Opinnäytetyö (AMK)

Tieto- ja viestintätekniikka

2019

Marko Pekkala

VAIHEOHJATUN ANTENNIRYHMÄN KEILANMUODOSTUSOHJAIN

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tieto- ja viestintätekniikan koulutus

2019 | 26 sivua

Marko Pekkala

VAIHEOHJATUN ANTENNIRYHMÄN KEILANMUODOSTUSOHJAIN

Tämän opinnäytetyön tavoitteena oli keilanmuodostusohjaimen toteutus vaiheohjatulle antenniryhmälle. Keilanmuodostukseen käytettiin perinteistä keilanmuodostustekniikkaa, jolla tarkoitetaan vaiheohjatun antenniryhmän viivelinjojen säätämistä. Ohjausyksikön isäntälaitteelle toteutettiin käyttöliittymä, jolla voidaan sarjaliikenteen välityksellä lähettää keilanmuodostukseen liittyviä komentoja ohjausyksikölle.

Työ alkoi tutustumalla vaiheohjatun antenniryhmän rakenteeseen sekä keilanmuodostuksen teoriaan. Tämän jälkeen alkoi työn suunnittelu sekä toteutus. Ohjausyksikön ohjelmistossa käytettiin FreeRTOS-käyttöjärjestelmää tehtävien aikataulutuksen. Isäntälaitteen käyttöliittymä toteutettiin Python-ohjelmointikielellä sekä PyQt-kirjastolla.

Lopputuloksena toteutettiin toimiva keilanmuodostusohjain, joka toimii osana antennijärjestelmää. Antenniryhmän säteilykeilaa voidaan ohjausyksiköllä ohjata enintään ±60º, joka on tyypillistä vaiheohjatussa antenniryhmässä.

ASIASANAT:

keilanmuodostus, vaiheohjattu antenniryhmä, antenniryhmä

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information and Communications Technology

2019 | 26 pages

Marko Pekkala

BEAMFORMING CONTROLLER FOR PHASED ARRAY ANTENNA

The aim of this thesis was to implement a beamforming controller for phased array antenna. Conventional beamforming technique was used, meaning that the beamforming was achieved by adjusting the delay lines of the phased array antenna. A user interface was created for the host device of the controller to generate and send commands via serial communication.

The thesis began by gathering information about the phased array antenna and the theory of beamforming. Based on this information began the planning of the implementation and execution. The controller utilizes FreeRTOS real-time operating system to schedule tasking. The user interface was implemented using Python programming language and PyQt Python binding.

As a result, a functional beamforming controller was implement, which functions as part of the antenna system. The beam can be steered with the controller by ±60º, which is typical for phased array antennas.

KEYWORDS:

beamforming, phased array antenna, array antenna

SISÄLTÖ

SANASTO

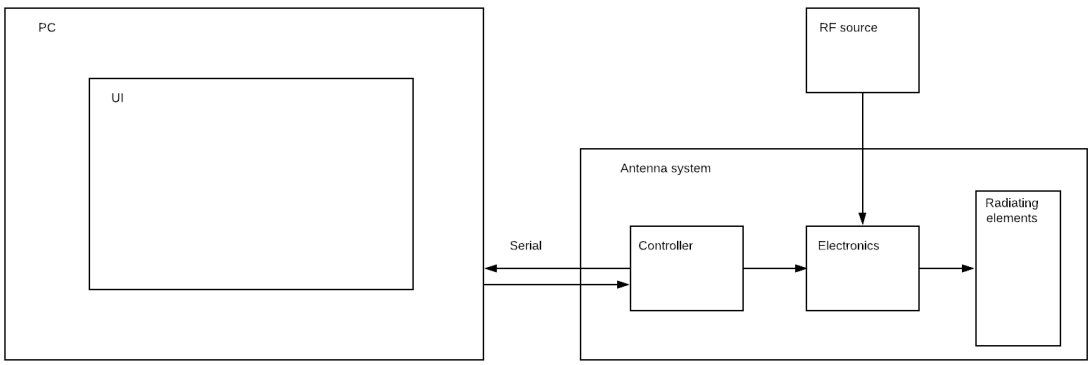
d	etäisyys
λ	aallonpituus
Θ	säteilykulma
Δφ	vaihe-ero
ASCII	standardisoitu 7-bittinen tietokoneiden merkistö (american standard code for information interchange)
FPGA	ohjelmoitava logiikkapiiri (field-programmable gate array)
FreeRTOS	sulautettuihin laitteisiin tarkoitettu reaaliaikakäyttöjärjestelmä
Ghz	gigahertsi
I2C	tiedonsiirtoväylä (inter-integrated circuit)
IO	siirräntä (input/output)
Mhz	megahertsi
PC	tietokone (personal computer)
PyQt	Qt laajennus Python-ohjelmointikieleen
Python	Ohjelmointikieli
Qt	alustariippumaton ohjelmistojen ja käyttöliittymien kehitysympäristö
RDR	sarjaliikenteessä vastaanotetun datan rekisteri (receive data register)
RF	radiotaajuus (radio frequency)
UI	käyttöliittymä (user interface)
USART	sarjaliikenteen lähetys- ja vastaanottopiiri (Universal Synchronous/Asynchronous Receiver-Transmitter)
XOR	poissulkeva tai -looginen portti
ms	millisekunti
µs	mikrosekunti

1 JOHDANTO

Antennitekniikassa keilanmuodostusta käytetään antennin säteilykuvion muokkaamiseen. Antennin säteilykuviota muokkaamalla voidaan sen lähetystehoa ohjata halutulle alueelle. Lähetystehon ohjaaminen on toivottu ominaisuus esimerkiksi antenneissa, joita käytetään matkapuhelinverkon tukiasemissa.

Tukiasemissa käytettävät antennit eivät voi olla pelkästään ympärisäteileviä, koska silloin tukiaseman kantama jäisi suhteellisen lyhyeksi. Käyttämällä pelkästään kapeakeilaisia suunta-antenneja tukiaseman kantamaa voidaan pidentää, mutta käytettävien antennien lukumäärä kasvaisi. Tästä syystä ratkaisu ei ole kustannustehokas. Käyttämällä keilanmuodostusta tukiasemissa käytettävissä antenneissa saadaan tukiasemalle suurempi kattavuusalue pienemmällä antennimäärällä, sillä yksittäisen antennin lähetyskeila voidaan asettaa kapeaksi ja se voidaan ohjata älykkäästi haluttuun suuntaa.

Työn tavoitteena on tutustua ojattavien suunta-antennien toimintaan ja toteuttaa ohjausyksikkö, jolla keilanmuodostus aikaansaadaan. Lisäksi ohjausyksikköä haluttiin hallita isäntälaitteelta. Tätä varten luotiin Python-ohjelmointikielellä työkalu sekä yksinkertainen käyttöliittymä, jolla keilanmuodostusta voidaan helposti ohjata isäntälaitteelta. Työ toteutetaan osana toista opinnäytetyötä, jossa suunnitellaan antennijärjestelmään kuuluva elektroniikka.



Kuvio 1: Lohkokaavio antennijärjestelmästä

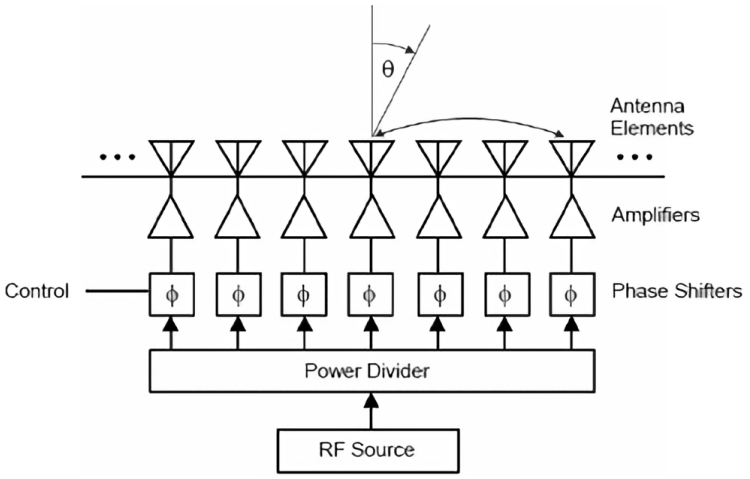
Työssä käytetty keilanmuodostustekniikka rajataan perinteiseen keilanmuodostukseen, joka perustuu vaiheohjatun antenniryhmän viivelinjojen säätämiseen.

Aluksi selvitetään vaiheohjatun antenniryhmän rakenne sekä keilanmuodostuksen teoriaa. Tämän jälkeen tarkastellaan eri toteutustapoja, jolla keilanmuodostus aikaansaadaan käytössä olevalla antennijärjestelmällä. Lopuksi testataan antennijärjestelmän keilanmuodostuksen toimivuus sekä mitataan ohjausyksikön toimintanopeus.

2 VAIHEOHJATTU ANTENNIRYHMÄ

Vaiheohjatulla antenniryhmällä tarkoitetaan antennia, joka koostuu useasta säteilevästä elementistä. Jokainen elementeistä on kytketty erilliseen vaihesiirtimeen. Vaiheohjatun antenniryhmän säteilykuviota voidaan muuttaa ohjaamalla sen vaihesiirtimiä. Tätä kutsutaan perinteiseksi keilanmuodostukseksi (engl. Conventional beamforming). Säteilykuviota muuttamalla voidaan lähetettävän keilan suuntaa, sekä vahvuutta ohjata. Keilaa vastaanottaessa säteilykuvion muuttamisella voidaan vaimentaa häiriösignaaleja ja lisätä herkkyyttä halutulle alueelle. [1]

Elementit asetetaan useimmissa käyttötapauksissa rinnakkain. Riviin asetettuina antenniryhmän säteilykulmaa voidaan muuttaa yhdessä ulottuvuudessa. Elementit voidaan asetella myös tasoon, jolloin antenniryhmän säteilykulmaa voidaan muuttaa kahdessa ulottuvuudessa. [2]



Kuva 1: Lineaarisen antenniryhmän lohkokaavio [3]

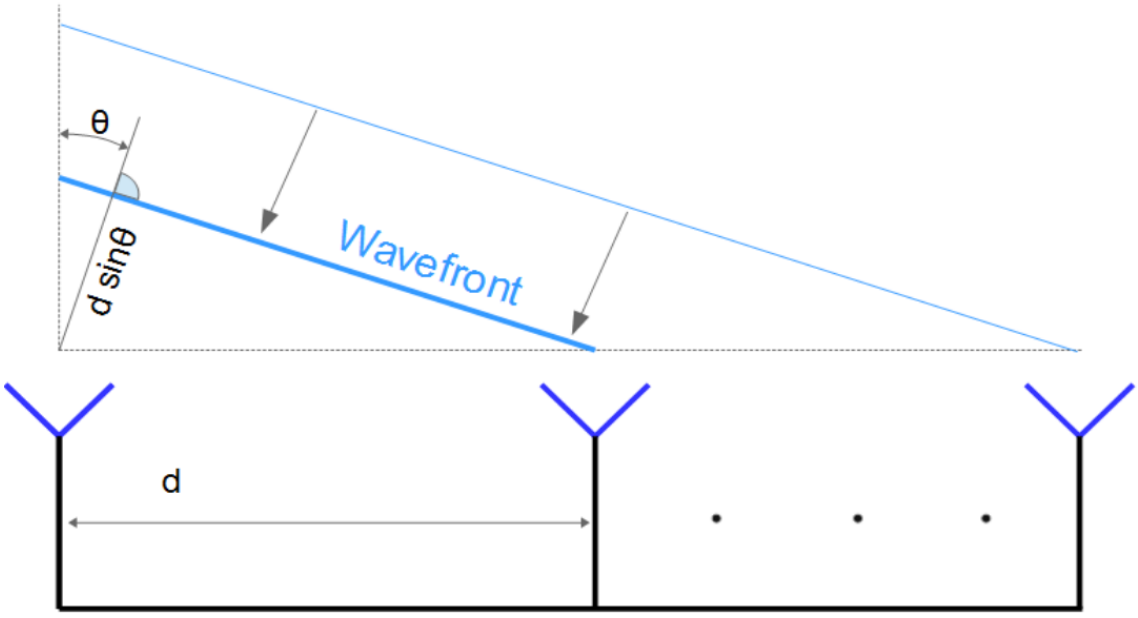
2.1 Keilanmuodostus

Keilanmuodostuksella tarkoitetaan tekniikkaa, jolla muokataan vaiheohjatun antenniryhmän sätelykuviota. Perinteinen keilanmuodostus vaiheohjatussa antenniryhmässä perustuu säteilevien elementtien väliseen vaihe-eroon. Vaihe-ero aiheuttaa interferenssiä säteileviin aaltoihin, josta seuraa vahvistuneita tai vaimentuneita alueita antenniryhmän säteilykuviossa.

Elementtien välinen vaihe-ero voidaan laskea kaavalla 1, jossa Δφ on vaihe-ero elementtien välillä, d on elementtien välinen etäisyys, θ on haluttu säteilykulma ja λ on signaalin aallonpituus. [5]

$$\Delta\varphi = \frac{2\pi d \sin\Theta}{\lambda}$$

Kaava 1: Vaihe-eron laskukaava

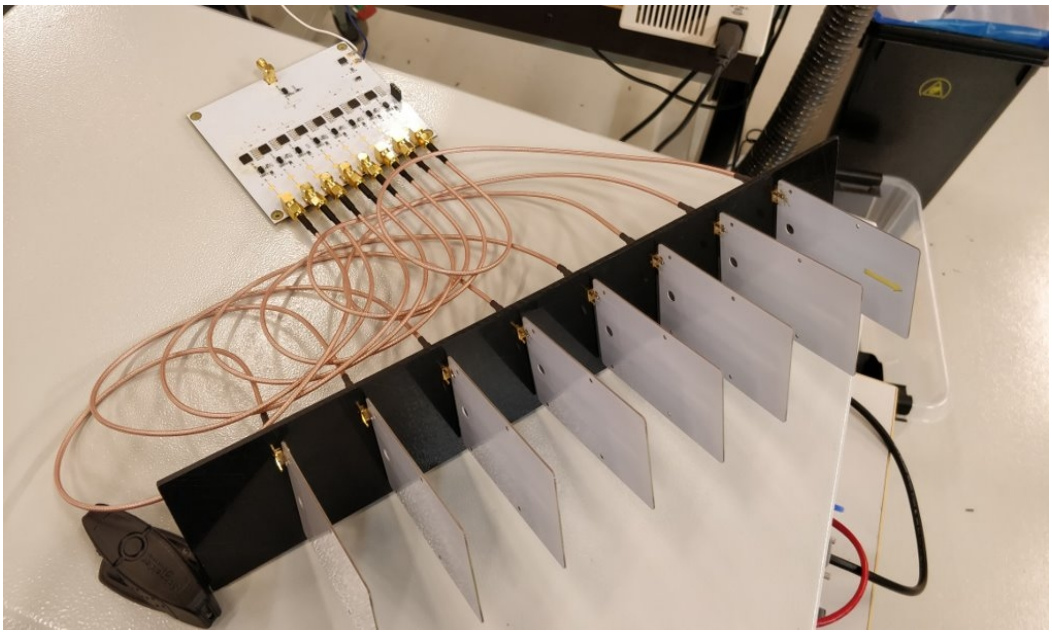


Kuva 2: Visualisaatio vaihe-eron tuottamasta etäisyydestä säteiltävään signaaliin [5]

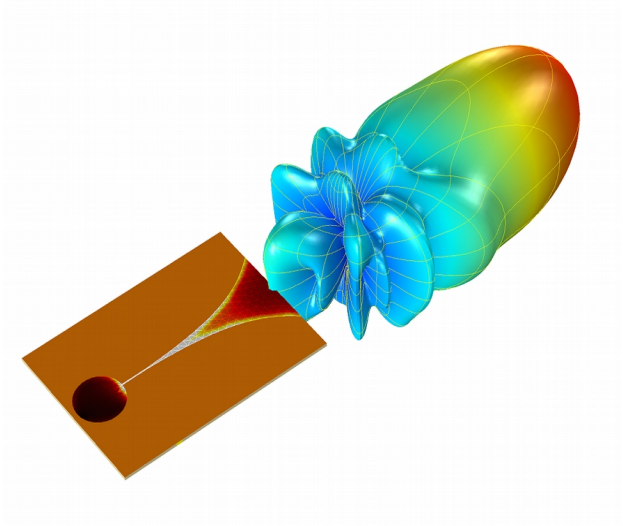
2.2 Työssä käytetty antenniryhmä

Työssä käytetty vaiheohjattu antenniryhmä koostuu kahdeksasta laajakaistaisesta piirilevyantennista, jotka on asetettu riviin. Antenniryhmän elementit sekä elektroniikka on sovitettu 3.6 Ghz:n taajuuteen.

Antennielementtien viivelinjoja ohjataan 6-bittisillä digitaalisilla vaiheensiirtimillä. Vaiheensiirtimien vaiheresoluution on 5,652º, joka vastaa antenniryhmän säteilevän keilan kulmaresoluutiota.



Kuva 3: Työssä käytetty antenniryhmä



Kuva 4: Laajakaistainen piirilevyantenni [6]

3 OHJAUSYKSIKÖN TOTEUTUS

3.1 Vaativuusmäärittely

Työn tavoitteeksi määriteltiin ohjausyksikön suunnittelu, joka ohjaisi vaiheohjatun antenniryhmän vaihesiirtimiä. Antennin ohjausyksikkö toimii orjalaitteena, jota ohjataan isäntälaitteelta. Isäntälaitteessa olevalla käyttöliittymällä voidaan luoda ja lähettää komentoviestejä ohjausyksikköön.

Työssä käytettiin STMicroelectronicsin STM32F091RC kehitysalustaa. Kehitysalustassa oleva mikroprosessori on ARM Cortex-M0, jonka kellotaajuus on 48 MHz.

Toteutuksen tulee olla helposti skaalattava sekä modulaarinen. Näin varmistetaan, että antennielementtien lukumäärää ja muita laitteiston komponentteija voidaan myöhemmin muuttaa vaivattomasti ilma suurta uudelleensuunnittelua.

3.2 Vaihesiirtimien ohjaaminen

Riippuen antenniryhmän vaihesiirtimien määrästä, ohjattavien bittien lukumäärä voi kasvaa suureksi. Koska laitteistosta haluttiin helposti skaalattava sekä modulaarinen, bittien ohjaamiseen oli löydettävä tapa, joka ei olisi sidoksissa valittuun mikroprosessoriin.

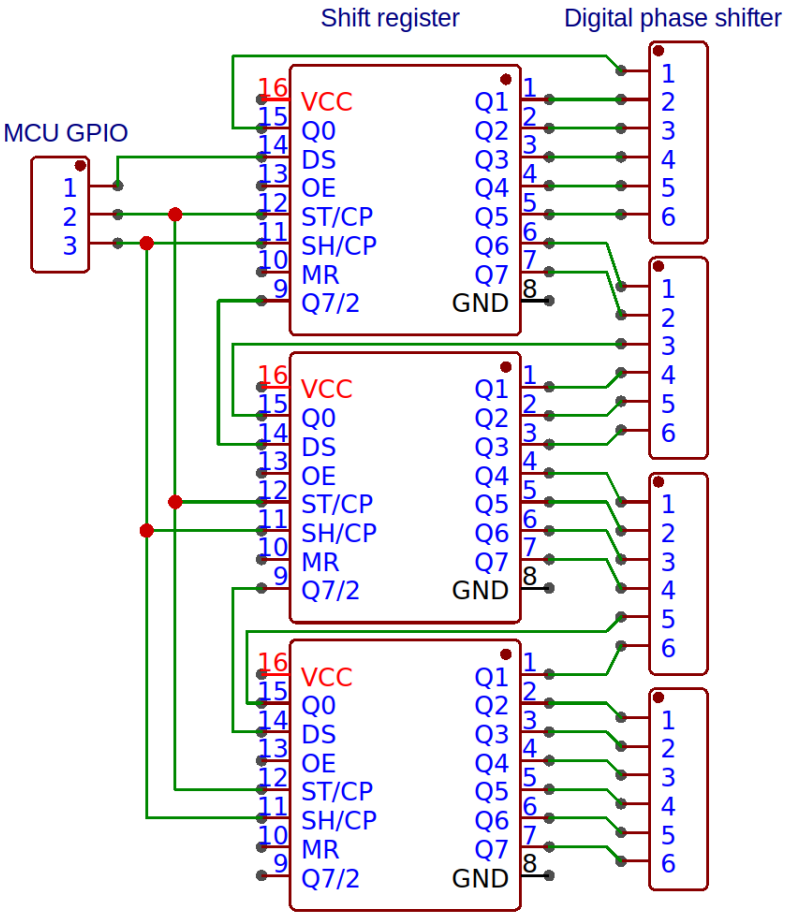
Harkittuja vaihtoehtoja usean bitin ohjaamiseen olivat mikroprosessorin IO-linjojen jakaminen ja ohjaus käyttäen multipleksereitä, I2C laajentemia tai siirtorekistereitä. Lisäksi yksi vaihtoehdoista olisi ollut käyttää FPGA-piiriä mikroprosessorin sijasta. FPGA-piirillä laskusuoritukset vaihe-erojen selvittämiseksi sekä bittien asettaminen on huomattavasti nopeampaa kuin samojen toimenpiteiden suorittaminen mikroprosessorilla.

Ketjuttamalla multipleksereitä tai I2C laajentemia, yksittäisen bitin asettaminen olisi vaatinut useita operaatioita, jotta mikroprosessorin IO-linja voitaisiin ohjata oikeaan vaihesiirtimen bittiin. Tämä olisi tuonut ylimääräistä monimutkaisuutta ohjelman toteutukseen ja hidastanut laitteiston toimintaa.

Muuttamalla antennin konfiguraatiota yksi vaihesiirtimen bitti kerrallaan olisi tarkoittanut sitä, että antenni suuntaus olisi ollut hetkellisesti väärä muutoksen aikana. Tämä voi aiheuttaa häiriösignaalien syntymistä radioaaltoa lähettäessä. Siksi konfiguraation muutoksen tulee olla välitöntä. FPGA-piiri ja siirtorekisterit olivat tästä syystä hyviä vaihtoehtoja vaihesiirtimien ohjaamiseen, sillä niitä käyttämällä voidaan asettaa usean IO-linjan tila samanaikaisesti.

FPGA-piirin käyttö olisi kuitenkin poissulkenut toteutuksen skaalattavuuden sekä modulaarisuuden, sillä toteutus olisi ollut sidottu valittuun FPGA-piiriin.

Lopulta ohjaus päätettiin toteuttaa ketjuttamalla siirtorekisterejä. Siirtorekisterin ohjaukseen vaaditaan ainoastaan 3 IO-linjaa, eikä siihen vaikuta ketjutettujen siirtorekisterien lukumäärä. Siirtorekisteriketjun ohjaukseen voidaankin siis käyttää lähestulkoon mitä tahansa mikroprosessoria.



Kuva 5: Havainnollistava kuva vaihesiirtimien kytkennästä

3.3 Sulautettu ohjelmisto

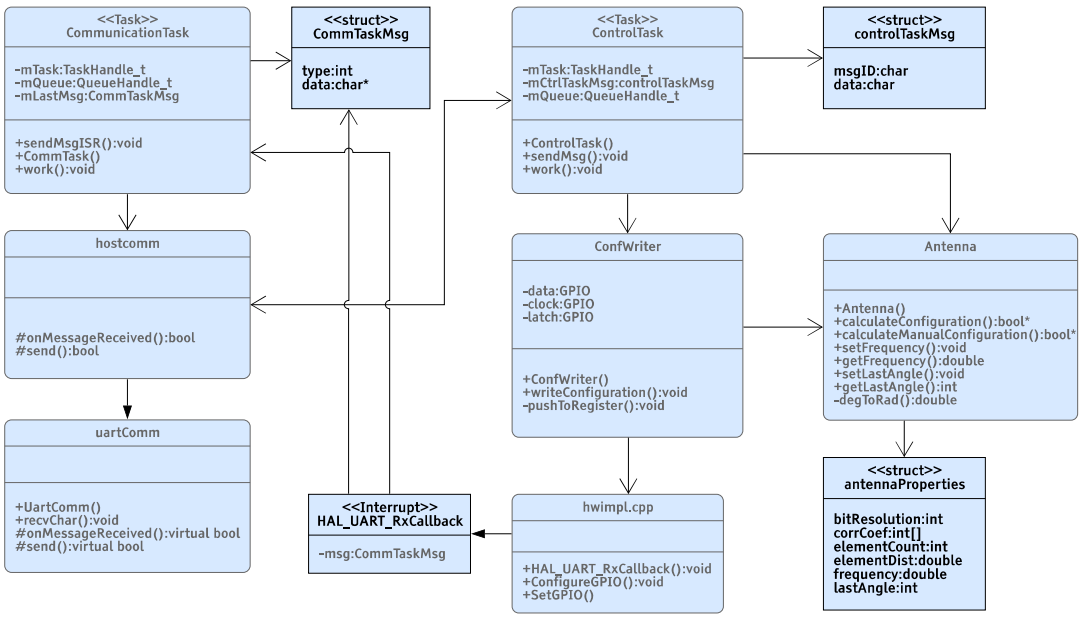
Ohjelmisto on toteutettu C++ -ohjelmointikielellä. Ohjelmisto suorittaa kahta tehtävää (säiettä) samanaikaisesti. Tehtävien aikatauluttamiseen käytettiin FreeRTOS-reaaliaikakäyttöjärjestelmää.

3.3.1 FreeRTOS-käyttöjärjestelmä

FreeRTOS (Free Real Time Operating System) on sulautettuihin laitteisiin tarkoitettu reaaliaikakäyttöjärjestelmä. FreeRTOS tarjoaa tärkeimmät realiakaisen aikataulutuksen toiminnot, kuten säikeidenvälisen viestinnän, aikataulutuksen ja synkronoinnin primitiivit [7]

3.3.2 Arkkitehtuuri

Ohjelmisto suorittaa rinnakkain kahta tehtävää, viestintä- ja ohjaustehtävää. Viestintätehtävä vastaa sarjaliikenteen välityksellä kulkevien viestien lähettämisestä sekä vastaanottamisesta. Vastaanotetut viestit luovutetaan ohjaustehtävälle, joka tulkitsee sekä suorittaa viestien sisältämät komennot.



Kuvio 2: Havainnollistava luokkakaavio ohjelmiston arkkitehtuurista

Keskeytysrutiini, jota kutsutaan sarjaportin rekisterin täyttyessä, on toteutettu hwimpl.cpp lähdekooditiedostossa. Keskeytysrutiinia kutsutaan, kun rekisteriin on vastaanotettu 8 databittiä. Vastaanotettu tavu asetetaan keskeytysrutiinissa viestintätehtävän puskuriin.

Viestintätehtävä tarkistaa ajoittain puskirin sisällön, ja ohjaa sen sisältämät tavut uartComm luokan recvChar funktion käsiteltäväksi. Funktio muodostaa vastaanotetuista tavuista ControlTaskMsg-tyyppisen viestin. Kun kokonainen viesti on vastaanotettu ja sen tarkistussumma vastaa viestin sisältöä, kutsutaan hostComm luokan onMessageReceived funktiota, joka välittää viestin ohjaustehtävälle.

Ohjaustehtävä tarkistaa ajoittain puskurin, johon vastaanotetut ohjausviestit saapuvat. Vastaanotetun ohjausviestin sisältämä tunniste luetaan, jonka perusteella suoritetaan eri toimenpiteitä.

Antenna luokka sisältää antenniin liittyvän toiminnallisuuden. Funktio calculateConfiguration vastaanottaa halutun antennin lähetyskulman ja palauttaa osoittimen bittijonon alkuun. Bittijono vastaa vaihesiirtimien konfiguraatioita, jolla haluttu lähetyskulma muodostetaan. Bittijonon laskemisessa käytetään antennaProperties tyyppistä tietuetta, joka sisältää antenniin liittyviä ominaisuuksia.

ConfWriter luokka sisältää funktiota, joilla vaihesiirtimien konfiguraatiot asetetaan. Funktio writeConfiguration kirjoittaa calculateConfiguration funktion palauttaman bittijonon siirtorekisteriketjuun.

3.3.3 Kommunikaatio

Kommunikaatio isäntälaitteen ja ohjausyksikön välillä toteutettiin sarjaliikenteellä. Isäntälaitteesta lähetetään ohjausyksikölle jäsenneltyjä viestejä, joiden mukaan ohjausyksikkö suorittaa toimenpiteitä.

Viestien merkistönä käytettään ACII-merkkejä jotka on koodattu UTF-8 koodaustavalla. Yhden merkin pituus on 8 bittiä. STM32F0xx perheen USART implementaatioissa lähetettävän tai vastaanotettavan sanan pituudeksi voidaan asettaa 7, 8 tai 9 databittiä. Vastaanotetut databitit tallennetaan USART RDR rekisteriin (receive data register), jonka koko on enintään 9 bittiä.

Viestin rakenne koostuu komennosta, komentoon liittyvästä arvosta sekä tarkistussummasta. [(Komento)(Arvo)/(Tarkistussumma)+]. Vinoviivaa käytetään erottelemaan arvo ja tarkistussumma toisistaan. Plusmerkillä ilmaistaan viestin päättymisestä. Mahdolliset komennot näkyvät taulukossa 1.

Kulman asettaminen	A
Kulman pyyntö	a
Taajuuden asettaminen	F
Taajuuden pyyntö	f
Manuaalinen konfiguraation asettamien	M

Taulukko 1: Ohjauskomennot

Esimerkkinä viesti, jolla asetetaan antennin lähetyskulma -30º:seen: A-30/111+.

Tarkistussumma lasketaan suorittamalla XOR operaatio iteratiivisesti viestin jokaiseen merkkiin. Operaatio suoritetaan ASCII-merkin numeeriseen arvoon. Toimenpide suoritetaan isäntälaitteessa viestin muodostuksen yhteydessä, sekä ohjausyksikön

viestintätehtävässä. Vastaanotettu komento suoritetaan, jos tarkistussumma vastaa vastaanotetun viestin laskettua tarkistussummaa.

Ohjausyksikön on myös lähetettävä vastausviesti tiettyjen komentojen kohdalla. Vastausviesti sisältää pelkästään pyydetyn arvon, eikä se sisällä tarkistussummaa.

3.3.4 Kilpailutilanteiden tarkastelu

Koska ohjelma suorittaa samanaikaisesti kahta asynkronista tehtävää, on tärkeää tarkastella mahdollisten kilpailutilanteiden (data race) olemassaoloa. Kilpailutilanteita voi syntyä, kun usea asynkroninen tehtävä muokkaa jaettua resurssia.

Kuviosta 2 nähdään, että viestintätehtävällä sekä ohjaustehtävällä on assosiaatio asynkroniseen olioon tai metodiin. Tehtävät on kuitenkin toteutettu siten, että ulkopuolinen entiteetti voi asettaa arvoja pelkästään tehtävän puskuriin. Näin varmistetaan, ettei tehtävän resursseja muuteta ulkopuolelta kesken tehtävän suorituksen.

Kun hostcomm olio lähettää resurssin ohjaustehtävälle, se luo controlTaskMsg-tyyppisen tietueen. Tietueesta lähetetään kopio FreeRTOS:n tarjoamalla xQueueSendToBackFromISR-funktiolla. Vastaavanlainen toimenpide suoritetaan, kun sarjaliikenteellä vastaanotettu tavu aiheuttaa keskeytysrutiinin, jossa tavu siirretään viestintätehtävän puskuriin.

Antenna luokkaa käytetään viestintätehtävässä useaan otteeseen, ControlTask luokasta sekä ConfWriter luokasta. Koska Antenna luokka on kuitenkin pelkästään viestintätehtävän käytössä ja tehtävä suorittaa pelkästään puskurin käsittelyä, ei kilpailutilanne ole mahdollista.

3.3.5 Realiaikavaatimusten tarkastelu

Tässä kappaleessa tarkastellaan järjestelmän kulmanohjauskomennon suorittamisen realiaikavaatimuksia ja komennon vastaaottamiseen liittyvistä rajoitteista.

Sarjaportin RDR rekisterin täyttyessä tapahtuu laitteistokeskeytys. Keskeytysrutiinissa RDR rekisterin sisältö tallennetaan CommTaskMsg-tietueeseen. Kopio tietueesta siirretään viestintätehtävän puskuriin. Tietue siirretään puskuriin FreeRTOS:n tarjoaman xQueueSendToBackFromISR-funktiolla. Funktio suoritetaan välittömästi, eikä se odota esimerkiksi tehtävän suoriutumista, jolla on suurempi prioriteetti.

Sarjaliikenteen tiedonsiirtonopeudeksi (baudinopeus) on asetettu 115200 bps. Yhden tavun siirtämiseen kuluva aika on noin 69,4 µs. Sekunnissa keskeytyksiä voi siis olla enintään 14400 kappaletta. Jotta keskeytysrutiini ehditään suorittaa loppuun ennen

seuraavan keskeytysrutiinin alkua, se saa sisältää enintään 3333 prosessorisykliä 48 Mhz:n kellotaajuuden omaavassa prosessorissa.

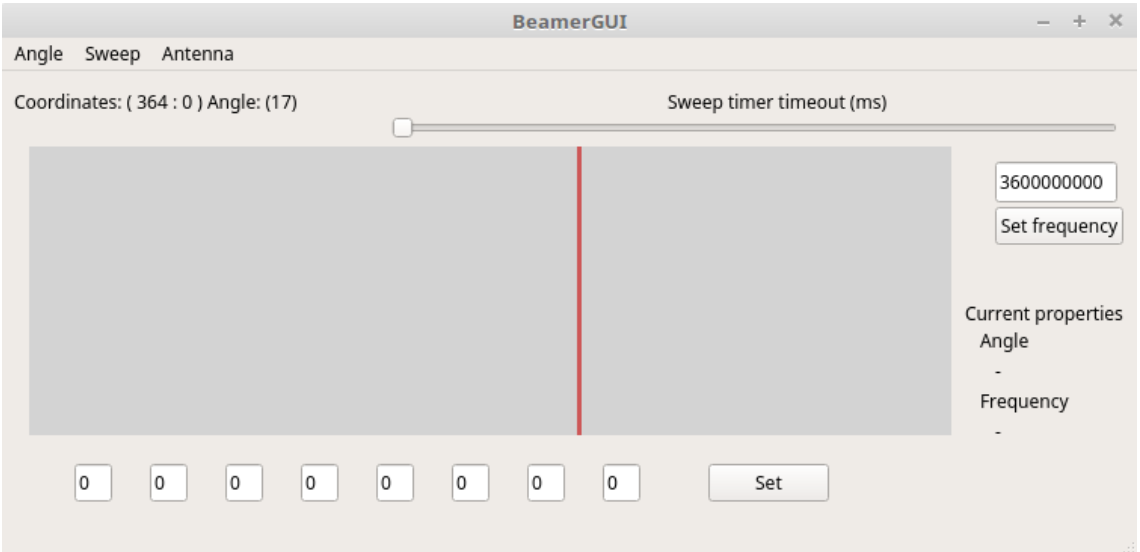
Komennon pituus isäntälaitteelta ohjausyksikölle, jolla asetetaan antennijärjestelmän säteilykulma, on enintään 9 tavua. Viestin siirtämiseen kuluu aikaa vähintään 625 µs. Kappaleessa 4.1 selvitetään kulman asettamiseen kuluvaksi ajaksi 1 ms. Tästä voidaan päätellä, että viestintätehtävän puskuri täyttyy 1,6 kertaa nopeammin, kuin sitä ehditään käsittelemään.

Viestintätehtävän puskuriin mahtuu 128 CommTaskMsg-tietuetta, joka vastaa vähintään 14 kulmanohjauskomentoa. Jos kulmanohjauskomentoja lähetettäisiin jatkuvasti isäntälaitteesta, ehtisi ohjausyksikkö käsitellä 37 komentoa, ennen kuin puskuri täyttyisi.

Kun viestintätehtävän puskuri täyttyy, sarjaliikenteen yli vastaanotettuja tavuja ei tallenneta. Tämä johtaa tilanteisiin, jossa osa ohjauskomennon arvoista on virheellisiä. Tästä syystä, kun ohjauskomentoja muodostetaan vastaanotetuista tavuista, lasketaan komennolle lopuksi tarkistussumma. Näin virheellisiä komentoja ei koskaan suoriteta, jos tiedonsiirto epäonnistuu.

3.4 Käyttöliittymä

Isäntälaitteessa käytettävän ohjelman tarkoituksena on tarjota käyttöliittymä, jolla antennia on yksinkertaista ohjata. Käyttöliittymä on toteutettu Python-ohjelmointikielellä ja käyttöliittymän graaffiset komponentit on toteutettu PyQt-kirjastolla. PyQt on Pythonille luotu kirjasto, joka perustuu Qt ohjelmistokehykseen. Qt on alustariippumaton ja sitä käytetään ohjelmistojen sekä käyttöliittymien kehittämisessä. [8]



Kuva 6: Ohjausyksikön käyttöliittymä

Antennin lähetyskulmaa voidaan säätää angle-valikosta 15º:n välein -90º:sta 90º:seen. Lähetyskulma voidaan asettaa myös raahaamalla kursoria tunnistusalueella. Hiiren painikkeen ollessa alhaalla kulman asettavia komentoviestejä lähetetään ohjausyksikölle. Näin lähetyskeilaa voidaan helposti suunnata kursorin osoittamaan suuntaan 1º:n asteen resoluutiolla.

Sweep-valikosta voidaan käynnistää pyyhkäisy. Pyyhkäisyn ollessa päällä, antenni asettaa uuden lähetyskulman ajoittain kiertäen -90º:sta 90º:seen ja takaisin. Pyyhkäisyominaisuus on toteutettu käyttöliittymässä ja sen toiminta perustuu ajoittaiseen komentoviestien lähettämiseen ohjausyksikölle. Pyyhkäisyn nopeutta voidaan säätää liukusäätimellä.

Antenna-valikosta voidaan kysyä antennin senhetkisiä asetuksia kuten viimeksi asetettu taajuutta tai kulmaa.

Käyttöliittymän oikeasta reunasta voidaan ohjausyksikölle asettaa myös lähetettävän signaalin taajuus. Taajuutta käytetään, kun vaihesiirtimien konfiguraatioita lasketaan. Alareunassa sijaitsevilla kentillä voidaan vaihesiirtimien konfiguraatiot asettaa manuaalisesti.

4 OHJAUSYKSIKÖN TESTAUS

4.1 Toimintanopeus

Ohjausyksikön toimintanopeutta tarkasteltiin kellottamalla IO-linjoja oskiloskoopilla. Laitteiston toimintanopeutta ei ollut määritelty vaatimusmäärittelyssä, mutta laitteistosta haluttiin kuitenkin mahdollisimman nopea.

Antenna luokan calculateConfiguration funktio, jossa selvitetään vaihesiirtimien konfiguraatio antennin lähetyskulmalle, kuluttaa aikaa 875 µs. ConfWriter luokan writeConfiguration funktio, jossa vaihesiirtimien konfiguraatio kirjoitetaan siirtorekistereihin, kuluttaa aikaa 125 µs. Yksittäisen kulman asettamiseen kuluu siis 1 ms.

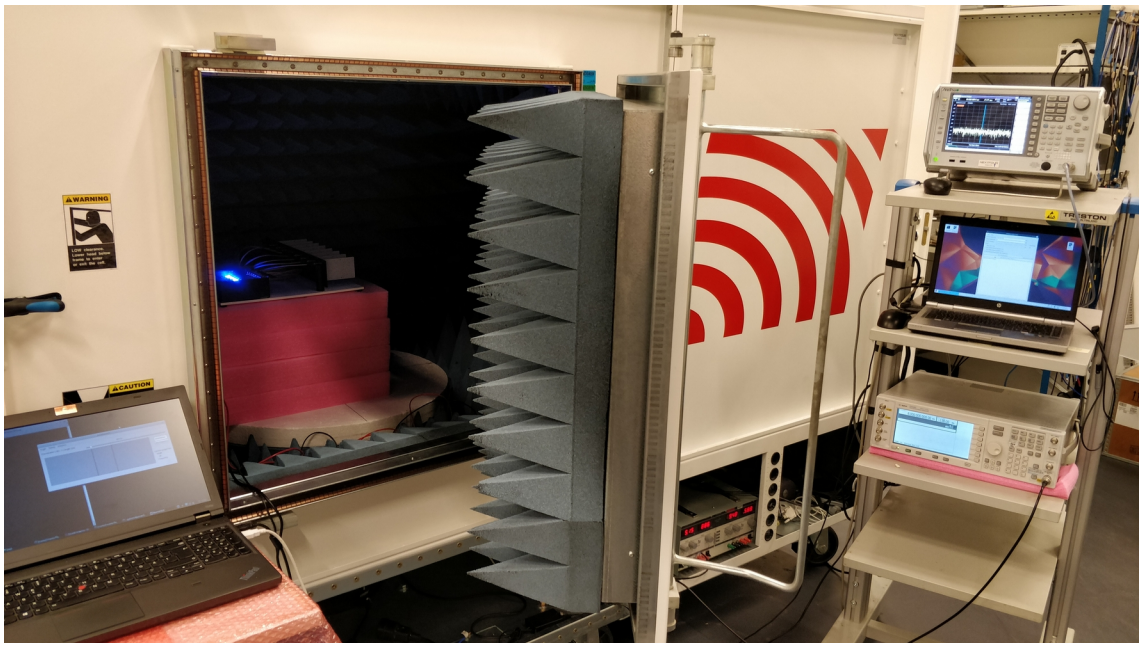
Tämän lisäksi viivettä järjestelmään aiheuttaa sarjaliikennteen siirtonopeus sekä viestin tulkistemiseen kuluva aika. Sarjaliikenteen tiedonsiirtonopeudeksi on asetettu 115200 bps. Kulmanohjauskomennon koko voi olla enintään 9 tavua. Ohjauskomennon siirtämiseen aikaa kuluu siis vähintään 625 µs. Kokonaisuudessa ohjauskomennon siirtämiseen sekä kulman asettamiseen ohjausyksikössä kuluu aikaa noin 1,6 ms.

Kulman asettamiseen liittyvät laskutoimenpiteet sekä tiedonsiirto ovat järjestelmän pullonkaula. Laskusuorituksia voidaan jatkossa optimoida esimerkiksi käyttämään pelkästään kokonaislukuja sekä hyödyntämällä sini-hakutaulukkoa. Tiedonsiirtoon voidaan käyttää vaihtoehtoisia tiedonsiirtomenetelmiä.

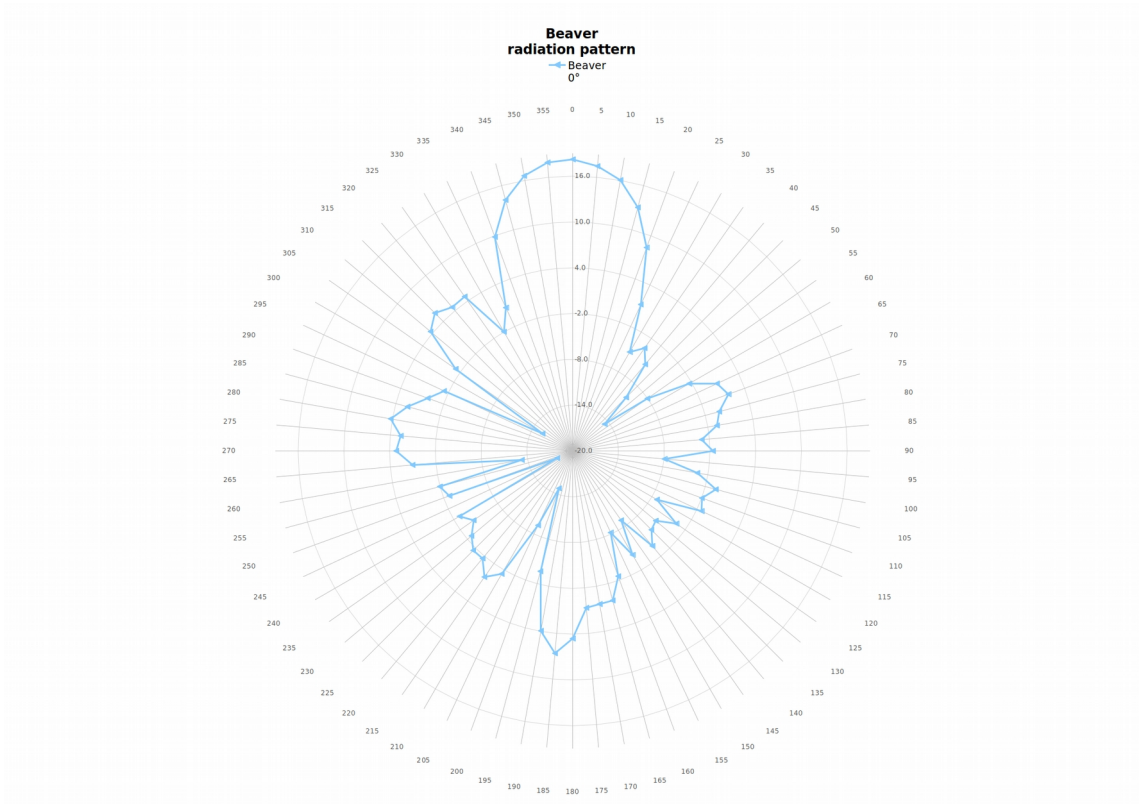
4.2 Suuntakuviomittaukset

Suuntakuviomittaukset toteutettiin antenni-mittakammiossa. Antennijärjestelmän lähetyskulma mitattiin -90º:sta 90º:seen 15º:n välein. Tuloksista nähdään, että keilanmuodostusohjain toimii odotetusti.

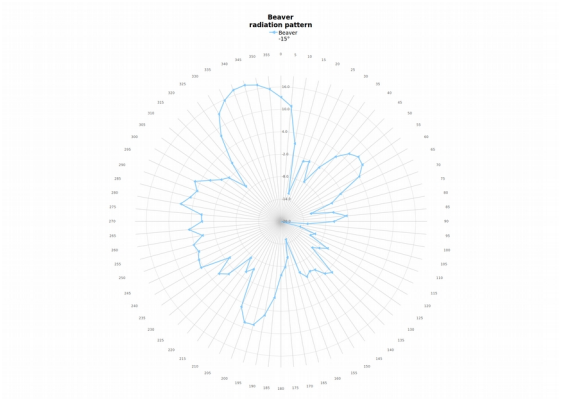
Antennijärjestelmä saavuttaa pääkeilaan noin 18 dB:n vahvistuksen ja sivukeilat jäävät suurimmassa osassa tuloksista alle 10 dB:n vahvistukseen. Tuloksista huomataan, että suurimmissa säteilykulmissa pääkeilan leveys kasvaa huomattavasti ja sivukeilojen vahvistus kasvaa. Tämän lisäksi voidaan nähdä, että antennijärjestelmällä saavutetaan maksimissaan ±60º:n säteilykulma, joka on tyypillistä vaiheohjatussa antenniryhmässä. [9]



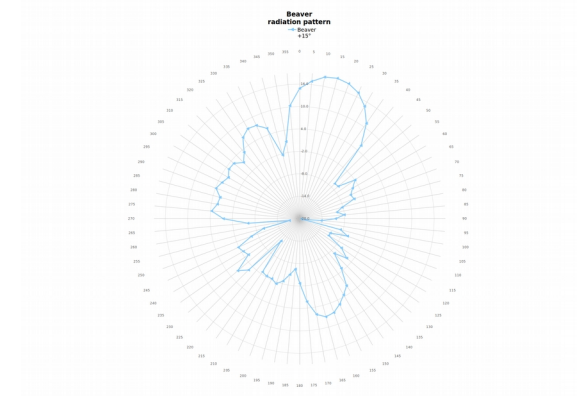
Kuvio 3: Mittausympäristö



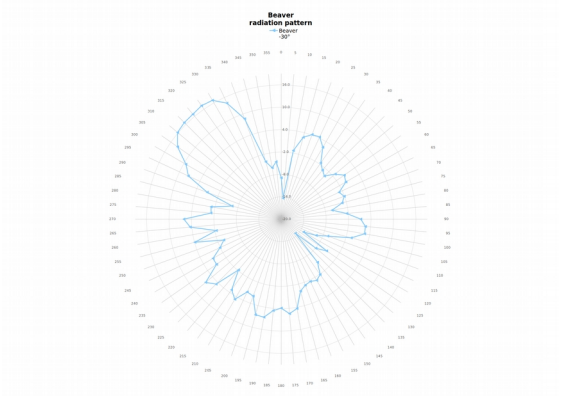
Kuvio 4: 0° suuntakuvio



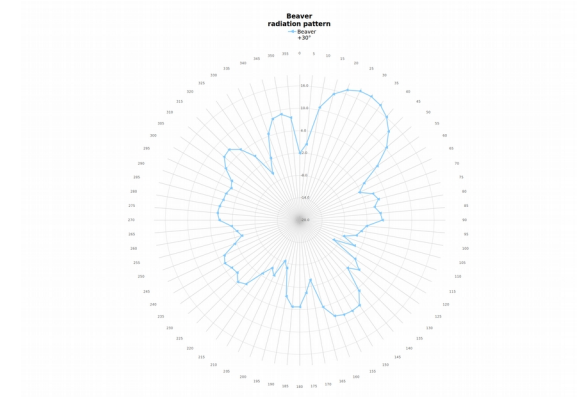
Kuvio 6: -15° suuntakuvio



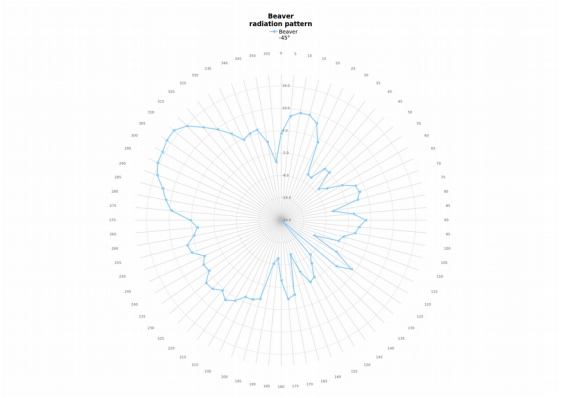
Kuvio 5: 15° suuntakuvio



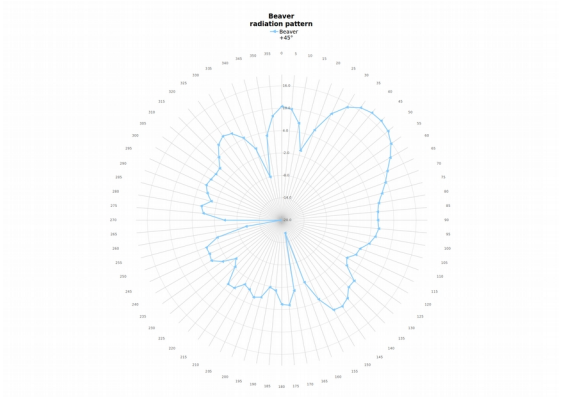
Kuvio 7: -30° suuntakuvio



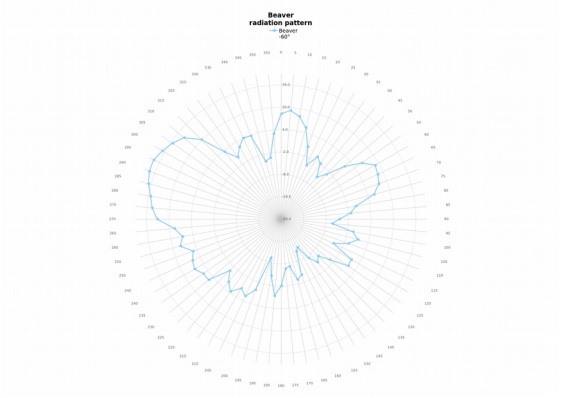
Kuvio 8: 30° suuntakuvio



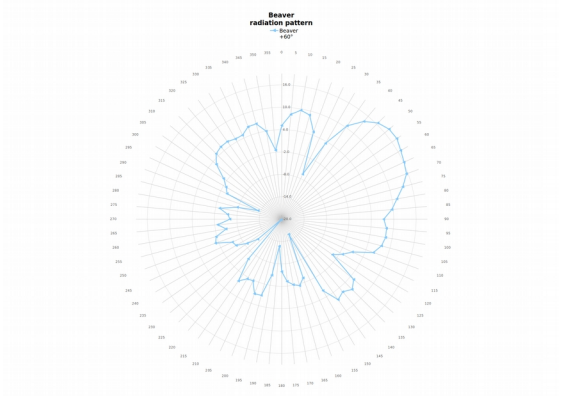
Kuvio 9: -45° suuntakuvio



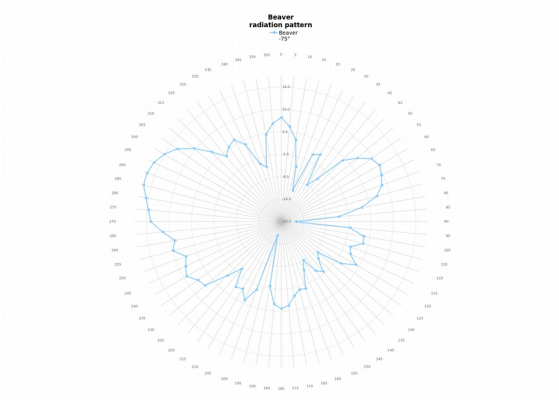
Kuvio 10: 45° suuntakuvio



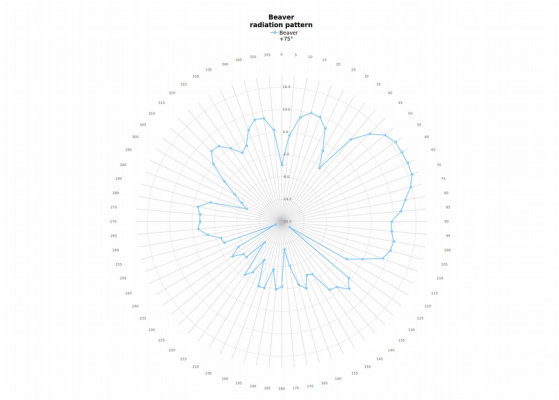
Kuvio 11: -60° suuntakuvio



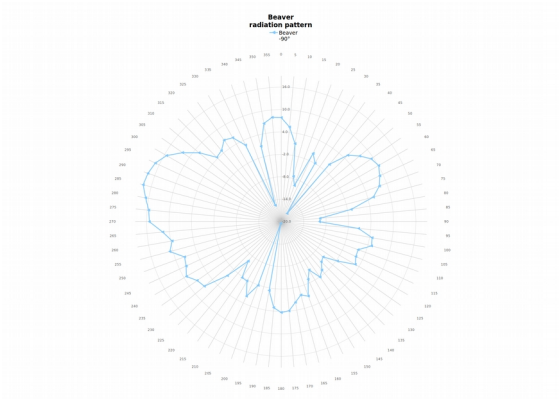
Kuvio 12: 60° suuntakuvio



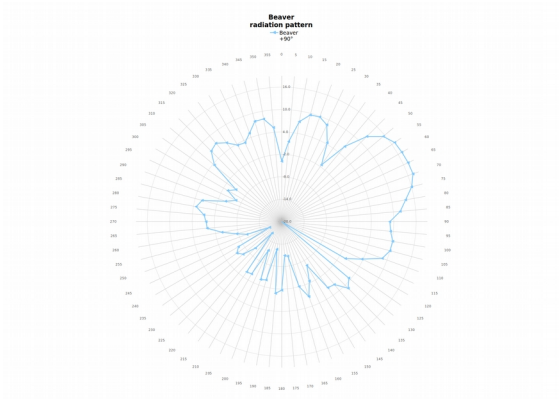
Kuvio 13: -75° suuntakuvio



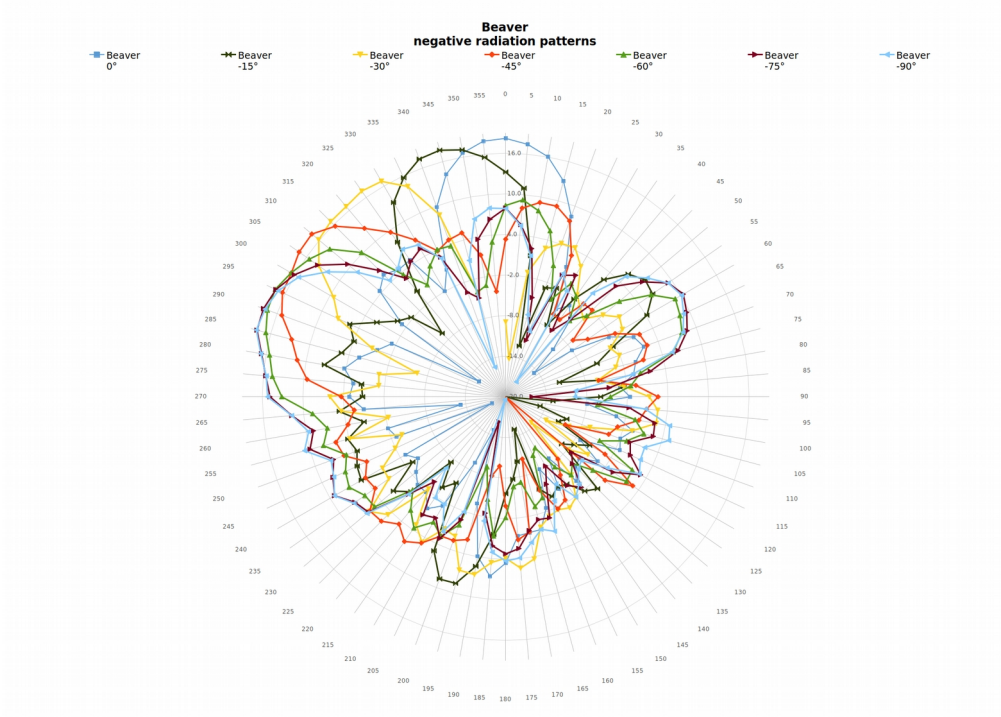
Kuvio 14: 75° suuntakuvio



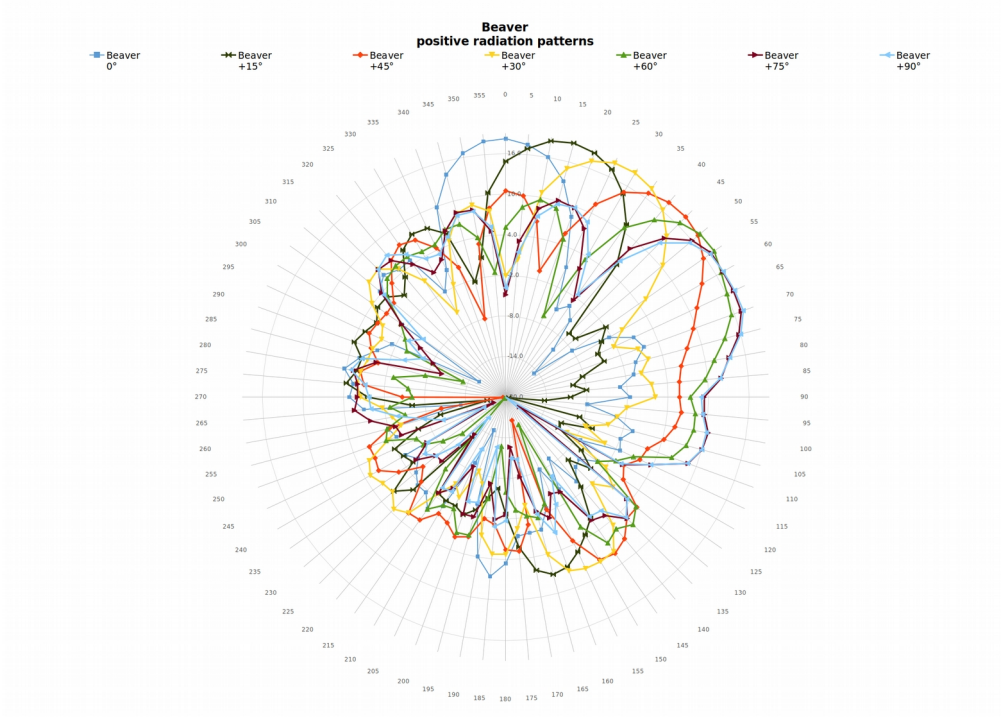
Kuvio 15: -90° suuntakuvio



Kuvio 16: 90° suuntakuvio



Kuvio 17: Negatiiviset säteilykulmat



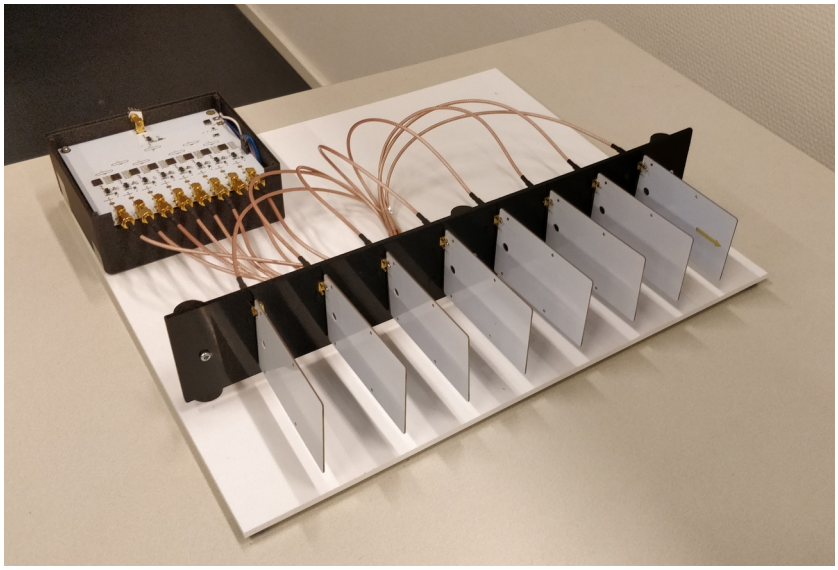
Kuvio 18: Positiiviset säteilykulmat

5 YHTEENVETO

Opinnäytetyön tavoitteena oli tarkastella vaiheohjatun antenniryhmän toimintaa, sekä toteutaa keilanmuodostusohjain käyttäen perinteistä keilanmuodostustekniikkaa vaiheohjatussa antenniryhmässä.

Opinnäytetyön alussa perehdyttiin vaiheohjatun antenniryhmän rakenteeseen sekä keilanmuodostuksen teoriaan. Ohjausyksikön toteutus-luvussa käytiin läpi vaativuusmäärittely, sekä vaihtoehtoja sen toteuttamiseksi. Ohjausyksikön testaus-luvussa mitattiin ohjausyksikön toimintanopeus, sekä varmistettiin sen toiminta antennijärjestelmän suuntakuviomittauksilla

Lopputuloksena saatiin toteutettua toimiva ohjausyksikkö, jolla voidaan demonstroida keilanmuodostusta. Koska ohjausyksikön asetuksia voidaan helposti muuttaa käyttöliittymän avulla, voidaan sitä jatkossa käyttää testityökaluna suunnitellessa elektronisesti ohjattavia suunta-antenneja.



Kuva 7: Valmis antennijärjestelmä

Luonteva jatkokehitys työlle olisi toteuttaa keilanmuodostus käyttäen digitaalista keilanmuodostustekniikkaa työssä käytetyn analogisen keilanmuodostustekniikan sijasta. Digitaalisen keilanmuodostuksen etuna on se, että sillä voidaan tuottaa ja ohjata useita samanaikaisia keiloja sekä dataväyliä käyttämällä esimerkiksi MIMO -tekniikkaa. (engl. Multiple Input and Multiple Output). [10]

LÄHTEET

1. Salminen, O. Satelliittijärjestelmien ja satelliittiantenniteknologian kehitys. Insinöörityö. Turun ammattikorkeakoulu, Turku. 2017. 67 s.

2. Phased Array Antenna. Saatavissa: http://www.radartutorial.eu/06.antennas/Phased%20Array%20Antenna.en.html. Hakupäivä 5.1.2019.

3. Alan, F. RES.LL-002: Adaptive Antennas and Phased Arrays. Luento esitetty: 2010, Massachusetts Institute of Technology.

4. Antenna theory – radiation pattern. Saatavissa: https://www.tutorialspoint.com/antenna_theory/antenna_theory_radiation_pattern.htm. Hakupäivä: 5.1.2019

5. Rohde & Schwarz, Millimeter-Wave Beamforming: Antenna Array Design Choices & Characterization (valkoinen paperi). Saatavissa: http://www.rohde-schwarz.com/appnote/1MA276.

6. Vivaldi antenna design analysis. Saatavissa: https://www.comsol.jp/blogs/vivaldi-antenna-design-analysis. Hakupäivä 5.1.2019

7. What is FreeRTOS? Saatavissa: https://www.freertos.org/about-RTOS.html. Hakupäivä 23.3.2019.

8. About Qt. Saatavissa: https://wiki.qt.io/About_Qt. Hakupäivä 25.5.2019

9. Okkonen, J. Uniform linear adaptive antenna array beamforming implementation with a wireless open-access research platform, Diplomityö. Oulun yliopisto, Oulu, 58 s

10. Myllyniemi, J. MIMO-Antennitekniikka. Insinöörityö, Tampereen ammattikorkeakoulu, Tampere, 2011, 26 s.