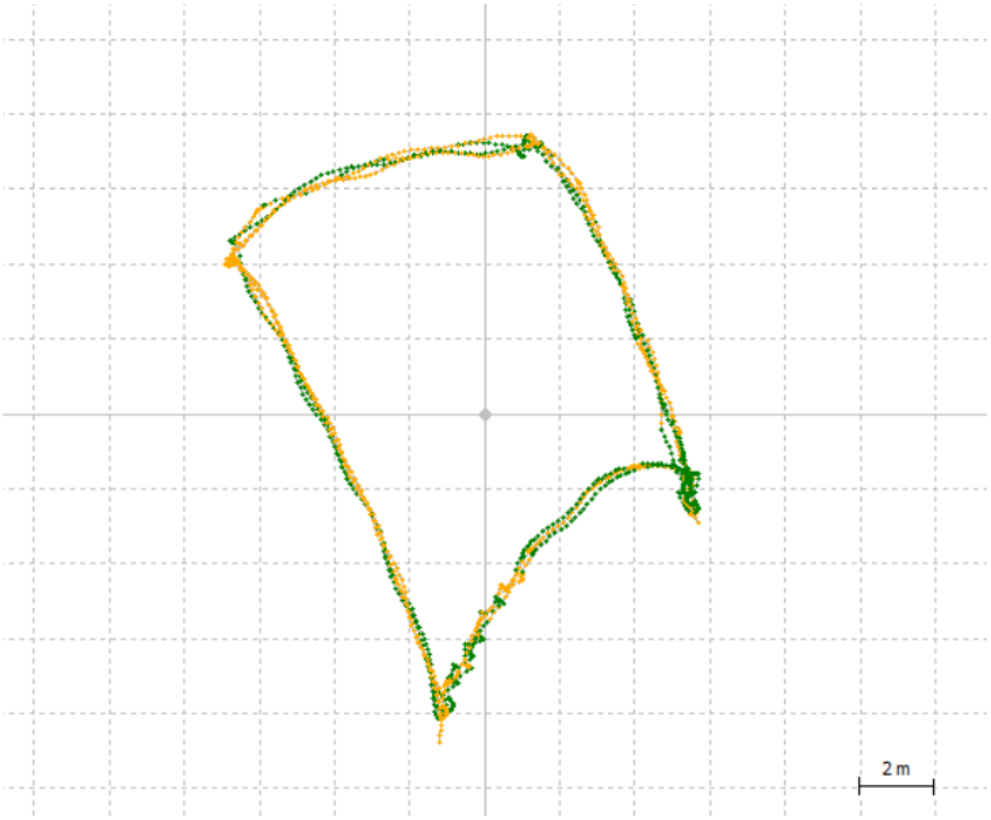
CHALMERS



Satellitnavigerad robotgräsklippare med RTK

Kandidatarbete inom produkt- och produktionsutveckling

Martin Baerveldt
Pontus Helmersson
Samuel Ingemarsson
Viktor Ohm
Nicholas Uusitalo
Arthur Wessman

Institutionen för produkt- och produktionsutveckling
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2016

Satellitnavigerad robotgräsklippare med RTK

Martin Baerveldt, Pontus Helmersson, Samuel Ingemarsson, Viktor Ohm, Nicholas Uusitalo, Arthur Wessman

Kandidatarbete inom produkt- och produktionsutveckling
Institutionen för produkt- och produktionsutveckling
Chalmers tekniska högskola
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

Omslag:
Bild från RTKLIB vid navigering av prototypen mellan koordinater

Sammandrag

Robotgräsklippare är en växande marknad, framförallt i Europa. Trots detta är produkterna på dagens marknad långt ifrån fullkomliga och det finns stort utrymme för förbättrande och nyskapande idéer. Något gemensamt för i princip alla robotgräsklippare idag är att de styrs slumpmässigt mellan så kallade begränsningskablar. Sammanfattningsvis dokumenterar denna rapport arbetet att utveckla en robotgräsklippare, som styrs med hjälp utav RTK-teknik. Denna idé skulle kunna lösa flera problem, inte minst vad gäller effektivisering och enkelhet vid användning.

Fokus för projektet var navigationen av roboten, därför lades inte fokus på att tillverka en fullt fungerande robotgräsklippare. Resultatet blev en prototyp, liknande en robotgräsklippare, vilken med hjälp av en basstation använde sig av RTK för att navigera. Den framtagna prototypen kunde köra en förprogrammerad rutt, undvika hinder och hålla sig inom en programmerad avgränsning. Under god sikt mot himlen hade positioneringen en precision på under 4 cm vid rörelse och 4,5 cm när den var stillastående. Då sikten mot himlen var skymd sjönk dock precisionen avsevärt vilket begränsar eventuella användningsområden. Detta innebär att navigationssystemet lämpar sig för användning om god sikt kan garanteras.

Nyckelord: RTK, RTKLIB, GNSS, Robotgräsklippare, Satellitnavigering

Abstract

The market for robotic mowers is growing, especially in Europe. Despite this, the products on the market are far from complete and there is a lot of room for improvements and innovative ideas. Something that virtually all robotic mowers have in common today is that they are controlled randomly between so-called boundary wires. In summary, this report documents the work to develop a robotic mower, which is navigated using RTK-technique. This is an idea that would solve several problems, not least in terms of efficiency and ease of use.

Since the focus of the project was the navigation of the robot, the focus was not to manufacture a fully functional robotic mower. Instead, the result was a prototype, much like a robotic lawn mower, that with the help of a base station used RTK to navigate. The developed prototype was able to drive around a programmed path, avoid and navigate around obstacles and stay within a programmed boundary. With clear skyview the positioning had a precision of less than 4 cm during movement and less than 4.5 cm when it was stationary. With obstructed skyview the precision was a lot worse which limits potential applications. Therefore the navigation system is suitable in applications where a clear skyview is guaranteed.

Keywords: RTK, RTKLIB, GNSS, Robotic Lawn Mower, Satellite Navigation

Förord

Denna rapport behandlar ett kandidatarbete utfört våren 2016 på Chalmers tekniska högskola. Kandidatarbetet avslutar den treåriga grundutbildning efter vilken studenten erhåller en kandidatexamen inom sitt studieområde. Tre av de medverkande studenterna läser det femåriga civilingenjörsprogrammet Automation och mekatronik, två läser Maskinteknik och en läser Teknisk fysik. Dessa tre studieinriktningar är alla på 300 högskolepoäng, varav kandidatarbetet utgör 15 av dessa.

Projektgruppen vill rikta ett tack till alla som hjälpt till under projektarbetets gång. Särskilt tack riktas till handledare Per Nyqvist, Göran Stigler samt till de anställda i prototyplaboratoriet.

Ordlista

Galileo Satellitnavigationssystem skapat av Europeiska unionen och Europeiska rymdorganisationen. Är för närvarande under utveckling. 2, 12, 35

GLONASS Globalnaja navigatsionnaja sputnikovaja sistema, Ryskt satellitnavigationssystem. 12, 15, 30, 33–35, 37

GPS Global Positioning System, satellitnavigationssystem utvecklat av det amerikanska försvarsdepartementet. 5, 6, 9, 12, 15, 30

Integer ambiguity Antal hela bärvågscykler mellan satellit och mottagare. En central variabel för att beräkna avstånd mellan satellit och mottagare med hjälp av satellitsignalens bärvåg. 8, 10, 12, 13, 16, 33–35

PPP Precise Point Positioning. Metod som genom användning av korrektionsdata kan beräkna väldigt noggranna positioner med en GNSS-mottagare. 17

Pseudoslumpmässig Skenbart slumpmässig. Syftar ofta på datorgenererad slumpmässighet. 7

RTCM3 En standard som används för överföring av differentiella- och RTK-korrektioner. 13, 17

RTK Real Time Kinematic. En noggrann positionsmätning med minst två samverkande antennmottagare och utnyttjandet av satelliternas bärvågssignaler. 1, 2, 4, 5, 10, 12, 15, 17, 23, 30, 32, 37

TCP Transmission Control Protocol. Dataöverföringsprotokoll vilket används vid kommunikation via internet. 17, 18, 21

VNC Virtual Network Computing. Programvara som kan fjärrstyra en annan dator genom att överföra mus- och tangentbordskommandon från en klient till en server. Servern skickar en skärmbild tillbaka till klienten. 18

Förkortningar

CUI Command-line User Interface. 12

DGPS Differential Global Positioning System. 2, 10

GDOP Geometrical Dilution Of Precision. 9

GNSS Global Navigation Satellite System. 2, 6, 12, 13, 15–17, 21, 24, 35, 36

GPIO General-Purpose Input/Output. 15, 20, 21, 36

GUI Graphical User Interface. 12–14, 18

LORAN Long Range Navigation. 6

RINEX Reciever Independent Exchange Format. 13

SBAS Satellite Based Augmentation System. 13–15, 17, 35

Innehåll

1 Inledning

I detta kapitel presenteras bakgrund och syfte till uppgiften projektet behandlar. Även problemet och dess avgränsningar presenteras.

1.1 Bakgrund

Automatiserade gräsklippare är en växande trend [1]. Marknaden för dessa ökar med ungefär 30-40 procent per år. Sverige, med flest robotgräsklippare per person, står för cirka 12 procent av den globala marknaden [2]. Dock har företag såsom *Husqvarna* visat stort intresse för att ytterligare globalisera marknaden [3].

Dagens befintliga robotgräsklippare använder i stor utsträckning en teknik som bygger på att de slumpmässigt klipper gräsmattan genom att köra framåt tills de når en avgränsning. Denna avgränsning definieras av en avgränsningskabel vilken läggs ut runt området som ska klippas. Denna kabel ger klipparen information om när den nått kanten av gräsmattan. Kabeln innebär en extra kostnad och ytterligare arbete vid installation. Laddstationen måste också, med denna teknik för navigering, ha extra utrustning för att leda in gräsklipparen på rätt sätt när den ska laddas.[4] När roboten når avgränsningen backar den tillbaka lite, justerar riktningen och fortsätter framåt tills den når avgränsningen igen. Samma princip används om den skulle köra in i ett hinder. Den här tekniken fungerar relativt bra vid mindre klippareor men betydligt sämre vid större. När gräsmattorna blir större får gräsklipparna problem med att klippa hela gräsmattan och de lämnar lätt bitar oklippta. Detta kan delvis vara en bidragande faktor till varför robotgräsklippare inte blivit lika populära i USA [5, 6] där gräsmattorna ofta är större än i Sverige [7].

För att lösa dessa problem kan en annan teknik användas för robotgräsklipparens navigering. En god kandidat till navigationssystem skulle kunna vara satellitnavigering. Dock har det i sitt standardutförande en noggrannhet på några meter. För att förbättra denna noggrannhet kan RTK-teknik användas. Då används bland annat en referensstation utrustad med ytterligare en satellitmottagare för att eliminera felkällor [8]. Denna teknik har dock tidigare varit oförsvarbart dyr för att användas som navigationssystem till robotgräsklippare. Till följd av att priserna för antenner och mottagare gått ned kan nu en centimeternoggrannhet uppnås, under gynnsamma förhållanden, till en kostnad under 5 000 SEK [9]. Detta innebär att RTK nu skulle kunna vara ett alternativ till navigationssystem för robotgräsklippare. Något vilket tidigare inte heller varit möjligt på grund av den befintliga tekniken är att bestämma exakt hur robotgräsklipparen skall köra. Detta skulle kunna realiseras med hjälp av ett precist navigationssystem.

En billig centimeternoggrann navigationsmetod är inte endast intressant för robotgräsklipparbranschen, utan har många fler tillämpningar. Till exempel som hjälpmedel för synskadade eller en del av styrningen i autonoma bilar. Allt detta tillsammans gör detta till ett mycket intressant ämne att undersöka och vidareutveckla.

1.1.1 Tidigare forskning

För att få en god bild av både navigationssystem och robotgräsklippare i allmänhet är det bra att få en inblick i liknande publiceringar inom dessa ämnen. Vid efterforskning har det hittats ett flertal bra rapporter vilka bland annat tagit upp RTK-system i robotar

och andra navigationssystem i gräsklippartillämpningar.

Ett projekt har gjorts för att utvärdera användbarheten av GNSS-styrning i robotgräsklippare. Här skapades det en mjukvaruarkitektur för att simulera en godtycklig miljö för att utvärdera en satellitstyrning antaget att Galileo skulle vara fullt operativt. De simuleringar som gjordes tydde starkt på att algoritmen fungerade väldigt bra vid fri sikt och stod sig bra mot till exempel traditionell DGPS-navigering. År 2010 gjordes tester på en golfbana vilka bekräftade resultaten ytterligare. [10]

Det har även tagits fram en robotgräsklippare vilken navigeras med hjälp av LPS. Systemet använder sig av en sensor monterad på robotens övre del och fyra aktiva fyrar vilka täcker ett målområde. För att bestämma robotens orientering har sensortornet cirkulära mottagare vilka känner av infraröda signaler som emitteras av varje fyr. Resultaten tyder på att denna styrning kan jämföras med en manuell gräsklippare förd av en professionell och erfaren förare. [11]

Ett multisensor-system för att styra en robotgräsklippare har tagits fram i Kina. Detta fungerar genom att sensorer kartlägger ett annars okänt område. Systemet består av och integrerar ultraljudssensorer, infraröda sensorer, kollisionssensorer, kodomvandlare, temperatursensorer och en elektrisk kompass. Flertalet simuleringar indikerar på att detta system ska vara väldigt noggrant och effektivt. [12]

Även ett projekt där både DGPS och RTK har använts i en robotgräsklippare har utförts. Detta arbete använde sig av RTK-program vilket via operativsystemet Linux implementerats i gräsklipparen. Det beskrivs kortfattat att dockning i laddstation lyckades, vilket innebär att en noggrannhet på fem centimeter uppnåtts. Slutsatsen drogs att prototypen fungerade väl och att detta positioneringssystem kan implementeras i flera olika användningsområden. [13]

1.2 Syfte

Syftet med rapporten är att undersöka huruvida en robotgräsklippare, navigerad med RTK-teknik, löser några av de problem en befintlig robotgräsklippare idag kan föra med sig. Alltså ska den praktiska användbarheten av RTK-satellitnavigering stå i centrum för resultatet.

Det huvudsakliga målet med projektet är att få ut en tillräckligt bra position till gräsklipparen för att denna ska kunna användas till navigationssystemet. De sekundära målen i projektet är själva körningen av gräsklipparen samt ruttplaneringen. Målen kring körningen och ruttplaneringen är att prototypen ska kunna uppvisa områdesbegränsning och hitta tillbaka till en teoretisk laddstation. Hinderdetektion faller även under det sistnämnda målet.

1.3 Problem

Det faktum att dagens robotgräsklippare inte vet var de befinner sig skapar en rad problem. Dels är det slumpmässiga mönstret väldigt ineffektivt på större ytor. Att roboten inte vet sin position och riktning gör också att tillägg krävs för att den ska hitta till laddstationen. Den slumpmässiga rutten leder även till en väldigt ineffektiv strömåtgång. Begränsning av området vilket gräsklipparna verkar på sker med särskilda kablar i marken. Dessa medför ytterligare en kostnad och dessutom kan begränsningskablar av olika märke från angränsade tomter störa ut varandra [14]. Allt detta går att lösa med en satellitnavigerad robot, men problemet har varit att satellitnavigering hittills inte varit exakt nog till ett konkurrenskraftigt pris [9].

Med noggrann positionering kan de slumpade rörelsemönstren ersättas med smartare rutter. För att inte lämna något oklippt, speciellt i kanterna, behövs en precision i storleksordningen centimeter. Kommunikationen mellan satellit och mottagarna samt mellan basstationen och roboten måste fungera i den miljön roboten kan önskas användas i, utan att nämnvärt tappa precision. Roboten behöver också kunna undvika hinder med hjälp av sensorer. Allt detta medan den fortfarande håller en prisnivå vilken gör den attraktiv på marknaden.

1.4 Avgränsningar

På grund av den begränsade tiden projektet har till förfogande kommer fokus i rapporten inte ligga på framtagning av gräsklipparen i sig, utan arbetet kommer ägnas till att ta fram och utvärdera navigationssystemet som ska styra roboten. Gräsklippningsfunktionen kommer tas upp mer ytligt. Gräsklipparen kommer motsvaras av en produkt som kan röra sig och uppvisa egenskaper liknande en traditionell robotgräsklippare, utan någon större fokus på design av chassit och dess uppbyggnad. Roboten kommer att bestå av ett enkelt chassi med plats för två enkortsdatorer, två motorer, tre hjul (ett fram och två drivhjul bak), en antenn, en strömkälla, ett styrkort för motorerna samt nödvändig elektronik. Den färdigställda roboten är även tänkt att fungera i en godtycklig miljö likt en gräsmatta. Inga orimliga hinder eller terränger kommer beaktas.

För att tekniken ska vara intressant som navigationssystem till robotgräsklippare måste kostnaden vara relativt låg. Därmed sätts en budget på approximativt 5000 SEK för navigationssystemet.

2 Teori

För att få en djupare förståelse i hur de olika komponenterna och delarna i projektarbetet fungerar och kommit till, är det viktigt att först samla fakta och förklarande information om dessa. Detta kapitel behandlar därför funktioner bakom dagens gräsklippare samt teorin bakom navigering och konfigurering av bland annat RTK och motorer.

2.1 Dagens robotgräsklippare

På den svenska marknaden fanns det år 2013 13 olika etablerade modeller av robotgräsklippare [1]. Idag finns det otaliga modeller från flera olika tillverkare. *Husqvarna*, *Honda*, *Viking*, *Bosch*, *Gardena* och *Lizard* är bara några av de många företag vilka producerat och sålt robotgräsklippare de senaste åren. Modellerna på marknaden har väldigt olika specifikationer, bland annat vad gäller klippyta och batteritid. I tabell 1 presenteras några modeller som finns på marknaden idag samt deras allmänna specifikationer.

Tabell 1: Exempel på några befintliga robotgräsklippare och deras specifikationer [15, 16, 17, 18, 19, 20]

Modell	Klippyta [m2]	Klipphöjd [mm]	Batteritid/laddning [min]	Vikt, kg	Pris [SEK]
Husqvarna Automower 220AC	1800	20-60	45	9	20 800
Gardena R50Li	500	20-60	60	11,1	8 500
Ambrogio Line 220 Basic	1900	20-70	180	12	19 700
Husqvarna Automower 105	600	20-50	70	06.07	11 900
Honda Miimo 300	2200	20-60	45	11,5	23 000
Biltema LMR 24	2500	30-70	180	22	8000
Worx Landroid L	1500	20-60	90	14,5	13 500

2.1.1 Uppbyggnad och styrning

En vanligt förekommande robotgräsklippare har idag en relativt enkel uppbyggnad. Den består först och främst av en elektrisk motor som driver roboten. Ytterligare en motor krävs för att driva gräsklippningsbladen vilka är horisontellt monterade på undersidan. En stötsensor är monterad för hinderhantering. Ytterligare en sensor finns för att roboten ska ha kontakt med de begränsningskablar som ramar in var roboten kan köra. Ett kretskort för klippningsprogrammen och ett batteri för strömförsörjning finns även i roboten. Den stationära laddstationen består av strömförsörjning samt teknik för kommunikation med gräsklipparen.

I princip alla befintliga robotgräsklippare på marknaden idag navigerar med hjälp av så kallad “studsteknik”. Detta innebär att de kan klippa gräs inom ett anvisat område genom att detta specifika område är begränsat av en kabel vilken sänder ut svaga signaler.

Enkelt kan det sägas att detta för roboten innebär ett osynligt "staket" vilket inte får passeras. Vissa robotar är även utrustade med GPS-mottagare, vilket innebär att de kan uppskatta var de redan har varit och klippt. Denna GPS är dock inte tillräckligt exakt för att styra hela processen, utan minskar bara klipptiden till viss del. [21, 22]

2.1.2 Installation, användning och kostnad

Under installationen av robotgräsklippare läggs största vikt på att lägga ut begränsningsslingan. Detta bör planeras noggrant för att minimera övergångslinjer i gräset och för att få så jämn och bra klippning som möjligt. När robotgräsklipparen sedan är igång och batteriet behöver laddas kör den självmant till en laddstation. Denna borde vara belägen på en lättåtkomlig och skuggig plats. Detta för att effektivisera tiden det tar att hitta till laddstationen och även för att behålla batteriet i bästa skick. När det gäller säkerhet använder sig majoriteten av robotarna idag av sensorer. Detta gör bland annat att de stängs av om de lyfts upp eller stannar ifall de når ett hinder. [21, 22]

Marknadspriserna på robotgräsklippare skiljer sig en hel del. Dessa kan variera från cirka 6 000 SEK till uppemot 50 000 SEK. Det som skiljer dessa åt är att de dyrare modellerna har större kapacitet, mer avancerad teknik och fler finesser. Vissa har bland annat möjlighet att programmera vilka dagar och tider gräsklipparen ska arbeta på samt förmågan att kunna upptäcka hinder på avstånd.

En robotgräsklippare har en inköpskostnad klart högre än traditionella gräsklippare. I en livscykelkostnadsanalys med en cylindergräsklippare, rotorgräsklippare, åkgräsklippare och en robotgräsklippare anses dock robotgräsklipparen billigast när både material- och arbetskostnader är inräknade [1]. Detta är ett bra stöd för att marknaden med all förmodan kommer att fortsätta växa och utvecklas ännu mer.

2.2 Referenslösning

För att enkelt kunna dra slutsatser och jämföra huruvida den framställda produkten är konkurrenskraftig på den rådande marknaden är det bra att ha en referensprodukt att jämföra med. Med över 20 års erfarenhet av robotgräsklippare är *Husqvarna AB* en av de ledande aktörerna i världen när det kommer till robotgräsklippare. *Automower*, vilket är namnet på deras samling av robotgräsklippare, har idag sex olika modeller. Dessa varierar i pris från cirka 12 000 SEK till 38 000 SEK. Eftersom en robotgräsklippare styrd med RTK kommer kosta en del, har den dyraste i samlingen, *Automower 450X*, valts som referens, se figur 1.

Figur 1: Referensprodukten Automower 450X.[23]

Husqvarnas förstklassiga och mest bekostade robotgräsklippare klarar av svåra terränger, komplexa trädgårdar och stora ytor. Den klarar även av lutningar upp till 45 grader och olika typer av hinder.

GPS-stödd navigering
Modellen är utrustad med en GPS-mottagare. Denna GPS fungerar så att den utnyttjar en karta över trädgården för att sedan notera var den redan har varit, för att sedan reglera klippmönstret efter detta. Detta effektiviserar klipptiden samtidigt som det ger bättre resultat vad gäller klippningen.

Klippaggregat
Klippsystemet består av en roterande cirkulär skiva med rörliga knivblad. Dessa knivar är tillverkade i kolstål och systemet har en låg ljudnivå, vilket uppskattas i tätbebyggda områden.

Mobilapplikation
Via *Automower Connect* går det att enkelt, via en smartphone, kontrollera sin gräsklippare. Det går bland annat att starta, parkera och stoppa gräsklipparen via applikationen. Det går även att ändra inställningar och spåra gräsklipparen via GPS.

Övriga egenskaper
Utöver det som redan diskuterats har modellen ytterligare funktioner som till exempel kodlås, timer, tål alla väder, hinderdetektion på avstånd samt elektrisk höjdjustering.

[23]

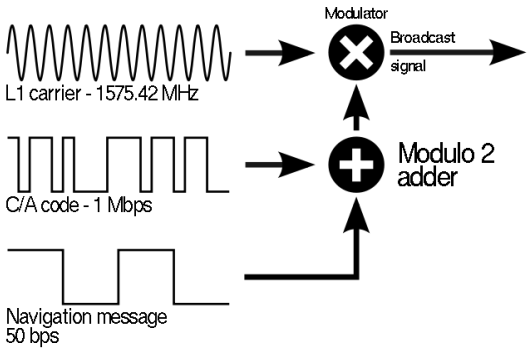
2.3 Navigering och programvara

Många olika navigationssystem har utvecklats under historien. De tidiga metoderna såsom navigering efter stjärnorna var manuella och tidskrävande att använda sig av. De första automatiska navigationssystemen kom när radio började användas. Det gick då att använda sig av radiosignaler för att bestämma sitt avstånd till en viss station och sedan använda sig av flera sådana stationer för att bestämma sin position. Ett sådant system var i bruk i USA och kallades *LORAN*. Ett problem med detta var att en radiosändare bara har en räckvidd på cirka 500 km och därför inte är heltäckande på jorden utan begränsade till var stationerna befinner sig. Därför används idag nästan uteslutande olika satellitnavigationsystem, ett samlingsnamn för dessa är GNSS. Dessa är heltäckande på jorden och kan beräkna en position utan att något manuellt arbete behöver göras, vilket ger möjlighet att få positioner i realtid. De har även mycket bättre noggrannhet än de tidigare navigationssystemen. [24]

2.3.1 GNSS

Ett GNSS-system består av ett nätverk av satelliter vilka befinner sig i en omloppsbana kring jorden. Dessa satelliter skickar data till mottagare vilket gör att mottagarna utifrån det kan bestämma sin position genom att beräkna avståndet till satelliten. Detta görs genom att beräkna tiden det tog för mottagaren att få datan från satelliten. Satellitsignalen består i huvudsak av tre komponenter. En bärvåg, “carrier”-komponent, en kodkomponent och en komponent för navigationsmeddelande. Dessa kan ses representerade i figur

2. [25]

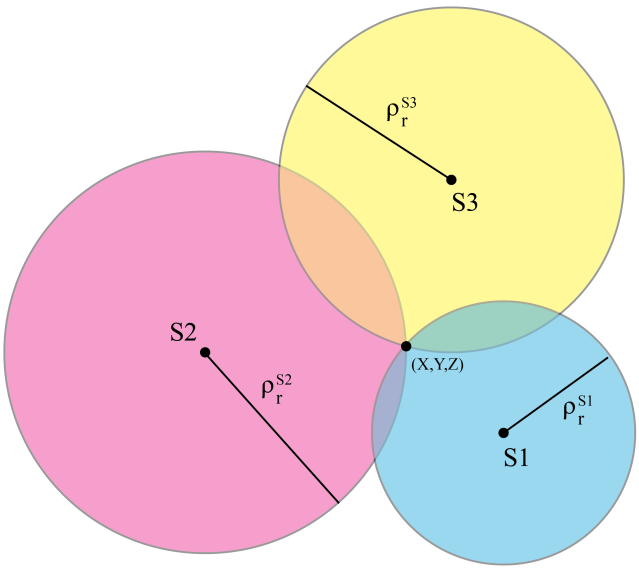


Figur 2: GNSS signalmoduleringsschema [26].

Kodkomponenten är det som används i de flesta tillämpningarna för att bestämma restiden av signalen. Det är en pseudoslumpmässig kod vilken genereras identiskt av både satelliten och mottagaren. Då det tar tid för signalen att nå mottagaren kommer dessa två koder vara ur fas. Mottagaren beräknar då hur mycket den måste ändra sin kod för att komma i fas med den mottagna satellitens kod. Detta är tiden det tar för signalen att nå mottagaren. Sedan beräknas avståndet mellan satellit s och mottagare r ut enligt:

$$\rho_r^s = c(t_r - t_s) \tag{1}$$

Där ρ_r^s är det så kallade geometriska avståndet vilket definieras som avståndet mellan en satellit och mottagares antenn utan att ta hänsyn till några felkällor, c är ljusets hastighet, t_r är tiden som ges av mottagaren då signalen tas emot och t_s är tiden då satelliten skickade signalen. För att få en absolut positionering krävs tre satelliter. Den absoluta postionen är området där alla tre sfärer, skapade av det geometriska avståndet, skär varandra, se figur 3. Detta kallas trilateration [27].



Figur 3: Trilateration med tre satelliter.

Att bara använda koden från signalen räcker dock inte för att uppnå tillräcklig precision. Kodsignalen har en frekvens i storleken MHz, vilket begränsar noggrannheten. Då koden är pseudoslumpmässig och distinkt kan förbättringar göras så att meterprecision erhålls. Detta är den bästa precisionen som kan återfås med endast kodkomponenten. [25, 28]

En annan komponent vilken är viktig för att beräkna positionen är navigationsdata, även kallad nav-data. Den innehåller information om var satelliten befinner sig utifrån de referensramar vilka används på jorden såsom latitud och longitud. Den innehåller också information om satellitens omloppsbana.

För att uppnå bättre precision kan bärvågen användas. Det är den som utgör huvuddelen av signalen och koden är bara en liten modulering av den. Bärvågen har en frekvens på GHz-nivå och om den kan detekteras kan en noggrannhet på en millimeter återfås. Problemet med denna metod är att bärvågen svänger så snabbt att det är svårt att veta hur många cykler mottagaren ligger bakom. Antalet cykler mottagaren ligger bakom kallas *unknown integer* eller *integer ambiguity*. För att bestämma avståndet mellan satellit och mottagare används fasen på vågen. Denna ges av:

$$\phi_r^s = \phi_r(t_r) - \phi^s(t_s) + N_r^s \tag{2}$$

Där ϕ_r^s är fasskillnad mellan satellit och mottagare, $\phi_r(t_r)$ och $\phi^s(t_s)$ är den observerade fasen vid mottagare respektive satellit och N_r^s är *integer ambiguity*. Fasen kan skrivas som $\phi = f(t - t_0) + \phi_0$ där f är frekvensen och ϕ_0 är fasen vid initialtiden t_0 vilket ger:

$$\phi_r^s = (f(t_r - t_0) + \phi_{0,r}) - (f(t_s - t_0) + \phi_0^s) + N_r^s \tag{3}$$

genom att skriva frekvensen $f = \frac{c}{\lambda}$ där λ är våglängden samt förkorta bort t_0 fås:

$$\phi_r^s = \frac{c}{\lambda}(t_r - t_s) + (\phi_{0,r} - \phi_0^s) + N_r^s \tag{4}$$

För att få ut ett avstånd multipliceras ϕ_r^s med λ och då fås det slutliga fasavståndet Φ_r^s som ges av:

$$\Phi_r^s = \rho_r^s + \lambda(\phi_{0,r} - \phi_0^s) + \lambda N_r^s \tag{5}$$

Detta ger som sagt mycket bättre precision än om bara kodkomponenten används. Det krävs dock beräkningar för att bestämma N_r^s vilket inte behövs för koden. Notera även att alla mottagare inte kan mäta bärvågscykler utan det är bara en funktion som de dyra mottagarna har. Det finns också andra felkällor, vilka medför att det inte går att uppnå millimeterprecision med bara en mottagare. [25, 28, 29]

Felkällor
Klockan i en mottagare måste vara exakt för att kunna räkna cykler ordentligt. En nanosekunds fel kan resultera i uppemot 30 centimeters fel i mätningen. Satelliterna har egna atomklockor vilka synkroniseras med markstationer för att motverka detta men mottagare har inte det. Däremot kan mottagarens klockfel beräknas genom att introducera det som en ytterligare variabel och denna kan lösas genom att använda ytterligare en satellit. För att beräkna en noggrann position behövs således fyra satelliter, tre för positionen (x,y,z) och en för klockfelet.

Nav-datan har även lägre frekvens och kan ofta behöva väntas på. Vidare har informationen som skickas avseende omloppsbanan ett fel på några meter. Detta fel kan dock elimineras med hjälp av differentiell teknik vilket kommer beskrivas senare i rapporten.

En annan felkälla kommer ifrån atmosfären som bromsar signalen så den inte riktigt färdas i ljusets hastighet. Dessa fel kan hanteras genom att applicera matematiska modeller för att beräkna fördröjningen atmosfären introducerar vilket kan minska felpåverkan. Dock introduceras mycket av felen av jonosfären vilken är svår att modellera. Därför skickar satelliterna ut två signaler, L1 och L2. Jonosfären påverkar dessa två signaler på olika sätt då de har olika frekvens vilket gör det möjligt att kompensera för felen. Mottagare och antenner som kan ta emot båda frekvenserna är dock fortfarande väldigt dyra.

Ytterligare en felkälla är att så kallade flervägsfel uppstår när signalen reflekteras och inte tar den direkta vägen till antennen. Detta både stör den direkta signalen och introducerar en osäkerhet när signalen faktiskt kommer fram till antennen. Mätningar i en stadsmiljö och under trädkronor kan orsaka dessa problem då signalen kan reflekteras många gånger. Detta fel kan minskas genom att jorda antennen och se till att den har en klar vy av himlen. Det beror även mycket på antennens kvalitet och dess kapacitet att avvisa reflekterade signaler.

Dessa fel kan modelleras genom olika variabler, så för ekvation (1) fås en ny ekvation:

$$P_r^s = \rho_r^s + I_r^s + T_r^s + c(\delta t_r - \delta t_s) + \epsilon_P \tag{6}$$

P_r^s är det så kallade pseudoavståndet mellan satellit och mottagare som räknar in felkällor, I_r^s är felet som introduceras av jonosfären, T_r^s är felet som introduceras av troposfären, δt är klockfel för satellit och mottagare och ϵ_P är mätfel som flervägsfel och fel i mottagaren. På samma sätt fås för ekvation (5):

$$\Phi_r^s = \rho_r^s + c(\delta t_r - \delta t_s) + \lambda(\phi_{0,r} - \phi_0^s) + \lambda N_r^s - I_r^s + T_r^s + d\Phi_r^s + \epsilon_\phi \tag{7}$$

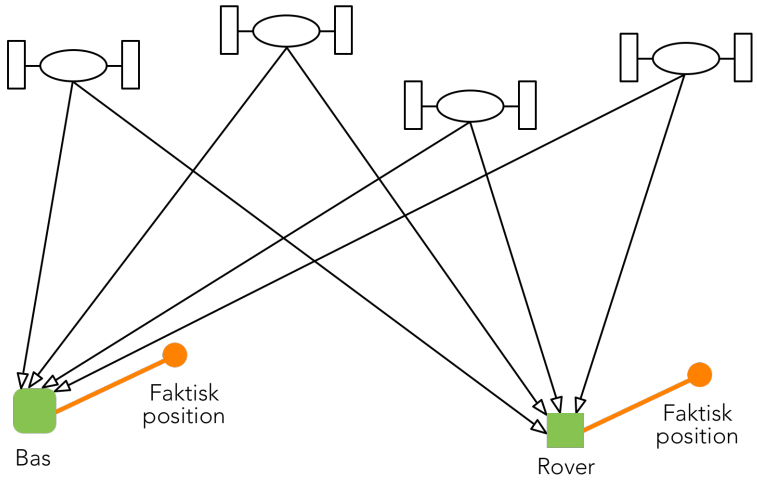
där $d\Phi_r^s$ är specifik felmätning av fas vilket främst orsakas av antenner. Teckenbytet för jonosfärtermen kommer av att den påverkar koden och bärvågen på olika sätt, koden fördröjs men fasen avanceras.

Det som behandlats har mest handlat om fel mellan en mottagare och en satellit. Dessa fel fortplantas genom att positionen beräknas av flera satelliter. Hur de fortplantas beror på hur satelliterna är utspridda över himlen. Är de nära varandra så kan felet vara tiotals meter medan om de däremot är utspridda över himlen kan felet nästan vara lika litet som felet de individuella satelliterna ger. Måttet på hur stort detta fel är kallas GDOP, och ju mindre detta värde är desto bättre. Det bästa sättet att minska GDOP är att öka antalet satelliter data inhämtas från. [29, 30]

2.3.2 DGPS

Differentiell GPS använder två mottagare, en mobil modul hädanefter kallad rover och en bas. Basen placeras statiskt på en plats och sedan noteras basens exakta position. Detta kan göras genom att utföra en positionsmätning under en längre tid och sedan ta medelvärdet av denna. Då många av felkällorna påverkar båda mottagarna lika mycket kan roverns exakta position beräknas genom att beräkna skillnaden mellan basens faktiska

position och där basens mottagare beräknar att den befinner sig, detta kan ses i figur 4. Denna skillnad skickas sedan till rovern så att den kan beräkna sin faktiska position.



Figur 4: Basstation och rover med gemensamma satelliter.

I verkligheten skickar basen sådan korrektionsdata för varje satellit. Detta betyder att för att få en bra position måste basen och rovern få data från samma satelliter. Notera här att det är väldigt viktigt att positionen på basen är exakt för att få en exakt position på rovern. Om den position som angetts för basen är fel i någon riktning får rovern samma fel i sin position. Avståndet mellan basen och rovern spelar även roll för hur noggrann positionen blir. Då avståndet mellan dem ökar blir felen mellan basens och roverns position mer och mer olika. En tumregel är att för varje kilometer mellan rover och bas ökar felet för positionen med två millimeter. [31]

I Norden används DGPS vanligast i marina förhållanden. I Sverige använder bland annat flottan tekniken för till exempel minröjning. I USA används den till flera områden, bland annat i motorvägs-, järnvägs- och marinnavigation. [32]

2.3.3 RTK

RTK eller Real-time Kinematic kallas tekniken där bärvågen och den differentiella tekniken används för att få ökad noggrannhet. Då bärvågen används introduceras felet som kallas *unknown integer* eller *integer ambiguity*. *Integer ambiguity* måste lösas innan det går att få en noggrann position. Denna beräknas med hjälp av programvara. När den väl lösts fås en noggrann position ända tills kontakten med satelliten försvinner. Tappas kontakten måste *integer ambiguity* lösas igen, vilket kan ta flera minuter. Om det finns minst fem gemensamma satelliter mellan bas och rover kan *integer ambiguity* lösas för en ny satellit omedelbart, finns det färre tar det längre tid.

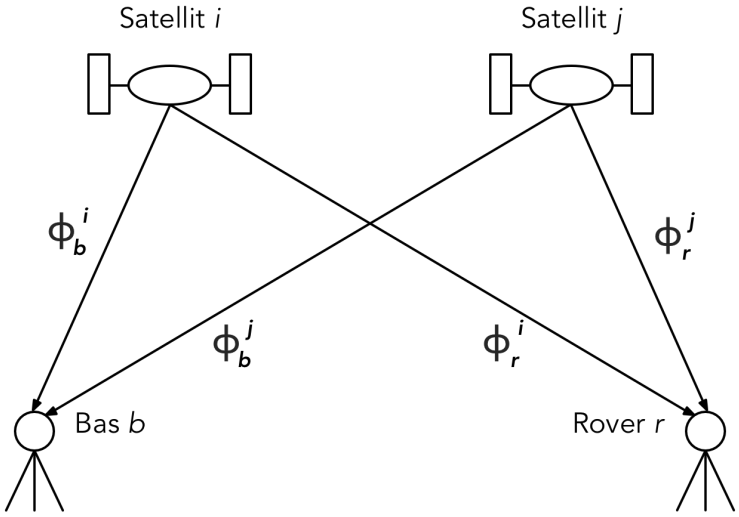
Den differentiella tekniken eliminerar felen i RTK enligt två steg. Anta att basen och rovern observerar samma satellit. Differensen av vad dessa två mäter upp kallas för singeldifferens. Om en bas b och en rover r används, vilka ser en satellit i, fås att differensen

mellan P_r^i och P_b^i blir enligt ekvation (6):

$$\begin{aligned}
&P_{br}^i = P_r^i - P_b^i = \rho_r^i + I_r^i + T_r^i + c(\delta t_r - \delta t_i) + \epsilon_{P,r} - (\rho_b^i + I_b^i + T_b^i + c(\delta t_b - \delta t_i) + \epsilon_{P,b}) = \\
&(\rho_r^i - \rho_b^i) + I_r^i - I_b^i + T_r^i - T_b^i + c(\delta t_r - \delta t_i + \delta t_i - \delta t_b) + \epsilon_{P,r} - \epsilon_{P,b} = \\
&\rho_{br}^i + I_{br}^i + T_{br}^i + c\delta t_{br} + \epsilon_{P,br}
\end{aligned} \tag{8}$$

Differensen mellan Φ_r^i och Φ_b^i blir på samma sätt enligt ekvation (7):

$$\begin{aligned}
&\Phi_{br}^i = \Phi_r^i - \Phi_b^i = \\
&(\rho_r^i - \rho_b^i) - I_r^i + I_b^i + T_r^i - T_b^i + c(\delta t_r - \delta t_i + \delta t_i - \delta t_b) + \epsilon_{\Phi,r} - \epsilon_{\Phi,b} + \\
&\lambda(\phi_{0,r} - \phi_0^i - \phi_{0,b} + \phi_0^i) + \lambda(N_r^i - N^b) + d\Phi_r^i - d\Phi_b^i = \\
&\rho_{br}^i + I_{br}^i + T_{br}^i + c\delta t_{br} + \epsilon_{\Phi,br} + \lambda N_{br}^i + \lambda\phi_{0,br} + d\Phi_{br}^i
\end{aligned} \tag{9}$$



Figur 5: Dubbeldifferens. De variabler som ses i figuren är samma som i ekvation (11).

Nu har satellitens klockfel δt_i samt den initiala fasen ϕ_0^i eliminerats. För att få bort fler fel används dubbeldifferens då basen och rovern ser två satelliter i och j och tar differensen mellan singeldifferensen för två satelliter. En bild på hur detta ser ut kan ses i figur 5. Alltså blir differensen mellan P_{br}^i och P_{br}^j:

$$P_{br}^{ji} = P_{br}^i - P_{br}^j = \rho_{br}^{ji} + I_{br}^{ji} + T_{br}^{ji} + c\delta t_{br}^{ji} + \epsilon_{P,br}^{ji} \tag{10}$$

och differensen mellan Φ_{br}^i och Φ_{br}^j blir på samma sätt:

$$\Phi_{br}^{ji} = \Phi_{br}^i - \Phi_{br}^j = \rho_{br}^{ji} - I_{br}^{ji} + T_{br}^{ji} + c\delta t_{br}^{ji} + \lambda N_{br}^{ji} + \lambda\phi_{0,br}^{ji} + d\Phi_{br}^{ji} + \epsilon_{P,br}^{ji} \tag{11}$$

Vissa av dessa termer kan dock elimineras. Då klockfelet δt är konstant och oberoende av satelliter fås att:

$$\delta t_{br}^{ji} = \delta t_r^{ji} - \delta t_b^{ji} = (\delta t_r^i - \delta t_r^j) - (\delta t_b^i - \delta t_b^j) = 0 \tag{12}$$

På samma sätt fås för de initiala fas-termerna på mottagarna att $\lambda\phi_{0,br}^{ji} = 0$. För korta avstånd mellan bas och rover är atmosfärförhållandena väldigt lika och det kan då antas att $I_{br}^{ji} \approx 0$ och $T_{br}^{ji} \approx 0$. Om de mottagare som används är likadana kan det även antas att $d\Phi_{br}^{ji} \approx 0$. Då fås slutligen dessa ekvationer för P_{br}^{ji} och Φ_{br}^{ji}:

$$\begin{aligned} \Phi_{br}^{ji} &= \rho_{br}^{ji} + \lambda N_{br}^{ji} + \epsilon_{\Phi} \\ P_{br}^{ji} &= \rho_{br}^{ji} + \epsilon_{P} \end{aligned} \tag{13}$$

Som ses här så måste ρ och N (*integer ambiguity*) räknas ut. Det som är kvar i ϵ är bland annat flervägsfel vilket beror på kvalitén på antennen samt eventuella fel på mottagaren. Mottagarfel är ungefär två centimeter horisontellt och tre vertikalt.

Det viktigaste för en bra position är alltså att minimera flervägsfelen och att få en snabb lösning på *integer ambiguity*. Det förstnämnda har främst att göra med kvalitén på antennen och det andra har främst att göra med antalet satelliter som mottagaren kan se. För att öka antalet synliga satelliter kan flera oberoende GNSS-system användas, som exempelvis GPS, GLONASS och i framtiden Galileo. [29, 31]

2.3.4 RTKLIB

RTKLIB är en programfamilj för positionering med GNSS. Framförallt kan den användas för att beräkna en position med RTK-teknik. Programfamiljen är framtagen av en forskare vid namn Tomoji Takasu. Programmen och källkoden distribueras under Open Source vilket innebär att de får användas, modifieras och distribueras vidare fritt så länge skaparen är omnämnd. Familjen består av en mängd program som till *Microsoft Windows* finns i GUI-utförande, alltså grafiska applikationer. Till *Linux* finns applikationerna i CUI-utförande. Noggrannheten hos *RTKLIB* är inte sämre än den hos liknande kommersiella programvaror. [33]

De fem program ur familjen vilka är intressanta för projektet är *RTKRCV*, *STR2STR*, *RTKCONV*, *RTKPOST* samt *RTKPLOT*. Dessa program kommer beskrivas nedan. [34]

RTKRCV
RTKRCV är det program som beräknar positionen i realtid. Detta är CUI versionen av programmet. GUI versionen heter RTKNAVI, men tekniken bakom dem är samma. För att beräkna positionen i RTKRCV behövs att den dator som programmet körs på har tillgång till mottagardata i realtid som inkluderar både kod och bärvågsdata. Den måste även ge den specifika nav-datan. Användaren måste även konfigurera var datan från basstationen ska hämtas samt var basstationens faktiska position är. Dessa inställningar görs i en konfigurationsfil som RTKRCV sedan läser av för att bestämma inställningarna som programmet ska köras med. RTKRCV tar sedan denna data som input och ger den beräknade positionen som output.

För en rover r och en bas b används ekvation (13) för att beräkna avståndet mellan satellit och rover. Utifrån dessa ekvationer bestäms positionen med hjälp av ett *Extended Kalman Filter* med en tillståndsvektor som innehåller position, hastighet och *integer ambiguity*. Detta är en utökad version av ett Kalmanfilter vilket även fungerar på icke-linjära

system. Dock lineariserar det tillståndsvektorerna så systemet måste vara differentierbart. Ett Kalmanfilter kan i allmänhet anses ha två steg, prediktion och korrektion. Ett värde på de nuvarande tillståndsvariablerna tas fram baserat på tidigare mätningar. När värdet på dessa tillståndsvariabler är kända uppdateras det uppskattade värdet genom att använda ett viktat medelvärde, där de värden som har mindre fel viktas mer. Detta filter ger de så kallade "float solutions". Det som kännetecknar denna är att *integer ambiguity* inte har lösts ännu, då de värden som ges av kalmanfiltrets tillståndsvektor inte är heltal.[34]

För att lösa *integer ambiguity* används kännedomen att *integer ambiguity* måste vara ett heltal. För att lösa detta används *Integer Least Square*-metoden vilken söker närmaste heltalslösningen till ett minsta kvadratproblem. Minsta kvadratproblemet formuleras med det av kalmanfiltret givna värdet på *integer ambiguity*. Efter att det har lösts jämförs den bäst erhållna lösningen med den näst bästa. Om förhållandet mellan dessa är större än ett på förhand konfigurerat värde (*Arthres*) används det erhållna heltalet för att förbättra den position som erhölls ur kalmanfiltret. Om lösningsmetoden är 'kinematic' tillkommer även hastigheten i ekvationen ovan som en variabel. Denna finns då även som en tillståndsvariabel vilken löses av Kalmanfiltret. Det finns även en speciell metod i RTKLIB för att öka stabiliteten i lösningen. Denna kallas fix-and-hold och om en fix-lösning uppnås tar den det erhållna heltalsvärdet på *integer ambiguity* och använder det som indata i Kalmanfiltret. Detta gör att stabiliteten på lösningen ökar då det är ganska osannolikt att ett annat heltalsvärde erhålls än det som gavs av den förra fix-lösningen. För mer information om kalmanfiltret och lösningen av *integer ambiguity*, se RTKLIBs manual.[29, 34]

SBAS är den status som fås när endast mottagaren hos rovern används. Att endast roverns och inte basstationens mottagare används beror ofta på anslutningsproblem i kommunikationen mellan basstationen och gräsklipparen. SBAS är noggrannare än helt vanlig GNSS då mottagaren får korrektionsdata från några speciella satelliter som fås från referensstationer runt om i världen. Referensstationerna skickar sin korrektionsdata via ett data-center till satelliterna och sedan från satelliterna till mottagaren via en modifierad version av det vanliga GNSS-meddelandet. [35]

STR2STR
Detta är det program som körs på basstationen. Det programmet gör är att skicka vidare mottagardatan från basstationen till rovern. Programmet skickar datan i ett format som kallas för RTCM3, vilket är ett standardformat för att skicka korrektionsdata. Detta innebär att basstationen skickar sin positionsdata samt även de meddelanden vilka mottagaren mottar från satelliten som nav-filer med mera. [29]

RTKCONV
Detta program tar rådata som kan ha tagits från vilken mottagare som helst och konverterar den till det standardiserade RINEX formatet. Detta är nödvändigt för att kunna efterbehandla datan. Detta är GUI-versionen av programmet. [29]

RTKPOST
Detta program använder data som är på RINEX form och beräknar sedan positioner utifrån den här datan. Programmet kan ses som en efterbehandlingsversion av *RTKRCV*

och beräkningstekniskt fungerar den i stort sett likadant. Detta är GUI-versionen av programmet. [29]

RTKPLOT

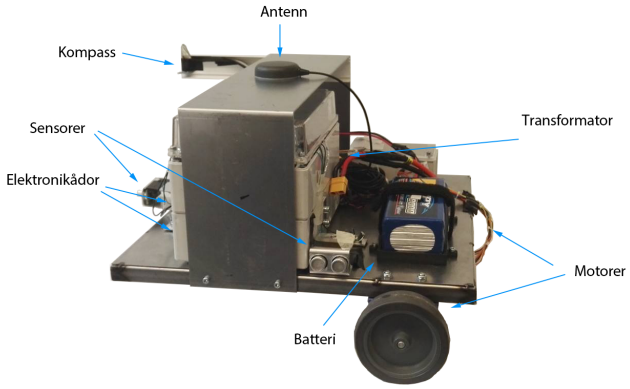
Programmet *RTKPLOT* finns bara som GUI-program och körs således på *Windows*. Det kan användas till att plotta de positionsfiler som återfås ur *RTKRCV* för att i efterhand analysera positionens noggrannhet och liknande. I programmet markeras punkter med olika färger beroende på lösning. Fix markeras grönt, float gult, SBAS rosa och single rött. Programmet har inbyggda funktioner för att plotta positionen i *Google Maps* samt *Google Earth*. [29]

3 Metod

För att på bästa sätt tydliggöra genomförandet av projektarbetet kan tillvägagångssättet delas upp i flera övergripande delar vilka beskrivs nedan.

3.1 Val av komponenter

Innan projektarbetet kunde ta form var det viktigt att vid ett tidigt skede bestämma vilka huvudkomponenter som skulle inhandlas. Genom att använda kunskaperna från teoriavsnittet, kunde en lista med de huvudsakliga delarna fastställas. Hur prototypen blev och var de olika komponenterna sitter kan ses i figur 6 nedan.



Figur 6: Den färdiga prototypen av den robotgräsklippare projektet tagit fram.

Enkortsdator och expansionskort
GNSS-mottagaren som använts i projektets två mottagare, basstationen och gräsklipparen, är NV08C-CSM. Denna är monterad på ett specialdesignat expansionskort för Raspberry Pi kallat RasPiGNSS som även har en kontakt för en antenn [36]. Den gör det möjligt att få data från mottagarmodulen direkt via seriellkopplingarna på en Raspberry Pi [37]. Anledningen till att valet av GNSS-mottagare föll just på denna var dess låga pris i förhållande till funktion. Mottagaren ger en rå GNSS-ström till skillnad från andra billigare alternativ vilket gör den bra för RTK-tillämpningar. Då RasPiGNSS är tillverkad för att fungera för Raspberry Pi Model 2 B blev det naturligt att valet föll på denna enkortsdator. En tredje Raspberry Pi användes för sensorer och motorstyrning. Detta då RasPiGNSS täckte viktiga GPIO-pins samt att det fanns en oro att en enhet inte skulle klara av både beräkning av position och styrning av motorer. Satellitmottagaren på gräsklipparen användes för att beräkna positionen och sedan fördes datan över till Pi-enheten som styrde motorerna och sensorerna.

Antenner
För att använda det valda expansionskortet till RTK-syften krävdes kvalitetsantenner med bra egenskaper att ta emot satellitsignaler. Med rekommendation från skaparen bakom den valda GNSS-mottagaren valdes två Tallysman TW-2410. Dessa antenner är tillverkade för industrier där exakt positionering är ett måste. Rekommenderade användningsområden är jordbruk och militär. Antennerna använder L1-bandet inom frekvenserna 1574-1606 MHz och är kompatibla med GPS, GLONASS och SBAS. Antennen ska dessutom vara

extra bra på att avvisa studsande indirekta signaler samt fungera under trädkronor vilket är viktigt för projektet. [38]

Motorer och hjul
För att gräsklipparen skall kunna köra på ett smidigt sätt valdes att ha en trehjulslösning. Två av hjulen är kopplade till varsin motor, vilka kan användas till att köra hjulen framåt och bakåt. Det tredje hjulet är ett apparathjul, mer allmänt känt som kundvagnshjul, vilket kan snurra fritt. Detta ger gräsklipparen en så pass liten svängradie att den har möjlighet att "snurra på stället" vilket är bra för dess flexibilitet.

Valet av motorer föll, via rekommendation av forskningsingenjör Göran Stigler, på två stycken EMG30 vilket är en likströmsmotor på 12 volt. Motorerna är designade för små eller medelstora robotapplikationer och kommer kompletta med inbyggda växellådor med en utväxling på 30:1, sensorer för varvräkning samt två gummihjul med en diameter på 100 millimeter. Motorkortet MD25 används för att styra motorerna. [39]

Motorkortet MD25 har en mängd ytterligare funktioner som till exempel ström och spänningsmätning. Kortet matas med en spänning på mellan 9-14 volt och kräver en strömkälla som kan leverera ström upp till 6 ampere under kortare perioder. Kommunikationen till kortet kan antingen göras via seriell- eller i2c-koppling på Raspberry Pi-kortet. Då gruppen hade tidigare erfarenhet av i2c valdes denna typ av koppling. Ännu en fördel med MD25 är att den har inbyggda funktioner för olika körstilar, till exempel att svänga [40]. Motorkortet har även flera i2c-kopplingar, vilket gör att flera enheter kan kopplas på samma i2c-koppling som motorkortet använder.

Programmeringsspråk
Programmeringen av styrprogrammet till robotgräsklipparen gjordes i *C*. Valet av programmeringsspråk gjordes dels på grund av gruppens tidigare erfarenhet i *C* men även då motorkortet levererades med exempelkod skrivet i *C*.

Sensorer
Till prototypen valdes två typer av sensorer. Ultraljudssensorer av modell *Devantech SRF05* [41], vilka användes till hinderdetektion och en elektronisk kompass *Honeywell HMC5883L* [42]. Kompassen användes för att avgöra gräsklipparens riktning vilket även fungerade när den var stillastående, till skillnad från om endast GNSS-mottagaren skulle användas till riktningsberäkning.

3.2 Programvara

RTKRCV var den programvara som användes på gräsklipparen. Inställningarna till programmet gjordes som tidigare nämnt i konfigurationsfiler och den konfigurationsfil som användes finns i Appendix D. En av de viktigare inställningar som gjordes var *posmode=kinematic* för att RTKRCV även skulle beräkna hastigheten och använda RTK-teknik för att beräkna positionen. En annan viktig inställning var *armode=fix-and-hold*, vilket gjorde att vi använde fix-and-hold metoden. Fix-and-hold var den inställning som gav den mest stabila lösningen eftersom *integer ambiguity* då hålls relativt konstant under mätningen. Andra inställningar som gjordes var att ta bort alla satelliter under 15^o sett från himlen för att inte få in dåliga signaler då dessa kan introducera betydande flervägsfel

speciellt i en tätbebyggd miljö. Signaler som var svaga filtrerades även de bort för att dessa signaler kunde ha stora fel vilket inte är bra att få in i Kalmanfiltret då detta kan ha en stark påverkan på positionen.

3.3 Positionsmätning

För att mäta gräsklipparens position användes RTK-teknik. Detta system bestod av en basstation som utgjordes av en Raspberry Pi med tillhörande expansionskort RasPiGNSS och antenn samt den mobila enheten vilken monterades på gräsklipparen som bestod av samma komponenter. På basstationen kördes programmet STR2STR vilket konverterade datan från rådataformatet *nvs*, vilket var specifikt för den använda mottagaren, till RTCM3 och skickade datan via en TCP-server. På den mobila enheten kördes programmet RTKRCV som beräknade dess position.

I starten av projektet, innan prototypen var färdigställd och för enkelheten med att vara inomhus för konfigureringen av programvaran, monterades antennerna i ett takfönster inne i laboratoriet. Detta fungerade relativt bra under konfigureringen, men när positionen mättes drev positionen iväg med tiden. Detta antogs bero på att glaset påverkade signalen genom flervägsfel.

För att lösa detta problem flyttades projektet ut, men det var svårt att hitta en plats där basstationen kunde stå då den behöver så fri sikt som möjligt. Detta på grund av alla höga byggnader runtom Chalmers samt i centrala Göteborg. För att lösa problemet placerades basantennen på maskinhusets tak där den hade fri sikt.

Sedan behövdes den exakta positionen för basstationen tas fram då detta är väldigt viktigt för en noggrann position. Först gjordes en mätning med basstationens GNSS-mottagare under ett par timmar för att sedan ta ett medelvärde av denna position. Detta gjordes med PPP-teknik som använder bärvågen men endast en mottagare och är i övrigt beroende av korrektionsdata såsom SBAS. Ett par timmar visade sig vara för kort tid samt att fel metod för höjdberäkning använts vilket gjorde att den angivna höjde var några meter fel. En ny mätning med korrekt metod för höjdberäkning utfördes under drygt 375 timmar. Medelvärdet togs från denna mätning och matades in som ny position för basstationen. Då PPP använder mycket korrektionsdata upptäcktes dock att en ännu bättre position kunde erhållas med *RTKPOST* som efterbehandlar rådata som samlats in från en mottagare. Ett test gjordes genom att samla in rådata över en natt och sedan efterbehandla denna nästa dag. Positionen som gavs av denna data ansågs vara bättre i förhållande till var antennen faktiskt satt så detta värde användes istället. Detta gjordes dock ganska sent i projektet vilket gjorde att bara mätningarna för ruttkörningen använde denna förbättrade position.

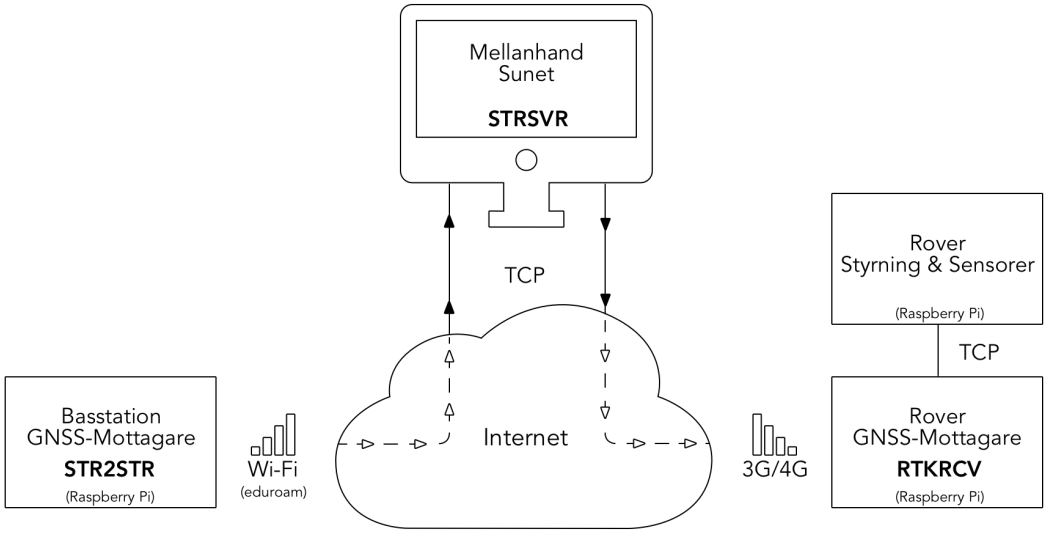
I projektets början gick det snabbt att få en position till mottagarna och denna kunde läsas av genom terminalen i Raspberry Pi. Sedan påbörjades arbetet med att få in denna positionsdata i programvaran som skulle användas, RTKLIB. För att få in positionsdatan krävdes mycket arbete med att redigera konfigurationsfilerna för att programvaran skulle, till exempel, hämta GNSS-datan från rätt seriell-port. RTKRCV fick data från basstationen via en TCP-klient, vilken hämtade datan genom TCP-servern som körs på basstationen. Hur detta gjordes rent praktiskt under projektets gång förklaras nedan.

3.4 Enhetskommunikation

Kommunikationen mellan basstationen och gräsklipparen gjordes via TCP. Till en början placerades basstationen i närheten av där gräsklipparen användes. Detta innebar att en WiFi-router räckte för att sköta kommunikationen mellan gräsklipparen och basstationen. Till Raspberry Pi fanns det en WiFi-mottagare vid namn WiPi. Två sådana införskaffades, en till varje Raspberry Pi. Nu kunde basstationen och gräsklipparen kommunicera trådlöst med varandra via WiFi-routern.

Senare under projektets gång upptäcktes dock att basstationen var tvungen att placeras på en plats med fri sikt mot himlen för att få tillräckligt bra mottagning. Placeringen blev därför på ett av Chalmersbyggnadernas tak. Med detta räckte inte längre endast WiFi-routerns räckvidd och en ny lösning för kommunikationen togs fram. Lösningen blev att ansluta basstationen till campusnätverket *Eduroam*, vilket hade en god täckning där basstationen placerades. Basstationen skickade därefter sin mottagardata via TCP på *Eduroam* över till gräsklipparen vilken kopplades upp mot en mobiltelefon via internetdelning. Anledningen till att inte gräsklipparen också kopplades upp mot *Eduroam* var att mottagningen till det nätverket inte var tillräckligt utomhus. Då *Eduroam* är ett publikt nätverk och tilldelar dynamiska IP-adresser blev det otympligt att varje gång ändra i konfigurationsfilen vilken IP-adress basen skickade från. För att kringgå detta problem användes en "mellanhand" i form av en dator placerad i PPU-laboratoriet. Den var ansluten till nätverket *Sunet* på en fast IP-adress. Sedan konfigurerades basen att skicka sin data till denna fasta IP-adress och gräsklipparen ställdes in att hämta data från samma IP-adress. Mellanhanden använde ett program i RTKLIB-familjen i GUI-utförande vid namn Stream Server, eller STRSVR. Se figur 7 för en överskådlig bild över nätverket.

För att kunna använda Raspberry Pi-enheterna på gräsklipparen samt att kunna kontrollera basstationen när den satt på taket användes VNC. Detta då det var nödvändigt att kunna kontrollera enheterna på avstånd. På Raspberry Pi-enheterna kördes en server som startade när enheterna startades. Klienterna kunde köras på valfri dator eller smartphone vilket gjorde att det var väldigt enkelt att skicka kommandon till Pi-enheterna. Då Raspberryn som styrde motorerna inte hade direkt tillgång till internet gick det dock inte att använda VNC direkt. Detta problem löstes genom att en klient kördes på mottagarenheten på gräsklipparen som kunde kommunicera med servern på Pi-enheten med motorkortet. Denna vy kunde sedan skickas till en extern klient.



Figur 7: Kommunikationsnätverket mellan basstationen och gräsklipparen, inklusive mellanhanden.

3.5 Design av prototyp

Relativt tidigt i projektet påbörjades konstruktionen av en prototyp för att kunna fästa enkortsdatorerna för att de inte skulle behöva kopplas på och av vid varje arbetstillfälle. Detta då dessa är relativt ömtåliga och hade kunnat gå sönder.

Av enkelhet och ur hållfasthetsperspektiv ansågs det vara en bra idé att svetsa ihop fyra ihåliga stållängder till en kvadratisk ram. På denna ram svetsades det på en två millimeter tunn plåtskiva som skulle agera underlag. På denna monterades sedan en elektroniklåda där en enkortsdator hade skruvats fast. Lådan innehöll ett antal hål vilket gjorde det enkelt att hantera kablage. Detta var den första konstruktion som gjordes, med avsikt att det ytterligare skulle monteras på batteri, en till enkortsdator, motorer och hjul samt att det fanns möjligheter att montera andra komponenter vilka eventuellt skulle behövas.

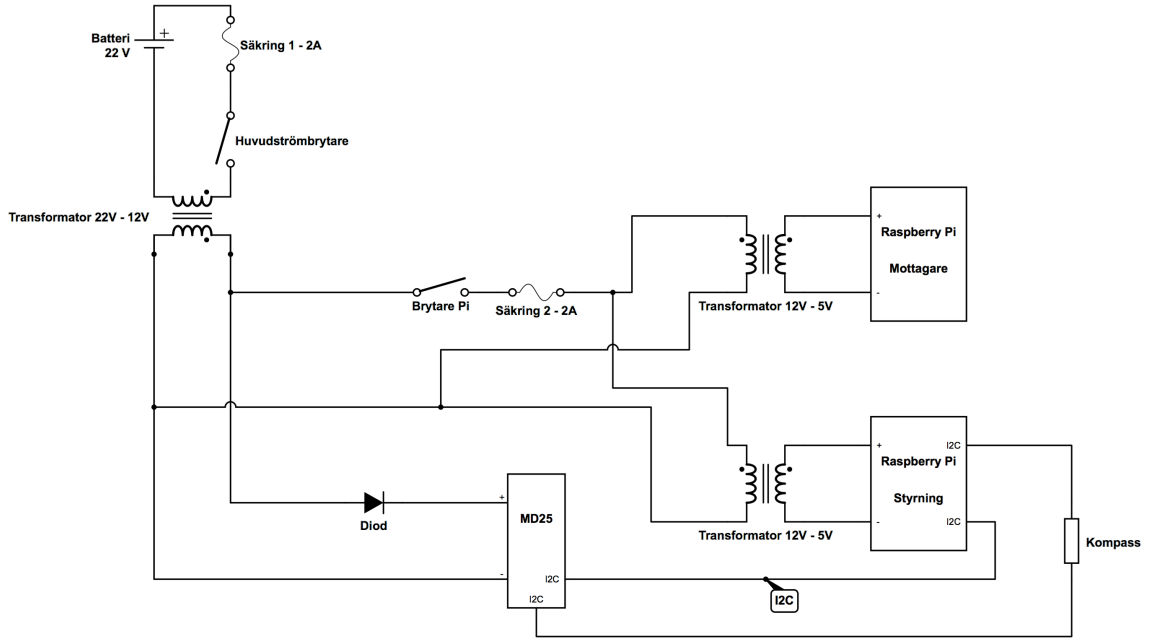
Det var även viktigt att, relativt tidigt i projektet, få en mobil robot. Detta på grund av att det skulle kunna vara möjligt att enkelt testa diverse program och kunna kalibrera dessa under projektets gång. Därför monterades motorer och hjul så snabbt som möjligt. I samband med detta gjordes en del kabeldragning för att koppla ihop komponenterna, detta innebar en hel del lödning.

Motorkortet och ytterligare en Raspberry Pi fästes sedan i ytterligare en elektroniklåda vilken även den monterades på plåtskivan. Bredvid de två elektroniklådorna monterades sedan ett 22 volts batteri som var lätt avtagbart. För att antennen skulle få bra sikt mot satelliterna och för att signaljordas fästes en bockad plåtskiva över och runt prototypen där antennen fästes med sitt magnetfäste.

Då batteriet levererade en spänning på 22 volt, medan motorkortet skulle matas med 12 volt och Raspberry Pi-enheterna med 5 volt behövdes transformatorer. Tre transformatorer införskaffades, en 22 till 13,6 volt och två 12 till 5 volt med USB-kontakt då Raspberry Pi-enheterna drivs via USB. Anledningen till att två 12-5 volt transforma-

torer införskaffades var för att inte belasta en sådan transformator med två Raspberry Pi-enheter. Valet av USB-transformator föll på just dessa då de fanns tillgängliga i laboratoriet som kunde lånas till projektet.

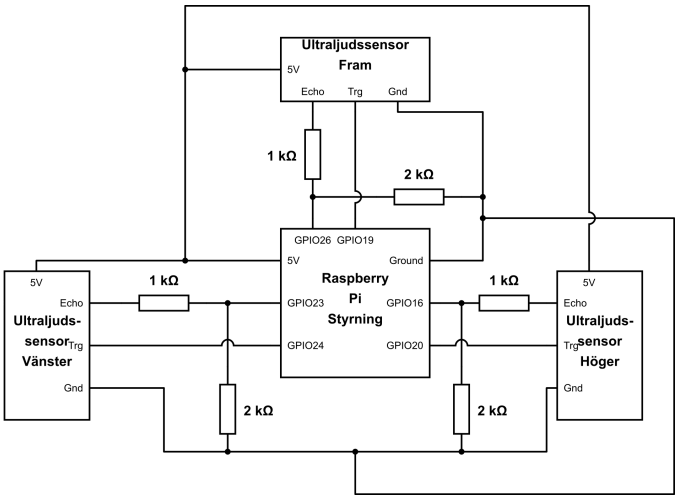
Motorkortet krävde att strömkällan kunde leverera toppströmmar på upp mot 6 ampere och utöver detta är två Raspberry Pi också kopplade på samma transformator. Transformatorn kunde leverera 6 ampere kontinuerligt och toppströmmar upp till 10 ampere. Kopplingsschemat för strömmen i gräsklipparen kan ses i figur 8.



Figur 8: Kopplingsschema för elektroniken i gräsklipparen.

För att säkra motorkortet och enkortsdatorerna monterades även två säkringar i en låda. Dessa säkringar valdes till vardera 2 ampere. Det monterades också två strömbrytare på denna låda, en användes till att koppla på batteriet och den andra för att koppla på enkortsdatorerna.
En elektronisk kompass monterades på en arm som stack ut framför roboten. Detta för att minska påverkan av störande magnetfält från de övriga komponenterna. Kompassen var kopplad till de extra i2c-kontakterna på motorkortet som kan ses i figur 8. Kompassen drevs med 3,3 volt från Pi-enhetens GPIO-kontakter.

På prototypen monterades tre ultraljudssensorer, en fram och två åt sidorna. Figur 9 visar kopplingsschemat för hur dessa är kopplade till enkortsdatorn. Ultraljudssensorerna drivs med 5 volt och när de skickar tillbaka signalen med avståndet görs detta med 5 volt. Då en Raspberry Pi endast kan ta emot signaler på upp till 3,3 volt måste resistorer användas.



Figur 9: Kopplingsschema över ultraljudssensorerna i gräsklipparen.

3.6 Styrning av prototyp

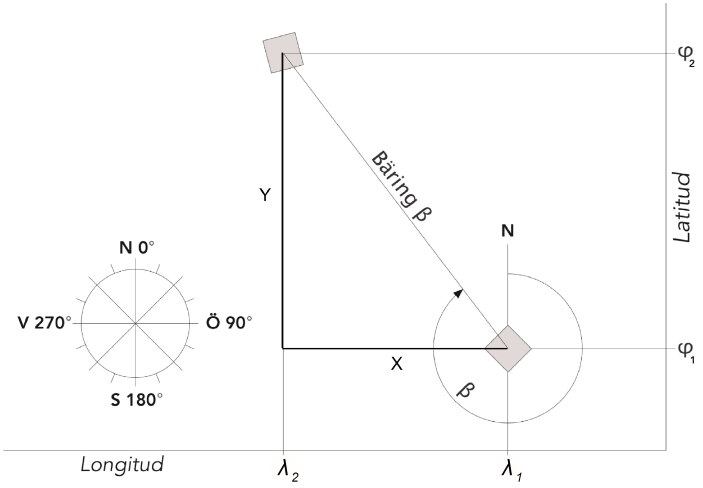
För att styra prototypen användes en tredje Raspberry Pi där motorkortet och kompassen kopplades till i2c-kontakterna och ultraljudssensorerna kopplades till GPIO-kontakterna. För att få positionsdata användes en Ethernetkoppling mellan den andra Raspberry Pi-enheten för att föra över datan via TCP. Programmet som utvecklades tog in data från kompassen och ultraljudssensorerna, öppnade en TCP-server vilken tog emot den relevanta datan från GNSS-mottagaren och omvandlade den till siffror. Ifall ingen position erhållits eller lösningen tappats så stänger RTKLIB TCP-klienten, vilket gör att programmet endast tar emot nollor. Programmet hanterar detta genom att stanna roboten och återställa TCP-kopplingen.

För att roboten skulle köra enligt en planerad rutt läste den in koordinater från en fördefinierad fil. Därefter räknades bäringen mellan den första fördefinierade koordinaten och den nuvarande koordinaten. Då avståndet mellan koordinaterna är kort, kan ett antagande göras att jorden är platt. Då kan bäringen β räknas ut med ekvation (14), där λ_1 och φ_1 är de nuvarande longitudinella och latitudinella koordinaterna, λ_2 och φ_2 är de önskade koordinaterna och r_1 och r_2 är jordradien för punkterna [43]. Bäringen beskrivs visuellt i figur 10.

$$
\begin{aligned}
&\Delta\lambda = \lambda_2 - \lambda_1 \\
&\Delta\varphi = \varphi_2 - \varphi_1 \\
&X = r_1 \cos(\varphi_1)\Delta\lambda \\
&Y = r_2\Delta\varphi \\
&\beta = \arctan(\frac{Y}{X})
\end{aligned}
\tag{14}
$$

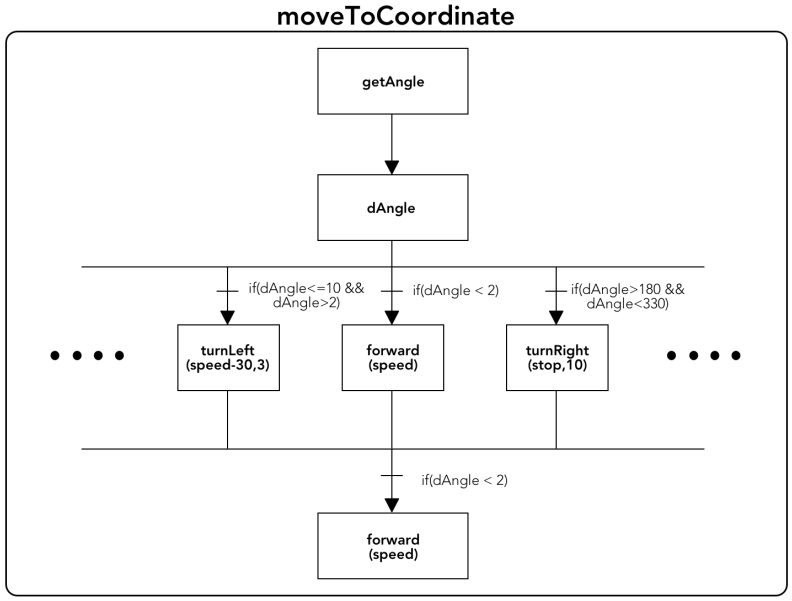
Notera att i beräkningarna antas att radierna r_1 och r_2 har samma storlek och på så sätt förkortas bort. Detta är en approximation och en korrekt uträkning kan ses i Aviation Formulary av Ed Williams [43]. Notera även att arctangens endast fungerar inom inter-

vallet -90° till 90°. I programmeringen löses detta genom att använda funktionen atan2, som även tar med information om vilken kvadrant vinkeln befinner sig i.



Figur 10: Bäring från den nuvarande positionen till den önskade positionen.

Roboten styrde sedan mot den önskade positionen genom att reglera motorerna med hjälp av bäringen. När gradskillnaden var större än ± 30° svängde roboten på stället genom att köra fram med ena hjulet och bak med andra. Med en gradskillnad inom ± 30° räckte det att roboten ökade hastigheten på ena hjulet och sänkte på andra och tillslut när gradskillnaden var under ± 2° körde roboten rakt mot den önskade positionen. När roboten befann sig inom ett visst intervall runt den önskade positionen läste den in nästa koordinat från filen och navigerade sedan till denna. Ett förenklat flödesschema kan ses i figur 11. För mer utförliga flödesscheman, se Appendix C.



Figur 11: Förenklat flödesschema över funktionen moveToCoordinate i programmeringen.

Ifall något hinder befann sig framför roboten under körning detekterade ultraljudssensorn detta. Roboten navigerade sedan runt hindret med hjälp av ultraljudssensorerna placerade på sidorna för att sedan återuppta sin planerade körbana.

3.7 Kostnad

För att kunna dra några slutsatser om huruvida en robotgräsklippare navigerad med RTK-teknik är konkurrenskraftig på marknaden gjordes det en kostnadsberäkning på de mest kostsamma delarna i prototypen. Plåtmaterial, skruvar och annat dylikt ansågs som småkostnader i jämförelse med de huvudsakliga delarna, och togs därför inte med i analysen. Det är viktigt att notera att dessa inköp är gjorda för endast en produkt och priserna kommer alltså att vara mycket förmånligare vid en massproduktion.

4 Tester och resultat

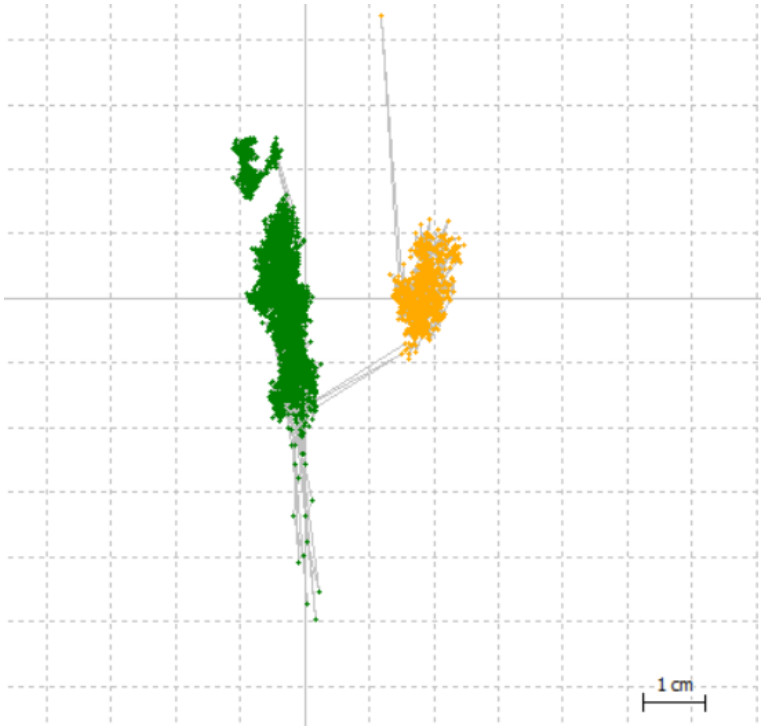
I detta kapitel presenteras de tester, och deras resultat, som utförts för att analysera noggrannheten i systemet och prototypens förmåga att uppnå de mål som definierats.

4.1 Tester

Inställningarna till RTKRCV, programmet som kördes på gräsklipparen, görs i konfigurationsfiler. Dessa inställningar tog mycket tid och för varje inställning som ändrades gjordes en testmätning för att se resultatet.

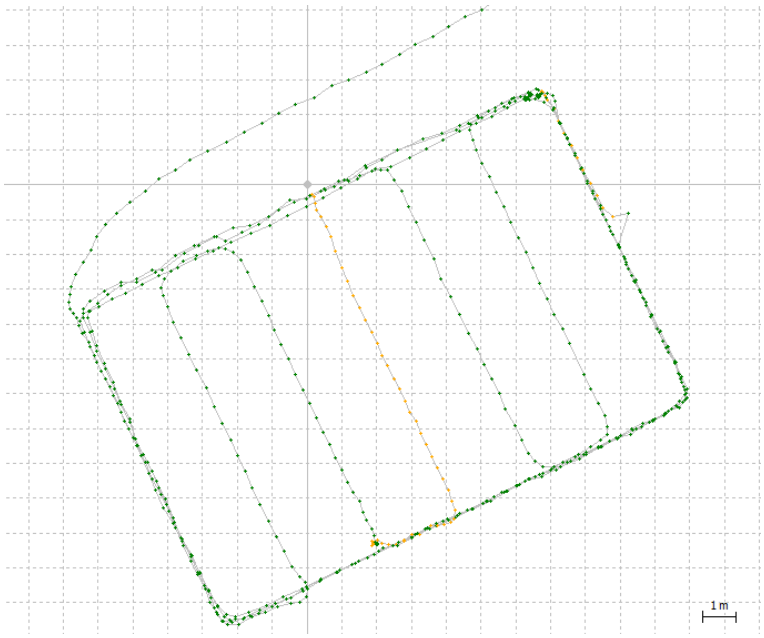
För att testa precisionen i navigationssystemet utformades ett antal tester. Det första testet som gjordes var att placera ut GNSS-mottagaren som skulle användas till gräsklipparen och se hur stor osäkerheten var för den återgivna positionen. Det som kunde undersökas i detta test var hur pass konstant positionen var. Till en början i projektet var positionen inte konstant, den drev sakta iväg, denna mätning kan ses i appendix B. Orsaken till detta var att basstationens position inte var korrekt. Höjden var satt för högt vilket orsakade positionens drift. Detta åtgärdades genom att mäta basstationens position mer noggrant.

Ett nytt test utfördes för att beräkna hur pass exakt den stationära positionen som återfås ur navigationssystemet är när gräsklipparen är stillastående. Resultatet kan ses i figur 12. Resultatet är mycket konstant vilket tyder på en låg osäkerhet i positionen. De flesta mätvärdena ligger inom 4,5 centimeter från norr till söder, endast 17 av punkterna ligger utanför detta område vilket motsvarar 0,33 % av det totala antalet punkter.



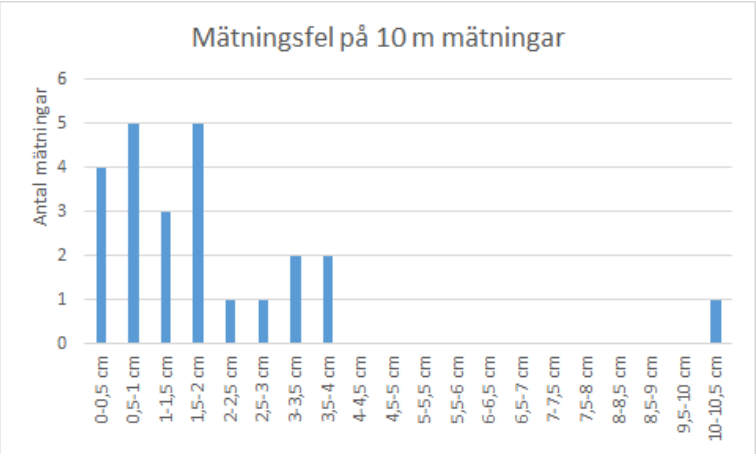
Figur 12: Statisk mätning för att avgöra osäkerheten i positionen. Alla förutom 0,33 % av punkterna var inom 4,5 cm.

När testerna för osäkerheten gjorts var felet vid rörelse intressant. För att testa hur positionen blev vid rörelse utfördes ett test där mottagaren placerades på huvudet på en projektmedlem varpå denne gick runt en parkeringsyta, banan sträckte sig 10x15 meter. Detta test gjordes endast för att på ett ungefär se hur exakt positioneringen är. Resultatet från testet kan ses i figur 13. Resultatet visar på en mycket god precision även vid rörelse. Dock spelar den mänskliga faktorn in i testet så ytterligare tester utformades för att mer exakt mäta precisionen vid rörelse.



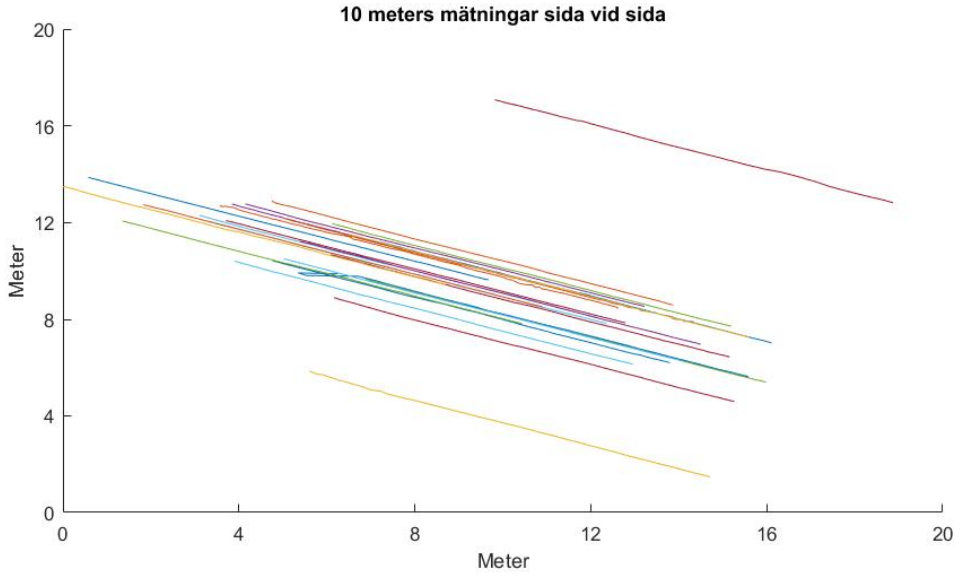
Figur 13: Mätning vid parkeringsruta där positionsmätning vid rörelse testas. Mätningen visar att positionen som erhålls är bra även vid rörelse.

Då testet ovan införde en mänsklig faktor skapades ett nytt test där gräsklipparen upprepade gånger dras längs en linje på tio meter. Vid tiometersmarkeringen placerades ett stopp så att gräsklipparen skulle stanna vid rätt ställe och gräsklipparen drogs längs denna linje 25 gånger. Resultaten från testet kan ses i figur 14. I testet kan den mänskliga faktorn helt uteslutas. Alla mätningar utom en resulterade i ett fel på under 4 cm och 72 % av mätningarna resulterade i ett fel under 2 cm. Exakta värden kan ses i tabell 3 i appendix B.



Figur 14: Avvikelse från 10 meter för mätningar vilka gjordes på en 10 meter lång linje. Förutom ett uteliggande värde så ses att all data är inom 4 cm. Majoriteten av datan ligger under 2 cm.

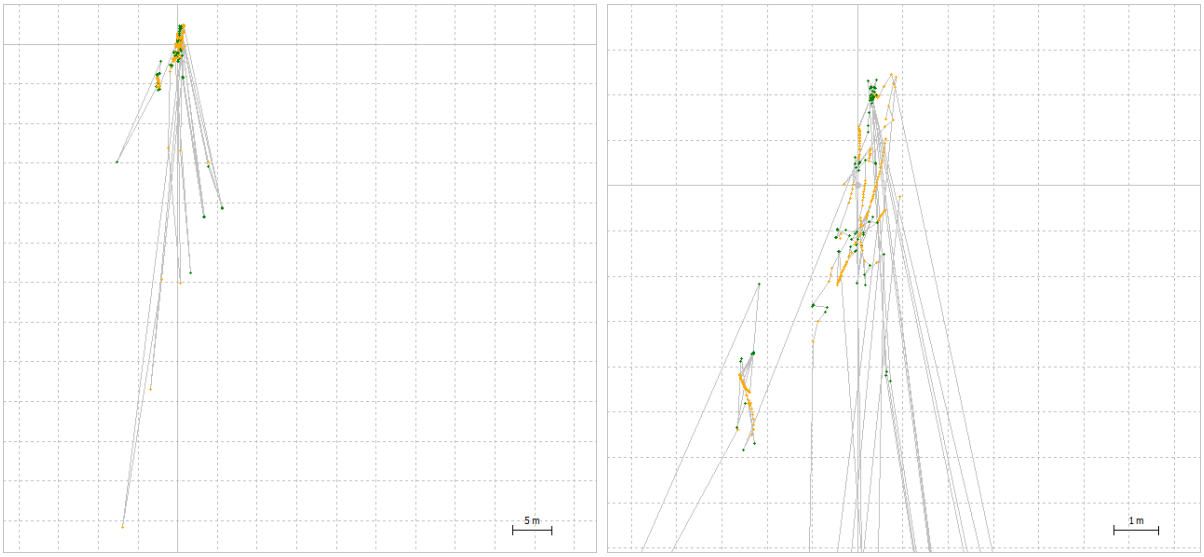
Under testerna till figur 14 var resultaten mycket noggranna inom mätningarna. Men när mätningarna placerades i samma plot så upptäcktes att startpositionerna skiljde sig åt mellan mätningarna, se figur 15. När RTKRCV startas om återgår inte positionen till exakt samma plats där den var när programmet avslutades. Ett fel introduceras vid uppstarten. Felet är i storleksordning meter.



Figur 15: Plot över alla mätningar som gjordes när gräsklipparen drogs på en 10 meter lång linje. Positionen är inte på samma ställe vid varje mätning utan varierar flera meter.

Alla tester ovan utfördes vid en plats med mycket god vy över himlen. Då en gräsklippare måste kunna navigera i en villamiljö utfördes ett statiskt test i närheten av skog. Mottagaren placerades tolv meter från träden åt ena hållet och 18 meter åt andra. Satellitmottagarens position blev mycket osäkrare och hoppade flera meter under mätningen. Positionen blev alltså inte lika robust som vid goda förhållanden, se figur 16 för en bild på platsen och figur 17 för resultatet av testet.

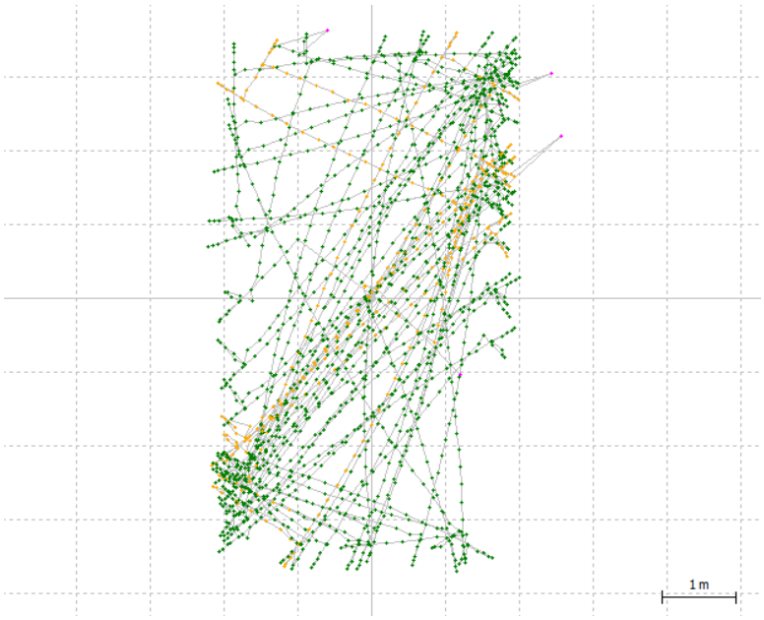
Figur 16: Bild på platsen för mätningar av statisk position vid delvis skymd himmel.



(a) Utzoomat *(b) Inzoomat*

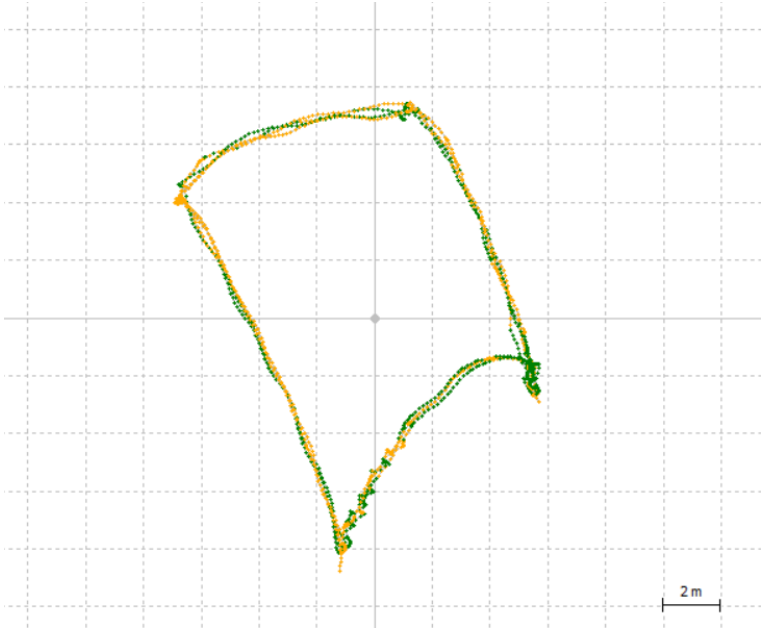
Figur 17: Statisk mätning i närheten av träd. Mätningen gav en stor osäkerhet i positionen.

Testerna ovan testade hur pass noggrann tekniken är. Sedan följde tester på de program vilka tagits fram för körningen av prototypen. I figur 18 visas det resultat som områdesbegränsningsprogrammet återgav. Rektangeln definierades utifrån den första koordinaten gräsklipparen uppmätte, se kod 4 i appendix E. Programmet satte att denna koordinat fick ändras inom ett visst intervall. På så sätt skapades en tillåten rektangel inom vilken gräsklipparen fick köra. Roboten åkte rakt fram tills den registrerade en position utanför området varpå den backade tillbaka och vände för att sedan fortsätta rakt fram. Vid detta test fungerade alltså roboten likt en vanlig gräsklippare med studsfunktion med skillnaden att den kände av området med dess koordinater istället för en avgränsningskabel.



Figur 18: Test där områdesbegränsning uppvisas. Roboten höll sig inom ett fördefinierat område.

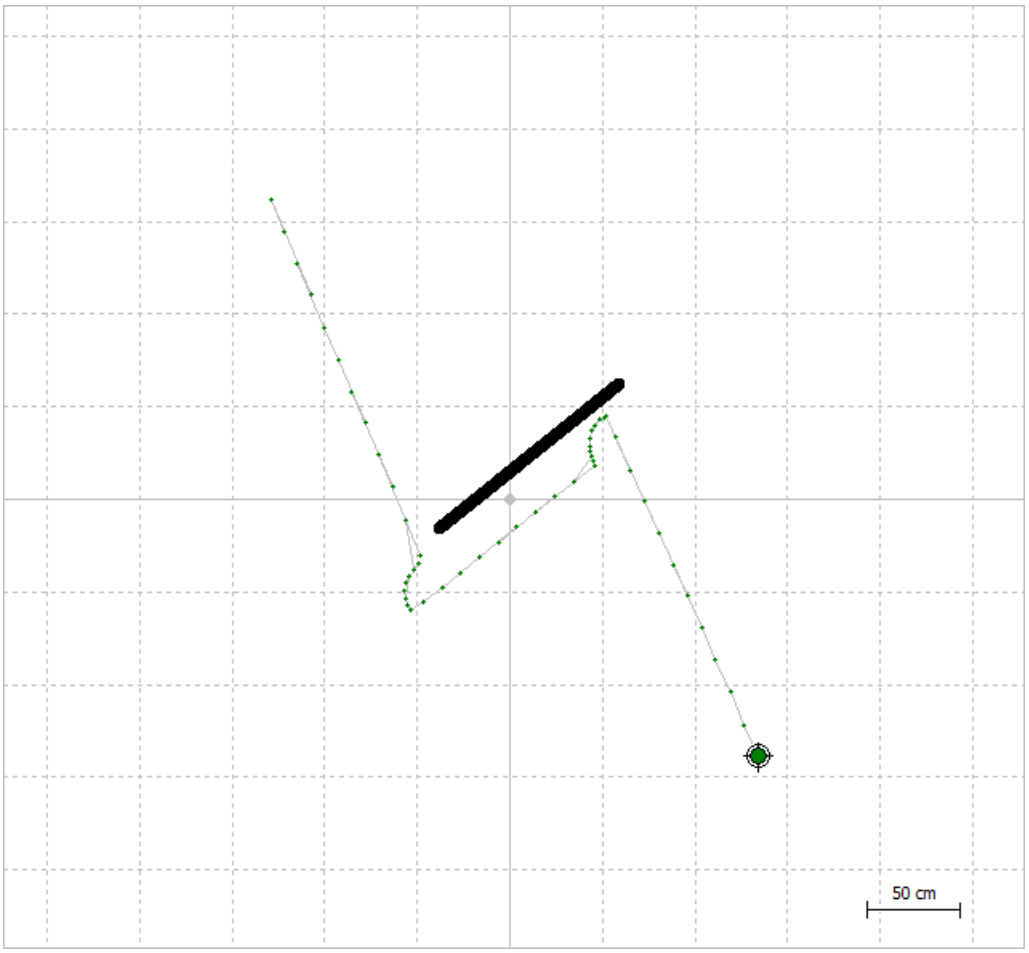
För att testa gräsklipparens förmåga att navigera mot en given koordinat skapades ett test för detta. Koordinaterna till en fyrkant hämtades från *Google Maps* och matades sedan in i en textfil vilken programmet på gräsklipparen läste in och navigerade utifrån. Gräsklipparen navigerade fyra varv kring den fyrkantiga banan och resultatet presenteras nedan i figur 19. Figuren visar på att prototypen kan navigera mellan förprogrammerade koordinater. Dock kör den inte den kortaste vägen till koordinaten när den navigerar mellan de nedre koordinaterna. Detta diskuteras djupare i kapitlet Diskussion.



Figur 19: Test där gräsklipparen navigerar mellan fyra koordinater. Resultatet visar att gräsklipparen kan navigera mellan koordinater.

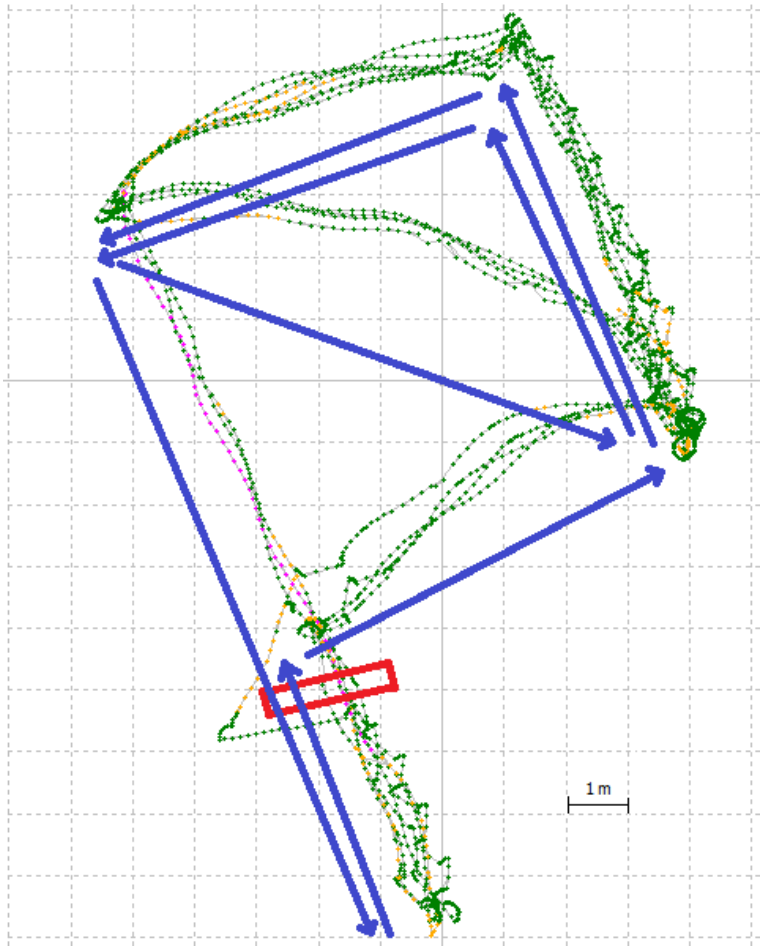
För att testa gräsklipparens möjlighet att hantera hinder utformades ett test för detta.

Roboten programmerades att köra framåt tills dess att den detekterade ett hinder framför sig. När hindret detekterats stannar roboten, svänger 90 grader och kör sedan framåt tills ultraljudssensorn på sidan slutar detektera hindret. Då svänger roboten tillbaka 90 grader och fortsätter framåt. Resultatet från testet kan ses i figur 20.



Figur 20: Test där roboten körs rakt fram mot ett hinder och sedan svänger runt det.

Ett test för att testa körning mot koordinat tillsammans med hinder genomfördes. Vid detta test skapades en rutt med åtta waypoints som roboten åkte till i tur och ordning. På ett av varven placerades ett hinder ut för att testa hinderhanteringen, resultatet kan ses i figur 21. Roboten navigerade efter koordinater och när hindret, den rödmarkerade rektangeln i figuren nedan, upptäcktes körde den runt det och återupptog sedan den planerade ruttkörningen.



Figur 21: Test där roboten undviker ett hinder under ruttkörning. Pilarna markerar hur roboten åkte och den röda rektangeln är hindret som placerades ut under ett av robotens varv. Roboten undviker hindret och återupptar sedan sin planerade rutt.

För att förbättra positionsmätningen önskades att även ta med GLONASS-systemets satelliter. Dessa inkluderades i inställningarna och statiska tester utfördes för att jämföra precisionen med endast GLONASS eller GPS och med de båda samtidigt. Positionen varierade med tiotals meter vid användning av GLONASS vilket gör att resultatet får anses mycket dåligt varför GLONASS ej användes i positionsmätningen. Resultatet från dessa mätningar kan ses i figur 25 och figur 26 i appendix B.

4.2 Kostnad

Enligt kostnadsberäkningen, se tabell 2, blev materialkostnaden för den befintliga prototypen 9 560 SEK. Kostnaden för RTK-navigationssystemet blev 5 200 SEK. Denna kostnad inkluderar två expansionskort, två antenner samt två Raspberry Pi-enheter. Kostnaden för navigationssystemet passerade således den budget som presenterades tidigare med 200 SEK.

Tabell 2: Beräkning av den totala kostnaden för prototypen.

Komponent	Pris/st (kr)	Antal
Motor, hjul, styrkort	1500	1
Raspberry Pi	300	3
22 Volt batteri	~500	1
SD-kort	~70	3
WiPi	250	2
RaspiGNSS	1400	2
Tallysman antenner	900	2
Kompass	100	1
Ultraljudssensor	100	3
Väggadapter	250	1
USB-kablar	~50	3
Transformator 22 volt	~500	1
Transformator 5 volt	~25	2
	TOTALT:	9560

5 Diskussion

I det här avsnittet diskuteras hur arbetet i projektet har gått och hur de erhållna resultaten blivit. En jämförelse mellan den prototyp som tagits fram mot den referensprodukt som tidigare presenterats kommer även göras. En diskussion kommer även föras huruvida resultaten talar för eller emot RTK som ett alternativ till navigering för robotgräsklippare.

5.1 Navigeringsteknik

De resultat som uppmätts under projektet är överlag mycket bra. Noggrannheten ligger i storleksordningen centimeter vilket är vad som krävs av tekniken för att den ska fungera som navigationssystem till en robotgräsklippare. Tester där gräsklipparen uppvisade områdesbegränsning samt förmågan att navigera mot en given koordinat fungerade bra. Möjligheten att navigera till en given koordinat innebär att gräsklipparen kan navigera tillbaka till laddstationen. Detta tyder på att målet med projektet är uppnått då gräsklipparen kan uppvisa områdesbegränsning samt navigera till laddstationen. Gräsklipparen har visat att den kan undvika hinder samtidigt som den kör mot koordinater. Målet att undvika hinder anses således vara uppfyllt, men det finns fortfarande möjlighet att vidareutveckla hinderhanteringen så denna blir smartare.

Den bästa lösningsstatusen som kan fås för en positionslösning är fix, vilket kan ses som gröna punkter i alla plottar. I resultaten kan det ses att även float-statusen är tillräckligt bra för att navigera en gräsklippare. I den statiska mätningen kan ses att när lösningen bytte till float så skiftade den bara 2 cm. I figur 13, där mottagaren flyttades längs parkeringsrutor, kan ses att den enda gula linjen är lika stabil som de gröna linjerna. Även i figur 19, där roboten navigerade mellan fyra koordinater, ses att de gula och gröna linjerna ligger i princip ovanpå varandra. För att uppfylla syftet spelar det alltså mindre roll om statusen är fix eller float. Fix-lösningen är dock mer stabil så det är ändå önskvärt att uppnå fix, men det är inget krav.

Värt att notera är att resultaten är uppmätta vid mycket goda förhållanden avseende fri sikt mot himlen. Vid lägen där himlen är skymd av hus eller träd fungerar inte tekniken lika bra. Därför lämpar sig tekniken bäst för gräsklippare som inte behöver köra i miljöer där himlen skyms av hus eller träd. Passande miljöer är exempelvis golfbanor, fotbollsplaner eller andra stora ytor där klippmönster är viktigt och kostnaden för att lägga ut avgränsningskabel blir för dyr. Vid användning av RTK i vanliga fall där hus, träd och liknande kan vara i vägen för himlen kan det behövas ytterligare teknik för att navigera när inte mottagaren får en position.

Noggrannheten avseende hur pass exakt gräsklipparen kan navigera en förprogrammerad rutt begränsas av programmeringen. I programmet definieras ett tillåtet intervall för målkoordinaterna. Detta för att gräsklipparen inte ska förbli en för lång tid vid koordinaten för att försöka passa in den exakt på millimetern. Detta intervall kan minskas för att öka noggrannheten, dock till lägst fyra centimeter då detta är storleken på osäkerheten i positionen beskriven i figur 12, där osäkerheten i positionen under en statisk mätning visas. När robotgräsklipparen navigerade mellan koordinater, se figur 19, tog robotgräsklipparen inte den snabbaste vägen men dock samma väg varje varv. Anledningen till att roboten inte valde den snabbaste vägen till koordinaten beror på körprogrammets

programmering, där variabler som påverkar svängradie och hastighet specificerats. Dessa behöver ha tillräckligt stora intervall för att klara av avvikelser från kompassens och positionens noggrannhet. Detta problem kan lösas genom att använda fler koordinater (waypoints) att navigera mot, med kortare avstånd mellan dem.

Något som dock noterades är att startpunkten för de mätningar som presenteras i figur 14 skiljde sig åt för varje mätning trots att alla tester gjordes på exakt samma ställe. Detta fel var i storleksordningen meter. Orsaken till felet är troligen ett fel ifrån RTKLIB och inte från själva mätningen. Detta kan bero på vissa inställningar som gjordes för RTKRCV. Speciellt *armode* som valdes till fix-and-hold samt att värdet *Arthres*, som bestämmer hur säker en lösning måste vara för att erhålla fix, var väldigt lågt. Problemet blir att fix-and-hold bara använder *integer ambiguity* från första mätningen som ger en position och sedan används den till alla efterföljande mätningar genom att *integer ambiguity* tas med i Kalmanfiltrets beräkning av positionen. Detta gör att den första mätningen blir väldigt viktig för den absoluta positionen. Detta bör inte ha påverkat det mätningarna skulle visa, nämligen systemets noggrannhet men det är inte bra att ha en robotgräsklippare som har flera meter fel när den startar upp igen. Det går att göra inställningar i RTKRCV som ökar antalet fix-lösningar som krävs för att en *integer ambiguity* ska låsas fast i ett värde och detta värde borde vara ganska högt för att vara säker på att fix-lösningen är bra nog. Det går också att höja *Arthres* som gör att en fix blir säkrare men svårare att uppnå. Dessa inställningar ändrades när mätningarna som testade ruttkörning gjordes och hur till exempel mätningen i figur 21 motsvaras i verkligheten kan ses i figur 22. Koordinaterna var valda genom att titta på satellitbilder från parkeringen för att sedan välja ut kanten på ett antal parkeringsrutor som koordinater. Bilden är ganska otydlig men det går ändå att se att gräsklipparen svänger vid en parkeringsruta så positionen som gavs stämmer bra överens med verkligheten.

Figur 22: En del av mätningen från figur 21 utritat i Google Earth. Kanterna på banan är vid olika parkeringsrutor. Detta visar att positionen som erhålls stämmer bra överens med verkligheten. Kartdata: 2016 CNES/Astrium via Google Earth.

Projektet försökte även använda GLONASS för att bestämma positionen. Enligt tidigare forskning på området ska inte införandet av dessa satelliter i beräkningen göra positionen mindre noggrann. Ungefär två millimeter extra i noggrannhetsfelet skulle införas [44]. Av-

vikelsen projektet fick vid användning av satelliterna var mycket större och detta lyckades inte lösas. Detta kan ha berott på ett problem med kompatibiliteten mellan Raspberry Pi-enhetens hårdvara och RTKLIB eftersom andra som har haft liknande konfiguration har haft samma problem [45]. Själva GLONASS-systemet har även ett problem då alla satelliter skickar sin data på olika frekvenser. Detta introducerar ytterligare ett fel då mottagarfel kan vara olika för olika frekvenser. På grund av detta blir det svårare att lösa *integer ambiguity* för GLONASS-satelliter. Dessa problem är något som behöver undersökas ytterligare. [46]

5.2 Jämförelse

För att jämföra den prototyp projektet tagit fram mot en på marknaden redan etablerad produkt presenterades i avsnitt 2.2 en referensprodukt. Nedan kommer en jämförelse utifrån denna produkt göras mot den produkt projektet utvecklat, mestadels med avseende på navigationssystemet.

Den befintliga produkten använder den välkända avgränsningskabeln och sedan "studsteknik" för att navigera kring gräsmattan. Utöver detta använder den även delvis satellitnavigering med en relativt osäker mottagare. Detta för att avgöra om den har varit på en viss del av gräsmattan under en längre tid och på så sätt kunna hålla sig undan denna del senare för att klippa gräsmattan effektivare. Den produkt som projektet har utvecklat använder en mer exakt satellitnavigering vilket medför att gräsklipparen alltid känner till sin position och på så sätt kan navigera runt gräsmattan på ett mycket effektivare sätt då den alltid vet var den har och inte har klippt tidigare. Detta medför många fördelar. Dels blir arbetet kortare på grund av den effektivare klippningen och det oljud gräsklipparen medför minskas. Vidare slits inte klippaggregatet ut mer än nödvändigt vid den mer effektiva körningen, vilket medför lägre driftkostnader då aggregatet inte behöver bytas lika ofta. Ännu en fördel med en effektivare körning är att batterierna inte behöver laddas lika ofta då den inte använder lika mycket energi som den befintliga produkten. Detta är bra ur en ekonomisk synvinkel men även bra för miljön.

Det helt satellitbaserade navigationssystemet medför dock några nackdelar. Avgränsningen av klippytan måste programmeras in i gräsklipparen. Detta skulle kunna göras genom att dra klipparen längs klippområdet. Alternativt kan koordinater plockas ut direkt ur någon karttjänst, vilka gräsklipparen sedan navigerar efter. Risken är att användarvänligheten sjunker då det kan bli lite krångligare än att gräva ned en kabel. Detta kan troligen undvikas genom att konstruera ett bra system för detta men det är inget som projektet har undersökt. Det finns idag aktörer som hjälper till vid installation av avgränsningskablar. Dessa aktörer kan även utbildas för att hjälpa till med installationen av en satellitnavigerad robotgräsklippare.

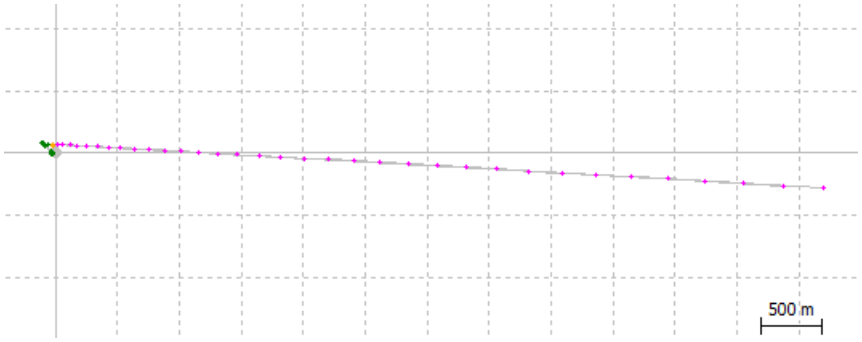
Ytterligare en nackdel mot den befintliga produkten är felsäkerheten. Referensprodukten har teknik för mätning av magnetfältet från avgränsningskabeln vilket gör det i princip omöjligt för den att färdas utanför området och om avgränsningskabeln skulle stängas av vid ett elfel stannar gräsklipparen. Den framtagna produkten har inga liknande säkerhetsfunktioner och om positionen av någon anledning skulle bli fel kommer den att köra på fel ställe och riskera att köra sönder rabatter, köra ned i poolen eller köra över till grannens trädgård. Detta är något som måste utvecklas vidare för att vara absolut säker

på att gräsklipparen inte kör på fel ställe.

Navigationssystemet kostade projektet 5 200 SEK. Detta pris avser inköpspriset projektet hade för styckinköp och vid en större produktion skulle detta pris troligen bli lägre. Budgeten passerades alltså med 200 SEK. Dock är den kostnaden fortfarande rimlig för navigationssystemet till en robotgräsklippare i det prissegment som valts. Priset lämnar rum för de andra delarnas kostnader, vilka behövs för att få en fullt fungerande produkt. Förutom de funktioner som behandlats i projektet har referensprodukten många funktioner som PIN-kodlås, vädersensorer, hög vädertålighet med mera. För att prototypen projektet tagit fram skulle vara intressant som konkurrent till referensprodukten måste även dessa delar behandlas och det är något som kommer lämnas till vidareutveckling.

5.3 Vidareutveckling

För vidareutveckling finns det några detaljer som inte har testats under projektet. Det kan vara intressant att skapa samma system fast med båda frekvenserna L1 och L2. Detta skulle innebära att *integer ambiguity* kan räknas ut snabbare vilket leder till att det går snabbare att få en fix-lösning. Detta är dock oförsvarbart dyrt, men för en robotgräsklippare i premiumsegmentet kan det kanske fungera i budgeten. I projektet användes inte GLONASS-systemets satelliter på grund av den stora osäkerheten i positionen det orsakade. Införandet av GLONASS-systemets satelliter skulle innebära ett högre antal satelliter synliga för mottagarna vilka positionen kan beräknas ifrån. Detta medför att positionen blir bättre där delar av himlen skyms, som nära hus eller andra hinder vilka kan skymma sikten. När senare Galileo, det europeiska GNSS-systemet, lanseras kan även detta implementeras i beräkningarna för att få tillgång till ett ytterligare antal satelliter.



Figur 23: Mätning där stort fel erhållits.

Några mätningar har fått väldigt stora fel, ett extremt exempel kan ses i figur 23. Detta har berott på att kommunikationen mellan basstationen och roboten har brutits eller att de inte har haft tillräckligt många gemensamma satelliter. Då erhålls lösningsstatusen SBAS som endast använder en mottagare för att beräkna positionen, vilket är mycket osäkrare. För att hantera detta kan programmet som körs på gräsklipparen stanna då dessa typer av lösningar erhålls och vänta på en bättre lösning. Programmet skulle även kunna hantera individuella mätningar som har ett flertal meter fel från tidigare mätningar och förkasta dessa.

Ytterligare vidareutvecklingspotentialer är de sensorer prototypen har tillgängliga. Alla

dessa har inte använts i programmeringen och många kan användas till att hjälpa gräsklipparen att navigera även när positionen från GNSS-mottagaren har förlorats. Exempel på sådana sensorer som finns tillgängliga är motorernas sensorer för avståndsmätning. Dessa kan användas på så sätt att produkten vet hur långt den har kvar till avgränsningen. Med detta kan den navigera och hela tiden hålla koll på hur långt den kör så att den inte passerar gränsen. Skulle den göra det så kan den helt enkelt vända. När sedan en position återfås kan gräsklipparen återgå till vanlig körning utifrån denna position.

Något som projektet inte har lagt arbete på, på grund av de avgränsningar som gjordes i början av projektet, är själva klipputrustningen. Vid vidareutveckling till en fullt fungerande gräsklippare skulle detta givetvis behöva beaktas.

Projektet har till prototypen för robotgräsklipparen använt två stycken Raspberry Pi-enheter. En användes till navigationssystemet samt en till körsystemet. Anledningen till att två enheter användes var att gruppen inte ville att körsystemet skulle ta beräkningskraft från navigationssystemet. Detta skulle kunna introducera ytterligare fel om navigationssystemet inte kan köras maximalt. Ännu en anledning till att gruppen valde att använda två stycken enkortsdatorer var att expansionskortet för GNSS-mottagaren använde många av GPIO-kontakterna, vilka behövdes till sensorer och motorkort för körsystemet. Detta motiverade valet av två enkortsdatorer ytterligare. Dock skulle en möjlig vidareutveckling vara att endast använda en enkortsdator till gräsklipparen och undersöka hur det skulle påverka noggrannheten. Då varken körprogrammet eller navigationsprogrammet visade sig kräva mycket prestanda skulle högst sannolikt inte noggrannheten påverkas märkbart. Problemet med brist på anslutningar går att kringgå genom att använda ett expansionskort för GPIO-kontakterna för att kunna ansluta fler sensorer och liknande till enkortsdatorn. Detta skulle sänka tillverkningskostnaden en aning.

5.4 Övriga tillämpningar

Noggrann positionsmätning kan ha många fler tillämpningar än hos gräsklippare och samma metodik och teknik som använts är applicerbara i alla möjliga autonoma applikationer som drönare, självkörande bilar eller båtar. Positioneringstekniken som har undersökts i detta projekt är relativt oberoende av vad den används till och det går med minimalt arbete att överföra till andra applikationer. De begränsningar som presenterats tidigare gäller för alla applikationer vilket utesluter inomhusanvändning och allt för komplexa miljöer där antennernas signaler störs ut. Systemets precision kan i nuläget inte helt garanteras och därför skulle säkerhet på eventuella applikationer behöva finnas i åtanke och kan lösas, exempelvis med hjälp av andra sensorer.

6 Slutsats

De resultat projektet fått visar på att satellitnavigeringstekniken RTK är en potentiell kandidat till robotgräsklippares navigationssystem. Den lösning som projektet har producerat fungerar bra för tillämpningar där god sikt över himlen finns. Detta inkluderar till exempel fotbollsplaner, golfbanor och liknande platser. Det finns dock många brister i tekniken vilket gör att det inte är möjligt att navigera nära hus, under täta träd och liknande miljöer på grund av behovet av en mycket god sikt över himlen. Tekniken lämpar sig därför inte för användning i en vanlig villaträdgård. Om GLONASS-systemets satelliter tas med i beräkningen kan fler satelliter fås in vilket gör att navigering nära hus borde bli bättre och detta är något som måste undersökas närmare. Noggrannheten för tekniken med dessa billiga komponenter är vid rörelse under fyra centimeter med en säkerhet på 96 % och under två centimeter med en säkerhet på 72 %. Detta motsvarar det mål som sattes för noggrannheten. Den framtagna produkten klarar även av att navigera längs en rutt, samt visar på grundläggande områdesbegränsning. Den klarar även simpel hinderdetektion samt att navigera runt dessa hinder. Något som dock talar emot tekniken är svårigheten att med 100 % säkerhet garantera att robotgräsklipparen, vid ett fel i navigationssystemet, inte åker utanför området. Detta är enklare med en avgränsningskabel då denna teknik i princip är helt felsäker när det kommer till detta. Möjligheten finns att åtgärda denna osäkerhet genom att använda fler sensorer för navigering och ett mer intelligent körprogram.

Referenser

[1] Sandberg D. *Robotens och andra gräsklipparmaskiners utveckling och framtida potential i hemträdgården [internet].* Alnarp: Trädgårdsingenjörsprogrammet: odling; 2013 [citerad 2016-03-15]. Hämtad från:
http://stud.epsilon.slu.se/6103/1/sandberg_d_130912.pdf

[2] Kjellín T. *Nordiska trädgårdar [internet]* . Stockholm: Nordiska trädgårdar; [citerad 2016-03-20]. Hämtad från:
http://www.nordiskatradgardar.se/press/pressmeddelanden/utstallare/7048

[3] Husqvarna AB. *Årsredovisning 2015 [internet].* Jönköping: Husqvarna AB; 2015 [citerad 2016-03-10]. Hämtad från:
http://www.husqvarnagroup.com/afw/files/press/husqvarna/201603155526-1.pdf

[4] Sal Vaglica. *Does a Robotic Lawn Mower Really Cut It? [internet].* New York: The Wall Street Journal; 2016 [citerad 2016-05-15]. Hämtad från:
http://www.wsj.com/articles/does-a-robotic-lawn-mower-really-cut-it-1460488911

[5] Hall R. *7 Small Robotic Mowers for Lawns [internet].* Turf Magazine; 2016 [citerad 2016-03-20]. Hämtad från:
http://www.turfmagazine.com/technology/7-small-robotic-mowers-lawns/

[6] Framtida Tillväxt. *Robotgräsklippare utan slinga från iRobot [internet].* Framtida Tillväxt; 2015 [citerad 2016-04-02]. Hämtad från:
http://framtidatillvaxt.se/robotgrasklippare-utan-slinga-fran-irobot/

[7] De Chant T. *The great (big) American lawn [internet].* Per Square Mile; 2011 [citerad 2016-02-20]. Hämtad från:
http://persquaremile.com/2011/04/08/the-great-big-american-lawn/

[8] Ahrenberg M & Olofsson A. *En noggranhetsjämförelse mellan Nätverks-RTK och Nätverks-DGPS [internet].* Gävle: Lantmäteriet; 2005. LMV-rapport; 2005:3. [citerad 2015-03-03]. Hämtad från:
https://www.lantmateriet.se/globalassets/kartor-och-geografisk-information/gps-och-matning/geodesi/rapporter_publikationer/rapporter/lmv-rapport_2005_3_exjobb_ahrenberg_olofsson.pdf

[9] *Centimeter-Level RTK Accuracy More and More Available — for Less and Less[Internet]* [Citerad 2016-05-15].
http://gpsworld.com/centimeter-level-rtk-accuracy-more-and-more-available-for-less-and-less/

[10] Carmelo D.M, Muscato G, Poncelet M. *A Simulation Environment for an Augmented GLobal Navigation Satellite System Assisted Autonomous Robotic Lawn-Mower [internet].* Catania: Springer Science + Business Media B.V; 2012 [citerad 2016-03-25]. Hämtad från:
http://link.springer.com/article/10.1007/s10846-012-9770-x

[11] Smith D.A, Chang J, Blanchard E.J. *An Outdoor High-Accuracy Local Positioning System for an Autonomous Robotic Golf Greens Mower [internet].* Saint Paul: International Conference on Robotics and Automation; 2012 [citerad 2016-03-20]. Hämtad från
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6224990&tag=1

[12] Cong M, Fang B. *Multisensor Fusion and Navigation for Robot Mower [internet].* Dalian: International Conference on Robotics and Automation; 2007 [citerad 2016-03-22]. Hämtad från:
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4522198

[13] Codol J, Poncelet M, Monin A, Devy M; *Safety robotic lawnmower with precise and low-cost L1-only RTK-GPS positioning [internet].* NAV ON TIME + BELROBOTICS; [Citerad 2016-04-05]. Hämtad från:
http://pnavhe11.irccyn.ec-nantes.fr/material/session3/Codol_paper.pdf

[14] Dagens Nyheter. *Därför blir robotgräsklipparna osams [internet]* [Citerad 2016-05-13]
http://www.dn.se/ekonomi/darfor-blir-robotgrasklipparna-osams/

[15] Ambrogio Robot. *Ambrogio Line 200 Basic [internet].* Ambrogio robot. [Citerad 2016-03-28]. Hämtad från:
http://www.ambrogio.se/CM.php?PageID=133862

[16] Husqvarna AB. *Husqvarna Automower 105 [internet].* Husqvarna AB. [Citerad 2016-03-25]. Hämtad från:
http://www.husqvarna.com/se/products/robotic-mowers/automower-105/

[17] Honda Power Equipment. *Miimo robotgräsklippare [internet].* Honda Power Equipment. [Citerad 2016-04-15]. Hämtad från:
http://www.honda.se/lawn-and-garden/products/miimo-2015/overview.html

[18] Worx Landroid. *Worx Landroid L [internet].* Worx Landroid. [Citerad 2016-04-15]. Hämtad från:
https://www.worxlandroid.com/sv-SE/catalog/landroid-l-1500-wg792e.1

[19] Gardena. *Robotgräsklippare R50Li [internet].* Gardena. [Citerad 2016-04-15]. Hämtad från:
http://www.gardena.com/se/lawn-care/robotic-mower/robotgrasklippare-r50li/

[20] Husqvarna AB. *Husqvarna Automower 220 AC bäst i test! [internet].* Husqvarna AB; 2011 [Citerad 2016-04-15]. Hämtad från:
http://www.husqvarna.com/se/press-listing/husqvarna-automower-220-ac-bast-i-test!

[21] Isaksson B. *Allt du behöver veta om robotgräsklippare. [internet].* Expressen. 2015 April [citerad 2016-04-13]. Hämtad från:
http://www.expressen.se/leva-och-bo/tradgard/gras/allt-du-behover-veta-om-robotgrasklippare/

[22] Husqvarna AB. *Vanliga frågor om robotgräsklippare [internet].* Husqvarna AB. [Citerad 2016-04-03]. Hämtad från:
http://www.husqvarna.com/se/products/robotic-mowers/faq/

[23] Husqvarna AB; *Husqvarna Automower 450X [internet]*. Husqvarna AB. [Citerad 2016-04-15]. Hämtad från:
http://www.husqvarna.com/se/products/robotic-mowers/automower-450x/

[24] North Surveying. *From Stones to Satellites [internet]*. North Surveying. [Citerad 2016-04-15]. Hämtad från:
http://northsurveying.com/index.php/soporte/gnss-and-geodesy-concepts#chapter-1-from-stones-to-satellites

[25] North Surveying. *Navigation Satellites [internet]*. North Surveying [Citerad 2016-04-15]. Hämtad från:
http://northsurveying.com/index.php/soporte/gnss-and-geodesy-concepts#chapter-2-navigation-satellites

[26] Wikipedia. *GPS broadcast signal [internet]*. Wikipedia. [Uppdaterad 2016-05-01; Citerad 2016-04-03]. Hämtad från:
https://en.wikipedia.org/wiki/GPS_signals#/media/File:GPS_signal_modulation_scheme.svg
Bild distribuerad under CC BY-SA 3.0 http://creativecommons.org/licenses/by-sa/3.0/

[27] Thomas Hellström. *GPS-så funkar det! [internet]*. Umeå Universitet. [Citerad 2016-05-29]. Hämtad från: http://www8.cs.umu.se/kurser/TDBC26/HT05/lectures/MBVR_F16_GPSdel1_051010.pdf

[28] Trimble. *Code-Phase GPS vs. Carrier-Phase GPS [internet]*. Trimble. [Citerad 2016-04-03]. Hämtad från:
http://www.trimble.com/gps_tutorial/sub_phases.aspx

[29] Takasu T; *RTKLIB ver. 2.4.2 Manual [internet]*. RTKLIB. [Citerad 2016-03-20]. Hämtad från:
http://www.rtklib.com/prog/manual_2.4.2.pdf

[30] North Surveying. *Sources of Inaccuracy: The Problems [internet]*. North Surveying. [Citerad 2016-04-01]. Hämtad från:
http://northsurveying.com/index.php/soporte/gnss-and-geodesy-concepts#chapter-3-sources-of-inaccuracy-the-problems

[31] North Surveying. *Sources of Inaccuracy: The Cures [internet]*. North Surveying. [Citerad 2016-04-05]. Hämtad från:
http://northsurveying.com/index.php/soporte/gnss-and-geodesy-concepts#chapter-4-sources-of-inaccuracy-the-cures

[32] Wikipedia. *Differential GPS [internet]*. [Uppdaterad 2016-04-16, Citerad 2016-04-06]. Hämtad från:
https://en.wikipedia.org/wiki/Differential_GPS

[33] Videkull R. *Evaluate and develop high-performance GPS navigation using free GPS software [internet]* Göteborg: Chalmers tekniska högskola; 2015 [Citerad 2016-05-03]. Hämtad från:
http://publications.lib.chalmers.se/records/fulltext/218271/218271.pdf

[34] Takasu T, Yasuda A. *Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB [internet].* Tokyo: Tokyo University of Marine Science and Technology. [Citerad 2016-04-06]. Hämtad från:
http://gpspp.sakura.ne.jp/paper2005/isgps_2009_rklib.pdf

[35] EGNOS. *What is SBAS? [internet].* EGNOS. [Citerad 2016-03-05]. Hämtad från:
http://www.egnos-portal.eu/discover-egnos/about-egnos/what-sbas

[36] Fasching F. *RasPiGNSS [Internet].* [Uppdaterad 2016-03-12, Citerad [2016-02-28]. Hämtad från:
http://drfasching.com/products/gnss/raspignss.html

[37] Raspberry Pi foundation. *Raspberry Pi 2 model B [internet].* Raspberry Pi foundation. [Citerad 2016-03-03]. Hämtad från:
https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[38] Tallysman. *TW2410/TW2412 Magnetic Mount Antenna for GPS/GLONASS [internet].* [Citerad 2016-02-28]. Hämtad från:
http://www.tallysman.com/index.php/gnss/products/antennas-gpsglonass/tw2410-tw2412/

[39] Robot Electronics. *EMG30, mounting bracket and wheel specification [internet].* Robot Electronics. [Citerad 2016-03-24]. Hämtad från:
http://www.robot-electronics.co.uk/htm/emg30.htm

[40] Robot Electronics. *MD25 Technical Documentation [internet].* Robot Electronics. [Citerad 2016-03-24]. Hämtad från:
http://www.robot-electronics.co.uk/htm/md25tech.htm

[41] Robot Electronics. *SRF05 Technical Documentation [internet].* Robot Electronics. [Citerad 2016-03-24]. Hämtad från:
http://www.robot-electronics.co.uk/htm/srf05tech.htm

[42] Honeywell. *3-Axis Digital Compass IC HMC5883L [internet].* Plymout: Honeywell; 2013 [Citerad 2016-03-15]. Hämtad från:
https://www.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

[43] Aviation Formulary. *Course equation [internet]* [Citerad 2016-05-12]. Hämtad från:
http://williams.best.vwh.net/avform.htm#flat

[44] Johnsson F, Wallerström M. *En nätverks-RTK jämförelse mellan GPS och GPS/GLONASS* Gävle: Högskolan i Gävle; 2007 [Citerad 2016-04-21]. Hämtad från:
https://www.lantmateriet.se/globalassets/om-lantmateriet/diariet-och-arkivredovisning/rapporter/2006_2012/lmv-rapport_2007_1.pdf

[45] Github *GPS+GLONASS not working on Raspberry Pi 2 [internet]* 2015 [Citerad 2016-05-16]. Hämtad från:
https://github.com/tomojitakasu/RTKLIB/issues/118

[46] Takac F. *GLONASS interfrequency biases and ambiguity resolution* Mars 2009; [Citerad 2016-05-16]. Hämtad från:
http://www.insidegnss.com/auto/marapr09-gnss-solutions.pdf

A Utrustningslista

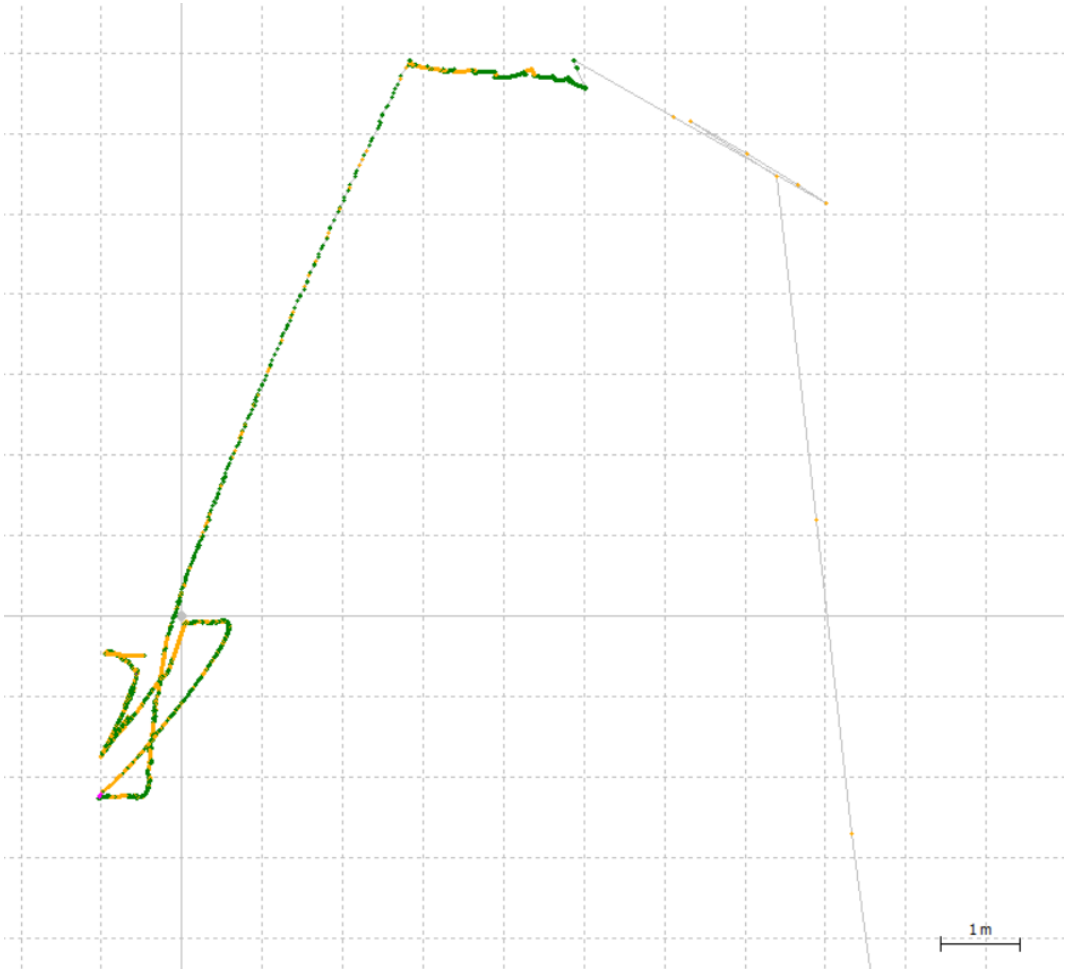
Materiallista:

- 2x *EMG30*-motorer.
- 2x 100mm hjul (tillhörande motorerna)
- 1x *MD25* motorkort
- 3x *Raspberry Pi Model 2 B* enkortsdatorer
- 1x 22-volt batteri
- 3x SD-kort
- 2x WiPi
- 2x *RasPiGNSS* expansionskort
- 2x *Tallysman TW-2410* antenner
- 1x *Honeywell HMC5883L* kompass
- 3x *Devantech SRF05* ultraljudssensorer
- 1x 22 till 13,6 volt transformatorer
- 2x 12 till 5 volt transformatorer

Mjukvarulista:

- RTKLIB
  - RTKRCV
  - STR2STR
  - RTKPLOT
  - RTKPOST
  - RTKCONV
- Google Earth
- TightVNC

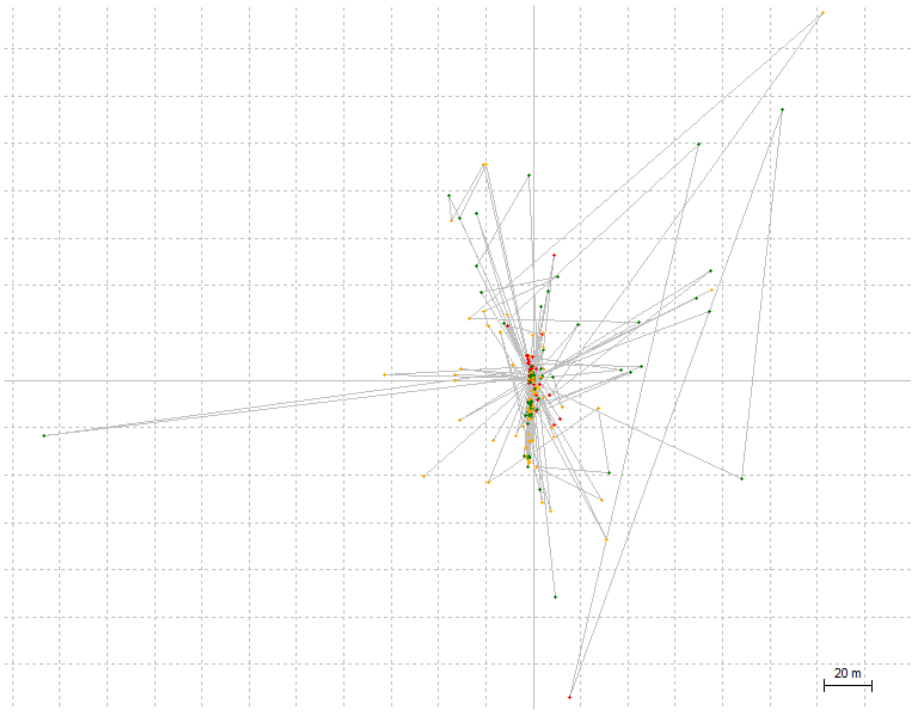
B Mätresultat ej presenterade i resultatavsnittet



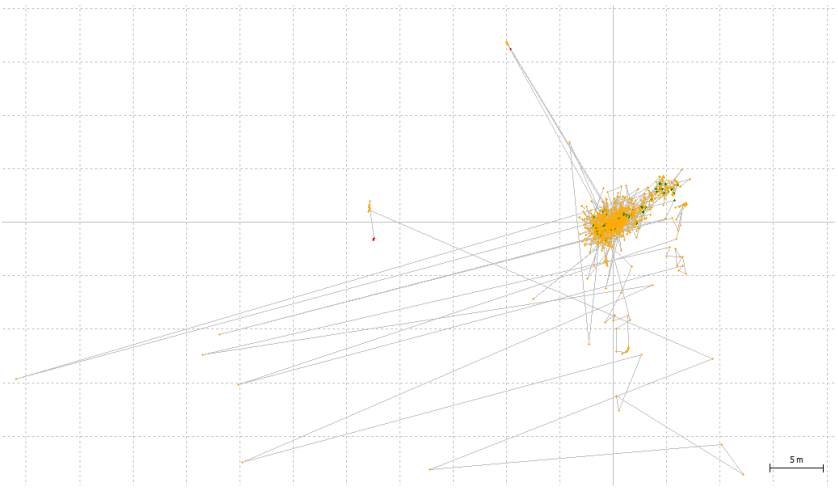
Figur 24: Drivande position vid statisk mätning med fel höjd angiven för basstation.

Tabell 3: Tabell över de uppmätta sträckorna och avvikelsen från tio meter i tiometersmätningen.

Test	Resultat [m]	Avvikelse 10m [cm]	Test	Resultat [m]	Avvikelse 10m [cm]
Test 1	9,9635	3,65	Test 14	10,0166	1,66
Test 2	9,9912	0,88	Test 15	9,987	1,30
Test 3	10,0037	0,37	Test 16	10,0177	1,77
Test 4	9,9922	0,78	Test 17	9,9965	0,35
Test 5	9,985	1,50	Test 18	10,0333	3,33
Test 6	10,1031	10,31	Test 19	10,013	1,30
Test 7	9,9968	0,32	Test 20	10,007	0,70
Test 8	10,0173	1,73	Test 21	10,0096	0,96
Test 9	10,0015	0,15	Test 22	10,0161	1,61
Test 10	10,0339	3,39	Test 23	10,0364	3,64
Test 11	10,0239	2,39	Test 24	10,0189	1,89
Test 12	10,0018	0,18	Test 25	10,0063	0,63
Test 13	10,0254	2,54			

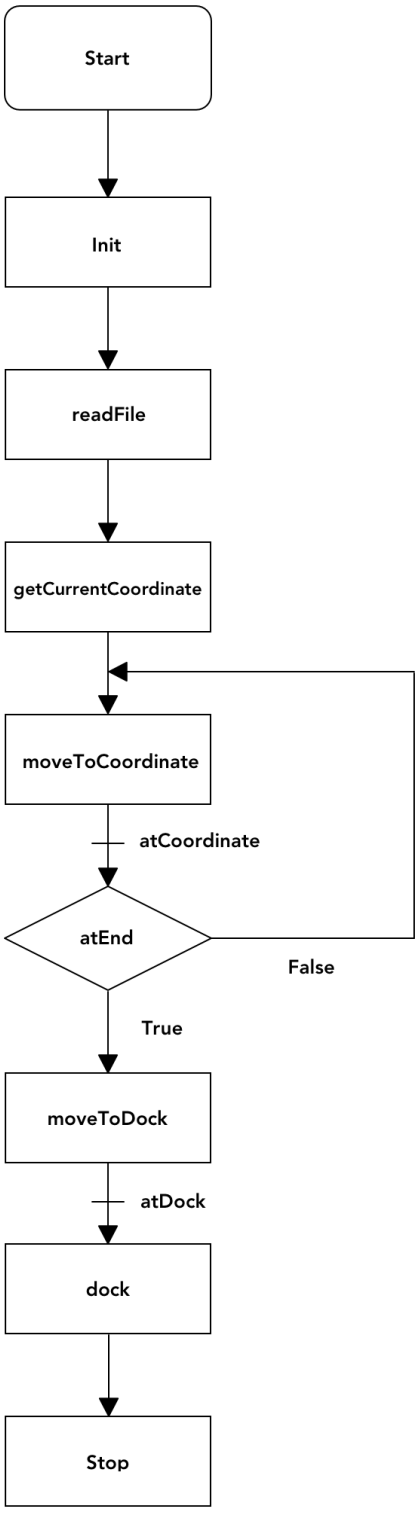


Figur 25: Statisk mätning med endast Glonass

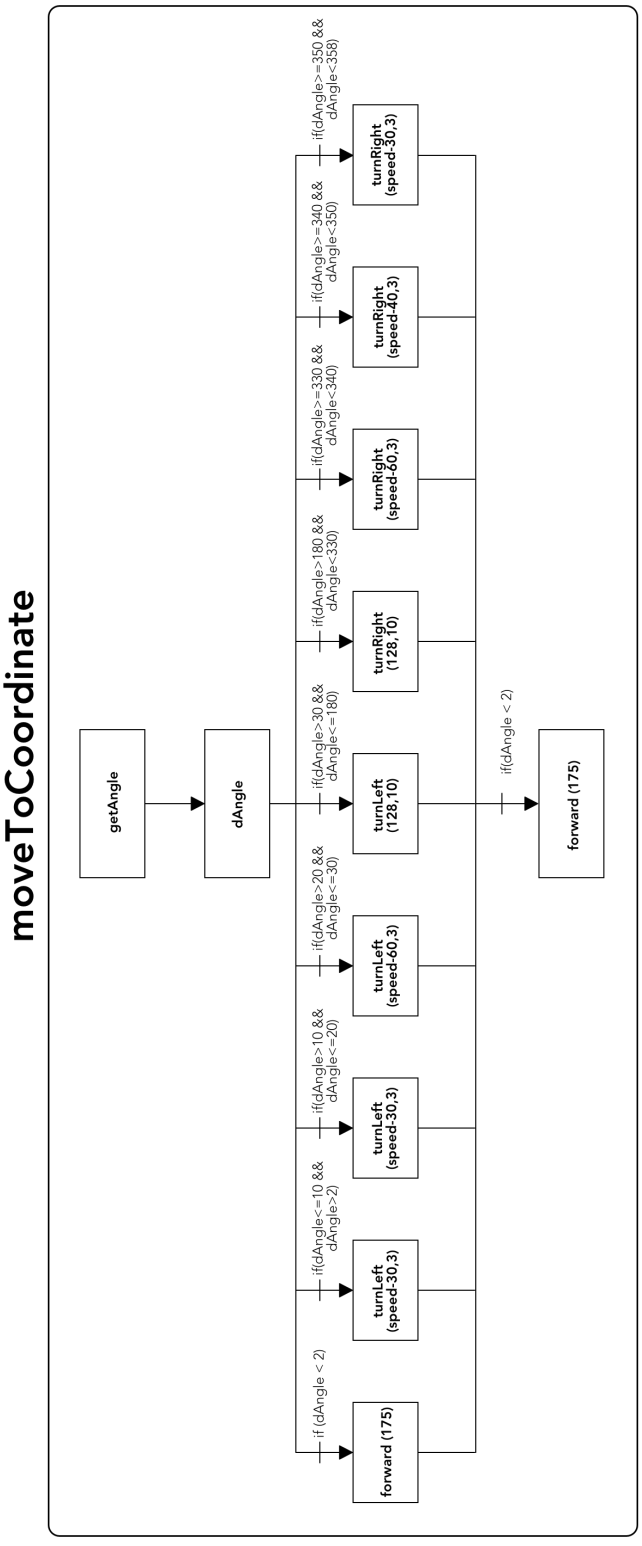


Figur 26: Statisk mätning med GPS och Glonass

C Flödesscheman



Figur 27: Flödesschema - Programmering.



Figur 28: Flödesschema - moveToCoordinate.

D Configfilen för RTKRCV

```
# rtkrcv options (2016/05/02 v.2.4.2)

console-passwd      =dgpsrover
console-timetype    =utc            # (0:gpst,1:utc,2:jst,3:tow)
console-soltype     =deg            # (0:dms,1:deg,2:xyz,3:enu,4:pyl)
console-solflag     =2              # (0:off,1:std+2:age/ratio/ns)

inpstr1-type        =serial         # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr1-path        =/ttyAMA0:230400:8:o:1:off
inpstr1-format      =nvs            # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:
    ↪ gw10,9:javad,10:nvs,15:sp3)

inpstr2-type        =tcpcli         # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr2-path        =129.16.63.92:43595
inpstr2-format      =rtcm3          # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:
    ↪ gw10,9:javad,10:nvs,15:sp3)
inpstr2-nmeareq     =off            # (0:off,1:latlon,2:single)
inpstr2-nmealat     =0              # (deg)
inpstr2-nmealon     =0              # (deg)

inpstr3-type        =off            # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,7:ntripcli,8:ftp,9:http)
inpstr3-path        =
inpstr3-format      =sp3            # (0:rtcm2,1:rtcm3,2:oem4,3:oem3,4:ubx,5:ss2,6:hemis,7:skytraq,8:
    ↪ gw10,9:javad,10:nvs,15:sp3)

outstr1-type        =tcpcli         # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr1-path        =169.254.134.101:43596
outstr1-format      =llh            # (0:llh,1:xyz,2:enu,3:nmea)

outstr2-type        =file           # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
outstr2-path        =Solution%Y%m%d%h%M.pos
outstr2-format      =llh            # (0:llh,1:xyz,2:enu,3:nmea)

logstr1-type        =file           # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr1-path        =LOGROVER%Y%m%d%h%M.log
logstr2-type        =file           # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr2-path        =LOGBAS%Y%m%d%h%M.log
logstr3-type        =off            # (0:off,1:serial,2:file,3:tcpsvr,4:tcpcli,6:ntripsvr)
logstr3-path        =

misc-svrcycle       =10             # (ms)
misc-timeout        =10000          # (ms)
misc-reconnect      =10000          # (ms)
misc-nmeacycle      =5000           # (ms)
misc-buffsize       =32768          # (bytes)
misc-navmsgsel      =all            # (0:all,1:rover,2:base,3:corr)
misc-proxyaddr      =
misc-fswapmargin    =30             # (s)
misc-startcmd       =
misc-stopcmd        =
file-cmdfile1       =
file-cmdfile2       =
file-cmdfile3       =

pos1-posmode        =kinematic      # (0:single,1:dgps,2:kinematic,3:static,4:movingbase,5:fixed,6:ppp-
    ↪ kine,7:ppp-static)
pos1-frequency      =l1             # (1:l1,2:l1+l2,3:l1+l2+l5)
pos1-soltype        =combined       # (0:forward,1:backward,2:combined)
pos1-elmask         =15             # (deg) 4421, 2410
pos1-snrmask_r      =on             # (0:off,1:on)
pos1-snrmask_b      =off            # (0:off,1:on)
# pos1-snrmask_L1 =25               # (dBHz) 4421
# pos1-snrmask_L1 =35               # (dBHz) 2410
pos1-snrmask_L1     =33,34,35,35,36,37,37,37,37   # (dBHz) 4421, 2410
pos1-snrmask_L2     =0,0,0,0,0,0,0,0,0            # (dBHz)
pos1-snrmask_L5     =0,0,0,0,0,0,0,0,0            # (dBHz)
pos1-dynamics       =on             # (0:off,1:on)
pos1-tidecorr       =on             # (0:off,1:on)
pos1-ionoopt        =sbas           # (0:off,1:brdc,2:sbas,3:dual-freq,4:est-stec,5:ionex-tec,6:qzs-
    ↪ brdc,7:qzs-lex,8:vtec_sf,9 :vtec_ef,10:gtec)
pos1-tropopt        =saas           # (0:off,1:saas,2:sbas,3:est-ztd,4:est-ztdgrad)
pos1-sateph         =brdc+sbas      # (0:brdc,1:precise,2:brdc+sbas,3:brdc+ssrapc,4:brdc+ssrcom)
pos1-posopt1        =on             # (0:off,1:on)
pos1-posopt2        =on             # (0:off,1:on)
pos1-posopt3        =on             # (0:off,1:on)
pos1-posopt4        =off            # (0:off,1:on)
pos1-posopt5        =on             # (0:off,1:on)
pos1-exclsats       =               # (prn ...)
pos1-navsys         =7              # (1:gps+2:sbas+4:glo+8:gal+16:qzs+32:comp) ??

pos2-armode         =fix-and-hold   # (0:off,1:continuous,2:instantaneous,3:fix-and-hold,4:ppp-ar)
pos2-gloarmode      =on             # (0:off,1:on,2:autocal)
pos2-arthres        =3
pos2-arlockcnt      =10
pos2-thresar2       =0.9999
pos2-thresar3       =0.2
pos2-lockcntfixamb  =10
pos2-fixcntholdamb  =10
pos2-arelmask       =15             # (deg)
pos2-arminfix       =10
pos2-elmaskhold     =15
pos2-aroutcnt       =5
pos2-maxage         =30
pos2-slipthres      =0.05
pos2-rejionno       =30
pos2-rejgdop        =30
```

```
pos2-niter          =1
pos2-baselen        =0
pos2-basesig        =0

out-solformat       =llh            # (0:llh,1:xyz,2:enu,3:nmea)
out-outhead         =on             # (0:off,1:on)
out-outopt          =off            # (0:off,1:on)
out-timesys         =utc            # (0:gpst,1:utc,2:jst)
out-timeform        =hms            # (0:tow,1:hms)
out-timendec        =1
out-degform         =deg            # (0:deg,1:dms)
out-fieldsep        =,
out-height          =geodetic       # (0:ellipsoidal,1:geodetic)
out-geoid           =internal       # (0:internal,1:egm96,2:egm08_2.5,3:egm08_1,4:gsi2000)
out-solstatic       =all            # (0:all,1:single)
out-nmeaintv1       =1              # (s)
out-nmeaintv2       =1              # (s)
out-outstat         =off            # (0:off,1:state,2:residual)

stats-errratio      =100
stats-errphase      =0.003          # (m)
stats-errphaseel    =0.003          # (m)
stats-errphasebl    =0              # (m/10km)
stats-errdoppler    =10             # (Hz)
stats-stdbias       =30             # (m)
stats-stdiono       =0.03           # (m)
stats-stdtrop       =0.3            # (m)
stats-prnaccelh     =1              # (m/s^2)
stats-prnaccelv     =0.1            # (m/s^2)
stats-prnbias       =0.0001         # (m)
stats-prniono       =0.001          # (m)
stats-prntrop       =0.0001         # (m)
stats-clkstab       =5e-12          # (s/s)

ant1-postype        =llh            # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm)
ant1-pos1           =90             # (deg|m)
ant1-pos2           =0              # (deg|m)
ant1-pos3           =-6335367.6285  # (m|m)
ant1-anttype        =
ant1-antdele        =0              # (m|m)
ant1-antdeln        =0              # (m|m)
ant1-antdelu        =0              # (m|m)
ant2-postype        =llh            # (0:llh,1:xyz,2:single,3:posfile,4:rinexhead,5:rtcm)
ant2-pos1           =57.689374498   # (deg|m)
ant2-pos2           =11.978251212   # (deg|m)
ant2-pos3           =114.330181482
ant2-anttype        =
ant2-antdele        =0              # (m|m)
ant2-antdeln        =0              # (m|m)
ant2-antdelu        =               # (m|m)
misc-timeinterp     =off            # (0:off,1:on)
misc-sbasatsel      =0              # (0:all)
misc-rnxopt1        =
misc-rnxopt2        =

file-satantfile     =/home/pi/rtklib/RTKLIB-master/data/ngs_abs.pcv
file-rcvantfile     =/home/pi/rtklib/RTKLIB-master/data/igs05.atx
file-staposfile     =
file-geoidfile      =
file-ionofile       =
file-dcbfile        =/home/pi/rtklib/RTKLIB-master/data/P1C1_ALL.DCB
file-eopfile        =
file-blqfile        =
file-tempdir        =
file-geexefile      =
file-solstatfile    =
file-tracefile      =
```

Kod 1: Konfigurationsfil för RTKRCV

E Programkod för styrning av prototyp

Nedan presenteras delar av programkoden vilken användes för styrning av roboten

```
//==={Includes}==
#include <stdio.h>
#include <stdlib.h>
#include <linux/i2c-dev.h>
#include <fcntl.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <time.h>
#include <math.h>
#include <errno.h>
#include <sys/time.h>
#include <wiringPi.h>
#include <signal.h>
//==={Globala variabler}==
int fdMotor, fdCompass;             // File descriptors som används för motorkort och kompass.
int speed=200;                      // Variabel som styr meed vilken hastighet roboten kör.
char *fileName = "/dev/i2c-1";      // I2C porten som vi använder
int  motoraddress = 0x58;           // Addressen för motorkortet MD25
int compassaddress = 0x1E;          // Addressen för kompassen
unsigned char buf[16];              // Buffer som används för att skriva och läsa på I2Cbussen
#define TIMEOUT 999                 // Används för ultraljudssensorn
double firstCoordinate[2];          // Används då roboten kör med
double currentCoordinate[2];        // De nuvarande koordinaterna
int size;                           // Används för att läsa in coordinates.txt
//==={TCP-server}==
int sockfd, newsockfd;              // Variabler för tcp, socket
int portno= 43596;                  // Portnummer för TCP
int clientlength;                   // Längden på klienten som ansluter
char buffer[256];                   // Buffer för tcp. Denna fylls på med koordinater från RTKRCV
struct sockaddr_in serveraddress, clientaddress;   // Variabler för tcp
//==={Länkad lista för koordinater}==
struct point *waypoints;            // Används för adress till början av länkade listan
struct point{                       // Länkad lista
   double latitude;                 // Koordinat
   double longitude;                // Koordinat
   struct point *next;              // Pekare på nästa point-struct
};
//==={Funktioner}==
//==={MotorStyrning Input: hastighet 128=stilla, 255 = full fram, 0 = full bak}==
void initMotor(void);               // Initialiserar motorkortet MD25
void forward(int);                  // Kör framåt med hastigheten på input
void backup(int);                   // Backar med hastigheten på input
void turnRight(int, int);           // Kör med hastighet från input1, svänger med input2
void turnLeft(int, int);            // Kör med hastighet från input1, svänger med input2
void stop(void);                    // Stannar båda motorerna
void eSleep(double);                // Skicka in tid (sek) för sleep, motverkar att motorkortet timeoutar
long readEncoderValues(void);       // Används för att läsa Encoders från motorer
void resetEncoders(void);           // Används för att nollställa Encoders
//==={Sensorer}==
void initCompass(void);             // Initialiserar kompassen
void writeToCompass(int, int);      // Startar mätning på kompass
double readCompass(void);           // Läs från kompass
int waitForRead(int, int, int);     // Används av ultraread(), väntar och läser av GPIO pin
double ultraRead (int, int);        // Läser avstånd till objekt med ultraljudssensor
//==={TCP}==
void connectTCP(void);              // Öppnar TCP för att ta emot positionsdata
void disconnectTCP(void);           // Stänger TCP
void getCurrentCoordinate(void);    // Läser av buffern sätter currentCoordinate till dessa koordinater
void error(char *);                 // Används för att skicka meddelande ifall något fel skulle inträffa
//==={Styrning och navigering}==
int allowed(void);                  // Används för att kolla om roboten är inom ett visst område
int atCoordinate(void);             // Används för att kolla om vi är vid koordinaten vi vill till
double getAngle(void);              // Beräknar vinkel mellan waypoint och koordinat
struct point *readFile(void);       // Skapar en linked list och returnerar adressen till första struct
int readFileSize();                 // Läser antal rader på fil med waypoints.

void obstacleTurn();                // Åker runt hinder
void moveToCoordinate(void);        // Kör till koordinaten som finns i structen
void goWaypoints(void);             // Kör mellan alla inlästa koordinater och upprepar sedan detta.
void goArea(void);                  // Kör inom ett område som sätt som koordinaterna i början.
```

Kod 2: Alla variabler och metoder som används

```
//========{Kollar om vi är inom ett visst område runt första koordinaten}==
int allowed(void){
   double cLat = firstCoordinate[0];          // firstCoordinates blir satta inom goArea och goWaypoints
   double cLon = firstCoordinate[1];          // Dessa blir vår mittpunkt
   double allowedOffSet = 0.001;              // Denna bestämmer hur långt ifrån mittpunkten vi får vara

   /*Jämför nuvarande position med den första och offseten   */
   if(fabs(currentCoordinate[0]-cLat)>allowedOffSet||fabs(currentCoordinate[1]-cLon)>allowedOffSet){
      printf("Not Allowed!\n");
      return 0;
   }
   printf("Allowed!\n");
   return 1;
}
```

```
15
16 //==={Kollar om vi är inom ett visst område vid målkoordinaterna}==
17 int atCoordinate(void){
18    double cLat = waypoints->latitude;
19    double cLon = waypoints->longitude;
20    double allowedOffSet = 0.000003;        //Denna bestämmer hur nära målkoordinaten vi behöver vara
21
22    /*Jämför nuvarande position med den önskade och offseten. */
23    if(fabs(currentCoordinate[0]-cLat)>allowedOffSet||fabs(currentCoordinate[1]-cLon)>allowedOffSet){
24       printf("Not at Coordinate!\n");
25       return 0;
26    }
27    printf("At Coordinate!\n");
28    return 1;
29 }
```

Kod 3: Avgör och bestämmer om vi är inom det tillåtna och önskade området

```
1  //==={Kör inom ett område. Storlek bestämt inuti allowed}==
2  void goArea(void){
3      getCurrentCoordinate();
4      firstCoordinate[0]=currentCoordinate[0];// Här sätts våra koordinater som används inuti Allowed()
5      firstCoordinate[1]=currentCoordinate[1];// Här sätts våra koordinater som används inuti Allowed()
6     while(firstCoordinate[0] == 0 || firstCoordinate[1] == 0){
7        printf("Coordinate error, restarting\n");
8        disconnectTCP();                       //Ifall ingen solution fås så kan tcp sluta fungera ibland.
9        connectTCP();                          //Deta är för att hindra att första koordinat blir (0,0)
10       getCurrentCoordinate();
11         firstCoordinate[0]=currentCoordinate[0];
12         firstCoordinate[1]=currentCoordinate[1];
13       }
14      while(1){
15       getCurrentCoordinate();
16       if(currentCoordinate[0] == 0 || currentCoordinate[1] == 0){
17          stop();
18          disconnectTCP();
19          connectTCP();
20       }
21       else{
22          if (sensorFront(300))
23          {
24             obstacleTurn();
25          }
26          else{
27             while(allowed()){                  // Om vi befinner oss innanför området
28                forward(speed);                 // så kör vi endast framåt
29                }
30             stop();                            //Detta händer då roboten befinner sig utanför området.
31             usleep(700000);
32             backup(70);                        //Den backar och svänger sedan.
33             usleep(500000);
34             turnRight(128,20);
35             usleep(500000);
36             turnRight(128,20);
37             usleep(500000);
38             stop();
39             }
40    }
41 }
42 }
```

Kod 4: Kör inom ett område bestämt inuti allowed

```
1  //=======Svänger och kör till coordinaterna i waypoints beroende på hur mycket skillnaden mellan
       ↪ kompassen och getAngle().======
2  void moveToCoordinate(void){
3     double dAngle;
4     double compass=readCompass();
5     dAngle=compass-getAngle();      //Skillnaden mellan kompass och getAngle()
6     getCurrentCoordinate();
7     if (!sensorFront(300))
8     {
9        obstacleTurn();
10    }
11    else{
12       if(fabs(dAngle)<=2)          //Beroende på hur mycket skillnaden är så svänger den olika mycket
13          forward(speed);           // Kan konfigureras mycket
14       if(dAngle<0)
15          dAngle=dAngle+360;
16       if(dAngle<=10 && dAngle>2){
17          turnLeft(speed-30,3);
18          }
19       else if(dAngle>=350 && dAngle<358){
20          turnRight(speed-30,3);
21          }
22       else if(dAngle>10 && dAngle<=20){
23          turnLeft(speed-30,3);
24          }
25       else if(dAngle>=340 && dAngle<350){
26          turnRight(speed-40,3);
27          }
28       else if(dAngle>20 && dAngle<=30){
29          turnLeft(speed-60,3);
30          }
31       else if(dAngle>=330 && dAngle<340){
```

```
32            turnRight(speed-60,3);
33            }
34        else if(dAngle>30 && dAngle<=180){
35            turnLeft(128,10);
36            }
37        else if(dAngle>180 && dAngle<330){
38            turnRight(128,10);
39            }
40    }
41 }
```

Kod 5: Kör till den önskade koordinaten

```
1  //=={Kör mellan koordinater inlästa från fil med koordinater}==
2  void goWaypoints(void){
3     getCurrentCoordinate();
4     firstCoordinate[0]=currentCoordinate[0];   // Här sätts våra koordinater som används inuti Allowed()
5     firstCoordinate[1]=currentCoordinate[1];   // Här sätts våra koordinater som används inuti Allowed()
6     while(allowed){                            // Detta är ett yttre område som ligger runt punkterna.
7        getCurrentCoordinate();                 // Ifall den kommer utanför kommer den stanna
8        if(currentCoordinate[0] == 0 || currentCoordinate[1] == 0){
9           stop();
10          disconnectTCP();
11          connectTCP();
12          }
13       stop();
14       eSleep(1);
15       while(!atCoordinate()){                //Kör så länge den ej är vid önskad koordinat
16          }
17       printf("At Coordinate, Changing waypoint\n");
18       stop();
19       waypoints=waypoints->next;             //När inom området så byt till nästa koordinat.
20       }
21       stop();
22       printf("OUT OF AREA! STOPPING!\n");    //Utanför det yttre området så stannar den
23 }
```

Kod 6: Kör mellan alla waypoints

```
1  //=={Tar sig runt hinder framför sig}==
2  void obstacleTurn(){
3     getCurrentCoordinate();
4     int direction=0;                           // För att veta vilket håll vi ska svänga
5     if(sensorLeft(300)){                       // Ifall inget till vänster, sväng vänster
6        turnLeft(128,12);
7        eSleep(4);
8        stop();
9        direction=1;
10       usleep(100000);
11       }
12    else if(sensorRight(300)){                 // Ifall inget till höger, sväng höger
13       turnRight(128,12);
14       eSleep(4);
15       stop();
16       direction=2;
17       usleep(100000);
18       }
19    if(direction=1){
20       while(!sensorRight(400)){
21          printf("Going forward then right");
22          forward(160);
23          usleep(100000);
24          }
25       sleep(1);                               //Kör lite extra för att komma förbi
26       stop();
27       turnRight(128,10);
28       eSleep(4);
29       stop();
30
31       }
32    else if(direction=2){
33       while(sensorLeft(400)){
34          printf("Going forward then left");
35          forward(160);
36          usleep(100000);
37          }
38       sleep(1);                               //Kör lite extra för att komma förbi
39       stop();
40       turnLeft(128,10);
41       eSleep(4);
42       stop();
43       }
44 }
```

Kod 7: Åker runt hinder som främre sensorn upptäcker