AUTOMATISERAD MÄTPLATTFORM FÖR ANTENNSTRÅLNING I TRE DIMENSIONER

Högskoleingenjörsutbildning i Maskinteknik
Automation & AI

Albin Svantesson
Oskar Larsson
Sebastian Johansson

Program: Maskiningenjör – Automation & AI
Svensk titel: Automatiserad mätplattform för antennstrålning i tre dimensioner
Engelsk titel: Automated measurement platform for antenna radiation in three dimensions
Utgivningsår: 2025
Författare: Albin Svantesson, Oskar Larsson, Sebastian Johansson
Handledare Högskola: Ali Davoodi
Handledare Företag: Peter Ankarson
Examinator: Michael Tittus
Nyckelord: EMC, Högfrekvent Strålning, Antenn, Vridbord, Stegmotor, Arduino, Spektrumanalysator, Python, C++, Signalgenerator, 3D – printing, CAD

Sammanfattning

Examensarbetet syftar till att utveckla en automatiserad mätplattform för analys av antenners strålningsmönster i tre dimensioner, med särskilt fokus på tillämpningar inom elektromagnetsik kompabilitet (EMC). Arbetet genomfördes i samarbete med RISE (Research Institutes of Sweden) och adresserar ett behov av att utöka befintliga mätmöjligheter från enbart horisontell till fullständig tredimensionell karakterisering av högfrekventa antenner.

En komplett lösning som kombinerar mekanisk konstruktion, elektronik och programmering har tagits fram. En kompakt och justerbar mätplattform konstruerades med hjälp av CAD och tillverkades huvudsakligen genom 3D-printing. Systemet styrs via en Arduino-plattform och kommunicerar med en datorapplikation, utvecklad i Python, som hanterar mätsekvenser, datainsamling, samt interaktion med en spektrumanalysator.

Plattformens mekaniska delar inkluderar ett vridbord med två roterande axlar (azimuth och roll) som möjliggör noggranna positioneringar med en upplösning på minst 0,06 grader. Ett flertal iterationer genomfördes under utvecklingsprocessen för att optimera funktionalitet, hållfastighet och användarvänlighet. Det färdiga systemet har integrerats i en heldämpad kammare för att möjliggöra precisa och reproducerbara mätningar av antenners strålningsdiagram inom frekvensområdet 5,8-40 GHz.

Resultatet visar att det är möjligt att med begränsade resurser skapa en flexibel, kostnadseffektiv och användarvänlig mätplattform för avancerade EMC-applikationer. Arbetet belyser även hur modern prototyptillverkning och agil utvecklingsmetodik kan användas för att snabbt realisera tekniskt komplexa system.

Abstract

This thesis aims to develop an automated measurement platform for analyzing the three-dimensional radiation patterns of antennas, with a particular focus on applications within electromagnetic compatibility (EMC). Conducted in collaboration with RISE, the work addresses the need to extend current measurement capabilities from solely horizontal characterization to full three-dimensional analysis of high-frequency antennas.

A complete solution was developed by combining mechanical design, electronics, and software. The resulting measurement platform is compact, adjustable, and was created using CAD, with most components produced through 3D printing. The system is controlled via an Arduino-based platform and communicates with a Python-based computer application that manages measurement sequences, data acquisition, and interaction with a spectrum analyzer.

The platform's mechanical components include a turntable with two rotational axes (azimuth and roll), allowing for precise positioning with a resolution of at least 0.06 degrees. Multiple design iterations were conducted to optimize functionality, structural integrity, and user experience. The finalized system has been integrated into an anechoic chamber, enabling accurate and repeatable measurements of antenna radiation patterns in the frequency range of 5.8–40 GHz.

Results demonstrate that a flexible, cost-effective, and user-friendly measurement solution for advanced EMC applications can be realized with limited resources. The project also highlights how modern prototyping techniques and agile development methods can accelerate the realization of technically complex systems.

Förord

Denna kandidatuppsats har genomförts vid forskningsinstitutet RISE (Research Institutes of Sweden) i Borås, under vårterminen 2025. Vi vill rikta ett varmt tack till avdelningen för elektromagnetisk kompatibilitet (EMC) för att ha tagit emot oss och försett oss med värdefull handledning, givande diskussioner och ett kontinuerligt stöd genom hela arbetet.

Tiden på RISE har varit både lärorik och inspirerande. Vi har fått inblick i nya arbetsområden som skiljer sig från tidigare utbildningserfarenheter, vilket har bidragit till vår personliga och professionella utveckling. Utmaningarna vi stött på har gett oss nya perspektiv och stärkt vår förståelse för praktisk tillämpning av teoretiska kunskaper.

Innehållsförteckning

1. INLEDNING 1

1.1 Syfte 1

1.2 Frågeställningar och kravspecifikation på konstruktionen 1

1.3 Avgränsningar 2

2. TEORI 3

2.1 EMC 3

2.2 Antenner 3

2.3 Högfrekvent Strålning 3

2.4 3D-strålningsmätning 3

2.5 Simuleringsmodeller för antenner 4

2.6 3D – Printning 4

2.7 Produktutveckling 4

2.8 Agil systemutveckling 4

3. KOMPONENTER OCH MATERIEL 5

3.1 Materiel 5

3.1.1 Antenner 5

3.1.2 Program & drivkomponenter 5

3.1.3 3D - printning 5

3.1.4 Stödkomponenter och verktyg 5

3.2 Arduino Uno 6

3.3 Motordrivare TB6600 6

3.4 Stegmotor 1.89 Nm 1.8° NEMA 23 6

3.5 Stegmotor 9.6VDC 167Nmm 1.8° NEMA 17 6

3.6 Transformator 12V 6A 72W 6

3.7 Signalgenerator - Rohde & Schwarz SMCV100B 7

3.8 Spektrumanalysator 7

3.9 Heldämpad kammare 7

3.10 Absorbenter 8

3.11 Computer-Aided Design (CAD) – Creo Parametric 8

3.12 Filament (PLA) 8

3.13 Bambu Lab X1E 8

3.14 Original Prusa XL 8

4. METOD 9

4.1 Produktutveckling - Konstruktion 9

4.2 Agil systemutveckling - Programmering 11
4.2.1 Mätningsrutin 11
4.2.2 Kommunikation mellan Arduino och Python 12

4.3 Datainsamling – mätningar 12

5. RESULTAT - PROTOTYPFRAMTAGNING 14

5.1 Konstruktion 14
5.1.1 Hållare azimuth-motor 14
5.1.2 Aluminiumfläns stor och liten 15
5.1.3 Vridbord 15
5.1.4 Antennpinne 18
5.1.5 Hållare roll-motor 20
5.1.6 Roll-motor 20
5.1.7 Lock/baksida för hållare roll-motor 21
5.1.8 Hasp 21
5.1.9 Nollställare Azimuth 22
5.1.10 Antennhållare 23
5.1.11 Sladdlist 27
5.1.12 Elcentral 27
5.1.13 Slutgiltig konstruktion 28

5.2 Kopplingsschema 28

5.3 Programmering 30
5.3.1 Arduino Flödesschema 30
5.3.2 Förklaring av Arduinokod 31
5.3.3 Python Flödesschema 36
5.3.4 Förklaring av Pythonkod 37

5.4 Kalibrering av system 42

6. VALIDERING AV STRÅLNINGSDIAGRAM 43

6.1 Allmänt om strålningsdiagrammen 43

6.2 Validering av den smala gröna antennen 48

6.3 Validering av den breda gröna antennen 49

6.4 Sammanfattning av valideringen 50

7. DISKUSSION 51

7.1 Diskussion av prototyp 51
7.1.1 Materialval 51
7.1.2 3D-skrivare 51
7.1.3 Nollställe för motorerna 53
7.1.4 Framtagning av hasp 53
7.1.5 Vridbord 54
7.1.6 Python 54
7.1.7 Arduino 55

7.2 Diskussion av metod 56

7.3 Praktisk tillämpning av resultat 57

7.4 Förslag på fortsatt utredning 57
7.4.1 Automatiskt nolläge 57
7.4.2 Implementering av AI 57
7.4.3 Skala upp konstruktionen 57

8. SLUTSATS 58

9. REFERENSER 59

9.1 Bildkällor: 59

9.2 Källförteckning: 59

BILAGOR 61

Tabellförteckning

Tabell 1: Dämpningsegenskaper för användna absorbenter, artikelnummer 3640–50 (Holland Shielding 2024). 8
Tabell 2: Exempelvärden för input till interface. 55

Figurförteckning

Figur 1: Heldämpad kammare. Mått: H:119cm, B:148cm, D:89cm 7
Figur 2: Första skissen på konstruktionen. 10
Figur 3: Exempel på vridbord från RISE. 10
Figur 4: Andra skissen på konstruktion 11
Figur 5: Mätuppställning med spektrumanalysator till vänster och signalgenerator till höger kopplade med gröna linjer till respektive antenn. 12
Figur 6: Del av mätdata från en hel mätning med Azimuth: stegstorlek 5, start 0, stopp 90. Roll: steg 10, start 40, stopp 0. 13
Figur 7: Slutlig konstruktion med siffror som indikerar delarna som 3D-printades i konstruktionen. 14
Figur 8: Hållare till azimuth-motor i CAD utan motor till vänster och med motor i till höger. 15
Figur 9: Aluminiumflänsarnas placering på konstruktionen, stora till vänster, CAD-fläns i mitten, liten till höger. 15
Figur 10: Första konceptet på vridbordet ovanifrån. 16
Figur 11: Första koncept på vridbordet underifrån. 16
Figur 12: Simulering på nedböjning med aluminium som material & data för delen. 16
Figur 13: Plan B, ALU-profil som tillverkas på RISE. 17
Figur 14: Slutgiltigt vridbord ovanifrån, röd pil pekar på skruvhål för montering av antennpinne. 17
Figur 15: Slutgiltigt vridbord underifrån. 18
Figur 16: Första koncept antennpinne, ej tvådelad. 18
Figur 17: Den första typen av avrundning på antennpinnes nedre del. 19
Figur 18: Till vänster visas första steget på "Chamfer" som sedan med upprepade steg utvecklades till avrundningen till höger. 19
Figur 19: Slutgiltig tvådelad konstruktion av antennpinne. 20
Figur 20: Hållare roll-motor i CAD utan baksida och med motorn monterad. 20
Figur 21: Roll-motor i CAD med standardlängd på drivpinne till vänster samt längd på avsågad pinne till höger. 21
Figur 22: Lock/baksida i CAD för hållare till roll-motorn. 21
Figur 23: Hållare till roll-motor i CAD med lock på baksidan och hasp. 22
Figur 24: Nollställare visas i färgen den printades i till vänster och till höger ett förtydligande kring vilka väggar som vinklades på insidan/undersidan. 22
Figur 25: Från vänster: hornantenn, grön bred antenn, gul antenn, grön smal antenn. 23
Figur 26: Hela antennhållaren för den gula antennen. 23
Figur 27: Första designen på flänshållaren med hål för montering av antennkabel på framsidan. 24
Figur 28: Utvecklad design på flänshållaren med avkapade sidor. 24
Figur 29: Första designen på universalhållaren med plats för antennkabel på baksidan och ett universalfäste framtill. 25
Figur 30: Utvecklad design på universalhållaren med avkapade sidor. 25
Figur 31: Antennfäste grunddesign. 25
Figur 32: Antennfäste för den gula antennen med bottendel separat till höger. 26

Figur 33: Antennfäste för den gröna breda antennen med bottendel separat till höger. 26
Figur 34: Till vänster visas utvecklad design med nedsänkning på insidan av antennfästet, i mitten syns antenn utan antennkabel, till höger antenn med antennkabel monterad. 26
Figur 35: Antennfäste för hornantennen. 27
Figur 36: Sladdlist från ovan till vänster och underifrån till höger. 27
Figur 37: Elcentralen med tillhörande komponenter monterade i Creo m.h.a Assembly. 28
Figur 38: Hela antennhållaren i isolerad kammare. 28
Figur 39: Kopplingsschema över hela systemet. 29
Figur 40: Flödesschema Arduino. 30
Figur 41: Matematik för arduino. 32
Figur 42: Ökande stegfel visat från CSV-filen, första kolumnens decimaler illustrerar detta. 33
Figur 43: Värden med fungerande felstegsfunktion, första kolumnens decimaler illustrerar detta 33
Figur 44: Stegfelskompensation fall 4. 34
Figur 45: Stegfelskompensation fall 1-3. 34
Figur 46: Stegfelskompensation roll-motor. 35
Figur 47: Flödesschema Python 36
Figur 48: GUI utan värden, innan drivning av system. 38
Figur 49: GUI med värden under mätning. 38
Figur 50: Funktion send_values() 39
Figur 51: Funktion update_progress() 39
Figur 52: Funktion update_plot() 40
Figur 53: Funktion read_serial() 41
Figur 54: Bilden visar uppställning med laser för att validera noggrannheten i rotationen av vridbordet. 42
Figur 55: Input-värden till samtliga mätningar till vänster, början av mätdatan till första mätningen till höger. 44
Figur 56: Strålningsdiagram för smal grön antenn, utan isolering. 90° på azimuthskalan innebär 0° i figuren. Felaktig skalering. 44
Figur 57: Strålningsdiagram smal grön antenn, fullt isolerad. 45
Figur 58: Del av figur 57 ovanifrån sett till vänster, resultat från rektangulär simulering av smal grön antenn till höger. 45
Figur 59: Strålningsdiagram bred grön antenn, utan isolering. 46
Figur 60: Strålningsdiagram bred grön antenn, fullt isolerad. 46
Figur 61: Strålningsdiagram, samma som figur 60 fast ovanifrån till vänster i bild, rektangulär simulering ovanifrån i mitten, 3D-simulering till höger. 47
Figur 62: Utskuren absorbentplatta fäst bakom antenn. 47
Figur 63: Strålningsdiagram för andra mätning av fullt isolerad bred grön antenn, sidovy till vänster, ovanifrån till höger. 47
Figur 64: Vridbordets placering på den mindre Bambu Lab-skrivaren. 52
Figur 65: Vridbordets misslyckade utskrift från Bambu Lab-skrivaren där ytan inte blev som önskat. 52

1. INLEDNING

I takt med att samhället blir alltmer beroende av elektronisk utrustning och trådlös kommunikation, ökar även behovet av att säkerställa att systemen fungerar tillsammans utan störningar. Elektromagnetisk kompatibilitet (EMC) är därför en avgörande faktor i utvecklingen av modern teknik. EMC handlar om att elektrisk och elektronisk utrustning ska kunna samexistera utan att orsaka eller påverkas av oönskad elektromagnetisk strålning (RISE 2024).

Elektromagnetiska störningar kan medföra reducerad systemprestanda, driftavbrott samt, i vissa fall, allvarliga säkerhetsrisker. Därför är det av stor vikt att utveckla och tillämpa metoder för effektiv identifiering, mätning och hantering av sådana störningar, i syfte att säkerställa en robust och störningsfri drift i miljöer där flera system är integrerade och samverkar (RISE 2024).

I nuvarande 5G och kommande 6G telekom- och nätverksplattformar används antenner i allt högre frekvensband, där strålningsdiagrammet för antenner är en viktig parameter för att förstå hur antennens strålning breder ut sig. RISE har en stor heldämpad testkammare för mätning av antenners strålningsdiagram i enbart horisontalplanet, vilket är en begränsning. Problemet är att mätningar i horisontalplanet ej är tillräckligt för att testa antenners strålningsdiagram och därmed roteras antennerna manuellt i nuläget, för att skapa ett strålningsdiagram i flera dimensioner. Genom att automatisera den nuvarande manuella processen för mätningar i flera dimensioner, effektiviseras framtagningen av strålningsdiagram.

1.1 Syfte

Syftet med arbetet är att möjliggöra för RISE att mäta strålning från antenner i alla plan och därigenom skapa förutsättningar för att ta fram strålningsdiagram i tre dimensioner. För detta utvecklas en prototyp av en roterbar antennhållare med hög positionsupplösning, anpassad för mätningar i en heldämpad kammare. Arbetet syftar även till att undersöka hur de uppmätta strålningsdiagrammen kan jämföras och valideras mot ideala simuleringar, vilket skapar en grund för framtida verifiering av nya eller befintliga antenner.

1.2 Frågeställningar och kravspecifikation på konstruktionen

- Hur kan en prototyp för en roterande antennhållare utvecklas, som möjliggör strålningsmätning i tre dimensioner i en heldämpad kammare, med en rotationsupplösning på minst 0,2 grader per steg?

- Validera strålningsdiagram mot ideala simuleringar.

Följande krav ställdes på prototypen:

Prototypen ska bestå av ett vridbord som kan rotera runt horisontalplanet med en monterad hållare för antenner som kan rotera runt vertikalplanet (se Figur 2).

Konstruktion och tillverkning av litet vridbord

- Konstruera och ta fram CAD-modeller av ingående komponenter. Inköp av tekniska styr- och stödkomponenter. Innefattar tekniska och stationära komponenter för att få en helhet av vridbordet.
- Konstruktionen ska vara liten och flyttbar av en person.
- Måtten ska maximalt vara 40x40 cm med en höjd på maximalt 60 cm.
- Kammarens insida ska vara klädd med utstickande absorbenter med en höjd på 5 cm.
- Konstruera och ta fram fixturer för små antenner.
- 3D-printa modellerna och finjustera för att kunna fästa på ett vridbord.
- Minsta stegstorlek på 0.2 grader för vridbordet.
- Tillse att antennerna kan roteras för att skapa 3D-mätning.

Programmera systemet för att möjliggöra drift med en dator

- Programmering i Python eller liknande för att styra ett vridbord. Programmet ska implementeras och kunna köras på en dator.
- Programmera ett fungerande interface som skickar värden via pythonkoden till styrkoden för vridbordet.

Färdigställ liten befintlig heldämpad kammare (ca 1.5x2m)

- Inköp och montering av absorbenter för höga frekvenser till kammaren.
- Justering av kammare för att kunna montera vridbord.

Mätning av strålningsdiagram för antenn/sändare i liten heldämpad kammare

- Färdigställ mätuppställning med framtagna vridbordet, antenner, effektmätare, m.m.
- Skapa program, i exempelvis Python, för att mäta strålningsdiagram för antenn. Programmet ska implementeras och kunna köras på en dator.
- Mätning av strålningsdiagram sker för olika antenner.

1.3 Avgränsningar

Antennernas strålningsdiagram mäts endast i frekvensintervallet 5.8-40GHz. Laboranterna hade endast tillgång till programmeringsspråket Python. Arbetet innefattade inte att sälja eller massproducera produkten. Antennerna som arbetet avgränsades till var (givet från RISE):

- Grön bred antenn 5.9 GHz (“two stacked dipoles over ground plane”)
- Grön smal antenn 5.9 GHz (“stacked dipole over ground plane”)
- Gul antenn 5.9 GHz (Monopolantenn)
- Hornantenn 26.4 - 40.1 GHz

Mätningar för strålningsdiagram genomfördes endast på två antenner, grön bred antenn och grön smal antenn.

2. TEORI

Första delen av avsnittet innefattar teorier om strålning och antenner. Därefter tas teorier om 3D-printning relaterat till tillverkning av prototypen upp, och till sist teorierna som var till stöd för metoden.

2.1 EMC

Elektromagnetisk kompabilitet (EMC) innebär att elektriska och elektroniska enheter samverkar utan att de stör ut varandra genom elektromagnetiska fält eller störningar. Precis som att driftsäkerhet och prestanda är ett kvalitetskrav är även EMC det och regleras med lagar och standarder som EMC-direktivet (2014/30/EU). Inom EMC finns det två huvudområden. Det första är emission som innebär hur mycket elektromagnetiska störningar en enhet avger. Det andra är immunitet som visar en enhets tålighet för störningar från omgivningen. För att CE-märka en produkt enligt EMC-direktivet krävs det att den uppfyller krav för både elektromagnetisk emission och tålighet (Elsäkerhetsverket 2021).

2.2 Antenner

En antenn är en komponent inom elektroteknik som kan användas för att antingen skicka ut eller ta emot elektromagnetiska vågor. Sändarantennen omvandlar elektisk energi till elektromagnetiska vågor. Mottagarantennen fungerar tvärt om och tar emot vågor för att därefter omvandla det till energi. Beroende på hur antennen är konstruerad och hur många element den är uppbyggd av så strålas de elektromagnetiska vågorna ut på olika sätt (NE u.å.).

2.3 Högfrekvent Strålning

Radio- och mikrovågor utgör exempel på högfrekvent elektromagnetisk strålning och används i stor utsträckning inom trådlösa kommunikationstekniker, såsom television, mobiltelefoni och Wi-Fi. Strålningen skapas genom att högfrekvent växelström passerar genom antenner som i sin tur skickar ut korta snabba sinusvågor. Högfrekvent strålning infaller under kategorin ”icke-joniserande strålning” som är mindre farlig än den joniserade strålningen. Anledningen är att högfrekvent icke-joniserande strålning saknar tillräcklig energi för att frigöra elektroner och därmed jonisera atomer eller molekyler. Dock kan mycket stark högfrekvent strålning kan ge upphov till uppvärmning av kroppsvävnad, vilket vid tillräckligt höga nivåer kan orsaka biologiska skador. Det är inte bara frekvensen som avgör hur skadlig strålningen är utan även effekter mätt i Watt. Ett lågt frekvensintervall med en hög effekt kan nå upp till skadliga nivåer (Strålsäkerhetsmyndigheten 2024).

2.4 3D-strålningsmätning

Strålningsmätning i 2D innebär att antennens utstrålning mäts i ett enda plan. På RISE finns i nuläget endast möjligheten att rotera mottagarantennen i 360° på marken, vilket ger en begränsad bild av antennens egenskaper. Strålningsmätning i 3D täcker alla vinklar runt antennen och ger därmed en fullständig bild av hur den strålar i rummet, vilket är avgörande för mer avancerad analys och jämförelser mot simuleringar. 2D kan i projektet ses som endast vridplattans rotation, och 3D med mottagarantennens rotation inräknat (illustreras i Figur 2). I samma figur visas hur sändarantennen är fast och mottagarantennen roteras för att skapa en strålningsmätning i tre dimensioner.

Azimuth är ett namn på riktningen för ett objekts rotation i horisontalplanet (”markens” plan). Måttet anges i en vinkel från bestämd referenspunkt. Graderna ökar medurs från referenspunkten vilket ger 0-180 grader i vänstra halvplanet och 180-360 grader i högra halvplanet (Time and Date u.å). I projektet kallades stegmotor 1 för ”Azimuth” från dess

liknande rörelse. För rörelsen runt axeln som sträcker sig mellan mottagar- och sändarantenn i Figur 2, kallades rörelsen för ”Roll”. Alltså styr stegmotor 2 denna rörelse i projektet.

2.5 Simuleringsmodeller för antenner

I projektet valideras mätresultaten mot simuleringsmodeller för respektive antenn. Dessa modeller betraktas som ideala. Med "idealt" avses att antennen är perfekt konstruerad (form och material), att all signal överförs utan förluster i kablarna, att det inte finns några glapp i kabelfästen, inga reflektioner från omgivningen samt att både sändning och mottagning sker helt utan fel i systemet. Om någon av dessa förutsättningar inte uppfylls i verkliga mätningar kommer resultaten att avvika från simuleringen.

2.6 3D – Printning

3D-printing är en process där digitala filer omvandlas till tredimensionella modeller. Genom att bygga upp objekt lager för lager möjliggör 3D-printing skapandet av komplexa former med mindre materialåtgång jämfört med traditionell tillverkning, där material avlägsnas från ett större block för att skapa önskad form. För att skapa digitala filer avsedda för 3D-printing krävs ett program som stöder tredimensionell design. Det finns flera olika typer av 3D-printing, där Fused Deposition Modeling (FDM) är en av de vanligaste. FDM använder ett plastfilament som matas in från en spole till ett munstycke som smälter plasten. Munstycket kan röra sig horisontellt och vertikalt och använder en extruder (filamentmatare) för att trycka ut den smälta plasten och forma objektet lager för lager. Plasten stelnar omedelbart efter att den lämnat extrudern i munstycket (3D Printing, 2025).

2.7 Produktutveckling

Produktutvecklingsprocessen kan delas in i sex huvudfaser där varje fas i sin tur kan brytas ned i mindre moment för att möjliggöra en djupgående analys och skapa en användbar och relevant produkt. Den första fasen är idéutveckling där olika koncept och lösningar genereras utifrån identifierade behov och marknadsundersökningar. Därefter följer produktdefinitionen där produktens funktioner fastställs och målgruppen definieras. I nästa steg, prototypframställning, tas koncept fram och visualiseras genom simuleringar eller fysiska modeller. Virtuella tester spelar en central roll då de möjliggör kostnadseffektiv och snabb validering av idéer innan eventuella fysiska prototyper tas fram. När en färdig produkt börjar ta form genomförs validering och testning för att säkerställa att den uppfyller uppsatta krav och specifikationer. Slutligen sker kommersialiseringen då produkten lanseras på marknaden. Användningen av digitala verktyg och sociala medier har nuförtiden blivit allt viktigare för att effektivisera produktutvecklingen samt möjliggöra smidigare kommunikation och samordning (Asana, 2024).

2.8 Agil systemutveckling

Agila metoder används vanligtvis inom systemutveckling. Agila metoder innebär att man arbetar rörligt och flexibelt genom att leverera mindre delar av projektet för sig och därefter utvärdera varje del för att utveckla den. Arbetet utförs i små grupper för att ha en snabb dialog inom gruppen. Grupparbetet gör att en slutlig produkt kan levereras inom en kortare tidsram. Det finns flera olika typer av agil systemutveckling. Kanban är en av dem och fokuserar på ständiga förbättringar för att identifiera och undvika problem. Kanban ger utrymme för små förändringar under pågående arbete för att ge snabba resultat. Arbetsflödet förbättras kontinuerligt vilket ökar effektiviteten under pågående arbete (Projektledning 2018).

3. KOMPONENTER OCH MATERIEL

I detta avsnitt beskrivs de komponenter och materiel som användes i projektet. Först presenteras en lista på samtliga materiel och därefter den elektroniska hårdvaran såsom mikrokontroller, motorer och givare. Därefter följer beskrivningar av mätutrustning, inklusive signalgenerator och spektrumanalysator. Slutligen redovisas den utrustning och de material som användes vid konstruktion och tillverkning av prototypen, inklusive 3D-skrivare, CAD-program och filament.

3.1 Materiel

Avsnittet sammanställer allt som användes för att genomföra arbetet.

3.1.1 Antenner

- Grön antenn bred 5.9 GHz (“2 stacked dipole over ground plane”).
- Grön antenn smal 5.9 GHz (“stacked dipole over ground plane”)
- Gul antenn 5.9 GHz (Monopolantenn)
- Hornantenn 26.4 - 40.1 GHz

3.1.2 Program & drivkomponenter

- Creo Parametic (CAD)
- Python
- Arduino IDE
- Arduino UNO
- Dator
- Stegmotor NEMA 23
- Stegmotor NEMA 17
- Motordrivare TB6600 (2 st)
- Breadboard
- Transformator
- Spektrumanalysator
- Signalgenerator

3.1.3 3D - printning

- Bambu Lab X1E (3D-printer)
- Prusa XL (3D-printer)
- Filament (PLA)

3.1.4 Stödkomponenter och verktyg

- Kammare
- Absorbenter
- Antennkabel
- Arduinosladdar
- Flänskopplingar
- Krympslang
- Kabelstrumpa
- Plywoodskiva 12 mm
- Papper
- Träskruv
- Bult (M2, M3, M4, M5)
- Mutter (M2, M3, M4, M5)

- Lödstation
- Lödtenn
- Tejp
- Skjutmått
- Sax
- Fogsvans
- Skruvdragare
- Metallsåg
- Skruvmejsel (platt och stjärna)
- Tång
- Skiftnyckel
- Avbitare
- Penna

3.2 Arduino Uno

Arduino Uno är en mikrokontroll-bräda med 14 digitala in eller outputpins samt 6 analoga inputs. Brädan har en USB-B port som används för både strömförsörjning och kommunikation till datorn. Arduinon opererar på en spänning av 5V och har en maximal inspänning på 20V. Programmet Arduino IDE används för att skriva och ladda upp kod på språken C/C++. Arduino används inom IoT (internet of things), automation, mätning och styrning (Arduino 2025).

3.3 Motordrivare TB6600

Motordrivare TB6600 är en stegmotordrivare som är till för att driva tvåfasiga stegmotorer med mikrosteg. En stegmotor har möjligheten att dela upp en rotation (till exempel 1 grad) till ett antal lika stora steg. Med hjälp av 6 omkopplare kan mikrosteget ställas in i sju olika lägen (1, 2/A, 2/B, 4, 8, 16, 32) samt åtta nivåer för strömmen (0.5 A – 3.5 A) genom olika kombinationer av omkopplarna. Motordrivaren kan ta in mellan 9-24 Volt och leverera upp till 4 Ampere. Motordrivaren är kompatibel med Arduino och andra mikrokontroller som kan generera pulssignaler på 5V (Farnell 2024).

3.4 Stegmotor 1.89 Nm 1.8° NEMA 23

Stegmotor Nema 23 är en hybridmotor med en stegvinkel på 1.8 grader per steg. Motorn kan drivas med upp till 75 V och hantera en märkström på 2.8 Ampere. När motorn är strömförd har den ett hållmoment på 1.89 Nm (Elfa 2025). Motorn hanterar en axialkraft på max 15 N och en radial kraft på max 75 N. Drivpinnens diameter är på 6.35 mm och motorns totala vikt ligger på 1000 gram (Farnell 2019).

3.5 Stegmotor 9.6VDC 167Nmm 1.8° NEMA 17

Stegmotor Nema 17 är en hybridmotor med en stegvinkel på 1.8 grader per steg. Motorn kan drivas med upp till 24 V och hantera en märkström på 0.4 Ampere. När motorn är strömförd har den ett hållmoment på 0.17 Nm. Motorn hanterar en axialkraft på max 10 N och en radial kraft på max 28 N. Drivpinnens diameter är på 5 mm och motorns totala vikt ligger på 150 gram (Elfa u.å). Valet av Nema 17 baserades bland annat på dess låga vikt vilket var essentiellt för att ej överskriva maximala axialkraft för Nema 23.

3.6 Transformator 12V 6A 72W

Transformator som transformerar växelspänning på 230-240 Volt AC till likspänning på 12 Volt DC. Den har en effekt på 72 W och en maxström på 6 Ampere (LED Grossisten 2025).

3.7 Signalgenerator - Rohde & Schwarz SMCV100B

Rohde & Schwarz SMCV100B är en modern vektorsignalgenerator som stöder flera standarder och används inom broadcast, navigation, mobiltelefoni och trådlös kommunikation. Den bygger på ett innovativt koncept med direkt digital RF-uppkonvertering (Direct RF DAC), vilket möjliggör I/Q-modulering direkt i den digitala domänen. Tekniken eliminerar vanliga problem såsom I/Q-obalans och läckage från lokaloscillatorn (LO), som annars uppstår vid traditionell analog modulering. Enheten har ett frekvensområde från 4 kHz till 7,125 GHz och erbjuder hög uteffekt samt en moduleringsbandbredd på upp till 240 MHz. Generatorn har ett användarvänligt gränssnitt och är särskilt lämpad för produktionstillämpningar där flexibilitet och precision krävs. Signalgeneratorn används för att skicka energi till sändarantennen som antennen i sin tur omvandlar till strålning (Rohde & Schwarz, 2020).

3.8 Spektrumanalysator

En spektrumanalysator används för att mäta signalers amplitud som funktion av frekvensen och används ofta vid felsökning och karakterisering av radiosignaler. ESIB26 är en EMI-testmottagare och spektrumanalysator som täcker ett frekvensområde från 20 Hz upp till 26,5 GHz. Instrumentet kan arbeta i olika lägen, däribland spektrumanalys och mottagarläge, och har stöd för flera detektorer såsom max- och min-peak, RMS, medelvärde samt CISPR-genomsnitt. Mätningarna kan genomföras med hög upplösning, noggrann amplitudvisning och möjligheten till automatiserad datareduktion vid EMC-tester. Spektrumanalysatorn kopplades i arbetet till mottagarantennen (Rohde & Schwarz 2001).

3.9 Heldämpad kammare

För att minimera att brus och oönskade signaler kommer in när man gör mätningar inom EMC så används isolerade kammare. Dessa är generellt gjorda med ledande material i grunden såsom koppar, aluminium eller stål vilket i sig skapar en Faradays-bur. Insidan på kammaren täcks av absorberande material, så kallade absorbenter som fångar upp strålningen. Beroende på frekvensintervall så är materialen konstruerade på olika sätt och innehåll för att minimera reflektioner och studsar. Vid lister och dörrar används oftast koppartejp för att täta igen hål och minimera extra brus (RISE 2024). I Figur 1 visas kammaren som användes i arbetet med stängd dörr.

Figur 1: Heldämpad kammare. Mått: H:119cm, B:148cm, D:89cm

3.10 Absorbenter

Pyramidformade absorbenter av PU-skum från Holland Shielding, artikelnummer 3640–50, användes för att klä in kammaren. En platta, 60x60 cm, består av en "matta" med 900 pyramider, i vilken varje pyramid har en höjd av ca 50 mm och bas om ca 15x15mm. Absorbenterna har specificerade dämpningsegenskaper enligt Tabell 1 men dämpar även en bit utanför frekvensintervallet 6-40GHz. Reflektionsförlust (=Dämpning) av infallande elektromagnetiska fält enligt Tabell 1 (Holland Shielding 2024).

Tabell 1: Dämpningsegenskaper för användna absorbenter, artikelnummer 3640–50 (Holland Shielding 2024).

Frekvens (GHz)	Reflektionsförlust (dB)
6	30
10	35
18	40
40	50
100	45

3.11 Computer-Aided Design (CAD) – Creo Parametric

Creo Parametic är ett CAD-program som är till för att skapa och designa produkter i 3D. Creo använder teknologier som generativ design, realtidsimulering och sammanställning (Assembly). Vertygen möjliggör för användaren att designa, analysera och validera produkter samt optimera vikt, prestanda och hållbarhet. Creo kan användas från tidig produktdesign till digital prototyp och slutligen tillverkning (PTC 2025).

3.12 Filament (PLA)

Fused Deposition Modeling (FDM) är den vanligaste 3D-printingmetoden som används av cirka 70% av 3D-skrivarna på marknaden. Materialen som används kallas filament och är till stor del PLA, ABS, PETG och Nylon. PLA (Polylactic Acid) är det mest användbara filamentet och tillverkas endast av förnybara resurser. Det kräver ingen uppvärmd byggplatta i 3D-printern, har låg risk för deformation och är luktfritt. PLA är ett passande material för prototyper men är ej särskilt värmetåligt eller anpassat för utomhusbruk. Det är ett prisvärt och lättanvänt material som har lång livslängd och god styvhet (3D Guide 2024). PLA har en böjstyvhet på 2750 MPa (Bambu Lab u.å).

3.13 Bambu Lab X1E

Bambu Lab X1E är en 3D-skrivare som är kompatibel med flera material så som PLA, PETG och TPU. Byggvolymen för skrivaren är 256*256*256 mm^3 (Bambu Lab u.å).

3.14 Original Prusa XL

Original Prusa XL är en 3D-skrivare som är kompatibel med flera material så som PLA och PETG. Byggvolymen för skrivaren är 360*360*360mm^3 (Prusa Research u.å).

4. METOD

Metoden behandlar de övergripande tillvägagångssätten som användes för att systematiskt utveckla kod och konstruktion från grunden till slutgiltig produkt. Den innefattar bland annat materielen som användes och de första teoretiska skisserna från RISE.

4.1 Produktutveckling - Konstruktion

Produktutveckling användes för att ta fram en fungerande konstruktion. Genom att jämföra olika idéer om hur konstruktionen kunde komma att se ut kunde den mest lämpande designen väljas ut och utvecklas. Majoriteten av delarna i konstruktionen gick igenom samma procedur. Enklare delar i konstruktionen som sladdlisten och elcentralen togs fram på enklaste sätt genom CAD-ritning för att uppfylla dess syfte och 3D-printning utan utveckling och iterationer.

Problemidentifiering och kravspecifikation

Krav från specifikationen diskuterades samt vilka problem som var de övergripande för konstruktionen. Konstruktionen delades upp i flera delar för att fokusera på en del i taget vid framtagningen.

Idéutveckling

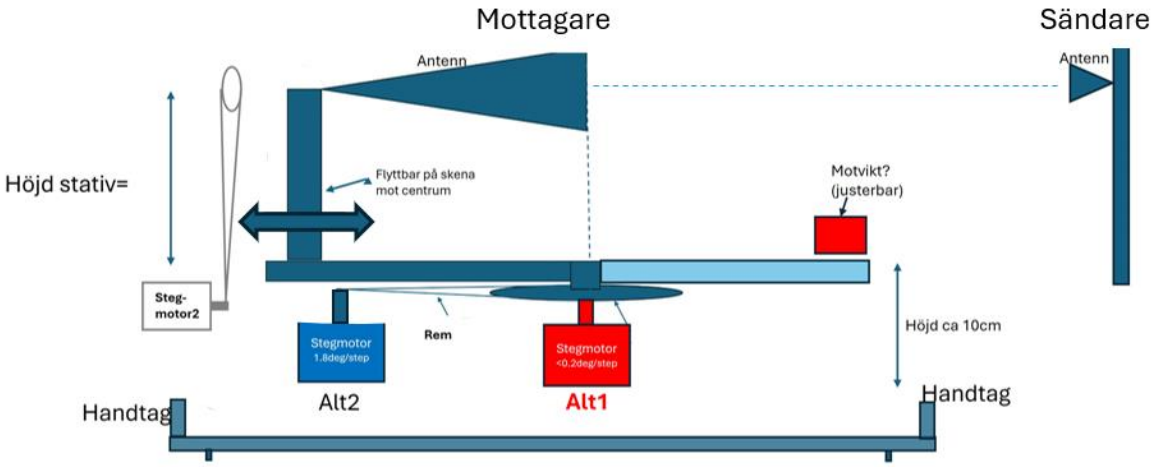
Tillfällen skapades för brainstorming, skissning och öppna diskussioner av möjliga lösningar. I de fallen det fanns möjlighet skapades flera olika koncept för de olika delarna för att jämföra möjliga lösningar. Dessa presenteras i resultatavsnittet.

Konceptutveckling

Detaljerade CAD-skisser skapades med hjälp av Creo Parametic och sammansattes med hjälp av CAD-verktyget assembly. Filerna omvandlades från .prt-filer till .stl-filer för att stödja 3D-skrivarnas program. Prototyper 3D-printades för att kontrollera passformen mot övriga komponenter samt utvärdera funktionaliteten. Vid nya iterationer skapades kopior på .prt-filerna i Creo för att behålla de tidigare modellerna om något skulle gå fel. Vid omvandlingen från .prt till .stl fil användes “Customize Export” i Creo för att redigera “Step Size” i filen till 1 eller mindre. Vid skapandet av delar som skulle passa till tillhörande komponenter användes ett skjutmått för att ta fram rätt dimensioner.

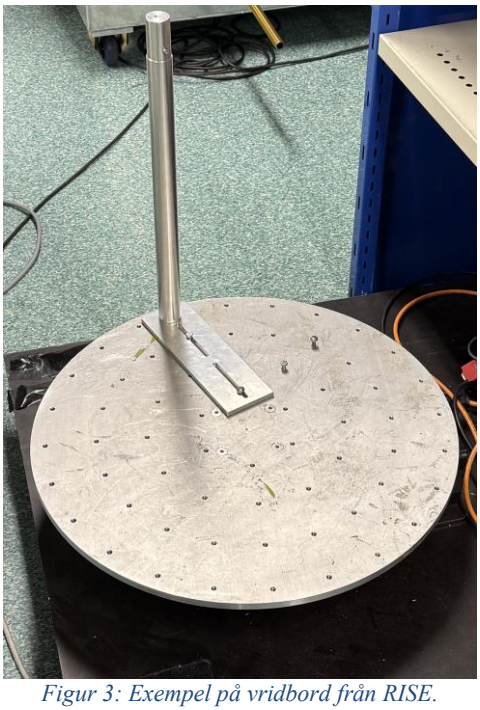
Utvärdering och iteration

Justeringar och förbättringar genomfördes på prototyperna. Processen upprepades till dess att konstruktionens delar uppnådde kravspecifikationen och god funktionalitet. Dessa utvärderingar för respektive koncept förklaras djupare under resultatdelen.

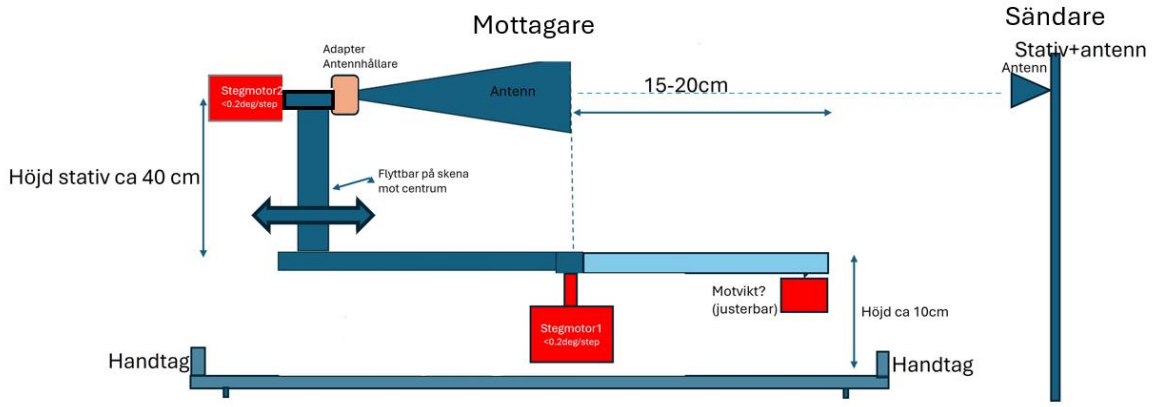


Figur 2: Första skissen på konstruktionen.

Det första designförslaget på konstruktionen kom från företaget själva som visade en övergripande ritning i 2D (se Figur 2). Där visades två alternativ för montering av azimuth-motorn, blå och röd rektangel nederst i Figur 2. Det tydliggjordes här från handledare på RISE att rotationscentrum för respektive antenns spets behövde vara vertikal med axeln för azimuth-motorn. Dessutom behövde antennerna sitta på samma höjd. Roll-motorns placering var tänkt att monteras med rem och visas som vit rektangel enligt figuren. På företaget fanns en liknande konstruktion i större skala som var till stöd för designarbetet, se Figur 3. Under plattan på samma figur satt styrenheten som hade köpts in i en färdig låda. Eftersom lådan inte fick öppnas upp var det svårt att tillämpa en liknande idé i projektet. Den befintliga konstruktionen var en enklare konstruktion och hade endast möjlighet att rotera i ett plan (Azimuth).



Figur 3: Exempel på vridbord från RISE.



Figur 4: Andra skissen på konstruktion

I Figur 4 visas den andra skissen som togs fram tillsammans med företaget. Azimuth-motorn bestämdes för att ha sin placering rakt under vridbordet då det andra alternativet endast var till för om motorn inte kunde röra sig med tillräckligt små steg. Med en motordrivare som gick att drivas med mikrosteg från 1 till 1/32 steg, kunde stegstorleken ställas ner till från 1,8° till 1,8/32 = 0,05625°. Konceptet uppfyllde därmed kravspecifikationen på 0,2 grader utan att använda rem och rem-skiva. Roll-motorn flyttades upp och drivremmen kunde tas bort. Eventuell motvikt flyttades till undersidan för att inte vara i vägen för strålningen från antennen. Motvikten användes ej i den slutliga konstruktionen.

4.2 Agil systemutveckling - Programmering

En agil metod användes för att möjliggöra ett rörligt och flexibelt arbetssätt under hela programmeringsarbetet. För att öka effektiviteten delades programmeringen upp i flera mindre delar. Kanban, som är en typ av agil metod, användes för att underlätta förbättringar samt för att identifiera och undvika potentiella problem.

Arduino-programmet utvecklades för att styra två motorer med hög precision. Båda motorerna har en startpunkt som definieras som nollpunkt. Från denna punkt räknar programmet upp till 180 grader i medurs riktning, samt upp till 180 grader i moturs riktning. Biblioteket *AccelStepper* användes för att möjliggöra en mer exakt och kontrollerad styrning av motorernas rörelser.

4.2.1 Mätningsrutin

Under en mätningsrutin rör sig azimuth-motorn inom ett fördefinierat intervall med start och stoppvärde enligt användarens angivelse. När en rörelse inom azimuth-intervallet är genomförd, förflyttar sig roll-motorn ett steg, där även detta stegs storlek definieras av användaren. Därefter återställs azimuth-motorn till nolläge, och rörelserna upprepas tills roll-motorn når sin angivna slutposition. När detta är uppnått har programmet genomfört en fullständig mätningsrutin och återgår till nolläge.

Vid vissa typer av specialmätningar kan det vara nödvändigt att azimuth- och roll-motorn byter funktion med varandra. En separat körning har därför implementerats i programmet, där motorernas funktioner omfördelas. Detta ökar systemets flexibilitet och möjliggör anpassning till olika mätbehov och konfigurationer.

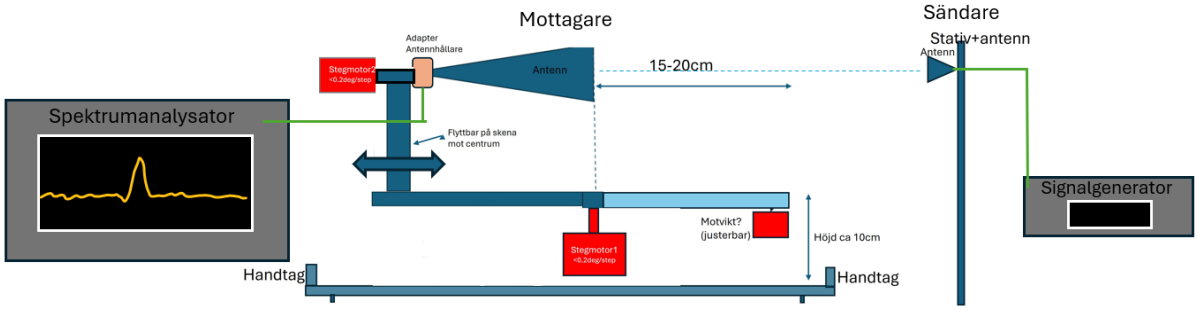
En begränsande faktor var kablarna, då de inte tillät en fullständig 360-graders rotation. Azimuth-motorn är därför begränsad till maximalt 180 graders rotation från nollpunkten, i både positiv och negativ riktning. Programmet måste hantera styrningen på ett sätt som förhindrar att azimuth-motorn roterar förbi denna gräns, då det annars kan leda till skador på kablarna.

4.2.2 Kommunikation mellan Arduino och Python

Vid uppstart av systemet så står arduinon i vänteläge och väntar på seriell kommunikation från pythonkoden. Datan som skickas från python är en lista med värden av intervall, stegstorlek, väntetid och roll-funktion. När datan kommit in så kör arduinokoden systemet enligt beskrivning. Vid specifika ställen i koden sitter Serialprint-satser som printar ut aktuell data i arduinoterminalen. Genom kommunikation på samma baudrate och COM-port för dator och arduino så kan pythonkoden ta upp vad som skrivs i arduinoterminalen och skicka ut det till pythonterminalen. Denna data kan senare användas till live-plottning och datainsamling. All kommunikation sker trådbundet.

4.3 Datainsamling – mätningar

Antennerna som användes kunde skicka och ta emot frekvenser runt 5.9 GHz. Därmed ställdes signalgenerator på frekvensen 5.9 GHz och med en uteffekt på 14 dBm. Mätuppställning med spektrumanalysator och signalgenerator visas i Figur 5, där den gröna linjen visar vart kablarna kopplades in till respektive antenn. För att skicka ut strålning kopplades signalgeneratorn till sändarantennen som visas till höger i figuren. Avläsningen av strålningen gjordes med spektrumanalysatorn inkopplad till mottagarantennen som visas till vänster i samma figur.



Figur 5: Mätuppställning med spektrumanalysator till vänster och signalgenerator till höger kopplade med gröna linjer till respektive antenn.

Vid utsatta rader i arduinokoden finns kommandon när mätningar ska göras i form av ”Serial.println(”Mätning”). Informationen tolkas av Python-koden, som med hjälp av egna funktioner instruerar spektrumanalysatorn att mäta den maximala amplituden. När datan kommit in till pythonprogrammet så uppdateras en CSV-fil med den aktuell position och maxamplitud för varje mätning. Exempel på CSV-filens utseende visas i Figur 6 där varje rad uppdateras med den aktuella positionen och maxamplituden.

mätdata_20250417_121109.csv

```
Azimuth,Roll,Peak Level (dBm)
0.0,39.99,40.06521
5.01,39.99,40.08352
10.01,39.99,40.49026
15.02,39.99,39.98996
20.02,39.99,39.76625
25.03,39.99,37.93995
30.04,39.99,32.71119
35.04,39.99,23.4373
40.05,39.99,31.74109
45.06,39.99,37.72031
50.06,39.99,40.85023
55.07,39.99,41.57222
60.07,39.99,40.1669
65.08,39.99,35.23303
70.09,39.99,31.9587
75.09,39.99,39.77438
80.1,39.99,41.22445
85.11,39.99,40.20351
90.11,39.99,32.75389
0.0,29.98,37.36033
5.01,29.98,38.10468
10.01,29.98,38.71481
```

Figur 6: Del av mätdata från en hel mätning med Azimuth: stegstorlek 5, start 0, stopp 90. Roll: steg 10, start 40, stopp 0.

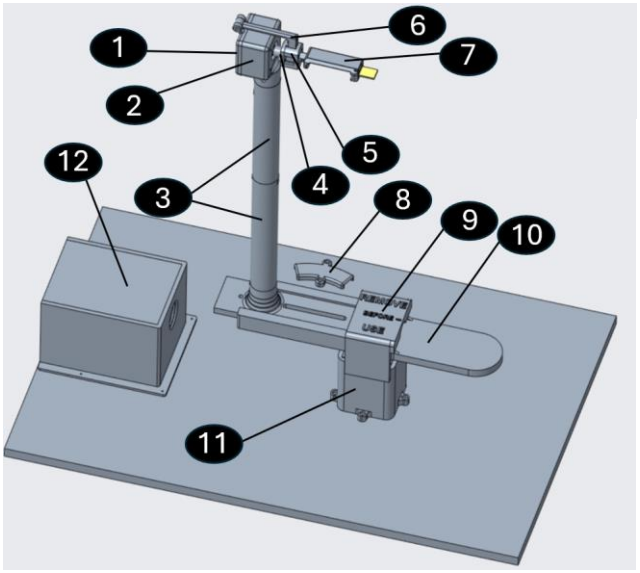
I avsnitt 5.3.4.1 tydliggörs det hur värdena för mätningen anges i interfacet.

5. RESULTAT - PROTOTYPFRAMTAGNING

Avsnittet presenterar de mest relevanta koncepten samt de slutgiltiga resultaten för respektive CAD-fil, konstruktion och programmering. Första delen går igenom konstruktionen strukturerad nerifrån och upp. Mellan avsnitten för hållare till roll-motor och antennhållarna visas nollställaren till Azimuth. I de senare delarna tydliggörs kopplingsschema för hela systemet, flödesschema och förklaring av koderna. Slutligen visas hur kalibreringen av systemet gjordes.

5.1 Konstruktion

Konstruktionsdelen innefattar CAD-modellerna och de tekniska komponenterna som användes för att stödja upp och styra antennerna. Avsnittet går igenom delarna från botten av konstruktionen och upp till hållaren för roll-motorn och dess tillhörande delar, vidare förklaras nollställaren och därefter antennerna med tillhörande hållare, slutligen förklaras övriga delar utanför antennkonstruktionen. I Figur 7 numreras delarna för att tydliggöra vart delarna sitter på konstruktionen.

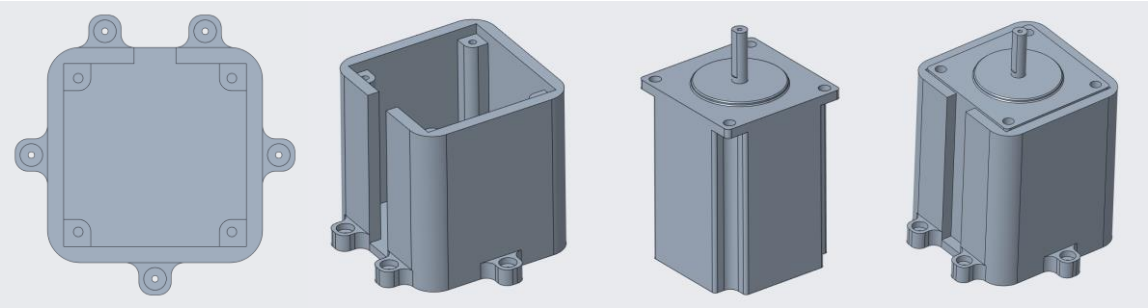


1. Lock/baksida för hållare roll-motor
2. Hållare roll-motor
3. Tvådelad antennpinne
4. Flänshållare
5. Universalhållare
6. Hasp
7. Antennfäste
8. Sladdlist
9. Nollställare
10. Vridbord
11. Hållare azimuth-motor
12. Elcentral

Figur 7: Slutlig konstruktion med siffror som indikerar delarna som 3D-printades i konstruktionen.

5.1.1 Hållare azimuth-motor

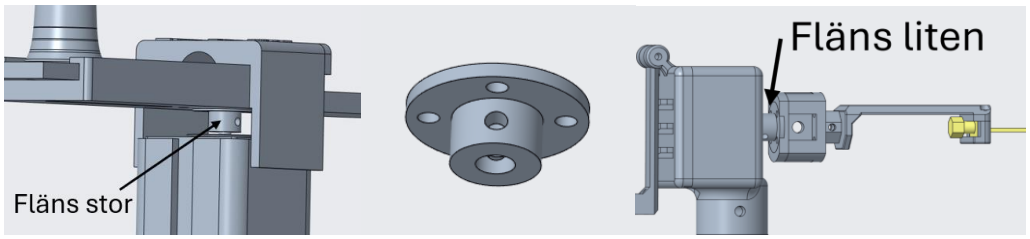
Hållaren till azimuth-motorn, NEMA 23, designades för att kunna monteras på en bottenplatta med träskruv. Med hjälp av ett skjutmått togs mätningar för att rita upp azimuth-motorn i CAD. Måtten från mätningarna användes därefter för att skapa en hållare som var formad som motors yttersta hölje. En tolerans på 0,1 mm användes för att uppnå god passform runt motorn. En öppning från toppen till botten på en av sidorna krävdes för att motorns sladdar inte skulle ta i vid monteringen. För att fästa motorn i motorhållaren användes 4 träskruv som skruvades ner genom befintliga hål på motorn och vidare ner i motorhållarens 4 hål på insidan. Motorhållare och azimuth-motorn visas i Figur 8.



Figur 8: Hållare till azimuth-motor i CAD utan motor till vänster och med motor i till höger.

5.1.2 Aluminiumfläns stor och liten

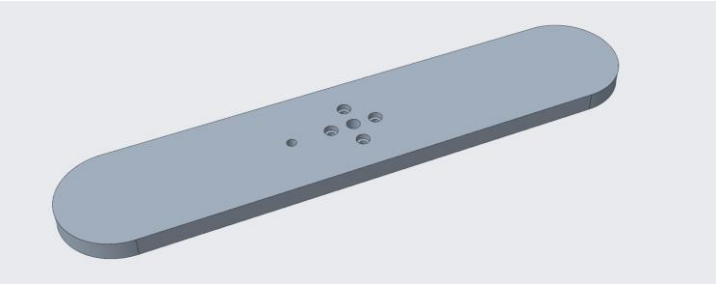
För att få över vridmomentet från stegmotorerna till vridbordet och antennhållarna behövdes något som fäste dessa samman. Till konstruktionen valdes aluminiumflänsar för montering. Dessa sattes över stegmotorernas drivpinnar ovanifrån och skruvades fast med två tillhörande skruvar genom gängade hål in i drivpinnen på motorerna för att hålla aluminiumflänsen på plats. Aluminiumflänsarna beställdes och dimensionerades utifrån stegmotorerna. Stegmotor Nema 23 hade en drivpinne med 6,35 mm i diameter och stegmotor Nema 17 hade en drivpinne på 5 mm i diameter. Flänsarna kunde endast beställas i hela millimeter som till slut valdes till 7 respektive 5 mm i innerdiameter. Med hjälp av specifikationerna på flänsarnas mått från hemsidan skapades de även i CAD för att kunna visualiseras i helheten (se Figur 9). Den större flänsen till Nema 23-motorn hade ett glapp på 0,65 mm mot drivpinnen vilket ansågs som ett möjligt problem i början. Men när flänsen monterades fast med tillhörande skruvar satt den bra. Då den mindre flänsen hade en innerdiameter som nästan var identisk med pinnens ytterdiameter på Nema 17-motorn så fanns det så gott som inget glapp.



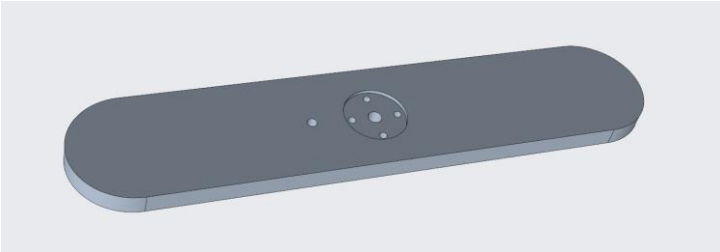
Figur 9: Aluminiumflänsarnas placering på konstruktionen, stora till vänster, CAD-fläns i mitten, liten till höger.

5.1.3 Vridbord

Först var tanken att göra en konstruktion i antingen aluminium eller kolfiber för att få en så stabil och lätt konstruktion som möjligt. Från specifikationer från handledare så ritades ett cirkulärt vridbord upp i Creo Parametric 8.0 med tvärsnittsdiametern på 30 cm. Tjockleken på plattan valdes till 10 mm för att från start minimera vikt. NEMA 23-motorn, som valdes till azimuth-motor, hade en maximal axiell lastkapacitet på 15 N vilket gjorde att vikten blev ett problem. Den första lösningen var att minska storleken på vridbordet till ritningen i Figur 10 och Figur 11. Längden av vridbordet behölls oförändrat till 30 cm medan bredden reducerades till 6 cm.

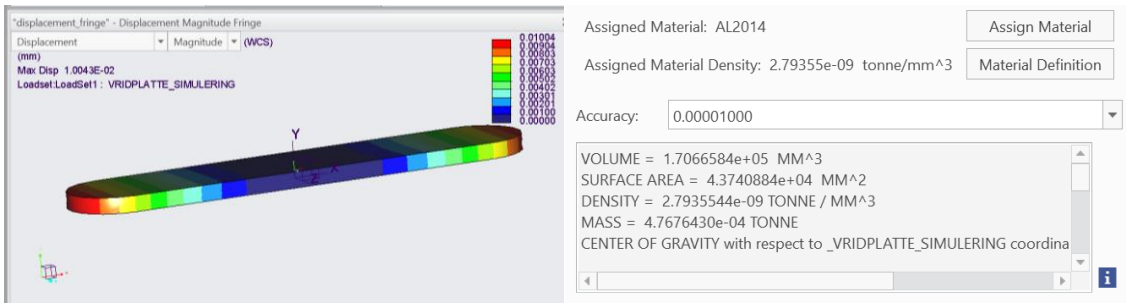


Figur 10: Första konceptet på vridbordet ovanifrån.



Figur 11: Första koncept på vridbordet underifrån.

För att säkerställa så exakta mätningar som möjligt med antennerna krävdes en konstruktion som var stabil, med minimala nedböjningar eller svängningar i materialet. Vridbordet testades redan i ett tidigt skede med en multi-pass simulering i Creo Parametrics simuleringsverktyg. Simuleringen, som visas i Figur 12, visar att en last på 5 N på vardera sida resulterade i en maximal nedböjning på 0,01 mm. Enligt handledaren på RISE bedömdes nedböjningen som försumbar vid framtida strålningsmätningar.



Figur 12: Simulering på nedböjning med aluminium som material & data för delen.

För att testa konstruktionen bestämdes det att allt skulle 3D-printas för att kontrollera passformen och göra eventuella justeringar innan aluminiumdel beställdes. Samtidigt diskuterades det om att 3D-printa en alternativ design i PLA för att testa hållfastheten i det materialet.

Externa företag kontaktades för att se hur mycket en laser-printad aluminiumprofil hade kostat. Laboranterna skickade CAD-filen till ett företag för offert och fick tillbaka ett uppskattat pris på 50 000 kr vilket bedömdes som orimligt högt. En plan B enligt Figur 13 skapades då den interna verkstaden på RISE eventuellt kunde tillverka ett vridbord i aluminium. Det hade inneburit spår som visas i Figur 13 i stället för bågarna i Figur 14. För att säkerställa korrekt passform behövdes vridbordet, oavsett val av material, 3D-printas. Beslutet togs att 3D-printa vridbordet i PLA och vid tillfredställande resultat använda PLA som slutligt material alternativ

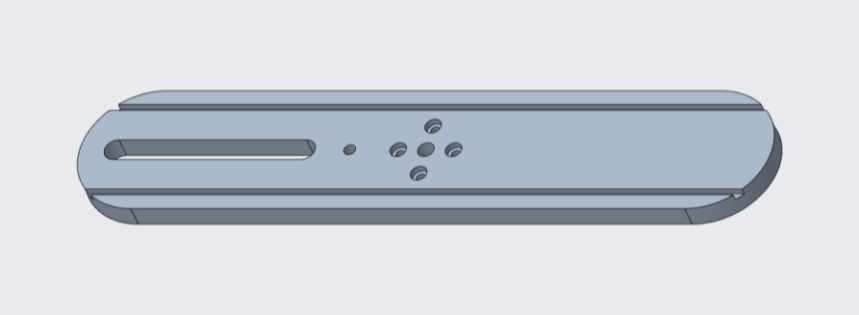
gå vidare med de andra materialen. En jämförelse mellan PLA och aluminium skapades enligt nedan.

ALU
Positivt: Hållfast konstruktion, god stabilitet.
Negativt: Sämre vid strålning (strålningen studsar, hade behövt klä in hela konstruktionen i absorbenter), tyngre vikt, dyrare, längre leveranstid, svårare att göra förändringar på.

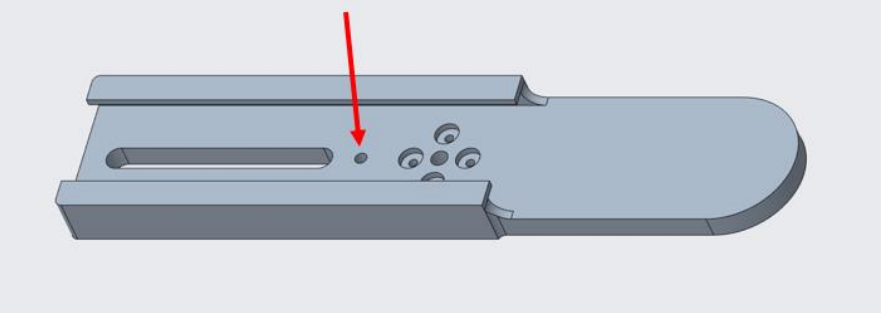
PLA
Positivt: Billig, enkel tillverkning, kort tillverkningstid, lätt att designa om, låg vikt, påverkar inte strålning.
Negativt: Sämre hållfasthet. (PLA styvhet på 2,75 GPa, ALU styvhet på 69 GPa) (Bambu Lab u.å.) (Metallkompetens u.å.).

Figur 14 och Figur 15 illustrerar den slutgiltiga konstruktionen för vridbordet som 3D-printades i PLA. Denna innefattade fästen för aluminiumfläns, hål för kabeldragning samt låsningshål för antennpinnen. Konstruktionen printades i 90 procent ifyllnad för att maximera hållfastheten. Det diskuterades att använda mer hållfast plast såsom NylonPower Carbon Fibre från Prima Select. Dock har tester med det materialet från tidigare projekt fungerat dåligt med missformning och sprickor. Svart PLA från Prima Select var plasten som i slutändan användes. Denna hade en printtemperatur på 190-220°C.

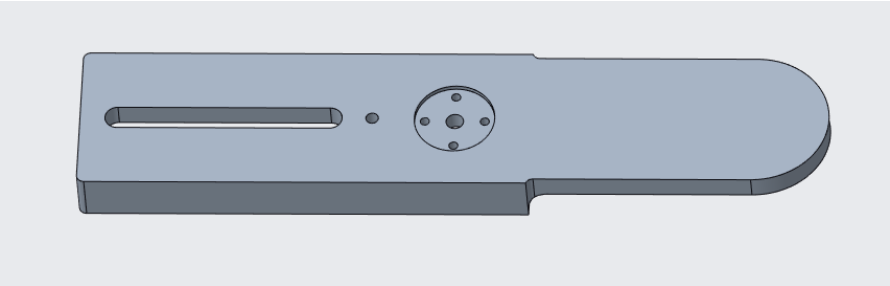
Ett skruvhål (markerat med röd pil i Figur 14) konstruerades till vänster om flänsens centrala fäste, vilket framgår i både Figur 14 och 15. Syftet var att möjliggöra justering av avståndet till rotationscentrum för antenner av olika storlek. Skruvhålet på Figur 14, tillsammans med det avlånga hålet på antennpinnens botten, visad i Figur 16, skulle göra det möjligt att justera distansen upp till 7 cm. Det avlånga hålet till vänster på vridplattan bredvid skruvhålet är för sladdarna från roll-motorn som dras på insidan av antennpinnen och ner genom vridplattan.



Figur 13: Plan B, ALU-profil som tillverkas på RISE.



Figur 14: Slutgiltigt vridbord ovanifrån, röd pil pekar på skruvhål för montering av antennpinne.

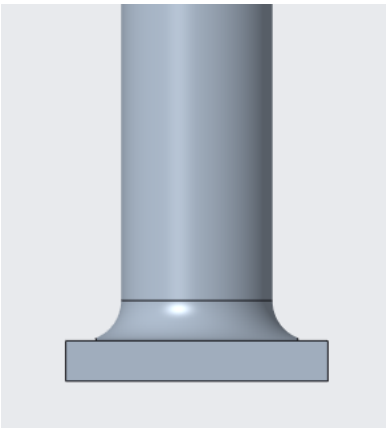


Figur 15: Slutgiltigt vridbord underifrån.

5.1.4 Antennpinne

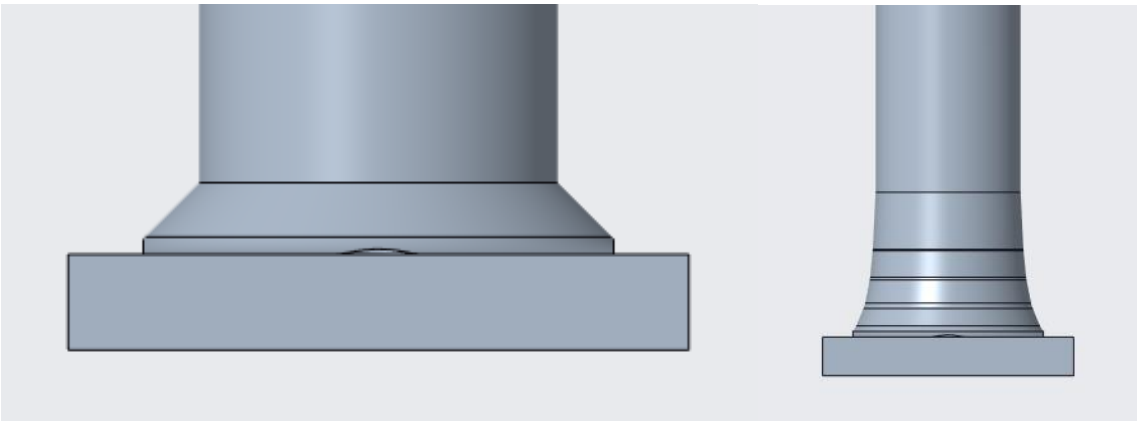
Ställningen för antennen behövde enligt kravspecifikationen ha en höjd på 40 cm för att få en optimal mätuppställning. Material som kolfiber, aluminium och PLA jämfördes för att uppnå önskade resultat. Med utgångspunkt från den tidigare befintliga konstruktionen som fanns på företaget (Figur 3) ritades en anpassad antennpinne i CAD. PLA testades först för att sedan ha de andra materialen som en plan B likt vridbordets framtagning. Bambu Lab printerns maximala höjd för utskrift var 25 cm, vilket innebar att pinnen behövde delas på två delar (se Figur 19). För att skapa en stabil tvådelad antennpinne skapades en inre cylinder på den övre delen som fördes ner i den nedre delen. Två kilar på fram och baksida skapades för att styra ner överdelen i rätt vinkel då en motorhållare senare skulle monteras på toppen. På antennpinnens topp förbereddes en inre cylinder med hål genom för att lätt kunna montera en hållare till roll-motorn. Den första designen, innan den tvådelades, var inte tillräckligt stabil då den flexade för mycket i både bottenplattan och nederdelen av cylindern (se Figur 16 och 17).

Figur 16: Första koncept antennpinne, ej tvådelad.

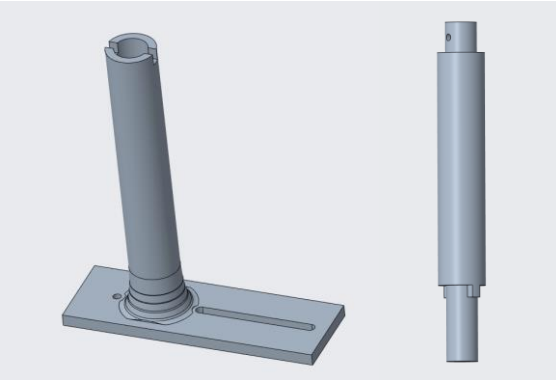


Figur 17: Den första typen av avrundning på antennpinnes nedre del.

Förbättringar som bättre avrundning nedtill och mindre innerdiameter samt tjockare bottenplatta gjorde att antennpinnen blev mycket stabilare visat i Figur 18. I Figur 17 visas den första avrundningen som gjordes med verktyget Round i Creo. Då denna avrundning inte gav tillräcklig stabilitet testades en ny avrundning med hjälp av Chamfer verktyget. Chamfer fyller ut en skarp vinkel eller ett hörn med en rak linje till skillnad från Round som använder en rund ifyllnad. Genom att använda Chamfer flera gånger på varje ny vinkel som skapades vid varje steg blev den slutliga designen en avrundning med mer material runt den nedre delen av antennpinnen som gav en stabilare konstruktion (Se Figur 18). Den slutliga tvådelade antennpinnen (se Figur 19) fördes sedan in i vridbordets bågar. Dessutom förbättrades formen på botten av antennpinnens platta till rektangulär för att få bättre stabilitet mot vridbordets bågar. Ett till hål skapades även bakom antennpinnen (se Figur 19) för att möjliggöra montering av ytterligare en bult och mutter för att vid behov öka stabiliteten mot vridbordet. Användning av detta fall uppstår om antennpinnen behöver monteras närmare centrum än vad det avlånga hålet på tillåter. I så fall används en skruv med mutter som går genom hålet i antennplattan och det avlånga hålet i vridplattan. Den slutliga iterationen gav en tillräckligt robust antennpinne i alla led.



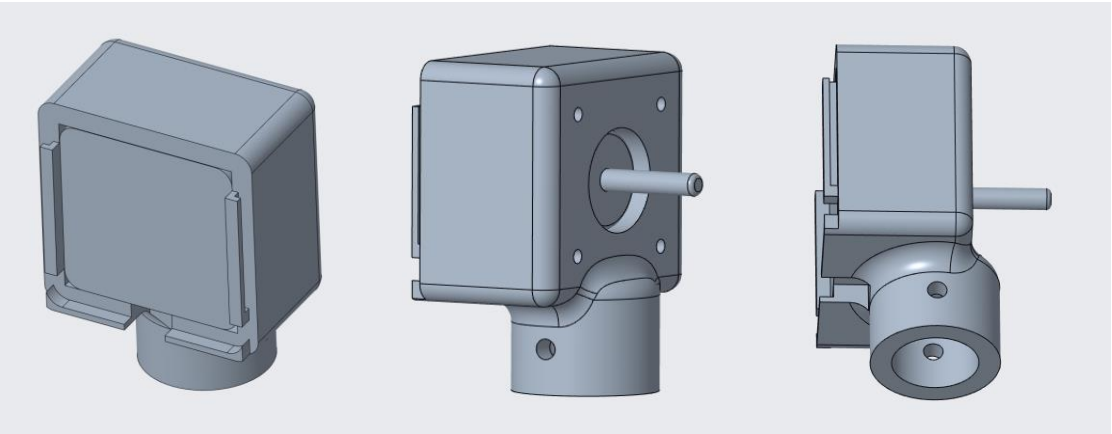
Figur 18: Till vänster visas första steget på "Chamfer" som sedan med upprepade steg utvecklades till avrundningen till höger.



Figur 19: Slutgiltig tvådelad konstruktion av antennpinne.

5.1.5 Hållare roll-motor

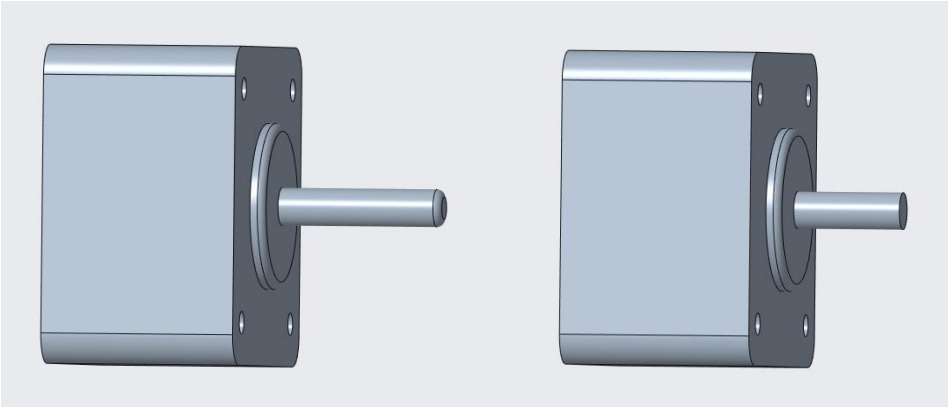
Hållaren designades för att fästa roll-motorn (NEMA 17) med 4 bultar i befintliga gängade hål på motorn (se Figur 20). Baksidan förbereddes för att senare möjliggöra montering av ett avtagbart lock. Botten på hållaren konstruerades för att monteras på antennpinnens topp och fästas med en M3-bult och mutter genom ett hål. En öppning på baksidan av hållaren möjliggjorde för sladdragning från motorn och in genom ett hål i antennpinnens baksida. Hållaren 3D-printades med 90 procents ifyllnad för att bli mer hållfast och klara av vikten från antennen och dess hållare. Ökad ifyllnadsprocent gav dessutom en stabilare konstruktion under drift.



Figur 20: Hållare roll-motor i CAD utan baksida och med motorn monterad.

5.1.6 Roll-motor

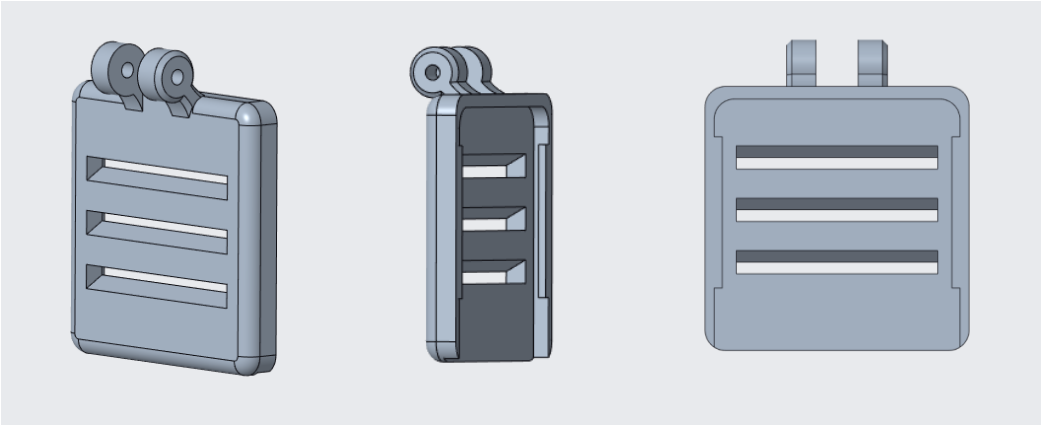
Drivpinnen till roll-motorn sågades av till 14,4 mm (7,5 mm togs bort) med metallsåg. Åtgärden vidtogs eftersom den stack ut onödigt mycket, vilket gjorde att antennhållarens och antennens gemensamma vikt skapade ett alltför stort moment på pinnen. Avsågningen illustreras virtuellt i Figur 21.



Figur 21: Roll-motor i CAD med standardlängd på drivpinne till vänster samt längd på avsågad pinne till höger.

5.1.7 Lock/baksida för hållare roll-motor

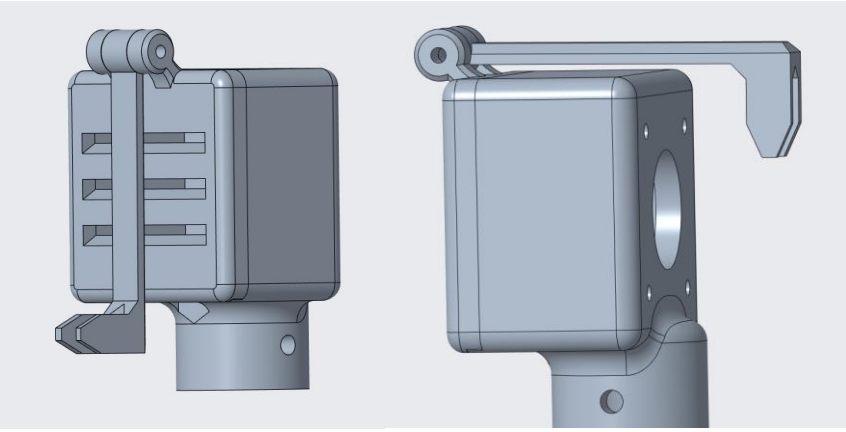
Ett lock skapades med en glidlist på insidan som monterades på baksidan av hållaren till roll-motorn. För att undvika att motorn blev för varm skapades ventilationshål genom locket (se Figur 22).



Figur 22: Lock/baksida i CAD för hållare till roll-motorn.

5.1.8 Hasp

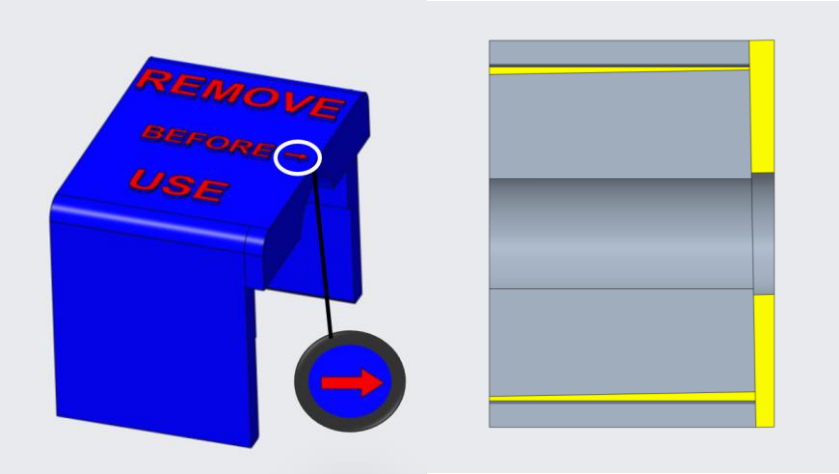
Roll-motorn som användes hade ingen funktion för att komma ihåg ett nolläge. En hasp skapades som manuellt fästes i antennkonstruktionen för att ställa in motorns position i ett nolläge till att bli horisontell. Antennkonstruktionen förklaras i senare avsnitt. Locket på hållaren till roll-motorn utvecklades med två cylindriska hållare på toppen, en på vardera sida, för att fästa haspen. Haspen monterades i mitten som därefter fästes med en M3-bult och mutter (se Figur 23). M3-bult och mutter visas inte i figuren.



Figur 23: Hållare till roll-motor i CAD med lock på baksidan och hasp.

5.1.9 Nollställare Azimuth

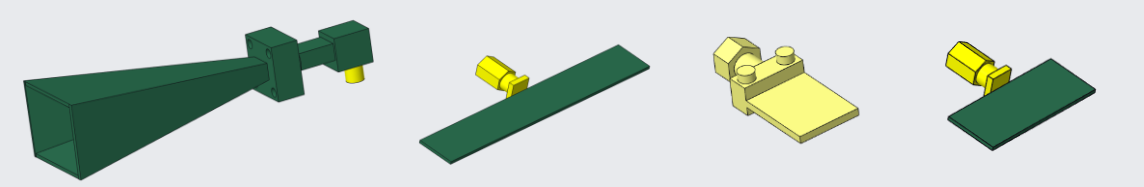
Azimuth-motorn rörde sig alltid till förbestämda positioner när den strömfördes. Alltså gick det inte att hitta ett exakt ”nolläge för motorn”. Det gjorde att vridbordet inte ställde sig rakt fram i förhållande till hållaren för motorn. När vridbordet manuellt styrdes till ett läge som ansågs vara rakt fram kunde motorn gå både mot- och medurs vid strömförning. En nollställare, visad i Figur 24 konstruerades för att styra vridbordet till önskad riktning. Nollställaren monterades ovanpå vridbordets mitt och över hållaren för motorn. Genom att vinkla insidan av väggarna (gul markering i Figur 24) justerades vridbordet med en liten vinkel i medurs riktning. När nollställaren därefter lyfts bort från vridbordet står vridbordet i vinkel vid strömförning. Genom denna metod ställdes vridbordet alltid till samma position vid start. Från detta implementerades en startsekvens i koden där motorn började med att köra tillbaka 72 mikrosteg i moturs riktning för att rikta in vridbordet rakt fram. Nollställaren printades blå med röd text som löd ”REMOVE BEFORE USE”. Anledningen till färgvalet var att den skulle sticka ut från resterande konstruktion i svart, samt med en text för att inte strömföra konstruktionen med nollställaren på. En pil på ovansidan markerade riktningen för korrekt montering av nollställaren på vridbordet. Nollställarens pil ska peka i samma riktning som antennen riktas åt. Pilen är endast en symbol för att förhindra att nollställaren placeras i motsatt riktning.



Figur 24: Nollställare visas i färgen den printades i till vänster och till höger ett förtydligande kring vilka väggar som vinklades på insidan/undersidan.

5.1.10 Antennhållare

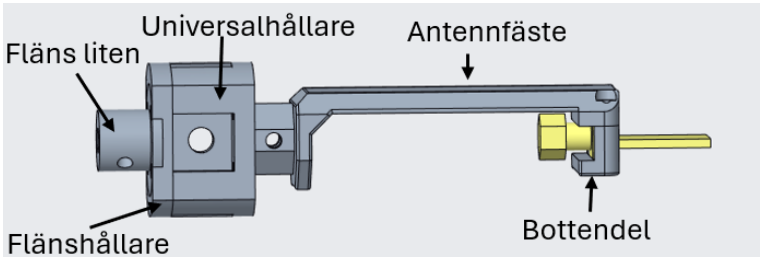
Fyra antenner av olika typer tilldelades från företaget. Genom att mäta samtliga dimensioner på antennerna med ett skjutmått kunde de ritas i CAD enligt Figur 25. CAD-antennerna användes endast för att dimensionera hållarna till antennerna korrekt.



Figur 25: Från vänster: hornantenn, grön bred antenn, gul antenn, grön smal antenn.

Respektive antennhållare består av ett flertal delar. Delen som överför vridmomentet från flänsen till roll-motorn är flänshållaren. I denna sitter antennkabeln monterad. Nerskruvad i flänshållaren sitter universalhållaren som har ett hexagon-format fäste framtill som passar till alla de olika antennfästena.

Hela antennhållaren med samtliga delar visas nedan ihopsatt i Creo med hjälp av assembly (se Figur 26). Den kompletta hållaren monteras på roll-motorns drivpinne med två skruvar genom gängade hål i aluminiumflänsen.

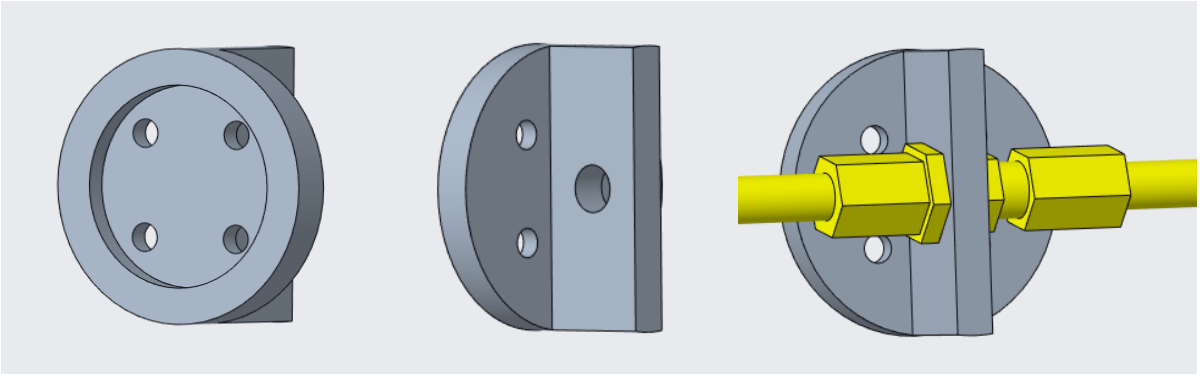


Figur 26: Hela antennhållaren för den gula antennen.

Nedan beskrivs och visas varje del i detalj.

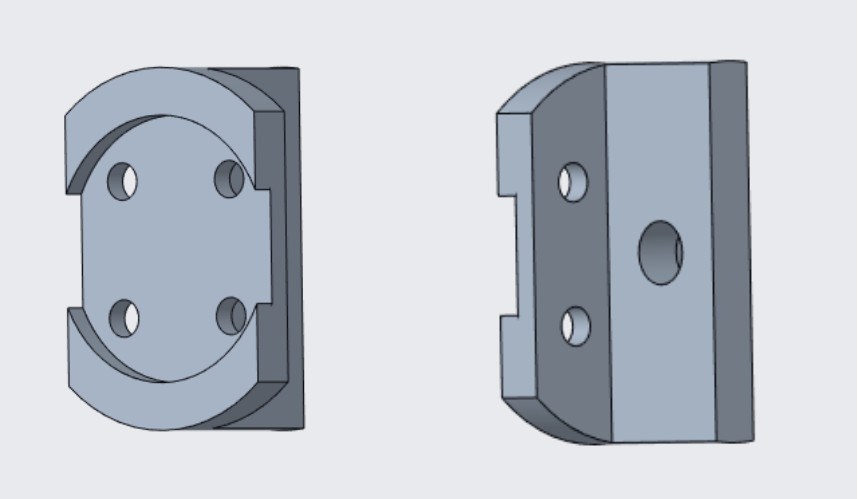
Flänshållare:

En hållare konstruerades för att kunna montera aluminiumflänsen till antennhållaren. Flänsen fästes med M3 muttrar genom fyra förborrade hål på flänsen och de fyra hålen skapade i hållaren. En försänkning i flänshållaren skapades för att göra flänshållaren stabil och förbättra hållfastheten på hållaren. På framsidan fästes en koppling till antennkabeln (gul kabel i Figur 27) genom ett hål där den sedan skruvades fast med mutter från båda hållen. Kopplingen till antennkabeln, som fästes genom hållaren gjorde det möjligt för kabeln att rotera med tillsammans med antennen och motorn. Samtidigt skapades en enkel övergång mellan två antennkablar.



Figur 27: Första designen på flänshållaren med hål för montering av antennkabel på framsidan.

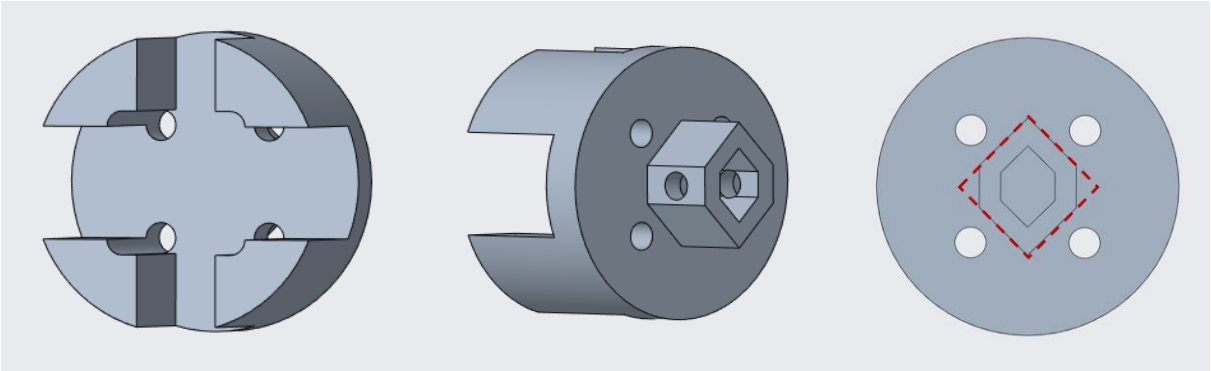
När hållaren testades i konstruktionen framkom det att antennkabeln var svår att skruva fast vid dess koppling. För att underlätta monteringen ritades en förbättrad design på flänshållaren med avkapade sidor som visas i Figur 28. Konstruktionen som utvecklades förenklade av- och påmontering av antennkabeln.



Figur 28: Utvecklad design på flänshållaren med avkapade sidor.

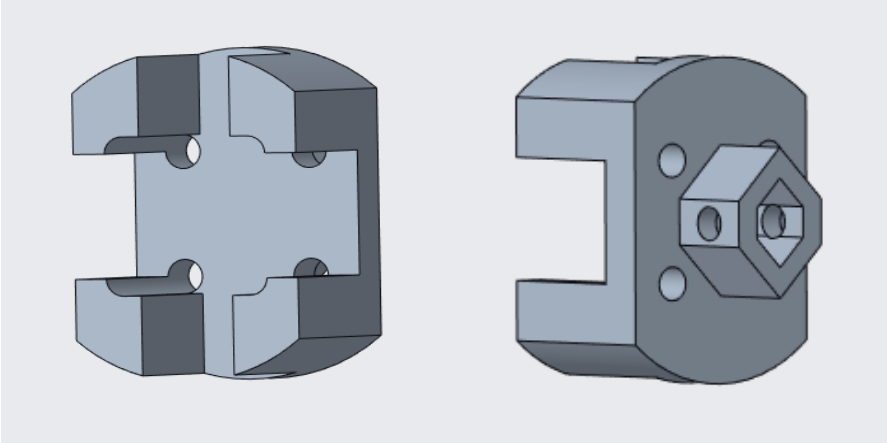
Universalhållare:

För att montera antennfästena till flänshållaren ovan skapades en universalhållare (se Figur 29). Universalhållaren skruvades fast med samma bultar som aluminiumflänsen och flänshållaren. En koppling längst fram på universalhållaren, i form av en hexagon, skapades för att kunna byta mellan de olika antennfästena med endast en bult. Formen på fästet i mitten togs fram genom att göra en så stor kvadrat som möjligt (röd streckad) utan att vara i vägen för bultarna och dess muttrar som går genom hållaren. Därefter kapades hörnen på sidorna för att ge en platt yta mot M3 bulten och dess mutter som fäster antennfästet genom kopplingen.



Figur 29: Första designen på universalhållaren med plats för antennkabel på baksidan och ett universalfäste framtill.

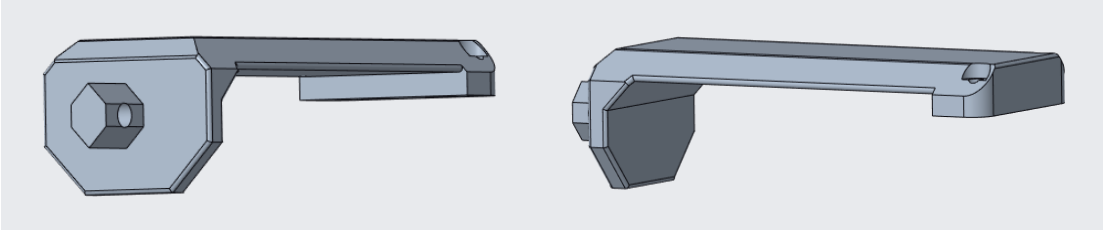
Även universalhållarens sidor kapades bort i en ny design som visas Figur 30. Anledningen till att den utvecklade designens avkapade sidor togs fram var för att kunna komma åt med skruvverktyg till antennkabelns koppling.



Figur 30: Utvecklad design på universalhållaren med avkapade sidor.

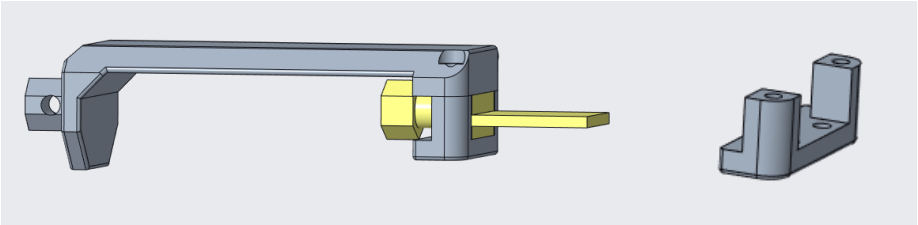
Antennfäste:

Grunden för antennfästet (se Figur 31) skapades till den gula antennen. På baksidan ritades motsvarande form från universalhållarens framsida, med en tolerans på 0,1 mm på alla sidor, för att möjliggöra montering av antennhållaren. Antennfästet konstruerades för att fästa antennen längst ut och göra plats för antennkabeln bakom antennen. Varje antenn fick ett eget specifikt antennfäste som de monterades fast i och för att byta mellan olika antenner för mätningar kopplades hela antennfästet bort från universalhållaren med endast en bult och mutter. Som tidigare visades i Figur 26, monteras antennfästet på universalhållaren.



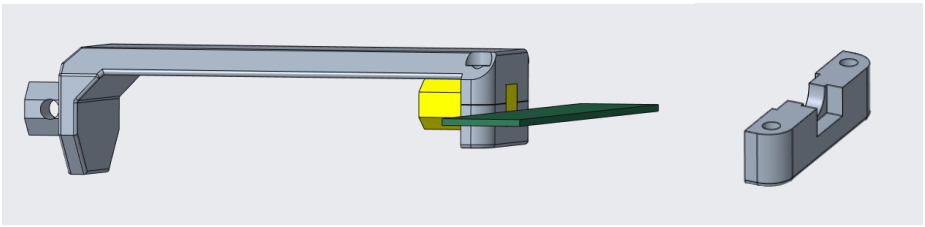
Figur 31: Antennfäste grunddesign.

En bottendel konstruerades för att fästa den gula antennen och skruvades fast med M2-bult och mutter i antennfästet (se Figur 32). Genom att mäta den gula antennen med skjutmått skapades bottendelens passform runt antennen. Bottendelen spänner fast antennen och håller den stabilt på plats. En assembly skapades i Creo Parametrics för att säkerställa att antennen placerades på korrekt höjd i antennfästet och därmed roterade korrekt kring roll-motorns rotationspunkt.



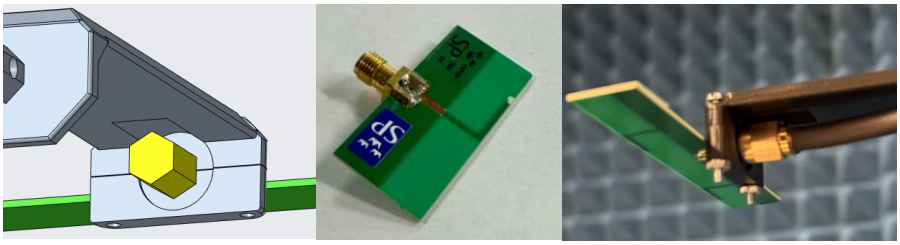
Figur 32: Antennfäste för den gula antennen med bottendel separat till höger.

För att fästa den gröna breda antennen användes samma bas till antennfästet som för den gula antennen. Genom att göra om designen längst fram på antennfästet, passades den gröna antennen in tillsammans med en ny anpassad bottendel som spänner fast antennen likt tidigare lösning (se Figur 33). På samma sätt som tidigare så användes assembly för att kontrollera att antennen hamnar i rätt höjd kring rotationspunkten.



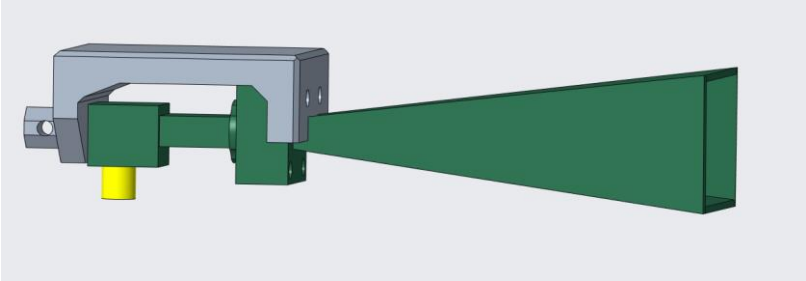
Figur 33: Antennfäste för den gröna breda antennen med bottendel separat till höger.

I CAD-modellen konstruerades antennerna med antennkabelns fäste monterat baktill, vilket i figurerna 25, 26, 32 och 33 representeras av den gula sexkantiga delen. I mitten av Figur 34 visas hur antennen ser ut när kabeln ej är monterad med dess synliga gängor och till höger när kabeln är monterad. Den gröna breda antennen hade en kortare koppling till antennkabeln på baksidan, alltså inte lika långt från gängorna till antennen, vilket resulterade i att tjockleken på väggen längst fram på antennfästet var för tjockt för att kunna gänga i antennkabeln hela vägen i botten. Problemet löstes genom att skapa en nedsänkning på insidan av väggen (se Figur 34 till vänster), vilket gjorde det möjligt att gänga in antennkabeln hela vägen. Fästet som skapades till den gröna breda antennen användes även till den gröna smala antennen.



Figur 34: Till vänster visas utvecklad design med nedsänkning på insidan av antennfästet, i mitten syns antenn utan antennkabel, till höger antenn med antennkabel monterad.

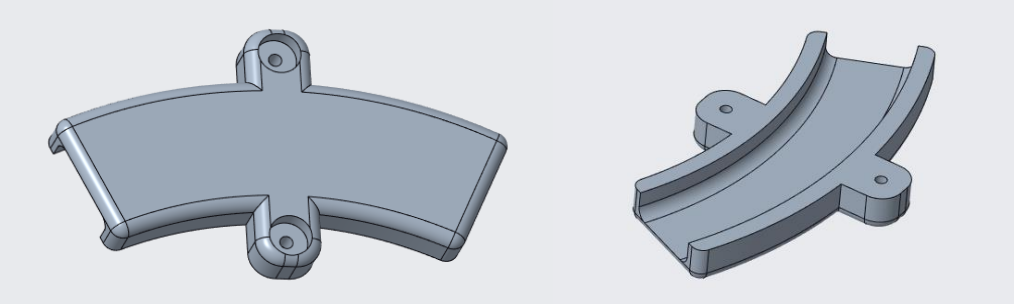
En ny design togs fram för att möjliggöra infästning av hornantennen (se Figur 35). Hornantennen är större än de tidigare antennerna och har en större vikt. Fästet för antennkabeln är placerat på undersidan av antennen vilket skapade möjlighet för att flytta bak antennen i fästet för att minska momentet på hela antennhållaren. Antennfästet har samma koppling till universalhållaren och i grunden samma design som tidigare antennfästen. Skillnaden för denna var att tjockleken på konstruktionen ökades för att klara av vikten från antennen. Hornantennens redan förborrade hål användes för att fästa antennen med M3 bult och mutter vilket resulterade i att en bottendel inte krävdes för att hålla antennen på plats. För att kontrollera alla mått till den första designen på fästet 3D-printades delen i 10 procents ifyllnad för att minimera spill. Efter några få justeringar på konstruktionen 3D-printades antennhållaren med 90 procent ifyllnad för att öka hållfastheten på grund av antennens vikt. Hela hållaren med hornantennen monterad illustreras i Figur 35.



Figur 35: Antennfäste för hornantennen.

5.1.11 Sladdlist

Sladdlisten togs fram för att fästa kablarna från azimuth-motorn mot bottenplattan av konstruktionen och förhindra att de är i vägen för de rörliga kablarna från roll-motorn under rotation (se Figur 36). Sladdlisten monteras med två träskruvar ner i bottenplattan med sladdarna iförda genom utrymmet på undersidan.

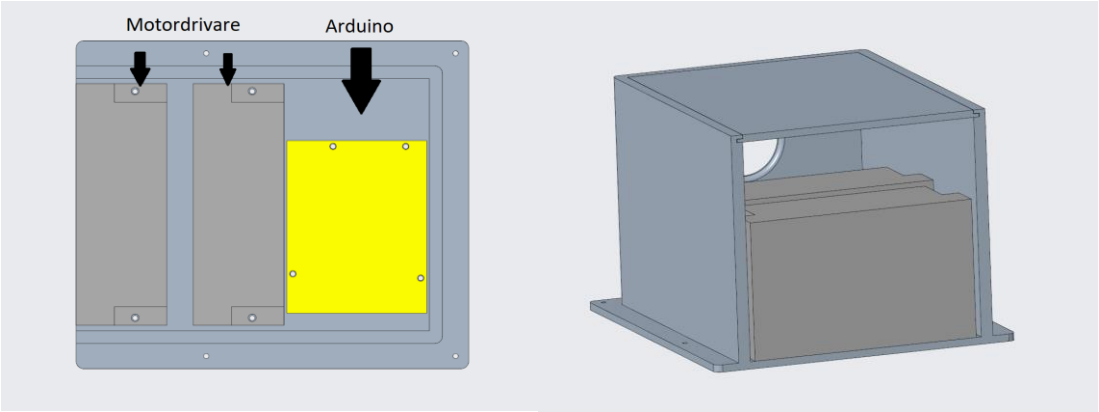


Figur 36: Sladdlist från ovan till vänster och underifrån till höger.

5.1.12 Elcentral

En elcentral konstruerades för att vid färdigställning av konstruktionen kunna gömma undan alla tekniska drivkomponenter, sladdar och breadboard. Resultatet blev en konstruktion med ett mer stilrent utseende. Första konceptet gjordes med endast en motordrivare där den andra motordrivaren var en mindre variant som sattes bredvid arduinon. Då två stora motordrivare till slut krävdes så blev den slutgiltiga elcentralen som Figur 37. Genom hålet på framsidan dras de två kabelstrumporna med tillhörande kablar till stegmotorerna in i elcentralen. Elcentralen är öppen på baksidan för att ventilera eventuell värmeökning från motordrivarnas kylflänsar. Strömkabel dras in från vänster sida sett i Figur 37 (till vänster) och input till arduinon nerifrån sett. Högersidan av figuren visar den slutgiltiga lådan i Creo från den öppna baksidan. Kraven på lådan var endast att komponenterna skulle passa in på ett relativt lätt sätt. Lockets passform blev något precist men medger fortfarande enkel demontering och montering utan större

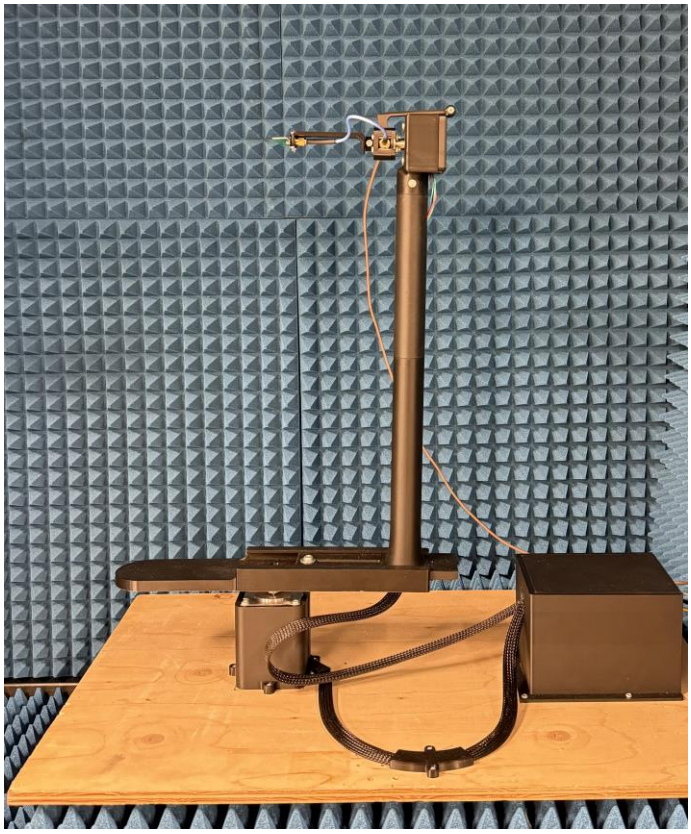
motstånd. Inför montering av locket krävs det att samtliga kablar leds ner i lådan så att de inte kläms.



Figur 37: Elcentralen med tillhörande komponenter monterade i Creo m.h.a Assembly.

5.1.13 Slutgiltig konstruktion

Figur 38 visar en bild på hur den verkliga konstruktionen blev när den var färdigställd inuti kammaren. Här kunde verkliga tester göras i en isolerad miljö. Kammaren är inklädd i högfrekventa strålningsabsorbenter med en höjd på 5 cm.

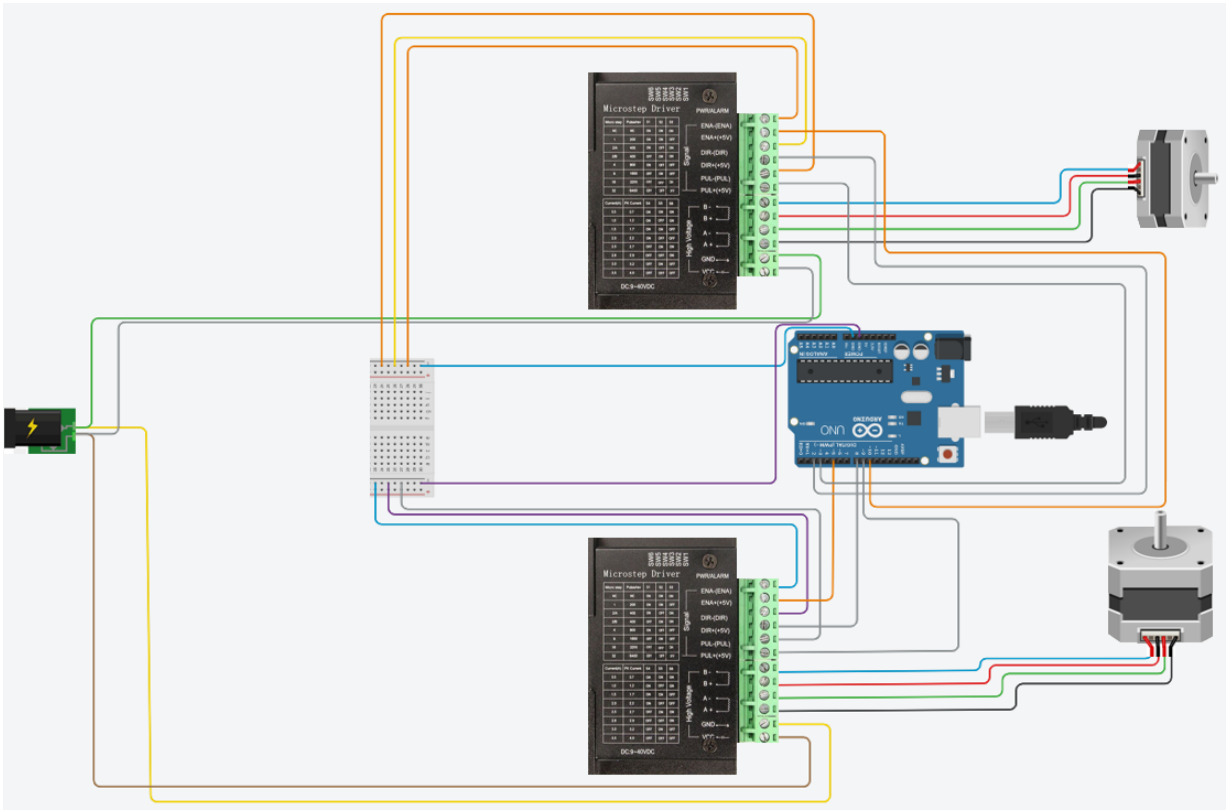


Figur 38: Hela antennhållaren i isolerad kammare.

5.2 Kopplingsschema

För att koppla ihop alla komponenter som styr konstruktionen användes komponenternas datablad och tillhörande beskrivningar. Ett kopplingsschema för hela systemet skapades och slutfördes enligt Figur 39. För att minimera lådan till elcentralen kapades en breadboard. Storleken på breadboarden anpassades till antalet sladdar som behövdes monteras på den.

Kopplingsschemat visar azimuth-motorn nederst i högra hörnet och roll-motorn överst i högra hörnet med tillhörande motordrivare till vänster om motorerna. Motordrivarna kopplas till Arduinon för styrning och till breadboarden för jord. Gemensam jord från motordrivarna går från Arduinon genom breadboarden. Ström och jord till motordrivarna kopplas in från en transformator, symbolen längst till vänster i figuren, som omvandlar 230V till 12V. All elektronik från kopplingsschemat ryms i elcentralen bortsett från motorerna och dess sladdar som går ut från elcentralen i kabelstrumpor genom ett hål till respektive motorhållare i konstruktionen. Färgerna på samtliga sladdar (förutom vita sladdar som är grå i kopplingsschemat) stämmer överens med sladdarna som är kopplade i verkligheten. Orsaken till att de vita sladdarna sattes som gråa var för att synas bättre i bilden för kopplingsschemat.



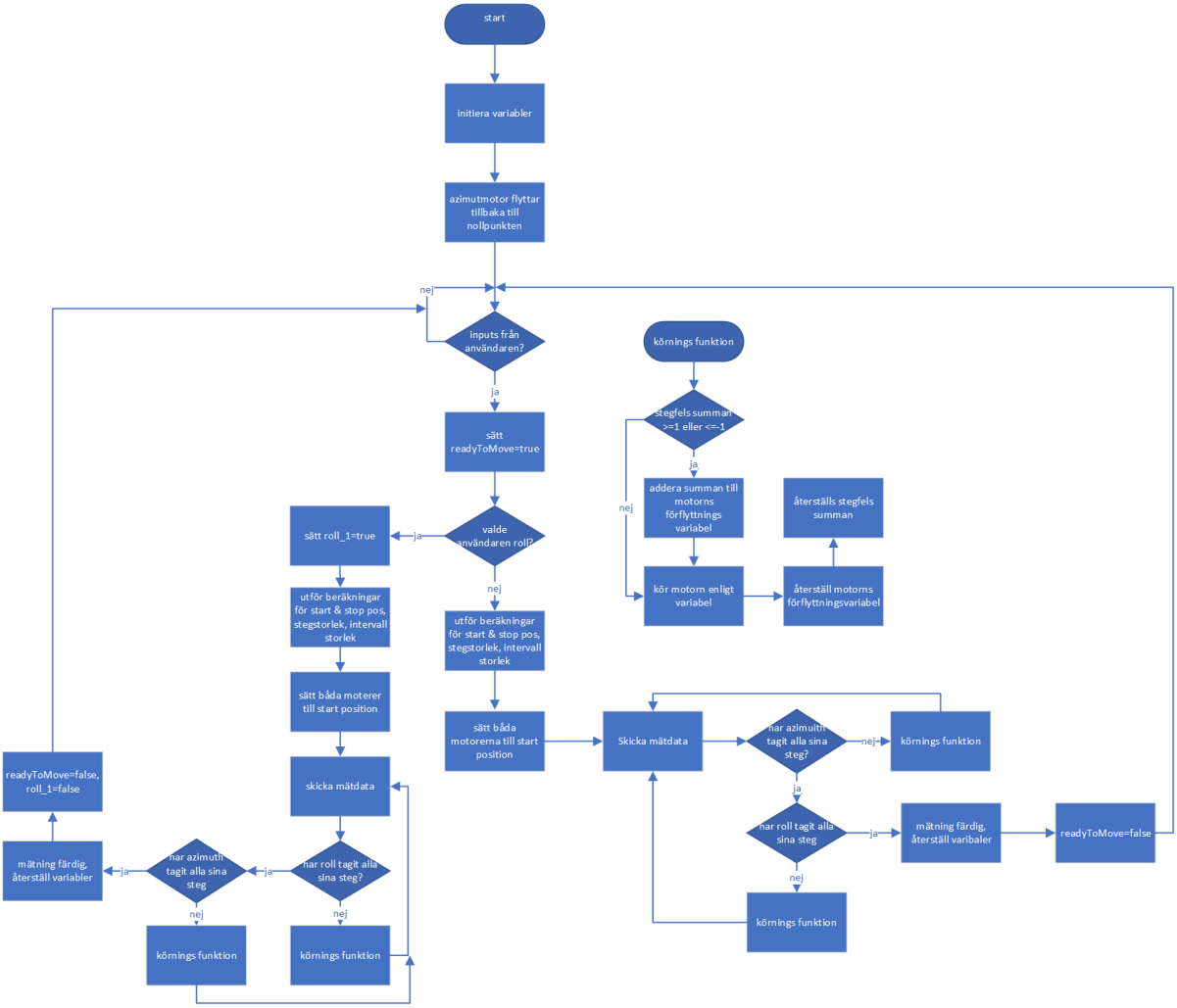
Figur 39: Kopplingsschema över hela systemet.

5.3 Programmering

Under resultatet för programmering förklaras först flödesschema för respektive del och därefter funktioner, matematik och uppbyggnad för koderna, både grundläggande och mer specifikt.

5.3.1 Arduino Flödesschema

Flödesschemat för arduinokoden visas i Figur 40.



Figur 40: Flödesschema Arduino.

5.3.2 Förklaring av Arduinokod

Programmet som styr motorerna är utformat för att ta emot indata från användaren, utifrån vilka ett mätintervall definieras. Motorerna utför därefter sina rörelser inom det angivna intervallet enligt de parametrar som specificeras av användaren. Både azimuth- och rollmotorerna är stegmotorer med ett standardsteg på 1,8°. För att uppnå önskad upplösning i mätdata krävs en mindre stegvinkel. Därför implementerades mikrostegning för att förbättra upplösningen.

Mikrostegning delar upp varje standardsteg i mindre delsteg genom att modifiera styrpulserna till motorerna. Projektet tillämpar en mikrostegsindelning på 1/32, vilket innebär att standardsteget på 1,8° delas upp i 32 delsteg. Resultatet blir en vinkelupplösning på 0,05625°.

En begränsade faktor som programmet måste ta hänsyn till är att azimuth-motorn inte får rotera mer än 180° åt vare sig medurs eller moturs från nollpunkten eftersom kablarna kopplade till rollmotorn inte klarar av för stora rotationsrörelser.

Användarens indata anges i grader, medan stegmotorerna opererar i mikrosteg, vilket kräver en konvertering mellan dessa enheter. Dessutom utgår motorernas koordinatsystem från nollpunkten, där en motursrotation upp till 180° representeras som negativa värden, och en medurs rotation upp till 180° som positiva värden. Nollpunkten är det läget som användaren manuellt roterar vridplattan till innan strömförsörjning av systemet.

För att hantera fall där användaren exempelvis anger ett mätintervall från 170° till 200° krävs en särskild logik i programmet. Enligt motorernas koordinatsystem motsvarar detta ett intervall från 170/0,05625 steg till -(160/0,05625) steg. Programmet måste tolka och korrigera denna övergång korrekt för att undvika överrotation och potentiella mekaniska problem.

Hanteringen av variabler och motorernas positionering har förenklats genom att definiera fyra olika fall, där start- och stoppvärdena för mätintervallen avgör vilket fall som ska tillämpas.

- **Fall 1**: 0° < start < 180°, och stop < 180°
- **Fall 2**: start < 180°, och stop > 180°
- **Fall 3**: start > 180°, och stop < 180°
- **Fall 4**: start > 180°, och stop > 180°

5.3.2.1 Iteration av mätpunkter

Programmet fungerar så att när en körning startar så väntar koden på att få initiala värden från användaren genom GUI:n som styrs av python-programmet. När programmet får dessa värden så finns det två if-satser som kollar ifall rollfunktionen ska användas eller inte.

Om rollfunktionen inte önskas så börjar programmet översätta de värden som användaren har valt som inputs till användbara värden som motorerna kan använda. Motorerna ställs då till sina startpositioner och sen skickas ett mätvärde till Python. En for-loop itererar azimuth-motorn och efter varje steg stannar motorn och ett mätvärde skickas till Python. När for-loopen har gått färdigt förflyttas rollmotorn ett steg och rutinen startar om. Denna rutin kommer att köras till

rollmotorn har nått sitt stoppläge. Mätningen är då färdig och programmet återställs. Programmet kommer köra denna rutin på ungefär samma sätt för varje fall men det finns små skillnader. Skillnaderna är för azimuth-motorn medan rollmotorn körs på samma sätt oavsett fall.

Ingen rollfunktion
Fall 1 fungerar enligt ovan, programmet körs som vanligt.
Fall 2 kommer köra azimuth-motorn till sin startposition och iterera fram till att motorn är 180 grader från nollpunkten, då kommer azimuth-motorn roteras 360 grader i motsatt håll av iterationerna och därifrån kommer motorn fortsätta iterera. När stoppläget är nått så startar processen om.
Fall 3 skiljer sig inte från fall 1 i körningen men körningsintervallet är på den negativa halvan.
Fall 4 används när mätningen startar på negativa halvan och går över nollpunkten för att senare fortsätta in på den positiva halvan.

Med rollfunktion
Då användaren väljer rollfunktionen kommer programmet att börja iterera roll-motorn i stället för azimuth-motorn och efter att roll-motorn har nått sitt stoppläge så kommer azimuth-motorn att förflytta sig ett steg, motorerna byter alltså roller. Programmet hanterar körningen av roll-motorn på samma sätt oavsett variation i körningsintervall men för azimuth-motorn används fortfarande fall 1-4.

5.3.2.2 Styrning av motorerna

För att kunna styra motorerna korrekt måste de angivna vinkelintervallen och stegstorlekarna som användaren matar in via interfacet översättas till motsvarande motordrivsteg. Vinkelvärdena divideras med motorns specifika upplösning, 0,05625° per steg, vilket motsvarar hur mycket motorn roterar för varje enskilt steg (Se Figur 41).

```
float start_position = intervall_motor1_start / 0.05625;
float stop_position = intervall_motor1_stop / 0.05625;
float step_size = stegstorlek / 0.05625;
int round_start_position = round(start_position);
int round_stop_position = round(stop_position);
int round_step_size = ceil(step_size);
//motor2
float steg_storlek_motor2 = stegstorlek2 / 0.05625;
float start_position_motor2 = intervall_motor2_start/0.05625;
float stop_position_motor2 = intervall_motor2_stop/0.05625;
int round_start_position_motor2 = round(start_position_motor2);
int round_stop_position_motor2 = round(stop_position_motor2);
int round_step_size_motor2 = ceil(steg_storlek_motor2);
```

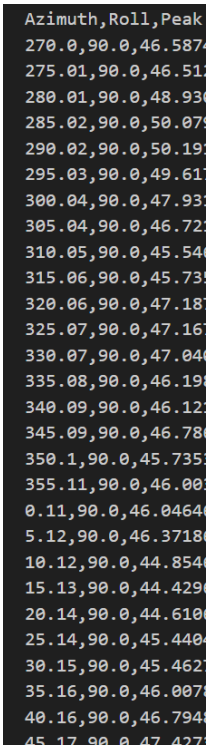
Figur 41: Matematik för arduino.

Motorerna fungerar på så sätt att om ett kommando ges för att förflytta motorn 10°, vilket motsvarar 177,7 mikrosteg, kommer motorn i praktiken att förflytta sig 177 mikrosteg. Detta motsvarar en faktisk rörelse på 9,96°, vilket innebär att 0,04° förloras eftersom motorn enbart kan utföra hela mikrosteg. Problemet som uppstår är att detta fel ackumuleras vid varje iteration, vilket leder till ett ökande positioneringsfel över tid.

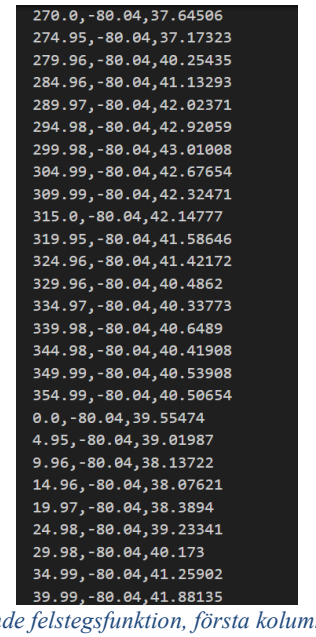
En möjlig lösning för att minska detta fel är att avrunda antalet steg till närmaste heltal. Detta reducerar felet men kan fortfarande leda till en avvikelse på upp till 0,5 steg per förflyttning. I

Figur 42 visas azimuth-motorns position som det första värdet på varje rad. Det framgår tydligt av figuren hur positioneringsfelet ökar successivt i takt med att fler iterationer körs, då decimaldelen i positionsvärdena växer.

Figur 43 visar motorns position efter att den föreslagna lösningen tillämpats. Enligt denna figur är positionen betydligt mer exakt och uppvisar inte längre ett konstant ökande fel.



Figur 42: Ökande stegfel visat från CSV-filen, första kolumnens decimaler illustrerar detta



Figur 43: Värden med fungerande felstegsfunktion, första kolumnens decimaler illustrerar detta

För att hantera det ackumulerade positionsfelet skapades särskilda funktioner för att korrigera motorns position. Totalt utvecklades tre funktioner, vilka används på olika ställen i koden beroende på sammanhang. Funktionen som visas i Figur 44 används i *Azimuth – Fall 4*. Denna funktion fungerar genom att addera det önskade stegvärdet med det ackumulerade felet, och därefter avrunda resultatet nedåt till närmaste heltal. Skillnaden mellan det avrundade och det ursprungliga (icke-avrundade) värdet sparas och adderas nästa gång funktionen anropas. Motorn förflyttas sedan med ett steg som motsvarar det avrundade värdet.

```
// Ersätter moveMotorWithCompensation
void moveMotorWithErrorTracking(AccelStepper& motor, float& accumulated_error, float step_size) {
  float to_move = step_size + accumulated_error;
  int full_steps = floor(to_move);  // Heltalssteg
  accumulated_error = to_move - full_steps;  // Spara decimaldelen till nästa gång

  if (full_steps != 0) {
    moveMotor(motor, full_steps);
  }
}
```

Figur 44: Stegfelskompensation fall 4.

För fall 1-3 skapades funktionen ”moveMotorWithCompensation”, som framgår i Figur 45. Funktionen arbetar med att förminska stegfel på samma sätt men har en extra funktion för att hantera fall 2. När motorns position går över 180° (negativa värden), så adderar funktionen 6400 på variablerna, vilket motsvarar ett helt varv av 360°. Exempel: azimuth-motorn tar ett steg över 180° och stannar på -178°, vilket ger -3164 mikrosteg. Efter att 6400 mikrosteg adderas (-3164+6400) ger det en position på 3236 mikrosteg vilket motsvarar 182°. Genom att addera 6400 mikrosteg på variablerna så kan funktionen fortsätta arbeta med positiva värden.

```
void moveMotorWithCompensation(AccelStepper& motor, float& exact_position, float& accumulated_error, float desired_step_size) {
  float current_pos = motor.currentPosition();

  // Korrigera för negativa positioner (cirkulär logik: 0-6400 steg för 0-360°)
  if (current_pos < 0) {
    current_pos += 6400;
    hej = true;
  }
  if (exact_position < 0) {
    exact_position += 6400;
  }


  exact_position += desired_step_size;
  float target_with_error = exact_position + accumulated_error;

  int steps_to_take = round(target_with_error - current_pos);
  accumulated_error = target_with_error - (current_pos + steps_to_take);

  moveMotor(motor, steps_to_take);
}
```

Figur 45: Stegfelskompensation fall 1-3.

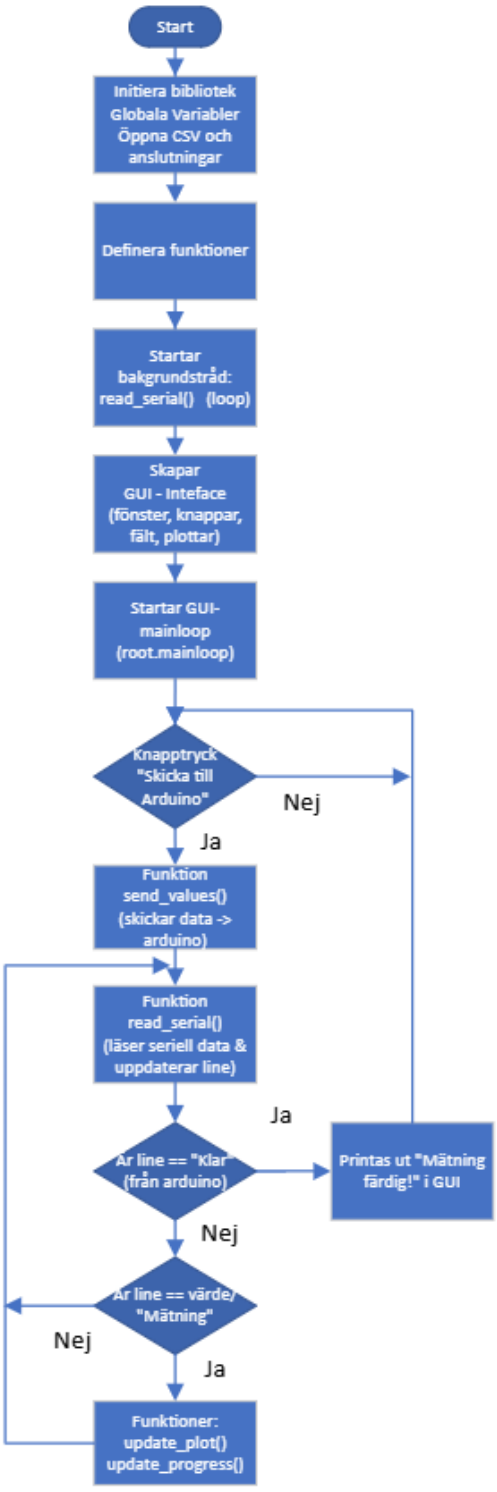
För att kompensera positionsfelet för rollmotorn skapades funktionen ”moveMotorWithErrorTrackingMotor2” som visas i Figur 46. Funktionen fungerar genom att konstant addera eller subtrahera differensen mellan det önskade steget och avrundade steget. Ett exempel är när det önskade steget är 17.8 mikrosteg och det avrundade steget är 18 mikrosteg. Enligt funktionen ger de oss ett kontinuerligt steg fel på 0.2 mikrosteg varje gång motorn förflyttar sig. Stegfelet adderas till en variabel varje iteration och när variabeln når värdet av ett heltal kompenseras motorn med att ta ett extra eller ett mindre steg. Funktionen återställer sedan variabeln och fortsätter.

```
void moveMotorWithErrorTrackingMotor2(AccelStepper &motor, float &accumulated_error, float step_size_steps) {
  int rounded_steps = round(step_size_steps);           // avrundade steg
  float error = step_size_steps - rounded_steps;        // differens från exakt värde
  accumulated_error += error;                           // lägg till fel till ackumulerad

  // Om tillräckligt mycket fel har ackumulerats – justera
  if (accumulated_error <= -1.0) {
    rounded_steps -= 1; // ta ett extra steg i negativ riktning
    accumulated_error += 1.0; // återställ fel
  }
  if(accumulated_error>=1.0){
    rounded_steps +=1;
    accumulated_error -=1;
  }

  // Garantera att vi rör oss i negativ riktning
  int steps_to_move = -abs(rounded_steps);

  moveMotor(motor, steps_to_move);  // Använd befintlig funktion
}
```

Figur 46: Stegfelskompensation roll-motor.

5.3.3 Python Flödesschema

Flödesschemat för pythonkoden visas i Figur 47.



Figur 47: Flödesschema Python

5.3.4 Förklaring av Pythonkod

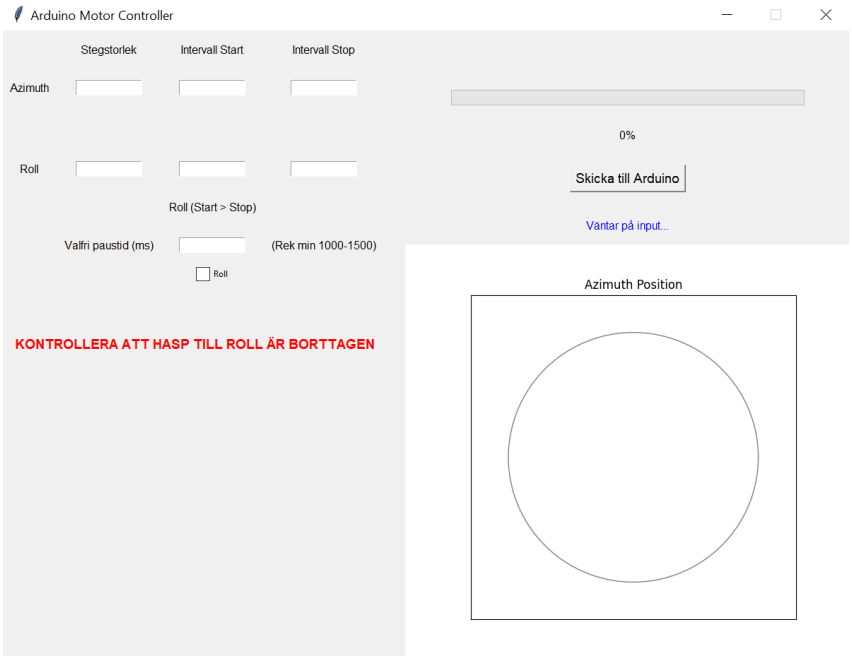
Nedan sker en genomgång på pythonkodens specifika delar och hur den är uppbyggd för att systemet ska fungera korrekt. Först sker en genomgång av interfacen. Därefter beskrivs huvudfunktionerna som är: sändning av data, avläsning av data, mottagen strålningseffekt, uppdatering i CSV samt plottning i GUI. Avsnittet avslutas med allmän information om koden i form av användna bibliotek och stödprogram.

5.3.4.1 Interface

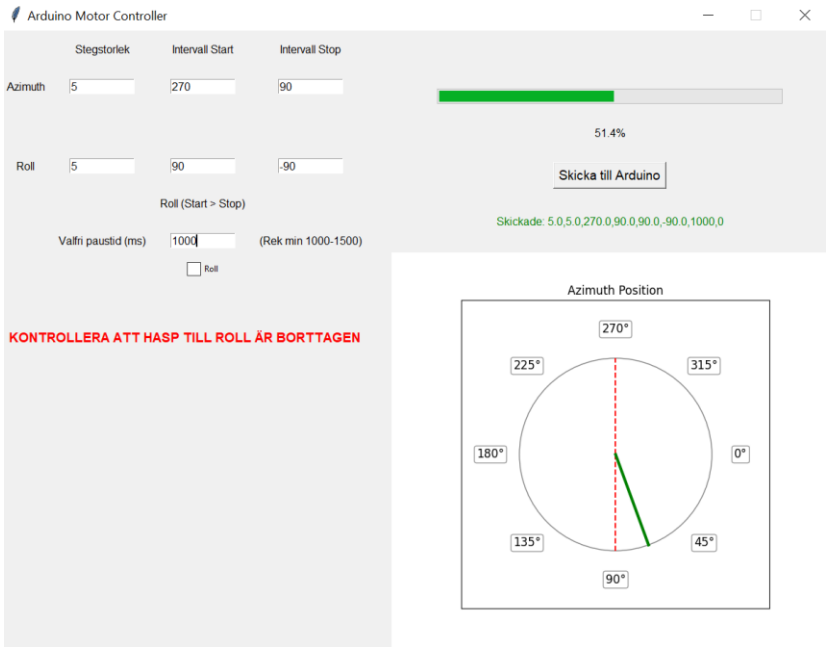
Genom en GUI i Python matas vald stegstorlek samt start- och slutposition för respektive stegmotor in (se Figur 48). Därefter kan man välja valfri paustid som sätts in i varje rörelse som motorerna gör. Det rekommenderas en paustid runt 1-1,5 sekunder här för att systemet ska stabiliseras innan nästa rörelse görs. Enhet för stegstorlek är grader per steg, enhet för intervall är grader och för paustid är millisekunder. Interfacen har även en ”Roll”-funktion som fungerar så att den vänder på systemets funktionalitet. Utan roll-funktion så vrids roll-motorn till startpositionen och därefter går azimuth-motorn från start till stopp med valt stegintervall. Processen itereras för varje roll-vinkel.

Ifall checkboxen för roll-funktionen i stället är markerad, så går azimuth-motorn till vald vinkel och itererar varje roll-vinkel vid den satta vinkeln. Alltså på samma sätt som utan roll-funktionen fast tvärt om. För att skicka i väg ordern till styrkoden i arduinon så måste man trycka på knappen ”Skicka till Arduino”.

När processen är i gång så plottas den aktuella positionen på azimuth-motorn ut på skärmen som visas med ett grönt sträck i cirkeldiagrammet nederst i Figur 49. Här plottas start- och stopposition ut som rödstreckade linjer och den aktuella positionen uppdateras med den gröna linjen. Ovanför knappen ”Skicka till Arduino” så visas dessutom en progressionsbar som uppdateras under processens gång. Ifall roll-funktionen inte är markerad så uppdateras plotten med uträknat procentsteg varje gång Azimuth-motorn når sin slutposition. Ifall roll-funktionen är markerad så uppdateras progressionsbaren varje gång Azimuth-motorn flyttar sig ett steg. Runt cirkeln visas dessutom etiketter på hur processen rör på sig i vinklar. Plottningen gör det möjligt för användaren att se den aktuella positionen för azimuth-motorn när konstruktionen är instängd i den isolerade kammaren. Det ger även möjlighet för användaren att se hur långt procentuellt av den totala mätningen det gått vid en viss tidpunkt.



Figur 48: GUI utan värden, innan drivning av system.



Figur 49: GUI med värden under mätning.

5.3.4.2 Funktioner

Pythonkoden är uppbyggd av fyra huvudfunktioner. Första funktionen som startar och styr hela systemet är ”send_values” och syns i Figur 50. Den fungerar så att den lägger in alla valda variabler i en egen lista och därefter skickar variabeln ”data” till arduinon via baud-raten 9600 på vald COM-port. På GUI:n visas efter sändning av värden, att datan har skickats eller att något blivit fel i ordern. I funktionen återställs även progressbarens gröna linje och procentsats från eventuell föregående mätning.

```python
def send_values():
    global intervall_start, intervall_stop, intervall_motor2_start, intervall_motor2_stop, stegstorlek2, stegstorlek, roll_valu
    try:
        stegstorlek = float(stegstorlek_entry.get())
        stegstorlek2 = float(stegstorlek2_entry.get())
        intervall_start = float(intervall_motor1_start_entry.get())
        intervall_stop = float(intervall_motor1_stop_entry.get())
        intervall_motor2_start = float(intervall_motor2_start_entry.get())
        intervall_motor2_stop = float(intervall_motor2_stop_entry.get())
        paus_tid = int(pause_entry.get())

        # Lägg till "Roll" i data (om roll_enabled är 1, lägg till den)
        roll_value = roll_enabled.get()  # 1 om markerad, 0 om inte markerad

        # Återställ progress-baren, texten till 0% och klar_label
        progress_bar["value"] = 0
        progress_label.config(text="0%")
        klar_label.config(text="")

        # Skapa data-strängen som inkluderar "roll_value"
        data = f"{stegstorlek},{stegstorlek2},{intervall_start},{intervall_stop},{intervall_motor2_start},{intervall_motor2_sto
        ser.write(data.encode())
        status_label.config(text=f"Skickade: {data}", foreground="green")
    except ValueError:
        status_label.config(text="Fel: Ange giltiga värden!", foreground="red")
```

Figur 50: Funktion send_values()

Den andra funktionen används för att uppdatera progressionsbaren och dess procentsats och illustreras i Figur 51. Här finns olika matematiska funktioner beroende på vilket intervall man satt som start eller stopp, eller ifall roll-funktionen är på. När roll = 0 så blir ”antal_procent_steg” absolutbeloppet av ”motor2start” subtraherat med ”motor2stop” delat med ”stegstorlek2” + 1.

```python
def update_progress():
    if roll_value == 1:
        if intervall_start >= 180 and intervall_stop < 180:
            antal_procent_steg = (360 - intervall_start + intervall_stop)/stegstorlek + 1
        else:
            antal_procent_steg = abs(intervall_start-intervall_stop)/ stegstorlek + 1
        storlek_procent_steg = 100 / antal_procent_steg
        progress_bar["value"] += storlek_procent_steg
        progress_label.config(text=f"{progress_bar['value']:.1f}%")
        root.update_idletasks()
    else:
        antal_procent_steg = abs(intervall_motor2_start - intervall_motor2_stop) / stegstorlek2 + 1
        storlek_procent_steg = 100 / antal_procent_steg
        progress_bar["value"] += storlek_procent_steg
        progress_label.config(text=f"{progress_bar['value']:.1f}%")
        root.update_idletasks()
```

Figur 51: Funktion update_progress()

Den tredje funktionen syns inte helt i Figur 52, men innefattar plottning av azimuth-motorns aktuella position i GUI:n. Då biblioteket numpy räknar cirkeln i programmet som en enhetscirkel så behövde en omvandling från mikrostegvärdena göras till 0-360°. Värdet som läses av från terminal går för 0-180° mellan 0 och 3200. Från 180-360° så kommer värdena in som -3200 till 0. Ekvationerna i if-satserna ger korrekt omvandling. Alltså ger ett mikrostegvärde på till exempel -1600 en vinkel på 270° vilket möjliggör korrekt uppdatering i cirkeldiagrammet. Då enhetscirkeln räknar graderna i moturs riktning och azimuth-motorn rör sig i medurs riktning så inverteras vinkeln för att stämma överens med den verkliga rörelsen.

```python
def update_plot(position):
    global motor1_position
    motor1_position = position

    # Omvandlar position till en vinkel mellan 0° och 360°:
    if position >= 0:
        angle = (position / 3200) * 180   # För positiva positioner (0 till 3200), omvandlas till 0° till 180°
    else:
        angle = (position / 3200) * 180 + 360  # För negativa positioner (-3200 till 0), omvandlas till 180° till 360°

    # För att säkerställa att vinkeln alltid ligger mellan 0° och 360°
    angle = angle % 360

    angle_rad = np.deg2rad(-angle)  # Invertera vinkeln för medursrotation

    ax.clear()
    ax.set_xlim(-1.6, 1.6)
    ax.set_ylim(-1.6, 1.6)

    ax.set_xticks([])  # Ta bort x-axelns tickmarks
    ax.set_yticks([])  # Ta bort y-axelns tickmarks
    ax.set_aspect('equal')
    ax.set_title("Azimuth Position")
```

Figur 52: Funktion update_plot()

Den sista och mest komplexa funktionen visas i Figur 53, och är den som läser av den seriella kommunikationen som skickas ut från arduinokoden. Denna väntar tills det kommit någon data tillbaka från arduinon och printar därefter ut den i pythonterminalen. Om datan som skickas säger ”KLAR” så är mätningen färdig och återställer ”motor_read_count”. Den skickar även ut i GUI:n att mätningen är färdig för användaren att se. Ifall datan är ett värde så checkar den ifall det är det första eller andra värdet som kommer in i ”line”. Då printsatserna från arduinon är satta så att azimuth-positionen alltid är först, och roll-positionen efter så fungerar beräkningen att med ”motor_read_count” delat med två inte har något rest, vise verse. Alltså är motor_read_count jämnt blir det ingen rest och är den udda blir det rest. Det bidrar till att hålla reda på vilket värde som innefattar vilken motors position. Azimuth-värde uppdaterar även cirkeldiagrammet för användaren att se i GUI:n. Längre ner i funktionen skickas kommandon till spektrumanalysatorn att ta ett peak-värde i amplitud och skicka tillbaka till pythonkoden. Detta sker endast där det satts in en print-sats i arduinokoden, ”Mätning”, visat i avsnitt under Arduinokod i Bilagor. Dessa värden printas ut i terminalen och skickas vidare till en CSV-fil för att senare kunna använda datan till annat. CSV-filen uppdateras rad för rad med (Azimith Pos (°), Roll Pos (°), Amplitud (dBm).

```python
def read_serial():
    global motor_read_count
    while True:
        if ser.in_waiting > 0:
            try:
                line = ser.readline().decode('utf-8').strip()
                print(f"Data arduino: {line}")

                if line == "KLAR":
                    root.after(0, lambda: klar_label.config(text="Mätning färdig!", foreground="green"))
                    motor_read_count = 0  # Återställ när mätningen är klar

                elif line.isdigit() or (line.startswith("-") and line[1:].isdigit()):
                    position = int(line)
                    angle = (position / 3200) * 180 if position >= 0 else 360 + (position / 3200) * 180

                    if motor_read_count % 2 == 0:
                        azimuth_deg = round(angle, 2)
                        current_measurement["azimuth"] = azimuth_deg
                        print(f"Azimuth vinkel: {azimuth_deg}°")
                        root.after(0, update_plot, position)
                    else:
                        if angle > 180:
                            angle -= 360
                        roll_deg = round(angle, 2)
                        current_measurement["roll"] = roll_deg
                        print(f"Roll vinkel: {roll_deg}°")
                    motor_read_count += 1

                elif line == "Mätning":
```

Figur 53: Funktion read_serial()

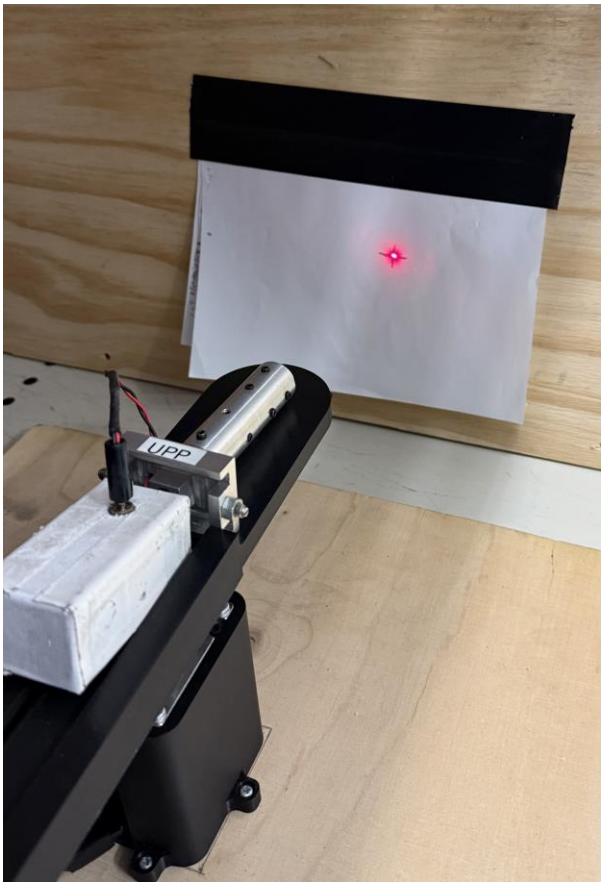
5.3.4.3 Bibliotek och externa moduler

De importerade biblioteken som användes i koden var tkinter, serial, time, threading, matplotlib, numpy, pyvisa, csv, datetime och os. All plottning i GUI skapades med matplotlib.

För att kommunikationen mellan python och spektrumanalysatorn skulle kunna ske så behövdes VISA laddas ner via biblioteket pyvisa. Det krävdes dessutom att ett program från National Instruments laddades ner till den aktuella styrenheten, som i detta fall var en dator. Versionen från National Instruments hemsida som användes i projektet var NI 488.2 (version 25.0).

5.4 Kalibrering av system

För att validera systemets funktion och träffsäkerhet genomfördes ett test med en laser. Lasern monterades på vridbordets främre del riktat utåt (se Figur 54). Azimuth-motorn roterades 180° medurs från startpositionen och laserns position, vid 180°, markerades med ett kryss på ett papper enligt bilden. Därefter roterades motorn moturs till 180° och lasern stannade på samma punkt. Om motorns mikrosteg eller beräkningarna i koden hade varit fel hade lasern stannat på två olika punkter på pappret. Det medför att motorns mikrosteg på 0,05625 stämmer enligt specifikation. Laserns rödpunkt i Figur 54 är större än storleken i verkligheten.



Figur 54: Bilden visar uppställning med laser för att validera noggrannheten i rotationen av vridbordet.

6. VALIDERING AV STRÅLNINGSDIAGRAM

I valideringsdelen diskuteras resultat från strålningsdiagrammen som framställdes utifrån mätdata. För att hålla jämförelse mellan antennmätningar och simuleringar så kort och sammanhängande som möjligt användes samma indata till stegmotorer för samtliga mätningar. Dessa indata visas på vänster sida i Figur 55. Startintervall på 270° motsvarar i respektive figur på strålningsdiagrammen -90°, vilket är anledningen till att intervallen går mellan -90° och 90° i Figur 57 - 63. I Figur 56 användes en tidigare variant av matlabkoden vilket är anledningen till att intervallen där var 0 till 180°. Intervallet i figuren har alltså sitt nolläge (antenner rikade mot varandra) där vinkeln är 90°. De antenner som i slutändan valdes att analyseras och jämföras med simuleringarna var:

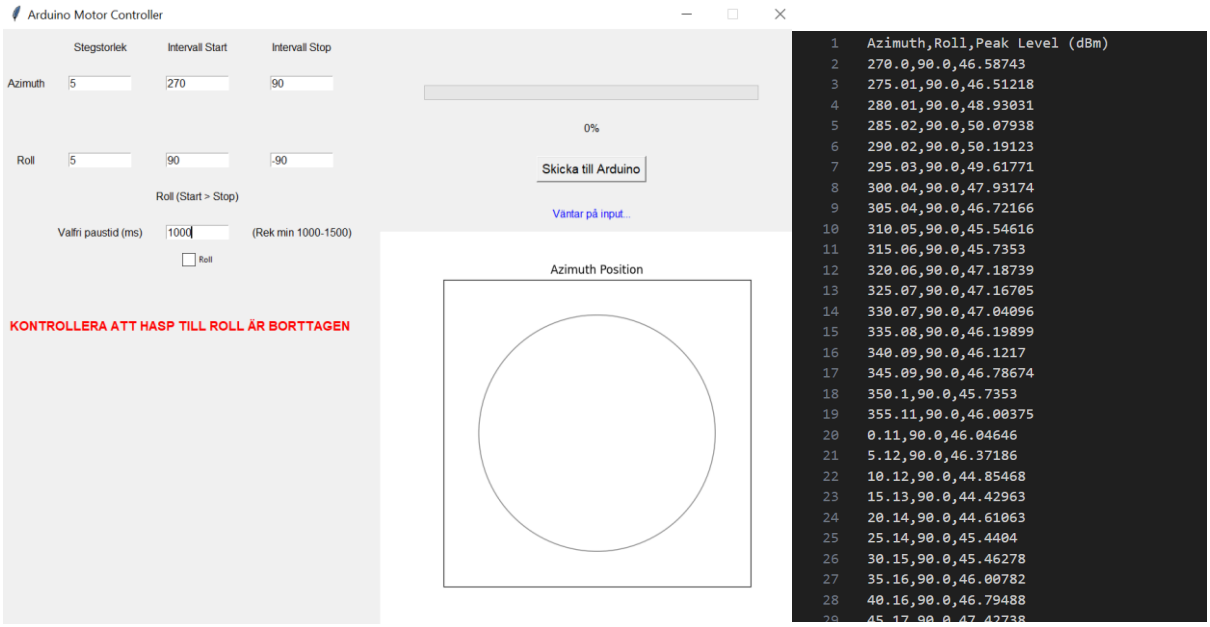
- den smala gröna antennen (“stacked dipole over ground plane”)
- den breda gröna antennen (“two stacked dipoles over ground plane”).

6.1 Allmänt om strålningsdiagrammen

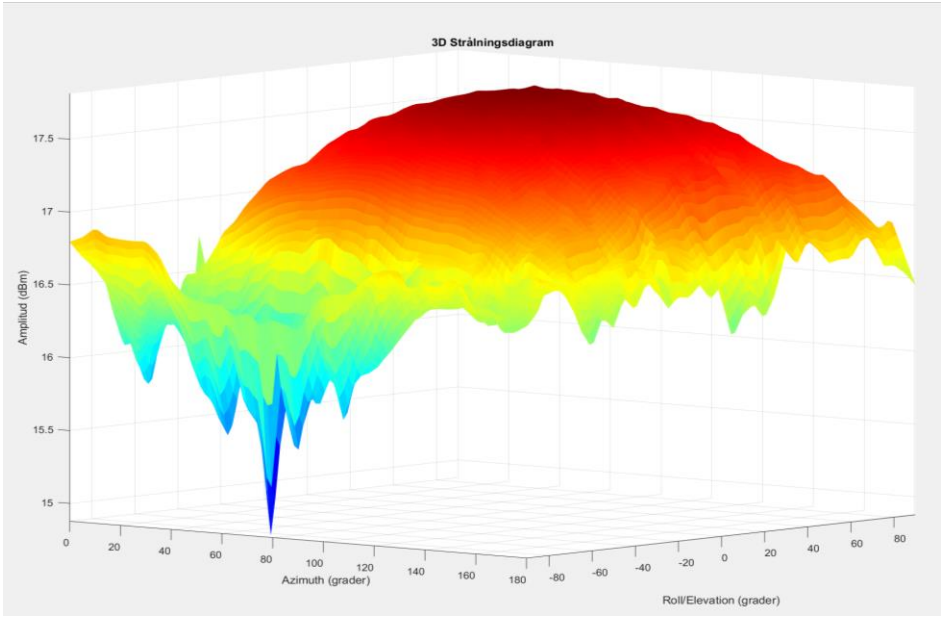
Med använd CSV-fil kunde därefter diagram i matlab skapas som visualiserade hur mottagen strålningseffekt varierade under mätningen. Ett exempel på en mätning där parametrarna i Figur 55 skickades in, gav utdata som visas på högersida i samma figur. Här visas tydligt hur prototypen itererar från 270°, upp till 360° (0°), och sedan vidare till 90°. Applicering av exemplets data i matlabkoden som visas i avsnitt Bilaga 2, gav ett strålningsdiagram som illustreras i Figur 56. Strålningsdiagrammet visar tydligt hur mätningen med en antennvariant gav den maximala effekten när sändar- och mottagarantenn var riktade rakt mot varandra. Effektens amplitud avtog ju längre ifrån parallell vinkel den var. De räta vinklarna mot signalantennen gav generellt lägst amplitud. Här skickades effekten på 14 dBm ut från signalgeneratorn med frekvensen 5.9 GHz.

I Figur 56 till Figur 61 och Figur 63 visualiseras uppmätta antenners strålningsdiagram i olika miljöer: utan isolering (i kontorsmiljö), med halv isolering (i delvis dämpad kammare) och med full isolering (i heldämpad kammare).

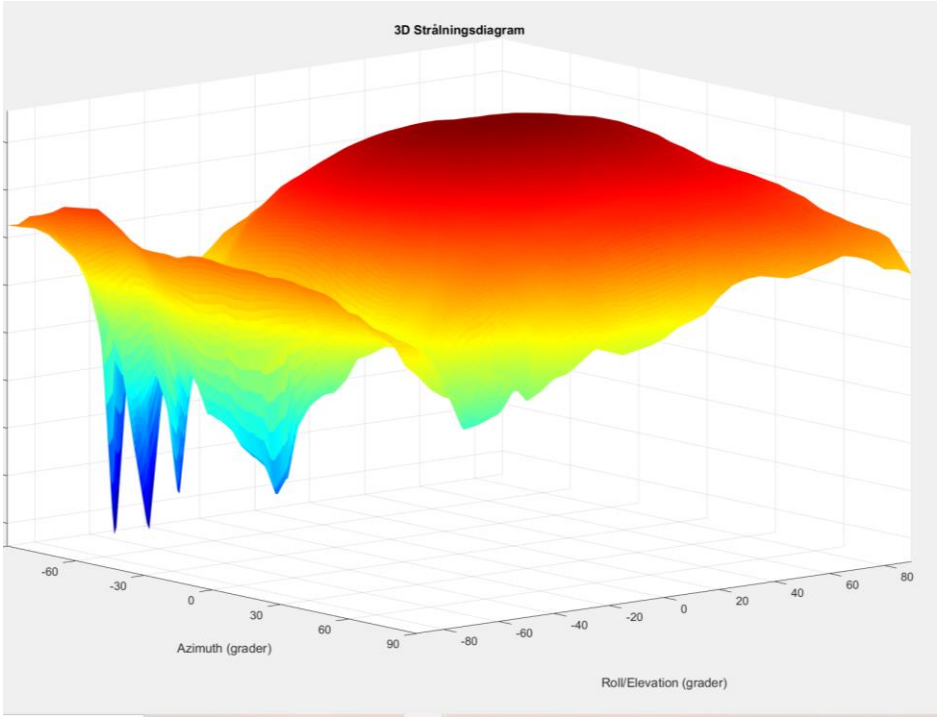
Viktigt här att förstå att ordet heldämpad eller ”fullt isolerad” i rapporten innebär absorbenter på alla väggar, golv och tak i kammaren. Absorbenterna sattes tillfälligt upp med tejp och frigolitblock. Vid en komplett installation ska absorbenter skäras ut och limmas upp på väggar, golv och tak, vilket inte utfördes under detta arbetet.



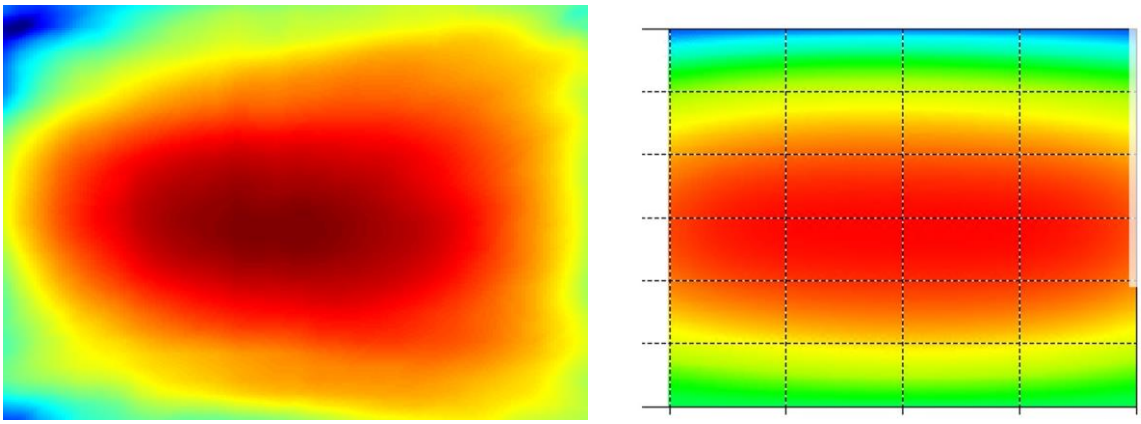
Figur 55: Input-värden till samtliga mätningar till vänster, början av mätdatan till första mätningen till höger.



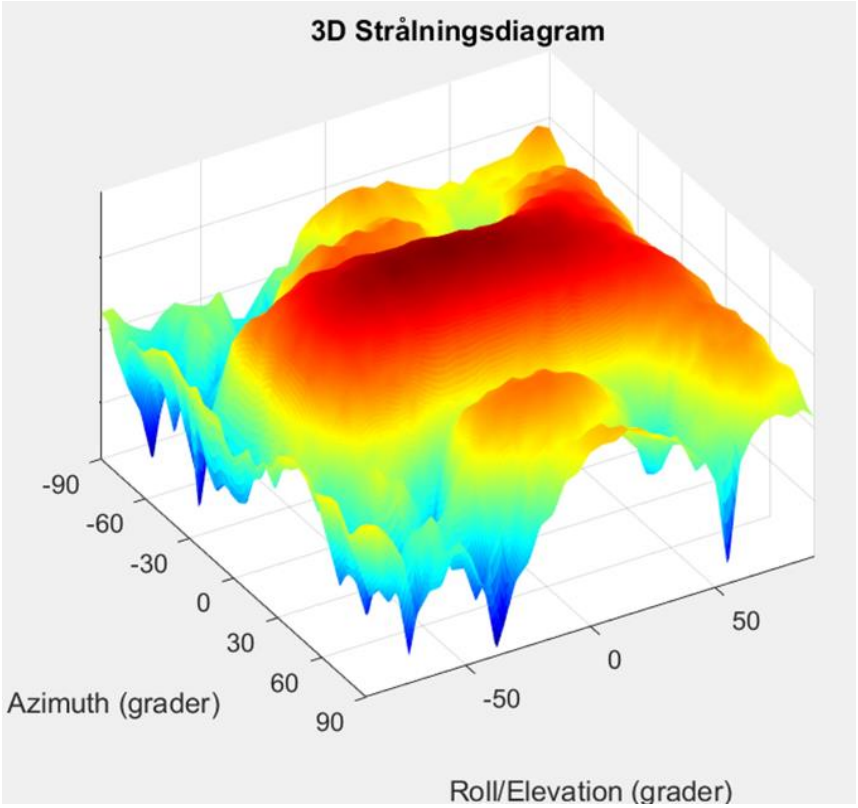
Figur 56: Strålningsdiagram för smal grön antenn, utan isolering. 90° på azimuthskalan innebär 0° i figuren. Felaktig skalering.



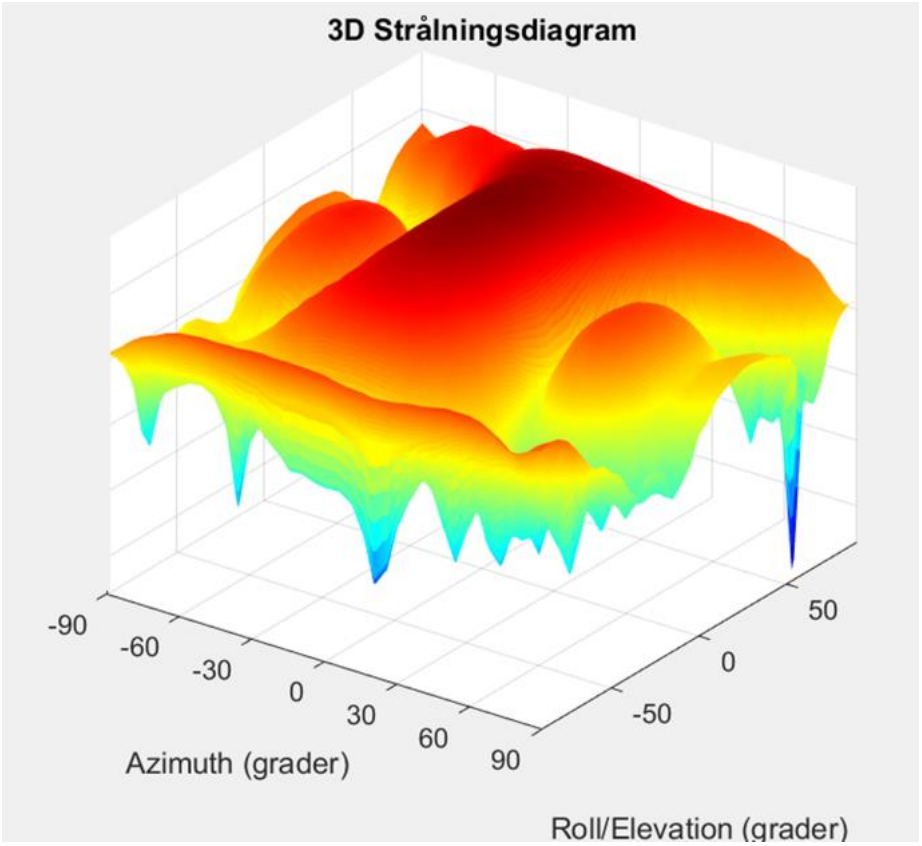
Figur 57: Strålningsdiagram smal grön antenn, fullt isolerad.



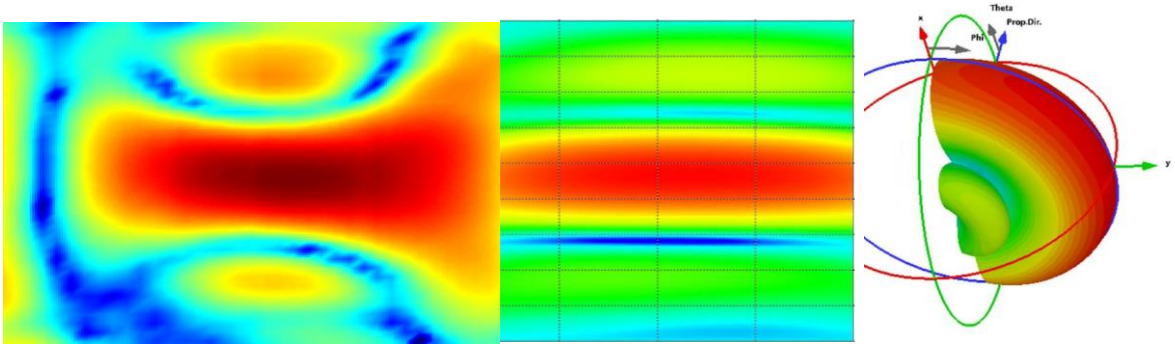
Figur 58: Del av figur 57 ovanifrån sett till vänster, resultat från rektangulär simulering av smal grön antenn till höger.



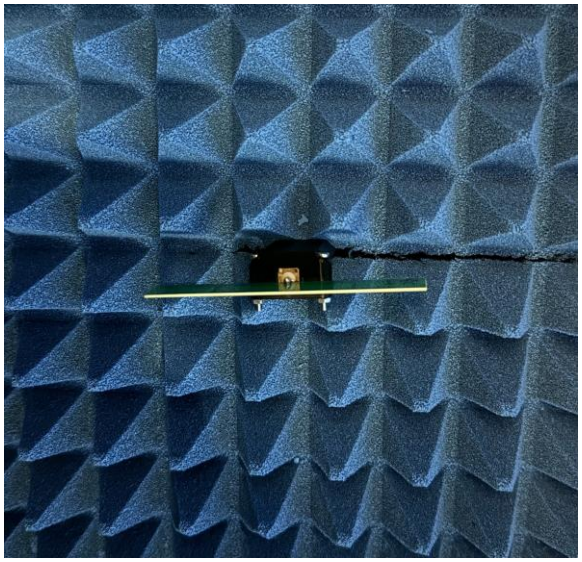
Figur 59: Strålningsdiagram bred grön antenn, halv isolering.



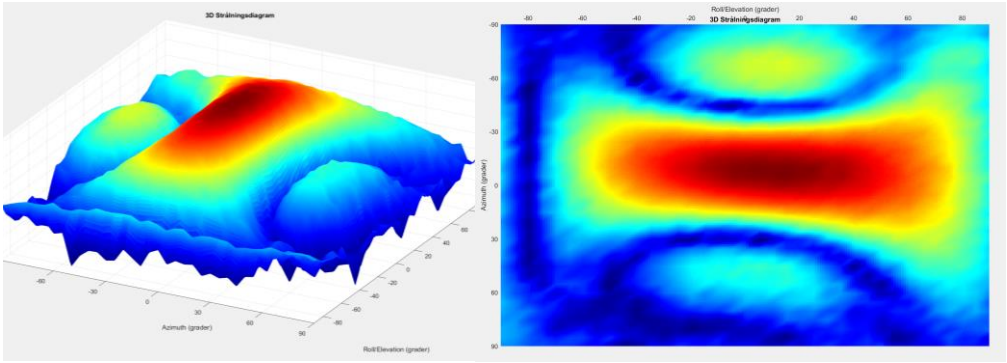
Figur 60: Strålningsdiagram bred grön antenn, fullt isolerad.



Figur 61: Strålningsdiagram, samma som figur 60 fast ovanifrån till vänster i bild, rektangulär simulering ovanifrån i mitten, 3D-simulering till höger.



Figur 62: Utskuren absorbentplatta fäst bakom antenn.



Figur 63: Strålningsdiagram för andra mätning av fullt isolerad bred grön antenn, sidovy till vänster, ovanifrån till höger.

Anledningen till att arbetet avgränsades till de två valda antennerna var för att mätningarna blev tydligast och enklast för RISE att simulera med sina simuleringsverktyg. Laboranterna fick endast bilder av företaget från simuleringen och kunde inte själva använda verktyget. Möjligheten att göra mätningar med både den "gula antennen" och hornantennen skapades som nämnt tidigare i arbetet. Just strålningsmönstret för hornantennen var däremot inte lika relevant då det redan var känt från innan.

Avståndet mellan sändar- och mottagarantenn kan justeras. I de första mätningarna sattes avståndet till 36 cm. Justering kan göras genom att ändra bottenplattans position i förhållande till sändarantennen. Det är viktigt att mottagarantennen roteras kring sitt centrum, vilket innebär att kalibrering av antennens position i förhållande till rotationscentrum är väsentlig. Det här uppnåddes genom att justera antennpinnens position på vridplattan.

6.2 Validering av den smala gröna antennen

Mätningen som visas i Figur 56 genomfördes i ett kontorslandskap (utan isolerad kammare) med den smala gröna antennen. Mätuppställningen konstruerades med hjälp av ett stort plaststativ. Sändarantennen monterades på stativet på samma höjd som mottagarantennen och riktades rakt mot denna för att erhålla så exakta mätvärden som möjligt.

På det uppmätta strålningsdiagrammet i Figur 56 framgår att den mottagna effekten är relativt jämn när antennerna riktas rakt mot varandra, alltså när Azimuth-motorn är på 0° (i figur 90°). Däremot ses tydliga ”nedåtspikar” i värdena när mottagarantennen vinklades bort från sändarantennen. Att effekten minskar vid dessa vinklar är å ena sidan förväntat, då strålningen naturligt är starkare när antennerna är riktade mot varandra. Å andra sidan noterades en utstickande ökning av amplituden i området mellan -60° och -90° för rollmotorn, vilket troddes ha orsakats av yttre påverkan. Den yttre påverkan antogs vara det stora plaststativet till en början.

Inledningsvis antogs även spikarna i strålningsdiagrammet bero på reflektioner från det stora stativet. Eftersom mätningen dessutom utfördes i ett kontorslandskap (utan elektromagnetisk isolering), kunde även omgivande objekt i rummet bidra till strålningsreflektioner. För att erhålla mer tillförlitliga mätningar, med mindre brus och färre "spikar" i signalen, kan en metallkammare med absorberande material användas. I en sådan konfiguration placeras absorbenter på samtliga väggar, samt på golv och tak, för att minimera interna reflektioner. Vid fogar och lister förses kammaren med koppartejp, tillverkad av ledande kopparfolie, för att förhindra att elektromagnetisk strålning tränger in i eller ut ur kammaren. Denna typ av konstruktion benämns som en heldämpad kammare. I detta arbete användes som nämnt tidigare, en temporär lösning, där absorbenter tejpades upp på väggarna och kompletterades med stödjande frigolitblock. Lösningen bedömdes ge tillräcklig isolering för att möjliggöra rimliga slutsatser, och benämns därför, enligt tidigare definition, som en heldämpad eller "fullt isolerad" kammare.

De tidigare antaganden om att det stora plaststativets reflektioner påverkade mätningen kunde slopas när resultaten för mätningen i den fullt isolerade kammaren plottades, vilket visas i Figur 57. I detta fallet användes ett mindre plaststativ, med en nättare design, för att minimera potentiella reflektioner. Här förekom spikarna fortfarande, vilket innebar att det stora stativet eller andra närliggande objekt inte påverkade mätningen mer. Möjliga orsaker till spikarna kunde i stället vara mätfel från spektrumanalysatorn, brus, antennfel eller kraftig dämpning i kablarna. I CSV-filen för strålningsdiagrammet i Figur 57 visas den lägsta spiken till 1.96 dBm, vilket avviker markant från övriga mätvärden vars medelvärde ligger på cirka 35 dBm. Eftersom orsaken till spikarna är svår att fastställa och ligger utanför arbetets avgränsningar, studerades det inte vidare.

Då stativet var tillverkat av plast kunde man här dra slutsatsen att materialet inte påverkade mätningen särskilt mycket. Hade materialvalet istället varit metall, som reflekterar strålning betydligt bättre än plast, hade påverkan sannolikt varit mer påtaglig. Den isolerade mätningen

resulterade emellertid, enligt Figur 57, i ett betydligt jämnare strålningsdiagram jämfört med den oisolerade. Här kunde ytterligare en slutsats dras: att reflektioner från omgivande föremål i rummet ger upphov till brus och störsignaler i den oisolerade mätningen, vilket framgår i form av den knaggliga ytan i det tidigare uppmätta strålningsdiagrammet.

En simulering sett ovanifrån för den smala gröna antennen "Stacked dipole over reduced ground plane" från RISE visas till höger i Figur 58. Till vänster i samma figur visas en mätning av samma antenn ovanifrån. Som framgår är resultaten snarlika, vilket tyder på att mätsäkerheten för produkten är relativt hög, då mätningen efterliknar ett idealt resultat. Att mätningen inte överensstämmer exakt med simuleringen kan bero på glapp mellan absorbentplattorna i kammaren, reflektioner från metalldelar i konstruktionen eller en icke-ideal antenn. I praktiken måste, som nämnt i teorin, samtliga parametrar stämma överens för att simuleringen ska vara helt identisk med mätresultaten. Något som i verkligheten inte är fullt möjligt att uppnå.

6.3 Validering av den breda gröna antennen

Den andra antennen som testades var den breda gröna antennen "two stacked dipoles over ground plane". Den testades först i en delvis dämpad kammare (halvisolerad), där endast den inre och högra väggen (sett från öppningen) kläddes med absorbenter. Halvisoleringen innebar att reflektioner från bland annat taket och vridbordets komponenter kunde påverka mätresultatet. Likt diagrammet för den smala gröna antennen uppvisade även den breda antennens diagram en ojämn yta. Strålningsdiagrammet som visas i Figur 59 innehöll dessutom flera nedåtspikar, vilka antogs bero på bland annat brus.

När den andra mätningen av antennen genomfördes i heldämpad kammare (fullt isolerad), plottades ett betydligt jämnare diagram, vilket illustreras i Figur 60. Här minimerades även förekomsten av nedåtspikar och ytan blev avsevärt tydligare. Även i detta fall drogs slutsatsen att absorbenterna effektivt eliminerade oönskat brus och reflektioner. Grundformen för strålningsdiagrammet kvarstod dock.

I den vänster delen av Figur 61 illustreras mätningen sedd ovanifrån för att möjliggöra jämförelse med simuleringens resultat. Mätningen uppvisade en huvudlob och två mindre sidolober, vilket formas utifrån antennens dubbla uppsättning av dipolelement. En simulering av antennenens strålning visas i den mittersta och högra delen i Figur 61**,** där en utdragen huvudlob och två något utdragna sidolober framträder. Då strålningsdiagrammen skapades utifrån ett fyrkantigt plan (position Azimuth och Roll) med en tredje axel för effekten, så skapades även en "rektangulär" simulering från RISE. Alltså var den mittersta delen av Figur 61 en rektangulär utdragning av den högra delen i samma figur. Allt för att möjliggöra jämförelse mot strålningsdiagrammen.

Det noterades att sidoloberna i strålningsdiagrammet från mätningen var betydligt tydligare än i simuleringen. Därför genomfördes ytterligare en mätning med samma antennpar, men med förbättrad kalibrering av absorbenter och vridbord. Kammaren kläddes som tidigare helt med absorbenter (heldämpad), och höjd samt bredd för båda antennerna justerades extra för att uppnå så exakt inriktning som möjligt. Antennens framkant lodades in med hjälp av ett snöre och en vikt för att säkerställa att rotationen skedde exakt kring centrum för azimutmotorn. Under sändarantennens fasta stativ, samt mellan sändarantenn och vridbordets konstruktion sattes även absorbenter. Dessutom monterades en utskuren absorbentplatta bakom mottagarantennen, vilket framgår i Figur 62. Den extra isolering syftade till att minimera oönskade reflektioner från omgivande komponenter under mätningen. Distansen mellan

antennerna ökades också till 73 cm vilket var det maximala avståndet i kammaren. Den ökade distansen var till för att säkerställa att den tidigare mätningen inte påverkats negativt av att antennerna stod för nära varandra.

Hela den sista mätningen visualiseras i Figur 63. Här beaktades så många parametrar som möjligt vid mättillfället för att uppnå bästa möjliga resultat. Resultatets form kvarstod mellan den första och den andra fullt isolerade mätningen av den breda gröna antennen. En slutsats som kunde dras var att den verkliga formen på antennens strålningsmönster avvek från simuleringen och snarare överensstämde med de uppmätta resultaten i Figur 59, 60, 61 och 63. Den sista mätningen uppvisade dessutom tydligare sidolober än den första, vilket antogs bero på den förbättrade isoleringen.

Om man i Figur 63 jämför mätningens form ovanifrån med simuleringens mer rektangulära form, kan man även se likheten att ena sidoloben är större än den andra i båda fallen. Den vänstra loben i båda delarna av figuren har alltså något högre effekt än den högra. Något som stärker detta är strålningsdiagrammet ovanifrån (till höger i Figur 63), som ännu tydligare visar skillnaden i storlek mellan sidoloberna. Det är positivt eftersom mätningarna efterliknar de ideala resultatet från simuleringen.

Jämförelser mellan respektive mätning är svåra att analysera till 100 procent eftersom mätutrustningen är mycket känslig och ligger på ett unikt sätt varje gång. Minsta lilla justering av antennkabelns placering påverkar hur mycket energi som överförs, vilket i sin tur påverkar mängden strålning i luften. Det exakta avståndet mellan sändar- och mottagarantenn måste också beaktas, då det kan påverka mätningarnas resultat. Andra faktorer, såsom hur mycket dämparen attenuerar strålningen och hur väl kablarna är inkopplade i varandra, instrumenten och antennerna, påverkar även mätresultatet. Simuleringarna från RISE visar hur antennerna ska fungera i ett idealt fall. Skillnader från verkligheten förekommer alltid, eftersom en ideal antenn är svår att tillverka och mäta på exakt samma sätt.

För att rollmotorn skulle orka rotera användes mycket tunna signalkablar som vägde lite. Dessa medförde i sin tur en större dämpning till mottagarantennerna, vilket gjorde att värdena i spektrumanalysatorn kunde vara felaktiga till viss del. Jämförelser av detta ingick inte i kraven, och därför resonerades det inte vidare kring det. Om möjliga förbättringar av arbetet skulle göras i framtiden, hade en starkare motor möjliggjort användning av tjockare antennkablar och därmed skapat mer ideala mätningar.

6.4 Sammanfattning av valideringen

Resultaten från de uppmätta strålningsdiagrammen visade på en god överensstämmelse med simuleringarna, särskilt efter att mätningarna utförts i en fullt isolerad kammare och med förbättrad kalibrering. Den smala gröna antennens diagram liknade väl den simulerade huvudloben, medan den breda antennens sidolober framträdde tydligare i mätdata än i simuleringen. Det indikerar att simuleringarna fångar huvuddragen i antennernas strålningsmönster, men att vissa detaljer i praktiken påverkas av exempelvis antennkonstruktion, dämpning och omgivningsreflektioner. Sammanfattningsvis anses valideringen lyckad då mätresultaten i stort sett bekräftar den simulerade funktionen hos de två testade antennerna.

7. DISKUSSION

I avsnittet tas några av de mest relevanta delarna och problemen upp för respektive huvuddel i arbetet. Det innefattar konstruktion, programmering, praktisk tillämpning på resultat samt förslag på fortsatt utredning.

7.1 Diskussion av prototyp

Alla delar som behövde ha en god hållfasthet printades med 90 procent ifyllnad för att bli starkare. Exempel på dessa delar är vridbordet, antennpinnen, motorhållarna och vissa delar av antennhållaren. Övriga delar i konstruktionen printades i 15 procents ifyllnad för att spara material. Delar som endast 3D-printades för att testa passform hade en ifyllnad på 10 procent.

7.1.1 Materialval

Valet av material för konstruktionen stod mellan aluminium, PLA och kolfiber. Kolfiber uteslöts tidigt i processen, då det visade sig vara svårt att hitta någon som kunde tillverka delarna enligt de angivna specifikationerna. För att uppnå kravet på en precision av 0.2 grader per steg togs speciellt hållfastheten i föraktning.

En vidare undersökning visade att det var möjligt att tillverka delarna i aluminium genom att skicka CAD-ritningarna till ett företag som specialiserar sig på tillverkning av specialbeställda komponenter i olika material. Nackdelen var en hög kostnad samt att det var svårt att genomföra modifikationer på delarna efter tillverkning.

Ytterligare ett problem var att aluminium reflekterar den elektromagnetiska strålning som ska mätas, vilket kan leda till mätfel. En möjlig lösning hade varit att täcka konstruktionen med absorbenter. Därtill hade motorerna begränsningar i hur mycket vikt de klarade axialt och radialt. För att aluminiumkonstruktionen skulle hålla sig inom den förutbestämda viktgränsen ca (1,5 kg) behövde den göras tunnare, vilket väckte oro kring hur tillförlitlig mätprecisionen skulle vara.

Tillverkning med PLA medförde inga större kostnader och möjliggjorde modifikationer genom att nya delar enkelt kunde tillverkas. Problemet var osäkerhet kring om PLA var tillräckligt styvt för att inte böjas eller deformeras under en mätning.

Beslutet fattades, baserat på dessa observationer, att först tillverka delarna i PLA och undersöka om mätningsprecisionen höll en tillräckligt hög standard. Vid otillfredsställande resultat skulle konstruktionen tillverkas i aluminium. Då resultaten av mätningarna gav bra resultat och tillräcklig precision användes PLA som slutligt material.

7.1.2 3D-skrivare

Under projektets gång uppstod vissa utmaningar vid 3D-utskrifter, vilket är vanligt då 3D-skrivare inte alltid är helt tillförlitliga. Vissa utskrifter misslyckades på grund av stopp i skrivaren eller problem med vidhäftning på 3D-skrivarens bottenplatta, vilket resulterade i att delarna inte slutfördes eller misslyckades. I ett fall misslyckades en utskrift på grund av dess storlek trots att den fysiskt fick plats på byggplattan (se Figur 64). Det övre lagret blev ojämnt till följd av att delen låg för nära kanten som påverkade temperaturen på materialet (se Figur 65). Bambu Lab-skrivaren användes för majoriteten av delarna, men i detta fall tvingades en annan 3D-skrivare användas med större byggplatta. För att skriva ut vridbordet som var

projektets största del användes därav en större 3D-skrivare, Prusa XL. När Prusa-skrivaren användes lyckades en utskrift skapas som motsvarade förväntningarna både vad gäller form och kvalitet. Liknande fel inträffade under framtagning av elcentralens lock då både locket och elcentralen printades samtidigt bredvid varandra vilket medförde att locket placerades nära kanten på byggplattan. Felet löstes när delen printades enskilt och flyttades närmare mitten.

Figur 64: Vridbordets placering på den mindre Bambu Lab-skrivaren.

Figur 65: Vridbordets misslyckade utskrift från Bambu Lab-skrivaren där ytan inte blev som önskat.

Antennhållarna krävde noggrann passform vilket resulterade i att den gröna antennhållaren fick skrivas ut en extra gång efter korrigeringar i måtten. Denna del fick även utvecklas för att den gröna antennen hade en mindre koppling till antennkabeln till skillnad från den gula antennen vilket gjorde att material fick tas bort för att möjliggöra korrekt gängning av kabeln. Även hållaren till hornantennen fick skrivas ut två gånger, i detta fall testades först passformen med en utskrift i 10 procents ifyllnad för att sedan göra nödvändiga justeringar innan slutliga delen printades i 90 procents ifyllnad då denna del skulle klara av ett större moment från den tyngre hornantennen.

7.1.3 Nollställe för motorerna

För att skapa ett nolläge för motorerna diskuterades flera olika lösningar. Motorerna är exakta och kan köras med mycket hög precision. Problemet med stegmotorerna är att de inte har något minne eller ett utsatt nolläge som de alltid startar ifrån, till skillnad från servomotorer som har utsatta vinklar. Det innebar alltså att om vridbordet roterades manuellt när motorerna inte var strömförsörjda, påverkade det vart ifrån motorn startade vid nästkommande mätning. Konstruktionen krävde ett nollställe för att alltid starta rotationen från samma punkt och därmed krävdes en lösning till problemet.

Eftersom roll-motorn och antennen hade sladdar som hängde ner kunde konstruktionen inte snurra ett helt varv i samma riktning. Det hade alltså lett till att man behövde ställa vridbordet åt något håll och sedan låta motorn köra tillbaka tills den får en signal. Svårigheten med dessa två lösningar hade varit monteringen av avståndsmätaren eller mikrobrytaren då de hade behövt sitta på exakt rätt plats för att få till vinkeln rätt på vridbordet.

Ett första förslag som diskuterades var att montera en avståndsmätare under vridbordet, vilken skulle kunna detektera när bordet befann sig i sitt nolläge genom att registrera en referenskloss på bottenplattan. Ett alternativt förslag innebar att använda en mikrobrytare som, vid aktivering, bryter en krets och därigenom skickar en signal om nolläge. Med någon av dessa två lösningar skulle motorn kunna köras i en förutbestämd riktning tills nolläget identifierats inför varje mätning. Eftersom roll-motorn och antennen hade kablar som hängde nedåt, kunde konstruktionen inte rotera ett helt varv i samma riktning. Det innebar att vridbordet först behövde placeras i en viss riktning, varefter motorn kunde köras till nolläget där en signal mottogs. Utmaningen med dessa lösningar låg i monteringen av sensorn eller brytaren, då dessa skulle behöva placeras med hög precision för att nolläget skulle kunna detekteras med tillräcklig noggrannhet.

Eftersom de två möjliga lösningarna ovan inte framstod som enkla att implementera diskuterades ytterligare en lösning. På grund av att azimuth-motorn hoppar till ett fast läge när den får ström skapades nollställaren till motorn som manuellt styr motorn till en viss position. Nollställaren visade sig ha brister på grund av glapp vilket resulterade i att vridbordet kunde gå både till höger och vänster till närmsta helsteg i vissa fall. Nollställaren är konstruerad för att få motorn att ställas i vinkel medurs för att sedan tas av och därefter strömföra motorn för att få den att hoppa medurs. Motorn måste gå i medurs riktning för att ge korrekta värden i mätningen då det i koden finns en startsekvens som kör tillbaka vridbordet från högerposition till rakt fram. I de fall den går i moturs riktning får proceduren göras om och därefter tillse att den går medurs när motorn får ström. Denna kontroll skrivs med i användarmanualen. För att helt utesluta att motorn får ström när nollställaren sitter på monterades änden på strömkabeln på nollställaren. Det säkerställde att nollställaren inte kunde sitta på plats när strömkabeln fördes in i vägguttaget. Nollställaren fungerade för att utföra korrekta mätningar men det krävdes en observation på hur vridbordet rörde sig vid strömförning.

7.1.4 Framtagning av hasp

Roll-motorn behövde också ställas in till ett nollställe innan mätning. Till skillnad från azimuth-motorn så hoppar inte roll-motorn till ett fast läge. Tack vare detta kunde en hasp fästas ner i universalhållaren som låser fast motorn. Genom att föra ner haspen och sedan strömföra motorn så låser sig motorn i korrekt läge och haspen kan avlägsnas. En mikrobrytare vid haspen hade varit bra för att tillse att programmet inte kan köras när haspen är nere för att utesluta risken för att motorn körs när haspen är nere. Om programmet körs med haspen nere kan haspen gå sönder

eller motorn slira och eventuellt gå sönder. En tydlig varningstext skrevs på interfacet för att tillse att en kontroll sker innan programmet körs.

7.1.5 Vridbord

Vridbordets utveckling utgick från att konstruktionen främst skulle fungera som en prototyp snarare än en färdig produkt. Då syftet var att testa funktion och uppfylla den kravspecifikation som tagits fram, bedömdes PLA som ett tillräckligt materialval. PLA valdes i första hand på grund av dess låga kostnad, god tillgänglighet, låg vikt samt enkelhet vid utskrift och modifiering. Dessa egenskaper gjorde det möjligt att snabbt genomföra nya iterationer och anpassningar under utvecklingsprocessen. För en färdig produkt i större skala hade däremot andra material och tillverkningsmetoder behövts för att uppnå högre hållfasthet och långsiktig hållbarhet.

Ett problem som upptäcktes i slutet av projektet var att omvandlingen från azimuth-motorns drivpinne via flänsen till vridbordet glappade en aning. Eftersom tyngdpunkten för hela antennpinnen och dess komponenter hamnar utanför azimuth-motorns centrum uppstod en snedbelastning. Belastningen var ett av de potentiella problemen som anades uppstå när flänsarna hade beställts. Skillnaden mellan innerdiametern på den stora flänsen och azimuth-motorns drivpinne var 0,75 mm vilket resulterade i att flänsen inte höll fast vridbordet som önskat. En möjlig framtida förbättring på prototypen hade därför varit att konstruera om denna omvandling till någon mer robust. Problem mellan diametrarna uppstod inte för roll-motorn eftersom dess drivpinne hade en ytterdiameter som var exakt lika stor som innerdiametern på den lilla flänsen.

7.1.6 Python

Kommunikationen mellan python och arduino var vid start inte en självklarhet. Tillslut löstes detta genom att använda samma COM-port som arduinon var inkopplad i, samt samma baudrate-frekvens i båda koderna. Denna valdes till 9600 vilket fungerade väl för projektets krav. Trådbunden kommunikation gav en stabil dataöverföring.

För att skicka tillbaka värden och mätdata från arduinokoden så sattes Serial.print-satser ut. Dessa innefattade den aktuella positionen av azimuth- och rollmotorn samt när mätning skulle tas och när hela mätningen var klar. När systemet var igång så tog pythonkoden över arduinokoden och agerade som en ”master”, vilket förhindrade avläsning av terminalen i Arduinon. Lösningen till detta var en funktion i pythonkoden som läste av den seriella kommunikationen som skickades ut från arduinokoden. Från detta kunde felsökning av styrning och mätningsdata ske på liknande sett som när man läser av print-satserna direkt i arduinoterminalen. Print-satserna användes dessutom för att plotta aktuell position och progressionsbaren i interfacens GUI, samt logga mätvärdena i CSV-filen. Det förenklade hanteringen av datan för att uppfylla kraven i kravspecifikationen.

Lösningen som implementerades för att särskilja på vilket värde som var vilken position av azimuth- eller rollmotorn står beskrivet i 5.3.4.2. Alltså tas alltid förstavärdet i listan ”Data” som azimuth och det andra värdet för roll. För att skapa ett mer robust system som aldrig ”chansar” vilket värde som är vilket så hade man kunnat sätta print-satsen som en string där första delen av stringen är antingen ”Azimuth” eller ”Roll” och därefter splitta datan. Här hade man då kunnat ha två olika satser, ifall split_värde_1 är ”Azimuth” så är det värdet för den

azimuth-motorns position och vice versa. Å andra sidan så fungerade ”chansnings”-metoden hela tiden vilket resulterade i att inga ändringar behövdes göras funktionsmässigt.

GUI:n var i sig problematisk att framställa kodmässigt. Flera olika kodvarianter testades för att få fram liveplottning av azimuth-positionen och progressionsbaren. Att få till så att kombinationerna av alla 4 olika fall för azimuth-motorn och 3 olika fall för roll-motorn fungerade tillsammans, krävdes olika if-satser som under testernas gång blev fler. Utöver möjliga 4 respektive 3 fall för motorerna i vanligt läge så skulle även den omvända ”Roll”-funktionen klaffa ihop med pythonkoden. Denna har lika många fall fast styrning tvärt emot det vanliga. Då liveplottningen av den gröna linjen och progressionsbaren dessutom (i det normala läget) var baserat på att uppdateras varje gång azimuth-motorn når sitt stoppintervall, så fick nya lösningar hittas för roll-motorn. Bland annat att progressionsbaren uppdaterades varje gång azimuth-motorn rörde på sig, alltså varje gång en iteration av roll-motorns alla valda vinklar tagits på en azimuth-vinkel. Det diskuterades även ifall även liveplottning för roll-motorn skulle kodas. Eftersom varje tillägg av nya funktioner i GUI:n ledde till att andra delar slutade fungera, och den initiala liveplottningen krävde mycket tid, genomfördes inte detta. Det antogs att det mest relevanta var att visa azimuth-motorns position samt hur långt (i procent) man kommit in i mätningen vid en viss tidpunkt. En funktion som vid ett vidarearbete hade kunnat läggas till i GUI:n är tiden i timmar och minuter det är kvar. Alltså en estimering på hur lång tid varje iteration tar och beräkna hur många mätningar som ska göras i förtid vid varje stor mätning.

Vid uppdatering av progressionsbaren så finns det i beräkningarna ett + 1 i slutet. Alltså abs(motor2start – motor2stop)/stegstorlek2 + 1. Samma funktionalitet finns i arduinokoden för att inte missa att mäta eller räkna startvärdet i varje iteration. Alltså när man till exempel vill mäta fall 1 i ”vanliga” läget så kan man skicka in värdena i tabell 2:

Tabell 2: Exempelvärden för input till interface.

	Startintervall	Stoppintervall	Stegstorlek
Azimuth	10	30	10
Roll	20	10	10

Om man använder ekvationens värden utan att lägga till +1 blir resultatet (20-10)/10 = 1. Progressionsbaren ska dock uppdateras varje gång rollmotorn tar en ny position, även vid startvärdet, vilket gör att +1 är nödvändigt.

När mätning ska tas från spektrumanalysatorn skickas kommandon som startar en svepning och aktiverar en markör på peak-nivån för den aktuella mätningen.

7.1.7 Arduino

Arduinokoden genomgick flera iterationer under projektets gång, och mellan de olika versionerna uppstod betydande skillnader. Detta försvårade implementeringen av nya funktioner samt felsökning av befintliga problem. En bidragande orsak var att äldre delar av koden inte var lika modulärt uppbyggda som de nyare, vilket försvårade vidareutveckling och integration.

Styrningen av motorerna visade sig vara särskilt utmanande. Azimuth-motorn mätte inte sin position från 0 till 360 grader, utan i stället i ett intervall på ±180 grader relativt till nollpunkten. Detta krävde att mätintervallen delades upp i fyra specifika fall, för att korrekt kunna tolka och hantera användarens angivna värden.

Ett annat problem som identifierades var stegfelen. Dessa uppdagades efter flera tester som syftade till att verifiera motorernas position under mätningar. Resultaten visade att motorernas position gradvis försköts för varje steg, vilket tydde på ett ackumulerande fel. Efter vidare analys kunde orsaken fastställas, och flera lösningar togs fram – anpassade till de olika scenarier motorerna kunde befinna sig i. Dessa lösningar testades systematiskt, och de mest tillförlitliga metoderna valdes ut och implementerades. I vissa fall krävdes skräddarsydda lösningar, särskilt för azimuth-motorns fall 4, som visade sig vara svår att kontrollera och därför behövde en specifik kompensationsfunktion.

Under körning uppstod ett problem där den motor som gjorde den initiala rörelsen passerade sin stoppposition med ett steg. Problemet uppstod eftersom en while-loop användes för att kontrollera motorns aktuella position mot den angivna stoppositionen. I teorin skulle detta fungera, men eftersom motorerna inte rör sig enligt ett koordinatsystem utan i diskreta steg, kunde motorn inte stanna exakt på stoppositionen. I praktiken innebar detta att motorn kunde hamna några decimaler kort om stoppositionen, vilket enligt loopen innebar att ytterligare ett steg skulle tas. Resultatet blev att motorn gick ett steg för långt, utanför det avsedda intervallet.

Lösningen på problemet var att innan körning beräkna det exakta antalet steg som krävdes för att placera motorn så nära stoppositionen som möjligt. En for-loop användes därefter för att iterera motorn över det beräknade antalet steg fram till målet.

7.2 Diskussion av metod

Igenom arbetets gång så har 5 av de 6 delarna inom produktutvecklingsfasen använts. Dessa är idégenereringen, konceptutvecklingen, design och utveckling, testning och validering samt utvärdering och förbättring. Delen som hamnade utanför avgränsningarna för arbetet var produktion och lansering som överlåts till RISE i detta fall. Att sälja och massproducera produkten var inte en del av kandidatarbetet. Metoden som användes var framgångsrik och resulterade i en väl fungerade prototyp med ingående delar som utvecklades under arbetets gång.

Under utvecklingsfasen av prototypen ägnades mest tid i början åt idégenerering och vidareutveckling av de mest lovande koncepten, med särskilt fokus på hållfasthet, viktoptimering och uppgiftens relevans. Vissa av konstruktionens delar behövde endast ritas och printas ut en gång. Exempel på det var motorhållarna för roll- och azimuth-motorerna. Anledningen till detta var att de första koncepten av dessa blev hållfasta och med bra passform för motorerna. Andra delar som till exempel vridbordet behövde testas och utvecklas flera gånger för att få fram en prototyp som uppfyllde kraven. Då prövning och omprövning är en huvuddel i produktutveckling så ses detta som en förbättring av hela produkten mer än något negativt.

Andra delar som tog mycket tid var utvärdering och förbättring. Här faller styrningen för de olika fallen och mikrostegen in. Mycket av problemen med extrasteg och felstegsfunktionen var här väsentligt att få förbättrade för fullständig funktionalitet genom styrningen. Här jämfördes även den praktiska prestandan vilket var viktig att säkerställa. Till exempel att stegmotorerna verkligen rörde sig i verkligheten till de intervall de sattes till.

Under framtagningen och utvecklingen av koden för att styra motorer, skapa lämpligt GUI och framställa strålningsdiagram passade agil systemutveckling mycket bra. Genom att använda en metod som var flexibel och anpassningsbar kunde flera problem lösas inom kort

tidsram. Tack vare att programmeringen delades upp i olika delar blev arbetet mer överskådligt och hanterbart.

7.3 Praktisk tillämpning av resultat

Vid utveckling av nya antenner genomförs design och elektromagnetiska simuleringar i ett program tills man uppnått önskat strålningsdiagram och verkningsgrad. Därefter tillverkas antennen och den framtagna konstruktionen används för att mäta det faktiska strålningsdiagrammet. Eftersom det endast var möjligt att testa strålningen i två dimensioner tidigare har resultatet från detta arbete möjliggjort en ny dimension av valideringen för nya antenner.

Den framtagna konstruktionen kan även användas för att mäta upp strålningsdiagrammet hos en redan befintlig antenn med okända egenskaper.

7.4 Förslag på fortsatt utredning

Här tas utvecklings- och användningsområden upp för framtida utredning kopplat till mätplattformen.

7.4.1 Automatiskt nolläge

I projektet implementerades ingen automatisk uppställning för nolläget för stegmotorerna, då någon form av inbyggt minne saknades. För att öka systemets effektivitet och robusthet hade en funktion som automatiskt söker efter ett nolläge vid varje uppstart varit en tydlig förbättring. Själva förbättringen hade kunnat realiseras genom att använda en extern sensor, exempelvis en lasersensor, för att genomföra nollpunktskalibreringen.

7.4.2 Implementering av AI

För närvarande tar en mätning med 1370 datapunkter cirka 1 timme och 45 minuter att genomföra, med en stegstorlek på 5° och en paustid på 1000 ms. Ifall man hade velat göra en lika bred mätning med en stegstorlek på 1° för azimuth i stället, hade mätningen tagit cirka 5 gånger så lång tid. Antagandet baseras på att datapunkterna hade blivit 5 gånger så många vid en minskad stegstorlek. Om man dessutom hade minskat stegstorlek på roll-motorn hade mätpunkterna blivit ännu fler och resulterat i en ännu längre mätning. För att effektivisera nya mätningar i framtiden hade en tränad AI varit användbar. Ett exempel på det hade varit att låta AI:n, efter ett visst antal datapunker, se ett mönster och förhandsplotta hur strålningsdiagrammet troligen kommer att se ut. En sådan modell hade både sparat tid för framtida arbete och skapat möjligheter att utveckla andra aspekter under tiden man hade fått vänta på en mätning i nuvarande system.

7.4.3 Skala upp konstruktionen

Eftersom den automatiserade mätplattformen i tre dimensioner utvecklades för första gången på RISE, finns potential att vidareutveckla lösningen för att hantera större antenner. En sådan skalning skulle kräva mer kraftfulla motorer samt anpassade drivkomponenter. En större konstruktion hade även varit användbar för att effektivisera mätningarna inom bland annat fordonsindustrin där de i nuläget roterar antennerna manuellt vid varje ny mätning.

8. SLUTSATS

Ifall man kopplar tillbaka till syftet och första frågeställningen för arbetet så har laboranterna lyckats med att ta fram en stabil och fungerande prototyp för strålningsmätning i tre dimensioner. Kravspecifikationen var dessutom att kunna välja stegintervallen till en upplösning av 0,2° per steg. Även detta uppfylldes med marginal då mikrostegen som valdes att användas var 0,05625° per steg. Alltså kan användaren få en precision på cirka 0,06° per steg vilket är mer än tre gånger så precist som kravet på arbetet.

Kopplat till den andra frågeställningen visade jämförelsen mellan den första antennens strålningsdiagram och dess simulering snarlika resultat. Formen överensstämde i båda fallen, vilket var positivt eftersom simuleringens utseende betraktades som idealt. Den andra antennen stämde inte lika väl överens med sitt simuleringsresultat, men vissa likheter kunde ändå urskiljas, bland annat i sidolobernas varierande höjder. Hur mycket simulering och strålningsdiagram skiljer sig är svårt att uppskatta men det förväntades märkbara skillnader. Slutsatsen som drogs var att en noggrann jämförelse mellan verkliga strålningsdiagram och ideala simuleringar är svår att genomföra med hög precision, eftersom väldigt många parametrar måste stämma överens.

Att stegfelsfunktionen lyckades implementeras var viktigt för prototypens precision. Det gör det möjligt att ta exempelvis 1000 mätningssteg på en mätning utan att felet ökar ju längre in i mätningen man kommer. Vid en icke fungerande stegfelsfunktion hade 100 mätningar med ett steg på 1° (start på 0° och stopp på 100°) landat på 101,7°. Beräkningarna till dessa fel tydliggörs i avsnitt 5.3.2.2.

Ifall man hade önskat en ökad robusthet för att minimera vridbordets lutning hade ett annat materialval varit en lösning. Sådan åtgärd är endast nödvändig om företaget eller användaren behöver öka precisionen för mer exakta mätningar, men bedömdes inte som relevant inom ramen för detta arbete.

Slutsatser för mätutrustningens exakta tillförlitlighet är svårt att dra då flera små parametrar ingår i slutgiltigt resultat. Delarna innefattar bland annat kablage och stegmotorerna, men även att kammaren aldrig fullständigt kläddes in. Jämförelser av mätningar och simuleringen hade vid en eventuell färdigställning av kammaren i framtiden varit relevant. En sådan lösning hade troligen resulterat i mer exakta mätningar, då endast en minimal mängd strålning hade kunnat passera in i eller ut ur kammaren.

9. REFERENSER

9.1 Bildkällor:

Stegmotor och strömkälla till kopplingsschema (tillhör figur 39).
https://wiringdcable.blogspot.com/2020/04/nema-17-stepper-motor-wiring-arduino.html [2025-03-07]

9.2 Källförteckning:

Arduino (2025). *Arduino Uno Rev3.*
https://store.arduino.cc/products/arduino-uno-rev3 [2025-03-25]

Asana (2024). *De 6 faserna i produktutvecklingsprocessen (med exempel).*
https://asana.com/sv/resources/product-development-process [2025-04-08]

Bambulab (u.å). Bambu Filament Guide. https://bambulab.com/en/filament-guide [2025-06-02]

Bambulab (u.å). *Bambu Lab X1E*. https://bambulab.com/en/x1e [2025-05-06]

CDON (u.å). *Breadboard kopplingsdäck 830 punkter Arduino Raspberry Pi.*
https://cdon.se/produkt/breadboard-kopplingsdack-830-punkter-arduino-raspberry-pi-764810f6f32f52c0/ [2025-04-08]

Electrokit (2020). *Avståndsgivare 3m VL53L3CX.*
https://www.electrokit.com/avstandsgivare-3m-vl53l3cx [2025-03-25]

Elfa (2019). *QSH5718-76-28-189 - Hybridstegmotor 1.89Nm 1.8° NEMA 23.*
https://www.elfa.se/sv/hybridstegmotor-89nm-nema-23-trinamic-qsh5718-76-28-189/p/15422351 [2025-03-25]

Elfa (u.å). *RND 420-00002 - Stegmotor 9.6VDC 167Nmm 1.8° NEMA 17.*
https://www.elfa.se/sv/stegmotor-6vdc-167nmm-nema-17-rnd-rnd-420-00002/p/30386283 [2025-03-25]

Elsäkerhetsverket (2021). *EMC - elektromagnetisk kompatibilitet.*
https://www.elsakerhetsverket.se/om-oss/vi-arbetar-med/EMC/ [2025-03-25]

Engineering Technology (2011). *Soldering.*
https://engineeringtechnology.org/manufacturing/thermal-joining/soldering/ [2025-05-07]

Farnell (2014). DRI0043 *Stepper Motor Driver, TB6600, TB67S109AFTG. Arduino Development Boards.* https://se.farnell.com/dfrobot/dri0043/stepper-motor-driver-arduino-board/dp/3517878 [2025-03-25]

Farnell (2019). *QMOT QSH5718 MANUAL.*
*https://www.farnell.com/datasheets/3119125.pdf?_gl=1*tkn6ov*_gcl_au*MTgzMjc0MjMzNi4xNzQ2OTc2MDk3* [2025-03-25]

Holland Shielding (2024). *PU FOAM BASED PYRAMID ABSORBERS 3640.* https://hollandshielding.com/pdfs/PU-foam-based-pyramid-absorbers-3640-TDS-rev-15-05-2024.pdf?_gl=1*1cdrtte*_up*MQ..&gclid=CjwKCAjw87XBBhBIEiwAxP3_AxcNzsvxl65a6K4dAdjP6mwUUFlbPF0GFXk_aSqXNa453xi7IHba5RoCVdoQAvD_BwE&gbraid=0AAAAADiTS89VHGXnO3Ku79Jx6T0uYU4SY [2025-05-21]

Ledgrossisten (2025). *Transformator 12V 6A 72W.* https://www.ledgrossisten.se/led-transformator-c-41/transformator-12v-6a-72w-p-320 [2025-03-26]

Metallkompetens. (u.å) *11.1 Allmänna konstruktionsregler*. https://metallkompetens.se/handbok/om-aluminium/konstruktionsregler/allmanna-konstruktionsregler/ [2025-06-02]

Nationalencyklopedin. (u.å.). *Antenn.* https://www.ne.se/uppslagsverk/encyklopedi/lång/antenn [2025-05-11]

Projektledning (2018). *Agila metoder: Levererar små delleveranser & utvärderar processen.* https://projektledning.se/agila-metoder/ [2025-04-07]

Prusa3D (u.å). *Original Prusa XL Assembled 2-Toolhead 3D Printer.* https://www.prusa3d.com/en/product/original-prusa-xl-assembled-2-toolhead-3d-printer/ [2025-05-06]

PTC (2025). *Creo: Design the Way It Should Be.* https://www.ptc.com/en/products/creo [2025-03-25]

Rohde & Schwarz. (2001). *Operating Manual: EMI Test Receiver ESIB7, ESIB26, ESIB40.* Munich: Rohde & Schwarz GmbH & Co. KG.

RS (2020). *Rohde & Schwarz SMCV100BP2 Bundle RF Signal Generator, 4kHz min, 3GHz max.* https://se.rs-online.com/web/p/rf-signal-generators/2307790 [2025-04-15]

RISE (2024). *Nordens största EMC-anläggning blir ännu större.* https://www.ri.se/sv/nyheter/nordens-storsta-emc-anlaggning-blir-annu-storre?utm.com [2025-05-08]

Strålsäkerhetsmyndigheten (2024). *Basstationer.* https://www.stralsakerhetsmyndigheten.se/omraden/magnetfalt-och-tradlos-teknik/tradlos-teknik/mobiltelefoni/basstationer/ [2025-05-08]

Time and Date (u.å.). *Altitude & Azimuth: The Horizontal Coordinate System.* https://www.timeanddate.com/astronomy/horizontal-coordinate-system.html [2025-04-11]

3D Guide (2024). *Vad är Filament för 3D Skrivare.* https://www.3dguide.se/vad_ar_filament [2025-03-26]

3D Printing (2025). *What is 3D printing?.* https://3dprinting.com/what-is-3d-printing/ [2025-04-11]

BILAGOR

Bilaga 1 Pythonkod:

```
import tkinter as tk
from tkinter import ttk
import serial
import time
import threading
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import pyvisa
import csv
from datetime import datetime
import os

# Buffert för mätdata
current_measurement = {
    "azimuth": None,
    "roll": None,
    "peak": None
}

# Skapa (eller återanvänd) fil och skriv header om filen är ny
script_dir = os.path.dirname(os.path.abspath(__file__))  # Hämta sökvägen till .py-filen
csv_filename = os.path.join(script_dir, f"mätdata_{datetime.now().strftime('%Y%m%d_%H%M%S')}.csv")

with open(csv_filename, mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["Azimuth", "Roll", "Peak Level (dBm)"])

#konfiguera VISA resource manager
rm = pyvisa.ResourceManager()

#öppna nätverk till spektrumanalysatorn
instrument = rm.open_resource('GPIB0::20::INSTR')  #skicka in adress till enhet, ta reda på enhetsnamn...

# Sätt timeout
instrument.timeout = 5000  # Timeout på 5

# Konfigurera seriell anslutning
ser = serial.Serial('COM5', 9600, timeout=1)  #com 8 i dator
time.sleep(2)

# Globala variabler
motor1_position = 0
intervall_start = 0
intervall_stop = 0
```

```
roll_value = 0
motor_read_count = 0
last_azimuth_angle = None

# Definiera större font
font_large = ('Arial', 12)  # Större font för textfält och etiketter
font_button = ('Arial', 14)  # Större font för knappar

# funktion som skickar värden
def send_values():
        global intervall_start, intervall_stop, intervall_motor2_start, intervall_motor2_stop,
stegstorlek2, stegstorlek, roll_value
    try:
        stegstorlek = float(stegstorlek_entry.get())
        stegstorlek2 = float(stegstorlek2_entry.get())
        intervall_start = float(intervall_motor1_start_entry.get())
        intervall_stop = float(intervall_motor1_stop_entry.get())
        intervall_motor2_start = float(intervall_motor2_start_entry.get())
        intervall_motor2_stop = float(intervall_motor2_stop_entry.get())
        paus_tid = int(pause_entry.get())

        # Lägg till "Roll" i data (om roll_enabled är 1, lägg till den)
        roll_value = roll_enabled.get()  # 1 om markerad, 0 om inte markerad

        # Återställ progress-baren, texten till 0% och klar_label
        progress_bar["value"] = 0
        progress_label.config(text="0%")
        klar_label.config(text="")

        # Skapa data-strängen som inkluderar "roll_value"
                                                                    data                    =
f"{stegstorlek},{stegstorlek2},{intervall_start},{intervall_stop},{intervall_motor2_start},{int
ervall_motor2_stop},{paus_tid},{roll_value}\n"
        ser.write(data.encode())
        status_label.config(text=f"Skickade: {data}", foreground="green")
    except ValueError:
        status_label.config(text="Fel: Ange giltiga värden!", foreground="red")

# funktion som uppdaterar progressionsbar
def update_progress():
    if roll_value == 1:
        if intervall_start >= 180 and intervall_stop < 180:
            antal_procent_steg = (360 - intervall_start + intervall_stop)/stegstorlek + 1
        else:
            antal_procent_steg = abs(intervall_start-intervall_stop)/ stegstorlek + 1
        storlek_procent_steg = 100 / antal_procent_steg
        progress_bar["value"] += storlek_procent_steg
        progress_label.config(text=f"{progress_bar['value']:.1f}%")
        root.update_idletasks()
    else:
```

```
        antal_procent_steg = abs(intervall_motor2_start - intervall_motor2_stop) / stegstorlek2 +
1
        storlek_procent_steg = 100 / antal_procent_steg
        progress_bar["value"] += storlek_procent_steg
        progress_label.config(text=f"{progress_bar['value']:.1f}%")
        root.update_idletasks()

# funktion som uppdaterar cirkeldiagrammets plot
def update_plot(position):
    global motor1_position
    motor1_position = position

    # Omvandlar position till en vinkel mellan 0° och 360°:
    if position >= 0:
        angle = (position / 3200) * 180   # För positiva positioner (0 till 3200), omvandlas till 0°
till 180°
    else:
        angle = (position / 3200) * 180 + 360  # För negativa positioner (-3200 till 0), omvandlas
till 180° till 360°

    # För att säkerställa att vinkeln alltid ligger mellan 0° och 360°
    angle = angle % 360

    angle_rad = np.deg2rad(-angle)  # Invertera vinkeln för medursrotation

    ax.clear()
    ax.set_xlim(-1.6, 1.6)
    ax.set_ylim(-1.6, 1.6)

    ax.set_xticks([])  # Ta bort x-axelns tickmarks
    ax.set_yticks([])  # Ta bort y-axelns tickmarks
    ax.set_aspect('equal')
    ax.set_title("Azimuth Position")

    # Rita cirkel
    circle = plt.Circle((0, 0), 1, color='gray', fill=False)
    ax.add_patch(circle)

    # Rita start/stopp-linjer
    start_rad = np.deg2rad(-intervall_start)  # Invertera även start och stopp för att passa medurs
    stop_rad = np.deg2rad(-intervall_stop)
    ax.plot([0, np.cos(start_rad)], [0, np.sin(start_rad)], 'r--')
    ax.plot([0, np.cos(stop_rad)], [0, np.sin(stop_rad)], 'r--')

    # Rita den gröna linjen för aktuell position
    ax.plot([0, np.cos(angle_rad)], [0, np.sin(angle_rad)], 'g-', linewidth=3)

    for deg in range(0, 360, 45):
        rad = np.deg2rad(-deg)
        x = 1.3 * np.cos(rad)
```

```
        y = 1.3 * np.sin(rad)
        ax.text(x, y, f"{deg}°", ha='center', va='center', fontsize=12, color='black',
                bbox=dict(boxstyle="round,pad=0.2", fc="white", ec="gray", alpha=0.8))

    # Uppdatera grafen
    canvas.draw()

    # Uppdatera progress bar varje gång ny position kommer – men olika logik beroende på roll
    if roll_value == 0:
        epsilon = 0.1
        if abs(angle - intervall_stop) <= epsilon:
            update_progress()
    else:
        global last_azimuth_angle

        if last_azimuth_angle is None:
            update_progress()  # Uppdatera direkt vid första värdet
            last_azimuth_angle = angle
        elif angle != last_azimuth_angle:
            update_progress()
            last_azimuth_angle = angle

# funktion som styr hela programmet, läser av seriell data, skickar kommandon och kallar på
respektive funktion
def read_serial():
    global motor_read_count
    while True:
        if ser.in_waiting > 0:
            try:
                line = ser.readline().decode('utf-8').strip()
                print(f"Data arduino: {line}")

                if line == "KLAR":
                                root.after(0,  lambda:  klar_label.config(text="Mätning  färdig!",
foreground="green"))
                    motor_read_count = 0  # Återställ när mätningen är klar

                elif line.isdigit() or (line.startswith("-") and line[1:].isdigit()):
                    position = int(line)
                    angle = (position / 3200) * 180 if position >= 0 else 360 + (position / 3200) * 180

                    if motor_read_count % 2 == 0:
                        azimuth_deg = round(angle, 2)
                        current_measurement["azimuth"] = azimuth_deg
                        print(f"Azimuth vinkel: {azimuth_deg}°")
                        root.after(0, update_plot, position)
                    else:
                        if angle > 180:
                            angle -= 360
                        roll_deg = round(angle, 2)
```

```
                current_measurement["roll"] = roll_deg
                print(f"Roll vinkel: {roll_deg}°")
            motor_read_count += 1

        elif line == "Mätning":
            try:

                # Starta svep och vänta på att det är klart
                instrument.write("INIT:IMM")
                instrument.query("*OPC?")

                # Markör + peak-sök
                instrument.write("CALC:MARK1 ON")
                instrument.write("CALC:MARK1:MAX")
                time.sleep(0.5)

                # Läs ut mätdata
                                                                peak_freq =
round(float(instrument.query("CALC:MARK1:X?").strip())/10**9,5)
                peak_power = round(float(instrument.query("CALC:MARK1:Y?").strip()),5)

                print(f"Peak-frekvens: {peak_freq} GHz")
                print(f"Peak-nivå: {peak_power} dBm")

                current_measurement["peak"] = peak_power

                # När vi har alla tre – skriv till fil
                if all(v is not None for v in current_measurement.values()):
                    with open(csv_filename, mode='a', newline='') as file:
                        writer = csv.writer(file)
                        writer.writerow([
                            current_measurement["azimuth"],
                            current_measurement["roll"],
                            current_measurement["peak"]
                        ])
                        print("Mätning sparad till CSV.")

                    # Återställ för nästa mätning
                    current_measurement["azimuth"] = None
                    current_measurement["roll"] = None
                    current_measurement["peak"] = None

            except Exception as e:
                print(f"Fel vid spektrumanalysator-mätning: {e}")

        except Exception as e:
            print(f"Fel vid seriell läsning: {e}")
    time.sleep(0.1)

threading.Thread(target=read_serial, daemon=True).start()
```

```
# Skapa huvudfönster
root = tk.Tk()
root.title("Arduino Motor Controller")
root.geometry("1200x900")
root.resizable(False, False)
root.configure(bg="#f0f0f0")

frame = tk.Frame(root)
frame.pack(side=tk.LEFT, expand=True, fill="both")

klar_label = tk.Label(root, text="", foreground="blue", font=font_large)
klar_label.pack(pady=20)

roll_enabled = tk.IntVar(value=0)
roll_checkbox = ttk.Checkbutton(frame, text="Roll", variable=roll_enabled)
roll_checkbox.grid(row=6, column=2, padx=5, pady=2)

progress_bar = ttk.Progressbar(root, orient="horizontal", length=500, mode="determinate")
progress_bar.pack(pady=20)
progress_label = tk.Label(root, text="0%", font=font_large)
progress_label.pack(pady=10)

tk.Label(frame, text="Stegstorlek", font=font_large).grid(row=0, column=1, padx=5, pady=15)
tk.Label(frame, text="Intervall Start", font=font_large).grid(row=0, column=2, padx=5, pady=15)
tk.Label(frame, text="Intervall Stop", font=font_large).grid(row=0, column=3, padx=5, pady=15)

tk.Label(frame, text="Azimuth", font=font_large).grid(row=1, column=0, padx=5, pady=15)
stegstorlek_entry = tk.Entry(frame, font=font_large, width=10)
stegstorlek_entry.grid(row=1, column=1, padx=5, pady=15)
intervall_motor1_start_entry = tk.Entry(frame, font=font_large, width=10)
intervall_motor1_start_entry.grid(row=1, column=2, padx=5, pady=15)
intervall_motor1_stop_entry = tk.Entry(frame, font=font_large, width=10)
intervall_motor1_stop_entry.grid(row=1, column=3, padx=5, pady=15)

tk.Label(frame, text=" ").grid(row=2, column=0, pady=20)

tk.Label(frame, text="Roll", font=font_large).grid(row=3, column=0, padx=5, pady=15)
stegstorlek2_entry = tk.Entry(frame, font=font_large, width=10)
stegstorlek2_entry.grid(row=3, column=1, padx=5, pady=15)
intervall_motor2_start_entry = tk.Entry(frame, font=font_large, width=10)
intervall_motor2_start_entry.grid(row=3, column=2, padx=5, pady=15)
intervall_motor2_stop_entry = tk.Entry(frame, font=font_large, width=10)
intervall_motor2_stop_entry.grid(row=3, column=3, padx=5, pady=15)

tk.Label(frame, text="Roll (Start > Stop)", font=font_large).grid(row=4, column=2, padx=5, pady=15)
```

```
tk.Label(frame, text=" ").grid(row=8, column=0, pady=20)

# Varningstext under Roll-checkbox
warning_label = tk.Label(frame, text="KONTROLLERA ATT HASP TILL ROLL ÄR BORTTAGEN",
                         font=("Arial", 14, "bold"), fg="red", anchor="w", justify="left")
warning_label.grid(row=9, column=0, columnspan=4, sticky="w", padx=10, pady=(10, 0))

tk.Label(frame, text="Valfri paustid (ms)", font=font_large).grid(row=5, column=1, padx=5, pady=15)
tk.Label(frame, text="(Rek min 1000-1500)", font=font_large).grid(row=5, column=3, padx=5, pady=15)
pause_entry = tk.Entry(frame, font=font_large, width=10)
pause_entry.grid(row=5, column=2, padx=5, pady=15)

send_button = tk.Button(root, text="Skicka till Arduino", font=font_button, command=send_values)
send_button.pack(pady=20)

status_label = tk.Label(root, text="Väntar på input...", foreground="blue", font=font_large)
status_label.pack(pady=15)

fig, ax = plt.subplots(figsize=(6, 6))
ax.set_xlim(-1.3, 1.3)
ax.set_ylim(-1.3, 1.3)
ax.set_xticks([])
ax.set_yticks([])
ax.set_aspect('equal')
ax.set_title("Azimuth Position")

circle = plt.Circle((0, 0), 1, color='gray', fill=False)
ax.add_patch(circle)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=True)
canvas.draw()

root.mainloop()
ser.close()
```

Bilaga 2 Matlabkod (plottning av strålningsdiagram):

```
plot_radiation_pattern('mätdata_20250507_123644.csv');

function plot_radiation_pattern(filename)
    % Läs in CSV-data
    data = readmatrix(filename);
    if size(data,2) < 3
        error('Filen måste innehålla minst 3 kolumner: azimut, elevation, amplitud.');
    end

    % Extrahera kolumner
    azimuth_deg = data(:,1);    % kolumn 1: azimut (i grader)
    elevation_deg = data(:,2);  % kolumn 2: roll (i grader)
    amplitude_dBm = data(:,3);  % kolumn 3: strålningsstyrka (redan i dBm)

    % Justera azimuth för att gå från 270° -> 360°/0° -> 90°
    azimuth_deg(azimuth_deg < 0) = azimuth_deg(azimuth_deg < 0) + 360; % Konvertera
negativa azimuth-värden
    azimuth_deg = mod(azimuth_deg, 360); % Säkerställ att allt är mellan 0-360°

    % Dela upp och sortera för att säkerställa rätt ordning
    azimuth_deg(azimuth_deg >= 270) = azimuth_deg(azimuth_deg >= 270) - 360; % Flytta
270-360 till negativa värden
    [azimuth_deg, sort_idx] = sort(azimuth_deg); % Sortera från -90 till 90
    elevation_deg = elevation_deg(sort_idx);
    amplitude_dBm = amplitude_dBm(sort_idx);

    % Skapa ett meshgrid för att interpolera data
    [azimuth_grid, elevation_grid] = meshgrid(linspace(-90, 90, 200), ...
                                    linspace(min(elevation_deg), max(elevation_deg), 200));

    % Interpolera amplituden för att skapa en kontinuerlig yta
    amplitude_grid = griddata(azimuth_deg, elevation_deg, amplitude_dBm, azimuth_grid,
elevation_grid, 'linear');

    % Skapa en 3D-ytplot
    figure;
    surf(azimuth_grid, elevation_grid, amplitude_grid, 'EdgeColor', 'none');

    % Färgsättning
    colormap jet;
    colorbar;
    c = colorbar;
    c.Label.String = 'Amplitud (dBm)';

    % Etiketter och titlar
    title('3D Strålningsdiagram');
    xlabel('Azimuth (grader)');
    ylabel('Roll/Elevation (grader)');
    zlabel('Amplitud (dBm)');
```

```
    xticks(-90:30:90); % Sätter x-axeln från -90 till 90
    axis tight;
    grid on;
    rotate3d on;
end
```

Bilaga 3 Arduinokod:

```
#include <AccelStepper.h>
#include <math.h>

#define DIR_PIN1 8
#define STEP_PIN1 9
#define DIR_PIN2 2
#define STEP_PIN2 3
#define ENA_PIN 5

AccelStepper motor1(AccelStepper::DRIVER, STEP_PIN1, DIR_PIN1);
AccelStepper motor2(AccelStepper::DRIVER, STEP_PIN2, DIR_PIN2);

float stegstorlek = 0;
float stegstorlek2 = 0;
float intervall_motor1_start = 0;
float intervall_motor1_stop = 0;
float intervall_motor2_start = 0;
float intervall_motor2_stop = 0;
bool readyToMove = false;
int round_measure_steps = 0;
float measure_steps = 0;
bool extra_steg = false;
bool extra_steg2 = false;
float diff_steg2 = 0;
float pause_tid = 0;
int roll = 0;
bool roll_1 = false;
float diff_steg_motor2_2 = 0;
bool extra_steg_motor2 = false;
bool extra_steg_motor2_2 = false;
int round_step_size = 0;
float diff_steg = 0;
int round_step_size_motor2 = 0;
int i = 0;
int tillbakasteg = 0;
int rounded_stop_position_motor1 = 0;
float step_size_float_motor1 = 0;
int steps_to_take = 0;

void moveMotor(AccelStepper &motor, float steps) {
  motor.move(steps);
  while (motor.distanceToGo() != 0) {
    motor.run();
    delay(5);
  }
}
// Ersätter moveMotorWithCompensation
void moveMotorWithErrorTracking(AccelStepper& motor, float& accumulated_error, float
step_size) {
```

```
  float to_move = step_size + accumulated_error;
  int full_steps = floor(to_move);  // Heltalssteg
  accumulated_error = to_move - full_steps;  // Spara decimaldelen till nästa gång

  if (full_steps != 0) {
    moveMotor(motor, full_steps);  // Använd din befintliga motorstyrningsfunktion
  }
}

void moveMotorWithCompensation(AccelStepper& motor, float& exact_position, float&
accumulated_error, float desired_step_size) {
  float current_pos = motor.currentPosition();

  // Korrigera för negativa positioner (cirkulär logik: 0-6400 steg för 0–360°)
  if (current_pos < 0) {
    current_pos += 6400;

  }
  if (exact_position < 0) {
    exact_position += 6400;
  }

  exact_position += desired_step_size;
  float target_with_error = exact_position + accumulated_error;

  int steps_to_take = round(target_with_error - current_pos);
  accumulated_error = target_with_error - (current_pos + steps_to_take);

  moveMotor(motor, steps_to_take);
}



void moveMotorWithErrorTrackingMotor2(AccelStepper &motor, float &accumulated_error,
float step_size_steps) {
  int rounded_steps = round(step_size_steps);          // avrundade steg
  float error = step_size_steps - rounded_steps;       // differens från exakt värde
  accumulated_error += error;                          // lägg till fel till ackumulerad

  // Om tillräckligt mycket fel har ackumulerats – justera
  if (accumulated_error <= -1.0) {
    rounded_steps -= 1; // ta ett extra steg i negativ riktning
    accumulated_error += 1.0; // återställ fel
  }
  if(accumulated_error>=1.0){
    rounded_steps +=1;
    accumulated_error -=1;
  }

  // Garantera att vi rör oss i negativ riktning
```

```
  int steps_to_move = -abs(rounded_steps);

  moveMotor(motor, steps_to_move);  // Använd befintlig funktion
}

void setup() {
  Serial.begin(9600);
  pinMode(ENA_PIN, OUTPUT);
  digitalWrite(ENA_PIN, LOW);

  motor1.setMaxSpeed(200);
  motor1.setAcceleration(50);
  motor2.setMaxSpeed(200);
  motor2.setAcceleration(50);

  tillbakasteg = 1;
}

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    float values[8];
    int index = 0;

    while (input.length() > 0 && index < 8) {
      int separatorIndex = input.indexOf(',');
      if (separatorIndex == -1) {
        values[index] = input.toFloat();
        break;
      }
      values[index] = input.substring(0, separatorIndex).toFloat();
      input = input.substring(separatorIndex + 1);
      index++;
    }

    stegstorlek = values[0];
    stegstorlek2 = values[1];
    intervall_motor1_start = values[2];
    intervall_motor1_stop = values[3];
    intervall_motor2_start = values[4];
    intervall_motor2_stop = values[5];
    pause_tid = values[6];
    roll = values[7];

    if (tillbakasteg == 1) {
      moveMotor(motor1, -72);  // Rör motor1 -72 mikrosteg
      tillbakasteg = 0;        // Se till att detta inte körs igen
      motor1.setCurrentPosition(0);
    }
```

```
  delay(pause_tid);

  readyToMove = true;
}
if (roll == 1){
  roll_1 = true;
}

if (readyToMove && roll_1 == false) {
  //motor1
  float start_position = intervall_motor1_start / 0.05625;
  float stop_position = intervall_motor1_stop / 0.05625;
  float step_size = stegstorlek / 0.05625;
  int round_start_position = round(start_position);
  int round_stop_position = round(stop_position);
  int round_step_size = ceil(step_size);
  //motor2
  float steg_storlek_motor2 = stegstorlek2 / 0.05625;
  float start_position_motor2 = intervall_motor2_start/0.05625;
  float stop_position_motor2 = intervall_motor2_stop/0.05625;
  int round_start_position_motor2 = round(start_position_motor2);
  int round_stop_position_motor2 = round(stop_position_motor2);
  int round_step_size_motor2 = ceil(steg_storlek_motor2);

  if(intervall_motor2_start>0 && intervall_motor2_stop>=0){
    float antal_steg_motor2 = start_position_motor2 - stop_position_motor2;
    measure_steps = antal_steg_motor2/steg_storlek_motor2;
    round_measure_steps = round(measure_steps);
  }
  else if (intervall_motor2_start<=0 && intervall_motor2_stop<0) {
    float absolut_start_motor2 = fabs(start_position_motor2);
    float absolut_stop_motor2 = fabs(stop_position_motor2);
    float antal_steg_motor2 = absolut_stop_motor2 - absolut_start_motor2;
    measure_steps = antal_steg_motor2/steg_storlek_motor2;
    round_measure_steps = round(measure_steps);
  }
  else if (intervall_motor2_start>0 && intervall_motor2_stop<0) {
    float absolut_start_motor2 = fabs(start_position_motor2);
    float absolut_stop_motor2 = fabs(stop_position_motor2);
    float antal_steg_motor2 = absolut_start_motor2 + absolut_stop_motor2;
    measure_steps = antal_steg_motor2/steg_storlek_motor2;
    round_measure_steps = round(measure_steps);
  }
  float diff_steg = (step_size - round_step_size);
  float diff_steg_motor2 = (steg_storlek_motor2 - round_step_size_motor2);
  float accumulated_error_motor2 = 0.0;


  // FALL 1: Om start och stopp är i vänstra halvplanet (0–180)
  if (intervall_motor1_start < 180 && intervall_motor1_stop <= 180) {
```

```
    moveMotor(motor1, round_start_position);
    moveMotor(motor2, round_start_position_motor2);
    Serial.println(motor1.currentPosition());
    Serial.println(motor2.currentPosition());
    float flytande_start_postition = motor1.currentPosition();
    float exact_position_motor1 = motor1.currentPosition();
    float accumulated_error_motor1 = 0.0;
    while (i<=round_measure_steps){

     delay(pause_tid);//mätning sker här
     Serial.println("Mätning");

     float exact_position_motor1 = motor1.currentPosition();
     float accumulated_error_motor1 = 0.0;

     int total_steps = round((round_stop_position - motor1.currentPosition()) / step_size);
     for (int j = 0; j < total_steps; j++) {

                              moveMotorWithCompensation(motor1,      exact_position_motor1,
accumulated_error_motor1, step_size);

       Serial.println(motor1.currentPosition());
       Serial.println(motor2.currentPosition());
       delay(pause_tid); //här sker en mätning
       Serial.println("Mätning");



     }
       if(intervall_motor2_start>intervall_motor2_stop && intervall_motor2_start<=90 &&
intervall_motor2_stop>=-90){
       if(i<round_measure_steps){
                  moveMotorWithErrorTrackingMotor2(motor2,   accumulated_error_motor2,
steg_storlek_motor2);
        float return_start_position = flytande_start_postition - motor1.currentPosition();
        moveMotor(motor1, return_start_position);
        Serial.println(motor1.currentPosition());
        Serial.println(motor2.currentPosition());
       }

       i+=1;

     }

    }
    moveMotor(motor1, -motor1.currentPosition());  // Återgår till 0

    if(intervall_motor2_stop > 0){
     moveMotor(motor2, -motor2.currentPosition());
    }
    else if(intervall_motor2_stop< 0){
```

```
    moveMotor(motor2, -motor2.currentPosition());
   }
  }

  // FALL 2: Om start är i vänster och stopp i höger (ex. 160–200)
  else if (intervall_motor1_start < 180 && intervall_motor1_stop > 180) {
   moveMotor(motor1, round_start_position);
   moveMotor(motor2, round_start_position_motor2);
   Serial.println(motor1.currentPosition());
   Serial.println(motor2.currentPosition());
   int total_steps_positiv = round(((180/0.05625) - motor1.currentPosition()) / step_size);
   int total_steps_negativ = round((round_stop_position - (180/0.05625))  / step_size);


   while (i<=round_measure_steps){
    delay(pause_tid);//mätning sker här
    Serial.println("Mätning");
    float flytande_start_postition = motor1.currentPosition();
    float exact_position_motor1 = motor1.currentPosition();
    float accumulated_error_motor1 = 0.0;


    for (int j = 0; j < total_steps_positiv; j++) {
                        moveMotorWithCompensation(motor1,      exact_position_motor1,
accumulated_error_motor1, step_size);
     Serial.println(motor1.currentPosition());
     Serial.println(motor2.currentPosition());

     delay(pause_tid);//mätning sker här
     Serial.println("Mätning");
    }

    moveMotor(motor1, -motor1.currentPosition()); // Återgår till 0
    moveMotor(motor1, -180 / 0.05625); // Går moturs till 180


    for (int j = 0; j < total_steps_negativ; j++) {
                        moveMotorWithCompensation(motor1,      exact_position_motor1,
accumulated_error_motor1, step_size);
      Serial.println(motor1.currentPosition());
      Serial.println(motor2.currentPosition());

      delay(pause_tid);//mätning sker här
      Serial.println("Mätning");
    }
      if(intervall_motor2_start>intervall_motor2_stop && intervall_motor2_start<=90 &&
intervall_motor2_stop>=-90){
     if(i<round_measure_steps){
              moveMotorWithErrorTrackingMotor2(motor2,   accumulated_error_motor2,
steg_storlek_motor2);
```

```
        float return_start_position = flytande_start_postition - motor1.currentPosition();
        moveMotor(motor1, return_start_position);
        Serial.println(motor1.currentPosition());
        Serial.println(motor2.currentPosition());
       }
       i+=1;
      }
     }
     moveMotor(motor1, -motor1.currentPosition()); // Återgår till 0

     if(intervall_motor2_stop > 0){
      moveMotor(motor2, -motor2.currentPosition());
     }
     else if(intervall_motor2_stop< 0){
      moveMotor(motor2, -motor2.currentPosition());
     }
    }

    // FALL 3: Om både start och stopp är i högra halvplanet (180–360)
    else if (intervall_motor1_start >= 180 && intervall_motor1_stop > 180) {
     // Gå moturs från 0 till startpunkten i höger halvplan
     float target_start = -((360 - intervall_motor1_start) / 0.05625);
     int round_target_start = round(target_start);

     moveMotor(motor1, round_target_start);
     moveMotor(motor2, round_start_position_motor2);

     Serial.println(motor1.currentPosition());
     Serial.println(motor2.currentPosition());

     int total_steps = round((round_stop_position - round_start_position) / step_size);
     float flytande_start_postition = motor1.currentPosition();

     while(i<=round_measure_steps){
      delay(pause_tid);//mätning sker här
      Serial.println("Mätning");
      float exact_position_motor1 = motor1.currentPosition();
      float accumulated_error_motor1 = 0.0;
      for (int j = 0; j < total_steps; j++) {
                              moveMotorWithCompensation(motor1,      exact_position_motor1,
accumulated_error_motor1, step_size);
       Serial.println(motor1.currentPosition());
       Serial.println(motor2.currentPosition());

       delay(pause_tid);//mätning sker här
       Serial.println("Mätning");
      }
        if(intervall_motor2_start>intervall_motor2_stop && intervall_motor2_start<=90 &&
intervall_motor2_stop>=-90){
       if(i<round_measure_steps){
```

```
                moveMotorWithErrorTrackingMotor2(motor2,  accumulated_error_motor2,
steg_storlek_motor2);
          float return_start_position = flytande_start_postition - motor1.currentPosition();
          moveMotor(motor1, return_start_position);
          Serial.println(motor1.currentPosition());
          Serial.println(motor2.currentPosition());
        }
        i+=1;
      }
    }

    // Återgåe till 0 via högra halvplanet
    moveMotor(motor1, (360 - intervall_motor1_stop) / 0.05625);

    if(intervall_motor2_stop > 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
    else if(intervall_motor2_stop< 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
   }

   //FALL 4
   else if (intervall_motor1_start > 180 && intervall_motor1_stop < 180) {
     float target_start = -(360 - intervall_motor1_start) / 0.05625;  // Beräkna target position
moturs
    moveMotor(motor1, target_start);
    moveMotor(motor2, round_start_position_motor2);
    Serial.println(motor1.currentPosition());
    Serial.println(motor2.currentPosition());
     int total_steps = round(((360/0.05625) - round_start_position + round_stop_position) /
step_size);
    while(i<=round_measure_steps){
      float flytande_start_postition = motor1.currentPosition();
      delay(pause_tid);//mätning sker här
      Serial.println("Mätning");
      float exact_position_motor1 = motor1.currentPosition();
      float accumulated_error_motor1 = 0.0;
      for (int j = 0; j < total_steps; j++) {
         moveMotorWithErrorTracking(motor1, accumulated_error_motor1, step_size);
         Serial.println(motor1.currentPosition());
         Serial.println(motor2.currentPosition());

         delay(pause_tid);//mätning sker här
         Serial.println("Mätning");
      }
        if(intervall_motor2_start>intervall_motor2_stop && intervall_motor2_start<=90 &&
intervall_motor2_stop>=-90){
       if(i<round_measure_steps){
```

```
                    moveMotorWithErrorTrackingMotor2(motor2,   accumulated_error_motor2,
steg_storlek_motor2);
          float return_start_position = flytande_start_postition - motor1.currentPosition();
          moveMotor(motor1, return_start_position);
          Serial.println(motor1.currentPosition());
          Serial.println(motor2.currentPosition());
        }
        i+=1;
      }

    }

    // Återgå till 0° via vänstra halvplanet (medurs)
    moveMotor(motor1, -motor1.currentPosition());

    if(intervall_motor2_stop > 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
    else if(intervall_motor2_stop< 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
   }
   Serial.println("KLAR");
   readyToMove = false;
   i=0;
  }

  //fixa start postiiton till motor1, de kommer vara olika för alla hur du räknar ut dem så klura
på de
  if(readyToMove && roll_1){
    //motor1-azimuth
    float start_position = intervall_motor1_start / 0.05625;
    float stop_position = intervall_motor1_stop / 0.05625;
    float measure_steps_motor1 = 0;

    float step_size = stegstorlek / 0.05625;
    int round_start_position = round(start_position);
    int round_stop_position = round(stop_position);
    int round_step_size = ceil(step_size);
    bool safety = true;
    //motor2-roll
    float steg_storlek_motor2 = stegstorlek2 / 0.05625;
    float start_position_motor2 = intervall_motor2_start/0.05625;
    float stop_position_motor2 = intervall_motor2_stop/0.05625;
    int round_start_position_motor2 = round(start_position_motor2);
    int round_stop_position_motor2 = round(stop_position_motor2);
    int round_step_size_motor2 = ceil(steg_storlek_motor2);

   float total_steps_roll = (start_position_motor2 - stop_position_motor2)/steg_storlek_motor2;
```

```
  if(intervall_motor1_start < 180 && intervall_motor1_stop <= 180){
    float antal_steg_motor1 = stop_position - start_position;
    measure_steps_motor1 = antal_steg_motor1/step_size;
  }
  else if (intervall_motor1_start < 180 && intervall_motor1_stop > 180) {
    float antal_steg_motor1 = stop_position - start_position;
    measure_steps_motor1 = antal_steg_motor1/step_size;
  }
  else if (intervall_motor1_start > 180 && intervall_motor1_stop > 180) {
    float antal_steg_motor1 = stop_position - start_position;
    measure_steps_motor1 = antal_steg_motor1/step_size;
  }
  else if (intervall_motor1_start > 180 && intervall_motor1_stop < 180){
    float antal_steg_motor1 = (360 - intervall_motor1_start)/0.05625 + stop_position;
    measure_steps_motor1 = antal_steg_motor1/step_size;
  }
  float diff_steg = (step_size - round_step_size);
  float diff_steg_motor2 = (steg_storlek_motor2 - round_step_size_motor2);
  //kanske inte använder det som är ovanför men lämnar kvar det för stunden
  if (intervall_motor1_start > 180 && intervall_motor1_stop > 180){
    float target_start = -((360 - intervall_motor1_start) / 0.05625);
    moveMotor(motor1, round(target_start));
  }
  else if (intervall_motor1_start > 180 && intervall_motor1_stop < 180) {
  float target_start = -(360 - intervall_motor1_start) / 0.05625;
  moveMotor(motor1, round(target_start));
  }
  else {
  moveMotor(motor1, round_start_position);
  }

  if (readyToMove){
    float accumulated_error_motor2 = 0.0;
    float accumulated_error_motor1 = 0.0;
    moveMotor(motor2, round_start_position_motor2);
    float current_start_motor2 = motor2.currentPosition();
    while (i<=round(measure_steps_motor1)){
      Serial.println(motor1.currentPosition());
      Serial.println(motor2.currentPosition());
      delay(pause_tid);//här sker en mätning per revolution
      Serial.println("Mätning");
      float exact_position_motor2 = motor2.currentPosition();
      float exact_position_motor1 = motor1.currentPosition();

      for (int j = 0; j < round(total_steps_roll); j++){
                  moveMotorWithErrorTrackingMotor2(motor2,    accumulated_error_motor2,
steg_storlek_motor2);
       Serial.println(motor1.currentPosition());
       Serial.println(motor2.currentPosition());
       delay(pause_tid);//här sker mätningar
```

```
      Serial.println("Mätning");
    }
    //fall 1
    if (intervall_motor1_start < 180 && intervall_motor1_stop <= 180){
      if(i<round(measure_steps_motor1)){
                              moveMotorWithCompensation(motor1,     exact_position_motor1,
accumulated_error_motor1, step_size);
       moveMotor(motor2, current_start_motor2 - motor2.currentPosition());
      }

      i+=1;

    }
    //fall 2
    else if (intervall_motor1_start < 180 && intervall_motor1_stop > 180){
      if(i<round(measure_steps_motor1)){
       if(motor1.currentPosition() < (180 / 0.05625)){
                              moveMotorWithCompensation(motor1,     exact_position_motor1,
accumulated_error_motor1, step_size);
        moveMotor(motor2, current_start_motor2 - motor2.currentPosition());
       }
       else {
        if(safety){
         moveMotor(motor1, -(360/0.05625));
         safety = false;
        }
                              moveMotorWithCompensation(motor1,     exact_position_motor1,
accumulated_error_motor1, step_size);
        moveMotor(motor2, current_start_motor2 - motor2.currentPosition());
       }
      }
      i+=1;
    }
    //fall 3
    else if (intervall_motor1_start > 180 && intervall_motor1_stop > 180){
      if(i<round(measure_steps_motor1)){
                              moveMotorWithCompensation(motor1,     exact_position_motor1,
accumulated_error_motor1, step_size);
       moveMotor(motor2, current_start_motor2 - motor2.currentPosition());
      }
      i+=1;
    }
    //fall 4
    else if (intervall_motor1_start > 180 && intervall_motor1_stop < 180){
      if(i<round(measure_steps_motor1)){
       moveMotorWithErrorTracking(motor1, accumulated_error_motor1, step_size);
       moveMotor(motor2, current_start_motor2 - motor2.currentPosition());
      }
      i+=1;
    }
```

```
    }
    if(motor1.currentPosition()>0){
      moveMotor(motor1, -motor1.currentPosition());
    }
    else if(motor1.currentPosition()<=0){
      moveMotor(motor1, -motor1.currentPosition());
    }

    if(intervall_motor2_stop > 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
    else if(intervall_motor2_stop< 0){
      moveMotor(motor2, -motor2.currentPosition());
    }
    Serial.println("KLAR");
    readyToMove = false;
    i=0;
   }
  }
}
```

HÖGSKOLAN
I BORÅS

Besöksadress: Allégatan 1 · Postadress: 501 90 Borås · Tfn: 033-435 40 00 · E-post: registrator@hb.se · Webb: www.hb.se