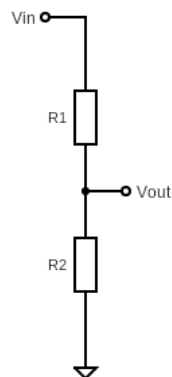
Figur 6: Pull-up-krets



Figur 7: Pull-down-krets

## Spenningsdeler

En spenningsdeler blir brukt for å trappe ned en spenning fra et høyere til et lavere nivå. Hvis en komponent er dimensjonert for 3.3 VDC og en mikrokontroller opererer på 5 VDC kan en spenningsdeler kobles inn for å trappe ned spenningen slik at den er kompatibel med komponenten (All About Circuits udatert). Spenningsdeleren fungerer ved at to motstander kobles i serie slik som i figur 8:



Figur 8: Enkel spenningsdeler

Spenningen som skal være nedtrappet hentes ut mellom motstandene og vil tilsvare



spenningen over  $R_2$ . Utgangsspenningen kan regnes ut med formelen:

$$V_{ut} = V_{inn} \frac{R_2}{R_1 + R_2} \quad (22)$$

hvor:

- $V_{ut}$  er utgangsspenning [V]
- $V_{inn}$  er inngangsspenning [V]
- $R_1$  er første motstand [ $\Omega$ ]
- $R_2$  er andre motstand [ $\Omega$ ]

Spenningsdelere er ikke egnet som strømforsyning, siden all strøm må gå gjennom motstand  $R_1$ . Likningen  $P = UI$  viser at om spenningen og strømmen er høy, vil effektenutviklingen bli høy. Effekten i motstanden blir avgitt som varme og kan i tillegg overskride grenseverdiene for motstanden (Jimblom udatert).

## 2.4 Algoritme og programmering

### 2.4.1 Datatyper

Kernighan og Ritchie (1991) definerer fem fundamentale datatyper i programmeringsspråket C: “character”, “integer”, “floating-point”, “double floating-point” og “valueless”. Senere i ISO (1999) ble datatypene utvidet med “bool”, “complex” og “imaginary”. Disse datatypene former grunnlaget for flere andre datatyper som man kan bruke ved programmering i C. Størrelsen og område til datatypene kan variere etter hvilken prosessortype eller kompilator som blir brukt, men normalt vil følgende gjelde:

- **char** - vil alltid ha størrelse på 8 bits og vil vanligvis inneholde verdier definert av ASCII tegnsettet.
- **int** - vil enten ha en størrelse på 16 bits eller 32 bits avhengig av hvilket programmeringsmiljø som brukes.
- **float** - har en størrelse på 32 bits og vil ha en presisjon på 6 desimaler.
- **double** - har en størrelse på 64 bits og vil ha en presisjon på 10 desimaler.
- **void** - vil enten deklare en funksjon som ikke returnerer en verdi eller skape en generell peker (2.4.5).

For alle datatyper i C utenom **void**, er det mulig å modifisere datatypene til et mer spesifikt bruksområde ved hjelp av tilleggsargumenter som “signed”, “unsigned”,

“long” og “short”. Ved å legge til **signed** for datatypen **int**, spesifiseres det at variabelen som blir definert kan holde både positive og negative verdier, ved å bruke det mest signifikante bitet til å indikere fortegn, og vil kunne holde dekadiske verdier opp til  $2^N - 1$ . Ved å bruke **unsigned** derimot, spesifiseres det at variabelen bare kan inneholde positive verdier, og vil derfor kunne holde dekadiske verdier opp til  $2^N$ .

Ved definering av variabler kan det kontrolleres hvordan de blir aksessert eller modifisert. Ved å deklare en variabel som **const** vil ikke variabelen kunne bli endret av programkoden etter kompilering. Modifikatoren **volatile** informerer kompilatoren om at variabelen kan bli endret uten at programkoden spesifikt endrer den.

For å spesifisere hvor en variabel skal bli lagret i minnet, kan det i C tas i bruk lagringsklasse-spesifikasjoner. Lagringsklassene kan ha formen: **extern**, **static**, **register** eller **auto**. Prinsippet bak å deklare en variabel som **extern** er at objektet er deklart med en ekstern kobling i en annen del av programkoden. En variabel kan bli deklart flere steder i programkoden, men bare definert én gang. Hvor forskjellen på deklarer og definere, er at ved deklarer så deklarerer en variabelnavn og datatype, mens ved definere av en variabel allokeres minne på mikrokontrolleren til objektet. En **extern**-variabel kan brukes på tvers av programkode som inneholder flere kildefiler.

Variabler som blir deklart som **static** vil ha egne egenskaper basert på om spesifikasjonen er på en lokal eller global variabel. Ved å deklare en lokal variabel som *static* vil minnet til objektet ikke bli slettet i løpet av kjøretiden til programmet. Og man unngår konflikten som kan oppstå ved bruk av globale variabler. Ved bruk av **static** på en global variabel spesifiseres det at objektet bare kan bli brukt i kildefilen hvor den er blitt deklart.

### 2.4.2 Egendefinerte datatyper

Dersom datatypene i C ikke tilfredsstillt behovet for datatyper, kan egendefinerte datatyper i form av *structure*- og **typedef**-nøkkelordene benyttes (Kernighan og Ritchie 1991). En *structure* er en samling av variabler under ett navn og gir en oversiktlig måte å holde informasjon som er relatert, samlet. Ved deklarer av en *structure* lages det et rammeverk på hvordan det i senere tid kan lages objekter av denne datatypen. Det er vanlig at elementene innenfor rammeverket er logisk relatert til hverandre. Ved bruk av nøkkelordet **typedef** lages det ikke en ny datatype, men det defineres et nytt navn til en eksisterende datatype. Dette gjør at programkoden blir mer oversiktlig og brukervennlig om programkoden skal brukes i et annet miljø, da av andre selskaper eller utvikling på andre MCU.

### 2.4.3 Array

Et array er en samling av variabler med lik datatype, som er referert til under samme navn. Enkeltelementer i et array kan aksesserer ved å benytte indekseringen elementene har i arrayet. I C er minneallokeringen til et array sammenhengende, som vil si

at den minst signifikante bit-adressen korresponderer til det første elementet i arrayet og den mest signifikante bit-adressen til det siste elementet i arrayet (Kernighan og Ritchie 1991).

#### 2.4.4 Operatører

I C finnes det en rekke operatører som kan tas i bruk. Operatørene er hovedsaklig delt opp i fire klasser: *arithmetic*, *relational*, *logical* og *bitwise*. I tillegg finnes det spesielle operatører som *assignment*-operatøren. Aritmetiske operatører som “+”, “-”, “\*” og “/” fungerer likt som forventet fra ordinær matematikk.

Logiske operatører og relasjonsoperatører referer til den logiske forbindelsen mellom variabler. Nedenfor i tabell 1 viser de mest brukte logiske- og relasjonsoperatørene og deres funksjonalitet:

Operator	Betydning
==	lik
>	større enn
<	mindre enn
!=	ikke lik
>=	større enn eller lik
<=	mindre enn eller lik
&&	logisk OG
!!	logisk ELLER
!	logisk IKKE

Tabell 1: Oversikt over logiske operatører og relasjonsoperatører

Uttrykk som tar i bruk disse operatørene returner 0 når uttrykket er falsk og 1 når uttrykket er sant.

**Ternary**-operatøren er en nyttig operator som kan erstatte “if-then-else”-formatet under visse omstendigheter. Operatøren har formen ? og tas i bruk slik:

```
variableName = condition ? exprIfTrue : exprIfFalse;
```

“?”-operatøren vil evaluere betingelsen, om den er sann vil variabelen få verdien til det første uttrykket og om det betingelsen er usann vil variabelen få verdien til det andre uttrykket.

Ved programmering i C kan det være nødvendig å ta i bruk minneadressen til et objekt for å modifisere inngangsparameterene til en funksjon, referering til array-elementer eller bruk andre dynamiske datastrukturer som binære trær og linkede lister (Kernighan og Ritchie 1991). Det kan tas i bruk peker-operatørene & og \*. Adresse-operatøren & vil returnere minneadressen til operanden, mens \*-operatøren vil returnere verdien lokalisert i minnet til operanden.

### 2.4.5 Funksjoner

En av de fundamentale byggesteinene i C er bruken av funksjoner. En generell funksjon består av en datatype som skal bli returnert, et funksjonsnavn, en parameterliste og logikken til funksjonen. Ved deklarerings av en funksjon bestemmes det hvilken datatype som skal bli returnert og det kan velges mellom alle datatyper bortsett fra typen array. Inngangsparameterne til funksjonen vil få verdiene til argumentene når funksjonen blir tilkalt. Funksjoner kan også deklarerer uten inngangsparametere.

Dersom en funksjon er deklarerert med inngangsparametere finnes det to metoder å overføre argumentene inn til funksjonen. Den første metoden kalt “*call by value*”, vil kopiere verdien til argumentet til parameterlisten. Endringer på den oppgitte parameteren i funksjonen vil ikke påvirke argumentet som blir gitt ved tilkalling av funksjonen. Den andre metoden kalt “*call by reference*”, kopierer adressen som argumentet har til parameterlisten. Endringer som forekommer på parameteren vil dermed også påvirke argumentet.

En funksjon som tar inn et argument som referer til en kjørbare kode kalles en “*callback*”-funksjon. Den kjørbare koden vil bli tilkalt av *callback*-funksjonen. For å implementere en slik funksjon kan en ta i bruk pekeroperatoren \*, som ble gjennomgått i 2.4.4.

I noen tilfeller kan det være ønskelig at en funksjon skal tilkalle seg selv. En slik funksjon vil være *rekursiv*. For hver gang den tilkaller seg selv vil et nytt sett med lokale variabler og parametere bli allokert og oppbevart i stacken, som er en abstrakt datastruktur for midlertidlig lagring av data. Funksjonen vil dermed bli kjørt på nytt med de nye variablene. Ved bruk av rekursjon er det viktig at en tar i bruk et betinget argument for å kunne returnere fra funksjonen.

### 2.4.6 Behandlere

Behandlere består av en kodeblokk assosiert med en spesifikk *interrupt*-tilstand. Ved en endring av tilstanden til et objekt enten i form av hardware eller software, vil programvaren kjøre en *interrupt*-rutine (2.3.3). Ved implementering av softwarebehandlere blir det ofte tatt i bruk *callback*-funksjoner, som ble gjennomgått i 2.4.5. Ved å ta i bruk behandlere får applikasjonen en asynkron oppførsel.

### 2.4.7 Prosesssynkronisering

Ved bruk av **semaforer** kan man signalisere mellom enkelttråder som kjører samtidig på mikrokontrolleren, for å synkronisere flere prosesser. En **semafor** er en variabel med type **unsigned interger**, som betyr at den bare kan inneholde positive verdier. Det finnes to typer semaforer, *binære semaforer/mutex-låser* og *tellende semaforer*. Binære semaforer kan bare ha to verdier: 0 og 1. Den andre typen, *tellende semaforer*, har ikke noen øvre grense på området den kan operere i. Ved å bruke

semaforer mellom tråder kan det oppstå et problem kjent som *deadlock*. Anstensrud (2021) definerer en *deadlock* som “et sett med tråder der alle er i en slik tilstand at ingen av de kan fortsette fordi alle venter på hverandre”. Det finnes en rekke ulike teknikker for å unngå, oppdage og håndtere *deadlocks*.

## 2.4.8 Databehandling

For å finne informasjon i et usortert array må man ta i bruk av et sekvensielt søk. Den sekvensielle søkemetoden starter ved det første elementet og stopper når den finner en samsvarende verdi, eller til den når enden på arrayet. Ved bruk av denne søkemetoden vil gjennomsnittlig iterering være  $\frac{n}{2}$ , det vil si 1 ved beste tilfelle og  $n$  ved verste tilfelle. Dette gjelder ved  $n$  elementer i arrayet (Cormen mfl. 2022). Det vil være mulig å bruke mer tidsbesparende og komplekse søkemetoder på array som er sorterte.

## 2.5 Bluetooth

### 2.5.1 Bluetooth

Bluetooth benytter seg av 79 kanaler i frekvensspekteret 2.402 og 2.480 GHz med 1 Mhz avstand og benytter seg av frekvenshopping for å bytte mellom disse. Den har en hoppehastighet på 1600 hopp per sekund, som tilsvarer en holdetid på  $1/1600h = 625 \mu s$  (Frenzel 2015, s. 825).

For å sende et pålitelig signal i trådløs kommunikasjon er det ofte nødvendig å modulere signalet. Bluetooth anvender gaussisk frekvensskiftning (GFSK), som tar i bruk et Gaussisk filter (Jiang og Scott 2020) før det blir modulert. Frekvensskiftning er en modulasjonsmetode som blir brukt for overføring av binær data i trådløs kommunikasjon. Signalene blir modulert ved at frekvensen til bærebølgen blir endret for de binære verdiene (0,1). Variasjonen for frekvenskomponenten er avhengig av forsterkningen til signalet, og vil for Bluetooth være på  $\pm 160$  kHz (Frenzel 2015, s. 707).

Kommunikasjonen mellom enhetene er tilkoblingsorientert, som impliserer at det må etableres en tilkobling før det kan oppstå dataoverføring mellom enhetene. Dette refereres til som en ende-til-ende-kommunikasjon (P2P).

Det blir brukt Bluetooth-klasser for å spesifisere signalstyrken som anvendes på applikasjonen. Denne avhenger av distansen som signalet skal propegeres, og vil være definert ved faste verdier uttrykt i dBm og mW (Laird Connectivity 2022).

<b>Bluetooth-klasse</b>	<b>Maksimal signalstyrke</b>	<b>Operasjonsområde</b>
Klasse 1	10 mW (20 dBm)	100 meter
Klasse 2	2.5 mW (4 dBm)	10 meter
Klasse 3	1 mW (0 dBm)	1 meter

Tabell 2: Bluetooth-klasser for signalstyrke

### 2.5.2 Bluetooth Low Energy

BLE er en variant som er derivert fra den klassiske Bluetooth-protokollen. Denne bruker samme frekvensbånd som Bluetooth, men er inndelt i 40 kanaler med 2 Mhz avstand (Bluetooth 2022), derav 3 kanaler er dedikert til annonsering.

Denne nettverksprotokollen bruker tilkoblingsorientert kommunikasjon, men har også støtte for tilkoblingsfri kommunikasjon gjennom broadcasting.

### 2.5.3 Generic Attribute Profile (GATT)

GATT er en generisk dataprotokoll som blir brukt i tjener-laget i BLE-protokollstabelen. Denne vil definere hvordan kommunikasjonen mellom enhetene skal foregå etter de har etablert en tilkobling (Cypress Semiconductor 2015).

I GATT-protokollen vil det være ulike roller enhetene kan besettes i systemet, som kalles server og klient. Klienten vil etterspørre data fra serveren, og deretter lese av attributtene fra serveren. Funksjonaliteten til serveren vil derfor være å lagre attributene, og gjøre den tilgjengelig ved forespørsel.

### 2.5.4 Generic Access Profile (GAP)

GAP definerer topologien til protokollstabelen. I likhet med GATT, er det moduser enhetene kan bruke for kommunikasjonen; disse er broadcasting og connecting.

For broadcasting, vil det være en kringkaster som sender ut data, og en observatør som leser av data. Denne topologien krever ingen tilkobling mellom enhetene, og vil være definert ved en-til-mange-kommunikasjon. Overføringen av data skjer kun i én retning, og vil være definert som simpleks kommunikasjon (Shanhaban 2022).

Ved bruk av connection-modusen vil enhetene sende data hvis og bare hvis de har etablert en tilkobling. Rollene vil være fordelt mellom periferier og sentraler. Periferiene vil annonsere tilværelsen sin til andre sentraler. Dersom den etablerer en tilkobling med en sentral, vil periferien avbryte annonseringen og kommunikasjonen vil kun foregå mellom de respektive enhetene. Sentralen vil lytte etter annonseringer, og deretter kunne koble seg til periferiene. Sentralen har en en-til-mange relasjon

mellom periferiene, og vil kunne opprettholde flere tilkoblinger samtidig (Townsend 2014).

Fordelen med connection vil derfor være at den kan opprettholde en forbindelse mellom periferien og sentralen og vil derfor kunne sende data over lengre tidsintervaller, som vil føre til et redusert strømforbruk. Broadcasting vil kunne starte kommunikasjonen mellom enhetene simultant, og derfor ikke være avhengig av en forbindelse for å utveksle data.

Broadcast-intervallet beskriver tidsintervallet mellom hver annonsering. For å unngå datakollisjoner mellom flere enheter blir det generert en tilfeldig generert forsinkelse (Actorsfit 2022).

Enheter som skanner etter annonseringer fra kringkasteren er også betinget av tidsintervaller. Disse parameterene kalles skanningsperiode og skanningmellomrom, og definerer henholdsvis tidsintervallene for aktiv skanning og ventetid mellom hver skanning.

## 2.6 Statistikk

Et utvalgs gjennomsnitt er den beste gjetning på en populasjons forventningsverdi, og estimatoren for  $\mu$  er  $\hat{\mu}$ , (Løvås 2018, s. 237):

$$\bar{X} = \frac{1}{n}(X_1 + X_2 + \dots + X_n) = \frac{1}{n} \sum_{i=1}^n X_i \quad (23)$$

Et utvalgs varians er den beste gjetning på en populasjons varians, og den forventningsrette estimatoren for variansen  $\sigma^2$  er (Løvås 2018, s. 239):

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^2 \quad (24)$$

Når et standardavvik  $\sigma$  er ukjent, kan  $\sigma$  estimeres ved å sette inn utvalgsstandardavviket  $S$  som en erstatning for den ukjente  $\sigma$ .  $S$  anses da som en punktestimator for  $\sigma$ , der standardfeilen er gitt som (Løvås 2018, s. 248):

$$SE(\bar{X}) = \frac{S}{\sqrt{n}} \quad (25)$$

Dersom standardavviket  $\sigma$  er ukjent, kan et  $100(1-\alpha)\%$  konfidensintervall for  $\mu$  med forutsetningen om at målingene  $x_i$  av den stokastiske variabelen  $X$  er normalfordelte, gis som (Løvås 2018, s. 249):

$$\left[ \bar{X} - t_{\alpha/2} \cdot \frac{S}{\sqrt{n}}, \bar{X} + t_{\alpha/2} \cdot \frac{S}{\sqrt{n}} \right] \quad (26)$$

## 3 Metode

### 3.1 Utstyrliste

Komponentene som inngår i systemet er listet opp i tabell 3:

Utstyr	Delnummer	Antall enheter
Antenne-patch	ANT-2.4-CPA	2
Breadboard	920-0031-01	1
Dupontkabler	1528-1964-ND	1
Enkoder	AMT103-2048-N4000-S	2
Enkoderkonnektor	3-640440-5	2
Evalueringskort	EK42442-01	1
Hybrid-kobler	3A0200	1
Koaksialkabel 152,4 mm	BU-4150029006	2
Koaksialkabel 457,2 mm	415-0029-018	2
Laserdiode	1528-1391-ND	1
Mikroservo	SER0049	1
Multiprotokoll SoC	nRF52-DK	2
Pigtail	MXHS83QE3000	1
Servomotor	SER0038	2
SMA-konnektor	60312202114514	2
Strømforsyning	GST40A07-P1J	1
Taktilbryter	CKN12302-ND	2

Tabell 3: En liste over komponentene som inngår i systemet

### 3.2 Programvare

Program	Utvikler	Funksjon
Altium Designer	Altium	Kretsdesign
Fusion 360	Autodesk	3D-modellering
GanttProject	BarD Software s.r.o. mfl.	Prosjektorganisering
GitHub	GitHub	Versjonskontroll
KiCad EDA	KiCad	Design av PCB-utlegg
nRF Connect for Desktop	Nordic Semiconductor	Programvare for utvikling med nRF52DK
PuTTY	Simon Tatham	Seriellterminal
Visual Studio Code	Microsoft	IDE

Tabell 4: Programvaren som ble benyttet under prosjektet



### 3.3 Robotikk

#### 3.3.1 Kraftberegninger

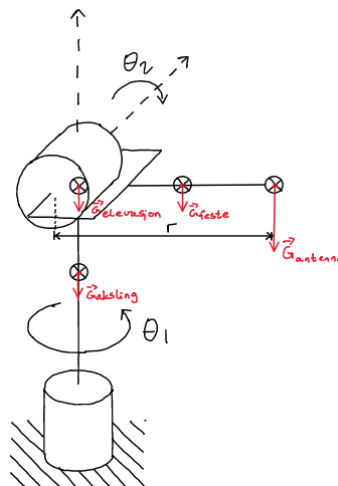
I en robotapplikasjon der det skal dimensjoneres og velges ut egne deler i en egenprodusert robot, må det gjøres beregninger basert på hva som er målet for roboten. I dette prosjektet skal det lages en robotløsning som kan orientere en retningsstyrt antenne, så dimensjoner og vekt for antennen er avgjørende egenskaper for hvilke motorer som trengs for å kunne pålitelig orientere antennen som ønsket. For å kunne finne en motortype for rotasjon i asimut- og elevasjonsvinkelen ble det gjort momentsberegninger for motorene, med en antakelse om at antennen som skulle anvendes i prosjektet maksimalt kom til å ha en masse på 0.60 kg med et massesenter på 20 cm fra motorens roterende akse.

Servo- og steppermotorer har oppgitte verdier for dreiemomenter i databladene fra produsentene, så for å finne en motor med tilstrekkelig dreiemoment ble det gjort beregninger for dreiemomentene som kunne oppstå ved orientering av antennen med massen 0.60 kg. For elevasjonsservoen ble det maksimale statiske dreiemomentet beregnet ved formelen gitt i 2.1.2. Det maksimale statiske dreiemomentet oppstår når antennen er orientert horisontalt som i figur 9. Det ble antatt at festeanordningen mellom antennen og elevasjonsservoen kom til å bli en stiv tynn stang med lengde  $l = 0.20$  m, masse  $m = 0.10$  kg og massesenter midt i festet. Da kan dreiemomentet på rotasjonsaksen til elevasjonsservoen beregnes slik:

$$\tau_{antennefeste} = 0.1 \text{ m} \cdot 0.10 \text{ kg} \cdot 9.81 \text{ m/s}^2 \cdot \sin 90^\circ = 0.0981 \text{ Nm}$$

$$\tau_{antenne} = 0.20 \text{ m} \cdot 0.6 \text{ kg} \cdot 9.81 \text{ m/s}^2 \cdot \sin 90^\circ = 1.1772 \text{ Nm}$$

$$\tau_{elevasjon} = \tau_{antennefeste} + \tau_{antenne} \approx 1.28 \text{ Nm}$$

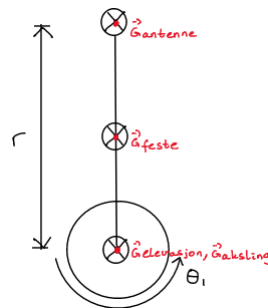


Figur 9: Illustrasjon av robotkonfigurasjonen ved maksimalt statisk dreiemoment for elevasjonsservoen

Siden det for elevasjonsservoen oppstår et maksimalt statisk dreiemoment på 1.28 Nm for en antenne på 0.60 kg plassert 20 cm fra rotasjonsaksen, må elevasjonsservoen ha et større dreiemoment enn dette. Ved å undersøke tilgjengeligheten for motorer på distributører som “DigiKey” og “Mouser” viste det seg at steppermotorer som ikke er giret har ikke stort nok dreiemoment. Servomotorer derimot har relativt stort dreiemoment, der for eksempel DFRobot sin SER0038-servomotor har oppgitt et maksimalt dreiemoment på 17 kg\*cm på 6.8 VDC driftsspenning, som kan omformes til 1.67 Nm ( $1 \text{ kg} \cdot \text{cm} \approx 0.098 \text{ Nm}$ ). DFRobot SER0038 ser derfor ut til å egne seg godt i prosjektet basert på beregningen av dreiemoment for elevasjonsplanet.

## Asimutservoen

For å beregne dreiemomentet som asimutservoen må ha for å rotere robotarmen rundt i horisontalplanet, kan likningen for rotasjonsdynamikk:  $\tau = I \cdot \alpha$ , benyttes. Robotarmen illustrert ovenfra kan ses i figur 10:



Figur 10: Illustrasjon av robotkonfigurasjonen sett ovenfra med elevasjonsservo i horisontal retning

For å beregne dreiemomentet  $\tau$  må treghetsmomentet  $I$  og vinkelakselerasjonen  $\alpha$  først beregnes. Det totale treghetsmomentet som asimutservoen opplever er en sammensetning av treghetsmomentene til akslingen mellom asimut- og elevasjonsservoen, elevasjonsservoen, antennen og den stive stanga (festet) mellom elevasjonsservoen og antennen. Treghetsmomentet  $I_{\text{elevasjonsservo}}$  kan beregnes ved å tilnærme elevasjonsservomotoren til et rektangulært volum. Elevasjonsservo har lengde  $l = 40$  mm, bredde  $b = 20$  mm og masse  $m = 25$  g og treghetsmomentet kan beregnes til å være:

$$I_{\text{elevasjonsservo}} = \frac{1}{12} \cdot 0.025 \text{ kg} \cdot (0.040^2 + 0.020^2) \text{ m} \approx 4.2 \cdot 10^{-6} \text{ kgm}^2$$

Det må være en rigid kobling mellom akslingen på asimut- og elevasjonsservoen for å forhindre nøyaktighetsdegradering for systemet, som nevnt i 2.1.1. For å sikre så høy stivhet som mulig, ble akslingen antatt utformet som en solid sylinder i aluminiumlegeringen EN AW-5052 med en masse  $m = 10$  g og radius  $r = 4.5$  mm. Treghetsmomentet for akslingen kan beregnes til å være:

$$I_{aksling} = \frac{1}{2} \cdot 0.010 \text{ kg} \cdot (0.0045 \text{ m})^2 = 2.25 \cdot 10^{-5} \text{ kgm}^2$$

Ved å tilnærme festeordningen for antennen til elevasjonsservoen som en tynn stang med masse  $m = 100 \text{ g}$  og lengde  $l = 200 \text{ mm}$ , kan treghetsmomentet beregnes til å være:

$$I_{antennefeste} = \frac{1}{3} \cdot 0.1 \text{ kg} \cdot (0.20 \text{ m})^2 \approx 1.3 \cdot 10^{-3} \text{ kgm}^2$$

Antennen tilnærmes en punktpartikkel med masse  $m = 0.6 \text{ kg}$  en avstand  $r = 0.20 \text{ m}$  fra rotasjonsaksen. Da kan treghetsmomentet for antennen beregnes til:

$$I_{antenne} = 0.6 \text{ kg} \cdot (0.20 \text{ m})^2 = 24.0 \cdot 10^{-3} \text{ kgm}^2$$

Det totale treghetsmomentet beregnes som:

$$I_{total} = I_{elevasjonsservo} + I_{aksling} + I_{antennefeste} + I_{antenne} \approx 25.3 \cdot 10^{-3} \text{ kgm}^2$$

Det er ingen strenge krav til hvor stor vinkelakselerasjonen  $\alpha$  må være i prosjektet, da nøyaktighet av søkeprosessen veies høyere enn søkehurtighet. Basert på databladet for DFRobot SER0038-servomotoren er vinkelhastigheten uten last oppgitt til å være  $\omega^{-1} = 0.16 \text{ s}/60^\circ$ . Dette er den inverse av vinkelhastigheten  $\omega$ , der  $\omega_{grader}$  kan beregnes til:

$$\omega_{grader} = \frac{60^\circ}{0.16 \text{ s}} = 375^\circ/\text{s}$$

Ved omforming til radianer beregnes  $\omega_{rad}$  til:

$$\omega_{rad} = 375^\circ/\text{s} \cdot \frac{\pi \text{ rad}}{180^\circ} = \frac{25}{12} \pi \text{ rad/s}$$

Med utgangspunkt i SER0038-servomotoren og en antakelse om at denne typen servomotor kan akselerere hele belastningen opp til terminalvinkelhastigheten  $\omega$  etter  $20^\circ$  rotasjon, kan vinkelakselerasjonen  $\alpha$  beregnes fra  $\omega^2 - \omega_0^2 = 2\alpha\theta$ :

$$\alpha = \frac{\omega^2 - \omega_0^2}{2\theta} = \frac{(\frac{25}{12}\pi \text{ rad/s})^2 - (0 \text{ rad/s})^2}{2 \cdot 20^\circ \cdot \frac{\pi \text{ rad}}{180^\circ}} = \frac{625}{16} \pi \frac{\text{rad}}{\text{s}^2} \approx 61.36 \frac{\text{rad}}{\text{s}^2}$$

Det maksimale dreiemomentet som asimutservoen må ha for å kunne rotere den ferdig monterte robotarmen ved maksimal hastighet, kan beregnes til:

$$\tau_{asimut} = I_{total} \cdot \alpha = 25.3 \cdot 10^{-3} \text{ kgm}^2 \cdot 61.36 \text{ rad/s}^2 \approx 1.55 \text{ Nm}$$

Det rettes oppmerksomhet til at det er knyttet en del usikkerhet til  $\tau_{asimut}$ , siden det er antakelser både i det totale treghetsmomentet og hvor mange grader en servomotor bruker på å nå terminalhastigheten. Men, siden  $\tau_{asimut}$  er basert på maksimale verdier for masse og armlengde antas DFRobot SER0038-servomotoren som en tilstrekkelig servomotor å benytte i prosjektet basert på kraftberegningene.

På dimensjonering- og beregningsstadiet i prosjektperioden var det antatt at antenntypen som skulle benyttes var en retningsstyrt antenne med en masse på maksimalt 0.60 kg og massesenter 0.20 m fra elevasjonservoens roterende akse. På bestillingstidspunktet for motorene var prosjektet fortsatt basert på en slik type antenne, men etter at servomotorene var bestilt og levert ble antenntype og søkemetode endret. Antennen som ble brukt i prosjektet er et antennearray bestående av to patch-antennert monterte på et PCB-kort, der PCB-utformingen er utført i samsvar med kravene for montering for de to patch-antennene. En slik antennearray-løsning har vesentlig mindre masse og kan monteres nærmere rotasjonsleddet enn de først antatte 20 cm.

## 3.4 Design av robotarm

### 3.4.1 Digitale verktøy

Siden det i prosjektet skulle utvikles en egen løsning for orientering av den retningsstyrte antennen som en RRR-robot med åpen kinematisk kjede, oppsto det et behov for å kunne produsere spesialtilpassede deler til bruk på robotarmen. Dette behovet ble dekket ved å benytte seg av programvaren Autodesk Fusion 360. I Fusion 360 ble det designet 3D-modeller av de nødvendige delene til robotarmen og deretter eksportert til .3mf-format. Eksporteringen av 3D-modellene ble gjort til .3mf-formatet for å beholde mest mulig informasjon om modellen, som beskrevet i 2.1.6. 3D-printing egnet seg godt i prosjektet til å produsere prototyper av de ulike delene på grunn av hurtigheten og kostnadseffektiviteten en ordinær FDM-3D-printer kan tilby.

I Autodesk Fusion 360 er det mulighet for å legge til egendefinerte parametre slik at endringer enkelt kan gjøres på dimensjonene til en modell ved et senere tidspunkt. Et slikt behov kan oppstå etter produksjon av modellen eller ved endringer på flere av enkeltmodellene som utgjør sluttmodellen. Gjennom designperioden av robotarmen ble det hele tiden brukt egendefinerte parametre nettopp fordi revisjoner av delene på robotarmen kunne, og trolig ville, oppstå. Et utsnitt av parameterlisten for egendefinerte parametre i Fusion 360 anvendt på modellen av DFRobots SER0038-servomotorene, er gitt i figur 11:

Parameter	Name	Unit	Expression	Value	Comments
Favorites					
User Parameters +					
☆ User Parameter	servo_width	mm	20 mm	20.00	
☆ User Parameter	servo_height	mm	40.5 mm	40.50	
☆ User Parameter	servo_mounting_hole_height_from_bottom	mm	27.8 mm	27.80	
☆ User Parameter	servo_mounting_hole_center_width	mm	10 mm	10.00	
☆ User Parameter	servo_height_total	mm	47.5 mm	47.50	
☆ User Parameter	servo_mounting_hole_length	mm	54.2 mm	54.20	
☆ User Parameter	servo_length	mm	40 mm	40.00	
☆ User Parameter	servo_mounting_hole_height	mm	4 mm	4.00	
☆ User Parameter	servo_body_filet	mm	2 mm	2.00	
☆ User Parameter	servo_round_horn_diameter	mm	24.4 mm	24.40	

Figur 11: Utsnitt av egendefinerte parametre i Autodesk Fusion 360

### 3.4.2 Designprosessen

#### Kontrollboks og robotarmbase

For bedre oversikt og lettere oppbevaring av nRF52-utviklingskortet, evalueringskortet, hybrid-kobleren og koblingsbrettet på en forsvarlig måte, ble det designet en boks med festeanordninger for enkeltkomponentene. Plasseringen av festeanordningene på hybrid-kobleren er basert på databladet, mens festet for nRF52-utviklingskortet og koblingsbrettet er målt med et skyvelære siden gode datablad ikke fantes for monteringshullene på brettene. For å gi inntrykk om at sluttproduktet er en helhetlig integrert løsning, ble kontrollboksen designet for å kunne benytte robotarmbasen som topplokk. I tillegg blir systemet mer service-vennlig dersom komponentene er systematisk plassert i en boks. Samhandlingen mellom kontrollboksen og robotarmbasen kan ses i figur 20.

Fra toppen av koblingsbrettet og opp til undersiden av servobasen ble det lagt inn en avstand på 35 mm for å kunne koble på koblingsbrettet med dupontkabler og komponenter som resistorer og transistor. Det ble laget gjennomføringshull for micro-USB-kontakten til utviklingskortet og for strømforsyningskontakten. Det ble også laget to gjennomføringshull for SMA-kontaktene på hybrid-kobleren og ett hull for servo-, laserpeker- og enkoderledningene.

For å sikre en rigid og solid base å montere robotarmen på ble det benyttet POM-C-materiale fra ITK med innfelte gjengede monteringshull som kan ses i figur 22. Monteringshullstørrelsen og avstandene ble hentet fra DFRobot sitt datablad for SER0038-servomotoren, som ga informasjon om hvor disse hullene må plasseres for å passe servomotoren. Fire av monteringshullene samsvarer med hullene til DFRobots SER0038-servomotorene slik at det kunne kappes til fire lengder av M4 gjengestang og sikres med en M4 mutter på toppen av asimutservo. På denne måten blir asimutservo festet godt til baseplaten og skaper et stivt utgangspunkt for videre montering av de resterende delene av robotarmen. De fire M3-monteringshullene på

baseplaten i figur 25 er for å feste festeanordningen for asimutenkoderen til baseplaten.

En 3D-modell ble også laget for SER0038-servomotoren som benyttes for rotasjon i asimut- og elevasjonsplanet, for å ha en referanse og en illustrasjon for hvordan denne typen servomotor ville se ut i den helhetlige robotarmmodellen i figur 20. Modellen av SER0038-servomotoren er gitt i figur 23.

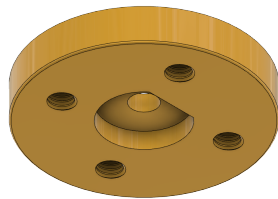
### **Lenke mellom asimut- og elevasjonsservoen**

Utgangspunktet for utformingen av akslingen mellom asimutservoen og festeanordningen for elevasjonsservoen er behovet for 30 mm klaring for montering av AMT103-enkoderen på akslingen. AMT103-enkoderen bygger 9.0 mm i høyden og er tenkt montert i underkant av festeanordningen i figur 25. Utformingen av endene på akslingen samsvarer med hornet som er festet på akslingen til asimutservoen, og er hentet fra DFRobots sitt eget datablad.

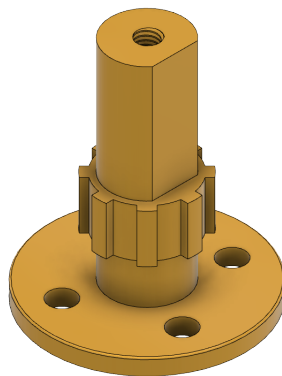
For å kunne rotere asimutenkoderhjulet likt med bevegelsen for asimutservoen, ble utsparingen i enkoderhjulet målt og tegnet inn på 3D-modellen for akslingen. Det er viktig at toleransene for koblingen mellom enkoderhjulet og metallakslingen er så små som mulig, siden unøyaktigheter og slark her vil introdusere unøyaktige rotasjoner av enkoderhjulet. Unøyaktigheten i rotasjon av enkoderhjulet vil påvirke nøyaktigheten i enkoderavlesningen og avslutningsvis nøyaktigheten til robotarmen.

Siden akslingen måtte gå fra hornet på asimutservoen, gjennom asimutenkoderen og opp til festeanordningen for elevasjonsservoen dukket det opp et behov for å gjøre den ene endeplaten på akslingen avtakbar. Løsningen på denne utfordringen ble å lage et spor langs akslingen som samsvarer med en utsparing på undersiden av toppen på metallakslingen med så små toleranser som mulig. Utsparingen i den avtakbare toppen på akslingen kan ses i figur 12 og aksling med spor kan ses i figur 13. Utsparingen og koblingen mellom aksling og akslingtopp bør ha så små toleranser som mulig, for å minske bidraget med unøyaktigheter i robotarmen.

Behovet for rigiditet er forklart i 2.1.1 og nettopp på grunn av rigiditeten ble lenken mellom asimutservoen og festeanordningen for elevasjonsservoen produsert på en CNC ved mekanisk verksted på Institutt for teknisk kybernetikk (ITK) i aluminiumlegeringen EN AW-5052.



Figur 12: 3D-modell av toppen av akslingen mellom asimut- og elevasjonsservoen



Figur 13: 3D-modell av akslingen mellom asimut- og elevasjonsservoen

### **Festeanordninger for elevasjonsservoen**

Festeanordningen for elevasjonsservoen i figur 26 ble basert på databladet for SER0038-servomotoren. Monteringsmønsteret på undersiden av braketten samsvarer direkte med monteringsmønsteret på akslingen mellom asimutservoen og braketten for elevasjonsservoen. Festet ble forespurt produsert i aluminium ved ITK for høyest mulig stivhet.

Braketten skiller seg fra databladet for SER0038-servomotoren i at det er tegnet inn et ekstra hull med samme diameter som tappen av modellen i figur 27. SER0038-servomotoren ble produsert med en flat bunn på motsatt side av motorakslingen. Det ble designet en ny bunn for SER0038-servoen med en tapp for å gi et feste for et hjullager med 8.00 mm innerdiameter og 16.20 mm ytterdiameter, som er festet til den roterende braketten på elevasjonsservoen. Tappen har også en forlengelse med et kryssmønster som samsvarer med koplingen til elevasjonsservohjulet, og kan ses i figur 27. På denne måten får man ved hjelp av den nye bunnen på elevasjonsservoen

mulighet til å feste den roterende braketten både på servoakslingen og en stasjonær aksling, altså i begge ender. Den roterende braketten kan ses i figur 28, med monteringshull for festet til SER0049-servomotoren. Denne braketten ble også produsert i aluminium ved ITK.

### **Festeanordning for SER0049-servomotoren og antennearray**

Servomotoren som ble benyttet i ytterste rotasjonsledd for å rotere antennearrayet, er mindre i både størrelse og styrke enn SER0038-servomotoren. SER0049-motoren blir brukt for å rotere antennearrayet mellom to distinkte vinkler:  $0^\circ$  og  $90^\circ$ . Festeanordningen for servomotoren kan ses i figur 29 og har utsparinger for ledningsføring og monteringshull på den roterende braketten for elevasjonsservoen.

For å feste antennearrayet til robotarmen ble festeanordningen i figur 30 designet og 3D-printet. Monteringshullene på baksiden av festeanordningen samsvarer med hullene i nylonfestet som følger med SER0049-servomotoren og passer på akslingen til servomotoren. Festeanordningen til antennen har også et feste på toppen for laserpekeren som angir retningen antennearrayet peker i. Det er ikke mulig å montere laserpekeren i nullpunktet for antennearrayet, så en løsning med å midtstille laserpekerfestet mellom antennepatchene i en vertikalavstand på 4.0 cm over nullpunktet ble beste mulighet.

#### **3.4.3 Produksjon av robotarmdeler**

Før produksjon av en 3D-modell kan settes i gang på en 3D-printer, må .3mf-formatet fra CAD-programmet omgjøres til .gcode-format som Ultimaker-ene på MakeNTNU krever. Omformingen av formatet ble gjort i Ultimaker sin egen programvare kalt “Ultimaker Cura”. Der ble det i tillegg huket av for å printe delene med en laghøyde på 0.10 mm i PLA-materiale på grunn av økt mekanisk styrke og fraværet om behov for temperaturbestandighet og den lange varigheten til PETG-materialet. For produksjonen av 3D-modellene til robotarmen ble det reservert printetid på “Ultimaker 2+”-printerene til MakeNTNU. Noen av delene til robotarmen krevde høy presisjon, derfor ble noen av 3D-modellene printet, prøvemontert, revidert og printet på nytt før resultatene sto til kravene. Arbeidsflyten for design og produksjon av robotarmen har gått i en syklus som kort kan beskrives som å lese eksisterende datablad for robotarmkomponenter, designe en 3D-modell basert på databladene, 3D-printe modellene, prøvemontere de, revidere 3D-modellen og printe på nytt. I noen få tilfeller ble noen 3D-modeller revidert opp til fire ganger. Ved svært lang produksjonstid eller størrelse har 3D-modellen i stedet blitt sendt til de frivillige på MakeNTNU for å få printe på en av “Raise3D Pro2 Plus”-printerne. Disse printerne har en ikke tilgang til uten et ekstra kurs arrangert av MakeNTNU, noe som ingen på prosjektgruppen har (MakeNTNU 2022).



## 3.5 Antennearray

### 3.5.1 Kravspesifikasjoner

Ved research og analyse av de ulike peile-metodene ble det konkludert med at bruk av både nullpunktorientering og sterkest RSSI, også kalt fase/motfase ville egne seg best for systemet. Dette kan gjøres ved å bruke et antennearray med to antenne-elementer, som beskrevet i 2.2.8. Signalene faseforskyves slik at man kan bruke sumforskjell-metoden for radiopeiling, som nevnt i 2.2.9

Grunnet kompleksiteten til antenntypen var det nødvendig å sette kravspesifikasjoner til antennen, slik at antennen er kompitabel med nettverksprotokollen og robotarmen som ble designet. De tekniske spesifikasjonene i tabell 5 omhandler egendefinerte kravspesifikasjoner for antennen med hensyn på de elektrotekniske og de elektromagnetiske egenskapene som trengs i prosjektet. Frekvensbåndet beskriver frekvensområdet antennen har best ytelse. BLE opererer i frekvensområde 2.402 Ghz og 2.480 GHz som gitt i 2.5.2. Med rom for sidebånd og støy utenfor frekvensområdene til BLE, vil det være hensiktsmessig at antennen dimensjoneres for et frekvensbånd fra 2.4 GHz til 2.5 GHz.

Impedans er en viktig parameter i radiokommunikasjon, siden har en kritisk innvirkning på tap og vinninger i systemet. For det gitte antennesystemet ble det valgt å bruke 50  $\Omega$  impedans i samsvar med teorien i 2.2.1.

Polariseringen avgjør orienteringen til antennen for avlesning av de elektromagnetiske bølgene. Antennen er et 1x2 array med nullpunktet målt i motfase, liggende i det asimutale planet dersom antennen er orientert normalt på senderen. For å beholde faseforskjellen ved søk i elevasjonsplanet vil antennen roteres 90°, som kan føre til polariseringstap dersom polariseringen mellom antennene ikke blir tatt til betraktning. Sirkulærpolarisering er derfor nødvendig for at antennen skal støtte signaler både i asimut- og elevasjonsplanet.

Det var kritisk at antennen er direksjonell, siden nullpunktene målt i motfase ( $\Delta$ ) verifiseres ved måling i fase ( $\Sigma$ ). Det var derfor hensiktsmessig å ha en antenne som har en HPBW på  $< 90^\circ$ . En smal HPBW vil validering av nullpunkter med en 90°nøyaktighet.

---

Parameter	Kravnummer	Verdi
Frekvensbånd	ANT0101	2.4 GHz - 2.5 Ghz
Impedans	ANT0102	50 $\Omega$
Polarisering	ANT0103	Sirkulær
HPBW	ANT0104	$< 90^\circ$
Returtap	ANT0105	$< 15$ dB

---

Tabell 5: Kravspesifikasjoner for antennearrayet

Robotarmkomponentene har dimensjonale begrensninger som antennen betinges av.

Derfor er det kritisk at vekt og dimensjoner blir definert for antennearrayet, slik at den ikke overskrider kapasiteten til robotarmen. Dimensjonene for antennen er bestemt av PCB-utlegget der avstanden mellom antennene ble beregnet til å være  $\frac{\lambda}{2}$  for å motvirke aliasing som beskrevet i 2.2.8. Dimensjonene til patch-antennene er oppgitt i databladet (Linx Technology 2019) til å være 20.0 mm x 20.0 mm x 4.0 mm. For å minimere diskontinuitet i impedansen og elektromagnetisk støy, ble det brukt SMA-konnektorer som kontaktortype for antennearrayet (Paleček mfl. 2012). Utlegget ble derfor designet med hensyn på avstand mellom matepunktene og jordplanet til antennene, samt klaring til SMA-kontaktene.

Parameter	Kravnummer	Verdi
Dimensjoner	ANT0201	100 mm x 50.0 mm x 10.0 mm
Patch-størrelse	ANT0202	20.0 mm x 20.0 mm x 4.0 mm
Avstand	ANT0203	62.5 mm
Kontakttype	ANT0204	SMA

Tabell 6: Kravspesifikasjoner for dimensjonering av antennearrayet

Antennen er primært tiltenkt innendørs bruk og det vil dermed være hensiktsmessig å gi antennespesifikasjoner for drift av systemet for å tilpasse innretningen til omgivelsene den skal brukes i.

Parameter	Kravnummer	Verdi
Montering	ANT0301	Festeanordning
Temperatur	ANT0302	20 °C
Miljø	ANT0303	Innendørs bruk
Operasjonsområde	ANT0304	5.0 meter

Tabell 7: Kravspesifikasjoner for drift av antennearrayet

### 3.5.2 Valg av antenne

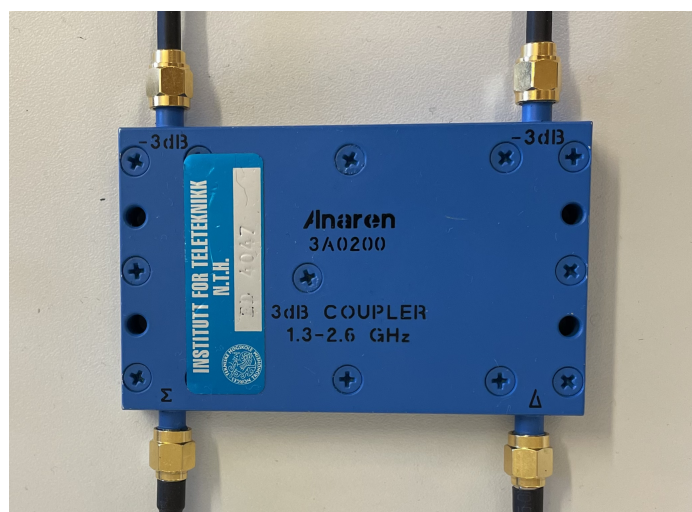
På grunn av vekten og dimensjonene til Yagi-Uda og antennens brede HPBW-sektor for de gitte kravspesifikasjonene i tabell 6, var det hensiktsmessig å gå for et antennearray. Et antennearray er lettere i vekt og har en smalere søkesektor. Patch-antennen i figur 14 er en av typen kvartbølgeantenne, og er dimensjonert etter  $\frac{\lambda}{4}$ . Den keramiske patch-antennen er sirkulærpolarisert, som beskrevet i databladet (Linx Technology 2019) og det er derfor mulig å plassere antennen på PCB-utlegget med vilkårlig orientering. Ved å konstruere et antennearray som beskrevet i 2.2.8, vil man kunne bruke søkemetoden for fase og motfase, beskrevet i 2.2.9, for å peile seg mot senderantennen som står et sted i rommet.



Figur 14: Linx Technologies, 2.4 GHz Keramisk patch-antenne

Etter undersøkelse av tilgjengelige komponenter fra flere leverandører ble det konkludert med at det vil være nødvendig å konstruere et PCB-utlegg. Patch-antennene og SMA-kontakter ble loddet på som overflatsmonterte komponenter. Konstruksjon av et PCB-utlegg gjør det mulig å modifisere avstanden mellom antenneelementene og kontrollere impedansen på microstrip-linjen.

Faseforskyvning mellom signalene fra de to antenne-patchene ble gjort ved bruk av en hybrid-kobler av typen Anaren 3A0200 3, ved å forlenge overføringslinjen til det ene signalet med  $\frac{\lambda}{2}$ , som beskrevet i 2.2.10. Kobleren er konstruert for frekvensområdet 1.3-2.5 GHz, som er innenfor frekvensbåndet som skal påtrykkes.



Figur 15: Anaren 3A0200 180° hybridkobler

For å alternere mellom signalene som kommer fra hybrid-kobleren ble det brukt et evalueringskort for svitsjing (se 3.7.2).

Ved å ta i bruk de overnevnte komponentene kan det konstrueres en krets som kan veksle mellom måling av de individuelle patch-antennene i fase og motfase. Vedlegg G viser RF-skjematikken for kretsen og inkluderer evalueringskortet for svitsjing, utviklingskortet nRF52D K med nRF52832 SoC, hybridkobleren og antennearrayet. SMA-koblingene ble tilstrammen med en momentnøkkel av typen HP 8710-1765

med et dreiemoment på 0.9 Nm for å sikre god forbindelse mellom kontaktene og for å oppnå konsekvente målinger.

### 3.6 Design av PCB-utlegg

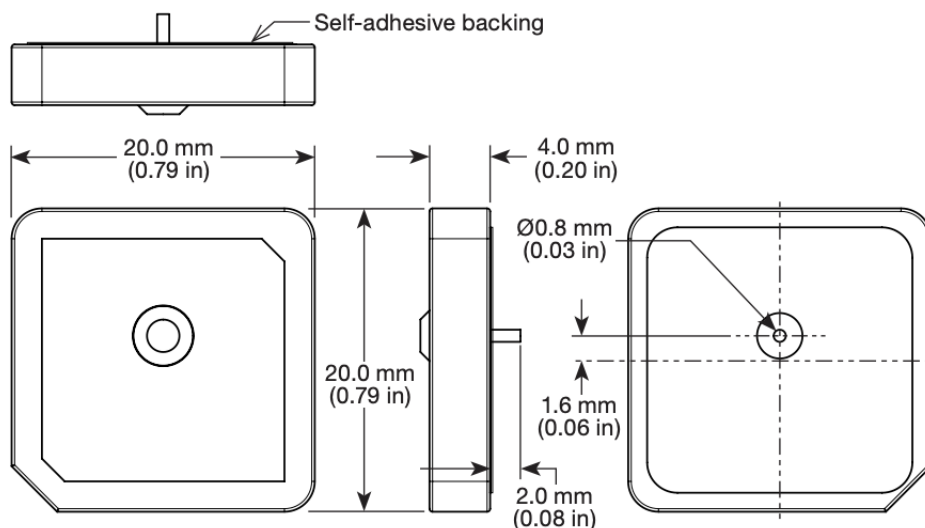
#### 3.6.1 1x2 Antenne-patch array

Antennen som ble designet er av typen patch-array og består av to patch-antenner montert på en PCB. PCB-utlegget ble derfor designet som en holder for antennene, og består av to SMA-kontakter og to microstrips som leder til en via der antennen skal kobles til.

PCB-utlegget er et to-lags PCB med FR4 substrat og kobberlag på begge sider. Tykkelsen på substratet er 1.6 mm med et kobberlag på 35  $\mu\text{m}$ . Det ble antatt en dielektrisk konstant  $\epsilon_r$  på 4.4, da den ligger mellom 3.8 og 4.8 (Peterson 2021). Baksiden av PCB-utlegget ble anvendt som jordplan, og forsiden som kombinert signalplan med omringende jord.

I henhold til kravspesifikasjonene i tabell 6 for antennen er det nødvendig å ha en avstand større enn  $\frac{\lambda}{2}$  for å unngå aliasing. For keramiske patch-antenner blir denne avstanden målt fra matekabelen til antennen. Frekvensen som blir påtrykt antennene er på 2.4 GHz og avstanden mellom antennene kan beregnes ved å bruke formel (15) med innsatte verdier:

$$d = \frac{c}{2 \cdot 2.4 \text{ GHz}} = 6.25 \text{ cm} \quad (27)$$



Figur 16: Linx Technologies, 2.4 GHz Keramisk patch-antenne

Jordplanet til de keramiske patch-antennene er plassert på baksiden av patchene. Jordplanet er adskilt fra matelinjen med en margin på 1.6mm, som vist i figur 16. For

å unngå kortslutning mellom matelinjen og jordplanet til antennen, ble det brukt design rules (Altium 2022) for å definere en margin mellom jordplanet og via på baksiden av utlegget på 2mm.

SMA-kontaktene ble lagt til ved å bruke forhåndsdefinerte som for SMA-kontakter for loddepunktene for å bidra til termisk avlastning. SMA-kontakten har 4 punkter for jordtilkobling og én senterleder for signalet som ble loddet til microstrip-linjen.

For å unngå uønsket strømgjennomgang på jordplanet, ble det lagt til via-er med jevne mellomrom på PCB-kortet. Via-ene går igjennom kortet og kobler begge kobberplanene til hverandre. En slik sammenkobling vil føre til at det blir bedre kontakt mellom komponentene som jordes på signalplanet og jordplanet. Det ble i tillegg lagt til 1.0 cm i lengden for å gi plass for SMA-kontaktene.

### 3.6.2 Impedansmatching

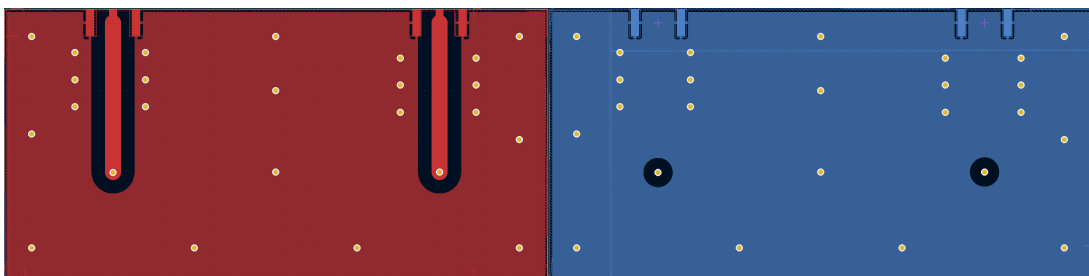
Microstrip-linjene ble designet til å ha en impedans etter kravspesifikasjonene i tabell 5. Tykkelsen på microstrip-linjene kan utledes ved bruk av formel (5), med hensyn på bredden til microstrip-linjen:

$$W = \frac{H - e^{Z \cdot \frac{\sqrt{\epsilon_r + 1.41}}{87}} \cdot T}{0.8 \cdot e^{Z \cdot \frac{\sqrt{\epsilon_r + 1.41}}{87}}} \quad (28)$$

Ved innsatte verdier for  $Z$ ,  $T$ ,  $H$  og en antakelse om at  $\epsilon_r = 4.4$  ved bruk av FR4-substrat (Peterson 2021), får man:

$$W = \frac{1.6 \cdot 10^{-3} \text{ m} - e^{50 \Omega \cdot \frac{\sqrt{4.4 + 1.41}}{87}} \cdot 35 \cdot 10^{-6} \text{ m}}{0.8 \cdot e^{50 \Omega \cdot \frac{\sqrt{4.4 + 1.41}}{87}}} = 0.00294 \text{ m} \quad (29)$$

Diskontinuitet i impedansen og impedans-mismatching vil kunne oppstå dersom det vil være en endring av tykkelse på microstrip-linjen. Diskontinuitet vil føre til signalrefleksjon på grunn av ulik impedans mellom overføringslinjene. For å redusere dette ble det gjort minimalt med endringer av banen til microstrip-linjen, ved å unngå svinger og endringer av tykkelsen.



Figur 17: Schematic-filen til PCB-utlegget der signalplanet er representert til venstre i rødt og jordplanet til høyre i blått

Figur 17 viser implementasjonen av den utregnede bredden på microstrip-linjen, plassering av jordplansvia-er samt marginene fra design rules. Ved å ta i bruk footprints for SMA-kontaktene kunne man legge til termiske pader, som begrenset varmetapet i kobberet under loddingen av komponentene.

### 3.6.3 Karakterisering av antennearray

For å finne returtapet i antennene ble det tatt i bruk en nettverksanalysator, som er et instrument som brukes for å måle impedans og refleksjoner av inngangssignalet for elektroniske kretser. For å få nøyaktige målinger må nettverksanalysatoren først kalibreres. Kalibreringen ble gjort ved hjelp av kalibrerings-instrumenter, som er elektriske komponenter med kjente, svært nøyaktige verdier. En nettverksanalysator av typen Rohde & Schwarz ZVA50 ble brukt for å påtrykke ulike frekvenser for å analysere returtapet for de gitte frekvensene. Videre ble disse grafene plottet og behandlet i resultater.

Målingen av strålingsmønsteret til antennearrayet med de to patch-antennene ble gjort på antennelabben som tilhører NTNU. I antennelabben blir målingene utført i et ekkofritt rom som har absorbenter som absorberer de elektromagnetiske bølgene. Det vil derfor ikke oppstå flerinterferens, som kan anses på ideelle testforhold. Antennearrayet ble brukt som mottakerantenne, og det ble brukt en retningsstyrt antenne som senderantenne i testmiljøet. Antennearrayet ble rotert  $360^\circ$  for å måle alle vinklene i asimutplanene ( $\theta$ ). Det ble brukt en nettverksanalalysator for å måle returtapet for en gitt asimutvinkel for mottakelse og sending av signaler. Strålingsdata som ble anskaffet under målingene ble anvendt til å definere antennekarakteristikken, som strålingsdiagram og HPBW. Målingene ble utført både i fase ( $\Sigma$ ) og motfase ( $\Delta$ ) da begge målingene brukes for søkemetoden beskrevet i 2.2.9. Under antennelabben ble det generert .h5-filer (HDF) som inneholder informasjon fra målingene som ble tatt. HDF er en filtype med hierarkisk formatert data. Denne inneholder data om vinklene som ble målt, styrken på målt signalt og rotasjonshastighet.

For miljøer som ikke er ekkofrie vil det oppstå flerveisinterferens som vil påvirke målingene som beskrevet i 2.2.5. Antennearrayet ble derfor testet i to ulike miljøer, et utendørs og et innendørs hvor det vil oppstå flerveisinterferens for å analyse ytelsen på systemet under ikke-ideelle forhold. Testene ble utført på en avstand på 1m, 5m, og 10m fra senderantennen og i likhet med målinger i antennelabben, ble det utført i både fase og motfase. De utvendige testene ble utført utenfor ved Elektrobygget på NTNU, mens testene inndendørs ble utført i Elektrobygget på NTNU, i et stort atrium for å redusere flerveisnterferens. Det ble utført 180 målinger for asimutplanet og 50 målinger for elevasjonsplanet, der én måling er ekvivalent med  $1^\circ$ . Dataen om vinkel og RSSI i fase og motfase lagret i en tekstfil. Dette gjorde det mulig å behandle dataen ved hjelp av Python, slik at informasjonen kan bli representert visuelt i form av ulike typer diagrammer.

For antennesystemet vil det i tillegg oppstå tap grunnet faktorer som siktlinje, refleksjoner og testomgivelser. Det var derfor nødvendig å utføre empiriske målinger for å estimere tapet som skyldes disse parameterene. Under testing av systemet ble

det tatt hensyn til fjern- og nærfeltene rundt antennen. Ved å sette inn verdier for diameteren til senderantennen, bekrrevet i 3.9.1, i formlene (9), (11) og (10) vil man få distansen for de ulike feltene:

$$R_{nearfield} = 0.62 \cdot \sqrt{\frac{(0.023 \text{ m})^3}{0.125 \text{ m}}} = 6.04 \cdot 10^{-5} \text{ m} \quad (30)$$

$$R_{farfield} = \frac{2(0.023 \text{ m})^2}{0.125 \text{ m}} = 8.46 \cdot 10^{-3} \text{ m} \quad (31)$$

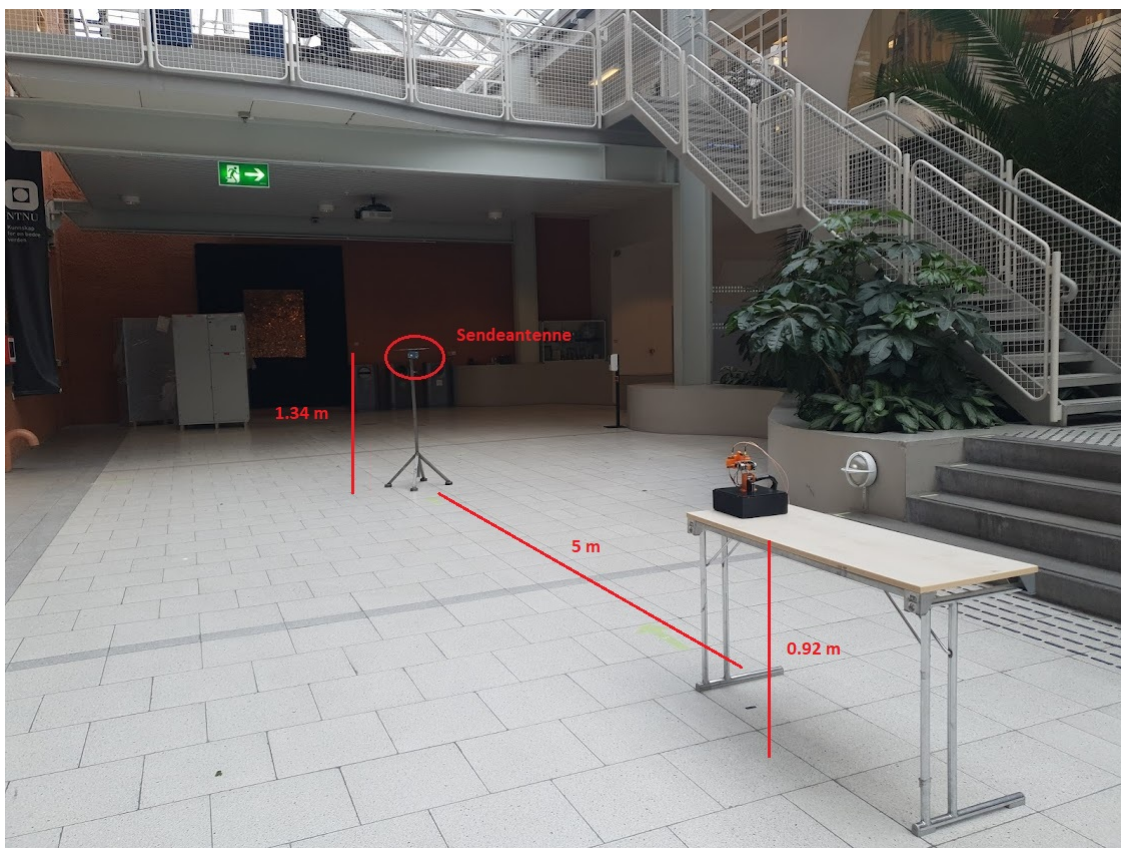
$$6.04 \cdot 10^{-5} \text{ m} < R_{radiative} < 8.46 \cdot 10^{-3} \text{ m} \quad (32)$$

Videre ble det også tatt hensyn til siktlinjen mellom sender- og mottakerantennene. Ved å utføre målingene i et større rom kunne objekter flyttes, slik at det ikke ble hindringer mellom antennen. For å unngå konstruktive og destruktive interferenser for signalet som ble propagert, ble det kontrollert at testmiljøet lå utenfor Fresnel-sonene.

Fresnel-sonene ble for testtriggene utregnet, ved hjelp av formel (12), til å ligge ved:

$$r = 17.32 \cdot \sqrt{\frac{0.005 \text{ km}}{4 \cdot 2.4 \text{ GHz}}} = 0.395 \text{ m} \quad (33)$$

Testtriggen bestod av et bord 92 cm over bakken og robotarmen ble plassert på enden, slik at den ikke interagererte med den første Fresnel-sonen. Mens senderantennen ble plassert på et stativ 134 cm over bakken. Nedenfor i figur 18 viser miljøet robotarmen ble testet i innendørs på 5m.



Figur 18: Miljø hvor innendørs testing ble gjennomført

Gjennom datablad og kjente parametere er det mulig å beregne det estimerte tapet i systemet. Dette gjøres ved å summere de kjente tapene, som i hovedsak vil oppstå i kontakter og ledninger. Komponentene i systemet som er blitt designet består også av mekaniske koblere og switcher, som beskrevet i 3.5.2. Dette medfører økt tap i ledninger og koblingspunkter, da disse er nødvendige for å behandle signalene. De passive tapene for kablene er beregnet ut fra standardverdien for kablene som ble anvendt i systemet, disse er av typen RG-316. For innsettingstapet for SMA-kontaktene ble det brukt verdiene som ble oppgitt av leverandøren av produktene 3.

## 3.7 Integreert system

### 3.7.1 Mikrokontroller

I prosjektet var det behov for å styre servomotorer, styre en laserdiode, hente posisjonsdata fra enkodere og motta radiosignaler ved hjelp av en retningsstyrt antenne. Utviklingskortet “nRF52 DK” fra Nordic Semiconductor innehar funksjonalitet for å tilfredsstille de nevnte kravene, og ble derfor anvendt som mikrokontroller i prosjektet. Denne baserer seg på nRF52832 SoC, og inkluderer alt av I/O, og funksjoner som trengs for å raskt og enkelt utvikle med produktet (Nordic Semiconductor 2020).



I tillegg ble det spesifisert i oppgavebeskrivelsen fra NTNU (2021) at det skal brukes nRF52 System on Chip i prosjektet.

### 3.7.2 Komponenter

#### Servomotorer

Servomotorer for styring av roboten i asimutplanet ble bestemt som i kapittel 3.3.1, og tilsvarende servomotor brukes for elevasjonsplanet. For å tilfrestille behovet for å rotere 1x2 antenne patch arrayet var det nødvendig med en servomotor av mindre karakteristikk som kunne monteres før endeffektor. I tillegg måtte den kunne enkelt roteres for hånd, om det var nødvendig å manuelt endre på antennerotasjonen uten å skade motoren. Derfor ble det valgt en mikroservo SER0049, som har en clutch funksjon. Dette vil si at når det ikke er påført spenning har den evnen til å rotere fritt. Når det er behov, kan utviklingskortet bestemme rotasjonen til antennen ved å påføre signal til mikroservo.

#### Enkoder

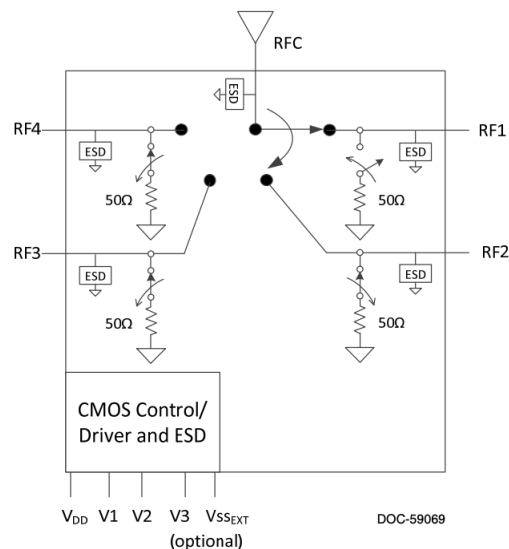
Servomotorene har ingen integrert tilbakekobling. Dette gjør at systemet ikke har noen måte å vite hva den faktiske posisjonen dens er. En løsning er å implementere enkoder separat, som ble gjort i det ferdige produktet. Siden servomotorene ikke har tilbakekobling ble det bestemt å bruke inkrementelle enkodere for posisjonstilbakemelding i asimut- og elevasjonsplanet. Inkrementelle enkodere tilbyr nøyaktig tilbakemelding om motorens relative posisjon, og er relativt billige.

#### Laserpeker

I prosjektet ble det benyttet en laserdiode på 5 mW i laserklasse IIIa. Denne laserdioden samsvarer ikke med NEK EN 60825-1, jf. Direktoratet for strålevern og atomsikkerhet (2019) men tilnærmes en klasse 3R-laser i prosjektet. Fra risikovurderingen i vedlegg E fremkommer det at laseren skal brukes innendørs påmontert en robotarm som kan orientere laseren 180° i horisontalplanet og 50° i vertikalplanet. Det vil si at laseren kan peke i alle retninger innenfor det området som spennes ut av robotarmen. For å forhindre at personer skal ta skade av laserstrålen iverksettes det et tiltak om at alle tilskuere ved bruk av systemet holder seg bak innretningen fra systemet er spenningsatt. Det må også sørges for at rommet systemet brukes i ikke har flater som kan reflektere laserstrålen tilbake til den sikre sonen bak innretningen. Det anses ikke som nødvendig å sette i bruk vernebriller på grunn av naturlige reaksjoner hos mennesker som nevnt i 2.3.1. Systemet er merket med symbolet for laserstråling med opplysninger om klassifisering og fysiske egenskaper i samsvar med “Veileder om sterke laserpekere (klasse 3R, 3B og 4)” av Direktoratet for strålevern og atomsikkerhet (2019). I programkoden er laserstrålen hindret fra å være aktiv fra radiopeilingen er ferdig, til eventuelt nytt søk startes.

## Svitsj

SoC nRF52832 har én inngang for RF-signaler, og det må derfor benyttes en svitsj for å veksle mellom hvilket RF-signal som mottas på utviklingskortet fra hybridkobleren. For å forenkle utviklingen ble det brukt et evalueringskort som tilbyr svitsjen ferdigmontert på PCB kort, med mulighet for å endre utgangssignal ved hjelp av pinner på kortet. For å svitsje mellom RF-signalene brukes GPIO-pinner på mikrokontrolleren til å styre evalueringskortet. En slik styring vil tillate digitalt styrt switching mellom RF-inngangene. Vekslingen mellom signaler må styres av mikrokontrolleren slik at riktig signal blir mottatt avhengig av om søkealgoritmen søker i sigma ( $\Sigma$ ) eller delta ( $\Delta$ ). Evalueringskortet består av 4 RF-innganger og 1 utgang og tar i bruk en digitalt styrt CMOS-kontroller (2.3.1). Inngangene har en impedans på  $50 \Omega$ , og har et isolasjon på tilnærmet lik 53 dB ved 2.4 GHz. De ubrukte inngangene blir terminert med en impedans på  $50 \Omega$  for å unngå diskontinuitet av impedansen for kretsen, som vist i 19.



Figur 19: PE42442 Evalueringskort

Tilstanden til evalueringskortet styres med en GPIO-utgang på utviklingskortet. Sannhetstabellen for svitsjingen på evalueringskortet ser ut som i tabell 8:

Modus	Pinne 2	Pinne 1
RF4 på	0	0
RF1 på	0	1
RF2 på	1	0
RF3 på	1	1

Tabell 8: Sannhetstabell for svitsjen på evalueringskortet (pSemi 2021)

I prosjektet benyttes det to patch-antenner, så det er bare behov for å bruke to av inngangene på evalueringskortet. RF4- og RF1-porten er benyttet i prosjektet.

Pinne 2 holdes alltid lav, så det kan veksles mellom mellom RF4- og RF1-porten ved å sette pinne 1 til logisk lav eller høy.

### 3.7.3 Tilkoblinger til utviklingskort

#### Konfigurasjon av GPIO-pinner

For å oppnå kommunikasjon med de ulike eksterne komponentene anvendt i prosjektet, ble GPIO-periferaleen på utviklingskortet benyttet. Utviklingskortet nRF52DK tilbyr 32 konfigurerbare GPIO-pinner, der pinne *P0.22* til *P0.31* er koblet i nærheten av radiomottaker og strømforsyning. For å unngå forstyrrelse av mottatt radiosignal er det derfor anbefalt å unngå å koble enheter med høye frekvenser og større strømmer på disse pinnene (Nordic Semiconductor 2021d). Robotinnretningen er avhengig av at radiomottaker fungerer optimalt, derfor er det viktig å benytte andre GPIO-pinner enn de tidligere nevnt om det skal kobles komponenter som produserer høye frekvenser og strømmer. Pinne *P0.09* og *P0.10* er koblet til NFC-antennen og bør unngås da det er begrensninger på disse for å beskytte kortet for potensielle skader, men kan konfigureres som vanlig GPIO ved å endre koblinger på kortet. Tabell 9 viser de anvendte GPIO-pinnene i prosjektet:

Tilkoblet enhet	GPIO pin	I/O
Servo, asimut	P0.02	Output
Servo, elevasjon	P0.03	Output
Servo, antennerotasjon	P0.04	Output
Knapp 1, utviklingskort	P0.13	Input
Knapp 2, utviklingskort	P0.14	Input
Knapp 3, utviklingskort	P0.15	Input
Knapp 4, utviklingskort	P0.16	Input
Taktilbryter, 0°	P0.22	Input
Taktilbryter, 90°	P0.23	Input
Enkoder, asimut CH a	P0.26	Input
Enkoder, asimut CH b	P0.27	Input
Laser	P0.28	Output
Evalueringskort	P0.29	Output
Enkoder, elevasjon CH a	P0.30	Input
Enkoder, elevasjon CH b	P0.31	Input

Tabell 9: GPIO-tilkoblinger

#### Tilkobling av komponenter

For å koble eksterne komponenter til utviklingskortet ble det benyttet dupontkabler. For å inkludere elektriske komponenter ble et breadboard brukt som mellomledd hvor komponentene kunne kobles sammen med tilhørende komponenter. I tilfelle

med servomotorene og svitsjen kunne signallinjene kobles direkte til utviklingskortet, og strøm og jord kobles respektivt hvor det er aktuelt. Ved oppkobling av enkodere hadde signallinjen for høy spenning for GPIO pinnene på utviklingskortet, og måtte derfor gå gjennom spenningsdelere som trappet ned spenningen til et akseptabelt nivå. Spenningen ut fra enkoderene var på 4.7 V, som er over det høyeste utviklingskortet tåler på 3.9 V (Nordic Semiconductor 2021a). Det ble brukt motstander på 1 k $\Omega$  og 1.5 k $\Omega$  for å trappe ned spenningen, disse verdiene ble valgt fordi det var mest praktisk i forhold til hvilke motstander som var tilgjengelige og førte til en spenning som vist i ligning 34.

$$V_{ut} = V_{inn} \frac{R_2}{R_1 + R_2} = 4.7 \text{ V} \frac{1500\Omega}{1000\Omega + 1500\Omega} = 2.87 \text{ V} \quad (34)$$

I tilfellet med laseren hadde den behov for mer strøm enn utviklingskortet kunne tilby gjennom GPIO utgangene. Derfor ble signallinjen koblet til en transistor som opererte som en knapp. Transistoren er koblet til 5 V fra utviklingskortet, som har evnen til å tilføre nok strøm til laserdioden. Når utviklingskortet sender signal om at laseren skal skrus på, sperres strømgjennomgang over transistoren slik at den heller går gjennom laserdioden. Når signalet går lavt, flyter strømmen heller gjennom transistoren og laserdioden skrus av.

## Strømforsyning

De eksterne komponentene i systemet som ble koblet til utviklingskortet hadde behov for strømtilførsel. Enkoderene, SER0049-servomotoren og laserpekeren som hadde behov for 3 eller 5 VDC er forsynt fra utviklingskortet. SER0038-servomotorene krever forsyningsspenning på 7.2 VDC for å kunne levere det annonserte 17 kg\*cm ( $\approx$  1.67 Nm) dreiemomentet. Servomotorene ble derfor forsynt fra en ekstern strømkilde.

### 3.7.4 Biblioteker og drivere (kode)

For å oppnå pålitelig styring av robotinnretningen, innhenting av sensordata og prosessering av data må det utvikles robuste drivere og bibliotek for å kontrollere de diverse komponentene som er nødvendige for styring av robotarmen. Mange av disse har kjente metoder for styring, men som må tilpasses utviklingskortet og koden den er bygd på. I noen tilfeller er det også nødvendig å utvikle kode.

## PWM-driver

For å styre servomotorene ble det utviklet et eget bibliotek med funksjonalitet for styring av servomotorer. Teori bak PWM signal finnes i 2.3.5. Etter anbefaling fra Edvind Holmseth, Application Engineer i Nordic Semiconductor (samtale under kurs på NTNU Trondheim, 8. mars 2022) egnet det seg bedre å styre servomotor ved å

generere et PWM signal med timer og GPIOE (se 2.3.3). Det ble også henvist til eksempelkode på GitHub (edvinand 2021). Koden på Github var originalt skrevet for styring av LED lys, men ble modifisert for å være kompatibel med servomotorer.

Koden baserer seg på timer, GPIOE og PPI (se henholdsvis 2.3.3, 2.3.3 og 2.3.3) for å generere et PWM signal med ønsket arbeidssyklus. Timeren settets opp slik at den har en periode som samsvarer med spesifikasjoner til servomotor, som i dette tilfellet er på 50 Hz. Når servomotoren initialiseres, assosieres den med en GPIO-pinne som den kobles til for styring. Ved hjelp av PPI samhandler timeren med GPIOE slik at den assosierte pinnen holdes høy inntil den spesifiserte arbeidssyklusen har forløpt, deretter settes pinnen lav i resten av perioden. Til slutt blir arbeidssyklusen bestemt ved å skrive posisjon i “tics” (se 2.3.3) til timer-registeret. For å enklere skrive vinkler ble det også utviklet en funksjon som omgjør vinkler i grader til tics, slik at ønsket vinkel på servomotoren kan oppgis i vinkelgrader.

### **Programmering av knapper, enkoder, laser og svitsj**

Det var nødvendig at utviklingskortet kontrollerte og tok mot data fra flere enheter. For å manuelt kontrollere robotinnretningen ble det brukt fire knapper som var forhåndsinstallerte på kortet. Det ble programmert interrupts (se 2.3.3) på disse som viste til ISR som utførte ønsket handling. For å hente inn data fra enkoderene ble QDEC-biblioteket fra nRF connect SDK-et, som samler inn akkumulerte pulser fra de inkrementelle enkoderene (se 2.3.1). Ved å hente ut de akkumulerte pulsene fra QDEC-registeret kan posisjonen til servomotorene bestemmes og huskes for å vende tilbake til samme posisjon senere. Det kan også verifiseres at servomotoren faktisk har ankommet ønsket posisjon. Svitsjen og laserpekeren ble kontrollert ved å sende logisk høy eller lav fra GPIO-utganger (se 2.3.3).

## **3.8 Algoritme og programmering**

Ettersom robotarmen skal kunne peile seg inn på en Bluetooth-sender dannes det et behov for å utvikle og programmere en søke- og databehandlingsalgoritme. Det er blitt tatt i bruk programvarene VS Code og NRF Connect for Desktop for å utvikle en programkode som ved bruk av RSSI-målinger skal kunne avgjøre retningen til sender antennen. VS Code er et redigeringsverktøy for programkode som er kompatibelt med NRF Connect for Desktop som gir brukeren verktøy for å utvikle applikasjoner. Programkoden har blitt utviklet i programmeringsspråket C og tar i bruk kodeprinsippene som ble gått igjennom i 2.4. I tillegg ble det opprettet et “Git”-miljø under programmering av programkoden for å sikre mot konfliktende kode og for å ha versjonskontroll på koden.

Ved utvikling av programkoden til robotarmen er det blitt tatt i bruk biblioteker som “Toolchain Manager” fra NRF Connect for Desktop. Dette er blitt gjort ved å ta i bruk *header*-filer. I tillegg til bruk for importering av biblioteker er det blitt brukt *header*-filer mellom de ulike kildefilene som er blitt utviklet under prosjektet.

### 3.8.1 Initialisering og konfigurering

For at applikasjonen skal kunne ta i bruk biblioteker fra NRF Connect SDK var det nødvendig å opprette en *prj.conf*-fil. Filen inneholder konfigurasjonsalternativene til programkoden som ble utviklet. For å finne ut av hvilke konfigureringer som var nødvendig for vår programkoden ble kildekodene fra Zephyr (2022c) undersøkt for relevante funksjonaliteter.

For at applikasjonen skal peile etter senderantennen må de nødvendige modulene som programkoden er avhengig av bli initialisert. Under initialisering vil modulene gi en exception om det oppstår noen problemer med initialiseringen. Ved eventuelle feil vil feilkoden bli skrevet ut i terminalen og programkoden vil bli avbrutt.

### 3.8.2 Bluetoothkommunikasjon

Som nevnt i tidligere avsnitt ble det brukt innholdet nRF Connect SDK tilbyr. Spesifikt under prosjektet ble eksempelkode for Zephyr (2022b) og Zephyr (2022a) benyttet for sending og mottak av Bluetooth-pakker. Disse eksempelkodene former grunnlaget til programkoden som er blitt utviklet og følger prinsippene nevnt i 2.5.4. *Observatoren* opererer etter behandler-prinsippet som ble gått i gjennom i 2.4.6 og tar i bruk en *callback*-funksjon for å utføre en spesifikk handling ved en interrupt i form av et bluetooth signal. For å unngå avlesning av datapakker sendt fra andre kilder enn senderantennen, ble det implementert et filter som ignorerer signaler fra andre kilder. For å kunne ta i bruk dette filteret må MAC-adressen til mikrokontrolleren som sender signaler være kjent og legges inn i programkoden.

### 3.8.3 Styring av servomotorer

Servomotorene til robotarmen ble styrt med flertråds programmering som nevnt i 2.3.4. Dette gjør at servomotorene kan endre posisjon samtidig som programkoden kjører algoritme for søk og databehandling. Ved at servomotorene kontinuerlig får et PWM-signal fra utviklingskortet, vil motorene motarbeide ytre påvirkninger slik at posisjonen ikke endrer seg. Servomotorene til robotarmen har et definert arbeidsområde på 0-270° for SER0038-servomotorene og 0-180° for SER0049-servomotoren. Siden arbeidsområdet for robotarmen er begrenset til retningspeiling fra 0-180° i asimutplanet og 0-50° i elevasjonsplanet, ble det implementert begrensninger i programkoden på hvor langt de individuelle motorene kan manøvrere seg (1.2).

Tilbakemelding på posisjonen til motorene kommer fra inkrementelle enkodere, som nevnt i 3.7.2. Mikrokontrolleren som ble benyttet under prosjektet hadde kun mulighet til å prosessere informasjonen fra én enkoder om gangen. Ettersom det ble brukt enkodere for servomotorene både i asimut- og elevasjonsplanet ble det implementert en funksjon for å kunne bytte mellom disse. Både programkoden til enkoderen og trykknappene på utviklingskortet benytter seg av prinsippet *behandlere*, som ble gått gjennom i 2.4.6. Trykknappene på utviklingskortet ble tatt i bruk for å manuelt

styre servomotorene i asimut- og elevasjonsplanet etter fullført søk og for å skrive ut posisjonen til servomotorene i nRF Terminal-en i VS Code.

Basert på spesifikasjonene til servomotoren gitt fra DFRobot (2022) og begrensningene som kom ved å bruke QDEC-biblioteket fra nRF Connect SDK, var det nødvendig å begrense hastigheten til servomotorene. Styringen som ble implementert vil gradvis inkrementere vinkelposisjonen i stedet for å bevege den direkte til den gitte posisjonen. Dette vil gi enkoderne nok tid til å sample de genererte pulsene som oppstår når motorene roterer.

### 3.8.4 Søkealgoritme

Søkemethoden til programkoden kan deles opp i to deler. Den første delen som blir gjennomført er et grovsøk først i asimutplanet og deretter i elevasjonsplanet. Under grovsøket vil robotarmen starte med ta en måling i både fase ( $\Sigma$ ) og motfase ( $\Delta$ ) før den ved bruk av semaforer signaliserer at målinger er gjennomført og posisjonen til servomotoren endres med ett steg. Prosedyren vil gjentas inntil målinger for hele søkesektoren til det horisontalplan er utført. Målingene vil deretter bli prosessert av programkoden og robotarmen vil bevege seg til retningen den tror senderantennen er plassert. Robotarmen roterer antennen  $90^\circ$  om x-aksen og gjennomfører samme prosedyre for hele søkesektoren i vertikalplanet.

Den andre delen av søkemethoden innebærer et finsøk etter samme struktur som grovsøket, men vil gjøre 10 målinger for hvert målepunkt og kalkulere gjennomsnittet. Etter gjennomført finsøk vil målingene bli prosessert. Dersom lokasjonen til senderantennen ikke er veldefinert vil prosedyren bli gjentatt (se 2.4.5). Søkesektoren til finsøket vil være basert på resultatene fra den forrige iterasjonen av søk og bli innsnevret for hver iterasjon av søk.

### 3.8.5 Databehandling

I 3.5.2 ble det slått fast å ta i bruk et antennearray med keramiske patch-antenner. Et antennearray medførte utfordringer i form av riktig avlesning og databehandling av målt RSSI-verdi på utviklingskortet. Det ble dermed opprettet en **typedef struct** som inneholder avlesninger for RSSI-målinger i fase ( $\Sigma$ ), motfase ( $\Delta$ ) og tilhørende enkoderverdi for servomotorene. Videre under databehandlingsdelen av programkoden var det behov for å sette og endre verdiene til i arrayene, som et argument til en funksjon. Som nevnt i 2.4.5 har ikke en funksjon muligheten til å returnere et array i programmeringsspråket C, så problemet ble løst ved å ta i bruk peker-operatorene forklart i 2.4.4 sammen med *pass by reference*-prinsippet.

For at programkoden skal kunne lokalisere senderantennen, ble det utviklet algoritmer for å prosessere målingene som ble utført, da i form av validering av målepunkter, avgjøre grenseverdi for RSSI i fase ( $\Sigma$ ) for kontrollering av nullpunkt og lokalisering av nullpunkt. Algoritmen for validering av målepunkter går gjennom gjennomførte målinger i fase ( $\Sigma$ ) og motfase ( $\Delta$ ) for å kontrollere at RSSI-verdiene

er innenfor den gitte terskelen for signalnivået, da -25 som øvre grense og -90 som nedre grense. Grensene ble funnet ved å plassere robotarmen foran senderantennen og lese av signalstyrken når robotarmen var orientert mot senderantennen og 180° unna. Ved deteksjon av ugyldige måleverdier vil algoritmen annullere de ugyldige målepunktene og oppdatere størrelsen til arrayet.

De validerte målingene blir deretter sendt for beregning av grenseverdien i fase. Algoritmen vil gå gjennom hver måling i fase ( $\Sigma$ ) og ut fra sterkeste målepunkt vil det bli utarbeidet en grenseverdi.

Til slutt vil algoritmen for lokalisering av nullpunkt gå igjennom det oppdaterte arrayet med måledataene. Algoritmen vil sammenligne signalstyrkene for motfase ( $\Delta$ ) for alle målepunktene mot hverandre og validere eventuelle nullpunkt ved å ta i bruk grenseverdien. Målepunktet som inneholder laveste RSSI-verdi i motfase ( $\Delta$ ) og en RSSI-verdi over grenseverdien i fase ( $\Sigma$ ) vil da inneholde enkoderverdien til nullpunktet.

## 3.9 Bluetooth

### 3.9.1 Senderantennen

Senderantennen på utviklingskortet nRF52DK er en kvartbølge-monopolantenne 3. Antennen har en sendeeffekt på 4 dBm og er omnidireksjonell. Lengden på senderantennen er 23 mm og har en tykkelse på 1.5 mm (Nordic Semiconductor 2021f).

### 3.9.2 Anvendning av BLE i applikasjonen

Det blir brukt Bluetooth Low Energy for kommunikasjonen mellom antenneinnretningen (Cypress Semiconductor 2015) og senderantennen. Dataen for antenneinnretningen og senderantennen prosesseres av mikrokontrollere av typen nRF52832, som nevnt i 3.9.1 og 3.7.1. GAP-protokollen ble brukt for å oppnå tilkoblings-fri kommunikasjon mellom mikrokontrollerne, dette i form av rollene; kringkaster og observatør, som beskrevet i 2.5.4.

Signalstyrken som ble brukt for sendermottakeren er på 4 dBm, som nevnt i 3.9.1. Dette kategoriseres som bluetooth-klasse 2 i henhold til tabell 2. Tidsintervallene for annonsering, skanning-periode og skanningmellomrom var satt til å være [100 ms, 150 ms], 30 ms og 60 ms, respektivt.

## 3.10 Empirisk testing av fullstendig system

Ved å utføre empiriske tester i henhold til kravspesifikasjonene gitt i tabell 7 kan man beregne ut presisjonen og nøyaktigheten til det fullstendige systemet. I tillegg til testingen ved 5 m gitt fra kravspesifikasjonene, ble det også testet på avstandene 1 m

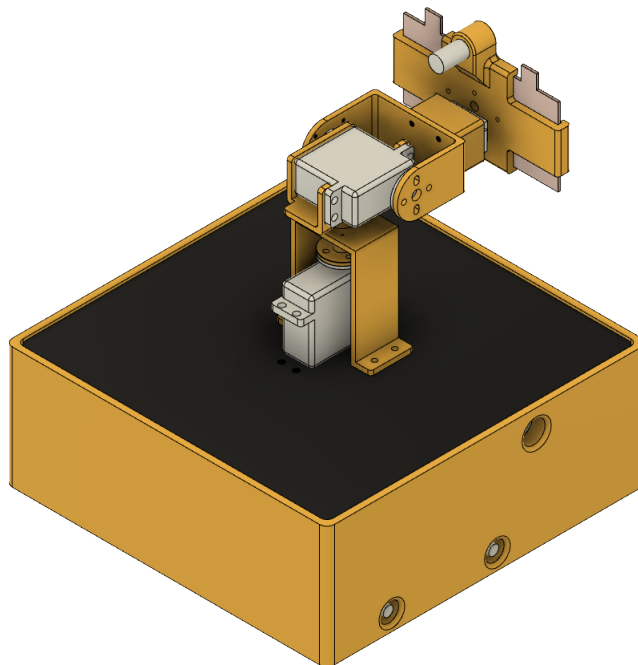


og 10 m. Testene ble i det samme miljøet som testing for beregning av flerveisinterferens og det ble brukt den samme testtriggen (3.6.3). For å minimere innvirkningen av eventuell feilmålinger, ble det gjennomført 10 målinger for hver avstand. Dataen ble som i 3.6.3 lagret i en tekstfil, slikt at resultatet kan visualiseres og analyseres.

## 4 Resultater

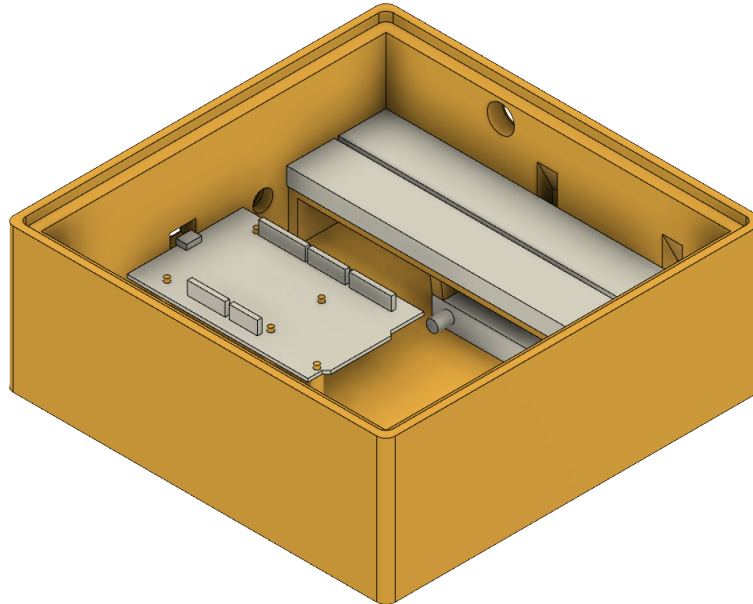
### 4.1 Design av robotarm

En 3D-modell for den komplette robotarmen kan ses i figur 20. I denne figuren betyr deler i gult og svart at de er produsert i PLA, POM-C og aluminium. Baseplaten i svart er maskinert ut ved mekanisk verksted ved ITK av POM-C-materialet. Deler i grått er ment som illustrasjoner for komponenter slik som asimut-, elevasjons- og endeffektorservo. Resterende deler er 3D-printet i PLA-materialet. Følgende 3D-modeller er ikke i perspektiv.



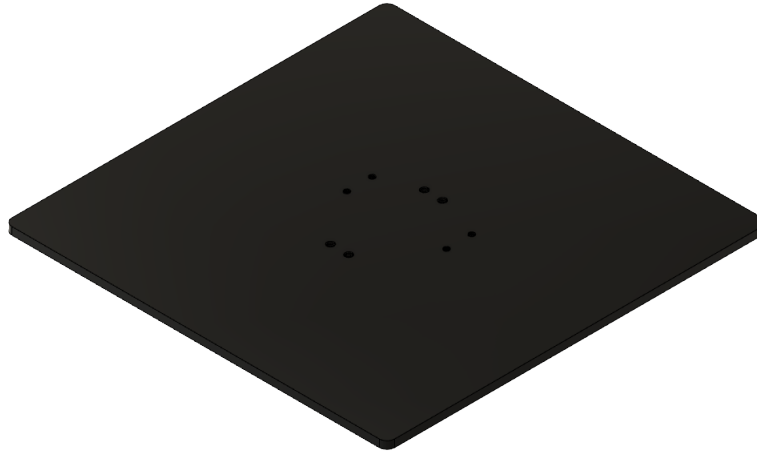
Figur 20: 3D-modell av robotarmen med komponenter og deler

Figur 21 viser endelig design av kontrollboksen med illustrasjoner i grått for nRF52-utviklingskortet, hybrid-kobleren og koblingsbrettet:



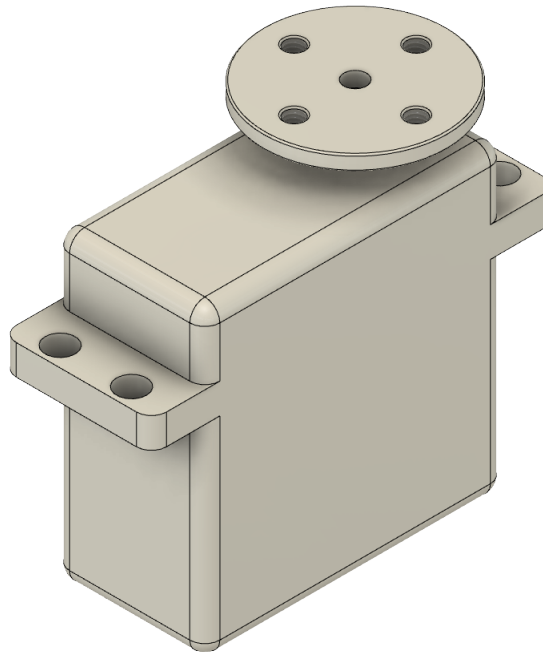
Figur 21: 3D-modell av kontrollboksen til robotarmen

På baseplatemodellen er det tegnet inn fire M4 gjengede monteringshull for å feste asimutservoen til basen og fire M3 gjengede monteringshull for å feste enkoderholderen for enkoder i asimutplanet:



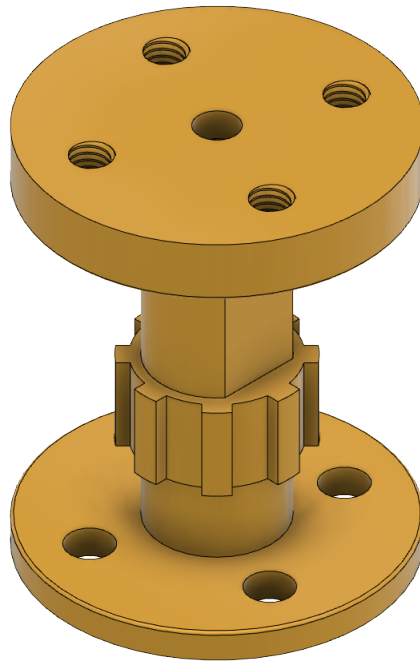
Figur 22: 3D-modell av baseplaten til robotarmen

En 3D-modell for SER0038-servomotoren kan ses i figur 23. Servomotormodellen benyttes til å illustrere dimensjoner til servomotoren i samspill med de resterende delene på robotarmen.



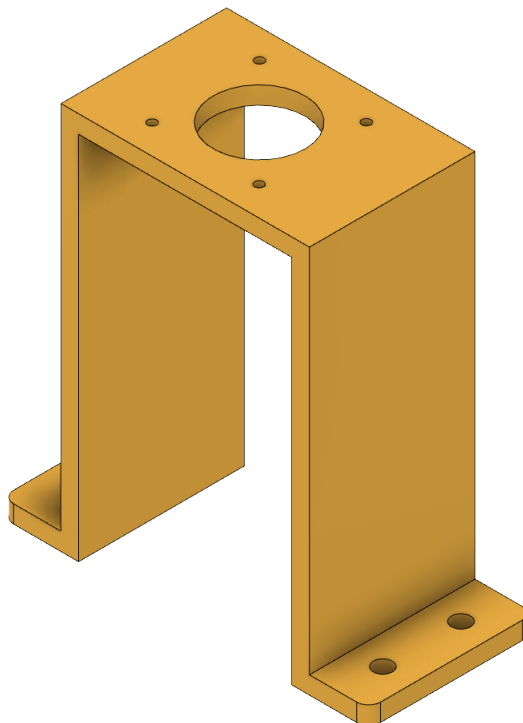
Figur 23: 3D-modell av servomotoren DFRobots SER0038

I figur 24 kan man se akslingen mellom asimutservoen og festeanordningen til elevasjonservoen med påmontert topplate:



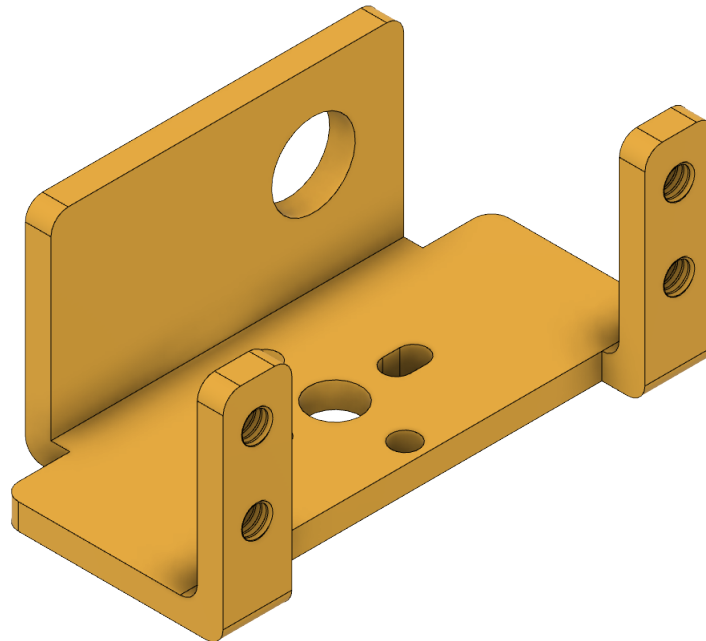
Figur 24: 3D-modell av akslingen mellom de to roterende leddene til robotarmen

Festeanordningen for asimutenkoderen er gitt i figur 25 og har en utsparing i toppen for gjennomtrekking av akslingen mellom asimut- og elevasjons servoen. I samme figur er to av monteringshullene for montering på baseplaten også synlige.



Figur 25: 3D-modell av festeanordningen for enkoder i asimutplanet

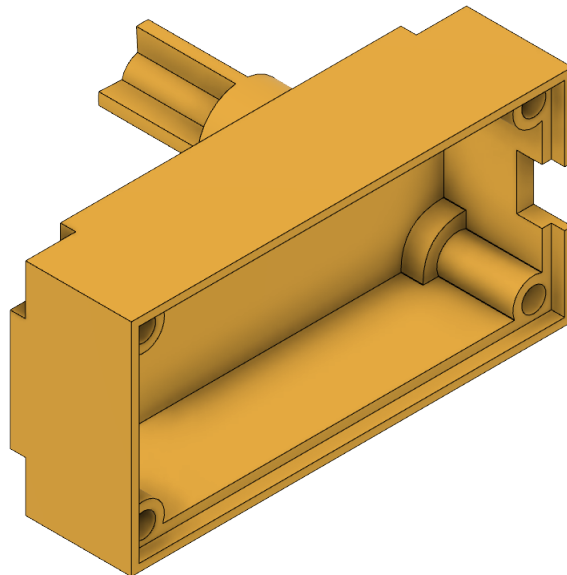
Festeanordningen for elevasjonsservoen med utsparing for egenprodusert servobunn:



Figur 26: 3D-modell av servomotorfestet til elevasjonsservoen til robotarmen

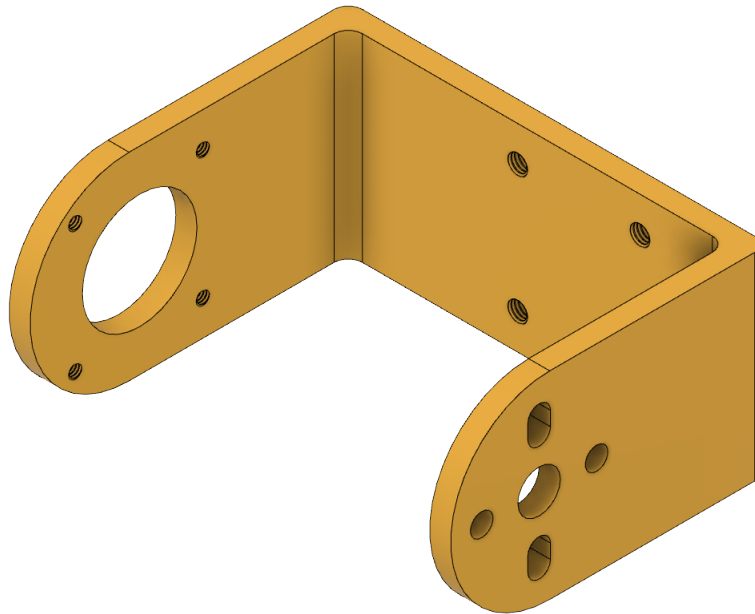


En illustrasjon for elevasjonsservobunnen kan ses i figur 27:



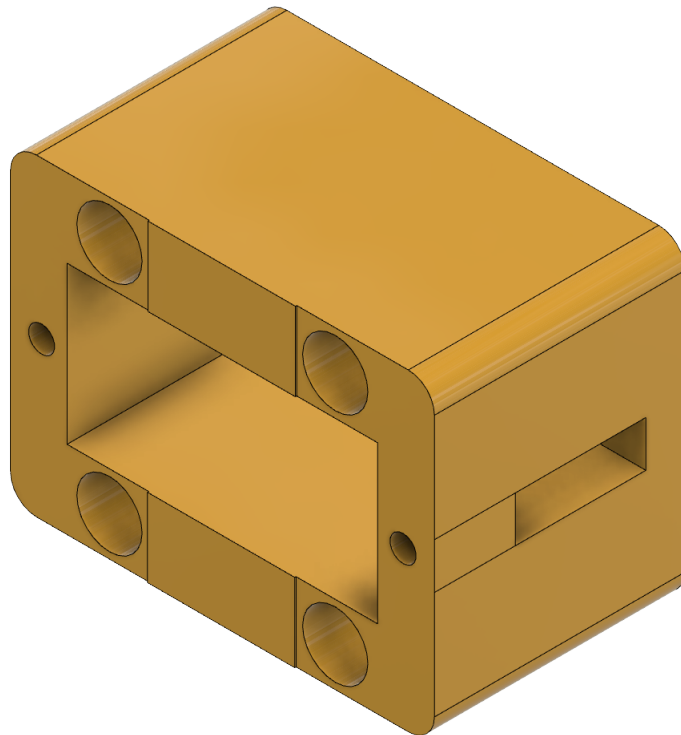
Figur 27: 3D-modell av nedre del av elevasjonsservoens til robotarmen

Den roterende braketten til elevasjonsservoen kan ses i figur 28, med utsparing for hjullager, monteringshull for enkoderbrakett og monteringshull for feste til SER0049-servomotoren:



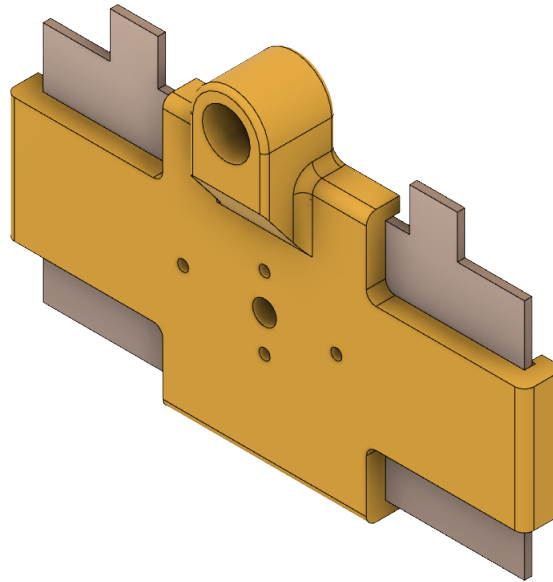
Figur 28: 3D-modell av det roterende festet til elevasjonsservoen til robotarmen

Festeanordningen for SER0049-servoen er illustrert i figur 29:



Figur 29: 3D-modell av festeanordningen for servomotoren DFRobots SER0049

I figur 30 er festeanordningen for antennearrayet og laserpekeren illustrert:



Figur 30: 3D-modell av antennefestet til robotarmen med forenklet illustrasjon for antennearray

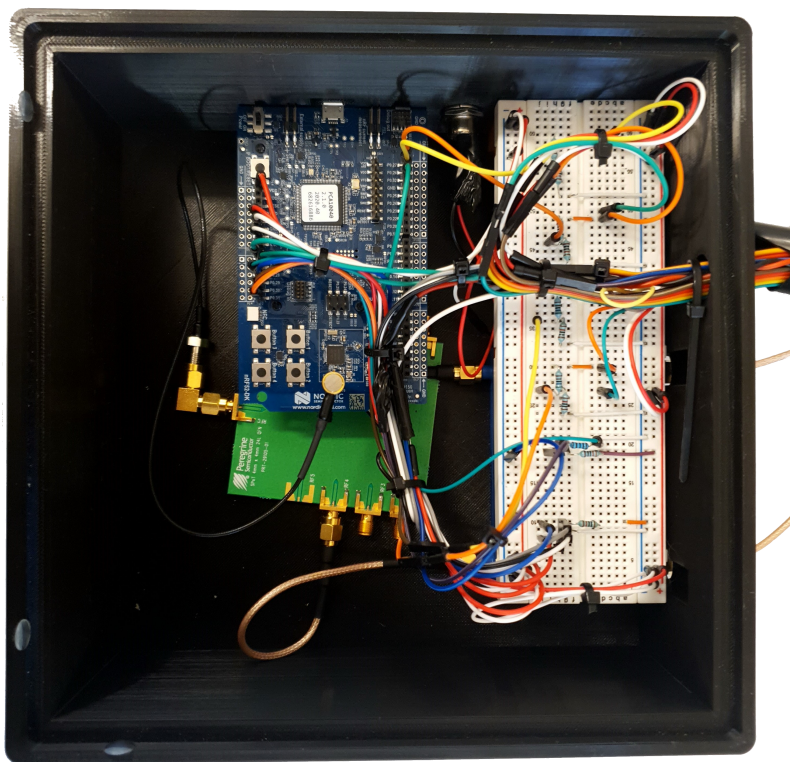
## 4.2 Integreerte systemer

### 4.2.1 Koblingsskjema

Det ble utviklet et koblingsskjema for dokumentasjon av alle nødvendige koblinger i det integrerte systemet. Nødvendige koblinger til og fra utviklingskortet samt på koblingsbrettet er illustrert i koblingsskjemaet i vedlegg F.

### 4.2.2 Ferdig oppkobling

Den endelige oppkoblingen av systemet er vist i figur 31. Koblinger ble gjort med jumper-kabler, hvor koblingsbrettet fungerte som mellomledd for å koble opp motstander og transistor, og for å oppnå delt jording mellom motor og utviklingskort.



Figur 31: Ferdig oppkobling av integrert system

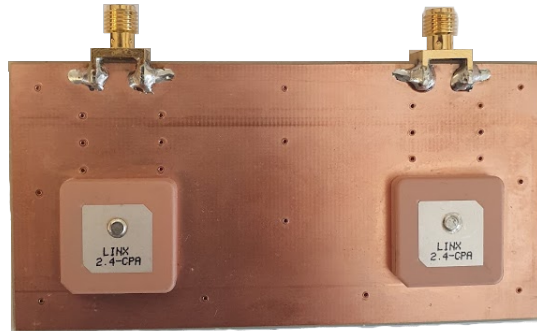
## 4.3 Algoritme og programmering

Resultatene til programmering og utvikling av applikasjonen kan bli funnet i vedlegg I og beskrivelse av funksjoner og definisjoner kan bli funnet i Github repository-et Huynh mfl. (2022). I tillegg kan testskjema bli funnet i vedlegg H

## 4.4 Antennearray

### 4.4.1 PCB-utlegg for 1x2 patch-antennearray

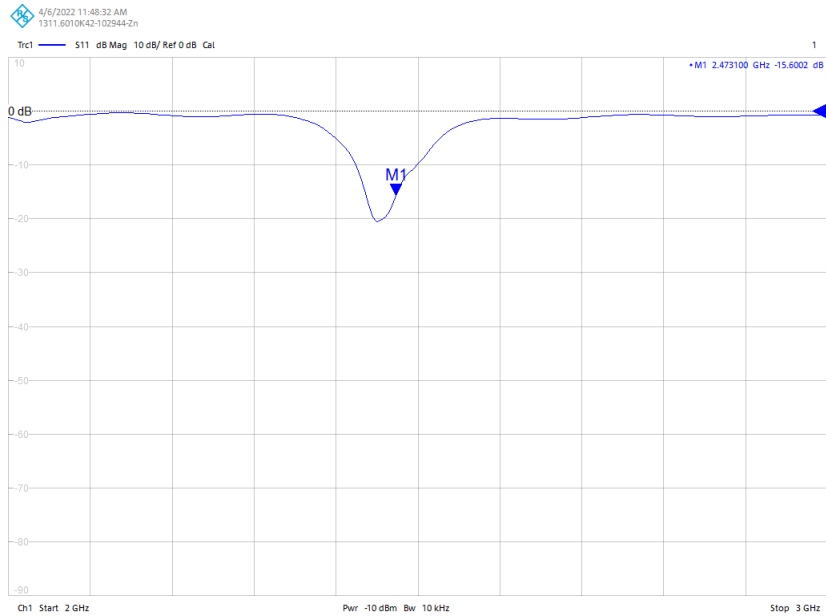
Antennens PCB-utlegg ble printet ut og loddet på som vist i figur 32. Jordingen til patch-antennene som vist i figur 16 ble limt på jordplanet, og senterlederen ble loddet på signalplanet til utlegget. Deretter ble SMA-kontaktene festet på de markerte sporene på kretskortet, som vist i figur 17.



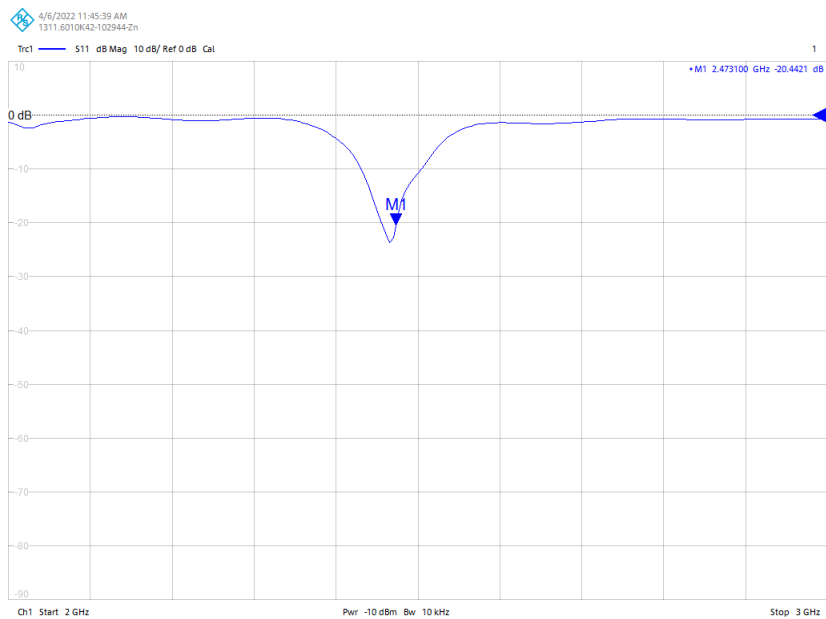
Figur 32: PCB-utlegg med på-loddede patch-antenner: Antenne 1 (venstre) og Antenne 2 (høyre)

### 4.4.2 Resultater av antennelaben

Resultatene fra impedansmålingen er vist i figur 33 og 34. Impedansmålingene viser returtapet til antennene og er blitt plottet i et dB-diagram. Den horisontale aksene viser frekvensspekteret og den vertikale aksene viser returtapet i desibel. Markøren i de respektive figurene indikerer returtapet for 2.4 GHz. Antenne 1 har et returtap på -15.6 dB og antenne 2 har et returtap på -20.4 dB.



Figur 33: Strålingsdiagram Antenne 1 (dB)



Figur 34: Strålingsdiagram Antenne 2 (dB)

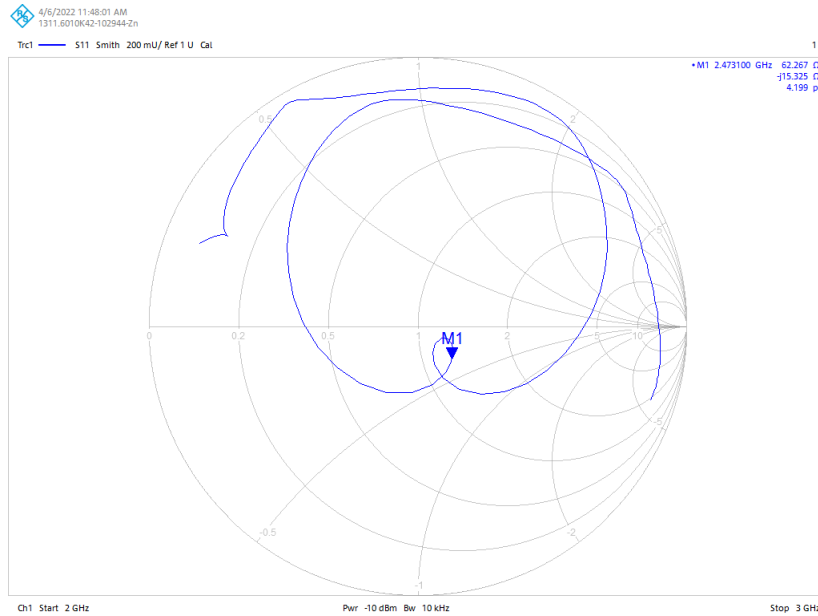
Båndbredden ble funnet ved å se på punktene for returtapet ved  $VSWR = 2 \equiv -10$  dB, som beskrevet i 2.2.1. For antenne 1 vil disse punktene være ved  $\approx 2.424$  GHz og  $\approx 2.498$  GHz som gir en båndbredde på:

$$BW = 2.498 \text{ GHz} - 2.424 \text{ GHz} = 74 \text{ MHz} \quad (35)$$

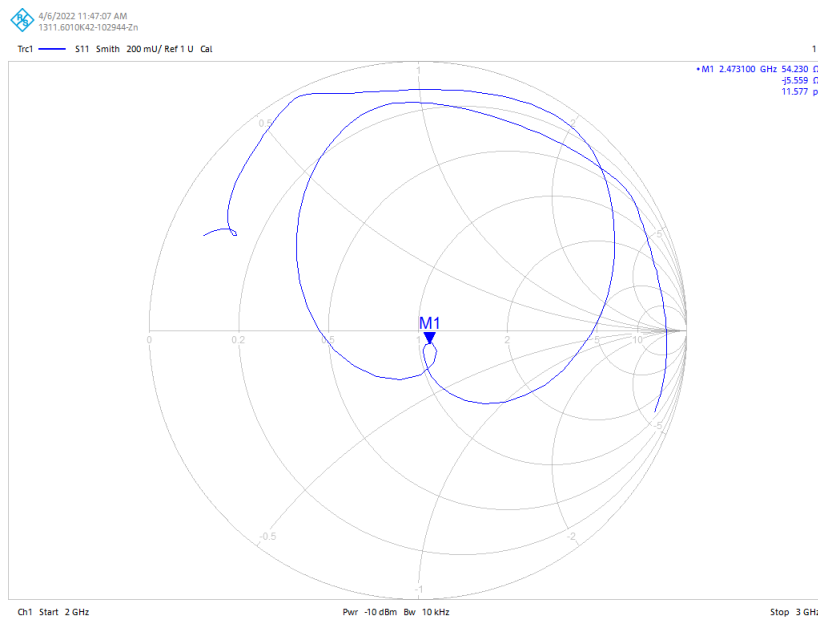
Antenne 2 har punktene for  $VSWR = 2$  ved  $\approx 2.430$  GHz og  $\approx 2.503$  GHz, som gir en båndbredde på:

$$BW = 2.504 \text{ GHz} - 2.432 \text{ GHz} = 72 \text{ MHz} \quad (36)$$

Smith-diagrammene i figur 35 og 36, skisserer impedansen ved et gitt frekvensspekter, der punktet til venstre representerer kortslutning ( $0 \Omega$ ) og punktet til høyre vil representere en åpen krets ( $\infty \Omega$ ).



Figur 35: Strålingsdiagram Antenne 1 (Smith)



Figur 36: Strålingsdiagram Antenne 2 (Smith)

Markøren i Smith-diagrammet viser at impedansen til antennene er  $Z_1 = (62.267 - j15.325) \Omega$  for antenne 1 og  $Z_2 = (54.230 - j5.55) \Omega$  for antenne 2.

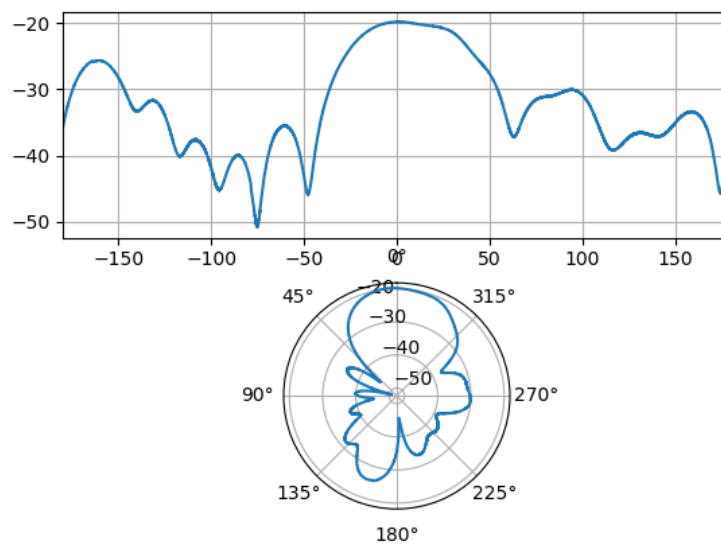


Refleksjonskoeffesienten  $\Gamma$  finnes ved å bruke formel (6). Med innsatte verdier for den målte impedansen og referanseimpedansen vil refleksjonskoeffesienten bli:

$$\Gamma_{ant1} = \left( \frac{(62.267 - j15.325) \Omega - 50 \Omega}{(62.267 + j15.325) \Omega + 50 \Omega} \right) = 0.1733 \quad (37)$$

$$\Gamma_{ant2} = \left( \frac{(54.230 - j5.550) \Omega - 50 \Omega}{(54.230 + j5.550) \Omega + 50 \Omega} \right) = 0.0668 \quad (38)$$

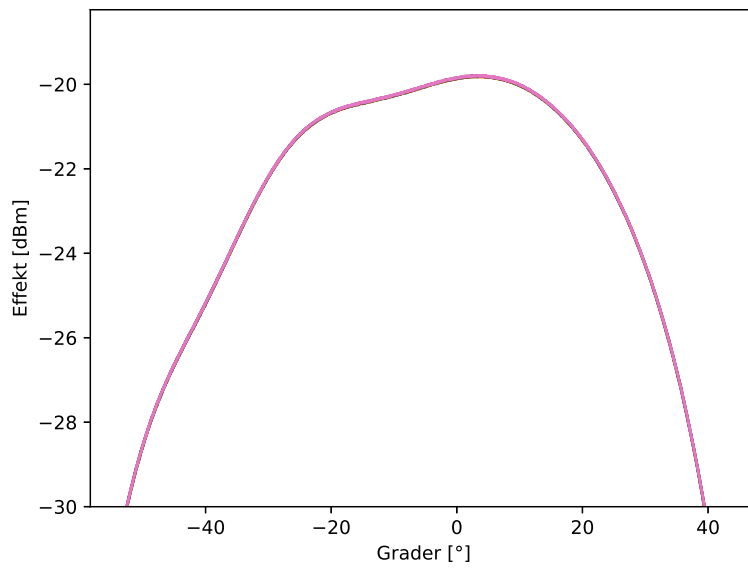
Figur 37 viser målingen gjort i fase ( $\Sigma$ ) i det ekkofrie kammeret.



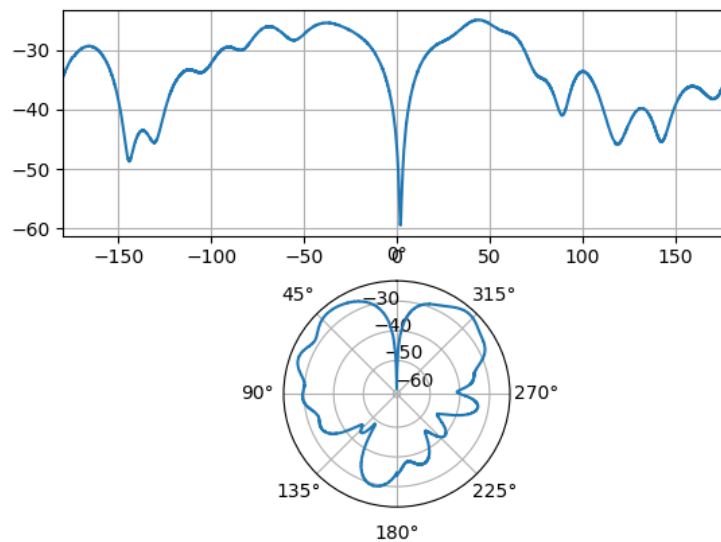
Figur 37: Strålingsdiagram for måling i fase ( $\Sigma$ ) der x-aksen er grader og y-aksen er dBm

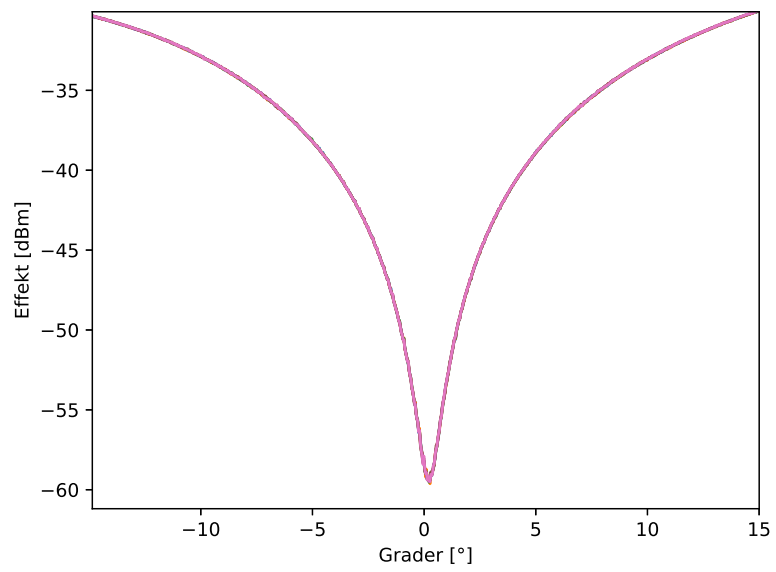
Dataen om vinkel og signalstyrke, fra .h5-filene generert fra antennelaben, ble brukt for å plotte en skalert versjon av strålingsdiagrammet som vist i figur 38. Fra -3dB-punktene i figur 38 vil punktene ligge på  $-32^\circ$  og  $30^\circ$ . Dette utgjør en HPBW på  $62^\circ$ . Forsterkningen kan dermed utregnes ved hjelp av formel (14) med innsatte verdier:

$$G_{TX} = 20 \cdot \log \left( \frac{203}{62^\circ} \right) = 10.3 \text{ dBi} \quad (39)$$

Figur 38: Strålingsdiagram for måling i fase ( $\Sigma$ )

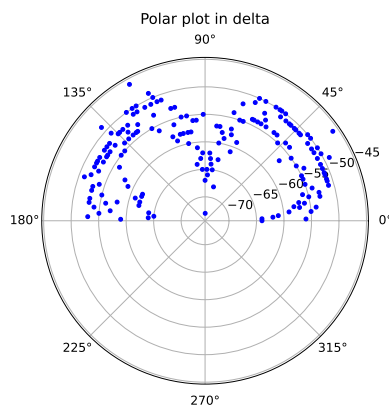
Figur 39 viser resultatet av målinger gjort i motfase og man ser at antennen har et nullpunkt nær  $0^\circ$ . Det teoretiske grunnlaget for denne målingen er beskrevet i 2.2.9 og vil gjengi en approksimasjon på når antennen er orientert mot senderen. Figur 40 viser et forstørret bilde rundt notchen til nullpunktet. Notchen har en forskyvning på  $0.5^\circ$  og en minimum målt effekt på  $-58$  dBm.

Figur 39: Strålingsdiagram for måling i motfase ( $\Delta$ ) der x-aksen er grader og y-aksen er dBm

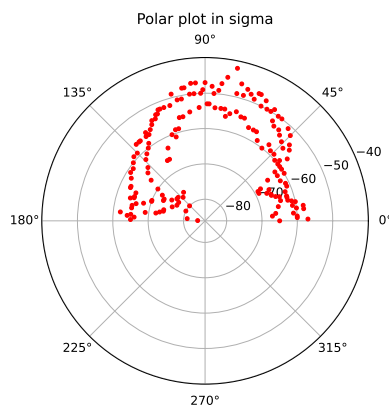


Figur 40: Strålingsdiagram for måling i motfase ( $\Delta$ )

#### 4.4.3 Empirisk testing av antennearrayet



Figur 41: Måling i motfase ( $\Delta$ ) i asimetplanet innendørs ved 5 meter avstand



Figur 42: Måling i fase ( $\Sigma$ ) i asimetplanet innendørs ved 5 meter avstand

Figurene 41 og 42 viser målinger som ble utført ved 5 meter avstand mellom sende- og mottakerantennen, som er operasjonsområdet til systemet (#ANT0304).

#### 4.4.4 Linkbudsjett

Linkbudsjettet vil gjelde for systemet vårt som opererer med frekvensbåndet til BLE. Tabell 10 vil gi en oversikt over de konstantene som gjelder for systemet, der bølgelengden er en funksjon av lysets hastighet og frekvensen.

Parameter	Variabel	Verdi	Enhet
Frekvens	$f$	2400	MHz
Lysets hastighet	$c$	299 792 458	m/s
Bølgelengde	$\lambda$	12.5	cm

Tabell 10: Systemvariabler i linkbudsjettet

Tabell 11 vil gi en oversikt over de passive tapene som oppstår i kabler og kontakter. Disse verdiene er primært hentet fra databladene til komponentene som er opplistet i 3. Innsettingstapene til kablene er hentet fra en tabell som viser standard tap for kabeltypen RG-316 (Times Microwave System 2022).

Parameter	Variabel	Enhet	Verdi
VSWR Koaksiale kabler	$VSWR$	-	1.3
Mismatch loss	$RX_{ml}$	dB	0.3
Innsettingstap (BU-4150029006)	$RX_{il1}$	dB	0.4
Innsettingstap (415-0029-018)	$RX_{il2}$	dB	1.3
Innsettingstap (MXHS83QE3000)	$RX_{il3}$	dB	0.1
Kontaktstap (SMA, svitsj)	$RX_{svitsj}$	dB	1.2
Kontaktstap (SMA, kabler)	$RX_{kabler}$	dB	1.8
Kontaktstap (SMA, kobler)	$RX_{kobler}$	dB	0.4
Kontaktstap (SMA, antenne)	$RX_{antenne}$	dB	0.4
Kontaktstap (SWF, pigtail)	$RX_{pigtail}$	dB	0.1
Totalt tap for passive komponenter	$L_{TX}$	dB	6.0

Tabell 11: Passive tap i linkbudssettet

Ved å sette inn verdier for distanse, frekvens og antenneforsterkning i formelen for FSPL (7) vil det være mulig å beregne tapet som skjer ved propagering i fritt rom. Ved 5 meter vil FSPL være:

$$FSPL = 20 \cdot \log(5.0 \text{ m}) + 20 \cdot \log(2.4 \cdot 10^9 \text{ Hz}) - 147.55 = 54.03 \text{ dB} \quad (40)$$

Flerveisinterferensen ble målt til å være differansen mellom målingene innendørs og utendørs med de respektive signalstyrkene vist i tabell 12. For å gi et estimat på flerveisinterferens ved flere distanser ble det utført målinger ved 1.0 m, 5.0 m og 10.0 m. Disse målingene ble utført i fjernfeltet, som beskrevet i 10.

Parameter	Variabel	Enhet	Verdi
Utendørs måling (1.0 m)	$ute_{1m}$	dB	-37.2
Utendørs måling (5.0 m)	$ute_{5m}$	dB	-45.1
Utendørs måling (10.0 m)	$ute_{10m}$	dB	-54.1
Innendørs måling (1.0 m)	$inne_{1m}$	dB	-39.3
Innendørs måling (5.0 m)	$inne_{5m}$	dB	-52.1
Innendørs måling (10.0 m)	$inne_{10m}$	dB	-63.5
Flerveisinterferens (1.0 m)	$MPI_{1m}$	dB	2.1
Flerveisinterferens (5.0 m)	$MPI_{5m}$	dB	7.0
Flerveisinterferens (10.0 m)	$MPI_{10m}$	dB	9.4

Tabell 12: Flerveisinterferens ved 1.0 m, 5.0 m og 10.0 m

Ved å summere vinningene og tapene i systemet i et linkbudsjett, vil det være mulig å beregne et estimat på styrken til mottakersignalet. Linkbudsjettet består av de passive tapene beskrevet i tabell 11, antenneforsterkningen beskrevet i formel (39), flerveisinterferensen ved hjelp av formel (2.2.5) for 1.0 m, 5.0 m og 10.0 m. Det vil også oppstå et tap på -3 dB, da det blir brukt en lineærpolarisert og en sirkulærpolarisert antenne, som beskrevet i 2.2.1.

Linkbudsjettet brukes for å beregne et estimat på signalstyrken til det mottatte signalet hos mottakerantennen ( $P_{RX}$ ). Denne kan videre brukes til å regne ut linkmarginen, som er differansen mellom den mottatte signalstyrken og støygulvet til mottakerantennen, som beskrevet i 2.2.1.

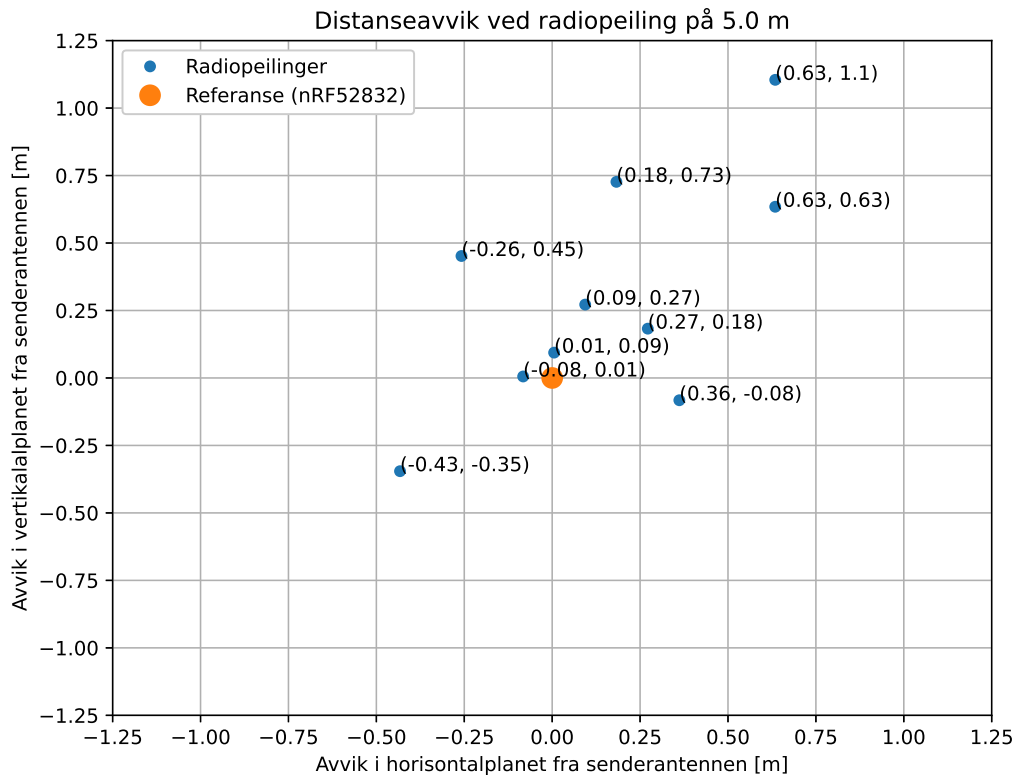
Ved å se på de forventede resultatene i tabell 13 og resultatene fra målingene utført innendørs i tabell 12, vil man kunne se avviket mellom den forventede signalstyrken og den målte signalstyrken. Mottakerantennen har et sensitivitet på -96 dBm, dette impliserer at signalnivået til det mottatte signalet må være av høyere verdi enn sensitiviteten. Tabell 13 viser en forventet signalstyrke ved 1.0 m, 5.0 m, og 10.0 m på henholdsvis -34.01 dBm, -52.49 dBm og -61.07 dBm. Dette vil gi en linkmargin på 61.99 dBm på 1.0 m, 45.51 dBm på 5.0 m og 34.93 dBm på 10.0 m, som er innenfor støygulvet til mottakerantennen. For operasjonsområdet til antennen (5.0 m) vil den forventede effekten bli  $P = 1 \text{ mW} \cdot 10^{\frac{-52.49 \text{ dB}}{10}} = 5.636 \text{ nW}$ .

Parameter	Variabel	Enhet	Verdi
Polariseringstap	$PL$	dB	3.0
Antenneforsterkning senderantenne	$G_{TX}$	dBi	0
Kontakttap sender	$L_{TXCL}$	dB	0.2
Kabeltap sender	$L_{TXIL}$	dB	0.1
Signaleffekt sender	$P_{TX}$	dBm	4.0
Distanser	$d_{1m,5m,10m}$	m	1.0, 5.0, 10.0
FSPL (1 m)	$FSPL_{1m}$	dB	40.0
FSPL (5 m)	$FSPL_{5m}$	dB	54.0
FSPL (10 m)	$FSPL_{10m}$	dB	60.0
Flerveisinterferens (1 m)	$MPI_{1m}$	dB	2.1
Flerveisinterferens (5 m)	$MPI_{5m}$	dB	7.0
Flerveisinterferens (10 m)	$MPI_{10m}$	dB	9.4
Antenneforsterkning mottaker (1 m)	$G_{RX}$	dBi	10.3
Kontakttap mottaker	$L_{RXIL}$	dB	2.1
Koblingstap mottaker	$L_{RXCL}$	dB	3.9
Mottaker-sensitivitet	$RX_{sensitivity}$	dBm	-96
Forventet signalstyrke ved 1.0 m	$P_{RX1m}$	dBm	-37.1
Forventet signalstyrke 5.0 m	$P_{RX5m}$	dBm	-56.0
Forventet signalstyrke 10.0 m	$P_{RX10m}$	dBm	-64.4
Linkmargin ved 1.0 m	$LKM_{1m}$	dBm	58.9
Linkmargin ved 5.0 m	$LKM_{5m}$	dBm	40.0
Linkmargin ved 10.0 m	$LKM_{10m}$	dBm	31.6

Tabell 13: Linkbudsjett for antennesystemet

## 4.5 Radiopeiling

I figur 43 er avviksresultatene for ti radiopeilinger fra 5.0 m avstand illustrert. I midten av plottet er den manuelle peilingen på senderantennen på nRF52-utviklingskortet angitt som referansen. Hvert radiopeilingsresultat i plottet er oppgitt med en målt avstand fra sendeantennen med målestokk. Det er avvikene fra referansen i hver radiopeiling som er representert i plottet.



Figur 43: Plott av distanseavvik ved ti radiopeilinger på 5.0 m

Avstandsmålene fra radiopeilingen i asimut- og elevasjonsplanet betraktes som to uavhengige normalfordelte stokastiske variabler  $X_{asimut, elevasjon} \sim N(\mu, \sigma)$  der forventningsverdiene for avstandsavvik  $\mu_{asimut}$  og  $\mu_{elevasjon}$  kan estimeres fra figur 43, som:

$$\bar{X}_{asimut} = \frac{1}{10} \sum_{i=1}^{10} X_i = 0.140 \text{ m}$$

og

$$\bar{X}_{elevasjon} = \frac{1}{10} \sum_{i=1}^{10} X_i = 0.303 \text{ m}$$

Den ukjente variansen  $\sigma^2$  kan videre estimeres som:

$$\sigma_{asimut}^2 \approx S_{asimut}^2 = \frac{1}{10-1} \sum_{i=1}^{10} (X_i - 0.140)^2 = 0.1222 \text{ m}^2$$

og

$$\sigma_{elevasjon}^2 \approx S_{elevasjon}^2 = \frac{1}{10-1} \sum_{i=1}^{10} (X_i - 0.303)^2 \approx 0.2147 \text{ m}^2$$

Standardavviket  $\sigma$  kan estimeres som:

$$SE(\bar{X}_{asimut}) = \frac{\sqrt{0.1222}}{\sqrt{10}} \approx 0.1105 \text{ m} \quad (41)$$

og

$$SE(\bar{X}_{elevasjon}) = \frac{\sqrt{0.2147}}{\sqrt{10}} \approx 0.1465 \text{ m} \quad (42)$$

Estimatene for standardavvikene kan benyttes i formel (26) for å regne ut et 95 % konfidensintervall for  $\mu$  i begge planene med kvantilet til t-fordelingen  $t_{0.025} = 2.262$  fra tabell E.5 i Løvås (2018):

$$[\bar{X}_{asimut} - t_{0.025} \cdot SE(\bar{X}_{asimut}), \bar{X} + t_{0.025} \cdot SE(\bar{X}_{asimut})] \approx [-0.11, 0.39] \text{ m}$$

og

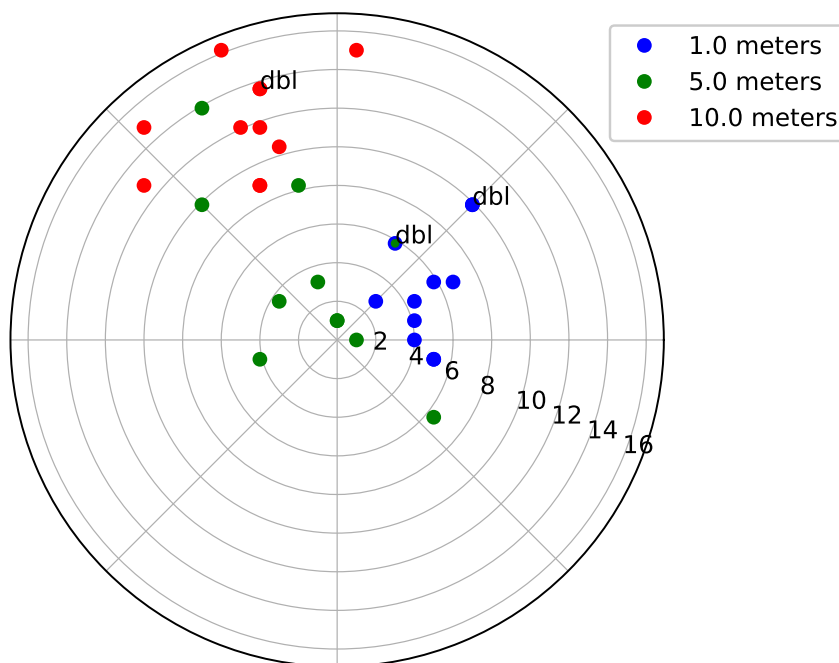
$$[\bar{X}_{elevasjon} - t_{0.025} \cdot SE(\bar{X}_{elevasjon}), \bar{X} + t_{0.025} \cdot SE(\bar{X}_{elevasjon})] \approx [-0.03, 0.63] \text{ m}$$

I figur 44 er resultatene for ti radiopeilinger på hver av de tre peileavstandene plottet. Ett punkt i plottet representerer én radiopeiling. Midtpunktet i plottet er den ukjente søkte verdien (lokasjonen for sendeantennen). Spredningen i punktsvermen rundt sentrum angir usikkerheten i målingene.

For å kunne samle alle de 30 radiopeilingsresultatene i ett plott, ble manuelle peilingsverdier trukket fra sluttvinklene fra enkoderene for hver radiopeiling på de tre radiopeilingsavstandene. Senter i plottet er ulike referanser for de tre radiopeilingsavstandene og ved å trekke manuelle peilingsverdier fra målte radiopeilingsverdier, vil differansen mellom de være den avstanden som et punkt er fra senteret i plottet. På grunn av differansemetoden vil to punkter kunne ligge rett over hverandre selv om radiopeilingene er gjort på ulike avstander.



Polarrepresentasjon av radiopeilinger i grader



Figur 44: Polardiagram for resultatene fra ti radiopeilinger på 1.0 m, 5.0 m og 10.0 m respektivt

## 5 Drøfting

### 5.1 Antennearray

#### 5.1.1 Valg av antenntype

Etter møter med Egil Eide (se vedlegg D), ble det konkludert med at et patch-antennearray var den beste tilnærmingen for systemet, gitt kunnskapsgrunnlaget innad i gruppen og tidsforbruket som var gitt for prosjektet. Antennevalget førte til et produkt som var innenfor antennekravene beskrevet i 3.5.1 og er utvidbar i videre utvikling. Dette kapitlet drøfter de ulike aspektene ved valgene for antenne-løsningen samt resultatene av antennearrayet.

Ved å ta i bruk direktive antenner vil en kunne gjøre retningspeilinger ved hjelp av RSSI-målinger. Det vil være mulig å få en nøyaktig måling, dersom hovedloben er smal nok. Som nevnt i 2.2.9, vil det være mulig å ta i bruk en Yagi-Uda ved bruk av RSSI som søkemetode. Fordelene med denne søkemetoden er at man vil kunne måle signalet for senderen direkte ved hjelp av hovedloben. RSSI vil derfor ha en høyere verdi dersom antennen er orientert mot senderen. For slike antenner vil også søkeloben, dersom antennen har høy forsterkning, være relativt smal, som gjør det mulig å oppnå en nøyaktig måling.

Antennen var en kritisk komponent, da den var nødvendig for BLE-kommunikasjon og de påfølgende arbeidspakkene. Det var derfor av høy prioritet å ferdigstille denne i et tidlig stadium av prosjektet. Tidsforbruket og arbeidspakkene for antennearrayet er beskrevet i vedlegg A. Arbeidsgiver ga uttrykk for at de var ønskelig å ta i bruk en enkel antenneinnretning som en Yagi-Uda-antenne eller en parabolantenne. For valg av antenne ble derfor disse antenntypene prioritert for arbeidspakke 1 (A).

Gitt designet beskrevet i 3.4 var det nødvendig å ta til betraktning parametere som vekt og dimensjoner. Størrelsen på disse parameterne førte til dimensjonale begrensninger for selve antennekonstruksjonen. Ved å bruke et antennearray vil kompleksiteten i systemet øke, og det vil være flere parametere å forholde seg til. Dette problemet ble drøftet med ressurspersoner fra NTNU og SINTEF for å utlede en løsning som tok hensyn til både presisjon og de dimensjonale begrensningene. Antennearray-løsningen resulterte med at antennen måtte designes og konstrueres og det ble lagt til en ekstra arbeidspakke (10) for antennevalget A. Etter interne diskusjoner innad i gruppen var det en felles konsensus om at det var hensiktsmessig å utvide arbeidspakkene med antennekonstruksjon, fremfor å redesigne robotinnretningen.

Svakhetene ved bruk av antennearray og sigma-delta-metoden er primært knyttet til nullpunkt-målingene. Strålingsdiagrammet i fig 39 tilsier at antennen, ved hjelp av faseforskyving, vil måle nullpunktet dersom antennen er orientert direkte mot senderkilden, men også mot reelle nullpunkter der det ikke er signaler som blir sendt. Dette medfører at nullpunktene må verifiseres, i motsetning til direktive antenner som detekterer senderkildene ved hjelp av høyeste RSSI-verdi. Dersom det kun blir utført målinger i motfase ( $\Delta$ ) vil øke sannsynligheten for feilmålinger for nullpunkter

utenom senderantennen og bli påvirket av flerveisinterferens under verifisering av nullpunktet. Dette kan resultere i at antennen mister det ønskede nullpunktet rundt senderen, og antennen vil søke i et område mot et falskt nullpunkt.

Antennen måtte konstrueres på stedet da det ikke var mulighet for å kjøpe ferdiglagde antenner. Å produsere en egen antenneløsning resulterte i et økt timeforbruk på dette arbeidsområdet. De påfølgende arbeidspakkene var avhengig av implementeringen av antennen, og det ble derfor valgt å ta i bruk en 1x2-patch-antennearray, fremfor et 2x2- eller 3x3-array. 1x2-patcharrayet begrunnes ved en sammenligning av tid og kompleksitet av systemene. Dette vil begrense søkeplanene for antennen til å peile i planene sekvensielt, og grunnet dimensjonene måtte det implementeres en ny frihetsgrad som beskrevet i 3.7.2. Det ble også brukt mekaniske enheter, som switcher og koblere, da det ville vært tidskrevende å designe et kretskort med overflatemonterte deler.

De keramiske patch-antennene ble produsert av en ekstern leverandør, og det var derfor en risiko for at det ville være ulikheter mellom patch-ene ved måling av returtap, som beskrevet i 4.4.2. Dersom produksjonen av antenne-patchene skulle bli utført internt, ville dette medført et økt tidsforbruk både i research, simulering og testing for prosjektgruppen. Det kan også oppstå avvik ved målinger dersom antennene ikke blir loddet likt, på grunn av diskontinuitet mellom mikrostriplinjene og matelinjen til antennen. Fordelen ved å ta i bruk en ferdigprodusert patch-antenne begrunnes ved at disse vil gi tilstrekkelige resultater, med mulighet for utvidelser, og at det var tidsbesparende med hensyn på de påfølgende arbeidspakkene. Antennearrayet er ment som et **proof of concept**, da omfanget av antennen var mer omfattende enn først tiltenkt.

Resultatene i figur 39 viser at antennearrayet ikke er symmetrisk rundt  $0^\circ$ . Ulikheten skyldes produksjonsvarianser i antenne-patchene, som vist i figur 33 og figur 34. Det er også et nullpunkt ved  $145^\circ$  som også kan gi utslag for falske nullpunkter. Et falskt nullpunkt kan resultere i at søkealgoritmen antar feil retning i radiopeilingen. Ved å analysere testresultatene til antennene i figur 33 og figur 34, vil differansen mellom returtapene gi en innvirkning på målingene. Differansen skyldes av antenne 1 da denne vil motta signaler bedre enn antenne 2, som vil resultere i usymmetriske målinger i fase- og motfase dersom senderantennen er orientert  $[-180, 0^\circ]$  og  $[0, 180^\circ]$  fra antennearrayet sin bredside.

Det ble brukt sirkulærpolariserte antenner på antennearrayet, noe som førte til et polariseringstap på  $-3.0$  dB. Dersom antennene hadde vært lineærpolarisert og orientert likt for å beholde faseforskyvingen av den elektriske og magnetiske komponenten, ville polariseringstaper vært  $0$  dB. Sumforskjell-metoden, som beskrevet i 2.2.9, bruker summen av signalene målt i fase og motfase, så en lavere amplitude vil derfor ha mindre innvirkning på resultatet. Ved bruk av to lineærpolariserte antenner vil senderantennen måtte roteres  $90^\circ$  ved søk i asimutplanet. Implementasjon av en roterende senderantenne ville vært tidskrevende å implementere, og var derfor nedprioritert til fordel for sirkulærpolariserte antenner. På bakgrunn av de overnevnte grunnene ble det derfor konkludert med at et konstant polariseringstap på  $-3.0$  dB mellom antennene ville ha mindre innvirkning på resultatet, og være en mer tidsbesparende løsning som er enklere å gjennomføre enn en roterende senderantenne, da

de må roteres likt for å beholde polariseringen.

Antennearrayet kan roteres ved hjelp av en servomotor, som beskrevet i 3.7.2, og det vil derfor være mulig å beholde strålmønsteret med det veldefinerte nullpunktet for asimut- og elevasjonsplanet. Antennearrayløsningen er designet med bakgrunn i at det kan utvides til et 2x2-array, siden evalueringskortet som er benyttet har støtte for opp til fire RF-innganger. Dersom et 2x2-array skal anvendes må det lages et nytt PCB-utlegg med SMA-kontakter for fire patch-antennene og en ny festeanordning for arrayet til robotarmen. Å benytte et 2x2-array gjør det mulig å radiopile i to plan samtidig ved å bytte mellom måling av signalene i radene og kolonnene i antennearrayet, avhengig av søk i elevasjon- eller asimutplanet. For disse løsningene vil antennene bli summert i fase avhengig av orienteringen på antennearrayet. 2x2-løsningen krever bruk av flere hybrid-koblere, siden flere operasjoner på signalene må gjøres sammenlignet med 1x2-arrayet. Ved bruk av disse måleteknikkene, vil det derfor være hensiktsmessig å designe egne utlegg med svitsjer, koblere og kretser for faseforskyvning av signalene. Ved bruk av et 3x3 array vil man lokalisere bevegelige Bluetooth-enheter ved å måle sumforskjellen mellom ulike soner av antennearrayet for å differensiere retningen senderen beveges. Det vil derfor være en antenne av typen fase-arrangert antenne, som kan brukes for å styre strålmønsteret fra en stasjonær tilstand (Blake 1971).

### 5.1.2 PCB-utlegget

Å konstruere en antenne er tidskrevende arbeid som krever spesialkompetanse innenfor antennteknikk. Det ble derfor utført research gjennom digitale ressurser for å utarbeide et design. For å unngå aliasing var det nødvendig at antennene ble plassert med en avstand på  $d = \frac{\lambda}{2}$ . Førsteutkastet ble designet med feil avstand, etter en mistolkning av den dielektriske innvirkningen på kretskortet. Utlegget ble derfor designet med en ny avstand mellom festepunktet som ga et ekstra tidsforbruk for arbeidspakke 10 (A).

Etter samtalen med Egil Eide D, ble det drøftet de ulike fremgangsmåtene for søkemetoden for radiopiling. Det ble foreslått et elektronisk kretskort med svitsjer og hybridkoblere, som styres digitalt ved hjelp av GPIO-pinner. Antennen som ble konstruert var betinget av dimensjonene til antennen, og det var derfor nødvendig å diskutere internt i prosjektgruppen de ulike alternativene med hensyn på tid og ressurser. Det var derfor hensiktsmessig å velge alternativet med bruk av en hybridkobler og et evalueringskort for å forenkle utlegget som skulle designes. Det var derfor kun nødvendig å designe et PCB-utlegg med hensikt om å overføre signalet fra hver patch-antenne til de sidemonterte SMA-kontaktene på evalueringskortet. Bruken av hybrid-kobleren gjorde designprosessen mer effektiv og mindre tidskrevende, noe som var hensiktsmessig med hensyn på tidsplanen for prosjektet.

Designet av utlegget ble drøftet med ressurspersoner fra NTNU, og deretter implementert i KiCad 4. Resultatene og skjematetegninger i figur 32 og figur 17 viser det ferdigstilte antennearrayet. Dette ble fremstilt ved NTNU av Overingeniør Terje Mathiesen. Da det ikke var behov for et effektplan på utlegget, ble det valgt et 2-lags

kretskort, som gjorde utskjæringen og rutingen enklere ved produksjon av kortet. Det ble utregnet en tykkelse på overføringslinjene med hensyn på impedansen i formel 5. Siden det ikke ble oppgitt en eksakt verdi for den dielektriske konstanten  $\epsilon_r$ , ble det gjort en antagelse basert på standardverdier for 2-lags kretskort med FR4-substrat Peterson 2021.

Kravspesifikasjonene for antennen er beskrevet i tabell 6 med prefiks ANT02XX. De endelige dimensjonene til antennearrayet ble 100 mm x 50 mm x 1.6 mm, som oppfylte kravspesifikasjon #ANT0201. Det ble tatt i bruk patch-antennene beskrevet i 16, der avstanden mellom matelinjene for matepunktene ble 6.25 mm, som er innenfor kravspesifikasjon #ANT0202 og #ANT0203.

Patch-antennene opererer på frekvensbåndet 2.4 Ghz til 2.5 Ghz, er sirkulærpolarisert og har en nominell impedans på 50  $\Omega$ . Dette er innenfor #ANT0101, #ANT0102 og #ANT0103 i de tekniske kravspesifikasjonene for antennen i tabell 5. Det ble brukt SMA-kontakter som nevnt i 3 som oppfyller krav #ANT0204.

### 5.1.3 Strålingsdiagram og antennekarakteristikk

Ved å analysere returtapet i figur 33 og figur 34, observeres det at returtapene for antennene er ulike. Dersom komponentene hadde vært ideelle, ville returtapet for antennene vært like og antennen hadde mottatt og utstrålt signalet symmetrisk rundt 0°, som kunne gi mer presise resultater. Ulikheten mellom antennene vil føre til at ved summering av signalene i fase og motfase kan produsere uønskede karakteristikk, som flere nullpunkter og sideløber.

Strålingsdiagrammene som ble produsert under antennelaben var i stor grad som forventet. Målingene i fase og motfase viser til en antennekarakteristikk som var innenfor kravspesifikasjonene, men hadde noen ufullkommenheter som kan gi diffuse målinger. Resultatet i figur 39 viser et punkt nær 135° med lav RSSI-verdi. Falske nullpunkter er ugunstige for målingene, siden søkemetoden vil kunne detektere falske nullpunkter ved denne vinkelen. Siden antenne 2 har et høyere returtap ved den gitte frekvensen, vil dette medføre at det mottatte signalet vil være høyere i antenne 2 til sammenligning med antenne 1. Dette resulterer i at for måling av signaler i motfase, der antennen er orientert mellom 0°-180°, vil signalene innsatt i formel (20) gi forskjellige verdier fra målinger utført ved 0° og -180°.

Ved å sammenligne målinger utført i et ekkofritt kammer i figur 37 og figur 39 med de empiriske målingene under ikke-ideelle forhold i figur 42 og figur 41, ser man at resultatene blir tydelig påvirket støy og interferens. Dette er forventet, da de empiriske målingene ble utført innendørs. Målingene under ikke-ideelle forhold viser at antennen beholder nullpunktet for målinger i motfase og hovedloben ved måling i fase, som er ønskelig for peilemetoden.

### 5.1.4 Linkbudsjett

Linkbudsjettet i tabell 13 viser vinningene og tapene i systemet, der den forventede signalstyrken ( $P_{TX}$ ) og linkmarginen ble estimert. Senderantennen som er blitt brukt er en antenne som er integrert på nRF52832-utviklingskortet, som beskrevet i 3.9.1. Det var derfor behov for å lese av dokumentasjonen til senderantennen for å kartlegge tapene som oppstår hos senderantennen. Det ble oppdaget at det ikke var tilgjengelig dokumentasjon for systemet fra produsenten, som medførte at kontakt- og kabeltap ble estimert i linkbudsjettet. Ved å gjøre antagelser av tap i systemet, vil dette medføre en usikkerhet i linkbudsjettet. Disse tapene ble estimert på bakgrunn av kontakttap fra mottakerantennen, og diskontinuitet i transmisjonslinjen til den integrerte antennen.

Det ble utført målinger utendørs, som ble brukt som referanse til målingene som ble gjort innendørs. Utendørsmålinger sammenlignet med inndendørsmålingene vil gi et estimat for demping og forsterkningen av signalet for rommet det ble utført målinger i innendørs. Siden målingene ikke ble utført under ideelle forhold, vil det være et avvik for målingene av flerveisinterferensen, da det vil være uunngåelig å motta reflekterte signaler under ikke-ideelle testforhold. Utrekningene av flerveisinterferensen, vist i tabell 12 vil derfor ha et avvik, siden signalene også vil reflekteres i omgivelsene systemet ble testet i. Ved å utføre målinger i et ekkofritt kammer, vil flerveisinterferensen som oppstår være neglisjerbar og vil få et mer nøyaktig estimat av flerveisinterferensen gjennom empirisk testing. Ved å ta i bruk integrerte kretser på PCB-utlegget, vil det være mulig å redusere de statiske tapene som oppstår i kontaktene og ledningene i kobleren og switchen. Å samle flere av komponentene i systemet på PCB-utlegget ville ført til at linkmarginen til systemet hadde vært større, og gitt en bedre forventet signalstyrke ( $P_{TX}$ ). De passive tapene som oppstår i sluttproduktet vil være uunngåelige, siden systemet er avhengig av de valgte komponentene for å kunne utføre en radiopeiling.

## 5.2 Integrerte systemer

### 5.2.1 Valg og styring av motor

Det var viktig at motoren som ble brukt som aktuator i robotinnretningen hadde god repeterbarhet og oppløselighet. Det ble bestemt at servomotorer skulle benyttes. Dette er fordi servomotorene er kraftige, presise og relativt billige, samtidig som de har grei oppløselighet. Steppermotor kunne også vært aktuelt, men på grunn av størrelse, mangel på holdemoment og nøyaktighet ble det konkludert med at det ikke var gunstig med denne typen motor (se 2.3.1). Det viste seg at oppløseligheten til servomotorene som ble valgt ikke var like bra som ønsket, som førte til at det ble begrenset hvor små intervaller det var mulig å flytte antennen. Det ble observert at ved minst mulig påtrykning forflyttet servoen seg i intervaller på mellom én til to grader. Over lengre distanser utgjorde dette store avstander, som førte til problemer for nøyaktigheten av målingene rundt disse avstandene. Ved å investere i servomotorer av høyere kvalitet kan oppløseligheten trolig bli bedre, men dette krevde at delene

til robotarmen ble re-designet, og dessuten var det nødvendig med større budsjett. En annen løsning kan være å benytte giring for å oppnå bedre oppløselighet, da oppløseligheten vil øke proporsjonalt med giringsfaktoren. Konsekvensen av dette er at arbeidsområdet faller omvendt-proporsjonal med samme faktor. Servomotoren som blir brukt har et arbeidsområde på  $270^\circ$ . Fordi det kun blir benyttet  $180^\circ$  i prosjektet kunne det blitt implementert giring på bekostning av de 90 ubrukte gradene, men på grunn av mangel av tid ble ikke dette gjort.

Under utvikling av robotinnretningen var det lenge antatt at det kom til å bli benyttet en betydelig lenger antenne med større masse enn det som sluttproduktet endte med. Endringen av antennetyper resulterte i en antenne med masse på ca. 50 gram og en neglisjerbar massesenteravstand på PCB-kortet da det bare er 1.6 mm dypt i den monteringsretningen som er gjort. I tillegg oppsto det et behov for å rotere antennen mellom  $0^\circ$  og  $90^\circ$ , så en ekstra servomotor måtte bestilles. På grunn av det nye antennedesignet og behovet for rotasjon fikk armen fra elevasjonservoer endret utforming. For å verifisere at dreiemomentene på servomotorene ikke overskrider grenseverdien på 1.67 Nm, kan det regnes ut et nytt dreiemoment der massesenter for den roterende braketten på elevasjonservoer, festet for antennerotasjonservoer, antennerotasjonservoer, festet for antennen og antennen er tatt i betraktning sammen med avstandene fra elevasjonservoers roterende akse. Men, siden både den totale massen for PCB-antenneløsningen og armlengden begge ligger langt unna de opprinnelige kravene på maksimalt 0.6 kg og 0.20 m, er dette strengt tatt ikke nødvendig. Resultatet er altså at servomotorene for rotasjon i asimut- og elevasjonsvinkelen er overdimensjonerte.

Servomotoren styres ved å påføre et PWM-signal med varierende arbeidssyklus (se teori fra 2.3.5). nRF Connect SDK har eget bibliotek for generering av PWM-signaler, men det ble valgt å utvikle egen programkode fremfor å bruke det som finnes i SDK-en. Å utvikle egen programkode ble gjort på bakgrunn av at PWM-driveren fra nRF Connect SDK behøver å ta inn både periode og aktiv tid for hver gang det skrives en vinkel til en servomotor. I tillegg må vinklene forhåndsdefineres. En slik fremgangsmåte er dårlig egnet for et sanntidssystem som ble utviklet i dette prosjektet. Det er dårlig egnet fordi når systemet kjører er det ikke bestemt hvilken vinkel som skal skrives, heller ikke hvor lenge det er nødvendig å holde vinkelen. Det er ønskelig å skrive en vikårlig vinkel til servomotor, og at denne vinkelen holdes til annet blir bestemt. Denne funksjonaliteten er implementert med driveren som ble utviklet.

### 5.2.2 Valg av enkoder

Ideelt sett ville absolutt enkoder vært det beste alternativet for robotinnretningen. Dette ville gitt muligheten til å alltid vite nøyaktig posisjon til endeffektor for å oppnå best mulig presisjon, uten behov for kalibrering eller re-kalibrering. Inkrementelle enkodere er også svært nøyaktige, men tellefeil kan oppstå om aktuator roterer over maksimal hastighet for enkoder, eller om MCU ikke evner å plukke opp alle pulser fra enkoder, for eksempel om den er opptatt med andre prosesser når enkoder sender pulser. Dette kan akkumulere feil til en grad at roboten blir unøyaktig.

Fordelene mot ulempene ble veid opp mot hverandre, og på grunn av den betydelige større kostnaden for absolutte enkodere med tilsvarende oppløselighet, ble konklusjonen å gå for inkrementelle enkodere.

Enkoderene som ble benyttet i prosjektet har en relativt høy oppløselighet på 2048 PPR per kanal. Under utvikling ble det observert at oppløseligheten var til den grad at det var upraktisk for anvendelsesområdet, siden presisjonen til servomotorene ble observert langt lavere enn enkoderene. Det ville vært hensiktsmessig med en lavere oppløselighet på enkoderene, siden hver puls på de to enkoderkanalene utløser en ISR, som kan føre til problemer da utføring mange av ISR tar opp mye ressurser i prosessoren.

### 5.2.3 Bruk av knapp eller hall-effect-sensor

For å rotere antennen ble det brukt en mikroservomotor. Denne viste seg å være for svak, som gjorde at antennen til tider ikke er helt i korrekt posisjon. I tillegg viste det seg at taktilbryterene som skulle anvendes for rotasjonsdetektering av antennen hadde for høy aktiveringskraft for SER0049-servomotoren, slik at de ikke pålitelig kunne holdes inne under radiopeilingen. Taktilbryterene ble derfor fjernet fra baksiden av antennearrayfestet. Resultatet av å ikke ha pålitelig rotasjonsdetektering av arrayet er at utviklingskortet har ingen måte avgjøre om alle radiopeilingsmålinger er pålitelige, og avvik fra rent horisontalt og vertikalt antennearray kan innføre feilaktige signalstyrkemålinger i søkealgoritmen. Det er mulig at et bedre valg for rotasjonsdetekteringen til servomotoren ville vært å bruke hall-effect-sensorer. Hall-effect-sensoren er ikke avhengig av mekaniske påtrykk for å gjøre en tilstandsending og virker i stedet ved å detektere magnetiske felt. På denne måten er det ikke et behov for servomotoren å trykke inn en knapp, men heller å rotere en magnet nært inntil deteksjonsfeltet til hall-effect-sensoren. Den ekstra belastningen på servomotoren forsvinner og kan trolig holde den ønskede posisjonen bedre enn ved å bruke taktilbrytere.

Antennearrayet holder heller ikke alltid riktig vinkel under radiopeiling på grunn av de litt korte koaksialkablene som ble brukt mellom arrayet og hybrid-kobleren. Servomotoren har et avviksområde på omtrent 5-7 ° rundt ønsket posisjon som den tolker som ønsket posisjon. Dersom antennearrayet blir rotert over avviksområdet vil servomotoren selv rette opp arrayet til riktig vinkel. For å forbedre rotasjonsproblemet kunne det blitt brukt en kraftigere servomotor, lengre koaksialkabler eller flytting av hybrid-kobleren i kontrollboksen. De to sistnevnte løsningene hadde trolig hjulpet på problemet ved å ikke trekke så veldig i koaksialkablene under radiopeiling, men servomotoren er fremdeles sensitiv for mekaniske påvirkninger utenfra. En annen løsning kunne vært å ha en fysisk sperre mellom antennefestet og servomotoren slik at antennen ikke kan rotere utenfor de to vinklene på 0° og 90°. Ulempen med denne metoden er at servomotoren kan komme til å prøve å nå en rotasjonsvinkel som er mindre enn 0° eller større enn 90°, som vil si den prøver å nå en vinkel den fysisk ikke kan flyttes til. Da vil servomotoren stå med fast påtrykk spenning og trekke unødvendig mye energi og potensielt bli skadet på grunn av varmeoppbygging.



## 5.3 Algoritme og programmering

### 5.3.1 Bruk av C, NRF desktop, Git og headerfiler

På grunn av valget av mikrokontroller (3.7.1) landet valg av programmeringsspråk på C og applikasjonen ble kodet deretter. Som nevnt i 3.8 ble det brukt eksternt utviklede biblioteker for å forenkle programmeringsprosessen. Alternativt kunne man ha programmert applikasjonen uten disse hjelpemidlene, men dette ville hadde økt både arbeidsmengden og tidsforbruket.

I vedlegg I kan man se at applikasjonen er delt opp i forskjellige kildefiler, hvor hver kildefil inneholder kode med relatert funksjonalitet. Dette gjør at applikasjonen vil yte fordelaktig sammenlignet en applikasjon som er blitt kodet i en enkel kildefil.

### 5.3.2 Begrensninger

Under utvikling av applikasjonen er det blitt tatt hensyn til begrensningene som følger med utstyret som er blitt tatt i bruk, både fysisk og programmessig. Som følge av at mikrokontrolleren har begrenset med lagringsplass (3.7.1), ble det under utviklingen av applikasjonen tatt i bruk ulike programmeringsprinsipper for å ta hensyn til minnebruken. Ved spesifisering av størrelsen og område på variablene som ble definert bruker man ikke mer minne enn nødvendig (2.4.1). Array som inneholder gjeldende målinger ble tilbakestilt etter hver søkeprosedyre og dermed klar til gjenbruk for neste søkeprosedyre. Funksjonene ble utviklet til å være dynamiske, hvor man ved bruk av inngangsparametere spesifiserer hvilke handlinger applikasjonen skal utføre. Implementering av disse prinsippene er ikke strengt nødvendig ved utvikling av en applikasjon, men vil spare applikasjonen for bruk av minne. Dette er kanskje ikke avgjørende for en mindre applikasjon, men for større og mer kompliserte applikasjoner er dette essensielt.

I 3.8.5 ble det nevnt at applikasjonen tar i bruk *call by reference* for returnering av et array. En alternativ løsning hadde vært å ta i bruk globale variabler sammen med *call by copy*. En slik tilnærming vil øke faren for *stack overflow* og det ble dermed bare brukt under omstendigheter hvor funksjonene utfører enkle handlinger. Utenom prosessering av store array-er, er bruken av rekursjon en av de vanligste årsakene til *stack overflow*. Siden rekursjon gir mulighet for å gjenbruke koden man har utviklet, vil den totale minnebruken reduseres så lenge funksjonen er gjennomtenkt og inneholder betinget argument som returner funksjonen.

### 5.3.3 Valg av søkealgoritme

Ved utvikling av søkealgoritmen fantes det flere tilnærminger, der en av dem var et fullstendig søk i enten det horisontale planet eller det vertikale planet før man går over på det andre planet. En slik algoritme vil ikke være optimal da hvert av planene vil kun bli søkt i en fast sektor og gjennomført søkesektor vil være

begrenset. Robotarmen vil dermed ikke ha muligheten til å korrigere seg selv hvis den bommer på senderantennen. Det kunne også vært mulig å ha foretatt et søk på hele arbeidsområdet til robotarmen, men dette ville ha vært en tidkrevende prosess og ble ikke implementert av den grunn. Valget av søkealgoritme falt deretter på noe imellom de to (se 3.8.4). Ved å veksle mellom hvilket plan robotarmen søker i vil søkesektoren øke og dermed også muligheten til å lokalisere senderantennen.

For å redusere påvirkningen av støy forårsaket av flerveisinterferens (se 2.2.5), ujevne bevegelser fra motorene eller generelt støy, ble mengden målinger under finsøket for hvert målepunkt økt. Økningen av målinger vil også øke tiden systemet bruker på å peile etter sendeantennen, men også øke muligheten for å lokalisere den. For å begrense søketiden til finsøket, vil finsøket kun søke i nærhet av retningen til senderantennen og ikke hele arbeidsområdet til robotarmen.

På grunn av begrensningene som BLE har når det gjelder sendefrekvensen for informasjon (3.9.1), kan ikke applikasjonen selv bestemme når den skal hente signalstyrken til senderantennen. Dermed fungerer den mottakende delen av applikasjonen etter prinsippet *behandlere* (se 3.8.2). En av utfordringene som følger ved å bruke en *handler* er å forsikre at målingene som er blitt tatt samsvarer med posisjonen til robotarmen. Som nevnt i 3.8.4 ble det tatt i bruk binære semaforer for å oppnå prosess synkronisering. Ved å ekskludere bruken av dette hjelpemiddelet kan man ikke bekrefte at målingene som er blitt utført er gyldige. Men, som nevnt i 2.4.7 er ikke bruken av semaforer uten risiko.

### 5.3.4 Utvikling av algoritmer

Som nevnt i 3.8.5 ble det utviklet egne algoritmer for å behandle radiopeilingsmålingene som ble foretatt. Algoritmene er utviklet for å være så dynamiske som mulig og med mulighet for å videreutvikle produktet. Avgjørelsen av hvilke algoritmer som var nødvendige ble basert på erfaring innen koding og behovet for å løse problemstillingene som oppstod underveis. Algoritmen som validerer rådataene fra søket vil filtrere ut dårlige målinger som kan oppstå på grunn av flerveisinterferens, bitfeil eller ved feil allokert minne. Ved å implementere en algoritme som kalkulerer en grenseverdi for signalstyrke i fase ( $\Sigma$ ), ender applikasjonen med å være dynamisk. Etttersom at det ikke er gitt at signalstyrken vil være lik når man forandrer arbeidsmiljøet robotarmen operer i, da påvirkningen fra flerveisinterferensen vil kunne variere (se 2.2.5).

Videre for robotarmen kunne det ha blitt utviklet en algoritme for å sortere de avleste målepunktene. En slik algoritme ville ha redusert den totale tiden brukt til å behandle måledataen.

## 5.4 Bluetooth

Bluetooth Low Energy ble valgt som nettverkprotokoll siden oppgavebeskrivelsen nevner at det skal brukes en nRF System on Chip for styring og prosessering av data (NTNU 2021). nRF SoC-ene støtter bare NFC, Bluetooth Mesh og BLE (Nordic Semiconductor 2021d), der BLE var mest hensiktsmessig å anvende i radiopeilingsystemet. Utviklingsverktøyene som tilhører SoC-ene inneholder biblioteker for bruk av BLE beskrevet i 2.3.2, som var gunstig med hensyn på tidsrammen som var satt til å etablere ende-til-ende-kommunikasjon i prosjektet.

Tilkoblingsfri kommunikasjon var ønskelig siden det ble brukt simpleks-kommunikasjon mellom enhetene, som nevnt i 2.5.4. tillater “Broadcasting” utvidelse av systemet med flere senderkilder, siden enhetene ikke trenger å pares. Arkitekturen til systemet ble derfor designet med hensyn på simpleks-kommunikasjonen, fremfor vedvarende tilkoblinger ved hjelp av “connection”-metoden beskrevet i 2.5.4. Connection-metoden er effektiv for kommunikasjon over lengre tidsintervaller, siden denne modusen støtter vedvarende tilkoblinger. Fokuset ligger derfor på energisparing, siden tidsintervallene kommunikasjon skal skje over en kortere tidsramme.

Mikrokontrolleren i systemet har en begrensning på hvor raskt den kan prosessere data. BLE var derfor hensiktsmessig å bruke med hensyn på energieffektivisering og tidsintervallene mellom dataoverføringene. Ved hyppigere overføring av data vil mikrokontrolleren ha problemer med å prosessere data raskt nok, som vil føre til unødvendig energibruk i senderantennen.

## 5.5 Radiopeiling

For å kunne antyde til et svar på hvor nøyaktig lokasjonen til en stasjonær radiosender på 2.4 GHz kan finnes gjennom å monitorere den mottatte signalstyrken og finne ut av hvilken retning signalstyrken er sterkest, må resultatene av radiopeilingsresultatene i 4.5 vurderes. Med utgangspunkt i resultatene for ti radiopeilinger fra en avstand på 5.0 m, kan resultatene benyttes som underlag for nøyaktighetsvurderingene målt i avstandsavvik fra sendeantennen for systemet. Fra resultatene kommer det frem at 95 % konfidensintervallet for forventningsverdiene for avstandsavvikene i asimut- og elevasjonsplanet er henholdsvis:

$$[-0.11, 0.39] \text{ m}$$

og

$$[-0.03, 0.63] \text{ m}$$

Konfidensintervallene er fremstilt fra betraktningen om at målevariablene  $X_{asimut}$  og  $X_{elevasjon}$  er uavhengige stokastiske variabler, med sannsynlighetsfordelingen som en t-fordeling som nærmer seg en normalfordeling ved et stort antall målinger ( $> 30$  målinger). Siden det ikke er fremstilt et stort antall målinger ( $10 < 30$ ) for peileavstanden på 5.0 m og det er benyttet en t-fordeling, der det er mer sannsynlig å observere store verdier. Det vil altså si at dersom det hadde blitt gjort flere radiopeilinger ville både konfidensintervallet og den estimerte forventningsverdien vært nærmere

de virkelige og ukjente verdiene. Det kan likevel sies at konfidensintervallene som er fremstilt er et mål på hvor gode estimatene av de ukjente forventningsverdiene er i akkurat det testmiljøet som ble benyttet ved datasamlingen.

I figur 44 kan resultatene for de ti radiopeilingene fra hver avstand brukes som vurderingsunderlag for presisjonen og nøyaktigheten til systemet. For alle punktsvermene ser det ut til at presisjonen er dårlig, siden det er spredning mellom alle punktene i hver punktsverm. Den røde punktsvermen har i tillegg til dårlig presisjon også dårlig nøyaktighet, siden alle punktene ligger et stykke fra diagramsenteret. Den grønne punktsvermen ligger nærmere diagramsenteret, men har også enkeltmålinger som ligger langt borte. Nøyaktigheten ser ut til å være best for den blå punktsvermen, siden samlingen av punktene ligger nærmest sentrum av de tre punktsvermene. Nøyaktigheten forventes å bli bedre til mindre avstand det er mellom sende- og mottaksantennen på grunn av rotasjonsoppløsningen i servomotorene. I forprosjektet ble den teoretiske maksimale nøyaktigheten funnet til å være omtrent 0.18 m fra en peileavstand på 10.0 m ved en rotasjonsoppløsning på  $0.9^\circ$ . Den samme utregningen som ble gjort i forprosjektet kan anvendes på peileavstandene 5.0 m og 1.0 m med en antatt rotasjonsoppløsning på  $1^\circ$ , og er henholdsvis på omtrent 0.09 m og 0.02 m. De teoretiske nøyaktighetene støtter under påstanden om at en kortere peileavstand vil gi bedre nøyaktighet dersom rotasjonsoppløsningen kan antas å være omtrent  $1^\circ$ .

Nøyaktigheten i systemet preges også av FSPL for de ulike peileavstandene. FSPL vil øke kvadratisk med avstanden, som vil påvirke signalstyrken for målingene. Det kan derfor være en utfordring å skille mellom nullpunktene, da signalnivået generelt vil være lavere. Ved å øke avstanden mellom senderantennen og radiopeileren vil det også være flere objekter som resulterer i flerveisinterferens. Flerveisinterferens i de første fresnelsonene vil interferere med signalet og vil kunne gi feilmålinger. Ved å ha et område større enn den første fresnelsonen fra siktlinjen, vil sannsynligheten for å få destruktive og konstruktive signaler mot peileren reduseres, da signalstyrken til de reflekterte signalene fra de ytre fresnelsonene vil interferere mindre på grunn av FSPL.

Presisjonen til systemet kunne muligens blitt forbedret dersom målesystemet ble undersøkt og videre justert eller korrigert for systematiske feil i måleverdiene, slik som monteringsavviket for laserpekeren på antennearray-et. En kjent instrumental feil er at laserpekeren ikke er sentrert i nullpunktssenteret på antennearray-et på robotarmen. På grunn av robotsammensetningen er det ikke mulig å montere laserpekeren i midten av antennearray-et, så den er montert 4.0 cm over midtpunktet på arrayet. Den kjente instrumentale feilen bidrar altså med et avlesningsavvik på radiopeilingsresultatene for elevasjonsverdiene. Dersom det justeres for avlesningsavvikene på resultatene i elevasjonsplanet, vil forventningsverdien for avstandsavviket i elevasjonsplanet bli  $\bar{X}_{elevasjon} = 0.263$  og 95 % konfidensintervallet for forventningsverdien kunne skrives:  $[-0.04, 0.57]$  m.

Servomotorene som ble benyttet i systemet viste seg å ha en lavere oppløsning enn ønsket. Erfaringsmessig viste det seg at SER0038-servomotorene har en oppløsning som ligger over  $1^\circ$ , siden et PWM-påtrykk som i teorien skulle tilsvare  $1^\circ$  vinkelendring i noen tilfeller ikke førte til en endring på servoene. I andre tilfeller kunne servomotorene hoppe  $2^\circ$  i gangen. Den lave oppløsningen for servomotorene kan bi-

dra til å gjøre systemet mer unøyaktig, siden søkealgoritmen baserer seg på å hente inn RSSI-verdier fra antennearrayet med en oppløsning på  $1^\circ$ . Dersom servomotoren ikke roterer når den skal, vil søkealgoritmen hente inn RSSI-avlesninger fra feilaktige vinkleposisjoner, men rette seg opp ved hjelp av enkoderavlesning til neste måling. SER0049-servomotoren kan også bidra til unøyaktige RSSI-målinger siden denne servomotoren ikke alltid klarte å holde antennearrayet i riktig vinkelposisjon ved søking i asimut- og elevasjonsplanet. Dersom antennearrayet ikke er rotert enten helt horisontalt eller helt vertikalt under hele søket i asimut- eller elevasjonsplanet, vil fasen mellom antennene bli forskjøvet og vil føre til en feilmåling avhengig av forskyvningen mellom antenneelementene. En mulig forbedring i prosjektet kan da være å benytte seg av steppermotorer med veldefinerte diskrete trinn og en servomotor som klarer å holde antennearrayet bestemte posisjoner under hele radiopeilingen.

Strålingsdiagrammet for antennen ved målinger i motfase, som vist i figur 39, viser forskjellen i signalnivåene rundt  $0^\circ$ . Ved empirisk testing ble det kjent at signalnivået var likt for målinger  $\pm 2^\circ$  fra nullpunktet, og vil derfor begrense nøyaktigheten til systemet for måling av nullpunktene. Målingene i figur 40 viser at nullpunktet er forskjøvet  $0.5^\circ$ , som beskrevet i 4.4.2. Denne forskyvningen av nullpunktet vil også forskyve målepunktene i radiopeilingene. Målingene i fase, som vist i figur 37, viser en hovedlobe med en HPBW på  $62^\circ$ . Nøyaktigheten for verifisering av nullpunktene vil derfor være på størrelse med HPBW-en, og det vil derfor være en sannsynlighet for at den kan få en høy signalstyrke fra kilder for målinger i fase utenom det målte nullpunktet målt i motfase. Nøyaktigheten vist i figur 43 vil derfor betinges av alle de individuelle komponentene i radiopeilingssystemet, da usikkerhetene i komponentene er uavhengige fra hverandre. En mulig forbedring i systemet kunne ha vært å hatt et antennearray med presist plassert nullpunkt og en smalere HPBW.

Testtriggen for målingene, beskrevet i 3.6.3, gjengir omgivelsene for den empiriske testingen. Figur 18 viser at den vertikale avstanden mellom gulvet og radiopeileren er kortere enn den horisontale avstanden mellom veggene og innretningen. Dette kan føre til at en større mengde elektromagnetiske bølger blir reflektert til senderen og vil gi diffuse målinger i vertikalplanet grunnet flerveisinterferens. Flerveisinterferens og FSPL beskrevet i 2.2.5 og 2.2.3, respektivt, vil påvirke måleresultatene for et gitt testmiljø. Unøyaktigheten vil derfor variere avhengig av testomgivelsene, og det presiseres derfor at forventningsverdier og konfidensintervallene for avstandsavvikene bare gjelder for det gitte rommet.

Målingene ble utført i fjernfeltet til senderantennen. Dette er for å unngå diffuse målinger, grunnet uavhengigheten mellom de elektriske- og magnetiske komponentene, som beskrevet i 2.2.4. Ved målinger i nærfeltet vil derfor målingene bli upålitelige og inkonsekvente målinger, da det elektromagnetiske feltet fluktuerer uforutsigbart.

## 6 Konklusjon

I dette prosjektet er det blitt designet og konstruert et system for radiopeiling for 2.4 GHz. Systemet består av et 1x2-antennearray påmontert en robotarm med tre frihetsgrader. Robotarmen ble designet i Fusion 360 og dimensjonert etter et utledet forslag for antennetype. Antennearrayet er blitt designet, konstruert og karakterisert gjennom antennelab og empirisk testing. Det er blitt programmert en algoritme for søk- og databehandling av målt RSSI-verdi fra antennearrayet i C, som gjennom en nRF52832 SoC brukes for å styre robotarmen og behandle signalene fra antennen.

Svaret på problemstillingen om hvor nøyaktig man kan finne lokasjonen til en stasjonær radiosender på 2.4 GHz, kan ikke gis som et definitivt svar. Det viste seg i prosjektet at faktorer som dårlig oppløsning på servomotorene, varianser i antennepatcher og valgt testmiljø alle påvirker nøyaktigheten til systemet. For det gitte testmiljøet på 5.0 m radiopeilingsavstand kan det likevel gis estimat for forventningsverdier for avstandsavvikene fra sendeantennen i både asimut- og elevasjonsplanet fra de empiriske testene som ble gjort. Forventningsverdien for asimutplanet estimeres til  $\bar{X}_{asimut} = 0.14$  m og elevasjonsplanet estimeres til  $\bar{X}_{elevasjon} = 0.26$  m etter korrigerings for laserpekeravviket. Videre fremstilles 95 % konfidensintervallene for forventningsverdien i asimut- og elevasjonsplanet til henholdsvis  $[-0.11, 0.39]$  m og  $[-0.04, 0.57]$  m etter korrigerings.

Systemet som er utviklet under prosjektet forsøker å gi ett svar på nøyaktigheten for radiopeiling for 2.4 GHz, som det ikke lett kan finnes mye informasjon om. Prosjektet danner derfor et datagrunnlag for radiopeiling ved bruk av en robotarm. Det som kunne blitt undersøkt ytterligere er å sammenligne radiopeiling etter høyeste RSSI-verdi for flere mottakerantenner, med resultatet fra sumforskjell-metoden med 1x2-antennearrayet.

En mulig utvidelse av prosjektet er å anvende systemet for sanntidslokalisering av Bluetooth-enheter ved hjelp av et 3x3-antennearray montert på robotarmen. Systemet som er utviklet er skalerbart, og vil ved modifisering av programkode og utvidelse av antennearrayet kunne søke i asimut- og elevasjonsplanet samtidig. Innretningen kan derfor brukes til å optimalisere signalstyrken for en bevegelig senderkilde, som kan bidra til redusert strømforbruk for enhetene i bevegelse. En annen utvidelse kan være å lage et brukergrensesnitt som lar brukeren velge hvilken Bluetooth-enhet som skal lokaliseres i en meny. Søketimes for systemet kan reduseres betraktelig ved bruk av bedre enkodere og søkealgoritmer.

## Referanser

- Actorsfit (2022). *Basic knowledge of BLE broadcast, scan, and connection*. URL: <https://blog.actorsfit.com/a?ID=00500-39b968b5-daaa-4fc7-8812-fed9f8948ebf> (sjekket 14. mai 2022).
- All About Circuits (udatert). *Voltage Divider Calculator*. URL: <https://www.allaboutcircuits.com/tools/voltage-divider-calculator/> (sjekket 9. mai 2022).
- All3DP (2022). *The Best PLA Filament Special Blends in 2022*. URL: <https://all3dp.com/1/pla-filament-3d-printing/> (sjekket 4. mai 2022).
- Altium (2022). *Design Rules Available for PCB Layout in Altium Designer*. URL: <https://www.altium.com/documentation/altium-designer/pcb-design-rule-types> (sjekket 15. mai 2022).
- Anstensrud, T. (2021). *Multitrådprogrammering, IELET2102 Digitale regulerings-teknikker*. URL: <https://ntnu.blackboard.com> (sjekket 9. mai 2022).
- Bardwell, J. (2002). *Converting Signal Strength Percentage to dBm Values*. URL: <https://www.researchgate.net/file.PostFileLoader.html?id=55df0dd85cd9e3ee388b45d7%5C&assetKey=AS:273840230338570@1442300006396> (sjekket 16. mai 2022).
- Blake, C. (1971). *Phased Array Antennas*. URL: <https://ieeexplore.ieee.org/document/4130008> (sjekket 19. mai 2022).
- Bluetooth (2022). *Bluetooth Technology Overview*. URL: <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/> (sjekket 16. mai 2022).
- Bonafacic, D., J. Janculan og N. Majurec (2007). *Model of a Monopulse Radar Tracking System for Student Laboratory*. URL: [https://www.researchgate.net/publication/236178525\\_Model\\_of\\_a\\_Monopulse\\_Radar\\_Tracking\\_System\\_for\\_Student\\_Laboratory](https://www.researchgate.net/publication/236178525_Model_of_a_Monopulse_Radar_Tracking_System_for_Student_Laboratory) (sjekket 16. mai 2022).
- Cisco (2007). *Antenna Patterns and Their Meaning*. URL: <https://www.industrialnetworking.com/pdf/Antenna-Patterns.pdf> (sjekket 15. mai 2022).
- Cormen, T. H. mfl. (2022). *Introduction to algorithms*. 4. utg. Cambridge, Massachusetts: The Mit Press.
- Craig, K. (udatert). *Optical Encoders*. URL: [http://engineering.nyu.edu/mechatronics/Control\\_Lab/Craig/Craig\\_RPI/SenActinMecha/S%5C&A\\_Optical\\_Encoders.pdf](http://engineering.nyu.edu/mechatronics/Control_Lab/Craig/Craig_RPI/SenActinMecha/S%5C&A_Optical_Encoders.pdf) (sjekket 5. mai 2022).
- Cypress Semiconductor (2015). *Bluetooth Low Energy (BLE)*. URL: [https://www.infineon.com/dgdl/Infineon-Component\\_BLE\\_V2.0-Software+Module+Datasheets-v03\\_66-EN.pdf?fileId=8ac78c8c7d0d8da4017d0eae085e299e%5C&utm\\_source=cypress%5C&utm\\_medium=referral%5C&utm\\_campaign=202110\\_globe\\_en\\_all\\_integration-files](https://www.infineon.com/dgdl/Infineon-Component_BLE_V2.0-Software+Module+Datasheets-v03_66-EN.pdf?fileId=8ac78c8c7d0d8da4017d0eae085e299e%5C&utm_source=cypress%5C&utm_medium=referral%5C&utm_campaign=202110_globe_en_all_integration-files) (sjekket 14. mai 2022).
- DevAcademy (udatert). *Bare metal vs RTOS programming*. URL: <https://academy.nordicsemi.com/topic/bare-metal-vs-rtos-programming/> (sjekket 9. mai 2022).
- DFRobot (2022). *SER0038*. URL: <https://www.dfrobot.com/product-1177.html> (sjekket 13. mai 2022).
- Direktoratet for strålevern og atomsikkerhet (2019). *Veileder om sterke laserpekere (klasse 3R, 3B og 4)*. URL: <https://dsa.no/laser-og-lys/laserklasser> (sjekket 10. mai 2022).
- DYNAPAR (udatert). *Gray Code Encoder Overview*. URL: [https://www.dynapar.com/technology/encoder\\_basics/gray\\_code\\_encoders/](https://www.dynapar.com/technology/encoder_basics/gray_code_encoders/) (sjekket 10. mai 2022).

- edvinand (2021). *ppi\_pwm\_hands\_on*. URL: [https://github.com/edvinand/ppi\\_pwm\\_hands\\_on](https://github.com/edvinand/ppi_pwm_hands_on) (sjekket 11. mai 2022).
- ElectronicsTutorials (udatert[a]). *Pull-up Resistors*. URL: <https://www.electronicstutorials.ws/logic/pull-up-resistor.html> (sjekket 9. mai 2022).
- (udatert[b]). *Transistor as a Switch*. URL: [https://www.electronicstutorials.ws/transistor/tran\\_4.html](https://www.electronicstutorials.ws/transistor/tran_4.html) (sjekket 9. mai 2022).
- freeRTOS (udatert). *What is An RTOS*. URL: <https://www.freertos.org/about-RTOS.html> (sjekket 10. mai 2022).
- Frenzel, L. (2015). *Principles of Electronic Communication Systems*. 4. utg. McGraw Hill.
- Frey, S. og A. Locker (2021). *PETG Filament: Everything you Need to Know*. URL: <https://all3dp.com/1/petg-3d-printer-filament-all-you-need-to-know/> (sjekket 4. mai 2022).
- Galuscak, R. og P. Hazdra (2005). *Circular Polarization and Polarization Losses*. URL: [http://www.om6aa.eu/Circular\\_Polarization\\_and\\_Polarization\\_Losses.pdf](http://www.om6aa.eu/Circular_Polarization_and_Polarization_Losses.pdf).
- Gastreich, W. (2018). *WHAT IS A SERVO MOTOR AND HOW IT WORKS?* URL: <https://realpars.com/servo-motor/> (sjekket 9. mai 2022).
- Hollander, D. (2019). *How AoA & AoD Changed the Direction of Bluetooth Location Services*. URL: <https://www.bluetooth.com/blog/new-aoa-aod-bluetooth-capabilities/> (sjekket 19. mai 2022).
- Huynh, K. mfl. (2022). *nrf52832\_dev*. [https://github.com/anderszk/nRF52832\\_dev/blob/master/Other/Doxygen/df\\_ble.pdf](https://github.com/anderszk/nRF52832_dev/blob/master/Other/Doxygen/df_ble.pdf).
- IEEE (1984). *IEEE Standard Letter Designations for Radar-Frequency Bands*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=29086> (sjekket 12. mai 2022).
- ISO (1999). *C99*. URL: <https://www.iso.org/standard/29237.html>.
- ITU Radiocommunication Sector (2015a). *Calculation of free-space attenuation*. URL: [https://www.itu.int/dms\\_pubrec/itu-r/rec/p/R-REC-P.525-4-201908-!!!PDF-E.pdf](https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.525-4-201908-!!!PDF-E.pdf) (sjekket 15. mai 2022).
- (2015b). *Nomenclature of the frequency and wavelength bands used in telecommunications*. URL: [https://www.itu.int/dms\\_pubrec/itu-r/rec/v/R-REC-V.431-8-201508-!!!PDF-E.pdf](https://www.itu.int/dms_pubrec/itu-r/rec/v/R-REC-V.431-8-201508-!!!PDF-E.pdf) (sjekket 12. mai 2022).
- Jiang, X. J. og P. J. Scott (2020). *Gaussian Filter*. URL: <https://www.sciencedirect.com/topics/engineering/gaussian-filter> (sjekket 14. mai 2022).
- Jimblom (udatert). *Voltage Divider*. URL: <https://learn.sparkfun.com/tutorials/voltage-dividers/all> (sjekket 9. mai 2022).
- Johnson, R., H. Ecrer og J. Hollis (1973). *Determination OF Far-Field Antenna Patterns From Near-Field Measurements*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=1451288%5C&tag=1> (sjekket 11. mai 2022).
- Kauppila, I. (2021). *3MF File Format – All You Need to Know*. URL: <https://all3dp.com/1/3mf-file-format-all-you-need-to-know/> (sjekket 4. mai 2022).
- Kernighan, B.W. og D.M. Ritchie (1991). *The C programming language*. 2. utg. Englewood Cliffs, Nj Prentice-Hall.
- Klepp, I. G. (2020). *PLA*. URL: <https://snl.no/PLA> (sjekket 4. mai 2022).
- Kothari, D. P. og I. J. Nagrath (2018). *Electric machines*. 5. utg. McGraw Hill Education (India) Private Limited: Chennai, s. 805–811.
- Krytar (2022). *90 & 180 Degree Hybrid Coupler Primer*. URL: <https://krytar.com/resources-3/applications/primers/hybrid-coupler-primer/> (sjekket 12. mai 2022).



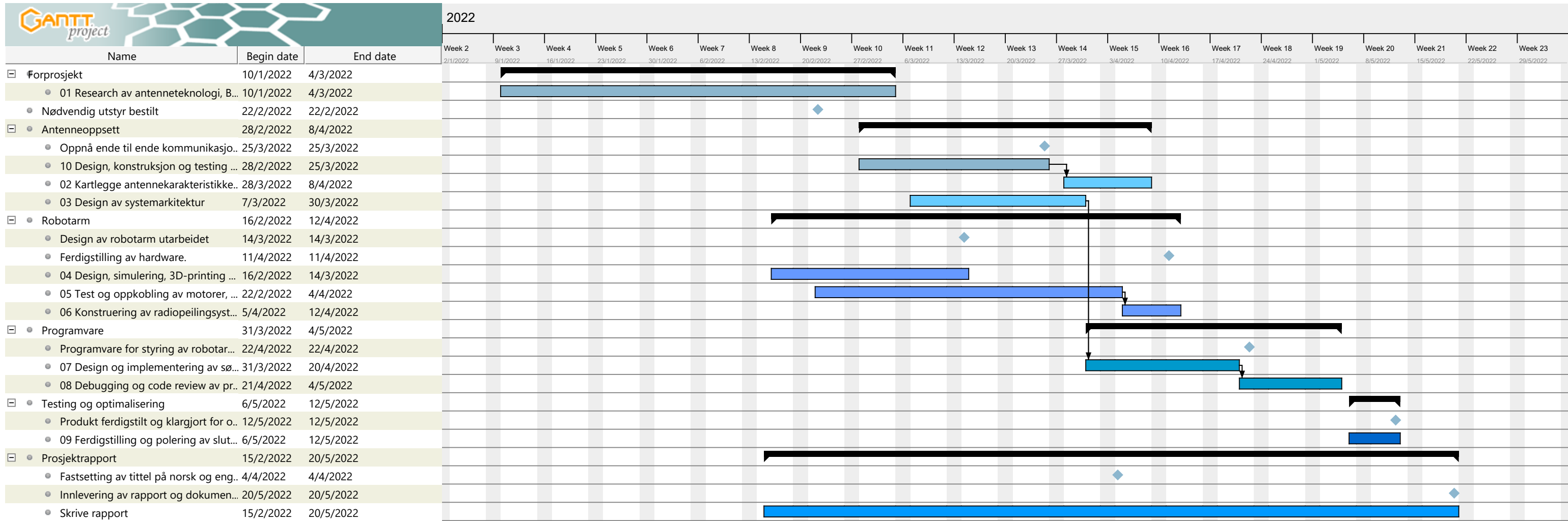
- Laird Connectivity (2022). *What is Bluetooth Class?* URL: <https://www.lairdconnect.com/support/faqs/what-bluetooth-class> (sjekket 16. mai 2022).
- Lambert, G. (2021). *How to build a pulse width modulation signal generator*. URL: <https://www.circuitbasics.com/pulse-width-modulation/> (sjekket 5. mai 2022).
- Library of Congress (2019). *STL (STereoLithography) File Format Family*. URL: <https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml> (sjekket 4. mai 2022).
- Ling, S. J., J. Sanny og W. Moebis (2021). *Unversity Physics Volume 1*. OpenStax, Rice University: Texas.
- Linx Technology (2019). *2.4 GHz Directional Embedded Ceramic Patch Antenna*. URL: [https://linxtechnologies.com/wp/wp-content/uploads/ant-2\\_4-cpa-ds.pdf](https://linxtechnologies.com/wp/wp-content/uploads/ant-2_4-cpa-ds.pdf) (sjekket 16. mai 2022).
- Lyons, R. (2011). *Sum of Two Sinusoids*. URL: [https://dspguru.com/files/Sum\\_of\\_Two\\_Sinusoids.pdf](https://dspguru.com/files/Sum_of_Two_Sinusoids.pdf) (sjekket 15. mai 2022).
- Løvås, G. G. (2018). *Statistikk for universiteter og høyskoler*. 4. utg. Universitetsforlaget: Oslo.
- MakeNTNU (2022). *Avansert 3D-printerkurs*. URL: <https://makentnu.no/news/events/67/> (sjekket 5. mai 2022).
- Mæhlum, L. (2021). *3D-printing*. URL: <https://snl.no/3D-printing> (sjekket 3. mai 2022).
- NetXL (2019). *The difference between Directional and Omni-Directional Antennas*. URL: <https://www.netxl.com/blog/networking/directional-or-omni-directional-antennas/> (sjekket 18. mai 2022).
- Nikitin, P. mfl. (2005). *Power Reflection Coefficient Analysis for Complex Impedances in RFID Tag Design*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=1504995>.
- Nikravan, A. og H. Schantz (2013). *Simple Formulas for Near-Field Transmission, Gain, and Fields*. URL: [https://www.researchgate.net/publication/263412970\\_Simple\\_Formulas\\_for\\_Near-Field\\_Transmission\\_Gain\\_and\\_Fields](https://www.researchgate.net/publication/263412970_Simple_Formulas_for_Near-Field_Transmission_Gain_and_Fields) (sjekket 15. mai 2022).
- Nordic Semiconductor (2020). *nRF52 DK*. URL: [https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fug\\_nrf52832\\_dk%5C%2FUG%5C%2Fnrf52\\_DK%5C%2Fintro.html](https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fug_nrf52832_dk%5C%2FUG%5C%2Fnrf52_DK%5C%2Fintro.html) (sjekket 13. mai 2022).
- (2021a). *Absolute maximum ratings*. URL: [https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps\\_nrf52810%5C%2Fgpio.html%5C&cp=3\\_4\\_0\\_5\\_7\\_2%5C&anchor=unique\\_1654232123](https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps_nrf52810%5C%2Fgpio.html%5C&cp=3_4_0_5_7_2%5C&anchor=unique_1654232123) (sjekket 12. mai 2022).
- (2021b). *GPIO - General purpose input/output*. URL: [https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps\\_nrf52840%5C%2Fgpio.html%5C&cp=4\\_0\\_0\\_5\\_8](https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps_nrf52840%5C%2Fgpio.html%5C&cp=4_0_0_5_8) (sjekket 11. mai 2022).
- (2021c). *GPiOTE - GPIO tasks and events*. URL: [https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps\\_nrf52840%5C%2Fgpiote.html%5C&cp=4\\_0\\_0\\_5\\_9\\_3%5C&anchor=topic](https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps_nrf52840%5C%2Fgpiote.html%5C&cp=4_0_0_5_9_3%5C&anchor=topic) (sjekket 11. mai 2022).
- (2021d). *nRF52832 Product Specification v1.8*. URL: [https://infocenter.nordicsemi.com/pdf/nRF52832\\_PS\\_v1.8.pdf](https://infocenter.nordicsemi.com/pdf/nRF52832_PS_v1.8.pdf) (sjekket 9. mai 2022).
- (2021e). *PPI - Programmable peripheral interconnect*. URL: [https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps\\_nrf52840%5C%2Fppi.html%5C&cp=4\\_0\\_0\\_5\\_15\\_1%5C&anchor=topic](https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fps_nrf52840%5C%2Fppi.html%5C&cp=4_0_0_5_15_1%5C&anchor=topic) (sjekket 11. mai 2022).
- (2021f). *Printed monopole antenna for 2.45GHz*. URL: [https://infocenter.nordicsemi.com/pdf/nwp\\_008.pdf?cp=17\\_19](https://infocenter.nordicsemi.com/pdf/nwp_008.pdf?cp=17_19) (sjekket 19. mai 2022).

- Nordic Semiconductor (2022). *Bluetooth Direction Finding*. URL: <https://www.nordicsemi.com/Products/Bluetooth-Direction-Finding> (sjekket 12. mai 2022).
- (udatert). *nRF Connect for Desktop*. URL: <https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop> (sjekket 13. mai 2022).
- NTNU (2021). *Alle oppgaveforslag for bacheloroppgave "elektroingeniør" i Trondheim, 2021/2022*. Institutt for teknisk kybernetikk, Norges teknisk-naturvitenskapelige universitet i Trondheim. Upublisert.
- Paleček, J. mfl. (2012). *Examination of SMA Connector Parameters*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=624984> (sjekket 13. mai 2022).
- Peterson, Z. (2021). *FR4 Dielectric Constant and Material Properties*. URL: <https://resources.altium.com/p/fr4> (sjekket 19. mai 2022).
- pSemi (2021). *pSemi product specification*. URL: <https://www.psemi.com/pdf/datasheets/pe42442ds.pdf> (sjekket 13. mai 2022).
- Shanhaban, A. (2022). *Introduction To Simplex, Half Duplex and Full Duplex*. URL: <https://www.scribd.com/document/480410477/Introduction-to-Simplex-Half-Duplex-and-Full-Duplex-n-pdf> (sjekket 19. mai 2022).
- Skøien, K. R. (2019). *RTOS: Real-Time Operating Systems for Embedded Developers*. URL: <https://blog.nordicsemi.com/getconnected/what-is-rtos-real-time-operating-systems-for-embedded-developers#:~:text=Using%5C%20an%5C%20RTOS%5C%20means%5C%20you,specific%5C%20requirements%5C%20of%5C%20your%5C%20project.> (sjekket 5. mai 2022).
- Sophocles, J. Orfandis (2016). *Electromagnetic Waves and Antennas*. 2-up ed. Rutgers University.
- Spong, M. W., I. Hutchinson og M. Vidyasagar (2020). *Robotic Modeling and Control*. 2. utg. 2020.
- Times Microwave System (2022). *Coaxial Cable Attenuation & Power Handling Calculator*. URL: <https://www.timesmicrowave.com/Calculator> (sjekket 19. mai 2022).
- Townsend, K. (2014). *Introduction to Bluetooth Low Energy*. URL: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap> (sjekket 14. mai 2022).
- VINK (2022). *POM*. URL: <https://vink.no/produkter/industri/konstruksjonsplast/pom> (sjekket 10. mai 2022).
- Wen, J. og Chang J. (1990). *The Effect of Multipath Interference on the Performance of Packet Radios*. URL: [https://www.researchgate.net/publication/3157512\\_The\\_Effect\\_of\\_Multipath\\_Interference\\_on\\_the\\_Performance\\_of\\_Packet\\_Radios](https://www.researchgate.net/publication/3157512_The_Effect_of_Multipath_Interference_on_the_Performance_of_Packet_Radios) (sjekket 16. mai 2022).
- Wheeler, H. (1977). *Transmission-Line Properties of a Strip on a Dielectric Sheet on a Plane*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=1129179> (sjekket 15. mai 2022).
- Würth (2019). *Range Estimator Application Note*. URL: [https://www.we-online.com/catalog/media/o164527v410%5C%20ANR010\\_Range\\_Estimation.pdf](https://www.we-online.com/catalog/media/o164527v410%5C%20ANR010_Range_Estimation.pdf) (sjekket 16. mai 2022).
- Yaacob, N. mfl. (2017). *Link Budget Calculator System for Satellite Communication*. URL: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=%5C&arnumber=8298397> (sjekket 15. mai 2022).
- Zephyr (2022a). *BLE Broadcaster*. URL: <https://docs.zephyrproject.org/latest/samples/bluetooth/broadcaster/README.html> (sjekket 10. mai 2022).

- Zephyr (2022b). *BLE Observer*. URL: <https://docs.zephyrproject.org/latest/samples/bluetooth/observer/README.html> (sjekket 10. mai 2022).
- (2022c). *Zephyr Examples*. URL: <https://docs.zephyrproject.org/latest/samples/index.html> (sjekket 12. mai 2022).

# A Gantt Untitled Gantt Project

## Gantt Chart



## B Budsjett

Utstyr	Delnummer	Produsent	Antall enheter	Sats (kr)	Kostnad (kr)
Antennepatch	ANT-2.4-CPA	Linx Technologies Inc.	2	34,12	34,12
Breadboard	920-0031-01	Smartboard Inc.	1	71,39	71,39
Dupontkabler	1528-1964-ND	Adafruit Industries LLC	1	17,51	17,51
Enkoder	AMT103-2048-N4000-S	CUI Devices	2	207,46	414,92
Enkoderkonnektor	3-640440-5	TE Connectivity AMP Connectors	2	3,95	7,9
Evalueringkort	EK42442-01	pSemi	1	942,9	942,9
Hybrid-kobler	3A0200	Anaren	1	-	-
Koaksialkabel 152,4mm	BU-4150029006	Mueller Electric co	2	89,53	179,06
Koaksialkabel 457,2mm	415-0029-018	Cinch Connectivity Solutions Johnson	2	133,53	267,06
Laserdiode	1528-1391-ND	Adafruit Industries LLC	1	53,43	53,43
Mikroservo	SER0049	DFRobot	1	44,9	44,9
Multiprotokoll SoC	nRF52-DK	Nordic Semiconductor ASA	2	349,05	698,1
Pigtail	MXHS83QE3000	Murata Electronics	1	275,48	275,48
Servomotor	SER0038	DFRobot	2	133,36	266,72
SMA-konnektor	60312202114514	Würth	2	64,84	129,68
Strømforsyning	GST40A07-P1J	Mean Well USA Inc.	1	182,76	182,76
Taktilbryter	CKN12302-ND	C&K	2	0,9	2,7
<b>Total:</b>					3622,75

## C Fullstendig delliste

Utstyr	Antall
Antennepatch	2x
Breadboard	1x
Dupontkabler	38x
Enkoder	2x
Enkoderkonnektor	2x
Evalueringskort	1x
Hybrid-kobler	1x
Koaksialkabel 152,4mm	2x
Koaksialkabel 457,2mm	2x
Laserdiode	1x
Mikroservo	1x
nRF52-DK	2x
Pigtail	1x
Servomotor	2x
SMA-konnektor	2x
Strømforsyning	1x
Taktilbryter	3x
Motstand 220 $\Omega$	1x
Motstand 1 k $\Omega$	5x
Motstand 1.5 k $\Omega$	4x
Transistor BC547B	1x
M4 Mutter	4x
40 mm M4 gjengestang	4x
8 mm M4 maskinskrue	4x
8 mm M3 maskinskrue	8x
6 mm M3 maskinskrue	12x
4 mm M3 maskinskrue	3x
5 mm M2 maskinskrue	4x
4 mm M3 monteringskrue	4x
4 mm M2 monteringskrue	6x
200x200x10 mm POM-C baseplate	1x
PLA-filament	350 gram

## D Møtereferat

### Møtereferat fra møte med Egil Eide.

- Gruppen ønsket et møte med Egil Eide for å høre hans sitt perspektiv på hvilken antennetype som egner seg best til peiling av antennesignal.
- Egil var fast bestemt på at beste fremgangsmåte var bruk av antennepatch arrays, da i form av 2x2 eller mer. Egil forklarte at peiling ved bruk av nullpunkt som oppstår når en setter antennene i motfase vil en få best resultater. Dette kan oppnås ved å sette to omni-direksjonelle med en gitt bølgelengde unna. Man vil dermed kunne spisse inn et nullpunkt.
- Egil forklarer videre at dette vil kreve at man tar i bruk analoge switcher, da kortet ikke har mulighet til å behandle denne dataen som vi ønsker. Dette er for å faseforskyve det ene signalet, slik at man får et nullpunkt.

Egil mener også det er hensiktsmessig å kunne bruke en switch da man ofte summerer målingene fra antenne både i fase og motfase. Dette gjør at man har mulighet til å peile seg inn nøyaktig i motfase, men også orientere seg mot riktig nullpunkt ved hjelp av summering av antenne i fase (max RSSI). Dette er fordi det kan finnes flere nullpunkt i søkeområde, og hvis algoritmen som blir utviklet ikke tar hensyn til dette vil den kunne sikte seg mot et annet nullpunkt enn det vi ønsker.

Switchene vil også ha som ulempe at det vil oppstå tap i connectoren og i selve switchen, noe som vil være med på å redusere kvaliteten til systemet, da signalet må gå gjennom flere ledd og koblinger.

- Problemet med denne løsningen er at gruppen måtte selv designe antennen og da lage selve arrayet. Dette syntes gruppen er utenfor den tekniske kompetansen gruppen har og at det heller passet på en oppgave på master-nivå. Dette er fordi konstruering av slike antenner krever høy posisjon og prosessen med å teste og modifisere antennen kan være meget tidskrevende. Dette synes vi virket mot sin hensikt, da oppgaven er avhengig av at antennen er på plass relativt tidlig.
- Etter samtale med Arne Midjo for å diskutere det som ble tatt opp på møte med Egil ble alle parter enig at design og produsere egen svitsje krets er utenfor forventet nivå på en bachelor og at gruppen mangler 3 emner for å utføre det arbeidet. Gruppen konkluderte dermed å kombinere bruken av en hybridkobler, digital RF svitsj og keramiske patch-antenner ville være den beste løsningen å gå for.

# E Risikoanalyse

## RISIKOANALYSE

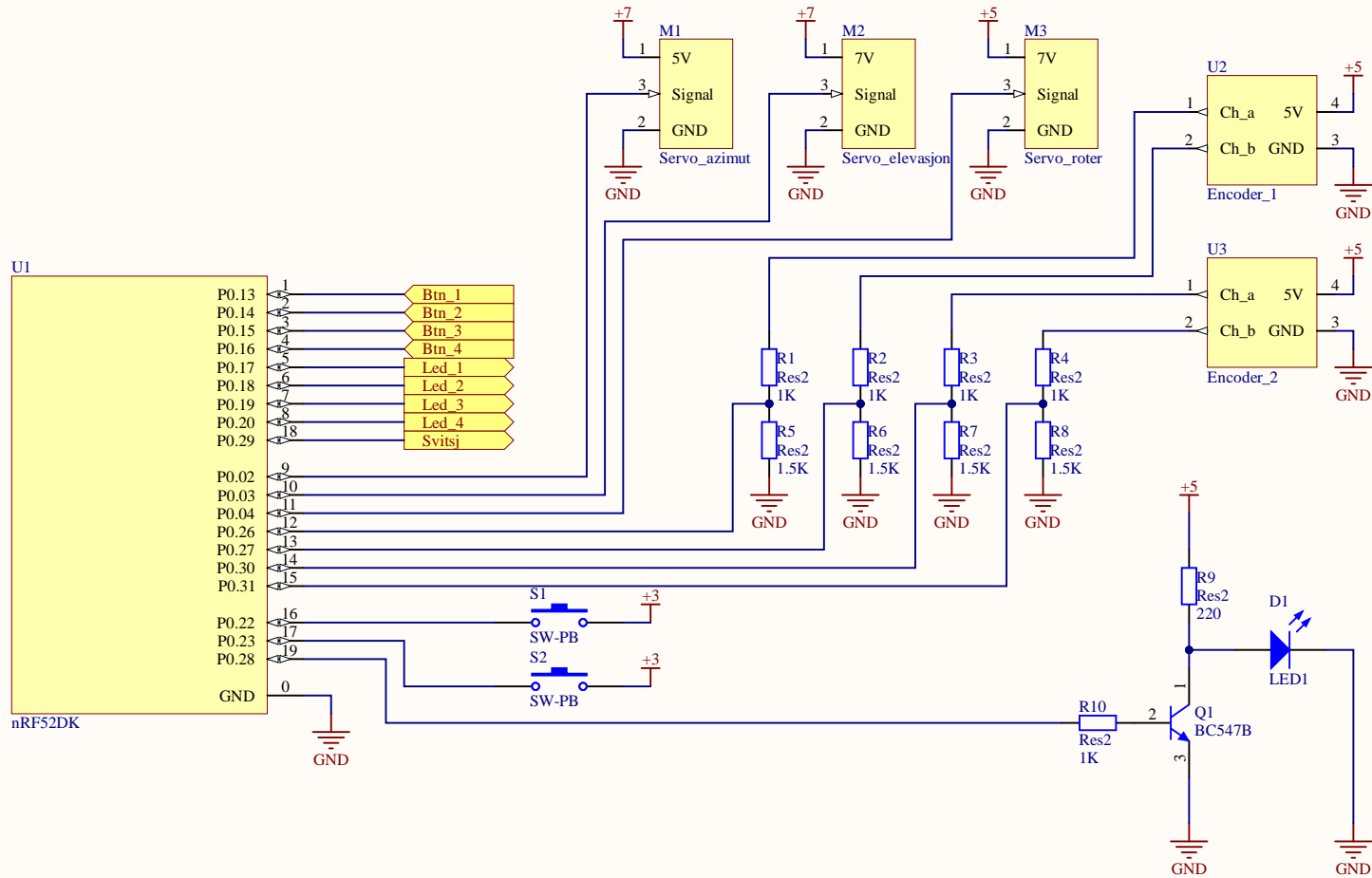
<b>Enhet/Institutt:</b>	NTNU	<b>Dato opprettet:</b>	28.01.2022
<b>Ansvarlig linjeleder (navn):</b>	Arne Midjo	<b>Sist revidert:</b>	10.05.2022
<b>Ansvarlig for aktiviteten som risikovurderes (navn):</b>	Khuong Huynh		
<b>Deltakere (navn):</b>	Anders Kristensen, Filip August Walla Saasen, Håvard Olai Kopperstad og Khuong Huynh		

<b>Beskrivelse av den aktuelle aktiviteten, området mv.:</b>
Gruppen skal designe, konstruere og teste en robotarm. Det skal både gjøres research, lodding, mekanisk arbeid og elektrisk arbeid.

Aktivitet/arbeidsoppgave	Mulig uønsket hendelse	Eksisterende risikoreducerende tiltak	Vurdering av sannsynlighet (S) (1-5)	Vurdering av konsekvens (K) <i>Vurder en konsekvenskategori om gangen. Menneske skal alltid vurderes.</i>				Risikoverdi (S x K)	Forslag til forebyggende og/eller korrigerende tiltak <i>Prioriter tiltak som kan forhindre at hendelsen inntreffer (sannsynlighetsreducerende tiltak) foran skjerpet beredskap (konsekvensreducerende tiltak)</i>	Restrisiko etter tiltak (S x K)
				Menneske (1-5)	Øk/materiell (1-5)	Ytre miljø (1-5)	Omdømme (1-5)			
Bestilling av deler	Forsinkelser av deler	Bestille i god tid	3		3			9	Sjekk lagerstatus på varer før en bestiller	6(2x3)
	Underdimensjonert utstyr	Kalkulering av nødvendige parametere	3		3			9	Bruk av ressurser på NTNU for beregning av nødvendige parametere	6(2x3)
Research av fagstoff	Ergonomiske komplikasjoner	Pauser, daglig strekk	4	2				8	Gruppen gjennomfører kurs i ergonomi og må delta i ukentlig fysisk aktivitet.	4 (2x2)
3D printing av deler	Hånd i 3D printer	Sensorer, varsel skilt	2	2	2		3	6	Kurs angående sikkerhetsrutiner rundt 3D printeren	3(1x3)
Lodding av utstyr	Brannskade	Loddekurs	4	2				8	PVU	4(2x2)
Oppkobling og test av robotarm	Strømgjennomgang	Lesing av brukermanualer	4	4	3			20	MLP, rutiner for strømsetting	8(2x4)
	Kutt/Klem skade	PVU	5	3	3			15	God kommunikasjon ved test av robotarm, oppmerking av faresone	6(2x3)
Antenne lab	Antennelab er fullbooket eller utilgjengelig	Kontakt med antennelabansvarlig	2		3			6	Holde dialog oppe med Nordic Semiconductor angående lab	3(1x3)
	Labveileder ikke tilgjengelig pga sykdom	i god tid for å planlegge								
Generelt arbeid	Sykdom/karantene	Godt smittevern	4	2				8	Begrense nærkontakter	6(3x2)
	Mangelfull kommunikasjon med veilederne	Opprettet kommunikasjonsplattform	4		2			8	Begrense mail til strengt nødvendig kommunikasjon	6(3x2)
Programmering av programvare	Utvikling av programvare stopper grunnet kompleksitet i nytt programmeringsspråk	Grunnleggende programmeringskunnskaper	3		3			9	Begynne tidlig med å lese seg opp i manualer/eksempelkoder	6(3x2)
	Bruk av laserpeker	Laserstrålen treffer øynene til tilskuere ved at robotarmen roterer 0-180 grader asimut og 0-50 grader elevasjon	Ingen	3	1		3	6	Systemet merkes i samsvar gul/svart trekant og tilskuere varsels på forhånd	3(1x3)

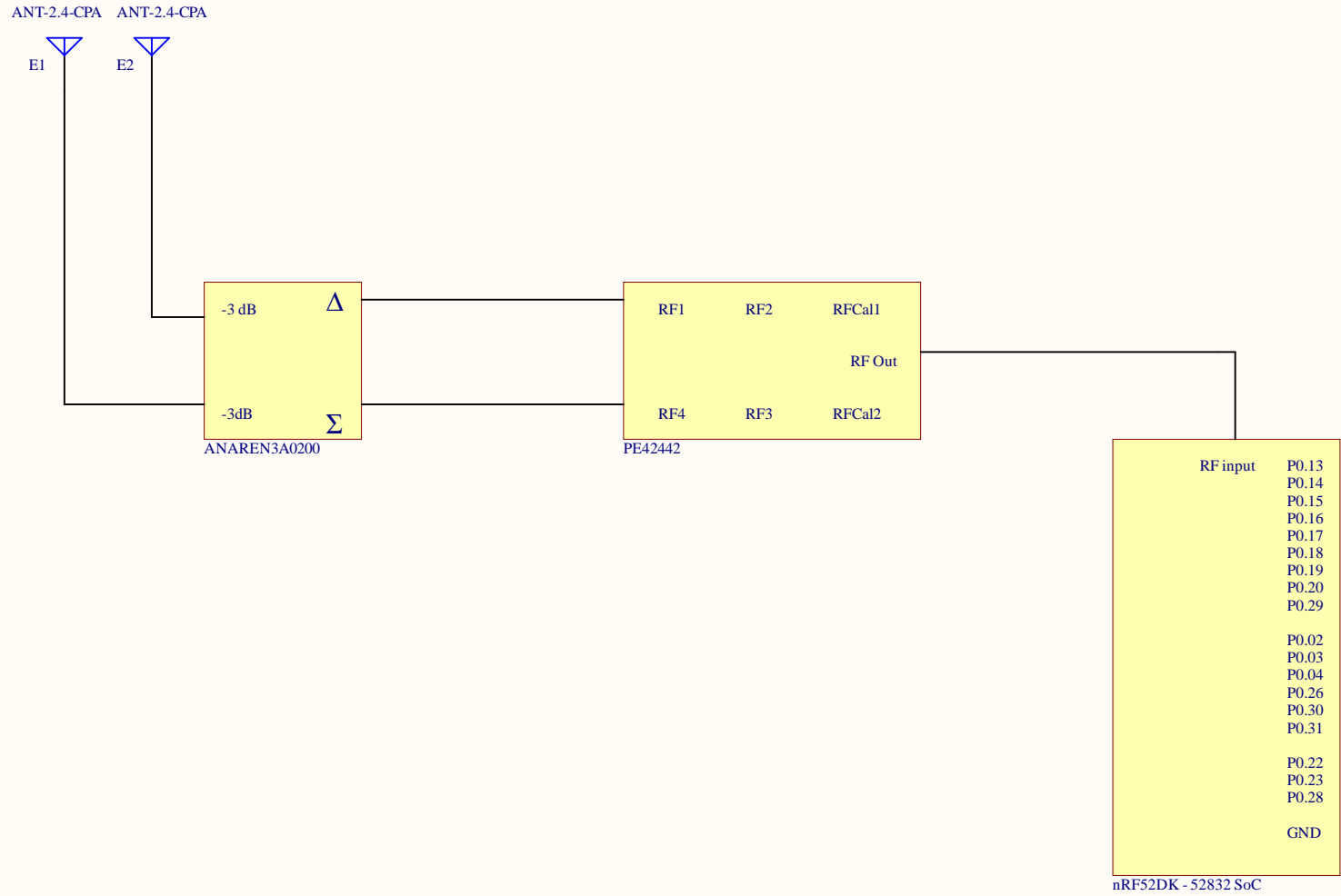


# F Koblingskjema



Title		
Size	Number	Revision
A4		
Date:	5.07.2022	Sheet of
File:	C:\Users\...\GPIO_connections_v2.SchDoc Drawn By:	

# RF Koblingskjema



Title RF Schematics		
Size A4	Number	Revision 1.0
Date: 5.18.2022	Sheet of	
File: C:\Users\...\RF_Schematics.SchDoc	Drawn By: <a href="#">Khuong Huynh</a>	

# Testskjema

E2226 – AUTOMATISK RADIOPEILING FOR 2.4 GHZ

# Introduksjon

## 1.1 Hensikt

Dette testskjemaet skal fungere som et dokument for verifisering av funksjonalitetene på innretningen som er blitt utviklet og skjemaet inneholder retningslinjer for utprøving av denne. Det skal blant annet verifiseres at robotarmen kan manøvrere seg i horisontal- og vertikalretning, at søkealgoritmen er funksjonell i asimut- og elevasjon planet og at robotarmen klarer å orientere seg mot en senderantenne.

## 1.2 Utstysrliste

Utstyret som trengs for å gjennomføre testen av robotarmen er listet nedenfor:

1. PC med følgende programvare:
  - VSCode
  - nRF Connect SDK v. 1.9.1
2. nRF52DK, med nRF52832 SoC – Programmert som robotarm
3. nRF52DK, med nRF523832 SoC – Programmert som Beacon
4. Anaren 3A0200
5. EK42442-01 – Evalueringskort for svitsjing av RF signaler
6. GST40A07-P1J Strømforsyning
7. Ferdig koblet robotarm (fullstendig utstyr for robotarm kan finnes i egen utstysrliste)
8. Micro USB kabel (til opplasting av kode)

### 1.3 Oppsett

1. Start med å laste ned repositorien fra «[https://github.com/anderszk/nRF52832\\_dev](https://github.com/anderszk/nRF52832_dev)» som en zip-fil. Ekstrakter deretter mappen «code» til ønsket lokasjon i mappestrukturen.
2. Nå skal programvare lastes opp til senderantennen. Åpne opp VSCode og trykk på «nRF Connect» ikonet som ligger på venstre side. Lag deretter en ny applikasjon etter «Zephyr/samples/blueooth/beacon». Legg så til en «build» konfigurasjon for gjeldende applikasjon (viktig at det blir valgt riktig utviklingskort). Last til slutt opp applikasjonen til mikrokontrolleren ved å trykke på «flash» under «Actions». Mikrokontrolleren er nå programmert til et «Beacon» om det ikke dukket opp noen feilkoder under opplasting (se bilde under for hvordan en vellykket opplasting ser ut).

```
*** Booting Zephyr OS build v2.7.99-ncs1-1 ***
Starting Beacon Demo
[0]Bluetooth initialized
Beacon started, advertising as EB:2C:98:21:C5:0A (random)
0:00:00.006,683] <inf> sdc_hci_driver: SoftDevice Controller build revision:
                                0e e7 c5 66 67 18 3c ac b3 d2 cc 81 a3 dc f1 c0 |...fg.<. ....
                                c0 36 02 22 |.6."
[00:00:00.009,216] <inf> bt_hci_core: HW Platform: Nordic Semiconductor (0x0002)
[00:00:00.009,246] <inf> bt_hci_core: HW Variant: nRF52x (0x0002)
[00:00:00.009,277] <inf> bt_hci_core: Firmware: Standard Bluetooth controller (0x00) Version 14.50663 Build 1008232294
[00:00:00.010,223] <inf> bt_hci_core: Identity: EB:2C:98:21:C5:0A (random)
[00:00:00.010,253] <inf> bt_hci_core: HCI: version 5.2 (0x0b) revision 0x12fe, manufacturer 0x0059
[00:00:00.010,253] <inf> bt_hci_core: LMP: version 5.2 (0x0b) subver 0x12fe
```

Figur 1: Viser forventet informasjon ifra NRF Terminal ved vellykket opplastet kode til senderantennen.

3. Nå må programvaren som skal lastes opp til robotarmen konfigureres. Åpne opp et nytt vindu i VSCode og trykk på «nRF Connect» ikonet som ligger på venstre side. Legg til mappen «code» som tidligere ble lastet ned. Det må nå spesifiseres hvilken adresse robotarmen skal peile seg etter. Dette gjøres ved å åpne «observer.c» under «src» → «Handlers». Kopier så adressen «Identity» ifra senderantennen til linje 87 i «observer.c» (se bilde nedenfor).

```
87     err = add_filter_accept_list_from_string("E7:9E:11:9E:33:EE", "(random)");
88     if (err){
89         printk("Could not add to acceptlist (error: %d).", err);
90         return err;
91     }
```

Figur 2: Viser hvor adressen til senderantennen skal kopieres til

Videre nå må applikasjonen konfigureres etter hvilken test som skal bli gjennomført og parameterne ligger under «src» → «Search\_algorithms» → «search.h».

- a. For testing av manøvrering av robotarmen må disse det konfigureres slik:

```
3 #define MOTOR_TEST 1
4
5 /**
6  * @brief Activates search in azimuth
7  * 0 to deactivate search in azimuth, 1 to activate search in azimuth
8  *
9  */
10 #define SEARCH_AZIMUTH 1
11 /**
12  * @brief Activates serach in elevation
13  * 0 to deactivate search in elevation, 1 to activate search in elevation
14  */
15 #define SEARCH_ELEVATION 1
16 /**
17  * @brief Activates fine search
18  * 0 to deactivate fine search, 1 to activate fine search
19  */
20 #define FINE_ACTIVATE 1
21
22
```

Figur 3: Viser konfigurering av parametere ved test av manøvrering av robotarm

- b. For testing av søkealgoritmen i det horisontale planet må parameterne konfigureres slik:

```
3 #define MOTOR_TEST 0
4
5 /**
6  * @brief Activates search in azimuth
7  * 0 to deactivate search in azimuth, 1 to activate search in azimuth
8  *
9  */
10 #define SEARCH_AZIMUTH 1
11 /*0
12  * @brief Activates serach in elevation
13  * 0 to deactivate search in elevation, 1 to activate search in elevation
14  */
15 #define SEARCH_ELEVATION 0
16 /**
17  * @brief Activates fine search
18  * 0 to deactivate fine search, 1 to activate fine search
19  */
20 #define FINE_ACTIVATE 0
21
```

Figur 4: Viser konfigurering av parametere ved test av søkealgoritme i det horisontale plan

- c. For testing av søkealgoritmen i elevasjonsplanet må parameterne konfigureres slik:

```
3 #define MOTOR_TEST 0
4
5 /**
6  * @brief Activates search in azimuth
7  * 0 to deactivate search in azimuth, 1 to activate search in azimuth
8  *
9  */
10 #define SEARCH_AZIMUTH 0
11 /*0
12  * @brief Activates search in elevation
13  * 0 to deactivate search in elevation, 1 to activate search in elevation
14  */
15 #define SEARCH_ELEVATION 1
16 /**
17  * @brief Activates fine search
18  * 0 to deactivate fine search, 1 to activate fine search
19  */
20 #define FINE_ACTIVATE 0
21
22
```

Figur 5: Viser konfigurering av parametere ved test av søkealgoritme i det vertikale plan

- d. For en fullstendig test av robotarmen, da i to plan og fullstendig søkealgoritme må parameterne konfigureres slik:

```
2
3 #define MOTOR_TEST 0
4
5 /**
6  * @brief Activates search in azimuth
7  * 0 to deactivate search in azimuth, 1 to activate search in azimuth
8  *
9  */
10 #define SEARCH_AZIMUTH 1
11 /*0
12  * @brief Activates search in elevation
13  * 0 to deactivate search in elevation, 1 to activate search in elevation
14  */
15 #define SEARCH_ELEVATION 1
16 /**
17  * @brief Activates fine search
18  * 0 to deactivate fine search, 1 to activate fine search
19  */
20 #define FINE_ACTIVATE 1
21
```

Figur 6: Viser konfigurering av parametere ved fullstendig test av robotarm

4. Etter at applikasjonen er konfigurert i henhold til hvilken test som skal bli gjennomført er det klart for å laste opp koden til robotarmen. Legg nå til en «build» konfigurasjon for gjeldende applikasjon (viktig at det blir valgt riktig utviklingskort). Last til slutt opp applikasjonen til mikrokontrolleren ved å trykke på «flash» under «Actions». Mikrokontrolleren er nå programmert om det ikke dukket opp noen feilkoder under opplasting (se bilde under for hvordan en vellykket opplasting ser ut).

```
*** Booting Zephyr OS build v2.7.99-ncs1-1 ***
Hello Nordic Semiconductor!
Buttons and leds initiated.
Timer initialized.
Timer started.
Starting Observer.
[0Bluetooth initialized.
Initializing switch on pin 29 success!
Initializing servo 0 on pin 2 success!
Initializing servo 1 on pin 3 success!
Initializing servo 2 on pin 4 success!
0:00:00.014,862] <inf> sdc_hci_driver: SoftDevice Controller build revision:
                                0e e7 c5 66 67 18 3c ac b3 d2 cc 81 a3 dc f1 c0 |...fg.<. ....
                                c0 36 02 22                                |.6."
Azimuth QDEC initialization was successful.
Laser init successfull.
Starting coarse sweep in Azimuth.
Azimuth QDEC initialization was successful.
█
```

Figur 7: Viser forventet informasjon ifra NRF Terminal ved vellykket opplastet kode til robotarmen.

5. Dersom oppsettet fra punkt 1 til punkt 4 har gått uten problemer, skal oppsettet være fullført. Testen kan nå startes.



## Testskjema

### Teknisk informasjon før igangsetting:

Testskjemaoppsett godkjent: (Marker med kryss)	Godkjent:	Ikke godkjent:
Hva som skal testes:	Prosessregulering av vannivå i tank med tilhørende brukergrensesnitt, feildeteksjons- og feilhåndteringssystem og lead-lag-element.	
Demonstrasjonsansvarlig:	Khuong Huynh (KH)	
Kunde/tester:	Torolv Skjølvsvik (TS)	
Andre tilstedeværende:	<b>Prosjektmedarbeidere:</b> Anders Ziener Kristensen (AZK) Fillip Saasen (FS) Håvard Olai Kopperstad (HOK) Khuong Huynh (KH)  <b>Veiledere:</b> Arne Midjo (AM) Torolv Skjølvsvik (TS)	
Dato og sted:	TBD	
Testoppsummering:	Testen skal stegvis ta for seg funksjonalitet for manøvrering av robotarmen, kontrollere at søkealgoritmen fungerer i elevasjon og asimut hver for seg og at robotarmen klarer å orientere seg etter senderantennen	
Kriterier for godkjenning:	Alle delpunktene i hver deltest skal være godkjente før testen av robotarmen godkjennes. Dette innebærer at robotarmen skal kunne bli styrt manuelt, muligheten til å søke i et enkelt plan og at robotarmen faktisk klarer å peile seg etter senderantennen.	
Resultat: (Marker med kryss)	Godkjent:	Ikke godkjent:

Dato: TBD

TBD

Testansvarlig:

Khuong Huynh

Kunde:

Torolv Skjølvsvik

## Testutførelse:

Manøvrering av robotarm	Forventet resultat	Godkjennelse
Beskrivelse: Testen skal fremvise at robotarmen kan manøvrere seg vertikalt og horisontalt og at man skal kunne avlese posisjonen til hver av motorene.		
1. Konfigurer parametere i henhold til punkt. 3.a i oppsett og last opp koden.	Program kode skal nå være opplastet til robotarmen og være klar for test av manøvrering.	
2. Trykk på knapp 1 på utviklingskortet	Vi ser at robotarmen beveger seg til venstre i det horisontale plan. Gjenta denne prosessen 10 ganger for å se at den flytter seg 10 grader.	
3. Trykk knapp 3 på utviklingskortet	NRF Terminalen vil skrive ut posisjonen til robotarmen i det horisontale plan.	
4. Trykk knapp 2 på utviklingskortet	Vi ser at robotarmen beveger seg til høyre i det horisontale plan. Gjenta denne prosessen 10 ganger for å se at den flytter seg 10 grader.	
5. Trykk knapp 3 på utviklingskortet	NRF Terminalen vil skrive ut posisjonen til robotarmen i det horisontale plan.	
6. Trykk knapp 4 på utviklingskortet	NRF Terminalen vil gi beskjed at styring det er nå mulig å styre robotarmen i det vertikale plan.	
7. Gjenta punkt 2 til punkt 5	Vi ser nå at robotarmen vil bevege seg i det vertikale plan. Da først oppover deretter nedover. NRF Terminalen vil også skrive ut posisjonen til robotarmen i det vertikale plan ved påtrykt knapp 3.	
Kommentar: Dersom robotarmen beveger seg som beskrevet i punktene og NRF Terminalen skriver ut den riktige informasjonen vil funksjonaliteten for manøvrering være verifisert.		

Test av søk i enkelt plan.	Forventet resultat	Godkjenning
<p>Beskrivelse: I denne deltesten skal det fremvises funksjonaliteten av søkealgoritmen i det horisontale- og vertikale plan. I tillegg at man har muligheten til å foreta søk i et enkelt plan.</p>		
1. Konfigurer parametere i henhold til punkt. 3.b i oppsett og last opp koden.	Program kode skal nå være opplastet til robotarmen og være klar for test av søk i det horisontale plan.	
2. Plasser robotarmen på ønsket plass og koble til strømforsyningen.	Robotarmen har nå fått strømforsyning og er klar nå klar for søk.	
3. Plasser senderantennen med opplastet applikasjon på ønsket posisjon og skru denne på.	Mikrokontrolleren skal nå lyse og indikerer at den sender ut signaler.	
4. Vent 2 minutter.	Robotarmen vil nå søke seg igjennom det horisontale plan og nRF Terminalen vil skrive ut verdiene som blir målt. Da i rekkefølgen: Posisjon, signalstyrke i fase, signalstyrke i motfase. Ved gjennomført søk vil robotarmen flytte seg til hvor den tror senderantennen er lokalisert.	
5. Skru deretter av senderantennen	Mikrokontrolleren vil nå slutte å lyse og indikerer at den ikke sender ut signaler	
6. Konfigurer så parameterne i henhold til punkt 3.c i oppsett og last opp koden.	Program kode skal nå være opplastet til robotarmen og være klar for test av søk i det vertikale plan.	
7. Gjenta punkt 2 til punkt 5.	Robotarmen vil snu antennen 90 grader og begynne å søke seg igjennom det vertikale plan og terminalen vil skrive ut i samme rekkefølge som tidligere. Robotarmen vil etter gjennomført søk flytte seg til hvor den tror senderantennen er lokalisert.	
<p>Kommentar: Dersom robotarmen gjennomfører søkene som beskrevet i punktene vil funksjonaliteten til søkealgoritmen bli verifisert og det viser at man har mulighet til å gjennomføre søk i kun et enkelt plan. (Ettersom at det er kun søk i et plan vil det være en sannsynlighet for at den bommer)</p>		

Fullskala test av robotarm	Forventet resultat	Godkjennelse
<p>Beskrivelse:</p> <p>I denne deltesten skal full funksjonalitet av robotarm samt søkealgoritme testes. Testen vil vise resultatet av design av robotarm, søkealgoritme og antennedesign.</p>		
1. Konfigurer parametere i henhold til punkt. 3.d i oppsett og last opp koden.	Program kode skal nå være opplastet til robotarmen og være klar for en fullskaletest.	
2. Plasser robotarmen på ønsket plass og koble til strømforsyningen.	Robotarmen har nå fått strømforsyning og er klar nå klar for søk.	
3. Plasser senderantennen med opplastet applikasjon på ønsket posisjon og skru denne på.	Mikrokontrolleren skal nå lyse og indikerer at den sender ut signaler.	
4. Vent til robotarmen har fullført søket.	Robotarmen vil starte å søke gjennom arbeidsområdet i det horisontale plan før den beveger seg til hvor den tror senderantennen er lokalisert. Antennen vil så roteres 90 grader før den gjennomfører et søk igjennom hele arbeidsområdet i det vertikale plan. Antennen vil rotere 90 grader tilbake og robotarmen vil starte finsøket i det horisontale plan. Avhengig av målingene robotarmen får vil denne prosessen gjentas «x» ganger før robotarmen avgjør posisjonen til senderantennen i det horisontale plan. Antennen vil deretter roteres 90 grader og sammen sekvens gjennomføres for det vertikale plan. Etter endt søk vil laseren skru seg på og robotarmen vil stoppe.	
<p>Kommentar:</p> <p>Dersom robotarmen gjennomfører søket som beskrevet i punktene vil funksjonaliteten til robotarmen bli verifisert og det viser at man har mulighet til å gjennomføre søk i kun et enkelt plan. (Ettersom at det er kun søk i et plan vil det være en sannsynlighet for at den bommer)</p>		

# I C kode

## Main.c

```
main.c
1  #include "initiator.h"
2  #include "nrfx_qdec.h"
3  #include <stdio.h>
4
5  K_SEM_DEFINE(my_sem,0,1);
6
7  zeros zero_enc_values;
8
9
10 void main(void)
11 {
12     printk("Hello Nordic Semiconductor!\n");
13     int err = initiate_modules();
14     if(err){
15         printk("Error while initiating modules (error:
16             ↪ %d). \n", err);
17         exit();
18     }
19     k_sem_take(&my_sem, K_FOREVER);
20     k_msleep(2000);
21
22     zero_enc_values = coarse_search();
23     printk("azimuth zero enc: %d, horizontal zero end: %d\n",
24         ↪ zero_enc_values.azimuth, zero_enc_values.elevation);
25     zero_enc_values = fine_search(zero_enc_values);
26     printk("Search is done\n");
27     printk("Zeroes found at Azimuth: %d, Elevation: %d\n",
28         ↪ zero_enc_values.azimuth, zero_enc_values.elevation);
29     angle_move_servo(0,zero_enc_values.azimuth);
30     angle_move_servo(1, zero_enc_values.elevation);
31     laser_set(0);
32
33     while(1){
34         k_sleep(K_FOREVER);
35     }
36 }
```

## initiator.c

```
initiator.c
1  #include "initiator.h"
2
3
4  int initiate_modules(){
5      int err;
6      err = configure_dk_buttons_leds();
7      if (err){
8          printk("Error while configuring buttons and leds
9              ↪ (error: %d).\n", err);
10         return err;
11     }
12     err = timer_init();
13     if (err){
14         printk("Error while configuring timer (error:
15             ↪ %d).\n", err);
16         return err;
17     }
18     err = timer_start();
19     if (err){
20         printk("Error while starting timer (error:
21             ↪ %d).\n", err);
22         return err;
23     }
24     err = init_bluethooth_scan();
25     if (err){
26         printk("Error while configuring bluethooth and
27             ↪ observer (error: %d).\n", err);
28         return err;
29     }
30     err = init_encoder_servos();
31     if (err){
32         printk("Error while configuring servomotors
33             ↪ (error: %d).\n", err);
34         return err;
35     }
36     err = init_encoder_azimuth();
37     if (err != NRFX_SUCCESS){
38         printk("Error while configuring Azimuth encoder
39             ↪ (error: %d).\n", err);
40         return err;
41     }
42     err = laser_init(28);
43     if (err){
44         printk("Error while configuring laser (error:
45             ↪ %d).\n", err);
46     }
47 }
```

```
39         return err;
40     }
41     set_average_counter(1);
42     laser_set(1);
43
44     if(MOTOR_TEST){
45         k_sleep(K_FOREVER);
46     }
47
48     return err;
49 }
```

## search.c

```
search.c
1  #include "search.h"
2
3  #define MY_STACK_SIZE 500
4  #define MY_PRIORITY 7
5  #define MAX_READINGS 270
6  #define AZIMUTH_DEGREES 180
7  #define ELEVATION_DEGREES 90
8
9  extern struct k_sem my_sem;
10 matrix_x3 readings[MAX_READINGS];
11 matrix_x3 azimuth_readings[AZIMUTH_DEGREES];
12 matrix_x3 elevation_readings[ELEVATION_DEGREES];
13
14
15 uint32_t *azimuth_thread_servo_angle;
16 uint32_t *elevation_thread_servo_angle;
17
18
19
20 extern void azimuth_servo_thread(uint32_t
   ↪ *azimuth_thread_servo_angle){
21     while(1){
22         angle_slow_move(0,*azimuth_thread_servo_angle);
23         angle_move_servo(2,85);
24     }
25 }
26
27 extern void elevation_servo_thread(uint32_t
   ↪ *elevation_thread_servo_angle){
28     while(1){
29         angle_slow_move(1,*elevation_thread_servo_angle);
30         angle_move_servo(2,180);
31     }
32 }
33
34
35 K_THREAD_DEFINE(my_tid_1, MY_STACK_SIZE,
36                 elevation_servo_thread,
37                 ↪ &elevation_thread_servo_angle, NULL, NULL,
38                 MY_PRIORITY, 0, K_TICKS_FOREVER);
39
40 K_THREAD_DEFINE(my_tid_0, MY_STACK_SIZE,
41                 azimuth_servo_thread,
42                 ↪ &azimuth_thread_servo_angle, NULL, NULL,
```



```

41         MY_PRIORITY, 0, K_TICKS_FOREVER);
42
43
44 void validate_servo_zero_moved(int N, uint32_t
45     ↪ zero_point_servo_angle){
46     uint32_t servo_angle;
47     if(N == 0){servo_angle = get_servo_angle(N) - 45;}
48     else if(N == 1){servo_angle = get_servo_angle(N) - 130;}
49
50     if(zero_point_servo_angle == servo_angle){
51         printk("Robot moved to zero-point\n.");
52         return;}
53     else{
54         k_msleep(1500);
55         validate_servo_zero_moved(N, zero_point_servo_angle);}
56 }
57
58 zeros fine_search(zeros enc_values){
59     zeros fine_zeros;
60     if(FINE_ACTIVATE){
61         if(SEARCH_AZIMUTH){
62             k_msleep(2000);
63             init_encoder_azimuth();
64             printk("Starting fine search in Azimuth.\n");
65             k_thread_resume(my_tid_0);
66             fine_zeros.azimuth =
67                 ↪ fine_sweeper(0,10,10,20,enc_values.azimuth);
68             printk("Moving robot to zero-point, encoder value:
69                 ↪ %d.\n", fine_zeros.azimuth);
70             azimuth_thread_servo_angle = fine_zeros.azimuth;
71             validate_servo_zero_moved(0,
72                 ↪ azimuth_thread_servo_angle);
73             k_thread_suspend(my_tid_0);
74         }
75         if(SEARCH_ELEVATION){
76             k_msleep(1000);
77             k_msleep(2000);
78
79             printk("Starting fine search in Elevation.\n");
80             init_encoder_elevation();
81             k_thread_resume(my_tid_1);
82             fine_zeros.elevation = fine_sweeper(1, 10, 10, 20,
83                 ↪ enc_values.elevation);
84             printk("Moving robot to zero-point, encoder value:
85                 ↪ %d.\n", fine_zeros.elevation);
86             elevation_thread_servo_angle = fine_zeros.elevation;
87             validate_servo_zero_moved(1,
88                 ↪ elevation_thread_servo_angle);

```

```

82         k_thread_suspend(my_tid_1);
83
84         k_msleep(1000);
85         angle_move_servo(2,85);
86         k_msleep(2000);
87     }
88     printk("Fine search finished.\n");
89     return fine_zeros;
90 }
91     return enc_values;
92 }
93
94
95 zeros coarse_search(){
96
97     zeros coarse_zeros;
98     coarse_zeros.azimuth = 0;
99     coarse_zeros.elevation = 0;
100     int zero_point_index_azimuth, zero_point_index_elevation;
101     int16_t min_encoder_search_azimuth = 0;
102     int16_t max_encoder_search_azimuth = 180;
103     int16_t min_encoder_search_elevation = 0;
104     int16_t max_encoder_search_elevation = 50;
105     int increment = 1;
106     int16_t size = (max_encoder_search_azimuth -
107     ↪ min_encoder_search_azimuth) / increment;
107     k_thread_start(my_tid_0);
108     k_thread_start(my_tid_1);
109     k_thread_suspend(my_tid_0);
110     k_thread_suspend(my_tid_1);
111
112     printk("Starting coarse sweep in Azimuth.\n");
113     if(SEARCH_AZIMUTH){
114         set_average_counter(1);
115         init_encoder_azimuth();
116         k_thread_resume(my_tid_0);
117         sweep_search(0, min_encoder_search_azimuth,
118     ↪ max_encoder_search_azimuth,increment);
118         get_readings(&azimuth_readings, &size);
119         zero_point_index_azimuth =
120     ↪ find_zero_point(azimuth_readings, size);
120         coarse_zeros.azimuth =
121     ↪ azimuth_readings[zero_point_index_azimuth].encoder;
121         printk("Moving robot to zero point, encoder value:
122     ↪ %d.\n", coarse_zeros.azimuth);
122         azimuth_thread_servo_angle = coarse_zeros.azimuth;
123         validate_servo_zero_moved(0, azimuth_thread_servo_angle);

```

```

124     k_thread_suspend(my_tid_0);
125     printk("Coarse sweep in Azimuth finished\n");
126 }
127
128
129 // for(int i = 0; i < size; i++){
130 //     printk("Encoder: %d, delta: %d, zigma: %d
131 //     ↪ \n", azimuth_readings[i].encoder,
132 //     ↪ azimuth_readings[i].delta,
133 //     ↪ azimuth_readings[i].zigma);
134 // }
135
136 if (SEARCH_ELEVATION){
137     k_msleep(1000);
138     k_msleep(2000);
139
140     init_encoder_elevation();
141     set_average_counter(1);
142     size = (max_encoder_search_elevation -
143     ↪ min_encoder_search_elevation) / increment;
144
145     printk("Starting coarse sweep in Elevation\n");
146     k_thread_resume(my_tid_1);
147     sweep_search(1, min_encoder_search_elevation,
148     ↪ max_encoder_search_elevation, increment);
149     get_readings(&elevation_readings, &size);
150     zero_point_index_elevation =
151     ↪ find_zero_point(elevation_readings, size);
152     coarse_zeros.elevation = elevation_readings[zero_point_in]
153     ↪ dex_elevation].encoder;
154     printk("Moving robot to zero point, encoder value:
155     ↪ %d.\n", coarse_zeros.elevation);
156     elevation_thread_servo_angle = coarse_zeros.elevation;
157     validate_servo_zero_moved(1,
158     ↪ elevation_thread_servo_angle);
159     k_thread_suspend(my_tid_1);
160
161     printk("Coarse sweep in Elevation done\n");
162
163     // for(int i = 0; i < size; i++){
164     //     printk("%d,%d,%d \n",
165     //     ↪ elevation_readings[i].encoder,
166     //     ↪ elevation_readings[i].zigma,
167     //     ↪ elevation_readings[i].delta);
168     // }
169
170

```

```

159     k_msleep(1000);
160 }
161
162
163     printk("Coarse search finished.\n");
164
165     return coarse_zeros;
166 }
167
168
169
170 void sweep_search(int state, int16_t min_encoder_search, int16_t
↪ max_encoder_search, int increment){
171     k_msleep(1000);
172
173     int16_t index = 0;
174     matrix_x3 buffer_data;
175
176     for (int i = min_encoder_search; i < max_encoder_search; i+=
↪ increment){
177         if(state){elevation_thread_servo_angle = (uint32_t) i;}
178         else{azimuth_thread_servo_angle = (uint32_t) i;}
179         set_observer(true);
180         k_sem_take(&my_sem, K_FOREVER);
181
182         get_data(&buffer_data, state);
183         readings[index].encoder = buffer_data.encoder;
184         readings[index].delta = buffer_data.delta;
185         readings[index].zigma = buffer_data.zigma;
186
187         printk("Encoder: %d, Zigma: %d, Delta:
↪ %d\n",readings[index].encoder, readings[index].zigma,
↪ readings[index].delta);
188         index+=1;
189     }
190     k_sem_give(&my_sem);
191 }
192
193 int16_t fine_sweeper(int state, int threshold_degrees, int
↪ threshold_search, int sweep_sector, int16_t zero_point){
194
195
196     matrix_x3 temp_data[sweep_sector + 1];
197     int increment = 1;
198     int16_t zero_point_index;
199     int16_t min_encoder_value = zero_point - (sweep_sector/2);
200     int16_t max_encoder_value = min_encoder_value + sweep_sector;

```

```

201     int16_t size = max_encoder_value - min_encoder_value;
202     bool lower_cap = true;
203     bool higher_cap = true;
204
205     k_msleep(1000);
206     set_average_counter(10);
207     angle_slow_move(state, min_encoder_value);
208     sweep_search(state, min_encoder_value, max_encoder_value,
209     ↪ increment);
210     get_readings(&temp_data, &size);
211     zero_point_index = find_zero_point(temp_data, size);
212
213     for (int i = 0; i < sweep_sector/2; i++){
214
215         if(lower_cap && temp_data[zero_point_index-i].delta <=
216         ↪ temp_data[zero_point_index].delta+threshold_degrees
217         ↪ && (zero_point_index-i) > 0){
218             min_encoder_value =
219             ↪ temp_data[zero_point_index-i].encoder;
220         }
221         else{lower_cap = false;}
222
223         if(higher_cap && temp_data[zero_point_index+i].delta <=
224         ↪ temp_data[zero_point_index].delta+threshold_degrees
225         ↪ && (zero_point_index + i) < size){
226             max_encoder_value =
227             ↪ temp_data[zero_point_index+i].encoder;
228         }
229         else{higher_cap = false;}
230     }
231
232     sweep_sector = max_encoder_value - min_encoder_value;
233
234     if(sweep_sector > threshold_search){
235         fine_sweeper(state, threshold_degrees/2,
236         ↪ threshold_search, sweep_sector,
237         ↪ temp_data[zero_point_index].encoder);
238     }
239     else{
240         angle_slow_move(state, temp_data[zero_point_index].encoder,
241         ↪ );
242         k_msleep(2000);
243         return temp_data[zero_point_index].encoder;
244     }
245 }
246
247

```

```
238 int get_readings(matrix_x3 *main_readings, int16_t *n){
239
240     value_validator(&readings, n);
241     update_matrix(&readings, n);
242     for(int i = 0; i < *n; i++){
243         main_readings[i].encoder = readings[i].encoder;
244         main_readings[i].delta = readings[i].delta;
245         main_readings[i].zigma = readings[i].zigma;
246     }
247     reset_readings();
248     return 0;
249 }
250
251 void reset_readings(){
252     for(int i = 0; i < 270; i++){
253         memset(&readings[i], 0, sizeof readings[i]);
254     }
255 }
```

## data\_processor.c

```
data_processor.c
1  #include "data_processor.h"
2
3  int16_t average_counter = 1;
4  int16_t data_delta[10];
5  int16_t data_zigma[10];
6
7  void send_data(int16_t rssi, int index, int state){
8      if(state == 1){data_zigma[index] = rssi;}
9      else{data_delta[index] = rssi;}
10 }
11
12 int16_t get_average(int16_t *list){
13     int16_t average = 0;
14     for(int i = 0; i < average_counter; i++){
15         average += list[i];
16     }
17     return average/average_counter;
18 }
19
20 void set_average_counter(int16_t value){
21     if(value > 10){value = 10;}
22     else if(value < 1){value = 1;}
23     average_counter = value;
24 }
25
26
27 void get_data(matrix_x3 *buffer_data, int N){
28
29     buffer_data->encoder = get_encoder(N);
30     buffer_data->delta = get_average(data_delta);
31     buffer_data->zigma = get_average(data_zigma);
32
33 }
34
35 void value_validator(matrix_x3 *raw_data, int16_t *n){
36
37     for(int i = 0; i < *n; i++){
38         if(raw_data[i].delta > MAX_VALID_RSSI ||
39            ↪ raw_data[i].delta < MIN_VALID_RSSI
40            || raw_data[i].zigma > MAX_VALID_RSSI ||
41            ↪ raw_data[i].zigma < MIN_VALID_RSSI){
42             raw_data[i].encoder = 0;
43             raw_data[i].delta = 0;
44             raw_data[i].zigma = 0;
45         }
46     }
47 }
```

```

43     printk("Changed values at: %d", raw_data[i].encoder);
44     }
45     }
46 }
47
48 void update_matrix(matrix_x3 *data, int16_t *n){
49
50     for(int i = 0; i < *n; i++){
51         if(data[i].delta == 0 || data[i].zigma == 0){
52             printk("Removed values at: %d", data[i].encoder);
53             for(int pos = i; pos < *n-1; pos++){
54                 data[pos].encoder = data[pos+1].encoder;
55                 data[pos].delta = data[pos+1].delta;
56                 data[pos].zigma = data[pos+1].zigma;
57             }
58             *n -= 1;
59         }
60     }
61 }
62
63 bool zero_point_validator(int16_t value_zigma, int16_t
↪ value_delta, int16_t ZIGMA_ZERO_VALUE){
64
65     return value_zigma >= ZIGMA_ZERO_VALUE && value_delta <
↪ value_zigma;
66 }
67
68
69 int find_zero_point(matrix_x3 validated_values[], int n){
70     int zero_point_index = 0;
71     bool first_zero = false;
72     int16_t ZIGMA_ZERO_VALUE;
73
74     if(n == 1){return 0;}
75
76     ZIGMA_ZERO_VALUE = find_zigma_zero_value(validated_values, n);
77
78     for(int i = 0; i < n-1; i++){
79         if(!first_zero && zero_point_validator(validated_values[i]
↪
↪ ].zigma,validated_values[i].delta,
↪ ZIGMA_ZERO_VALUE)){
80             zero_point_index = i;
81             first_zero = true;
82             printk("First zero at index %d\n", i);
83         }
84     }
85

```



```

86     for(int i = (zero_point_index + 1); i < n-1; i++){
87         if(validated_values[i].delta <=
            ↪ validated_values[zero_point_index].delta &&
            ↪ zero_point_validator(validated_values[i].sigma,
            ↪ validated_values[i].delta, ZIGMA_ZERO_VALUE)){
88             zero_point_index = i;
89         }
90     }
91     printk("encoder: %d, delta %d, zigma: %d\n",
            ↪ validated_values[zero_point_index].encoder,
            ↪ validated_values[zero_point_index].delta,
            ↪ validated_values[zero_point_index].sigma);
92     return zero_point_index;
93 }
94 }
95
96 int16_t find_zigma_zero_value(matrix_x3 values[], int n){
97     int16_t ZIGMA_ZERO_VALUE = -90;
98     for (int i = 1; i < n-1; i++){
99         if(values[i].sigma > ZIGMA_ZERO_VALUE){ZIGMA_ZERO_VALUE =
            ↪ values[i].sigma;}
100     }
101     printk("New zigma zero value validator:
            ↪ %d.\n",ZIGMA_ZERO_VALUE);
102     return ZIGMA_ZERO_VALUE - 3;
103 }

```

## observer.c

```
observer.c
1  #include "observer.h"
2  #define switch_pin 29
3
4  extern int16_t average_counter;
5
6
7  extern struct k_sem my_sem;
8  const struct device *dev;
9  bool send_data_state = false;
10
11 void set_observer(bool state){
12     send_data_state = state;
13 }
14
15
16 static void device_found(const bt_addr_le_t *addr, int8_t rssi,
17     ↪ uint8_t type,
18     struct net_buf_simple *ad)
19 {
20     if(send_data_state){
21         static int counter = 0;
22         static int state = 0; // 0 = delta, 1 = zigma
23         if(state == 0){
24             send_data(rssi, counter, state);
25             counter +=1;
26             if(counter >= average_counter){
27                 counter = 0;
28                 state = 1;
29                 gpio_pin_set(dev, switch_pin, 0);
30             }
31         }
32         else if(state == 1){
33             send_data(rssi, counter, state);
34             counter += 1;
35             if(counter >= average_counter){
36                 state = 0;
37                 counter = 0;
38                 gpio_pin_set(dev, switch_pin, 1);
39                 set_observer(false);
40                 k_sem_give(&my_sem);
41             }
42         }
43     }
```

```

44     }
45     // else{printk("rssi: %d\n", KALMAN(rssi));}
46 }
47
48
49 int add_filter_accept_list_from_string(const char *addr_str, const
↳ char *type){
50     int err;
51     bt_addr_le_t addr_le = {.a = BT_ADDR_LE_ANY, .type =
↳ BT_ADDR_LE_RANDOM};
52     err = bt_addr_le_from_str(addr_str, type, &addr_le);
53     if (err){
54         printk("Could not convert string to bluetooth
↳ address (error: %d).\n", err);
55         return err;
56     }
57     err = bt_le_filter_accept_list_add(&addr_le);
58     if (err){
59         printk("Could not add address to acceptlist
↳ (error: %d).\n", err);
60         return err;
61     }
62     return err;
63 }
64
65
66 int init_bluetooth_scan(){
67
68     struct bt_le_scan_param scan_param = {
69         .type      = BT_LE_SCAN_TYPE_ACTIVE,
70         .options   = BT_LE_SCAN_OPT_FILTER_ACCEPT_LIST,
71         .interval  = BT_GAP_SCAN_FAST_INTERVAL,
72         .window    = BT_GAP_SCAN_FAST_WINDOW,
73     };
74
75     dev = device_get_binding("GPIO_0");
76     if (dev == NULL) {
77         return;
78     }
79     int err;
80     printk("Starting Observer.\n");
81     /* Initialize the Bluetooth Subsystem */
82     err = bt_enable(NULL);
83     if (err) {
84         printk("Bluetooth init failed (error %d).\n",
↳ err);
85         return err;

```

```

86     }
87     err = add_filter_accept_list_from_string("E7:9E:11:9E:33:EE",
      ↪ "(random)");
88     if (err){
89         printk("Could not add to acceptlist (error:
      ↪ %d).", err);
90         return err;
91     }
92     printk("Bluetooth initialized.\n");
93
94     err = bt_le_scan_start(&scan_param, device_found);
95     if (err) {
96         printk("Starting scanning failed (err %d).\n",
      ↪ err);
97         return err;
98     }
99     err = gpio_pin_configure(dev, switch_pin,
      ↪ GPIO_OUTPUT_ACTIVE | GPIO_ACTIVE_LOW);
100    if (err) {
101        printk("Could not configure pin for switch (err
      ↪ %d).\n", err);
102        return err;
103    }
104    printk("Initializing switch on pin %d success!\n",
      ↪ switch_pin);
105
106    gpio_pin_set(dev, switch_pin, 1);
107    k_sem_give(&my_sem);
108    return err;
109 }

```

## encoder.c

```
encoder.c
1  #include "encoder.h"
2
3
4  int16_t azimuth_encoder_value = 0;
5  int16_t elevation_encoder_value = 0;
6  int16_t azimuth_encoder_degrees;
7  int16_t elevation_encoder_degrees;
8  extern struct k_sem my_sem;
9
10 static void qdec_nrfx_event_handler_azimuth(nrfx_qdec_event_t
    ↪ event){}
11 static void qdec_nrfx_event_handler_elevation(nrfx_qdec_event_t
    ↪ event){}
12
13
14 int init_encoder_azimuth(){
15
16     int err;
17     nrfx_qdec_disable();
18     nrfx_qdec_uninit();
19
20
21     nrfx_qdec_config_t qdec_config_azimuth =
22     {
23         .reportper           = NRF_QDEC_REPORTPER_DISABLED,
24         .sampleper           = NRF_QDEC_SAMPLEPER_128us,
25         .psela                = pin_a_azimuth,
26         .pselb                = pin_b_azimuth,
27         .pselled              = NRF_QDEC_LED_NOT_CONNECTED,
28         .dbfen                = NRF_QDEC_DBFEN_DISABLE,
29         .sample_inten        = false,
30         .interrupt_priority   =
    ↪ NRFX_QDEC_DEFAULT_CONFIG_IRQ_PRIORITY
31     };
32     err = nrfx_qdec_init(&qdec_config_azimuth,
    ↪ qdec_nrfx_event_handler_azimuth);
33     if (err == NRFX_SUCCESS) {
34         printk("Azimuth QDEC initialization was
    ↪ successful.\n");
35         nrfx_qdec_enable();
36         return err;
37     }
38     else if (err == NRFX_ERROR_INVALID_STATE) {
```

```

40     printk("Azimuth QDEC was already initialized.\n");
41     return err;
42 }
43     else {printk("Azimuth QDEC initialization failed.\n");}
44
45     return err;
46 }
47
48
49 int init_encoder_elevation(){
50
51     int err;
52     nrfx_qdec_disable();
53     nrfx_qdec_uninit();
54
55
56     nrfx_qdec_config_t qdec_config_elevation =
57     {
58         .reportper           = NRF_QDEC_REPORTPER_DISABLED,
59         .sampleper           = NRF_QDEC_SAMPLEPER_128us,
60         .psela                = pin_a_elevation,
61         .pselb                = pin_b_elevation,
62         .pselled              = NRF_QDEC_LED_NOT_CONNECTED,
63         .dbfen                = NRF_QDEC_DBFEN_DISABLE,
64         .sample_inten         = false,
65         .interrupt_priority =
66         ↪ NRFX_QDEC_DEFAULT_CONFIG_IRQ_PRIORITY
67     };
68
69     err = nrfx_qdec_init(&qdec_config_elevation,
70     ↪ qdec_nrfx_event_handler_elevation);
71     if (err == NRFX_SUCCESS) {
72         printk("Elevation QDEC initialization was
73         ↪ successful.\n");
74         nrfx_qdec_enable();
75         return err;
76     }
77     else if (err == NRFX_ERROR_INVALID_STATE) {
78         printk("Elevation QDEC was already initialized.\n");
79         return err;
80     }
81     else {printk("Elevation QDEC initialization failed.\n");}
82
83     return err;
84 }

```

```

84
85 int init_encoder_servos(){
86     int err;
87     err = servo_init(servo_azimuth_N, servo_azimuth_pin);
88     if (err){
89         printk("Could not Azimuth servomotor
90             ↪ (err:%d).\n", err);
91         return err;
92     }
93     err = servo_init(servo_elevationl_N, servo_elevation_pin);
94     if (err){
95         printk("Could not Elevation servomotor
96             ↪ (err:%d).\n", err);
97         return err;
98     }
99     err = servo_init(servo_antenna_N, servo_antenna_pin);
100    if (err){
101        printk("Could not configure Antenna servomotor
102            ↪ (err:%d).\n", err);
103        return err;
104    }
105
106    angle_move_servo(servo_azimuth_N, starting_angle_azimuth);
107    angle_move_servo(servo_elevationl_N, starting_angle_elevation);
108    angle_move_servo(servo_antenna_N, 90);
109
110    k_msleep(1000);
111    azimuth_encoder_degrees = starting_angle_azimuth;
112    elevation_encoder_degrees = starting_angle_elevation;
113    azimuth_encoder_value = azimuth_encoder_value * 23;
114    elevation_encoder_value = elevation_encoder_degrees * 23;
115
116    IRQ_CONNECT(QDEC_IRQn, 4, nrfx_isr, nrfx_qdec_irq_handler, 0);
117    irq_enable(QDEC_IRQn);
118
119 }
120
121 void update_encoder(int N){
122     int16_t acc;
123     int16_t accdbl;
124     nrfx_qdec_accumulators_read(&acc, &accdbl);
125     if (N == 0){
126         azimuth_encoder_value -= acc;
127         azimuth_encoder_degrees = azimuth_encoder_value/23;
128     }
129     else if (N ==1){

```

```

128     elevation_encoder_value += acc;
129     elevation_encoder_degrees = elevation_encoder_value/23;
130 }
131 else{
132     printk("Error, wrong encoder number.\n");
133     return;
134 }
135 }
136
137
138 void angle_slow_move(int N, uint32_t angle){
139     int size = 0;
140
141     if(N == 0){
142         angle += 45;
143         if (angle >= 225){angle = 225;}
144         else if(angle <= 45){angle = 45;}
145         size = angle-get_servo_angle(N);
146
147     }
148     else if(N == 1){
149         angle += 130;
150         if (angle >= 200){angle = 200;}
151         else if(angle <= 130){angle = 130;}
152         size = angle - get_servo_angle(N);
153
154     }
155     if(size > 0){
156         for(int i = 0; i < size; i++){
157             increment_servo(N);
158             k_msleep(60);
159             update_encoder(N);
160         }
161     }
162     else if(size < 0){
163         for(int i = 0; i > size; i--){
164             decrement_servo(N);
165             k_msleep(60);
166             update_encoder(N);
167         }
168     }
169 }
170 int16_t get_encoder(int N){
171     if(N < 1){
172         return azimuth_encoder_degrees;
173     }
174     else{

```



```
175     return elevation_encoder_degrees;
176 }
177 }
```

## servo.c

```
servo.c
1  #include "servo.h"
2
3
4  //Set servo period in Hz
5  #define SERVO_PERIOD 50
6
7  //Set timer settings
8  #define CLOCK_SPEED 16000000
9  #define TIMER_RELOAD 320000
10 // #define TIMER_RELOAD (CLOCK_SPEED/SERVO_PERIOD)
11
12 #define PWMN_GPIOTE_CH    {0, 1, 2, 3}
13 #define PWMN_PPI_CH_A    {0, 2, 3, 4}
14 #define PWMN_PPI_CH_B    {1, 1, 5, 5}
15 #define PWMN_TIMER_CC_NUM {0, 1, 2, 3}
16
17 static uint32_t pwmN_gpiote_ch[] = PWMN_GPIOTE_CH;
18 static uint32_t pwmN_ppi_ch_a[] = PWMN_PPI_CH_A;
19 static uint32_t pwmN_ppi_ch_b[] = PWMN_PPI_CH_B;
20 static uint32_t pwmN_timer_cc_num[] = PWMN_TIMER_CC_NUM;
21
22
23 // Timer CC register use to reset the timer.
24 #define TIMER_RELOAD_CC_NUM 5
25
26 int timer_init()
27 {
28     NRF_TIMER3->BITMODE =
29     ↪ TIMER_BITMODE_BITMODE_24Bit << TIMER_BITMODE_BITMODE_Pos;
30     NRF_TIMER3->PRESCALER = 0;
31     NRF_TIMER3->SHORTS =
32     ↪ TIMER_SHORTS_COMPARE0_CLEAR_Msk << TIMER_RELOAD_CC_NUM;
33     NRF_TIMER3->MODE = TIMER_MODE_MODE_Timer
34     ↪ << TIMER_MODE_MODE_Pos;
35     NRF_TIMER3->CC[TIMER_RELOAD_CC_NUM] = TIMER_RELOAD;
36
37     printk("Timer initialized.\n");
38     return 0 ;
39 }
40
41 int timer_start()
42 {
43     NRF_TIMER3->TASKS_START = 1;
44     printk("Timer started.\n");
45 }
```

```

42     return 0;
43 }
44
45 uint32_t azimuth_servo_angle;
46 uint32_t horizontal_servo_angle;
47
48 int servo_init(uint32_t N, int servo_pin)
49 {
50     if(N>3) {
51         return 1;
52     }
53
54     NRF_GPIOTE->CONFIG[pwmN_gpiote_ch[N]] =
55         GPIOTE_CONFIG_MODE_Task << GPIOTE_CONFIG_MODE_Pos |
56         GPIOTE_CONFIG_POLARIT
57         ↪ Y_Toggle <<
58         ↪ GPIOTE_CONFIG_POL
59         ↪ ARITY_Pos
60         ↪ |
61         ↪ servo_pin << GPIOTE_C
62         ↪ ONFIG_PSEL_Pos
63         ↪ |
64         ↪ GPIOTE_CONFIG_OUTINIT
65         ↪ _High <<
66         ↪ GPIOTE_CONFIG_OUT
67         ↪ INIT_Pos;
68
69     NRF_PPI->CH[pwmN_ppi_ch_a[N]].EEP = (uint32_t)&NRF_TIMER3->EV
70     ↪ ENTS_COMPARE[pwmN_timer_cc_num[N]];
71     NRF_PPI->CH[pwmN_ppi_ch_a[N]].TEP =
72     ↪ (uint32_t)&NRF_GPIOTE->TASKS_CLR[pwmN_gpiote_ch[N]];
73     if((N%2) == 0) {
74         NRF_PPI->CH[pwmN_ppi_ch_b[N]].EEP = (uint32_t)&NRF_TIMER3
75         ↪ ->EVENTS_COMPARE[TIMER_RELOAD_CC_NUM];
76         NRF_PPI->CH[pwmN_ppi_ch_b[N]].TEP =
77         ↪ (uint32_t)&NRF_GPIOTE->TASKS_SET[pwmN_gpiote_ch[N]];
78     } else {
79         NRF_PPI->FORK[pwmN_ppi_ch_b[N-1]].TEP =
80         ↪ (uint32_t)&NRF_GPIOTE->TASKS_SET[pwmN_gpiote_ch[N]];
81     }
82     NRF_PPI->CHENSET
83     ↪ = (1 <<
84     ↪ pwmN_ppi_ch_a[N]) | (1 << pwmN_ppi_ch_b[N]);
85
86     printk("Initializing servo %u on pin %i success!\n", N,
87     ↪ servo_pin);
88     return 0;
89 }

```

```

72
73 uint32_t convert_to_raw(int N, uint32_t value)
74 {
75     uint32_t angle = 0;
76     if (N < 2){angle = (value * 32000)/270 + 8000;}
77     else {angle = (value * 32000)/180 + 8000;}
78
79     return angle;
80 }
81
82
83 void raw_move_servo(int N, uint32_t position)
84 {
85     if (N > 3)
86     {
87         printk("Invalid N, %u > 3\n", N);
88         return;
89     }
90     position = (position <= 0) ? 1 : (position >= TIMER_RELOAD) ?
    ↪ TIMER_RELOAD - 1 : position;
91
92     NRF_TIMER3->CC[pwmN_timer_cc_num[N]] = position;
93 }
94
95 void angle_move_servo(int N, uint32_t angle)
96 {
97     if(N == 0){
98         angle += 45;
99         if (angle >= 225){angle = 225;}
100        else if(angle <= 45){angle = 45;}
101        azimuth_servo_angle = angle;
102    }
103
104    else if(N == 1){
105        angle += 130;
106        if (angle >= 200){angle = 200;}
107        else if(angle <= 130){angle = 130;}
108        horizontal_servo_angle = angle;
109    }
110
111    else if(N == 2){
112        if (angle >= 180){angle = 180;}
113        else if(angle <= 89){angle = 89;}
114    }
115
116
117    angle = convert_to_raw(N, angle);

```

```

118     raw_move_servo(N, angle);
119
120 }
121
122 void increment_servo(int N){
123     if(N == 0){
124         azimuth_servo_angle += 1;
125         raw_move_servo(N,convert_to_raw(N,azimuth_servo_angle));
126     }
127     else if(N == 1){
128         horizontal_servo_angle +=1;
129         raw_move_servo(N,
130             ↪ convert_to_raw(N,horizontal_servo_angle));
131     }
132 }
133
134 void decrement_servo(int N){
135     if(N == 0){
136         azimuth_servo_angle -= 1;
137         raw_move_servo(N,convert_to_raw(N,azimuth_servo_angle));
138     }
139     else if(N == 1){
140         horizontal_servo_angle -= 1;
141         raw_move_servo(N,
142             ↪ convert_to_raw(N,horizontal_servo_angle));
143     }
144 }
145
146 int16_t get_servo_angle(int N){
147     if(N == 0){
148         // printk("Azimuth servomotor angle: %d\n",
149             ↪ azimuth_servo_angle);
150         return azimuth_servo_angle - 45;}
151     else if(N == 1){
152         // printk("Horizontal angle: %d\n",
153             ↪ horizontal_servo_angle);
154         return horizontal_servo_angle - 130;}
155 }

```

## laser.c

```
laser.c
1  #include "laser.h"
2
3  static int _laser_pin;
4  extern const struct device *dev;
5
6  int laser_init(int laser_pin)
7  {
8
9      int err;
10     err = gpio_pin_configure(dev, laser_pin, GPIO_OUTPUT_HIGH);
11     if (err){
12         printk("Could not configure laser (err: %d).",
13             ↪ err);
14         return err;
15     }
16     _laser_pin = laser_pin;
17
18     printk("Laser init successfull.\n");
19     return err;
20 }
21
22 void laser_toggle()
23 {
24     gpio_pin_toggle(dev, _laser_pin);
25 }
26
27 void laser_set(int state)
28 {
29     gpio_pin_set(dev, _laser_pin, state);
30 }
```



