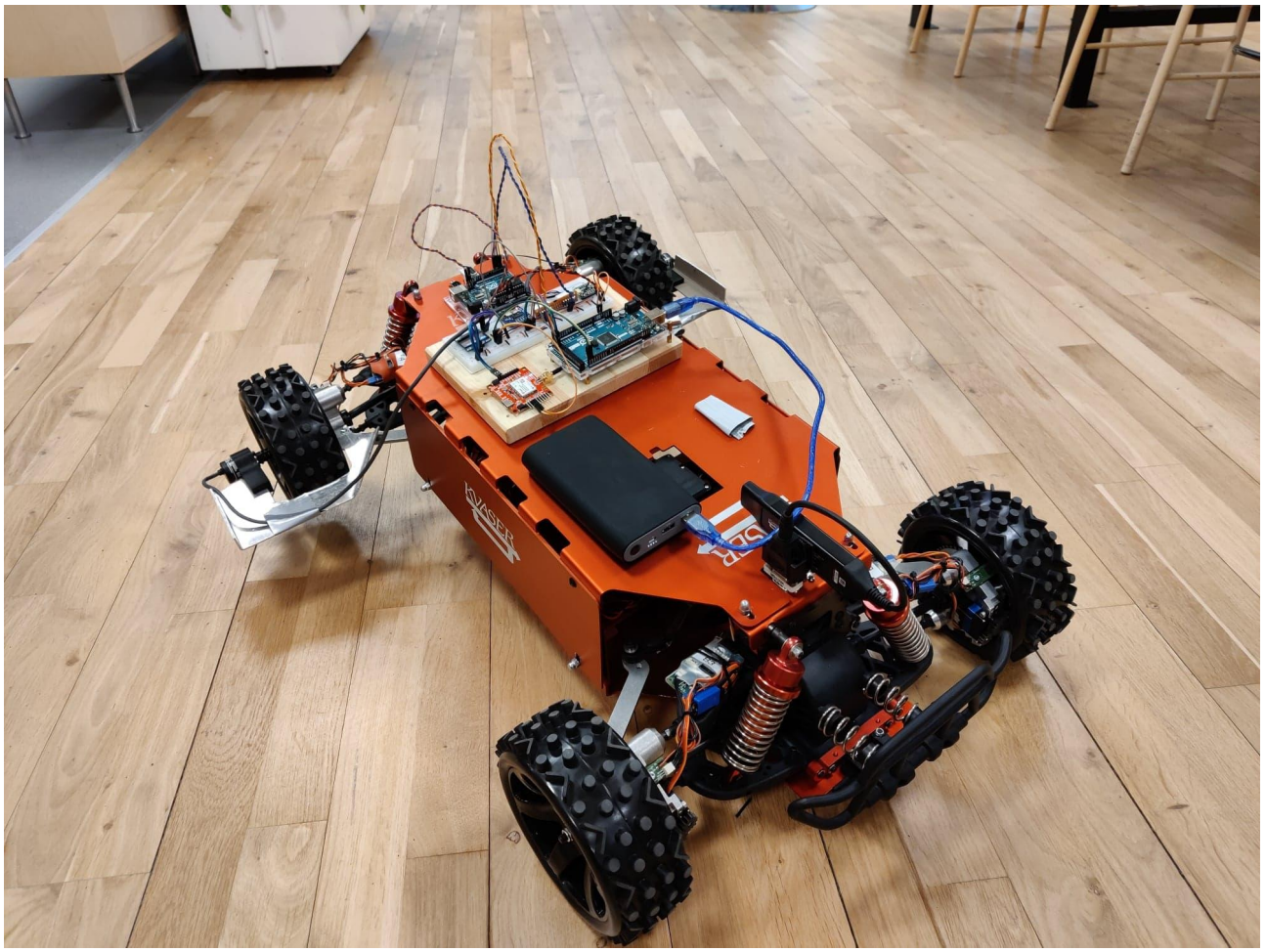
CHALMERS



Förbättrad positionering med hjälp av GPS och Sensor Fusion

Examensarbete inom Data- och Informationsteknik

Marcus Alvefelt
Hampus Strömberg

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

EXAMENSARBETE

Förbättrad positionering med hjälp av GPS och Sensor Fusion

Undersökning av precision vid GPS positionering och hur alternativa positioneringssystem med hjälp av Sensor Fusion kan stötta upp vid GPS signalens bortfall

Marcus Alvefelt
Hampus Strömberg

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET

Göteborg 2021

Förbättrad positionering med hjälp av GPS och Sensor Fusion

Undersökning av precision vid GPS positionering och hur alternativa positioneringssystem med hjälp av Sensor Fusion kan stötta upp vid GPS signalens bortfall

Marcus Alvefelt
Hampus Strömberg

Examinator: Johan Duregård

Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola
412 96 Göteborg
Telefon: 031-772 1000

Omslag:
Farkost varpå arbetet utförts på. Tillhandahållen av Kvaser. I Farkosten finns ett CANbus-system som kommunicerar med radiokontrollern via Kvasers Air Bridge. På farkosten kan två stycken mikrokontrollers, en GPS modul, en IMU, skärm samt SD-kortläsare ses.
Författarna har tagit bilden.

Institutionen för Data- och Informationsteknik
Göteborg 2021

SAMMANFATTNING

De GPS system som finns idag är välfungerande positioneringssystem men de har dessvärre något dålig precision. Detta arbete utforskar möjligheterna till alternativa positioneringsmetoder för att komplementera och förbättra positionsbestämning. Främst då GPS-signalen fallerar eller då den studsar i exempelvis stadsmiljöer. Fler positionsbilder baserade på olika data krävs för att skapa en så precis och pålitlig position som möjligt, speciellt när positionen ska användas till att köra ett fordon autonomt. För att ett positionssystem ska kunna litas på i en autonom bil krävs en noggrannhet på centimeter-, och helst millimeternivå. Arbetet undersöker aktuell problembild på marknaden, upprättar en relevant frågeställning utefter denna problembild och gör ett försök i att ta fram ett proof of concept i form av en radiostyrd bil med ytterligare positioneringssystem utöver GPS positionering. Utöver positioneringsfrågan går arbetet också in på Odometri, Dead Reckoning och Ackermann-styrning.

En viktig avgränsning som görs är att ingen autonomitet eller kursmodifiering utförs i arbetet. Endast en mer precis position eftersöks.

Rapporten redogör för ett arbete utfört hos Diadrom i Göteborg under vårterminen 2021 under institutionen för Data och Informationsteknik vid Chalmers Tekniska Högskola.

Nyckelord: GPS, GNSS, Galileo, Glonass, Position, Positionering, Odometri, Dead Reckoning, Ackermann, Sensor Fusion, Autonom, Autonomitet, RC, MATLAB, CAN

ABSTRACT

GPS positioning today is of a somewhat bad precision. The use of GPS has shifted from when it was invented and modern needs have higher demands for better positioning. This project is exploring the possibilities for alternative systems for positioning. GPS systems today also have a characteristic of unreliability. To remedy this there is a need for alternative positioning systems that can support the GPS system - especially when it's signal is unavailable, and it's a must if these types of positioning systems were to be implemented into autonomous cars. Autonomous cars require a position on a centi- or even a millimeter level. The report also briefly touches on Odometry, Dead Reckoning and Ackermann steering.

An important demarcation that is made is that no automacy is implemented. Only a better position is sought after.

The report accounts for a project which was carried out together with Diadrom in Gothenburg during the spring semester of 2021 for The Department of Computer Science and Engineering at Chalmers University of Technology.

FÖRORD

Examensarbetet, som denna rapport är ett resultat av har utförts på Diadrom Holding AB och under Chalmers institution för Data- och Informationsteknik. Det utfördes under en global pandemi som hade kunnat förhindra hela eller större delar av vårt arbete. Vi vill med detta stycke tacka bland andra Diadrom för att ha möjliggjort ett examensarbete trots omständigheterna. Vi tackar också Kvaser för att ha tillgodosett oss med en utvecklingsplattform för arbetet samt Hashem Hashem för välbehövlig hjälp att få plattformen att fungera som tänkt. Ett speciellt tack går ut även ut till Diadroms Henrik Fagrell som varit en professionell mentor vid avstämningar och ett ovärderligt bollplank i industrirelaterade frågor. Vi vill även tacka Sakib Sistek som varit vår handledare från Chalmers sida. Hans orubbliga humör och livssyn har varit en fröjd att få ta del av under arbetets gång. Han har dessutom varit en proffsig ledsagare och hans kunskap inom området har varit en riktig tillgång. Utöver dessa två eldsjälar till handledare vill vi även sträcka ut ett hjärtfyllt tack till Lindholmens Makerspace för utlåning av sina lokaler, utrustning och kunskap. Slutligen vill vi även tacka familjer och vänner för deras stöd under hela vår utbildning och hoppas att denna rapport kan ge en glimt in i vad dessa år inneburit för vår professionella utveckling.

INNEHÅLLSFÖRTECKNING

TERMINOLOGI

Nedan följer förberedande terminologi som används i arbetet

- Positionssystem - Ett system som producerar en position i ett koordinatsystem. Koordinatsystemen kan vara i exempelvis Longitud och Latitud eller i X och Y
- Positionsbild - Den position som framgår av sensorerna.
- Sensorbild - Tolkningen av de uppmätta sensorvärdena.
- Startvinkel - Den vinkel som bilen pekar i vid start av körning.
- Sensor Fusion - Användning av mer än en givare tillsammans för att få ett bättre resultat.
- Farkost, Plattform, Bil - Hänvisar till den sensorförsedda radiostyrda bil som arbetet utförts på.
- Hjulglidning - Försvenskat ord från engelskans “wheel slip”.
- GPS - Global Positioning System.
- Galileo - Europas GPS System.
- GLONASS - Rysslands GPS System.
- GNSS - Global Navigation Satellite System.
- DOF - Degrees of freedom.
- DOP - Dilution of precision.
- IMU - Inertial Measurement Unit

FIGURER OCH TABELLER

1 INLEDNING

Global Positioning System (GPS) är ett positionssystem som togs fram för navigering och positionering. Systemet bygger på att en signaltid mäts mellan en satellit med känd position och en mottagare. Genom att mäta tiden från flera satelliter kan mottagarens position trianguleras [1]. GPS är utvecklat av USA, Europa har utvecklat ett eget system som heter Galileo och Ryssland har ett system som heter GLONASS. GNSS (Global Navigation Satellite System) har framtagits för att kunna utnyttja samtliga satelliter och därför ge en bättre positionsbild [2]. Fortsättningsvis kommer namnet "GPS" att användas som ett generellt samlingsnamn för samtliga positionssystem om inget annat anges.

I autonoma fordon används GPS främst för navigering längs kända kartor. Fordonsindustrin jobbar idag med att förbättra positionen från GPS för att bli tillräckligt bra att navigera med vid anvädning av autonoma fordon. Exakt hur detta ska ske är ännu ganska outforskade vatten [3].

1.1 Bakgrund

I dagsläget jobbar många företag med att göra autonoma bilar säkra, användarvänliga och tillförlitliga. En viktig del i ett autonomt fordon är att med så stor säkerhet som möjligt kunna bestämma dess position. Noggrannheten i dagens GPS är inte tillräckligt hög för att kunna användas för att få en tillräckligt exakt position och framföra ett självkörande fordon [3].

Detta projekt kommer att utforska möjligheterna att kunna bestämma en noggrannare position med hjälp av andra sensorer genom så kallad Sensor Fusion.

1.2 Mål

De mål som är uppsatta för projektet redovisas nedan:

- Att kartlägga problem med GPS i tillämpningar av autonomitet.
- Att ta fram en alternativ positionsbild med hjälp av ett sensorsystem helt fristående från GPS.
    - Detta för att utforska möjligheterna att undgå de brister och problem som medföljer med positionering med endast GPS.
- Att denna positionsbild skall hålla en bättre precision än den producerad av den använda GPS-enheten.
    - Bättre än den utlovade positionsstandarden som finns för GPS, ca $\pm$ 1 m.

1.3 Syfte

Rapporten är av undersökande typ och avhandlar efterforskning av problembild på marknaden. Rapporten avhandlar även framtagande av en plattform vars position ska kunna uppmätas med hjälp av en egen simuleringsmodell. På sikt kan arbetet komma att användas som incitament för efterforskning angående applicering på autonoma fordon.

1.4 Precisering av frågeställning

Utifrån de mål som är uppsatta för arbetet och för att få ännu mer tydlighet vad för frågor som skall besvaras så följer nedan en punktlista:

- Kan GPS positionering stöttas upp av externa system?
- Hur kan mätbara data visa att positionen förbättrats?
- Vilka förbättringsmöjligheter finns det för positionering med avseende på autonomitet?
- Hur kan noggrannheten i ett fordons position förbättras med hjälp av Sensor Fusion?

1.5 Avgränsningar

En GPS position består av Longitud och Latitud. Dessa koordinataxlar befinner sig i ett sfäriskt koordinatsystem, medans de koordinatsystem som arbetet använder utgår ifrån ett euklidiskt koordinatsystem. Det går att projicera ett euklidiskt koordinatsystem på ett sfäriskt, men detta görs inte i detta arbete.

Arbetet behandlar ingen förbättring av själva GPS positionen. Endast ett substitut då GPS ej är tillförlitlig. Ingen korrektion av farkostens kurs kommer implementeras. Det kommer heller inte implementeras någon form av autonomitet i farkosten. Ingen ingående beskrivning av hur CANbus-systemen upprättades görs. Rapporten nämner även LIDAR men tar ej upp hur detta positionssystem fungerar.

2 TEORETISK REFERENSRAM

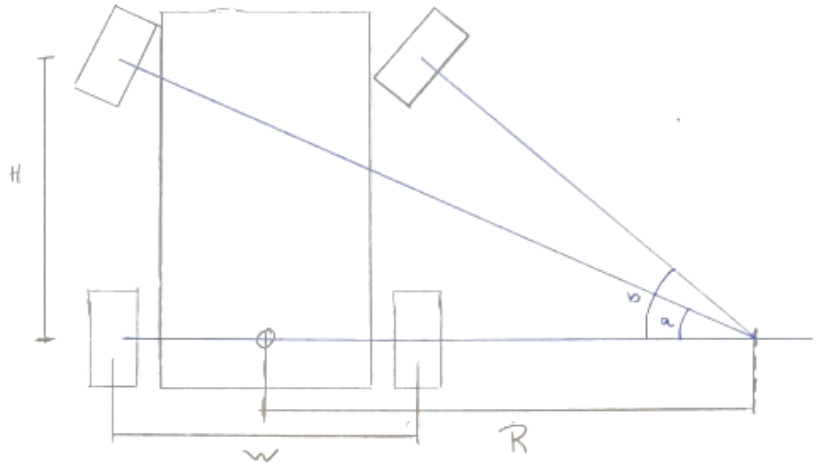
I detta avsnitt kommer de grundläggande koncepten som används i examensarbetet att förklaras för att få en bättre förståelse för arbetet och hur det har utförts.

2.1 Ackermann

År 1816 uppfann en vagntillverkare vid namn Georg Lankensperger tillsammans med Rudolph Ackermann en princip som skulle minska hjulglidning. Tanken var att minska påfrestningarna på dåtidens draghäst samt vagnen. Idag används samma princip för att minska slitage på däck, drivknutar eller dylikt [4].

Principen bygger på att genom att ha olika styrvinklar för in och ytterhjul kunna minimera glidningar hos hjulen. Ackermann-styrningen behövs framförallt när hastigheten av fordonet är låg, Ackermann används då för att undvika slirning [5].

I detta arbete används principen för att få så tillförlitliga sensorvärden från rotationsgivarna som möjligt. I figur 2.1 nedan visas ett exempel när ett fordon svänger höger.



Figur 2.1 - Illustration av hjulvinklarnas relation till farkostens mått. Farkostens inner- och ytterhjul har olika vinklar.

Som nämndes i tidigare stycke så behövs Ackermann vid låg hastighet för att undvika slirning. I ekvation (1) beskrivs de aktuella hjulvinklarna för Ackermann enligt

$$cot\,\delta_o - cot\,\delta_i = \frac{W}{H} \quad (1)$$

där δ_o är hjulvinkel för det yttre hjulet och δ_i är hjulvinkel för det inre hjulet.

2.2 Dead Reckoning

Metoden togs initialt fram för att beräkna kursdrift till sjöss men har på senare tid modifierats för att kunna användas inom kursberäkningar för robotar [6].

Dead-reckoning bygger på att man utgår från en sammanslagning av ett gyroskop och en accelerometer [6]. Ett problem med denna metod är att den blir mindre och mindre korrekt desto längre tid man använder sig av bara accelerometern och gyroskopet ihop. Detta är på grund av brus i sensorerna vilket leder till drift. För att kunna beräkna nästa position korrekt krävs en känd position och en känd riktning, om även hastigheten är känd kan dess nya position beräknas [7]. Tekniken kan dessutom användas för att upptäcka kursdeviation om det finns något opartiskt system att tillgå. Då gyrot och dessa beräkningar hamnar utanför en viss tolerans kan slutsatsen dras att vissa mätvärden ej längre är tillförlitliga.

2.3 Odometri

Odometri innebär att man använder rörelse-avkodare för att mäta hjulens rotation. Med hjälp av hjulens rotation kan man beräkna en rörelse på fordonet [8]. I detta arbete används rotationsgivare som rörelse-avkodare för att beräkna rörelse. För att beräkna hur långt varje hjul har färdats vid varje uppdatering används ekvation (2) och (3) nedan. Dessa ekvationer kan sedan användas ihop med ekvation (4) för att beräkna hur långt centrumpunkten har färdats och riktningsvinkeln Δ θuppdateras med hjälp av ekvation (5) och (6) enligt

$$Hjul_{omkrets} \; / \; Pulser_{varv} \; = \; avstånd_{puls} \tag{2}$$

$$Pulser_{upmätt} \; \bullet \; avstånd_{puls} \; = \; \Delta avstånd_{L,R} \tag{3}$$

$$\frac{\Delta \, Avstånd_{L} + \Delta \, Avstånd_{R}}{2} \; = \; \Delta \, Avstånd_{Centrum} \tag{4}$$

$$\frac{\Delta Avstånd_{R} - \Delta Avstånd_{L}}{H} \; = \; \Delta \, \theta \tag{5}$$

$$\theta_{old} \; + \; \Delta \, \theta \; = \; \theta_{new} \tag{6}$$

Det kan exempelvis vara en rörelse rakt fram (samma riktning som tidigare), vilket skulle innebära att vid en uppmätning till en annan så har rotationsgivarna uppmätt lika många pulser [8]. Detta går att utläsa enligt ekvation (5) då $\Delta\theta \; = \; 0$.

2.4 GPS

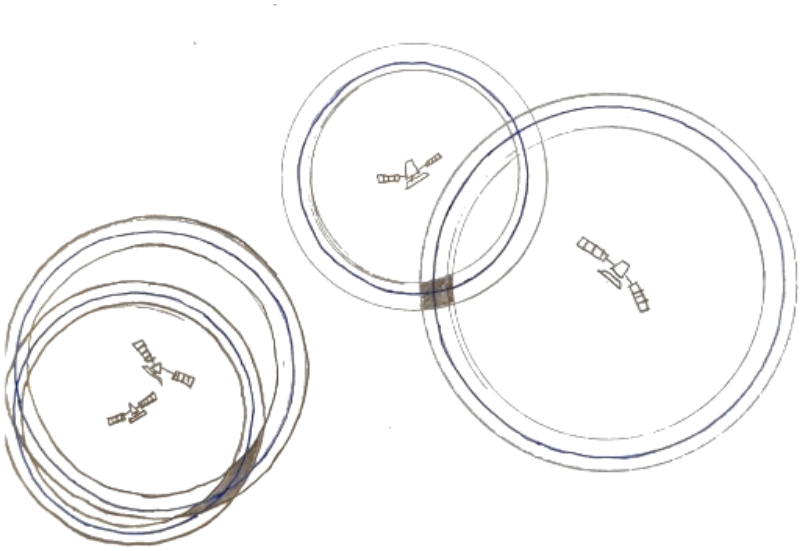
GPS är ursprungligen från USA och de satelliter som är avsatta för detta ändamål är "riktade" för USAs räkning [9]. Det finns både ryska, kinesiska och europeiska motsvarigheter på detta GPS system [10]. Det vill säga GLONASS, BeiDou och Galileo. GNSS är ett globalt system som är ämnat att utnyttja satelliter från flera system för att mer tillförlitligt kunna positionera mottagare runt jorden. GPS-enheten som används i detta arbete använder sig av GNSS och kan alltså motta positionsuppgifter från flera system.

2.4.1 DGPS

DGPS (Differential Global Positioning System) är en form av GPS. Skillnaden mot en vanlig GPS är att vid användande av en DGPS så finns det två stycken mottagare. En mottagare som är monterat på fordonet och en mottagare som är fast monterad med en känd position, en så kallad basstation. På grund av att basstationen har en känd position är det möjligt att få en bättre precision på den GPS enhet som är mobil . Anledningen till att positionen får en bättre precision är för att basstationen kan skicka information till den mobila GPS enheten att uppdatera dess beräkningar [11]. Denna positionsnoggrannhet går ifrån $\pm$ 1 m för en vanlig GPS till en noggrannhet på ungefär $\pm$ 2 cm enligt intervjuade intressenter.

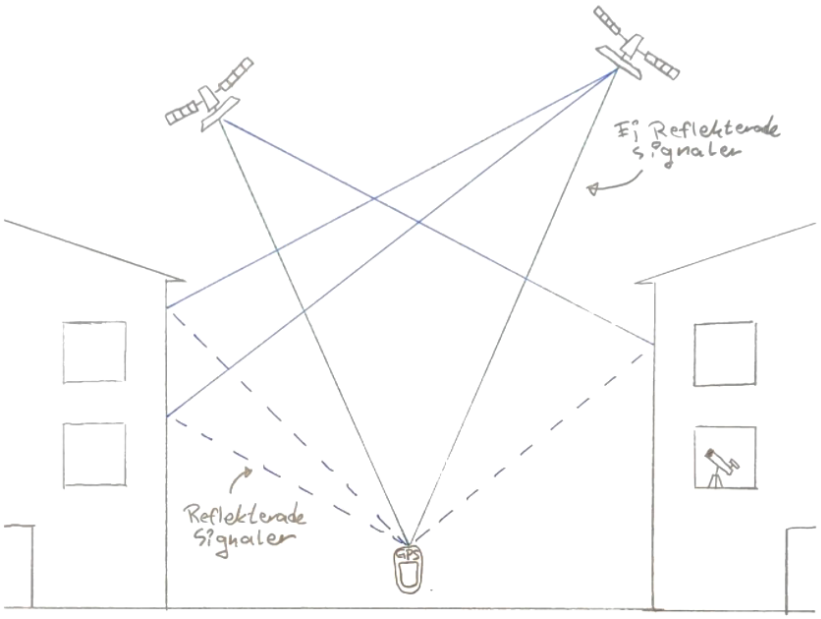
2.4.2 Felkällor

Det finns många olika källor till varför en GPS inte har en korrekt position. En faktor till detta är satelliternas geometri till varandra [12]. Om satelliterna ligger sämre geometriskt från GPS-enheten så blir osäkerheten i positionsmätningen större. Satelliterna kan exempelvis ligga nära varandra eller i en tänkt linje från GPS-enheten. Denna osäkerhet på grund av satelliternas geometri beskrivs som DOP (dilution of precision). DOP indikerar hur bra satelliternas geometri är [12]. Problemet med hur satelliterna är positionerade visas i figur 2.2, där GPS-enheten befinner sig i det gråa ifyllda fältet. I figuren kan man observera att det gråa fältet till vänster är större och därför fås en större osäkerhet om positionen.



Figur 2.2 - Illustration av satelliters geometri och dess påverkan på precision.

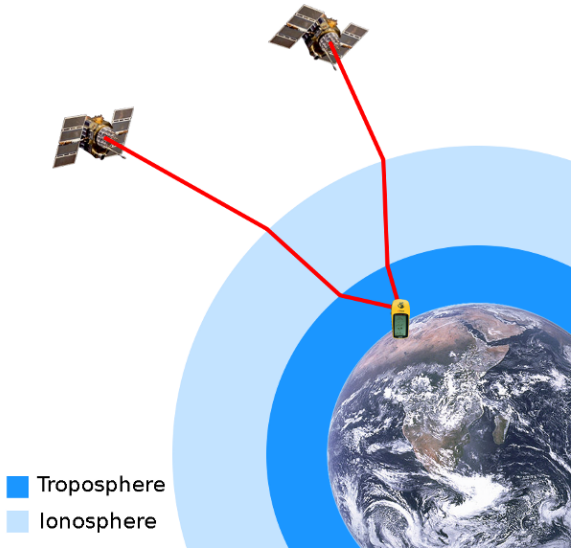
GPS signalerna kan studsa på vissa objekt på jorden [13]. Detta är ett problem i storstäder där det finns byggnader som gör att signalen kan studsa. Den studsande signalen tar längre tid att nå enheten vilket leder till en osäkerhet i positionen, detta visas i figur 2.3.



Figur 2.3. Illustration av studsande GPS signaler.

Det finns också olika delar i atmosfären som leder till att signalen har olika hastigheter [13]. I rymden färdas signalerna med ljusets hastighet. I de övriga lagrena så färdas signalen långsammare vilket illustreras i figur 2.4. När signalen kommer in i jonosfären

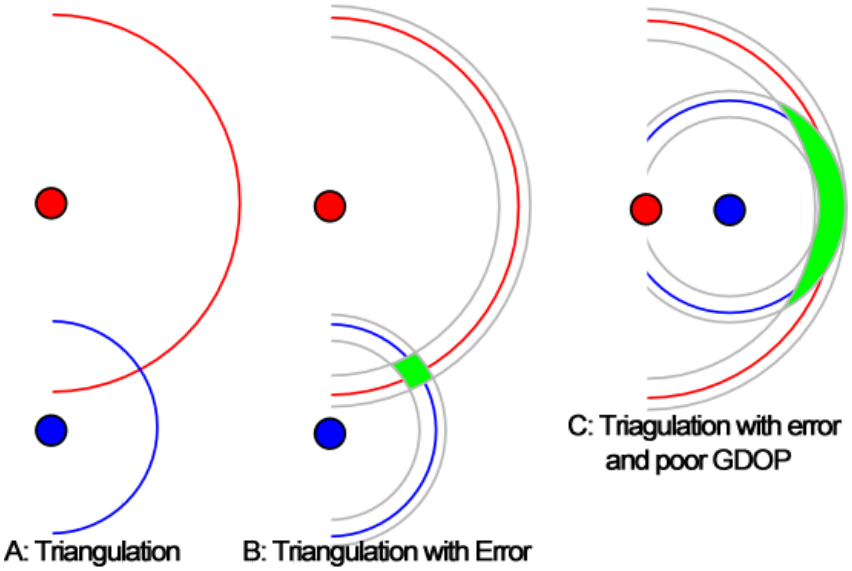
kommer en stor del av positivt laddade joner och elektroner påverka signalen. I normala fall kan en GPS-mottagare beräkna dessa fel och korrigera för detta. En GPS-enhet kan dock inte beräkna om det är något oförutsett som påverkar signalen som exempelvis solvindar [12].



Figur 2.4 Olika hastigheter av GPS signaler i olika atmosfärlager.

2.4.3 Accuracy

Inuti GPS-enheten finns en inbyggd "accuracy"[14]. Accuracyn är ett totalt värde för osäkerheten med GPS-enhetens beräknade position. Det är ett värde uttryckt i m och är en radie från den punkt där GPS:en beräknat att den befinner sig. Detta värde ska återspegla hur bra den estimerade positionen för objektet är, se figur 2.5.



Figur 2.5 - Triangulation och fel i GPS. Grönt fält illustrerar estimerad position för GPS.

GPS-mottagaren beräknar sina koordinater från minst fyra avståndsberäkningar. Dessa värden är pseudoavståndet från mottagaren (GPS-enheten) till de satelliter som är i kontakt med enheten [15]. Detta kan beräknas enligt följande formel

$$P = \rho + c \cdot (dT - dt) + d_{ion} + d_{trop} + e \qquad (7)$$

där P står för pseudoavståndet. Geometriska avståndet är ρ, c är klockan dT och dt är offsets för mottagaren och satellitens klocka. Där d_{ion} och d_{trop} är fördröjning som nämns i kapitel 2.4.2 och e är mätbrus. Det går sedan att använda ekvation (7) för att få fram GDOP vilket är en geografisk precisionsförsämring. Hur detta görs tas inte upp i denna rapport.

2.5 Utvecklingskort

För att behandla sensordata används i detta arbete ett par utvecklingskort från Arduino . Ett utvecklingskort är baserat på en mikrokontroller och innefattar nödvändig kringelektronik för att förenkla användningen av mikrokontrollern. Arduino UNO är baserad på mikrokontrollern ATmega328P [16], medans Arduino MEGA är baserad på mikrokontrollern ATmega1280 [17].

Arduino har även ett eget IDE (Integrated Development Environment) med färdiga bibliotek för hantering av de flesta sensorer och dylikt. Ett par av dessa bibliotek används i detta arbete men mer om hur själva koden fungerar kommer senare.

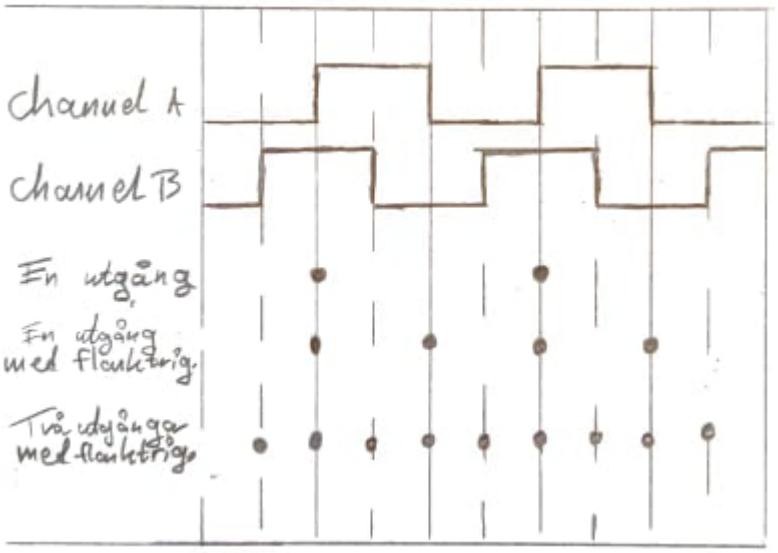
2.6 Sensorer

Här redogörs det för vilka sensorer som använts och hur de fungerar. Uppmätta data sparas sorterat på tidsintervall på ett SD-kort som sedan matas in i den MATLAB-modell som finns för jämförelser och testning.

2.6.1 Rotationsgivare

En rotationsgivare har en av två olika basfunktionstyper för att bestämma sin position. Den ena är inkrementell och den andra är absolut [18]. Den inkrementella givaren mäter och räknar upp eller ner pulser beroende på rotationsriktning. Den absoluta givaren har ett bestämt värde på varje position på hjulet, exempelvis ett värde från 0-100. Oavsett hur många varv hjulet har snurrat så är exempelvis position 30 alltid samma position på hjulet. Roterar hjulet ett varv kommer alltså givaren tillbaka till samma värde. Den givartypen som har använts i detta arbete är en inkrementell givare. Det innebär att oavsett var hjulet befinner sig vid start så börjar den på värde noll och sedan räknar antingen positivt eller negativt beroende på rotationsriktning. Dessa nollpunkter kan då dedikera avläsningsintervall av sensorerna exempelvis och är inte bundna till hjulens absoluta position. De rotationsgivarna som har använts har två utgångar. Detta gör att det markerade antal pulser enligt datablad kan dubblas eller till och med fyrdubblas, se figur

2.6 [18]. Detta görs genom att utnyttja pulsernas höger och vänsterflank. Med hjälp av flanktriggning kan två signaler på en puls användas. En precis då den blir hög, samt en precis då den blir låg. En illustration av detta kan ses i figur 2.6.



Figur 2.6 Illustration av pulsmanipulation med två utgångar (A och B) och flanktriggning. Punkterna visar då en signal skickas till mikrokontrollern.

2.6.2 Inertial Measurement Unit

En Inertial measurement Unit (IMU) består av en accelerometer och ett gyroskop i ett och samma hölje [19]. En accelerometer mäter skillnaden i hastighet, alltså acceleration eller retardation. Accelerometern består av en liten massa och dämpare som är monterade inuti höljet. Om höljet är monterad på exempelvis en bil som i detta fallet kan accelerometern beräkna skillnaden i hastigheten på grund av trögheten i massan inuti accelerometern. IMU:n erbjuder också ett accelerometerbaserat gyroskop. Detta gyroskop ger information om ändringar i roterande led och tillsammans med accelerometern ger dessa mätningar i 6 led (DOF).

2.7 Sensor Fusion

Sensor fusion innefattar att det finns mer än en fysisk sensor som samlar data. Målet är att dessa sensorer ska samarbeta för att få ett så noggrant resultat som möjligt. Sensor fusion och low-pass filter kan användas ihop för att minska sensor drift vilket leder till att det går att bestämma mer korrekt data från sensorvärden [20].

2.8 CAN

Controller Area Network(CAN) är ett bussystem som grundades av Robert Bosch GmbH år 1986. Fortfarande idag har i princip alla bilar som blir tillverkade ett eller flera CAN-nätverk [21].

CAN är ett meddelandestyrt bussystem där meddelanden skickas på bussen med olika prioritet. De meddelandet med högst prioritet får tillåtelse att skickas över bussen. Detta prioritetsystem är anpassat på ett sådant sätt att de mikrokontroller som finns placerade i bussystemet kan prata med varandra [21].

I detta projekt används ett protokoll som heter CAN Kingdom. Skillnaden mellan detta protokoll och ett vanligt CAN protokoll är att varje nod inte behöver ta del av information av vad hela systemet gör. Varje nod har en specifik uppgift och tack vare detta protokoll får den en utsatt tid då den får skicka meddelanden över bussen. Detta motverkar kollisioner och gör systemet snabbare [22].

3 METOD

Detta avsnitt redogör för metoden och dess delar. Kapitlet tar upp den modell och togs fram samt beskrivning av den utvecklingsplattform som användes för projektets proof of concept.

3.1 Modellering

I detta arbete användes en simulerad modell i MATLAB för att kunna modellera de sensorvärden som samlades in under testningens gång. Den modelleringen som gjordes används till att kontrollera positionen på farkosten.

3.1.1 Odometri och Dead Reckoning

För att mäta avståndet som ett fordon har färdats används Odometri (se kapitel 2.3) . Odometri är att man med hjälp av sensorer översätter data från dessa till en viss rörelse på hjulen [23]. I detta fallet har två stycken rotationsgivare används. För att få ut rätt avstånd som plattformen har färdats behöver man använda sig av upplösningen på rotationsgivarna och simpel geometri. Genom att använda formeln nedan får man ett avstånd per puls. Plattformens hjulomkrets blev uppmätt till $0.438\,m$ och rotationsgivarnas upplösning sattes till 400 pulser per varv. 400 pulser valdes då detta intervall var tillräckligt högt för att upplösningen skulle vara tillfredsställande hög men också tillräckligt låg för att inte överbelasta utvecklingkortens klocktakt. Detta gjordes genom att implementera flankkänsliga interrupts som tidigare beskrivits i rapporten.

$$Hjulomkrets \;/\; Pulser_{varv} \;=\; 0,001095\,m \;\approx\; 1,1\,mn \qquad (8)$$

För att få reda på vart plattformen har färdats så används odometri tillsammans med en uppmätt riktningsvektor för att kunna använda Dead Reckoning (se kapitel 2.2). Vilket innebär att hjulsensorerna plus en riktning ger den aktuella förflyttningen från en uppmätt position till nästa.

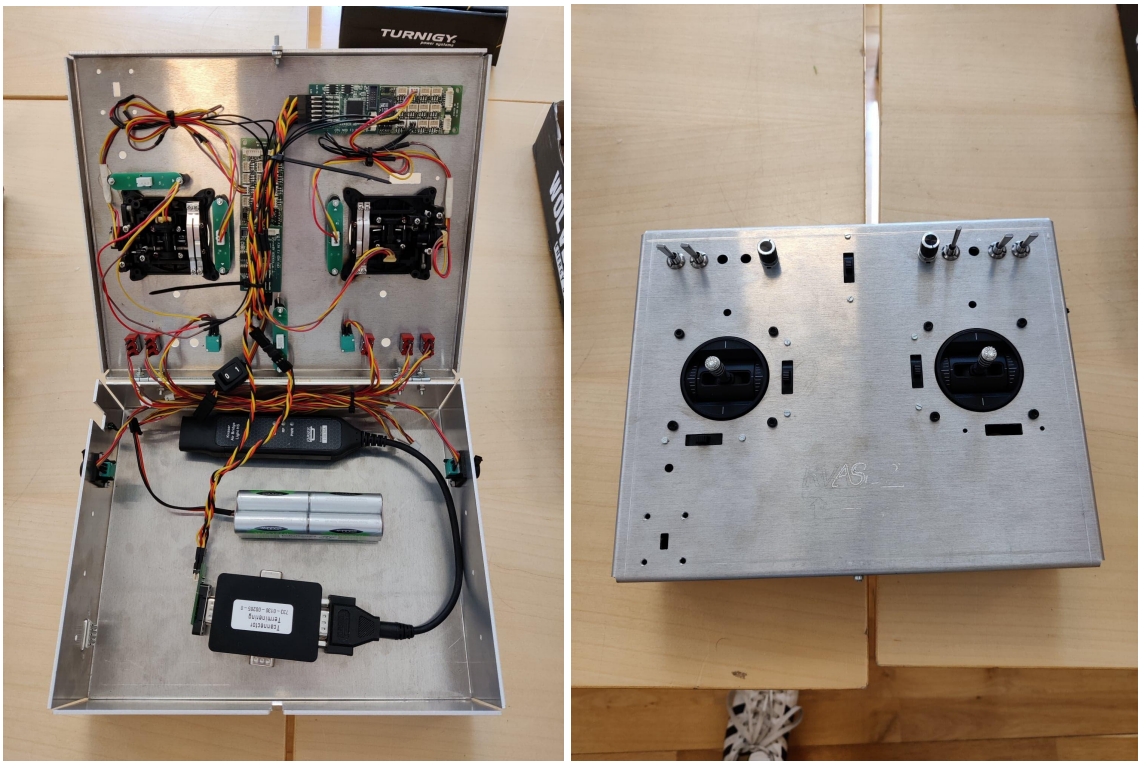
3.1.2 MATLAB

MATLAB har i detta arbete använts först för lite undersökning och inledande simulering men har senare anpassats för att ta emot de sensorvärden som uppmätts. Dessa sensorvärden har med hjälp av MATLAB producerat ut de positionsbilder som sökts. Hur MATLAB användes för att analysera sensordata kommer redogöras för senare i rapporten.

3.2 Utvecklingsplattformen

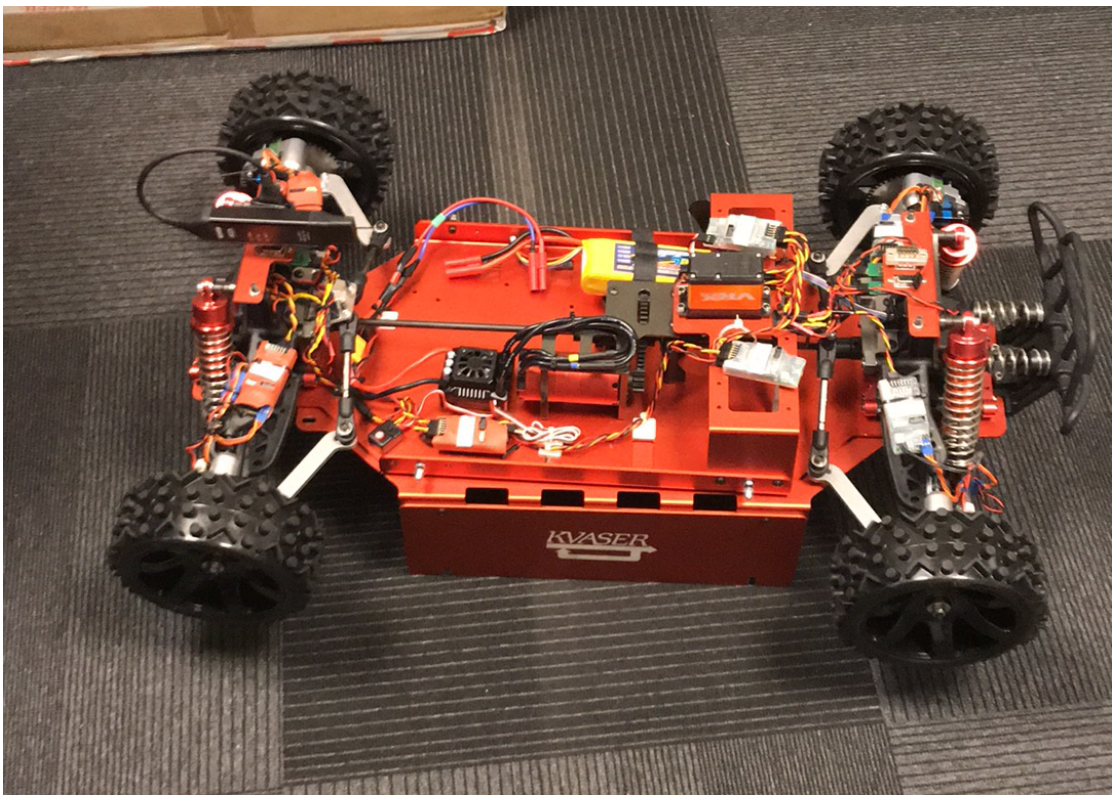
Plattformen som användes i projektet är en radiostyrd bil i skala 1:5 till en vanlig bil. Plattformen använder två CANbus-system sammanlänkade trådlöst för kommunikation. I plattformens system finns det 7 stycken noder där 4 noder hanterar varje hjul individuellt, en nod för styrning, en för motorn samt en CANkung(“huvudstaden”, se kapitel 2.8.1).

Utöver dessa har radiosändaren 2 noder i ett separat CANbus-system, en för gas och en för styrning. Utöver detta finns även en Air bridge vilket används för kommunikation mellan styrenheten och plattformen (se figur 3.1).



Figur 3.1 - Insidan- och utsidan av radiosändare, med synliga CANbus noder, air bridge och gimbals.

De två systemen kommunicerar via en "Air bridge" som Kvaser tillhandahållit. Denna setup har tidigare varit del av ett annat projekt som utvecklade ett kommunikationsschema samt exemplifierade den trådlösa kommunikationen [24]. Under arbetets gång finns samtliga noder monterade men endast 3 respektive 2 noder är aktiva i kommunikationen. Anledningen till detta tas upp senare i rapporten. En bild på hur plattformen såg ut när den överlämnades kan ses i figur 3.2.



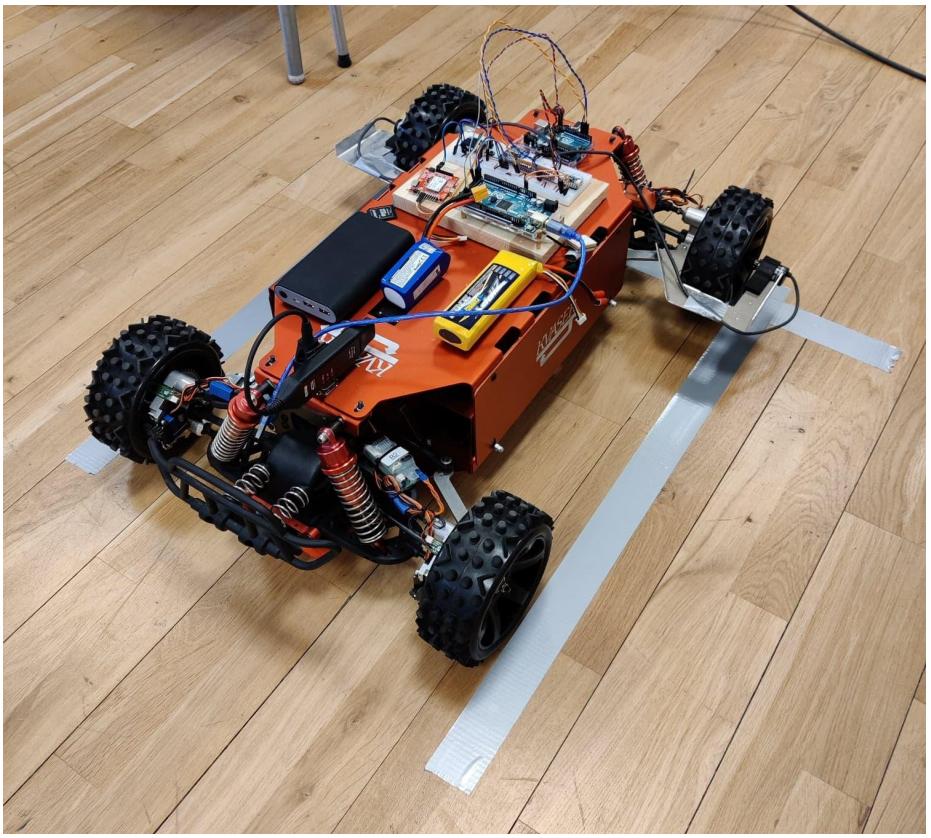
Figur 3.2 - Bilen som fungerade som utvecklingsplattform under arbetets gång.

3.3 Testsituationer

Här behandlas de testfall som tagits upp i arbetet för att kunna validera de alternativa positionssystemen. Här beskrivs hur de olika testsituationerna uppmättes, markerades samt vad testerna går ut på och vad som eftersöks.

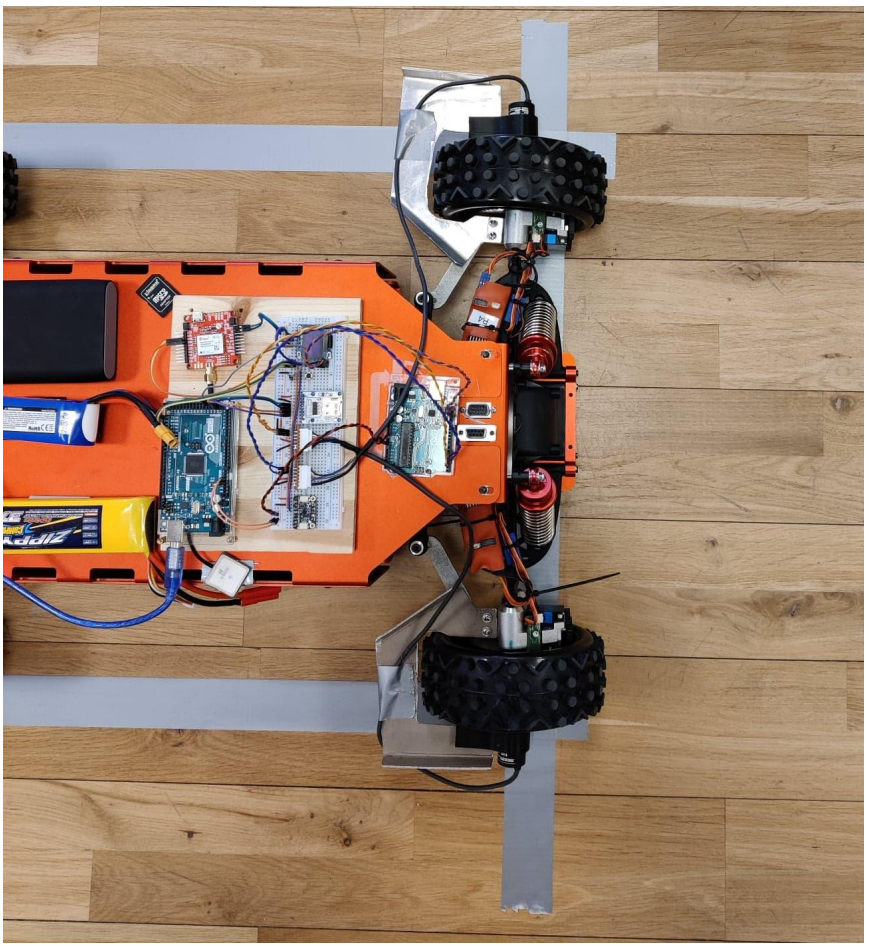
3.3.1 Markeringar och uppmätningar

Samtliga tester genomförs med en känd startpunkt som är markerad. Markeringen genomfördes med hjälp av tejp eller färg längs farkostens kanter (se figur 3.3). De längsgående markeringarna finns till för att se till att farkosten går i en ungefärlig lika riktning varje testkörning. Det går däremot inte att lova exakt samma startriktning vid varje test, i varje ny körning kan startvinkeln behöva korrigeras i syfte att jämföra två liknande körningar.



Figur 3.3 - Farkosten i en startpunkt med markeringar bestående av tejp.

För att få en högre precision har kanten på markeringen används som startpunkt (se figur 3.4). Vissa test har samma stoppunkt som startpunkt och andra har en separat stoppunkt. Då en separat stoppunkt användes har denna markerats upp på samma sätt som startpunkten. Farkosten placeras på en sådan punkt så att givarnas axel sammanfaller med markeringens kant. Vid denna startpunkt anses farkostens placering ej kunna variera mer än någon centimeter. Vid stoppunkten rullas bilen sista delen för hand för. På så sätt kan inte uppmätt stoppunkt och den faktiska stoppunkten deviera mer än $\pm$ 10 cm mot uppmätt körsträcka. Detta är den precisionsgrad som utlovas i mätningarna vid testkörningarna.



Figur 3.4 - Illustrering av hur hjulens och givarnas axel sammanfaller med startpunktens kantmarkering.

Den MATLAB-kod som finns för analys av mätvärdena utgår från en punkt i linje med rotationsgivarnas axlar. Denna punkt placeras alltså över tejpkantens linje och sedan körs farkosten den förutbestämda rutten för att sedan stanna i linje med nästa markering med tejp. För att få högre precision på slutpunkt kan farkosten dras eller rullas fram sista biten för hand så rotationsgivarnas axel sammanfaller med tejpkanten mer exakt. Uppmätningen av avstånd för testkörningarna är gjorda med hjälp av en lasermätare för avståndsmätning. Modellen som kan ses i figur 3.5 nedan har en noggrannhet på $\pm$ 2 mm över 50 m.

Figur 3.5 - Lasermätare av märket Mileseey.

Den använda GPS enheten har en inbyggd variabel för tillförlitligheten av positionen. I de kommande figurerna följer dessa detta färgschema:

Tabell 3.1 - Färgschema för GPS positionens tillförlitlighet.

Färg	Tillförlitlighet (Diameter)
Svart	50 cm
Röd	40 - 50 cm
Gul	30 - 40 cm
Grön	20 - 30 cm
Blå	10 - 20 cm
Cyan	Under 10 cm

3.3.2 Störningar

Under alla de testsituationer som nämnts nedan har störningar mot GPS:en introducerats. Detta för att simulera verkliga situationer där GPS signalen fallerar eller blir störd. Exempelvis vid tunnelkörning eller i stadsmiljöer där GPS signalerna kan studsa på hus eller dylikt. Störningarna simuleras genom att linda in GPS enhetens antenn med aluminiumfolie (signalstudsning), samt urskruvning av antennen (tunnelkörning).

4 GENOMFÖRANDE

I följande kapitel kommer genomförandet av arbetet redogöras. Först utfördes en marknadsundersökning för att skapa en anknytning till verkliga marknadsproblem, sedan utvecklades testsituationer för att testa de framtagna alternativa positionsbilderna och illustrera eventuella brister hos dem.

4.1 Marknadsundersökning

För att validera att arbetet behandlar riktiga problem och för att se hur problemet ser ut på marknaden idag har ett antal ingenjörer och andra experter rådfrågats. De företag som varit delaktiga i denna undersökning är främst Volvo, RISE och Kreolis.

Marknadsundersökningen inleddes genom ett möte med en insatt expert på Volvo. Från det mötet kunde det fastställas att Volvo i dagsläget arbetar med att kunna inkorporera GPS som ett stödsystem till sina autonoma tjänster. Dessvärre är de GPS-enheter som används av Volvo idag för lågupplösta för att kunna erbjuda en tillräckligt bra positionsbild för användning till autonomitet i bilar. Efter detta möte togs en enkät fram för att kunna hitta fler företag som arbetar med problematiken om självkörande bilar. Denna enkät går att läsa igenom i Bilaga 5. Tyvärr såg denna enkäten väldigt lite trafik och informationen som kunde tas ut ur den var mer eller mindre helt oanvändbar.

4.1.1 S3 projektet

Fortsättningsvis kontaktades RISE som tillsammans med Kreolis och Navya är delaktiga i ett projekt med en självkörande buss på Lindholmen. Projektet är stort med många delaktiga parter och väldigt betydelsefullt då det utförs i riktig trafik. Från denna dialog framkom det att de har ett omfattande problem med tillförlitlighet på den GPS som de använder. Bussarna som är i bruk på Lindholmen använder GNSS i ett DGPS för att placera sig själva på en karta. Detta system ger en positionsnoggrannhet under vanliga förhållanden på $\pm\, 2\, cm$ enligt de intervjuade företagen. I figur 4.1 nedan syns signalstyrkan hos GNSS signalen för bussen. De blåa cirklarna är områden där man manuellt lagt in exkluderingszoner för GPSen. I dessa förlitar sig bussen endast på sitt LIDAR-system.

Figur 4.1 - Signalstyrkan för S3-bussen utritade längs med sin rutt samt med exkluderingszoner inringade i blått.

Bussarna upplever dessa större signalbortfall(blå cirklar från figur 4.1) vid korsningen på Götaverksgatan och den del av rutten där bussen följer Regnbågsgatan. Man kan även se i figuren hur GPS-enheten har problem med positionering mellan hus där signalerna kan studsa. För att avgöra hur noggrann positionen är används en norm som heter NMEA 0183. Denna information kommer ifrån en av de ingenjörer vi intervjuade angående projektet. Idag körs endast en buss i taget då ingenjörerna har sett en sämre prestation när det körs flera bussar samtidigt. Exakt vad detta beror på är i dagsläget oklart men utreds enligt intervjuade parter.

4.2 Utvecklingsplattformen

Innan arbete kunde utföras med plattformen behövde den förberedas för att kunna skicka rätt sensorbilder. Då bilen redan var utrustad med sensorer så var målet att få samtliga 4 rotationsgivare att skicka över detta CANbus-system då detta hade kunnat ge sensorvärden live. Dessvärre dök vissa komplikationer upp och i tidsintresse valdes det att låta ett par externa Arduinos sköta sensorbilderna och dataloggningen tillsammans med andra givare. Det är som konsekvens av detta tidsintresse som vissa noder inte användes. Det fanns ett kodexempel från ett tidigare arbete där endast gas och styrning användes. Detta kodexempel kunde lättare anpassas till arbetet så testningen kunde börja i tid.

4.2.1 Modifikationer

På plattformen har nya rotationsgivare (Fabrikat: YUMO E6A2-CS3E), GPS (Ublox ZED-F9P) samt gyro (Sparkfun LSM6DSO32) monterats. Det finns två anledningar till att det monterades nya rotationsgivare. Den ena var som nämndes i tidigare stycke att det var problem med att få givarna att prata över CANbus-systemet och det andra var att det var för dålig upplösning på de givarna som tidigare fanns på plattformen. Det vill säga att de nya givarna hade mer pulser per varv och kunde därför mäta en mer noggrann position.

Enligt rekommendation från mjukvarubiblioteket Adafruit_SSD1306.h så behölls gyrots ursprungliga uppdateringsfrekvens då denna - något lägre uppdateringsfrekvens enligt dem skulle ge mer tillförlitliga resultat.

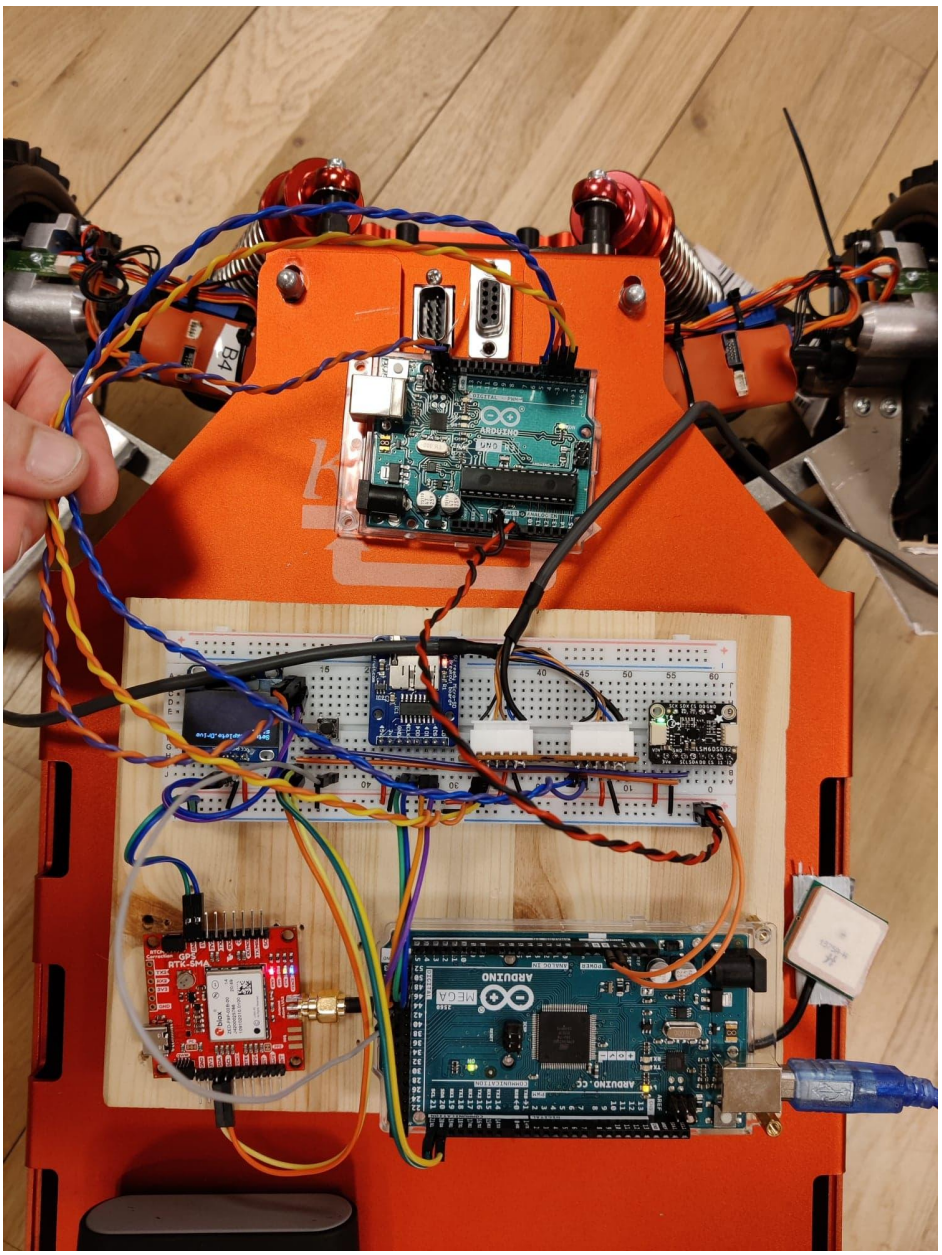
Plattformen har modifierats från fyrhjulsdrift till tvåhjulsdrift. Detta gjordes genom att drivaxeln till bakhjulen demonterades. Anledningen till att detta genomfördes var för undvika att få hjulspinn på bakhjulen vid körning. Detta för att få så rena sensorvärden som möjligt.

Det upptäcktes att hjulen som var monterade inte var raka vilket ledde till att plattformen inte körde rakt fram när den skulle det. Det gjordes försök att räta upp hjulen och implementera den Ackermann-styrning som undersökts men insågs efter ett tag vara ett orimligt mål på grund av till exempel styrglapp och svåråtdragna skruvar. Under testningens gång kvarstod därför ett kursdriftsproblem som föraren var tvungen att korrigera för med styrningsinput.

4.2.2 Sensorer och kommunikation

Rotationssensorer användes ihop med IMU-enheten för att kunna kartlägga hur bilen rörde sig. Sensordata matades in i den MATLAB-modell som tagits fram och jämfördes med den uppmätta GPS-positionen.

När sensorerna skulle implementeras via Arduino upptäcktes det ett problem. Den Arduino UNO som tänkts att användas för alla sensorer inte hade tillräckligt med dynamiskt minne. Därför byttes denna UNO ut till en Arduino MEGA, men p.g.a. en interrupttung kommunikation beslutades det slutligen att använda båda två utvecklingskorten. UNOn användes för att sköta endast rotationsgivarna för hjulen och skicka denna data över en I2C buss på begäran från MEGAn som blev utsedd till master. MEGAn skötte även de andra sensorerna då dessa inte var interruptberoende. Se nedan figur 4.2 på komplett setup.



Figur 4.2 - De två utvecklingskorten och brödkortet med alla sensorer. Den övre arduinon var slav och den undre var master.

4.2.3 Koden

Komplett kod finns att tillgå i bilaga 2 och bilaga 3, men kommer översiktligt förklaras här.

De två rotationsgivarna har två utgångar var, Dessa fyra utgångar kopplades till hårdvaruinterrupts på digitala ingångar i Arduino UNOn. Inget bibliotek behövs för att kunna hantera dessa sensorer utan UNOn reagerar endast på hög eller låg signal på dessa ingångar. Genom att göra UNOn flankkänslig dubblerades sensorernas upplösningar.

Kommunikationen som nämnts tidigare använder sig av I2C protokoll. Detta protokoll

tillåter 128 enheter att kommunicera med varandra med olika adresser [25]. För användning med Arduino är Wire.h biblioteket vanligt. Med hjälp av detta bibliotek kunde bytes skickas antingen enskilt eller kontinuerligt med en start- och stoppsignal beroende på hur kommunkationen upprättats.

För att dessa insamlade rotationsvärden ska kunna skickas över den I2C buss som upprättats behöver de sparas på ett speciellt sätt. Detta på grund utav paketbegränsningen på bussen. Ett paket får maximalt vara en byte och därför sparas de långa intarna i union med en byte array [26]. En union innebär att de två variablerna delar minnesplats och kan på så sätt skickas byte för byte och tolkas som en int av mottagaren.

I2C bussen hushåller även de andra mer avancerade sensorerna med mer omfattande bibliotek. Dessa två sensorer sattes upp enligt de önskade uppdateringsfrekvenserna och upplösningarna och hämtades av MEGAn då informationen skulle loggas. Informationen skevs till ett SD-kort som använde sig av ett något utdaterat bibliotek. Detta medförde att det ursprungliga minneskortet ej var kompatibelt och ett mindre kort behövde införskaffas.

En liten OLED display användes under setupen för att ge en indikation på när GPS-enheten har konvergerat till rätt position, men också för att upplysa om eventuella fel. Detta för att undvika körningar som resulterade i korrupt data. Varje dataloggning får en tidsstämpel och sparas som en sträng, radvis i ett textdokument. Ordningen i denna sträng är: Millisekunder sedan start, Rotationsgivare Höger, Vänster, Gyrot i Z-led, Latitude, Longitud, Accuracy. Detta textdokument tolkas sedan av MATLAB och lästes in som en datamatris.

4.2.4 MATLAB

Precis som Arduino-koden finns MATLAB-koden att tillgå via bilagorna. Denna hittas i Bilaga 4.

Den inlästa datamatrisen delades upp i mer hanterbara arrayer, detta var inte ett nödvändigt steg men gjorde koden mer lättläslig. Plattformens mått läggs också in som konstanter inför de beräkningar som utfördes. Beräkningsgången ser ut som sådan:
Först beräknades avståndet som vardera sensorförsedda hjul rullat i meter. Detta gjordes genom att räkna antalet pulser mellan sensorläsningarna för de båda hjulen, dela detta med pulsupplösningen för att få antal rotationer, och sedan slutligen gångra med hjulet omkrets för att få fram avstånd som plattformen färdats i meter. Formeln från MATLAB kan ses nedan.

$$Distance\ in\ m\ = \frac{Pulses(i+1) - Pulses(i)}{Pulses\ per\ Revolution} \cdot Wheel\ Circumference \qquad (9)$$

Där $i$är en loopvariabel som här illustrerar om aktuellt (i), föregående $(i - 1)$ eller nästa $(i + 1)$ värde är del av ekvationen. Efter denna uträkning skiljde sig tillvägagångssätten

något. Den positionsbilden som hade stöd av ett gyro fick en uppmätt kursändring medans positionsbilden med endast rotationsgivarna behövde beräknas. Formeln är:

$$\theta(i+1) = \frac{RW\ Distance\ in\ m(i) - LW\ Distance\ in\ m(i)}{Wheel\ width} + \theta(i) \qquad (10)$$

Där θär en kursvinkel, RW och LWär Right- respektive Left wheel och $Wheel\ Width$här avståndet mellan hjulen. Denna kursberäkning är en estimering baserat på en publicering från G.W. Lucas [27].

Det fanns även utrymme för att implementera en annan formel för korrektion av kurs i snävare kurvor men på grund utav den korta samplingstiden och de tester som utfördes valdes denna att inte implementeras. Det fanns även fler komplikationer med detta som tas upp senare i rapporten. Efter ett par tester märktes det att kursen var något känslig. Om plattformen körde helt rakt kunde rotationsgivarna ändå skilja på 1-2 pulser vilket ledde till att dessa små kursändringar filtrerades bort.

Vidare utfördes positionsberäkningar. Ett genomsnittligt avstånd beräknades från de två hjulens avstånd och nya X och Y koordinater beräknades med hjälp av kursriktning, avstånd och sinus samt cosinus.

$$Distance\ X(i) = Avarage\ distance\ (i) * cos(\theta(i)) \qquad (11)$$

$$Distance\ Y(i) = Avarage\ distance\ (i) * sin(\theta(i)) \qquad (12)$$

$$Position\ X(i+1) = Position\ X(i) + Distance\ X(i) \qquad (13)$$

$$Position\ X(i+1) = Position\ X(i) + Distance\ X(i) \qquad (14)$$

För den gyroassisterade positionsbilden utfördes dessa beräkningar på samma sätt. Enda skillnaden var att denna kurs är en uppmätt kurs.

De två sensorbilderna som förklarats här samlades ihop i ett diagram och presenteras tillsammans med de uppmätta GPS positionerna. Anledningen till att de finns i separata diagram beror på att de två positionsbilderna har olika koordinatsystem. För plottning av GPS positionerna har funktionen geoscatter använts. Det finns även ett tillägg i koden för att mäta upp avståndet mellan första och sista GPS positionen. Detta avstånd är fågelvägen men ger en generell uppfattning om GPS signalen fallerar eller dylikt.

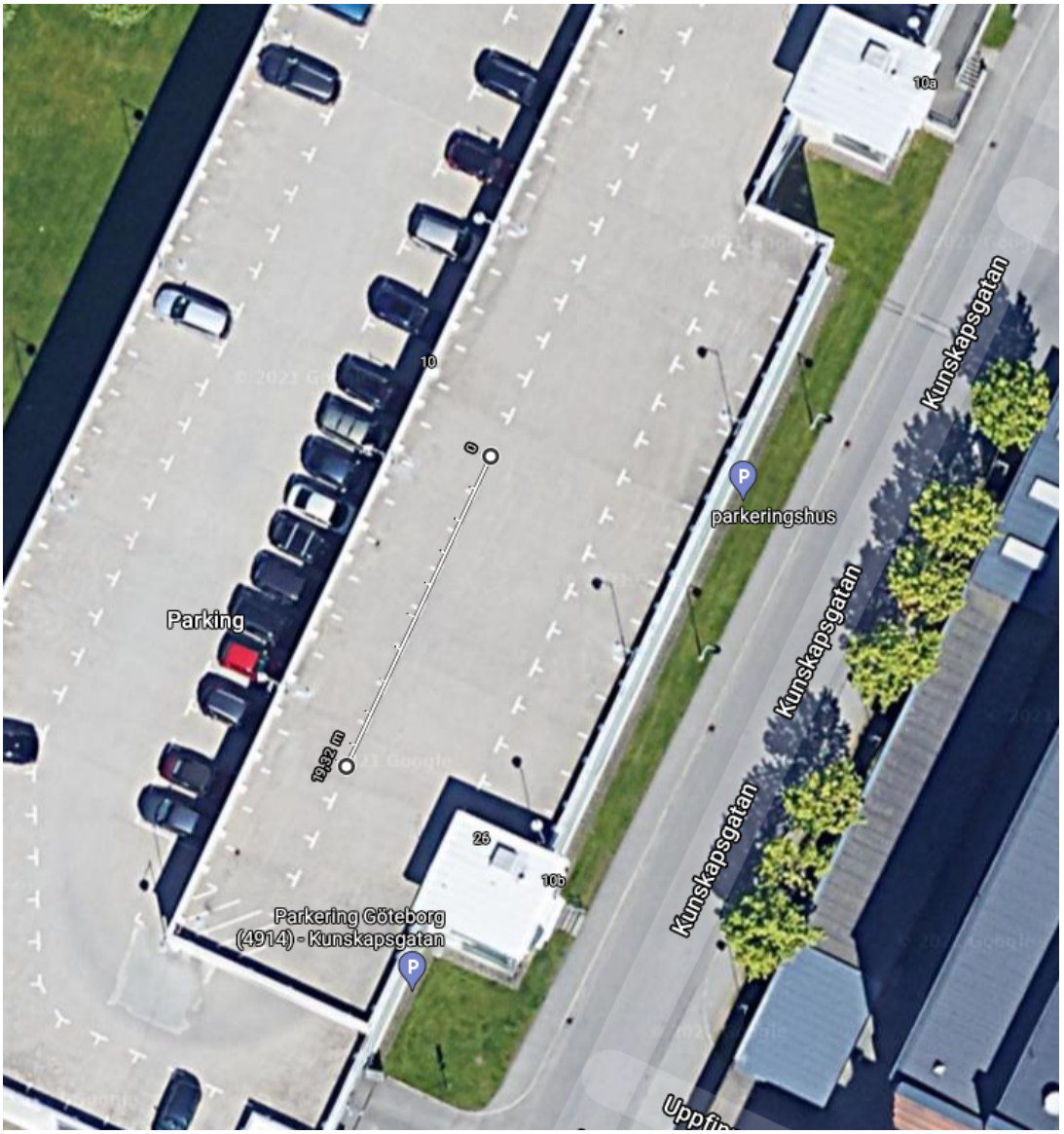
4.3 Körningar

I detta kapitel kommer alla de körningar som utförts att redovisas. Körningarna kommer att kort redovisas med figurer för de olika scenarierna. Körningarna utfördes först med alla sensorer i normalt läge, sedan störddes GPS och till sist med helt bortkopplad antenn.

Samtliga set om testkörningar har genomförts vid god väderlek och under ett kort tidsspann för att inte påverka GPS-enhetens signalstyrka. Alla körningar med tillhörande grafer finns att se i bilaga 1.

4.3.1 Rak körning

Farkosten placeras på en känd startpunkt och körs en uppmätt sträcka så rakt som möjligt. Testerna genomfördes med styrkompensation för den kursdiviation som farkosten hade. Testerna utfördes på taket till ett parkeringshus med målade parkeringsrutor som guide för kurs mellan start- och slutpunkt, se figur 4.3.



Figur 4.3 - Överblick över testområde, uppmarkerad rutt för rak körning.

Underlaget är gjuten betong och med minimal höjdskillnad. Teststräckan uppmättes till $19.3\ m$ och testerna utfördes med en start- och stopprecision på $\pm\ 10\ cm$.

4.3.2 Ögelkörning

I detta test ställs farkosten på en startpunkt på samma sätt som tidigare. Först körs farkosten rakt ut och sedan runt en förutbestämd halvcirkel med en diameter uppmätt på ca. 4 m, med mittpunkt 7 m bort från startpunkten - för att sedan stanna på samma punkt på som den började på, se figur 4.4.

Figur 4.4 - Uppmarkerad rutt för Öglekörningen. Till höger uppmätt diameter för sväng.

Genom att köra på sådant vis kan eventuella sensordeviationer tydligt illustreras. Under testningen märktes ingen märkbar skillnad mellan vänster- eller högervarv och därför har samtliga tester utförts i vänstervarv. Detta för att få en enhetlig jämförelse. I bilaga 1 finns dock ett par högervarv att tillgå för jämförelse.

4.3.3 Rektangelkörning

Här körs farkosten runt en utstakad fyrkantig bana som har innermått på ca $7,2\,m \times 7,3\,m$. Den återvänder till samma plats som den började vid se figur 4.5 nedan.

Figur 4.5 - Uppmarkerad rutt för rektangelkörningen.

Startpunkten är placerad mitt på en av rektangelns sidor, ca 1 m ut från rektangelns linje, detta för att underlätta körningen då svängradien på bilen inte är den bästa. Rutten har utförts på ett sådant sätt att dessa punkter använts som apex i kurvorna. Även i detta test kördes endast vänstervarv.

5 RESULTAT

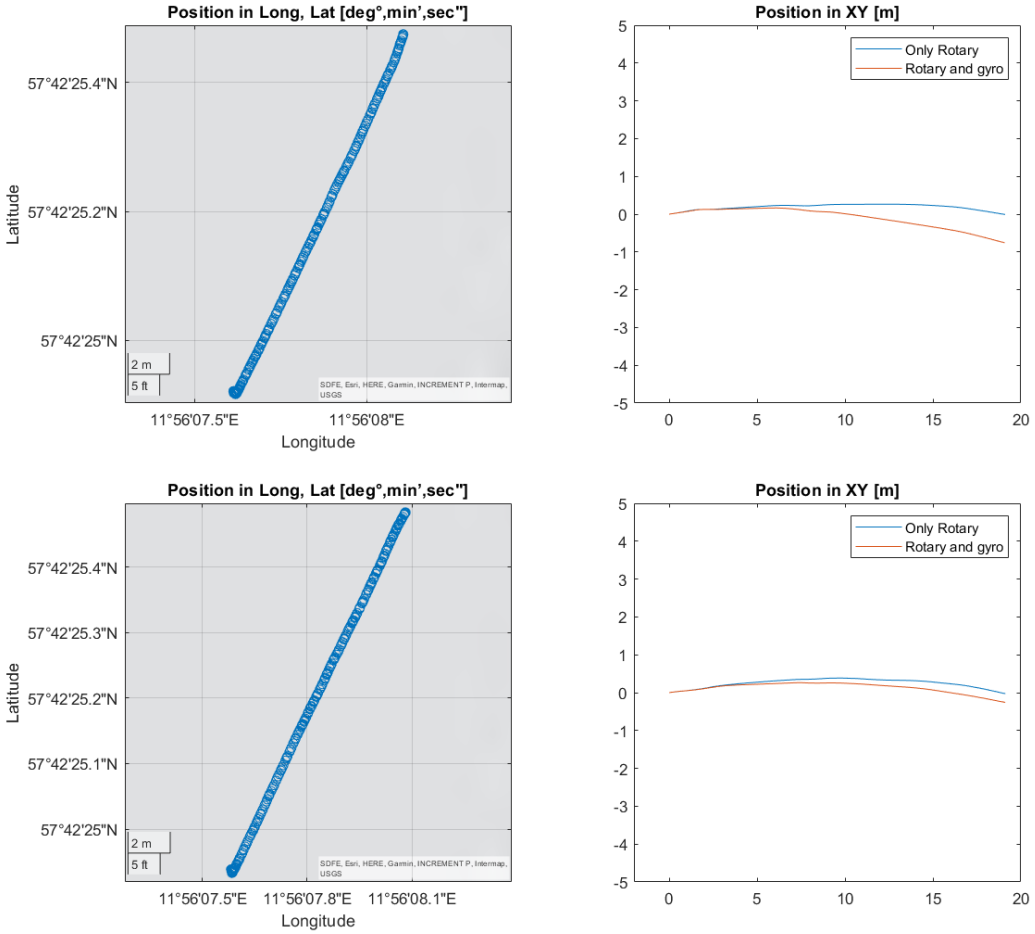
I detta stycket kommer resultaten från testsituationerna som genomfördes i föregående kapitel att redovisas. Resultaten kommer att visas med figurer från MATLAB där både positionen för rotationsgivare, gyro och GPS från plattformen kommer att visas. Då det inte har funnits något ytterligare system för en absolut känd position för farkosten under färd kommer främst resultatet att visa hur väl eller nära farkosten hamnar slutposition efter utförd körning.

5.1 Körningar

Resultatet av de genomförda körningarna presenteras i detta delkapitel. Kapitlet går mer grundligt igenom de raka körningarna då dessa körningar har mer omfattande villkor för en rättvis jämförelse.

5.1.1 Raka Körningar

Första raka körningen utfördes med ostörd GPS för vidare jämförelse. I figur 5.1 nedan kan de olika sensorbilderna ses. I vänsterspalten finns GPS enhetens positionsbilder, med en träffsäkerhet mellan 10 och 20 cm (som indikeras av den blåa färgen). GPS:ens positionsbild är okorrigerad då den inbyggda MATLAB funktionen geoscatter är automatiskt korrigerad efter longitud och latitud. Till höger syns de två andra sensorernas positionsbild. Startvinkeln för de båda sensorbilderna har korrigerats så att Rotationsgivarnas start- och stoppunkt sammanfaller med x-axeln. Detta för att kunna ge en bättre bild av sensorernas deviation.

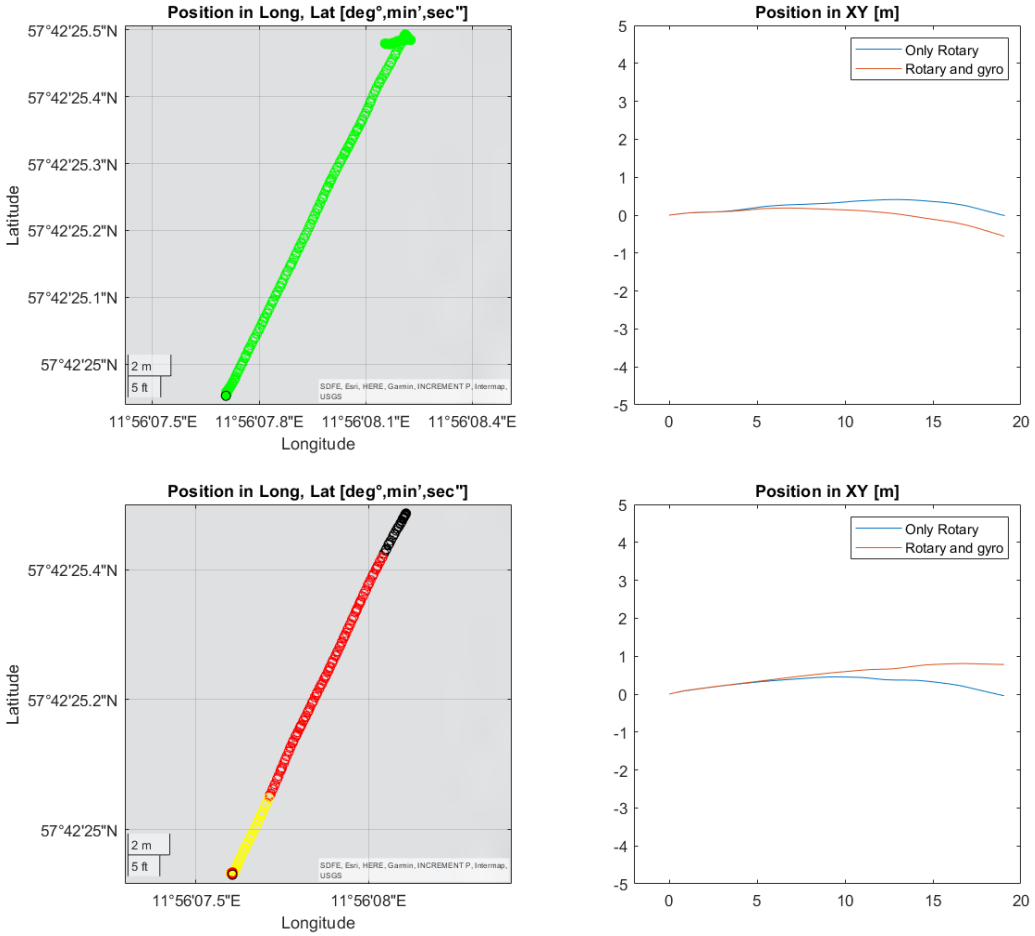


Figur 5.1 - Rak körning med ostörd GPS.

Anledningen till att rotationsgivarnas kurs är den som grafen är upprättad efter är på grund utav att IMU sensorn upplevs som brusig. Det innebär dock att den gyroassisterade kursen inte slutar på x-axeln.

Ur denna figur går det att konstatera att modellen har ett problem med sensordeviation. Över de körda $19.3\,m$ syns det en deviation på ca $20 - 40\,cm$ för rotationsgivarna medans den gyroassisterade kursen devierar mer. GPS enheten hävdar däremot en hög träffsäkerhet, men utan ett tredje opartiskt system kan inga välgrundade observationer göras angående vilken exakt färdväg som faktiskt tagits.

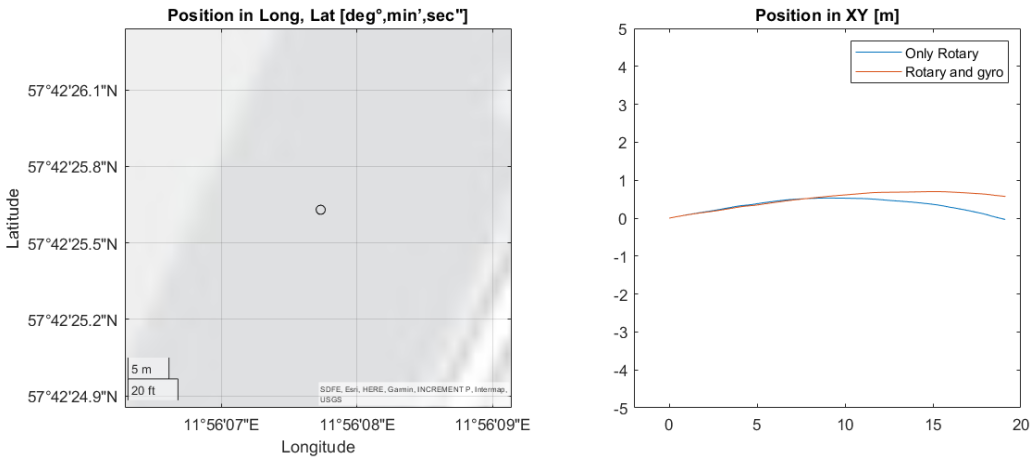
Inför nästa körning virades antennen in i aluminiumfolie. Detta för att simulera studsning, eller i allmänhet, en sämre signalstyrka. Körningarna visas i figur 5.2.



Figur 5.2 - Rak körning med Foliestörning runt antennen.

Här ligger positionsbilderna från gyro och rotationsgivare inom samma felmarginal som tidigare. Det går dock att se att mätningarna från gyrot går åt olika håll vilket inte var fallet under körningen. GPS-positionen har varierande längd på körningarna och dessutom en sämre accuracy. Färdat avstånd är $17.9\,m$ respektive $19.1\,m$ enligt GPS-enheten. Det går också att se viss vandring i positionen hos den gröna GPS rutten.

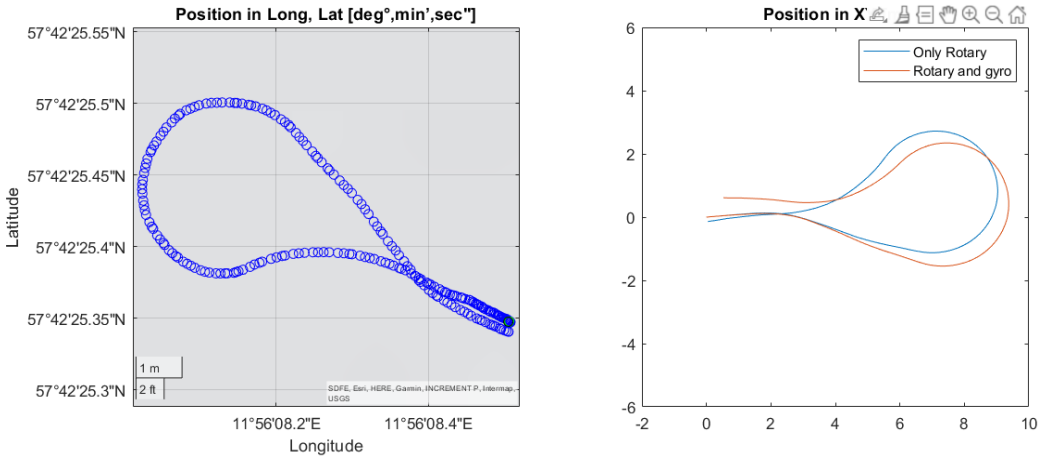
Vid körning med antennen helt utskruvad fallerar GPS positionen helt. I figur 5.3 nedan kan det ses att positionen som syns inte ändras, GPS-enheten uppdaterar inte positionen då den tappar signal. Utan den sista GPS positionen som enheten gav systemet ligger kvar och loggas om och om igen p.g.a. hur koden är skriven. GPS-enheten uppdaterar inte positionen för den kan inte hitta en ny pålitlig position. När signalen återfinns igen kommer positionen tillbaka till rimliga värden. Detta indikeras väl av punkternas färgkodning. Tittar man däremot på de andra positionsbilderna kan man se att de andra sensorsystemen fungerar korrekt (inom sin felmarginal) utan GPS mottagning.



Figur 5.3 - Rak körning med GPS-antennen urskruvad.

5.1.2 Öglekörningar

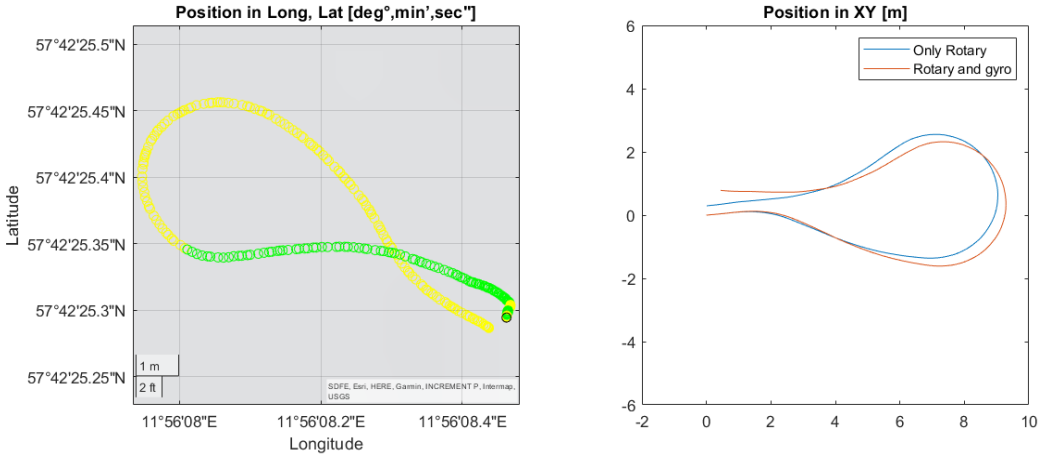
Öglekörningen skiljer sig från den raka körningen då startvinkeln inte korrigerats i graferna. Oavsett vilken startvinkel som används skall farkosten återvända till samma punkt som den startades från. I figur 5.4 nedan visas en körning med samtliga system ostörda.



Figur 5.4 - Öglekörning med ostörd GPS.

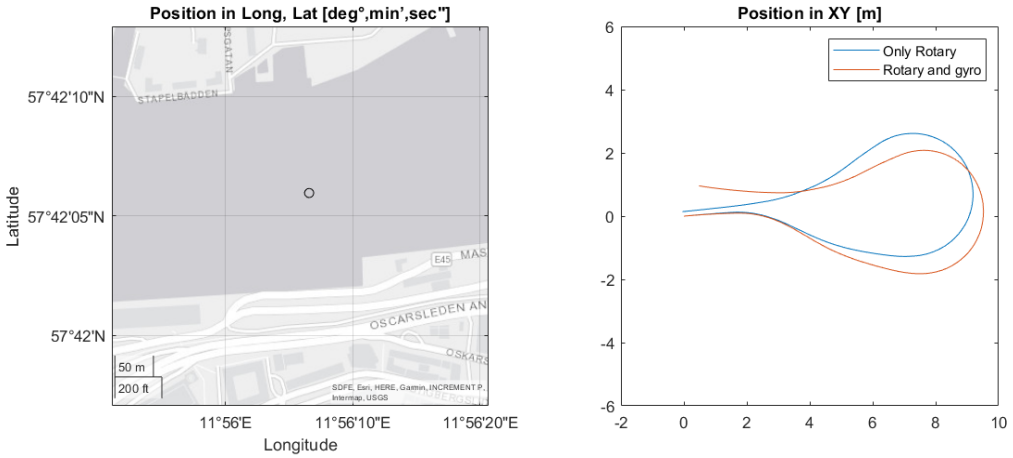
Noterbart i denna graf är att själva färdvägen inte är lätt att kommentera på. Vid närmare granskning har cirkelbågen en diameter på ungefär 4 m, men som synes i senare grafer så ligger inte bågen alltid på samma plats. GPS:en påstår att diametern på cirkelkörningen är $3,7\ m$ och rotationsgivarna och gyrot påstår $3,8m$ respektive $3,9m$. Däremot kan start- och stoppunkt kommenteras på som direkt kursdeviering då dessa punkter ska sammanfalla oavsett rutt. Rotationsgivarna ger ett bra resultat här och gyrot ger en tillförlitlig position med de felmarginaler som tidigare observerats.

I figur 5.5 nedan kan det observeras att så fort GPS signalen studsar så vandrar positionsbilden något. Detta är som mest tydligt då bilen står still i början och slutet av färden. I denna körning så säger samtliga sensorer att cirkelns diameter är $3,9\ m$.



Figur 5.5 - Öglekörning med foliestörning på GPS-antennen.

Då antennen för GPS:en kopplas ut fås samma positioneringsproblem som tidigare. I figur 5.6 nedan visas hur GPS:en har en felaktig position som inte ligger i närheten av vart körningen ägde rum.

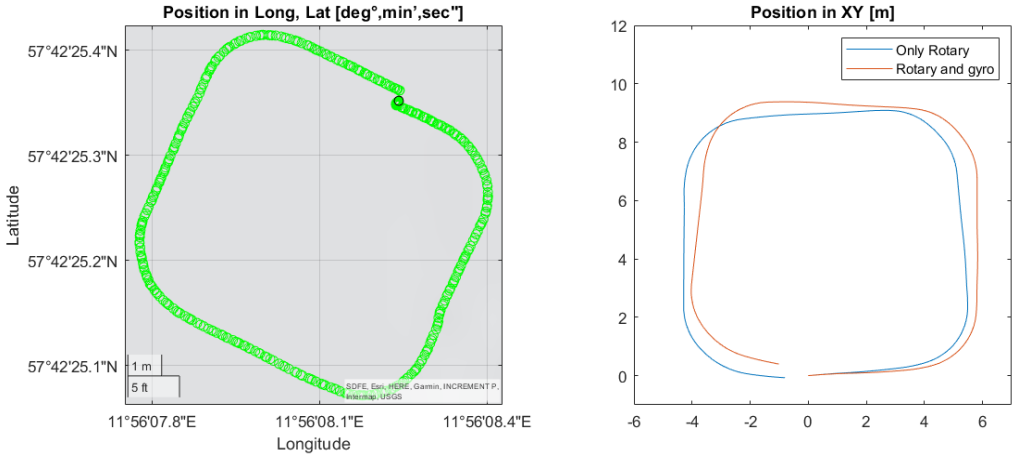


Figur 5.6 - Öglekörning med GPS-antennen urskruvad. Observera att GPS-position tror att den är någon annanstans än vad som är fallet

Som man ser i figur 5.6 syns det tydligt att GPS:en inte kan säga var farkosten har varit någonstans. Diametern på svängen är här uppmätt till ca $3,9\,m$ av både gyro och rotationsgivarna. Rotationsgivarna har en bättre slutposition vid alla körningar om man jämför med gyrot och en liknande position som GPS:en.

5.1.3 Rektangelkörningar

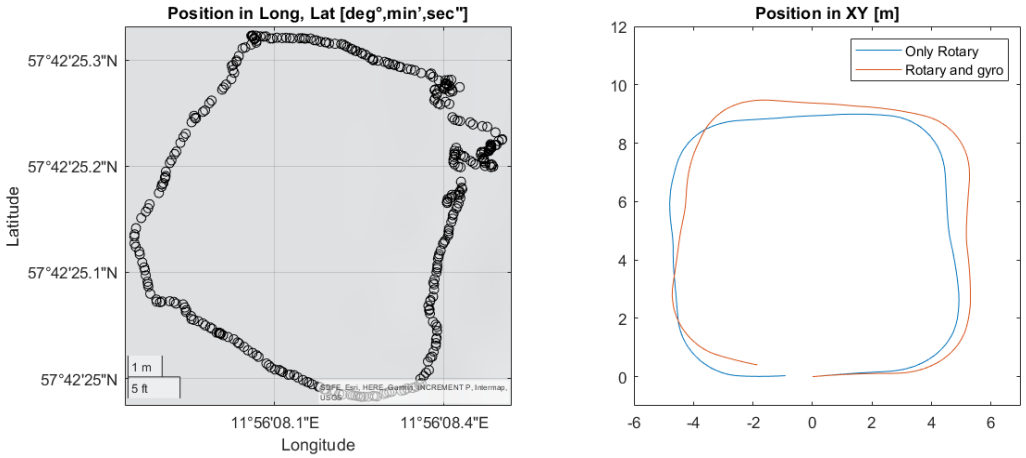
Dessa körningar har, precis som öglekörningen - ej korrigerad startvinkel. På samma sätt förväntas farkosten återvända till sin ursprungliga startpunkt. Fast denna gång från ett annat håll. Figur 5.7 nedan visar ett liknande resultat som de andra körningarna med ostörda system.



Figur 5.7 - Rektangelkörning med ostörd GPS.

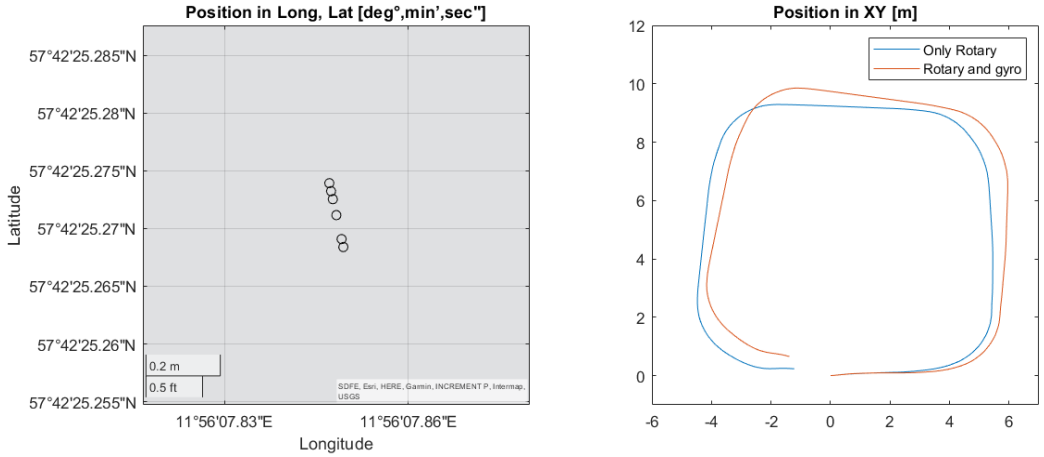
Däremot kan det noteras att GPS positionens accuracy har blivit något sämre. Under tidigare körningar har den varit inom 20 *cm* (blå) medans den här stigit en aning till inom 30 *cm* (grön).

Under folietestet är positioneringen dålig vilket kan ses både genom färgindikering samt genom GPS positionerna i figur 5.8 nedan. Notera även att uppdateringsfrekvensen hos de loggade positionerna till synes är färre.



Figur 5.8 - Rektangelkörning med foliestörning på GPS-antennen.

Med urskruvad antenn ses resultat liknande de från de tidigare testerna. I figuren nedan kan även en vandrande deviation observeras på GPS:en. Orsaken till denna deviation är okänd men har inget att göra med den kurs som körts.



Figur 5.9 - Rektangelkörning med GPS-antennen urskruvad.

De andra sensorbilderna ligger fortfarande inom de felmarginaler som tidigare påpekats. Resultaten från rektangelkörningen är konsekvent. Både rotationsgivarna och gyrot har liknande mönster och i inget av fallen så sticker gyrot iväg som har synts i de tidigare figurerna.

5.2 Övergripande Resultat

Genom alla körningar har en deviation i position på max $\pm$ 1 m varit närvarande i de två rotationsbaserade positionssystemen. I vissa fall är det svårt att kommentera på grund av de förutsättningar som funnits vid arbetets gång. Man kan däremot se att dessa positionsbilder är helt oberoende av en GPS signal.

Den körningen, förutom raka körningar, som visar mest liknande resultat varje gång är rektangelkörningen. Det går tydligt att se mönster hur farkosten har färdats och slutar på liknande position varje gång. Det finns dock även vissa konsekventa delar i öglekörningen med tanke på hur svängens diameter är liknande över alla körningar.

Den GPS som använts vid dessa körningar har visat på en bättre position än utlovat, i vissa fall även då den störts ut. Man kan däremot observera att den position som den producerar, även om "accuracyn" är förhållandevis hög, ej ligger rätt placerad med tanke på beskriven färdväg.

6. DISKUSSION

Nedan tas de problem som har uppkommit upp. Det redogörs även för vissa begränsningar som framkommit i de genomförda undersökningarna som hade kunnat fått mer fokus och uppmärksamhet om tidsramen tillät. Här tas även etik och hållbar utveckling upp och avslutas sedan med en slutsats där en sammanfattning av arbetet görs.

Enligt syftet skulle arbetet avhandla en aktuell problembild från dagens marknad. Detta har genomförts i form av litteraturstudie, enkät och rådfrågning av fältexperter. Arbetet skulle också redogöra för ett proof of concept för positioneringstjänster vilket har utförts om än med något ofullständig modell. Arbetet behövde avrundas tidigare än hoppats vilket lämnade ute vissa önskvärda funktioner. Överskådligt har arbetet varit lyckat även om den slutgiltiga modellen som idag finns behöver ses över innan större framsteg kan göras. Ett par av de felkällor som observerats beskrivs nedan.

6.1 Signalproblem

De olika sensorerna som använts i arbetet har alla olika begränsningar. Vissa lättare att mäta och upptäcka och därmed även korrigera än andra.

6.1.1 Rotationsgivare

De rotationsgivare som använts har haft en upplösning på $400\ pulser\ per\ varv$. Vid vanlig inkoppling fås då en upplösning på $0.9\ grader$ och med den farkosten som använts som har en hjuldiameter på ca $15\ cm$ ges en puls var $1.2\ mm$. En begränsning som upptäcktes här är att vissa pulser går förlorade vid snabb rotation då den Arduino UNO som från början användes inte är tillräckligt snabb för att kunna upptäcka alla pulser. En uppgradering som gjordes var att aktivera hårdvaruinterrupts på de pinnar som signalerna kommer, exekveringen anpassades även för att vara så liten som möjligt i interruptrutinerna. För att ytterliggare säkerställa att tappade pulser inte var ett problem så dedikerades en mikrokontroller för att endast sköta dessa rotationsgivare och skriva dessa på en kommunikationsbuss på begäran från en masterenhet (i detta fall en Arduino MEGA). Om ett nytt arbete genomförs med högupplösta rotationsgivare rekommenderas en högre klockfrekvens än vad dessa Arduinos kan frambringa (16 MHz) för att säkerställa att inga pulser missas. I de tester som genomförts har rotationsgivarnas positionsbild devierat från målet med ca $\pm\ 1\ m$. Dessa deviationer ligger inom den uppsatta målbilden, men endast nätt och jämt. Det finns även en observation att vid längre körningar kommer denna sensordrift byggas på och positionen därmed blir mindre tillförlitlig. Från de insamlade data har inget tydligt deviationsmönster framträtt och har därför inte heller kunnat filtreras bort på ett effektivt sätt. Filtrering borde däremot vara möjlig även om detta arbete ej kom så långt.

6.1.2 Gyroskop

Under arbetet användes ett gyroskop från Adafruit (LSM6DSO32) som hade 6 DOF, acceleration i x, y, z led samt rotation i x, y, z led. Upplösningen var inställbar och enligt rekommendationer användes en lägre upplösning för den skulle teoretiskt sett vara mer exakt. I de tester som genomförts verkar det dock som att gyroskopet stundvis är opålitligt. I vissa tester är gyrot helt pålitligt medans i andra inte alls. I den undersökning som gjorts framträder det att gyron i allmänhet har en tendens att drifta i sina mätvärden. Speciellt om mättiden sträcker sig över en längre tid [28]. Dessa drifter går troligtvis att kompensera för men är något som återigen har begränsats av den tidsram som arbetet utförts inom.

6.2 Signalproblematik med GPS

Nedan följer observationer som gjorts, känd problematik sedan tidigare samt nyfunnen problematik som den undersökning som genomförts har lagt ljus på. Det är dessa problem som arbetet hoppades på att kunna tackla eller bistå med en lösning till. Det tål även att påpekas att i testerna ovan sammanfaller inte rotationsgivarnas axel med monteringen av GPS enheten. Denna positionsoffset har i relation till de andra problemområden som presenterats verkat som en bagatell och därför inte åtgärdats.

6.2.1 Studsning

Som arbetet tidigare nämnt är det främst i stadsmiljöer som problemet att signalerna studsar finns. Signalerna studsar mellan husfasader eller dylikt och gör positionen mindre pålitlig. Ett första steg till en lösning hade kunnat vara ett system likt det som här försökt utvecklas, som kan bistå GPS signalen då den devierar med en opartisk position. När sedan GPS signalen återvänder så finns en annan uppmätt position tack vare det system som då utvecklats.

6.2.2 Signalförseningar eller bortfall

Ett annat, mindre känt problem som inte är lika utforskat är förseningar av signalen. Som arbetet tog upp i delkapitel 2.4.2 Felkällor, så påverkas signalen olika mycket av jordens atmosfär beroende på tryck, luftfuktighet och dylikt vilket kan påverka hur lång tid det tar för signalen att nå fram. Man kan också observera att antalet satelliter som har en stabil uppkoppling varierar vid tjocka molntäcken. Detta är ett mindre trivialt problem att lösa och kräver mer efterforskning innan man kan komma fram till ett komplett svar.

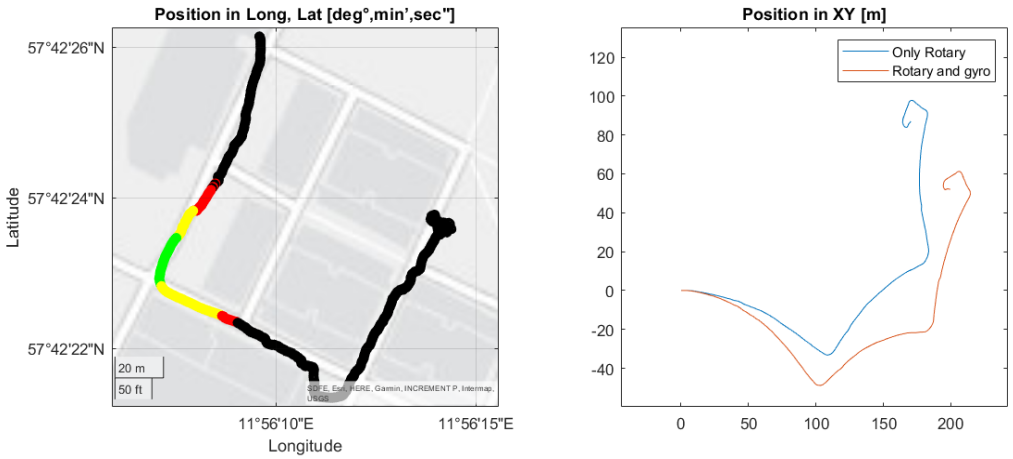
I vissa fall kan GPS signalen även helt fallera. Då signalen är helt borta kan ett liknande system till det arbetet beskriver bistå med en andra positionsuppfattning tills GPS signalen återvänder. Enligt de observerade signaldrifterna blir detta system bli bristfälligt med tiden. Något som behöver åtgärdas om GPS:en har ett längre signalbortfall.

6.3 Körningarna

Den modell som framtagits utgick ursprungligen från två estimeringar. En hade antaganden som förutsatte att vinkeln mellan nästa punkt och den föregående inte översteg 10 grader medans den andra skulle hantera svängar över 10 grader. Ett problem som dök upp var att då den senare estimeringen skulle användas behövde samplingsfrekvensen dubblas. Detta visade sig vara ett problem som ej gick att lösa med den sensoruppsättningen som användes.

De körningar som utfördes var ämnade att testa de olika delarna i systemet för att se så de båda estimeringarna fungerade som tänkt. Något som alltså inte fick utrymme då endast en estimering kunde användas. Det tål även att nämnas att den plattform som testerna utfördes på inte kunde svänga så snävt att man pålitligt kunde trigga den senare estimeringen. För att få en större svängradie mellan samplingspunkterna behövde plattformen köra fortare men denna förhöjda hastighet medförde även sämre bild från de övriga positionerna. Det var helt enkelt onödigt och omständigt att ha båda estimeringarna närvarande i systemet.

En annan körning värd att belysa som ej är del av de test som utfördes är en rutt som uppmättes på tillbakavägen från testområdet. Rutten gick runt ett kvarter, över ojämn mark och vid ett tillfälle under en gångbro. Se figur X nedan.



Figur 6.1 - Extra rutt på väg tillbaka från testområdet. Rutten går över både kullersten och under gångbro.

De observationer som tidigare belysts av S3 projektet med studsning och en bättre mottagningsbild utan husen kan även ses här. Så fort farkosten kommer ut från mellan husen som är uppmålade i mörkt grått på kartan, kan en ökad positionsprecision ses. Längst ner i GPS rutten kan en deviation ses då farkosten åker under en gångbro. Rotationsgivarnas rutt visar stor drift som framstår vid längre färd, men denna rutt går som sagt över mycket ojämn väg vilket bidrar till en mer oklar positionsbild från rotationsgivarna.

6.4 Frågeställningar

I delkapitel 1.4 presenterades projektets frågeställningar.

- Kan GPS positionering stöttas upp av externa system?

Absolut. Detta är ett logiskt steg i rätt riktning mot bättre positionering - något som är extremt viktigt i autonoma fordon.

- Hur kan mätbara data visa att positionen förbättrats?

För detta krävs ett externt opartiskt system. Ett förslag på hur detta kan gå till tas upp i kapitel 7.

- Vilka förbättringsmöjligheter finns det för positionering i anseende av autonomitet?

Efterforskningar i alternativa positionstjänster behöver mer tid för att på ett säkert sätt kunna implementeras i autonoma system. En mer tillförlitlig dynamisk GPS position vore önskvärd men under detta arbetes gång har dock en fråga väckts huruvida bandbredd är ett problem i ett mer precist positionssystem. Om bandbredd är ett problem kan mer omfattande positioneringssystem vara en viktig del i utvecklingen mot autonoma positioneringar.

- Hur kan noggrannheten i ett fordons position förbättras med hjälp av Sensor Fusion?

Sensor Fusion är ett självklart val i det enorma arbete som står för dörren. Den utvecklade plattformen är helt baserad på att flera system ska kunna samarbeta för att få en bättre och mer tillförlitlig position.

6.5 Etik & hållbar utveckling

Det arbete som utförts har inga negativa påverkningar i avseende på hållbar utveckling. Det finns däremot utrymme för en etisk reflektion. The Trolly Problem har säkert alla sett någon variant på. Problemet är ett etiskt dilemma där en vagn är på väg mot fem personer på ett spår. Personen som beskådar detta kan göra ett aktivt val att omdirigera vagnen på ett spår där det endast står en person på och på så sätt skona livet på de övriga fem. Det svåra med problemet är att om personen väljer att dra i spaken har den gjort ett aktivt val vars konsekvenser har kostat en oskyldig människas liv. En människa som inte hade omkommit om personen inte gjort detta val. Detta problem har fått ny luft under vingarna då självkörande bilar börjar presenteras på marknaden. Kan man lita på en dator att värdera liv på detta sätt? Vem vill vara del av att programmera en bil som är tvungen att ta dessa beslut? Den sista frågan här är den som kanske är mest aktuell för detta arbete. Genom att tillgodose en position som detta fordon baserar sina beslut på är man indirekt inblandad i de konsekvenser som kan dyka upp. Eller ännu värre - de konsekvenser som kan uppstå då detta positioneringssystem visar en felaktig position.

7 Slutsats och framtida arbeten

Arbetet som utförts har i efterhand observerats ha en kort tidshorisont. Ett så omfattande ämne som positionering kräver mer resurser och framförallt mer tid. Det är inte förvånande att de företag som arbetar med detta idag har stora arbetslag inom flera olika fält. De resultat som presenterats i rapporten är en god start på ett stort positioneringsarbete, men önskvärt hade varit om viss bättre filtrering hunnits med.

7.1 Rekommendationer

Det finns en del saker som behöver förbättras för att kunna få ett bättre resultat än det som redovisas i denna rapport. Nedan följer en punktlista över de saker som anses kan behöva förbättras för ett fortsatt arbete.

- Det rekommenderas att använda en bil med helt släta däck.
    - Gärna hårda då mjuka eller flexande däck medför att rotationsomkretsen ändras. Omkretsen på däcken är viktig för skalningen av modellen och en misstanke som fanns var att i kurvtagning så ändras omkretsen då belastningen blir olika. Ett mjukt däck kommer då kräva en dynamisk omkrets beroende på belastning vilket kan vara en ytterligare felkälla.
- Vid backning kan gyroskopet kräva viss uppsikt.
    - I alla fall om beräkningarna utförs som de gör i denna rapport. Här används en viss vinkel vid körning men vid backning blir denna då 180 grader åt andra hållet. Under de tester som genomförts här ingår dock endast körning framåt så detta var ej ett aktuellt problem.
- Implementera en magnetometer som kan hjälpa med bestämning av startkurs.
    - Detta är ett bra sätt att få de olika systemen vinklade åt samma håll men kanske också kan fungera som substitut till gyrot.
- Basera inte modellen på ett CANbus-system utan tidigare CANbus erfarenheter.
    - Under framtagningen av plattformen lades mycket tid på att få CANbus-systemet att fungera som tänkt. Detta är ett extra steg som är något onödigt om man inte har tidigare erfarenheter av dessa system.
- Använd en snabbare mikrokontroller.
    - De Arduinos som användes hade en klocktakt som inte kunde matcha de krav som behövde ställas på systemet.
- Lägg inte för mycket tid på den matematiska modellen innan provkörning har börjat.
    - Det är lätt att vilja starta igång men det är lika svårt att veta vad som kommer krävas av systemet. Ta fram sensorbilderna först och försök utvärdera vilka fel de har. Sedan sammanställ bilderna i en modell och fortsätt med testningen.

Ett problem som behöver tacklas om arbetet ska fortsätta är implementationen av ett opartiskt och exakt positionssystem som kan användas under testning. Detta för att kunna ha ett referensvärde på samtliga punkter under testning, inte bara start och stopp. Inför utvecklingen av plattformen diskuterades detta men valdes att inte lägga tid på då man visste redan från start att arbetet skulle vara omfattande och tidspressat. Ett förslag på hur ett sådant system kan vara utformat är som lyder: först och främst - se till att underlaget som körs på är halkfritt och helt plant. Storleken på ytan är viktig men med sig medför den också andra utmaningar. De positioneringsfel som uppkommer i detta system blir värre med tiden och därför skulle en större yta vara att föredra. För att bestämma positionen i detta system skulle en högt monterad kamera kunna användas tillsammans med ett verkligt koordinatsystem på marken. Alternativt att den lasermätare som använts kan utnyttjas tillsammans med rigida kanter för att bestämma en X och Y koordinat. Man kan då under ruttens gång stanna bilen, ta en kontrollposition och fortsätta köra. Detta hade underlättat i detta arbete då kommentarer och observationer under ruttens gång hade varit lättare att göra.

KÄLLFÖRTECKNING

[1] G. Xu och Yan. Xu, GPS, 3e uppl. Tyskland: Springer-Verlag Berlin Heidelberg, 2016 [Online] Tillgänglig: https://link.springer.com/book/10.1007%2F978-3-662-50367-6#authorsandaffiliationsbook. [Acc: 19-april-2021]

[2] "What is GNSS?," European Global Navigation Satellite Systems Agency, 19-Nov-2020. [Online]. Tillgänglig: https://www.gsa.europa.eu/european-gnss/what-gnss. [Acc: 19-april-2021]

[3] "Självkörande fordon., Självkörande fordon | Finska Lantmäteriverket [Online]. Tillgänglig: https://www.maanmittauslaitos.fi/sv/forskning/teman/sjalvkorande-fordon. [Acc: 19 -april-2021]

[4] "Landespatente". [Online], Tillgänglig : http://www.wolfgang-pfaller.de/landespa.htm . [Acc: 20-maj-2021]

[5] R. N. Jazar, "Steering Dynamics", i Vehicle Dynamics: Theory and Application, R. N. Jazar, Red. Boston, MA: Springer US, 2008, s. 379–454 [Online]. Tillgänglig: https://doi.org/10.1007/978-0-387-74244-1_7 . [Acc 20-maj-2021]

[6] Y. Wu, "Versatile Land Navigation Using Inertial Sensors and Odometry", Karlsruhe, sep. 2014 [Online]. Tillgänglig : https://arxiv.org/ftp/arxiv/papers/1409/1409.1078.pdf. [Acc 20-maj-2021]

[7] Y. Wu, J. Kuang, och X. Niu, "Wheel-INS2: Multiple MEMS IMU-based Dead Reckoning System for Wheeled Robots with Evaluation of Different IMU Configurations", arXiv:2012.10593 [cs], dec. 2020 [Online]. Tillgänglig: http://arxiv.org/abs/2012.10593 .[Acc 20-maj-2021]

[8] J. Borenstein, H. R. Everett, och L. Feng, "Where am I? Sensors and Methods for Mobile Robot Positioning", The University of Michigan, Michigan, 1996 [Online]. Tillgänglig: http://www-personal.umich.edu/~johannb/Papers/pos96rep.pdf . [Acc: 20-maj-2021]

[9] "GPS.gov: GPS Overview". [Online]. Tillgänglig: https://www.gps.gov/systems/gps/ . [Acc: 20-maj-2021]

[10] ”What is GNSS?”, 01-mar-2016. [Online]. Tillgänglig: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss . [Acc: 20-maj-2021]

[11] ”Difference Between GPS and DGPS (with Comparison Chart)”, Tech Differences, 21-okt-2017. [Online]. Tillgänglig : https://techdifferences.com/difference-between-gps-and-dgps.html. [Acc: 20-maj-2021]

[12] ”Sources of Errors in GPS”, KOWOMA GPS. [Online]. Tillgänglig: https://www.kowoma-gps.de/errors/ . [Acc: 20-maj-2021]

[13] ”Sam Wormley’s GPS Errors & Estimating Your Receiver’s Accuracy”, 21-okt-2013. [Online]. Tillgänglig: https://web.archive.org/web/20131021232453/http://edu-observatory.org/gps/gps_accuracy.html . [Acc: 20-maj-2021]

[14] ”ZED-F9P u-blox F9 high precision GNSS module”. [Online]. Tillgänglig: https://www.u-blox.com/en/docs/UBX-17051259 . [Acc: 21-maj-2021]

[15] Richard. B. Langley, ”Dilution of Precision”, University of New Brunswick, apr. 1999 [Online]. Tillgänglig: http://gauss.gge.unb.ca/papers.pdf/gpsworld.may99.pdf . [Acc: 21-maj-2021]

[16] ”Arduino - ArduinoBoardUnoSMD”. [Online]. Tillgänglig : https://www.arduino.cc/en/Main/ArduinoBoardUnoSMD . [Acc: 21-maj-2021]

[17] ”Arduino - ArduinoBoardMega”. [Online]. Tillgänglig : https://www.arduino.cc/en/Main/arduinoBoardMega/ . [Acc: 26-maj-2021]

[18] ”Encoder Basics”. Hengstler [Online]. Tillgänglig : https://ics.as/wp-content/uploads/2013/05/Hengstler-Encoder-basics.pdf. [Acc:21-maj-2021]

[19] N. Ferrete Ribeiro och C. P. Santos, ”Inertial measurement units: A brief state of the art on gait analysis”, i 2017 IEEE 5th Portuguese Meeting on Bioengineering (ENBENG), 2017, s. 1–4, doi: 10.1109/ENBENG.2017.7889458.

[20] M. Verma, T. Dehaeze, G. Zhao, J. Watchi, och C. Collette, ”Virtual sensor fusion for high precision control”, Mechanical Systems and Signal Processing, vol. 150, mar. 2021, doi: 10.1016/j.ymssp.2020.107241 .

[21] ”CAN in Automation (CiA): History of the CAN technology”. [Online]. Tillgänglig: https://www.can-cia.org/can-knowledge/can/can-history/ . [Acc: 21-maj-2021]

[22] L.-B. Fredriksson, ”A CAN Kingdom Rev. 3.01”. KVASER AB, 01-apr-1995 [Online]. Tillgänglig: https://www.kvaser.com/wp-content/uploads/2014/08/cankingdom301p.pdf . [Acc: 21-maj-2021]

[23] ”Knowing Where You Are”, Building Robots with LEGO Mindstorms NXT, s. 249–263, jan. 2007, doi: 10.1016/B978-159749152-5/50018-8.

[24] H. Hashem och A. Idoffson, ”Construction and programming of an RC car running CAN”, mar. 2020.

[25] ”I2C tutorial”. [Online]. Tillgänglig: https://www.robot-electronics.co.uk/i2c-tutorial . [Acc: 26-maj-2021]

[26] J. Valdez och J. Becker, ”Understanding the I2C Bus”. Texas Instrument, 01-juni-2015 [Online]. Tillgänglig: https://www.ti.com/lit/an/slva704/slva704.pdf . [Acc: 26-maj-2021]

[27] ”A Tutorial and Elementary Trajectory Model for The Differential Steering System”. [Online]. Tillgänglig: http://rossum.sourceforge.net/papers/DiffSteer/ . [Acc: 21-maj-2021]

[29] L. Shi, Y. He, B. Li, Y. Wu, Y. Huang and T. Cheng, "On-Line Measurement of Dynamic Tilt Angle by Compensating Gyroscope Drift Error," in IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 9, pp. 3244-3252, Sept. 2019, doi: 10.1109/TIM.2018.2878073 .

BILAGOR

Bilaga 1 - Samtliga körningar
Bilaga 2 - Kod för GADA Slav V6
Bilaga 3 - Kod för GADA Master V6
Bilaga 4 - Kod för GADA MATLAB V7
Bilaga 5 - Marknadsundersökning

Bilaga 1 - Samtliga Körningar

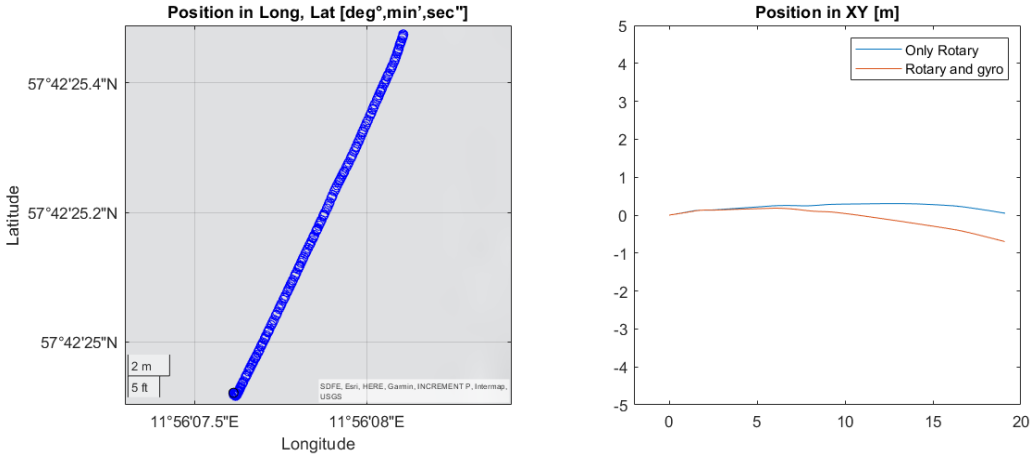
I denna bilaga kommer graferna från samtliga testkörningar presenteras. Inga kommentarer eller observationer kommer göras utan bilagans primära uppgift är att inte överfylla rapporten med grafer.

Som rapporten nämner har sensorbilderna från rotationsgivare och gyro upprättas så att Rotationsgivarnas start- och stoppunkter infaller med x-axeln. Detta för att på ett objektivt sätt kunna illustrera sensordeviation. Testerna utförs alltid med samtliga system inkopplade, men där GPS-signalen störs ut i olika hög grad. GPS positionens noggrannhet enligt tabell 1.

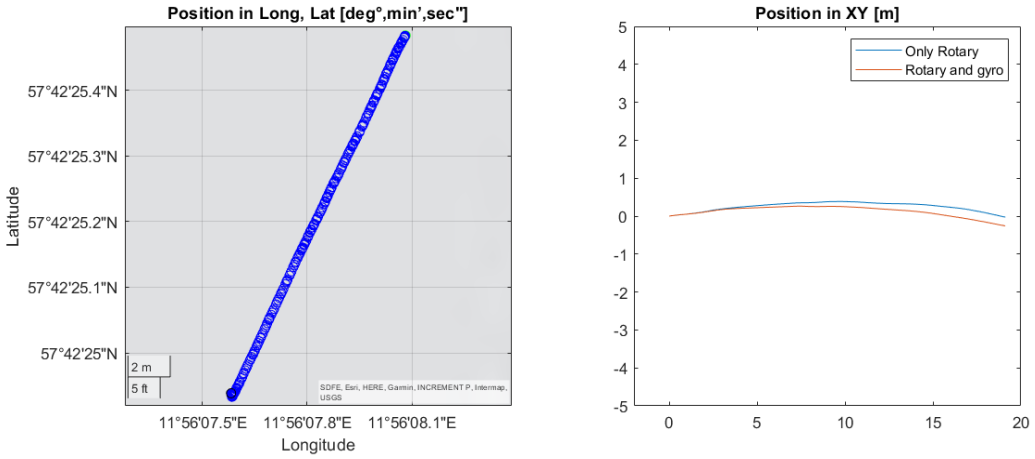
Tabell 1 - Färgschema för GPS positionens tillförlitlighet

Färg	Tillförlitlighet
Svart	Över 50 cm
Röd	40 - 50 cm
Gul	30 - 40 cm
Grön	20 - 30 cm
Blå	10 - 20 cm
Cyan	Under 10 cm

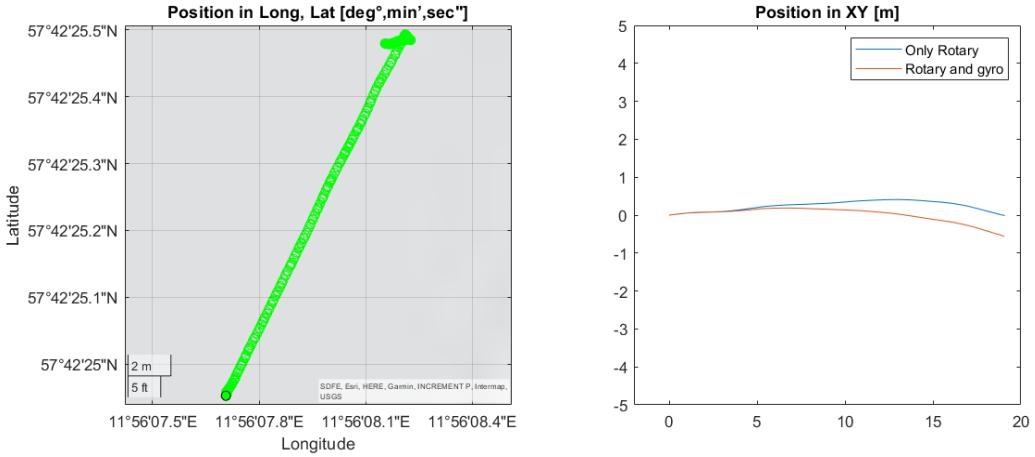
1 Rak körning



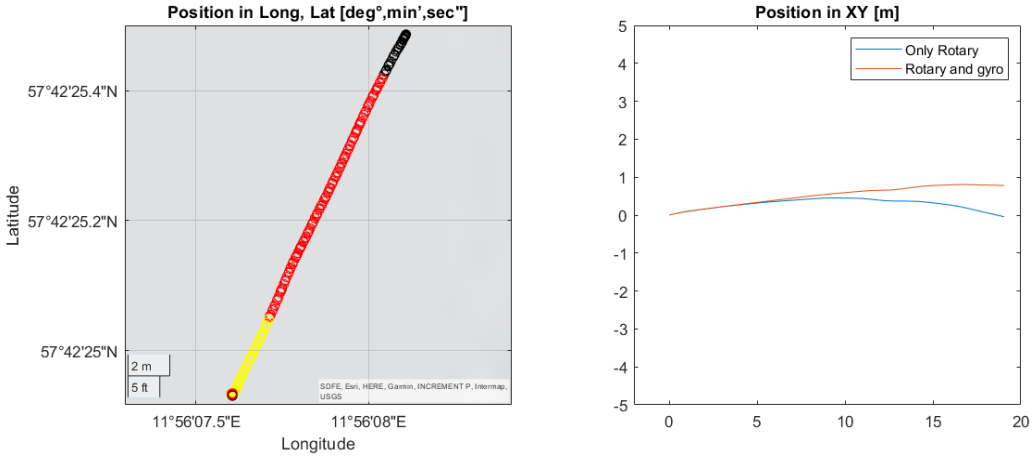
Figur 1 - Rak körning 1 med ostörd GPS



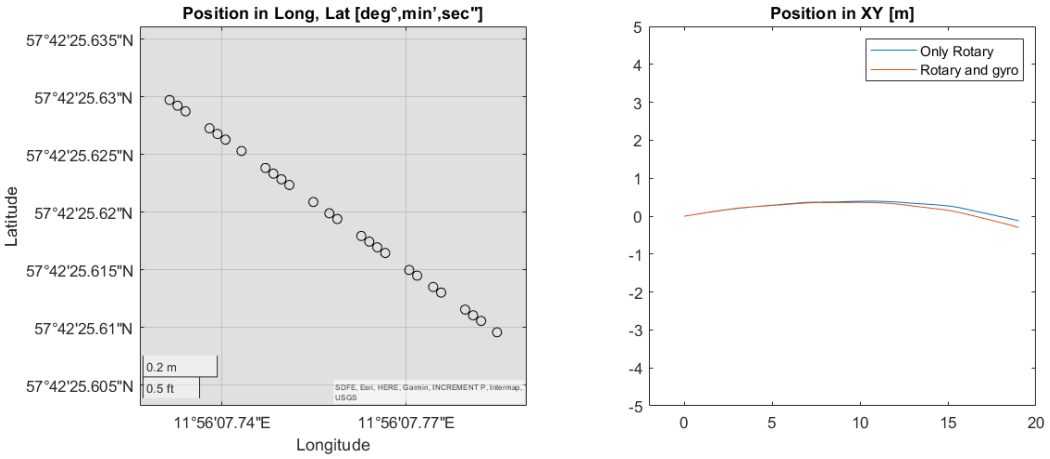
Figur 2 - Rak körning 2 med ostörd GPS



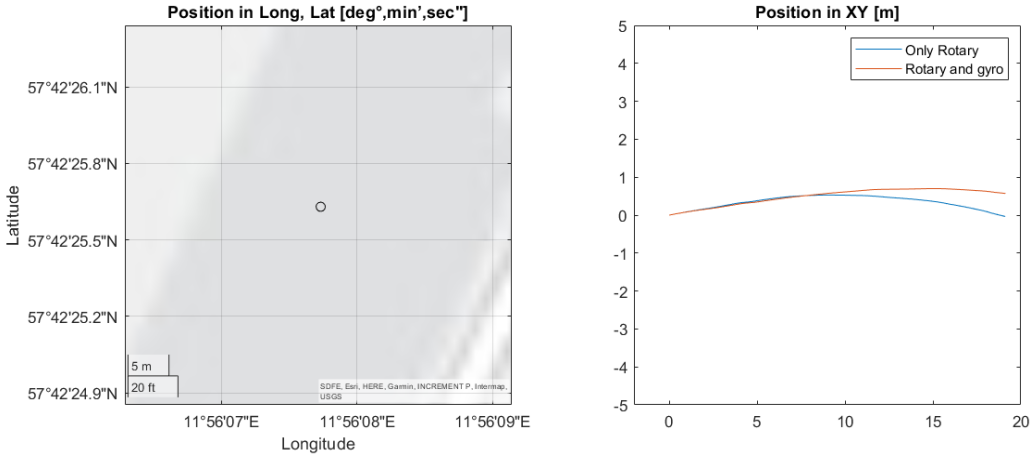
Figur 3 - Rak körning 3 med folie runt antennen



Figur 4 - Rak körning 4 med folie runt antennen

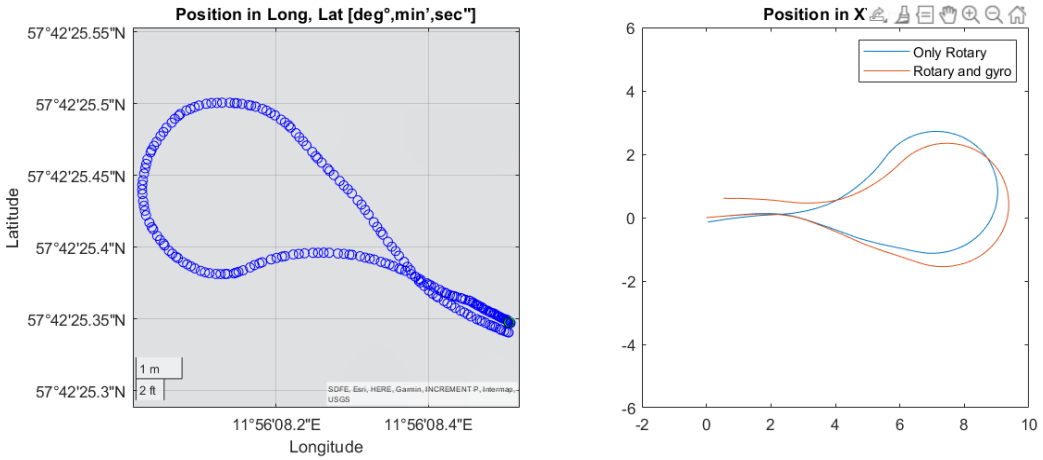


Figur 5 - Rak körning 5 med utskruvad antennen

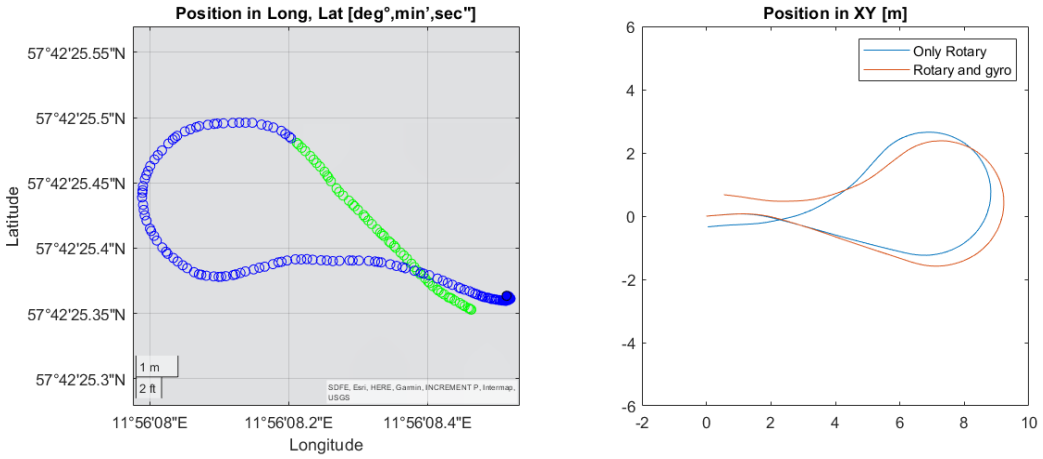


Figur 6 - Rak körning 6 med utskruvad antennen

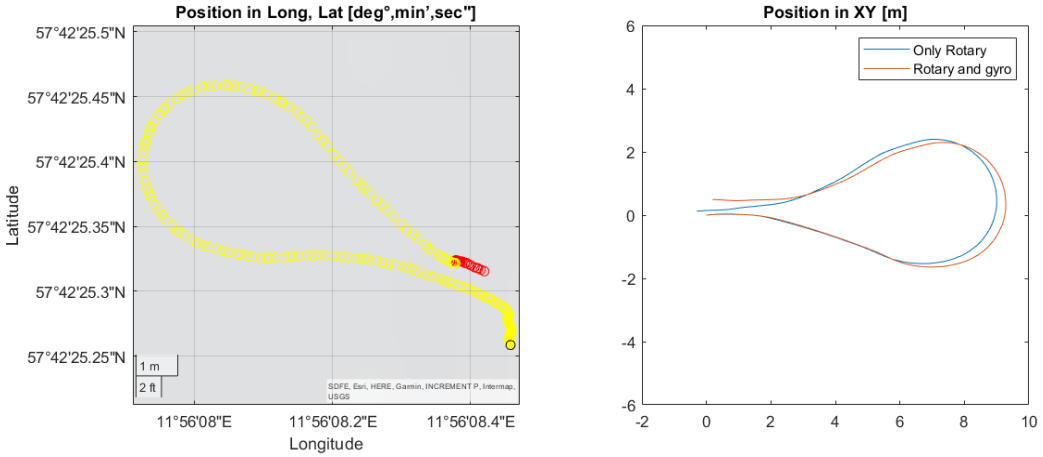
2 Öglekörning



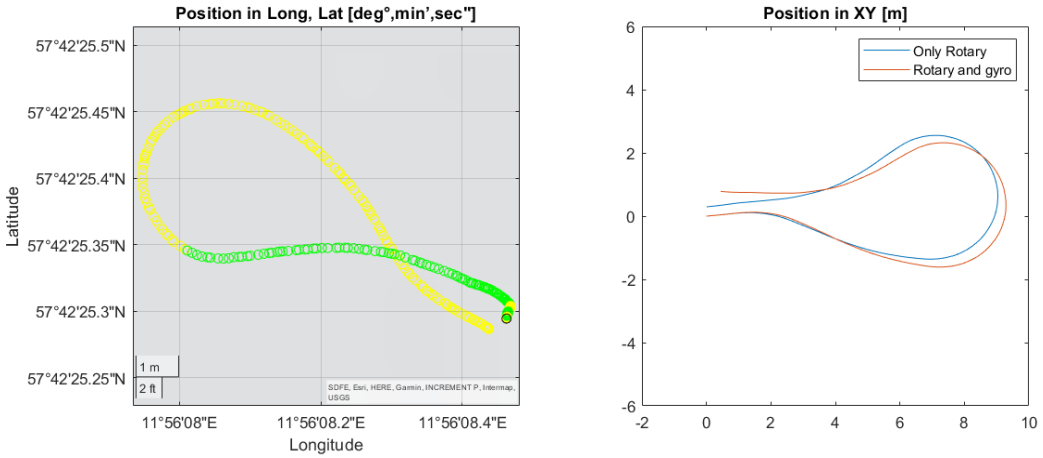
Figur 7 - Öglekörning 1 med ostörd GPS



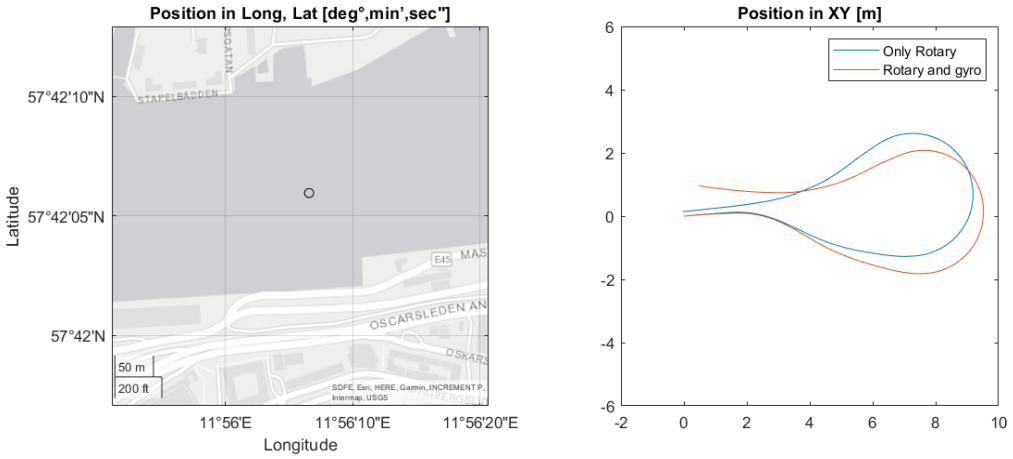
Figur 8 - Öglekörning 2 med ostörd GPS



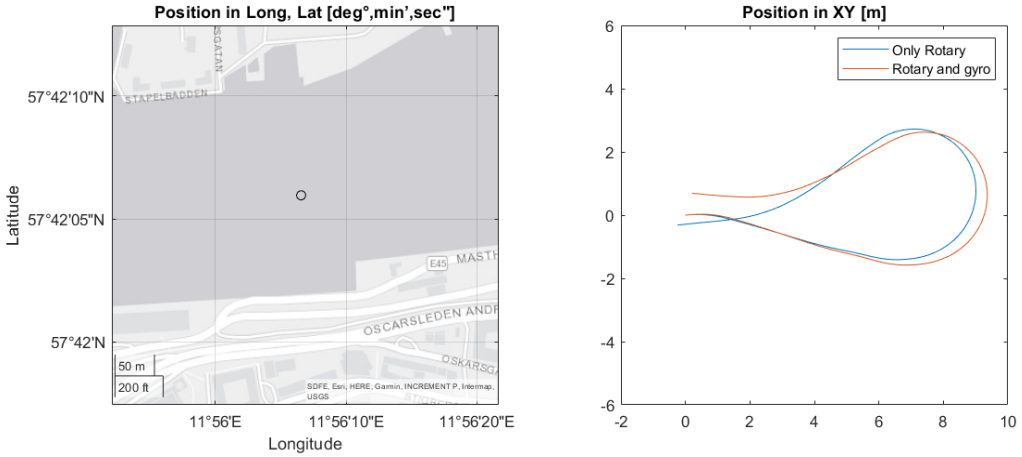
Figur 9 - Öglekörning 3 med folie runt antennen



Figur 10 - Öglekörning 4 med folie runt antennen

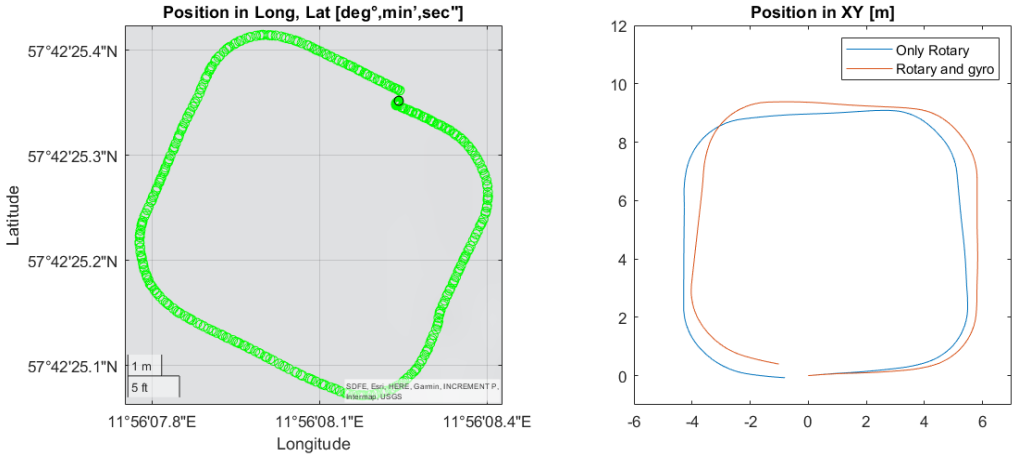


Figur 11 - Öglekörning 5 med urskruvad antennen

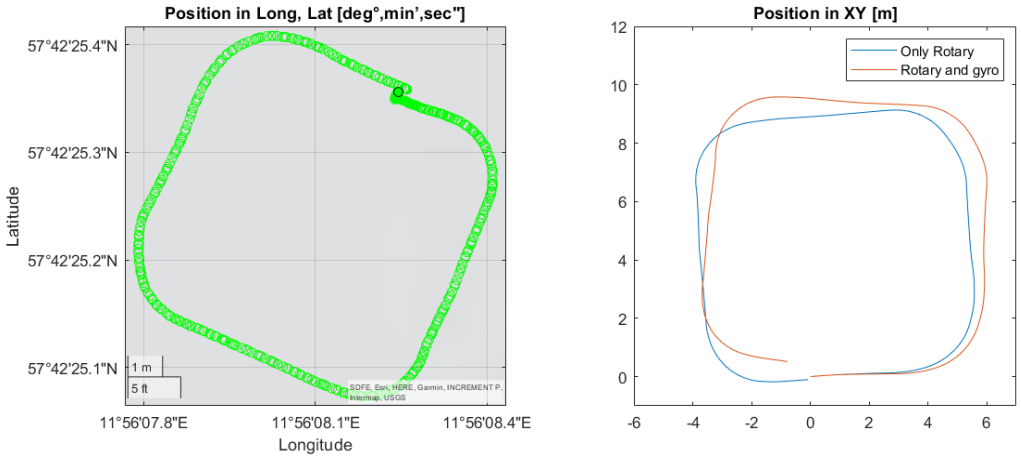


Figur 12 - Öglekörning 6 med urskruvad antennen

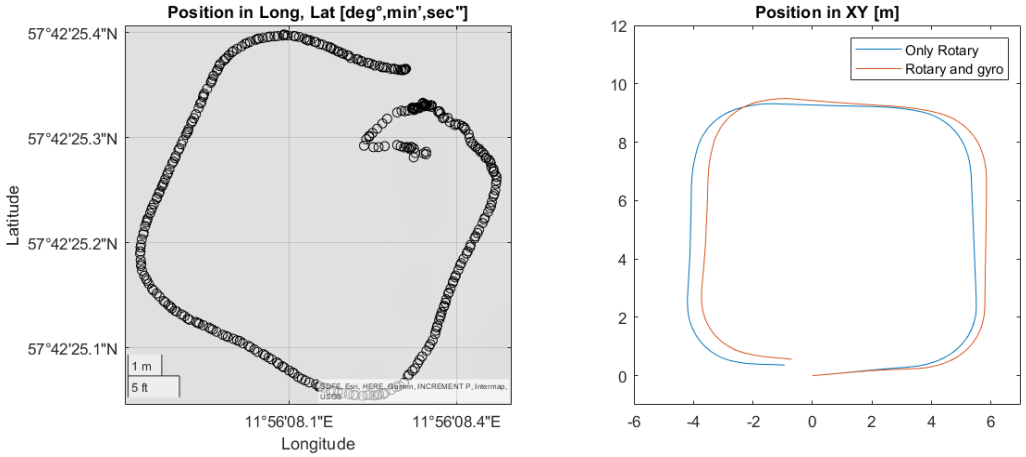
3 Rektangelkörning



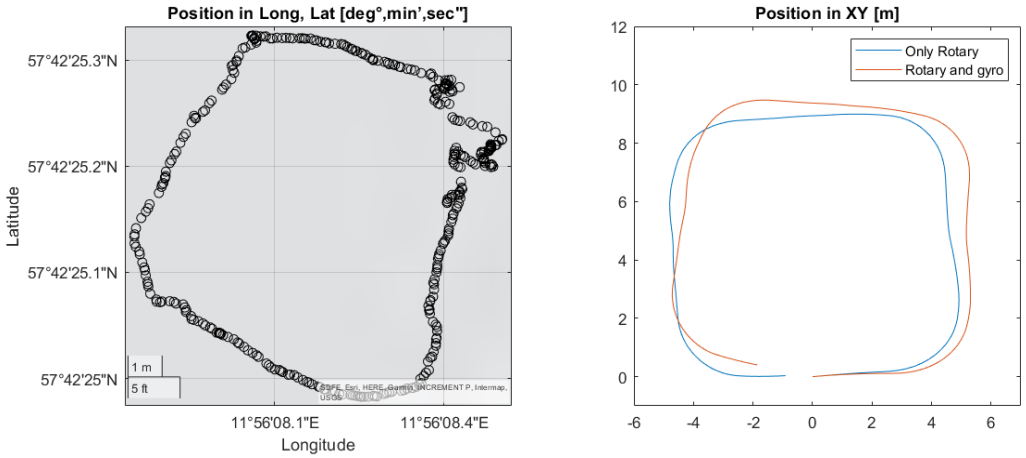
Figur 13 - Rektangelkörning 1 med ostörd GPS



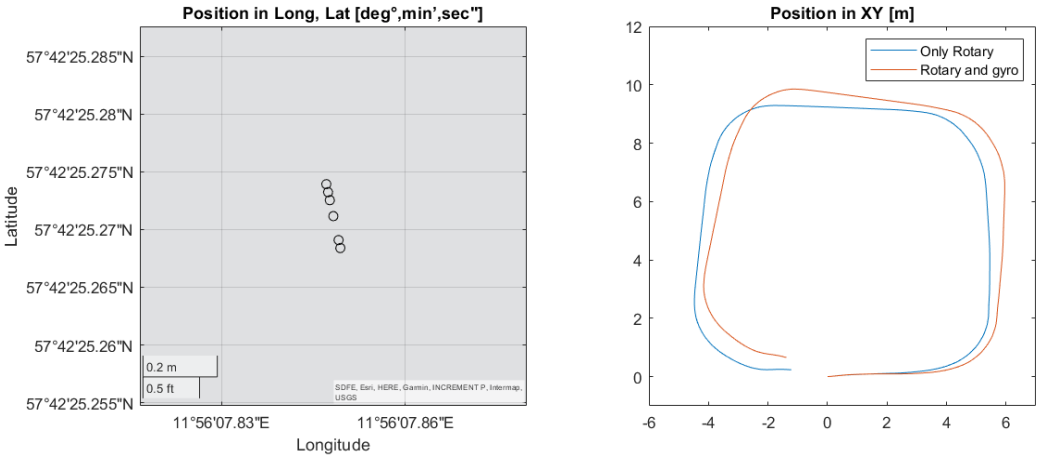
Figur 14 - Rektangelkörning 2 med ostörd GPS



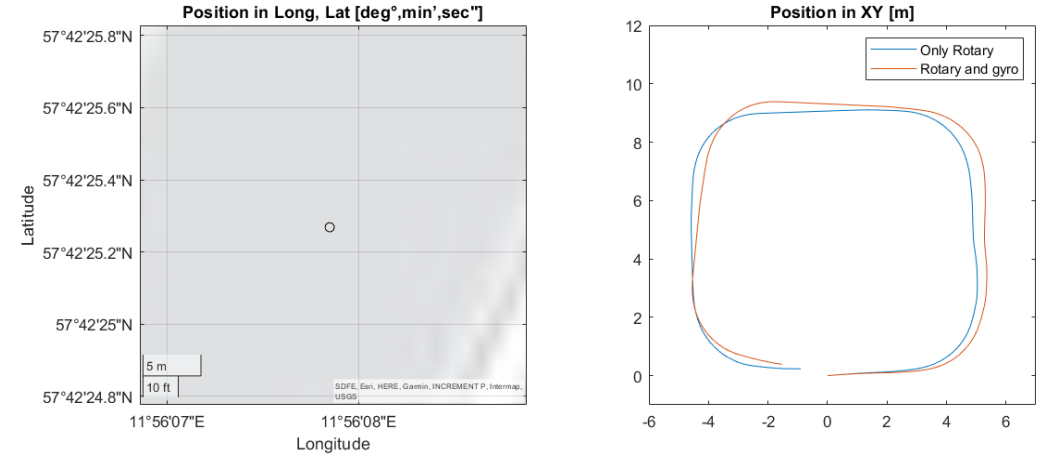
Figur 15 - Rektangelkörning 3 med folie runt antennen



Figur 16 - Rektangelkörning 4 med folie runt antennen

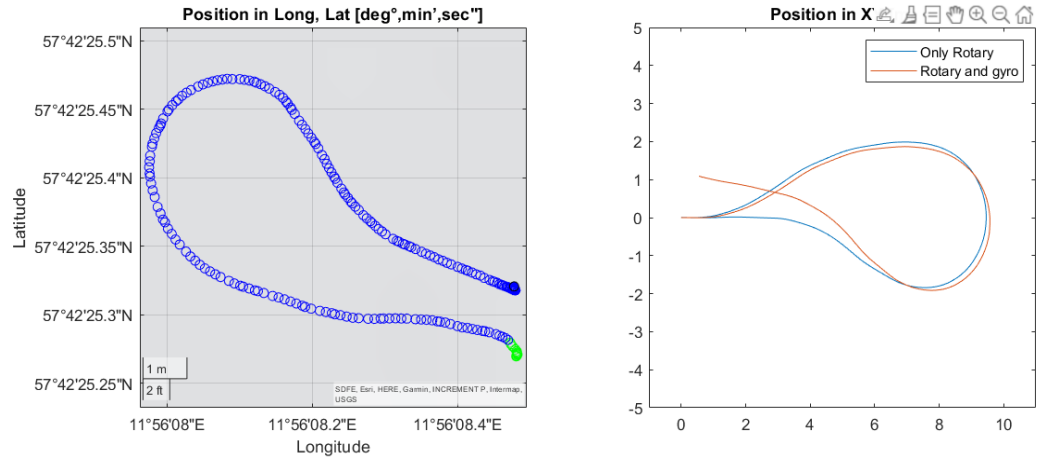


Figur 17 - Rektangelkörning 5 med urskruvad antenn

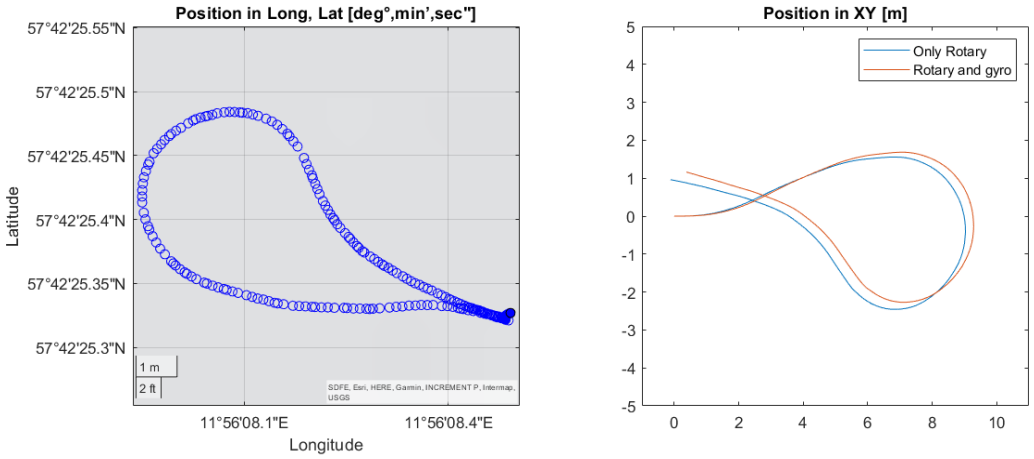


Figur 18 - Rektangelkörning 6 med urskruvad antenn

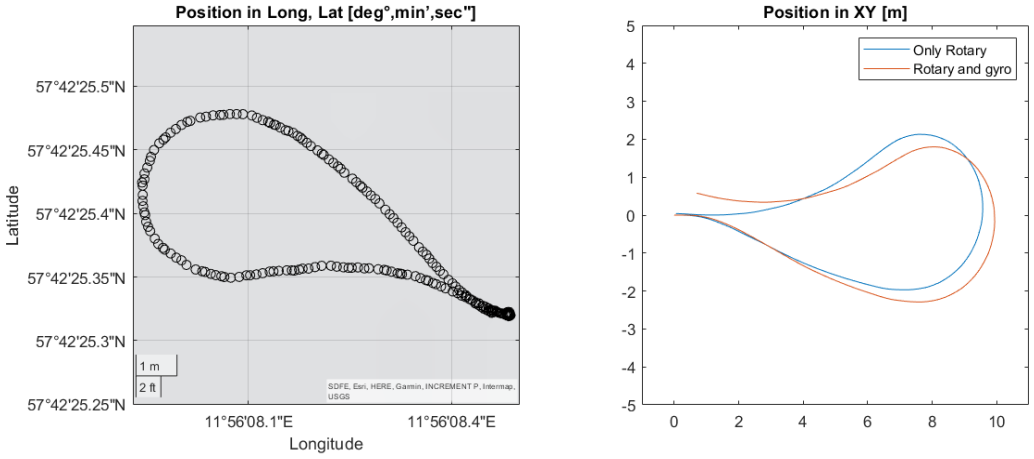
4 Extra körningar



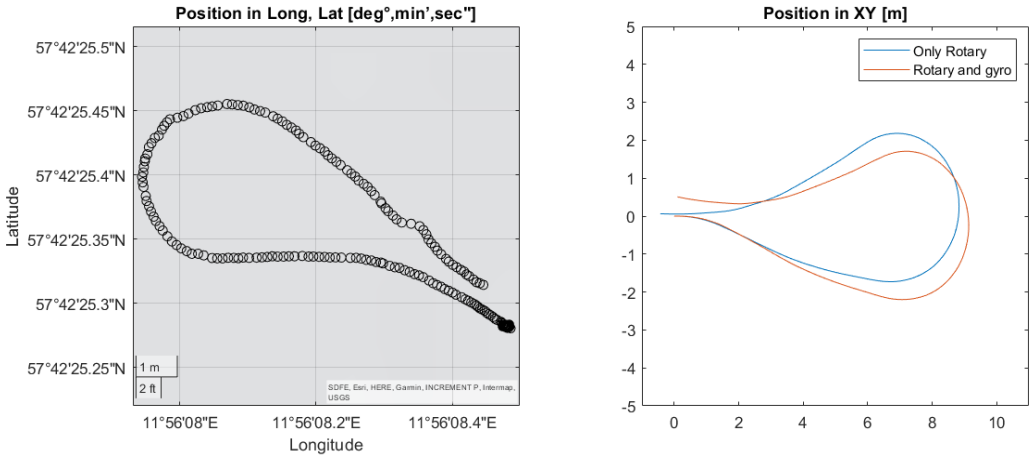
Figur 19 - Extra öglekörning i högervarv, körning 1



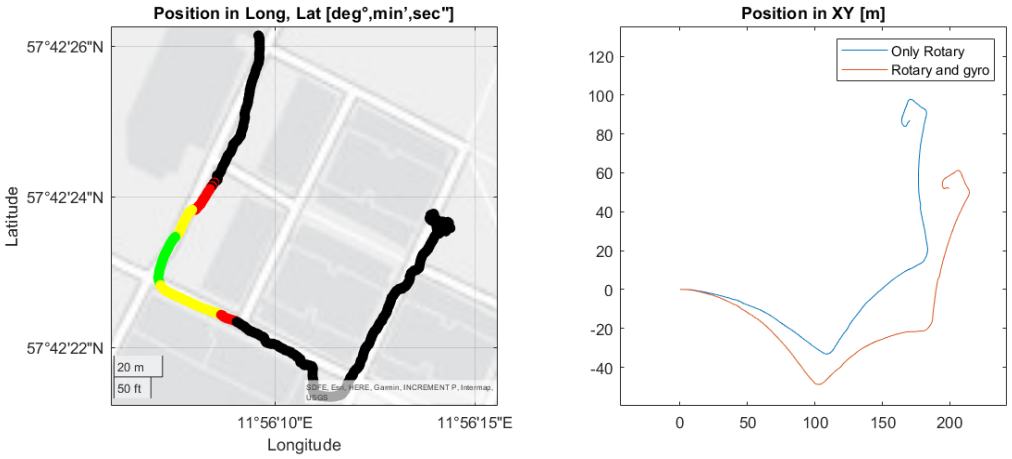
Figur 20 - Extra öglekörning i högervarv, körning 2



Figur 21 - Extra öglekörning med extra mycket folie, vänstervarv, körning 1



Figur 22 - Extra Öglekörning med extra mycket folie, vänstervarv, körning 2



Figur 23 - Extra körning utan förutbestämd rutt, runt kvarteret, över kullersten och ojämn mark. Observera nya skalor på axlarna.

Bilaga 2 - Kod för GADA Slav V6

```
/---------------- Model V6 for the GADA model ----------------------//
//  GPS, Ackermann and Dead Reckoning Assisted Model.
//  Developed by: Marcus Alvefelt, Hampus Strömberg
//  Thesis completed at Diadrom in Gothenburg 2021 and under tutorship from
//  Sakib Sistek at Chalmers University of Technology.
//
// Thank you to Andrew Kramer for use of his example with hardware interrupts.
//-------------------------------------------------------------------------//

//#define SERIAL_ON                         //vUncomment for Debugging with serial monitor

#include <Wire.h>
//*** Rot.giv setup
#define RH_ENCODER_A 3
#define RH_ENCODER_B 5
#define LH_ENCODER_A 2
#define LH_ENCODER_B 4                      // pins for the encoder inputs

// Variables for Rot.giv
long leftCount = 0;
long rightCount = 0;

void setup() {
  Wire.begin(8);                            // Slave joins i2c bus with address #8
  Wire.onRequest(requestEvent);             // Register Requestevent
  #ifdef SERIAL_ON
  Serial.begin(9600);
  #endif

  //---**** Setup Rot.Giv ****---

  pinMode(LH_ENCODER_A, INPUT);
  pinMode(LH_ENCODER_B, INPUT);
  pinMode(RH_ENCODER_A, INPUT);
  pinMode(RH_ENCODER_B, INPUT);             // Pins for Rotary Encoders
  attachInterrupt(0, leftEncoderEvent, CHANGE);     // Initiating hardware interrupts for Rotary Encoders
  attachInterrupt(1, rightEncoderEvent, CHANGE);
}

void loop() {
}

void requestEvent() {
Wire.write(rightCount  & 0x000000FF);
Wire.write((rightCount & 0x0000FF00) >> 8);
Wire.write((rightCount & 0x00FF0000) >> 16);
Wire.write((rightCount & 0xFF000000) >> 24);
Wire.write(leftCount  & 0x000000FF);
Wire.write((leftCount & 0x0000FF00) >> 8);
Wire.write((leftCount & 0x00FF0000) >> 16);         // Same nice union trick as in master.
Wire.write((leftCount & 0xFF000000) >> 24);         // We are saving our sensorvalues as floats and sending
them as bytes.

}
```

Bilaga 3 - Kod för GADA Master V6

```
//---------------- Model V6 for the GADA model ----------------------//
//  GPS, Ackermann and Dead Reckoning Assisted Model.
//  Developed by: Marcus Alvefelt, Hampus Strömberg
//  Thesis completed at Diadrom in Gothenburg 2021 and under tutorship from
//  Sakib Sistek at Chalmers University of Technology.
//
// A Special thank you goes out to the people behind these libraries.
// Thank you for making things open source, easier to use and more accessable.
// These people are:
// Limor Fried/Ladyada At Adafruit, Screen Libraries
// Bryan Siepert at Adafruit, Gyro
// Nicholas Zambetti, Wire Library
// The team at SparkFun, GPS Library
// Tom Igoe, SDcard Library
//-------------------------------------------------------------------//


#include "SparkFun_Ublox_Arduino_Library.h"
#include <Adafruit_SSD1306.h>                   // Library for Display
#include <Adafruit_GFX.h>                       // Graphic library for Display
#include <Wire.h>                               // Library for I2C
#include <SPI.h>                                // Library for alternative communication.
#include <SD.h>                                 // Library for SDcard.
// Note: This Library is outdated and only works with FAT16 och FAT32 type of cards.
// It also has some issues handling files on the SD card. Some work around are present in our code.

//#define SERIAL_ON                             // Toggles serial printing for debug


// *** Gyro
#include <Adafruit_LSM6DSO32.h>
  sensors_event_t accel;
  sensors_event_t gyro;
  sensors_event_t temp;
  Adafruit_LSM6DSO32 dso32;
float zGyro = 0.0;
float xAccel = 0.0;
float yAccel = 0.0;
```

```
//*** Rot.Givare (Rotary Encoder in english)
long rightCount;
long leftCount;

union wheel{
  byte recBuff[4];
  long count;
} right, left;                                  // Nice trick till send parts of our float byte-wise with I2C


//*** SDCard

// For SPI mode,
// Note: arduino MEGA has dedicated MISO, MOSI pin-set.
// We just left these declarations in because they don't interfere with our code
#define LSM_CS 10
#define LSM_SCK 13
#define LSM_MISO 12
#define LSM_MOSI 11
                                                // I2C pins for arduino MEGA 2560: 20(SDA), 21(SCL)
const int chipSelect = 53;                      // Uses MISO MOSI for coms.
File dataFile;
int runnr = 1;                                  // Run Number - This is just a variable that is parsed as a string
                                                // to save a new run on out plattform.
                                                // It's a bit fiddly but has worked decently enough.
                                                // Be careful about holding the increment button down. Will flood
                                                // the SD card.


//*** Screen
#define SCREEN_WIDTH 128                        // OLED display width, in pixels
#define SCREEN_HEIGHT 64                        // OLED display height, in pixels
#define OLED_RESET     -1                       // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C                     //< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
```

```
//*** GPS setup
SFE_UBLOX_GPS myGPS;
    long lastTime = 0;
    int32_t latitude;
    int8_t latitudeHp;
    int32_t longitude;
    int8_t longitudeHp;
   // int32_t ellipsoid;
   // int8_t ellipsoidHp;                  // We aren't using Ellipsoid
    uint32_t accuracy;

                                           // Defines storage for the lat and lon units integer and fractional parts
    int32_t lat_int;                       // Integer part of the latitude in degrees
    int32_t lat_frac;                      // Fractional part of the latitude
    int32_t lon_int;                       // Integer part of the longitude in degrees
    int32_t lon_frac;                      // Fractional part of the longitude
    String longitudeStr;
    String latitudeStr;
                                            // Now define float storage for the heights and accuracy
    //float f_ellipsoid;
    float f_accuracy;


long mill = 0;                             // Just a holder for millis passed since arduino start.
                                           // Gives us an index when sensor values were fresh. Should be
                                           //accurate for a couple of days.

void setup() {
    pinMode(47, INPUT_PULLUP);                  // Button for fetching GPS pos to screen and creating new
                                                // runfile
  Wire.begin();                                 // Joins Master to i2c bus (address optional)


#ifdef SERIAL_ON
  Serial.begin(9600);                             // start serial for output
  Serial.println("Serial on.");
#endif
```

```
// *** Screen Setup ***

  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    drawString("Screen Init. Failed. SSD1306 allocation failed");
    #ifdef SERIAL_ON
    Serial.println(F("SSD1306 allocation failed"));
    #endif
    for(;;);                                          // If failed, loop forever
  }
    display.clearDisplay();
    drawString("Setting up. Hold on cutie.");


  if (myGPS.begin(Wire) == false)                     //Connect to the Ublox module using Wire port
  {
    drawString("GPS not detected at correct adress.");
    Serial.println(F("Ublox GPS not detected at default I2C address. Please check wiring. Freezing."));
    for(;;);                                          // If failed, loop forever
  }

  myGPS.setI2COutput(COM_TYPE_UBX);                   //Set the I2C port to output UBX only (turn off NMEA noise)
  myGPS.setNavigationFrequency(20);                   //Set output to 20 times a second, thats 50 ms between
                                                      //each output. Which is about our outputrate for all sensors.
  //byte rate = myGPS.getNavigationFrequency();       //Get output, unsure what this does. uncommented
                                                      //without any problems. Might be needed for some
                                                      //special applications?
```

```
// *** Gyro Setup ***


  if (!dso32.begin_I2C()) {
    Serial.println("Failed to find LSM6DSO32 chip");
    drawString("Failed to initiate Gyro.");
    while (1) {
      delay(10);
    }
  }
    dso32.setAccelRange(LSM6DSO32_ACCEL_RANGE_8_G);             // -+4, 8, 16, 32 available
    dso32.setGyroRange(LSM6DS_GYRO_RANGE_250_DPS );             // 125, 250, 500, 1000, 2000, 4000
    dso32.setAccelDataRate(LSM6DS_RATE_12_5_HZ);                // 12.5, 26, 52, 104, 208, 416, 833,
                                                                // 1_66K, 3_33K, 6_66K
    dso32.setGyroDataRate(LSM6DS_RATE_12_5_HZ);                 // 12.5, 26, 52, 104, 208, 416, 833,
                                                                //1_66K, 3_33K, 6_66K
                                                                // If these setcommandos are
                                            //commented out the gyro will use the lower values (I think).



// **** SDcard Setup ****
Serial.print("Initializing SD card...");

                                            // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {

    Serial.println("Card failed, or not present");
    drawString("Card Failed. Check if card inserted. ;)");
    while (1);                                  // If failed, loop forever
  }
  Serial.println("card initialized.");

  dataFile = SD.open(String(String(runnr) + ".txt"), FILE_WRITE);
  dataFile.close();

  drawString("Setup complete." "Drive safe.");

}
```

```
void loop() {
  updateGPS();
  dso32.getEvent(&accel, &gyro, &temp);
  zGyro = gyro.gyro.z;                                  // Places sensor_event type var in handy float.
  //xAccel = accel.acceleration.x;
  //yAccel = accel.acceleration.y;                      // These sensors arent used right now in our
                                                        //simulations, but could be implemented in a more well
                                                        //filtered system.

mill = millis();
if ((mill-200) > 0){
    fetchRot();                                         // Fetches rotational values from Slave device.
                                                        // After this point all sensorvalues are fetched. We now
                                                        //determine at what time these values were fresh
                                                        //(freshish. might be some millis off).
}
writeToCard();
  if(digitalRead(47) == LOW){                           // If the run increment button has been pressed, step
                                                        //up filename by one. You also need to reset sensors
                                                        //on Slave device. Just reset Arduino and go again.
  drawString(String(latitudeStr), String(longitudeStr));
  runnr += runnr;
  }
}


void fetchRot(){
  Wire.requestFrom(8, 8);

for(int i = 0; i<4; i++){
  right.recBuff[i] = Wire.read();
}
for(int i = 0; i<4; i++){
  left.recBuff[i] = Wire.read();
}
#ifdef SERIAL_ON
    Serial.print("Right: ");
    Serial.println(right.count);
    Serial.print("Left: ");
    Serial.println(left.count);
#endif

}


void writeToCard(){

    dataFile = SD.open(String(String(runnr) + ".txt"), FILE_WRITE);

  if (dataFile) { //If datafile could be opened. Print stuff.
    //dataFile.println(String(mill) + ":" + String(right.count) + ":" + String(left.count) + ":" + String(zGyro) + ":" +
String(xAccel) + ":" + String(yAccel) + ":" + String(latitudeStr) + ":" + String(longitudeStr) + ":" +
String(f_accuracy));
    dataFile.println(String(mill) + ":" + String(right.count) + ":" + String(left.count) + ":" + String(zGyro) + ":" +
String(latitudeStr) + ":" + String(longitudeStr) + ":" + String(f_accuracy));

    #ifdef SERIAL_ON
```

```
    //Serial.println(String(mill) + ":" + String(right.count) + ":" + String(left.count) + ":" + String(zGyro) + ":" +
String(xAccel) + ":" + String(yAccel) + ":" + String(latitudeStr) + ":" + String(longitudeStr) + ":" +
String(f_accuracy));
    Serial.println(String(mill) + ":" + String(right.count) + ":" + String(left.count) + ":" + String(zGyro) + ":" +
String(latitudeStr) + ":" + String(longitudeStr) + ":" + String(f_accuracy));

    #endif
    dataFile.close();
  }
                                                        // if the file isn't open, pop up an error:
  else {
    Serial.println("error opening dataFile.txt");
  }
}


// ** Screen Drawing functions.
void drawString(String x) {
  display.clearDisplay();

  display.setTextSize(1);                               // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);                  // Draw white text
  display.setCursor(0, 0);                              // Start at top-left corner
  display.cp437(true);                                  // Use full 256 char 'Code Page 437' font

  for(int i = 0; i < x.length(); i++){
  display.write(x[i]);
  }
  display.display();
}

                                                        // Double string function is intended for printing GPS
coords on screen.
void drawString(String x, String y) {
  display.clearDisplay();

  display.setTextSize(2);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.cp437(true);

  for(int i = 0; i < x.length(); i++){
  display.write(x[i]);
  }
  display.setCursor(0, 30);                             // Start at top-left corner, Second row
  for(int i = 0; i < y.length(); i++){
  display.write(y[i]);
  }
  display.display();
}

void updateGPS(){

#ifdef SERIAL_ON
    Serial.println("Fetching GPS");
#endif

    lastTime = millis();                                //Update the timer
```

```
latitude = myGPS.getHighResLatitude();
latitudeHp = myGPS.getHighResLatitudeHp();
longitude = myGPS.getHighResLongitude();
longitudeHp = myGPS.getHighResLongitudeHp();
// ellipsoid = myGPS.getElipsoid();
// ellipsoidHp = myGPS.getElipsoidHp();
accuracy = myGPS.getHorizontalAccuracy();

                                                    // Calculate the latitude and longitude integer
                                                    // and fractional parts
lat_int = latitude / 10000000;                      // Convert latitude from degrees * 10^-7 to Degrees
lat_frac = latitude - (lat_int * 10000000);         // Calculate the fractional part of the latitude
lat_frac = (lat_frac * 100) + latitudeHp;           // Now add the high resolution component

if (lat_frac < 0)                                   // If the fractional part is negative, remove the minus sign
{
  lat_frac = 0 - lat_frac;
}

lon_int = longitude / 10000000;                     // Convert latitude from degrees * 10^-7 to Degrees
lon_frac = longitude - (lon_int * 10000000);        // Calculate the fractional part of the longitude
lon_frac = (lon_frac * 100) + longitudeHp;          // Now add the high resolution component
```

```
  if (lon_frac < 0)                                   // If the fractional part is negative, remove the minus
sign
  {
    lon_frac = 0 - lon_frac;
  }
#ifdef SERIAL_ON
                                                      // Print the lat and lon
  Serial.print("Lat (deg): ");
  Serial.print(lat_int);                              // Print the integer part of the latitude
  Serial.print(".");
  Serial.print(lat_frac);                             // Print the fractional part of the latitude
  Serial.print(", Lon (deg): ");
  Serial.print(lon_int);                              // Print the integer part of the latitude
  Serial.print(".");
  Serial.println(lon_frac);                           // Print the fractional part of the latitude
#endif

                                                      // Convert the horizontal accuracy (mm * 10^-1) to a float
  f_accuracy = accuracy;
                                                      // Now convert to m
  f_accuracy = f_accuracy / 10000.0; // Convert from mm * 10^-1 to m
/*
  // Calculate the height above ellipsoid in mm * 10^-1
  f_ellipsoid = (ellipsoid * 10) + ellipsoidHp;
  // Now convert to m
  f_ellipsoid = f_ellipsoid / 10000.0; // Convert from mm * 10^-1 to m
*/

#ifdef SERIAL_ON
                                                      // Finally, do the printing
  //Serial.print("Ellipsoid (m): ");
  //Serial.print(f_ellipsoid, 4);                     // Print the ellipsoid with 4 decimal places

  Serial.print(", Accuracy (m): ");
  Serial.println(f_accuracy, 4);                      // Print the accuracy with 4 decimal places
#endif
  longitudeStr = String(String(lon_int) + "." + String(lon_frac));
  latitudeStr = String(String(lat_int) + "." + String(lat_frac));
}
```

Bilaga 4 - Kod för GADA MATLAB V7

```
close
clear
clc

data = readtable(***YOUR DIRECTORY HERE***'); %Full Directory. Can also be relative path. (Aka if file is in same map as program. just type filename.

thetaoffset =  0. * pi;

totSamples = max(size(data));
millis = data{:,1};
Rrot = data{:,2};
Lrot = data{:,3};                                   %Raw data from Right and Left Rotary Encoders [pulses/rev]
rads = data{:,4};                                   %Raw data from the gyro in Z. [Rad/s]
lat = data{:,5};
long = data{:,6};                                   %Latitude and longitude.
acc = data{:,7};                                    %The accuracy of this GPS position. In meters.

      %---- Platform specific Parameters ----%
Wheel_Base = 0.57;      % 0.57                      %Measuments given in meters
Wheel_Width = 0.438;    %0.438
Wheel_Cir = 0.4764;     %0.4764                   %Measured from 5 rotations.
Wheel_Dia = Wheel_Cir / pi; %0.1516 with cir=0.4764
Pulses_Per_Rev = 400;                               %Resolution for rotational sensors
```

```
%% Only Rotary Encoders

      %---- Preallocations ----%
x_distance = zeros(totSamples,1);
y_distance = zeros(totSamples,1);
x_position = zeros(totSamples,1);
y_position = zeros(totSamples,1);                    %For Only Rotary and calculated theta

avg_distance = zeros(totSamples,1);
Distance_L = zeros(totSamples,1);
Distance_R = zeros(totSamples,1);

theta = zeros(totSamples,1);                 %Rotary Heading
theta(1) = thetaoffset;                      %Heading in rads -> = 0. (aligned with X-axis)
```

```
for i = 1 : totSamples-1
   Distance_L(i,1) = (((Lrot(i+1) - Lrot(i)) / Pulses_Per_Rev) * Wheel_Cir);
   Distance_R(i,1) = (((Rrot(i+1) - Rrot(i)) / Pulses_Per_Rev) * Wheel_Cir);
         %Rrot(i+1) - Rrot(i) = pulses traveled for right wheel until next sample. /
         %resolution = how many rotations * circumfrance = distance in m
   avg_distance(i) = (Distance_R(i) + Distance_L(i)) / 2;


   dist_diff(i) = abs((Distance_R(i)-Distance_L(i)));          %If sensors only diff 1
                                                               %pulse -> call it going
                                                               %straight.
   if dist_diff(i) < 0.0013
      theta(i+1) = theta(i);
   else
         theta(i+1) = ((Distance_R(i) - Distance_L(i)) / (Wheel_Width)) + theta(i);
% Estimation as used in rapport [25]
   end

   % --- Calculated theta with only Rotary --- %
   x_distance(i) = avg_distance(i) * cos(theta(i));
   y_distance(i) = avg_distance(i) * sin(theta(i));

   x_position(i+1) = x_position(i) + x_distance(i);
   y_position(i+1) = y_position(i) + y_distance(i);
end
```

```
%% Rotary Encoders and Gyro
%Note: Our expectation were that this setup would be more accurate because it
%doesn't use any estimations. Estimations comes with assumptions, which is
%a recipe for a bad time in the real world. However we found these
%sensorvalues to be quite noisy. With filtering these values might surpass
%the estimated position, but for now it aint cutting it.

gyro_x_distance = zeros(totSamples,1);
gyro_y_distance = zeros(totSamples,1);
gyro_x_position = zeros(totSamples,1);
gyro_y_position = zeros(totSamples,1);

theta2 = zeros(totSamples,1);
theta2(1) = thetaoffset;

for i = 1 : totSamples
   if abs(rads(i)) < 0.011
      theta2(i+1) = theta2(i);                          % Measured value, not estimated
   else
       theta2(i+1) = rads(i)*((millis(i+1)-millis(i))*10^-3) + theta2(i);
%Seems like the sensor likes go give +- 1 rad/s so we filter the smallest changes out to
%smooth our data a bit.
   end
%IMPORTANT NOTE: The gyro gives units by rad/s but our calculations for time are
%based on a register which is not counting in interrups. These values are therefore not
%correct.


   gyro_x_distance(i) = avg_distance(i) * cos(theta2(i));
   gyro_y_distance(i) = avg_distance(i) * sin(theta2(i));
   gyro_x_position(i+1) = gyro_x_position(i) + gyro_x_distance(i);
   gyro_y_position(i+1) = gyro_y_position(i) + gyro_y_distance(i);
end
```

```
%% GPS

%- Just a bit of ugly code. Please dont look to closely at this. :)
% Its a workaround for the Geoscatter function. To paint the gps dots
% according to their accuracy. With worst -> Black, red, yellow, green, blue, cyan -> to
best.

acc_color = zeros(totSamples, 3);
for i = 1 : totSamples-1
   if acc(i) < 0.50
      acc_color(i,1) = 1;
      acc_color(i,2) = 0;
      acc_color(i,3) = 0; % [1, 0, 0] which is red. (Accuracy under 0.5 m)
   end
   if acc(i) < 0.40
      acc_color(i,1) = 1;
      acc_color(i,2) = 1; % [1, 1, 0] which is yellow (Accuracy under 0.4 m)
      acc_color(i,3) = 0;
   end
   if acc(i) < 0.30
      acc_color(i,1) = 0;
      acc_color(i,2) = 1; % [0, 1, 0] which is green (Accuracy under 0.3 m)
      acc_color(i,3) = 0;
   end
   if acc(i) < 0.20
      acc_color(i,1) = 0;
      acc_color(i,2) = 0; % [0, 0, 1] which is blue (Accuracy under 0.2 m)
      acc_color(i,3) = 1;
   end
   if acc(i) < 0.10
      acc_color(i,1) = 0;
      acc_color(i,2) = 1; % [0, 1, 1] which is cyan (Accuracy under 0.1 m)
      acc_color(i,3) = 1;
   end

end
```

```
%% Plotting
figure(1)
subplot(1,2,2);
set(gcf, 'Position',  [100, 100, 1000, 400])
plot(x_position, y_position);
title('Position in XY [m]')

xlim([-1 11]);
ylim([-5 5]);
hold on
plot(gyro_x_position, gyro_y_position);
legend('Only Rotary', 'Rotary and gyro');

subplot(1,2,1);
geoscatter(lat,long, [], acc_color);
title('Position in Long, Lat [deg°,min',sec"]')

%% GPS distance
%Extra bit of code to find the distance between start and stop position in
%GPS coordinates (straightline distance). Uses lldistkm function, thank you
%to M Sohrabinia for putting this together. Viva la open source!

latlon1 = [lat(1,1),long(1,1)];
latlon2 =  [lat(end,1),long(end,1)];

[Distance1, Distance2]=lldistkm(latlon1,latlon2);
```

Bilaga 5 - Marknadsundersökning

Nedan följer inklippt den marknadsundersökning som skickades till företag. Denna undersökning var främst ej menad att hjälpa oss kartlägga ett problem utan användes som ett medel för att nå ut till företag som arbetade med positionering med hjälp av GPS. Som rapporten tar upp var denna enkät ett misslyckat försök till företagskontakt.

1 Undersökning Examensarbete VT21

Undersökningens avsikt är att få en första förståelse för de problem som marknaden idag kämpar med. Dessutom för att utforska potentialen för GPS inom autonomitet.

1. Vilket företag representerar du? (Lämna blank om ni vill vara anonyma)
2. Använder era produkter GPS för positionsbestämning?
3. Beskriv kort hur GPS används i dessa fall.
4. Använder ni några andra sensorer för att förbättra positionsbilder - i så fall vilka?
5. Om svaret var ja på föregående fråga, hur används dessa?
6. Vilka problem kämpar ni med idag angående GPS?
7. Hur anser ni att en “exakt” position skulle kunna samverka för att få en bättre autonomitet?
8. Finns det några andra punkter som vore intressanta för oss att utforska som vi inte tagit upp i denna undersökning? I så fall vilka då?