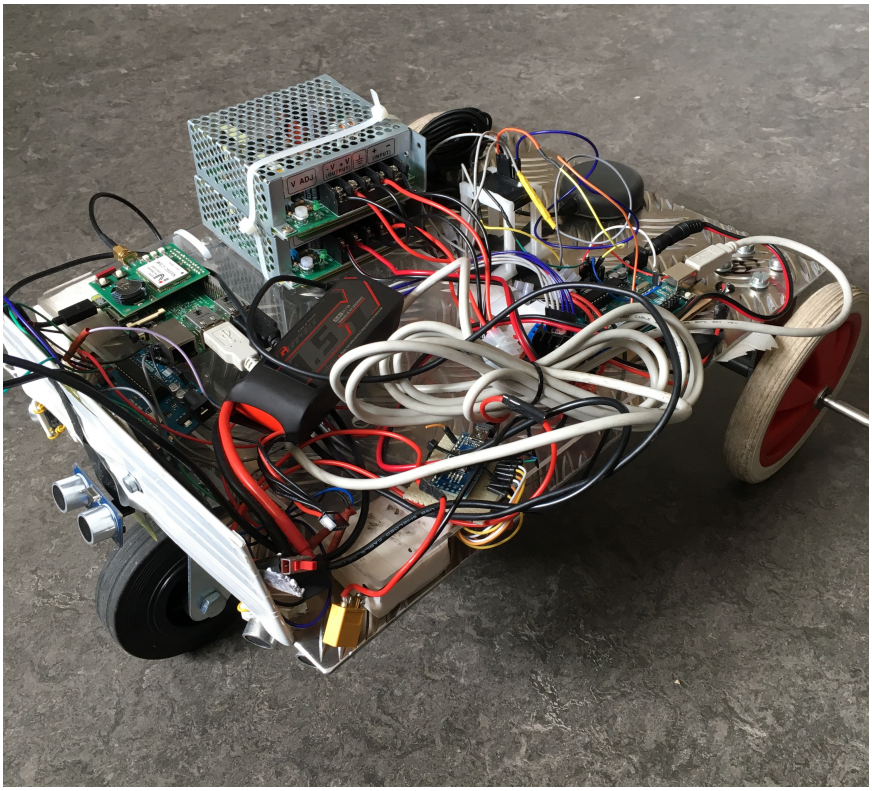


Navigationssystem för robotgräsklippare med INS och RTK

Ett kandidatarbete vid Institutionen för Data- och informationsteknik
DATX02-19-05

Louise Andersson
Hampus Carlsson
Hugo Ekelund
Madeleine Müller
Daniel Sonnert
Niclas Theodorsson

Institutionen av Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2019

Kandidatarbete 2019: DATX02-19-05

Navigationssystem för robotgräsklippare med INS och RTK

LOUISE ANDERSSON
HAMPUS CARLSSON
HUGO EKELUND
MADELEINE MÜLLER
DANIEL SONNERT
NICLAS THEODORSSON

CHALMERS

Instutitionen av Data- och informationsteknik
Chalmers tekniska högskola
Göteborg, Sverige 2019

Navigationssystem för robotgräsklippare med INS och RTK
LOUISE ANDERSSON
HAMPUS CARLSSON
HUGO EKELUND
MADELEINE MÜLLER
DANIEL SONNERT
NICLAS THEODORSSON

Handledare: Alexandra Angerd och Sven Knutsson, Data- och informationsteknik
Examinator: Lena Petersson, Data- och informationsteknik

Kandidatarbete 2019: DATX02-19-05
Institutionen för Data- och informationsteknik
Chalmers Tekniska Högskola
412 96 Göteborg
Sverige Telefon: +46 31 772 1000

Omslag: Foto på den slutgiltiga hårdvaran.

Skriven i LaTeX
Göteborg, Sverige 2019

Sammanfattning

Målet med det här projektet var att skapa ett robust navigationssystem för en robotgräsklippare som använder sig av både GPS- och INS-system för att navigera. Ett problem med dagens robotgräsklippare är att det behövs en avgränsningskabel för att roboten ska kunna veta var den ska klippa. Kabeln medför en installation som kan vara både tidskrävande och svårt då terräng i trädgården inte alltid är lätt att gräva i. Det händer även att dessa avgränsningskablar kan störa ut varandra ifall de är placerade tillräckligt nära varandra. Uppgiften i projekt blev därmed att skapa ett navigationssystem som skulle innebära att en kabel inte skulle behövas. Hårdvaran som används i projektet är en robotgräsklippare utan klippfunktion som är skapad i ett kandidatarbete från föregående år[1]. För att nå målet delades arbetet upp i tre olika iterationer där varje iteration hade specifika mål som skulle nås. Efter varje iteration utvärderades vad som hade gått som planerat och vad som behövde överlappa till nästa iteration.

Navigationssystemet som utvecklades testades i slutet av projektet och det fastställdes att det inte nådde de mål som satts upp, då systemet inte lyckades hålla robotgräsklipparen inom ett bestämt område, mestadels beroende på att den positionsdata som togs emot inte hade hög precision nog för att navigationssystemet skulle fungera som förväntat. Mycket pekade på att om GPS-positioneringen fungerat bättre hade navigationssystemet fungerat, men det kunde inte bekräftas.

Nyckelord: *GPS*, *INS*, *navigationssystem*, *robotgräsklippare*

Abstract

The goal of this project was to develop a robust navigation system for a robotic lawn mower. Today's robotic lawn mowers typically demands a boundary wire in order to stay within its working area. This means that an installation process of the boundary wire is necessary. This can be time consuming, costly, as well as inconvenient for the user. Another problem with the boundary wires is that adjoining wires can interfere with each other. With this in mind, the task for this project was to create a solution where a boundary wire will not be needed. In order to solve this a navigation system which used both GPS- and INS-technology was developed. The project used a robotic lawn mower constructed in a previous bachelor thesis[1]. In order to attain the goal of the project, the tasks were split into three iterations, where each iteration had specific goals which was to be reached. After each iteration was done, a reflection ensued.

The navigation system which was developed during the project was tested and it was concluded that it did not reached the goals which were set for the project, as it failed to stay within a defined area. This was foremost due to a lack of precision in the positional data which was attained by the GPS-system which was used. It is reasonable to assume that if this positional data was more precise, the navigational system would work well, as the rest of the system worked well, however, this could not be confirmed.

Keywords: *GPS*, *INS*, *navigation system*, *robotic lawn mower*

Tillkännagivanden

Vi vill tacka de här personerna som har hjälpt oss under projektets gång:

- Alexandra Angerd och Sven Knutsson - våra handledare som hjälpt oss under hela projektet.
- Lars Norén - för stor hjälp och rådgivning vid hårdvarufrågor.
- Marcus Nordin - för hjälp att montera basstationen på taket.
- Ingemar Johansson och Ulf Svensson - för hjälp att få tillstånd att komma upp på taket för montering av basstation.
- Göran Stigler - för att vi fick låna GPS:er.

Förkortningar och Koncept

Följande förkortningarna markeras med understrykning när de förekommer i rapporten.

- EKF - Utökat Kalman filter, ett filter utbyggt utifrån Kalmanfilter som även linjärt approximerar funktionen.
- Filter - Ett system som har till uppgift att minska brus i signalbehandlingen.
- GNSS - Global Navigation Satellite System, ett samlingsnamn för de olika satellitsystemen som används idag så som GPS, GLONASS och Galileo.
- GPS - Global Positioning System, ett sätt att mäta sin position med hjälp av satelliter runt jordens omloppsbana.
- IMU - Inertial Measurement Unit, en elektronisk enhet som mäter acceleration och vinkelhastighet lokalt med hjälp av accelerometer och gyroskop.
- INS - Inertial Navigation System, ett tröghetsnavigeringssystem som kontinuerligt mäter dess position med hjälp av en IMU och beräkningar utifrån givna värden från IMU-modulen.
- RPI3 - Raspberry Pi 3, en enkortsdator som primärt används i detta projekt för att göra navigationsberäkningar.
- RTK - Real-Time Kinematic är en form av positionsmätning som använder sig av två mottagare som samarbetar för att uppnå en högre precision.
- RTKLIB - Ett bibliotek som gör GPS-beräkningar för RTK.
- UKF - Sigmapunktsfilter, ett filter utbyggt utifrån Kalmanfilter som även gör en statistisk linjärisering.
- H-brygga - En krets som byter polariteten på spänningen, i vårt fall används H-bryggan för att byta riktning på motorerna.

Innehåll

1 Inledning

Den första robotgräsklipparen introducerades 1969 [2]. Den navigerade godtyckligt inom ett område definierat av en avgränsningskabel, vilket väsentligen fortfarande är hur dagens robotgräsklippare navigerar [3].

Installation och underhåll av en avgränsningskabel kan vara både tids- och resurskrävande. För att kabeln ska kunna avgränsa ett område behövs ett flertal parametrar tas i beaktande: förutom att kabeln måste vara intakt finns andra påverkande faktorer såsom signaler från utomstående källor vilka kan komma att konkurrera ut kabelns egna signaler [3]. Avgränsningskabeln sätter därför även begränsningarna på robotgräsklipparens användarvänlighet likväl som på dess tillämpningsområden. Användning av kabeln kan fungera väl på mindre områden men den blir till en begränsning vid användning på större gräsmattor, så som fotbollsplaner och parker. Då krävs stora mängder avgränsningskabel vilket gör att installation och underhåll blir ännu mer omständligt.

Ett annat navigationssystem utan avgränsningskabel skulle kunna lösa de problemen, samt attrahera en ny och större marknad. Många av senare generationers robotgräsklippare är utrustade med Global Positioning System (Navigationssystem, GPS) vilken i dagsläget endast används för positionering för till exempel stöldskydd. Ett möjligt alternativ för att eliminera avgränsningskabeln är att istället använda GPS:en för att navigera. Området skulle därmed kunna begränsas utifrån koordinater, vilket skulle förenkla installationsprocessen och dessutom inte heller låsa gräsklipparen till ett specifikt område som kabeln gör. Det skulle även göra robotgräsklipparen mer användarvänlig och dessutom möjliggöra effektivisering i form av till exempel ruttplanering. Det skulle även kunna tänkas tillämpas på förslagvis drönare och andra robotar inom till exempel ekologiskt jordbruk [4].

Det har tidigare gjorts kandidatarbeten om GPS-navigerade robotgräsklippare på Chalmers. Ett föregående projekt, *Satellitnavigerad robotgräsklippare med RTK* [5] från 2016, har lyckats uppnå en positionering med god precision men dock endast under förutsättningen att GPS-antennen hade fri sikt mot himlen. Det blir därför utgångspunkten för detta arbete. Dessutom finns hårdvara från tidigare projekt som kan återanvändas till detta kandidatabete [1], [5], vilken beskrivs närmare i kapitel 2.2.

1.1 Syfte

Syftet med projektet är att utveckla ett robust navigationssystem för robotgräsklippare som utnyttjar två GPS-moduler och en Inertial Measurement Unit-modul (tröghets-modul, IMU-modul) för att eliminera behovet av avgränsningskabeln. Begreppet robusthet i sammanhanget innebär att robotgräsklipparen inte åker utanför sitt givna område. Eftersom det finns risk för att GPS-systemet blir temporärt otillgängligt ska navigationssystemet även kunna navigera utan GPS-systemet med hjälp av INS-systemet. Navigationssystemet testas med en redan existerande robotprototyp från ett tidigare kandidtarbete [5].

1.2 Problemformulering

Den huvudsakliga uppgiften är att utveckla ett robust navigationssystem för en robotgräsklippare, som kan fungera utan en avgränsningskabel. Systemet kommer att köras på en redan existerande hårdvara som därefter konstruerats i samband med tidigare kandidatarbete [1], [5].

De problem som måste lösas för att huvuduppgiften ska lösas är de följande:

- Hur ska GPS-navigeringen fungera? Hur ska navigeringen robust nog för att kunna fungera även utan satellitmottagning?
- Hur ska hårdvaran kunna styras och ändras från tidigare arbeten [1], [5]?

- Hur ska klipparens position användas för att styra dess rörelse?
- Hur ska gräsklipparen veta vilket område den ska röra sig inom?

För att få en bättre översikt över vad som behöver göras delades huvuduppgiften upp i följande fyra deluppgifter:

- Den första deluppgiften är att navigationssystemet ska veta vart klipparen är med en tillräckligt tillförlitlig noggrannhet för användningen, även i lägen där satellitnavigeringssystemen inte är tillgänglig. Det betyder att det i den här deluppgiften ingår att utveckla ett lokalt navigationssystem, ett INS-system, till robotgräsklipparen som kan användas i de här situationerna. I uppgiften ingår även att ta reda på hur pålitligheten kan ökas, vilket bland annat kommer uppnås genom att minska bruset i positionsdatan.
- Den andra deluppgiften är utveckling av maskinnära mjukvara. Det innebär utveckling av program för motorkontroll, styrning samt förmågan att undvika kollision med objekt.
- Den tredje deluppgiften är att gräsklipparens position ska användas för att styra dess rörelse. Uppgiften är att ta reda på vilka algoritmer och metoder som ska användas för att ta de här besluten.
- Den sista deluppgiften är att utveckla en enkel webbapplikation som ska kunna användas för att definiera ett avgränsningsområde som robotgräsklipparen ska hålla sig inom.

1.3 Avgränsningar

Utgångsläget för det här projektet är att det redan existerar en fungerande gräsklippare som kan omarbetas. Hårdvarubasen som ska användas är skapad under ett kandidatarbete från 2018 vid namn *Smart Robot Lawn Mower* [1]. Den här robotgräsklipparen är autonom men kan inte klippa gräs. Området som roboten ska navigera inom ska vara en plan yta, vilket är på grund av att prestandan hos robotgräsklipparen som kommer användas i projektet inte tillåter att den kör i kuperad terräng. Navigationssystemet varierar dock inte nämnvärt vid olika lutningar [6], vilket innebär att beslutet inte kommer påverka resultatet. Projektgruppen har därmed beslutat att ej fokusera på att utveckla hårdvaran för att kunna hantera kuperad terräng, då projektets fokus är mjukvaran.

1.4 Mål

Målet med projektet är att skapa en robotgräsklippare som klarar av att befinna sig inom ett markerat område med hjälp av en GPS/INS-integration i 15 minuter. Eftersom GPS-signalen kan vara tillfälligt svag eller bortkopplad ska robotgräsklipparen även testas under sådana omständigheter, det vill säga roboten ska kunna hålla sig inom ett markerat område även med naturligt förekommande brus i GPS-signalen och även om GPS-signalen blir bortkopplad i upp till 10 sekunder.

En del av detta mål är att skapa ett användarvänligt gränssnitt i form av en webbapplikation. De funktioner som ska implementeras i gränssnittet är följande:

- Visa robotgräsklipparens position på en karta.
- Markera ett område på kartan som representerar det område som ska klippas.
- Stoppa robotgräsklipparen.
- Markera ut flera hinder.

1.5 Metod

Metoden som användes under projektet var att dela upp arbetet i olika iterationer. Iterationerna definierades av delmål, vilka beskrivs i kapitel 3. Vid slutet av varje iteration reflekterades det över huruvida målen nåtts eller inte, samt hur arbetet har gått.

Det huvudsakliga arbetet under projektet har varit att implementera mjukvara och hårdvara, vilket beskrivs i kapitel 4. När mjukvaran och hårdvaran var färdiga utfördes tester för att testa huruvida navigationssystemet uppnådde de mål som satts med målet, vilket beskrivs i kapitel 5.

2 Teori

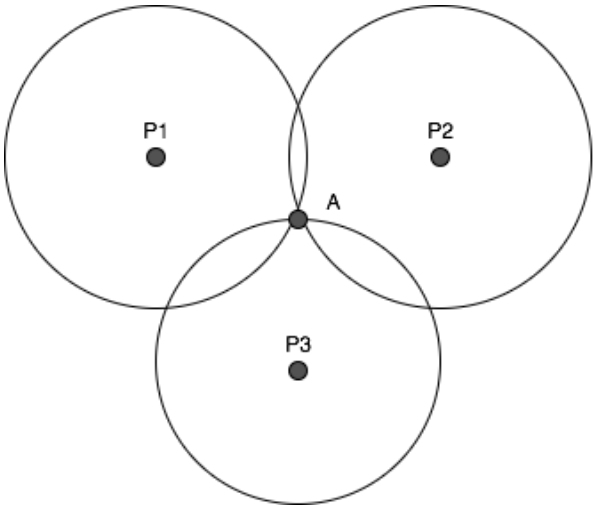
Ett GPS-system och ett INS-system är två fristående positioneringssystem vilka kan sammanfogas med ett Kalmanfilter till ett gemensamt navigationssytem. För att skapa ett sådant navigationsystem behövs olika komponenter så som enhetskort, GPS-moduler, och en IMU-modul. Robotgräsklippare är även utrustas med ultraljudssensorer för att kunna undvika kollision med närliggande objekt. I följande delavsnitt kommer dessa olika system och komponenter beskrivas.

2.1 Navigationssystem

Satellitbaserade navigationssystem, som till exempel GPS, kan användas för positionsbestämning samt för fastställande av latitud och longitud. Ett GPS-system är ett pålitligt positioneringssystem under villkoret att fri sikt mot himlen erhålls, däremot krävs dyrare utrustning för att uppnå bästa möjliga precision. Ett tröghetbaserat navigationsystem, INS-system är däremot inte beroende av externa källor som satelliter utan utnyttjar istället mekanikens lagar för iterativa positionsberäkningar. Ett INS-system utför postionsberäkningar genom att integrera accelerations- och rotationsdata från en IMU-modul över tiden. Nackdelen blir dock att felen kommer divergera med tiden. Ett GPS-system och ett INS-system komplimenterar varandra därför väldigt bra och sammanfogas till ett gemensamt navigationssystem med hjälp av ett Kalmanfilter.

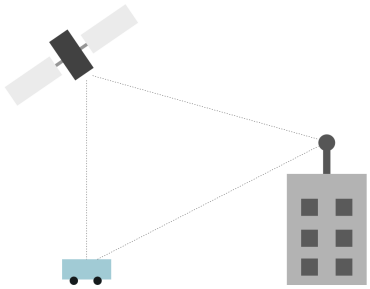
2.1.1 GPS och RTK

GPS är ett av flera satellitsystem tillhörande Global Navigation Satellite System (GNSS) [7]. Tekniken skapades till en början enbart för den amerikanska militären men har sedan 1993 kunnat användas av allmänheten. GPS-systemet består av 24 satelliter som cirkulerar i omloppsbana runt jorden [7]. Satelliterna är utsprida så att en mottagaren har tillgång till fyra satelliter samtidigt, oberoende av mottagarens position. Mottagrens position fastställs genom trilaterering där tiden för den utsända radiovågen att färdas fram till mottagaren mäts noggrant samt genom information av mottagarens nuvarande position [7]. Trilaterering går huvudsakligen ut på att hitta skärningspunkten (objektet) mellan tre sfärer som i det här fallet utgörs av avstånd till tre satelliter [8]. Det här illusteras i figur 1. En nackdel med GPS-positionering är att det ibland kan ske avbrott då signalen från satelliterna blockeras av exempelvis träd. Problemet kan dock lösas genom att kombineras GPS:en med ett självständigt navigationssytem som inte påveras av yttre miljöfaktorer.



Figur 1: Illustration av trilateration där A utgör skärningspunkten(objektet), P1, P2 och P3 är satelliternas positioner.

Som ett komplement till enbart en GPS-mottagare kan Real-Time Kinematic (bärvågsbaserad satellitnavigering, RTK) tillämpas för att öka precisionen. RTK kan åstadkommas genom att använda två GPS-mottagare, varav en agerar basstation placerad på en fast och känd position, medan den andra agerar rörlig mottagare vilket i det här fallet kommer vara robotgräsklipparen, uppsättningen illustreras i figur 2 [9]. För att sedan kunna få en mer noggrann positionsbestämning kommer den rörliga mottagaren att använda basstationen som stöd. Det här sker genom att de två mottagarna kommunicerar med varandra. För att denna kommunikation ska kunna upprätthållas behövs en datalänk, vilket skapas med hjälp av en enkortsdator i form av t.ex. en Rasberry Pi. Avståndet mellan mottagarna är en avgörande faktor i hur noggrant positionen kommer att kunna bestämmas, då ökat avstånd medför ökad mätosäkerhet [9].

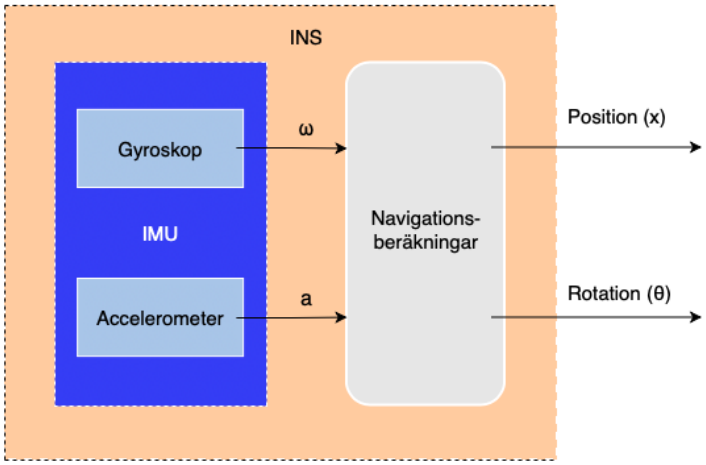


Figur 2: Illustration över GPS-system med RTK. Figuren innehåller en satellit, basstation och robotgräsklippare (rörlig mottagare).

2.1.2 Tröghetsnavigeringssystem

Ett tröghetsnavigeringssystem (INS) beräknar iterativt den nuvarande positionen utifrån föregående position genom att integrera accelerationsdata uppmätt av en Inertial Measurement Unit (Tröghetsmätare, IMU). En IMU-modul innehåller vanligtvis en accelerometer och ett gyroskop. Accelerometrern består av ett fjäder-mass-system vilket kan användas för att bestämma accelerationen enligt Hookes lag och gyroskopen utnyttjar Newtons tredje lag för att kunna bestämma vinkelhastigheten. Det här gör INS:en

till ett självständigt navigationsystem. En mer överskådlig bild av ett INS-system presenteras i figur 3. INS:en har en hög precision under ett kort tidsintervall men precisionen avtar med tiden.



Figur 3: Blockschema för ett INS-system bestående av en IMU som mäter accelerationen och vinkelhastigheten vilket används för att beräkna hastighet och position. Från [10]. CC-BY-SA. Modifierad genom översättning och simplifiering.

2.2 Komponenter

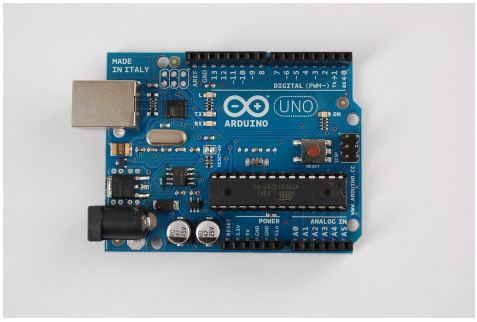
Robotgräsklipparen består av ett flertal komponenter vars funktioner är av relevans för att förstå roboten och navigationssystemets uppbyggnad samt begränsningar. Komponenterna av betydelse kommer därför förklaras i det här avsnittet.

Den ursprungliga robotgräsklipparen bestod av två DC/DC-omvandlare, två motorer, en H-brygga, tre ultraljudsensorer, en GPS-modul och mikrokontrollerkort (Arduino), allt monterat på en platta på hjul. Det fanns ursprungligen två Arduino-kort på roboten; en för styrningen av motorerna genom H-bryggan och en för kommunikation med ultraljusensorerna, samt en GPS-modul [11]. Till den ursprungliga robotgräsklipparen har en IMU-modul och ytterligare en Arduino, som läser av acceleration och vinkelacceleration lokalt från IMU-modulen lagts till. En RPI3 med en bättre GPS-modul fastmonterad lades även till.

2.2.1 Arduino

Arduino är en samling av olika mikrokontrollerkort. En bild på en Arduino Uno finns i figur 4. Arduinokorten programmeras med ett eget C/C++-baserat programspråk[12]. Arduinokorten har, beroende på det specifika kortet, varierande antal digitala I/O-pinnar och analoga pulsmodulerande pinnar. Vissa av de digitala I/O-pinnarna kan dessutom ha stöd för avbrottsanrop (beroende på kortmodell). Pinnarna är tänkta att användas för logik, vilket innebär att vid styrning av tyngre komponenter bör ett drivsteg användas, exempelvis ett relä eller H-brygga [12].

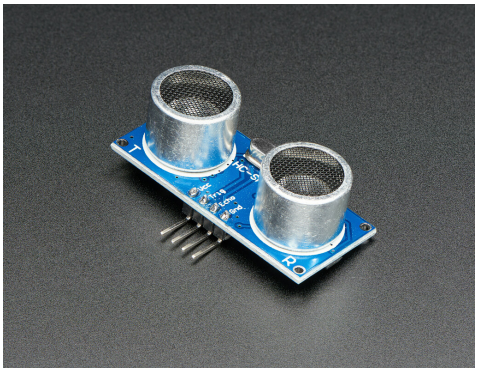
Det finns en rad olika Arduinomodeller, varav två är Arduino Nano samt Arduino Uno. Den största skillnaden mellan de här två varianterna är antalet analoga I/O-pinnar, spänningsmatning samt typen av USB-port. Båda kan drivas via strömtillförsel på pinnar eller via en USB-port, där även överföring av programkod sker. USB-porten kan även användas för seriekomunikation [12].



Figur 4: Arduino Uno. Från [13], CC BY-NC-SA 2.0.

2.2.2 Ultrasonic HC-SR04

HC-SR04 är en sensor som kan bestämma avstånd med hjälp av ultraljud. Modulen kan mäta avstånd från 2 cm upp till 400 cm med en noggrannhet upp till 0,3 cm. Sensorn består av en sändare (märkt T för ”transmitter”) och en mottagare (märkt R för ”receiver”, vilket illustreras i figur 5, samt en styrkrets som fungerar som ett arbetsminne. Sändaren sänder ut en ultraljud-signal med frekvensen 40 kHz [14]. När den utsända signalen träffar ett objekt reflekteras den och den reflekterade signalen registreras av mottagaren. Då ljudhastigheten är känd kan tidsdifferensen, tiden från att signalen sänds ut tills reflektionen mottas, beräknas och används för att beräkna avståndet till objektet. För att kunna beräkna avståndet mellan objekt och sensorn behöver modulen kopplas till ett mikrokontrollerkort, förslagsvis en Arduino.



Figur 5: Ultraljud avståndsmätare av modell HC-SR04. Från [15], CC BY-NC-SA 2.0.

Beräkningen görs genom att mäta den tid t från signalen sänds ut av sändaren tills signalens reflektion detekteras av mottagaren. Avståndet d [m] till objektet bestäms sedan genom att multiplicera tiden t[s] med ljudets hastighet i luft, $v \approx 340$ m/s, och därefter dividera med två. Faktorn en halv uppkommer eftersom signalen reflekteras och därmed färdas dubbla sträckan. Formeln ser ut på följande vis:

$$d = \frac{v \cdot t}{2}. \tag{1}$$

2.2.3 Raspberry Pi 3

Raspberry Pi 3 Model B(RPI3) är en enkortsdator av tredje generationen som introducerades år 2016 av Raspberry Pi Foundation [16]. Användningsområdena för de här enkortsdatorerna är många då de

är billiga, små och går att ansluta till annan elektronik via programmeringsbara pinnar likt en Arduino. Fördelen med en RPI3 jämfört med en Arduino är att den klarar av att härbärgera ett fullständigt operativsystem såsom Linux med tillhörande programbiblotek.

RPI3 kan agera som en brygga mellan hårdvara och användare i form av en webbserver samt utföra diverse beräkningar som krävs vid navigation.

2.2.4 RasPiGNSS

RasPiGNSS är ett expantionskort till Raspberry Pi som monteras på GPIO-pinnarna [11]. Expansionskortet möjliggör en direkt koppling till GNSS-modulen, NV08C-CSM samt anslutningsmöjligheter för GPS-mottagare. RasPiGNSS är byggd på en NV08C-CSM [17] och har support för satellitsystemen GPS, GLONASS, GALILEO och SBAS. Om RasPiGNSS och GPS-processeringsbiblioteket RTKLIB kombineras uppger återförsäljaren för RasPiGNSS att en standardavvikelse på under några decimeter kan uppnås vid ideala väderförhållanden[18].

I projektet ansluts antenner från Tallysman [19], modell TW-2410 till RasPiGNSS-korten. Modellen stödjer GPS, GLONASS och SBAS. Antennen är cirkulär vilket möjliggör ett cirkulärt respons över hela bandbredden.

2.3 Kalmanfilter

Mätdata som positionsdata från en GPS kan vara brusiga på grund av störningar från godtyckliga objekt som träd eller väderförhållanden. Därför kan GPS:ens signaler behövas filtreras för att minska felen. Om dessutom data från en IMU-modul och en GPS-modul (samt en GPS-basstation) ska kombineras för att skapa ett gemensamt navigationsystem för positionbestämning kan ett Kalmanfilter (KF) användas.

Ett Kalmanfilter estimerar tillstånd, till exempel koordinatposition, acceleration, vinkelacceleration, använder ett tillstånd för att förutspå nästa tillstånd, jämför estimeringen med verkligheten och gör en ny estimering baserat på verkligheten och den tidigare estimeringen [20], [21]. Det innebär att filtret arbetar rekursivt och använder endast den tidigare estimeringen, istället för alla estimeringar, vilket är optimalt för externa mikroprocessorer med begränsat minne. Ett Kalmanfilter är linjärt och kräver att indata är normalfördelade. Eftersom GPS- och INS-system är icke-linjära (och mätdatan därmed inte normalfördelade) är inte ett linjärt Kalmanfilter praktiskt användbart. Det finns däremot modifierade Kalmanfilter för icke-linjära system, *Extended Kalman Filter* (Utökat Kalmanfilter, EKF) samt *Unscented Kalman Filter* (sigmapunktsfilter, UKF.)

Ett Kalmanfilter estimerar nästkommande tillstånd x' baserat på tidigare tillstånd x_{n-1} enligt,

$$x' = Fx_{n-1}$$
$$p' = Fp_{n-1}F^T + Q,$$

där p representerar tillståndets kovarians, F är en transformationsmatris som relaterar tillstånden och Q är brus. Tillståndet uppdateras genom att bestämma nästkommande tillstånd x_n enligt,

$$x_n = x' + Ky$$
$$p_n = (I - KH)p',$$

där I motsvarar enhetsmatrisen och H observationsmatrisen som relaterar observationen z_n och estimeringen och y som är differensen,

$$y = z_n - Hx',$$

och K representerar Kalmanmatrisen,

$$K = \frac{p'H^T}{Hp_{n-1}H^T + R},$$

där R motsvarar observationens brus.

2.3.1 Utökat Kalmanfilter

Då linjära funktioner skapar normalfördelningar som är linjära och icke-linjära funktioner skapar normalfördelningar som är icke-linjära är ett sätt att tillämpa Kalmanfilter i praktiken att linjärt approximera funktioner. EKF tillämpar ett linjärt Kalmanfilter på icke-linjära modeller med hjälp av taylorutvecklingar [21]. Olinjära system av hög ordning kan därför resultera i att fel gör så att ett EKF divergerar med tiden [22].

2.3.2 Sigmapunktsfilter

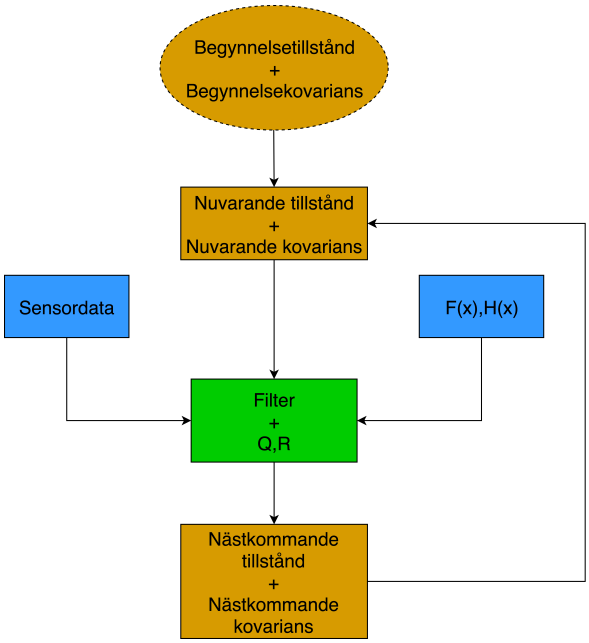
Till skillnad från EKF linjäriseras inte ett UKF funktionen utan väljer istället sigmapunkter kring det tidigare tillståndet vilka sedan viktas. Sigmapunkter är punkter som representerar kovariansen och medelvärdet för insamlad mätdata. Punkterna propageras sedan vidare till nästa sampelinterval genom numerisk integration. Som en följd av det kan då en kovariansmatris skapas som ligger till grund för nästa mängd av sigma-punkter [21]. Det innebär att skillnaden mellan EKF och UKF är att EKF gör en analytisk linjärisering medan UKF gör en statistisk linjärisering [22].

Huruvida EKF eller UKF är mest fördelaktigt i allmänhet är inte givet. Många analyser påpekar att UKF är stabilare än EKF på grund av den högt olinjära naturen av navigationssystem [23], andra analyser påpekar att andra ordningens EKF kan prestera bättre än UKF i vissa fall [24], eller presterar likadant som UKF [25]. En analys visar även empiriskt att UKF kräver mer beräkningstid än EKF trots att de har samma tidskomplexitet [22].

Oavsett så visar en överskådlig bild att även om det finns vissa tvetydigheter angående huruvida UKF är överlägsen EKF eller inte så visar en majoritet av forskningen att UKF är överlägsen vid sensorfusion. Det är troligtvis på grund av att UKF genererar sigmapunkter baserat på känd eller uppskattad statistik angående sensorbrus, medan EKF linjäriserar endast baserat på tillstånd utan hänsyn till icke-linjärt sensorbrus [22].

2.3.3 Pykalman

Pykalman [26] är ett pythonbibliotek för Kalmanfiltrering. Biblioteket stödjer linjär och olinjär, samt realtids och offline Kalmanfiltrering. Dessutom har biblioteket stöd för filtrering vid avsaknad av mätdata. Användning av biblioteket kräver modulerna *NumPy* och *SciPy* och för att därefter skapa ett pykalmanfilter krävs även att ett antal Kalmanmatriser ansatts. För både linjära och ickelinjära filter krävs ett initialtmedelvärde och en initialkovarians. För det linjära filtet krävs dessutom att en transformations- och observationsmatris ansätts för att relatera ett tillstånd med nästkommande tillstånd och för att relatera observationerna med uppskattningarna. Det ickelinjära filtret kräver i stället att transformations- och en observationsfunktion definierats vilka har motsvarande funktionalitet. Detta utgör en väsentlig skillnad då matriserna är statiska och till skillnad från funktionerna vilka kan uppdateras under filtreringens gång. En annan skillnad är att det ickelinjära filtret också kräver att en transformations- och en observationskovariansmatris ansatts vilket det linjära filtet kan göra själv. Transformations- och observationskovariansen påverkar hur filtet viktar observationerna relativt uppskattningarna och har därför stor inverkan på filtreringsresultatet. En överskådlig illustration av filtret finns i figur 6.



Figur 6: Illustration över ett Kalmanfilters inparametrar och utdata. Kalmanfiltet tar in ett tillstånd och dess kovarians samt observationer varutifrån Kalmanfilter estimerar nästkommande tillstånd. F och H är två funktioner beroende av tillståndet, i det ickelinjära fallet, eller två konstanta matriser, i det linjära fallet, vilka relaterar det nästakommande tillståndet med det nuvarande samt relaterar tillstånden med observationerna. Q och R är två konstanta matriser vilka viktar observationerna mot de uppskattade tillstånden.

3 Iterationer

Under projektets gång har många olika delar behandlats, så som att utveckla basfunktionalitet, installera och integrera IMU-och GPS-moduler och utveckla ett fungerande UKF. Därför delades projektet upp i iterationer. Det gjordes för att effektivt kunna dela upp arbetet samt reflektera över hur framtida arbete skulle förhålla sig till projektets tillstånd mellan iterationerna.

3.1 Iteration 1

Under första iterationen låg fokus på att få igång robotens basfunktionalitet vilket summerades i följande fyra punkter:

- Fungerande styrning av motorerna och övrig elektronik.
- Fungerande kommunikation med gräsklipparen genom en enkel webbapplikation där användaren kan styra enheten.
- Ett navigationssystem som kan ta emot instruktioner från användaren och styra robotgräsklipparen i önskad riktning.
- Inläsning av rådata från GPS.

För att kunna testa mjukvaran under utveckling var det viktigt att få igång motorerna samt kommunikation mellan de olika enhetskorten, vilket gjorde att detta prioriterades tidigt i den första iterationen. Det underlättade parallellisering av arbetet genom att fördela arbetsuppgifterna till mindre grupper.

Arbetsuppgifterna var att konfigurera biblioteket för Kalmanfiltret, installation av GPS:en, utveckling av mjukvara som styr motorerna samt utveckling av webbapplikationen.

3.2 Iteration 2

Målet med den andra iterationen var att möjliggöra de grundläggande funktionerna för ett GPS-baserat navigationssystem i en simuleringsmiljö. De krav som skulle uppfyllas var:

- Navigationssystemet ska klara av att hålla robotgräsklipparen inom ett hårdkodat område.
- Gräsklipparen ska vända i en slumpartad riktning när en kant av det hårdkodade området nås.
- INS-systemet ska användas för att hantera styrning vid förlust av GPS-signal i minst 5 sekunder. Den här tiden har valt då det anses rimligt för robotgräsklipparen att befinna sig under ett träd i ungefär 5 sekunder innan den har kört förbi trädet.

Under iterationen uppstod det problem i programmeringen av IMU-modulen. IMU-modulen som användes till en början var en ISM330DLC [27] som enligt specifikationer använder SPI-kommunikation (Serial Peripheral Interface, en full duplex buss för synkron seriekommunikation) eller I2C-kommunikation (Interintegrerad krets, en synkron seriell multimaster/slav buss) mellan modulen och mikrokontrollern. Efter 2 veckors utveckling gav projektgruppen upp på att uppnå kommunikationen mellan ISM330DLC-modulen och Arduino-kortet och istället beställdes en MPU-6050 [28].

Under iterationen gick även en motor sönder vilket innebar att projektgruppen behövde felsöka och beställa en ny motor vilket försenade hela arbetsprocessen. Den tredje iterationen var dock medvetet vagt definierat för att ge utrymme för hantering av sådana händelser.

3.3 Iteration 3

I den tredje iterationen skulle den slutgiltiga produkten vara färdig. Den skulle uppnå den funktionaliteten som gör att den kan lösa de problem som beskrevs i kapitel 1.2. I den här iterationen lämnades också utrymme till att färdigställa uppgifter som dykt upp under projektets gång, vilket främst blev att felsöka hårdvara.

Den enda punkt som hade satts upp sedan tidigare var:

- Användaren ska med hjälp av webbapplikationen kunna definiera området gräsklipparen får navigera inom.

Under den här iterationen lades mycket arbete på att integrera de olika utvecklade delarna till ett sammakopplat system, efter att den nya IMU-modulen hade programmerats. Under iterationen lades även mycket tid på att få upp basstationen för att möjliggöra användning av RTKLIB.

Det som låg till följd av den sista iterationen var testning av det sammankopplade systemet, dock utfördes även utveckling och felsökning efter den sista iterationen. Det var på grund av att problem i systemet upptäcktes under testningen av robotgräsklipparen.

4 Implementation

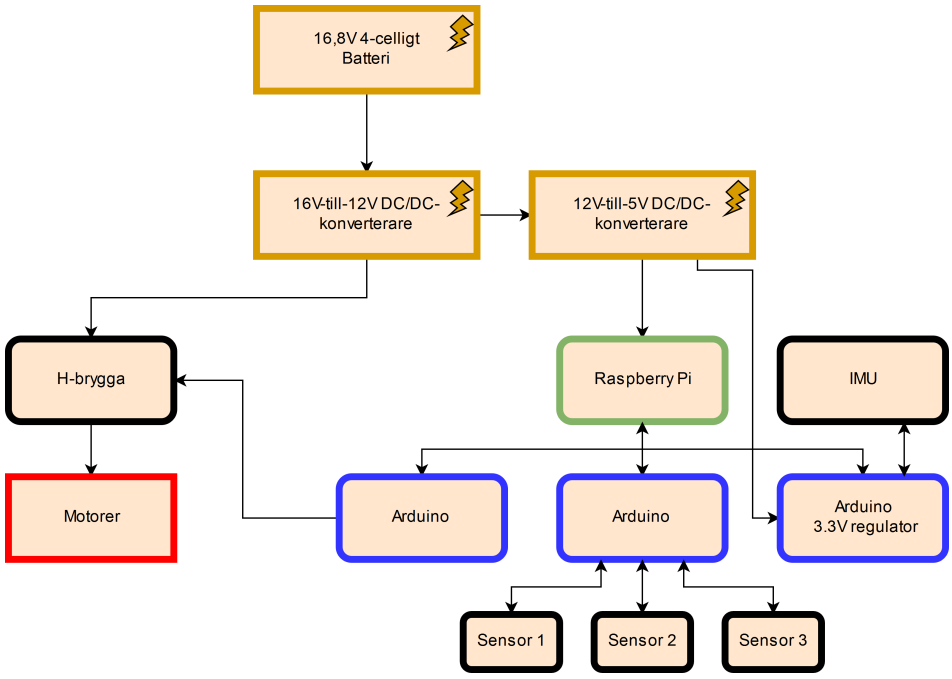
Den stora delen av arbetet som utfördes under projektets gång var att implementera mjukvaru- och hårdvarulösningar. De av systemets delar som implementerades var:

- Hårdvaran - robotgräsklipparens olika delar.
- Mjukvaran - de program som skrevs för att kontrollera och styra roboten.

- Navigationssystem - de olika delarna som behövdes för att styra robotrgräsklipparen, exempelvis GPS.
- Kommunikation - hur systemets olika delar kommunicerar med varandra.

4.1 Hårdvara

Som strömkälla till robotgräsklipparen användes ett 4-cells-LiPo-batteri på 16,8 V. Eftersom motorn drivs på 12 V samt att mikrokontrollerna drivs på 5 V användes DC/DC-omvandlare för att få ner spänningen till lämplig nivå. 12 V-omvandlaren kopplades till en H-brygga vilken kopplades till ett Arduino-kort samt motorerna. Genom att sätta två bitar till ett respektive noll från Arduinon till H-bryggan kan då polariteten ändras och därmed hjulens riktning. Överblick över kopplingarna finns i figur 7.



Figur 7: Kopplingsschema för robotgräsklipparen.

5V-omvandlaren kopplades till en RPI3 som kopplades till tre olika Arduino-kort via USB. Varje Arduino uppfyller en specifik funktion: En styr motorn, en läser och skriver till ultraljudssensorerna, samt en som kommunicerar med IMU-modulen.

IMU-modulen MPU-6050 [28] kräver max 3,5 V in, men utgångarna på mikrokontrollerkortet som modulen är kopplad till skickar ut 5 V från sina pinnar. På grund av det byttes 5 V-regulatorn på Arduinon ut mot en 3,3 V-regulator.

I projektet valdes tre Arduino-kort för att uppfylla de önskade funktionerna, däremot hade det varit möjligt att reducera antalet till två genom att sammanfoga funktionaliteten för motorerna och ultraljudssensorerna till ett Arduino-kort. Det är dock möjligt att en sammanfogning hade påverkat samplingsfrekvensen för ultraljudssensorerna, dessutom kräver sensorerna att mikrokontrollern väntar 20 mikrosekunder mellan varje sampling vilket möjligen hade påverkat motorernas funktionalitet då GPS:en har en uppdateringsfrekvens på 10 Hz och IMU-modulen en uppdateringsfrekvens på 40 Hz.

Eftersom det fanns tillgång till flera Arduino-kort valdes så många som möjligt för att säkerhetsställa högsta parallellism och därmed högre robusthet.

Under projektets gång visade det sig även att strömförsörjningen till Arduino-kortet genom USB från RPI3 inte var pålitligt, eftersom när värden skickades till motorn så kopplades IMU-modulen bort emellanåt vilket tydde på spänningsfall. Problemet löstes genom att koppla drivspänningen direkt från 12-5 V DC/DC-omvandlaren till Arduino-kortets DC-ingång.

4.1.1 Ultraljudsensorer

Robotgräsklipparen använder tre ultraljudsensorer av modellen Ultrasonic HC-SR04, funktionen beskrivs i avsnitt 2.2.2. Uppgiften för sensorerna är att avgöra ifall något objekt hindrar gräsklipparen från rörelse. Om sensorerna upptäcker ett blockerande objekt ska det kommuniceras till motorerna som då ska stanna och byta rörelseriktning. Sensorerna är placerade så att ett objekt som befinner sig rakt fram eller inom en 45 graders vinkel åt vadera sida ska kunna upptäckas.

Målet för avståndsmätningen är att robotgräsklipparen ska kunna upptäcka objekt som befinner sig med ett maxavstånd på 25 cm framför någon av de tre sensorerna. För att åstadkomma det behöver Arduino Uno-kortet kodas i motsvarande programvara. Koden består huvudsakligen av att först etablera en seriell kommunikation med RPI3. För att programmet ska kunna beräkna avståndet används ekvation 1. När ett objekt befinner sig framför minst en av sensorerna beräknar då programmet ett avstånd med hjälp av tiden som signalen har färdats och skickar vidare informationen via den seriella porten till RPI3. När informationen har nåtts ska denna behandlas av RPI3 och sedan kommunicera ut till motorerna att stanna och byta riktning ifall ett objekt befinner sig tillräckligt nära.

4.1.2 IMU

IMU-modulen som användes är en MPU-6050 [28] som mäter acceleration och vinkelacceleration lokalt för roboten. Modulen kommunicerar över I2C med en frekvens på 200 kHz och genererar data med en frekvens på 40 Hz. Modulen är fastmonterad på kretskortet GY-88[29] vilket även har en magnetometer (HMC5883L [30]) och en barometer (BMP085 [31]). De extra modulerna utöver IMU:n användes inte då de inte specificerades i syftet och hade ökat komplexiteten i sensorfusionen, implementeringen av drivrutinerna samt kopplingarna. Däremot finns det utvecklingsmöjligheter genom att använda magnetometern som beskrivs i avsnitt 7.3. Till IMU:n fanns det även ett *Digital Motion Processing*-bibliotek (Digitalt rörelseprocesseringsbibliotek) som användes för att räkna ut vinkeln längs med marken utifrån accelerometern och gyroskopet.

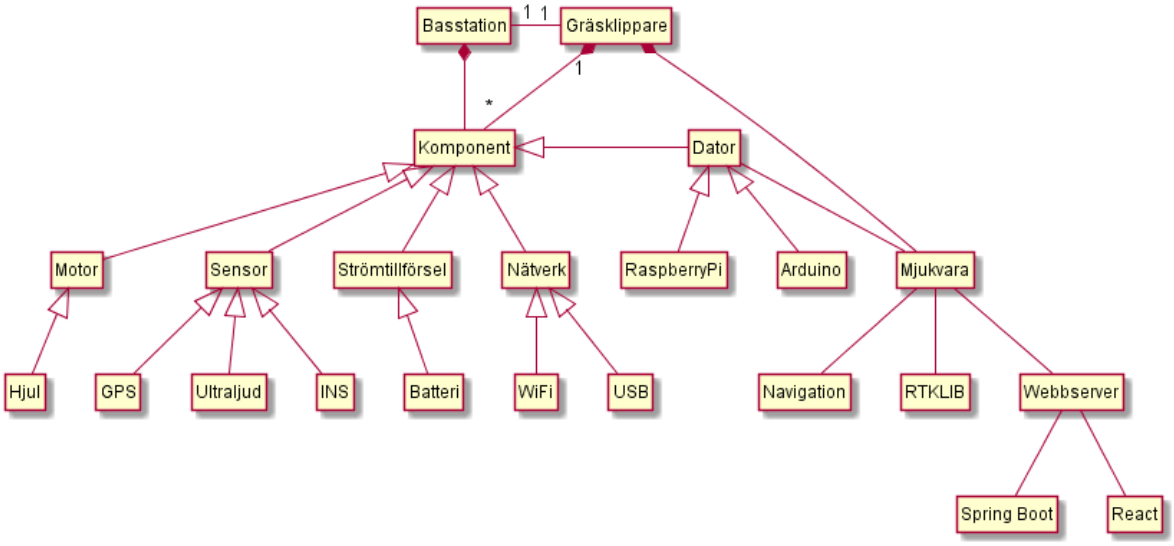
4.1.3 Motorstyrning

För motorstyrningens logik användes en Arduino Nano. Arduino-kortet får styrinstruktioner från en RPI3 genom en seriell kommunikationskanal. För att styra motorerna skickades pulsbreddsmodulerade signaler från Arduino-kortet till motorerna. Det var nödvändigt att koppla en H-brygga mellan Arduino-kortet och motorerna för att höja maxspänningen på grund av begränsad effekt från mikrokontrollerkortet.

Under projektets gång upptäcktes det att Arduino-kortet som styr motorerna emellanåt kopplades bort när polariteten på en motor ändrades. För att försöka undvika att det inträffar igen programmerades det in logik där robotgräsklipparen stannade en sekund innan den fortsatte köra. Lösningen eliminerade problemet för det mesta, förutom när flera styrinstruktioner med olika polariteter gavs nära på varandra (vilket inte är något som händer under robotgräsklipparens automatiska navigation).

4.2 Mjukvara

Mjukvarustrukturen som utvecklades under projektet kan översiktligt beskrivas med hjälp av en domänmodell. En domänmodell är en konceptuell modell över strukturen av olika komponenter i en domän. Den domänmodell som togs fram för att användas i projektet kan ses i figur 8.



Figur 8: Domänmodell över systemet

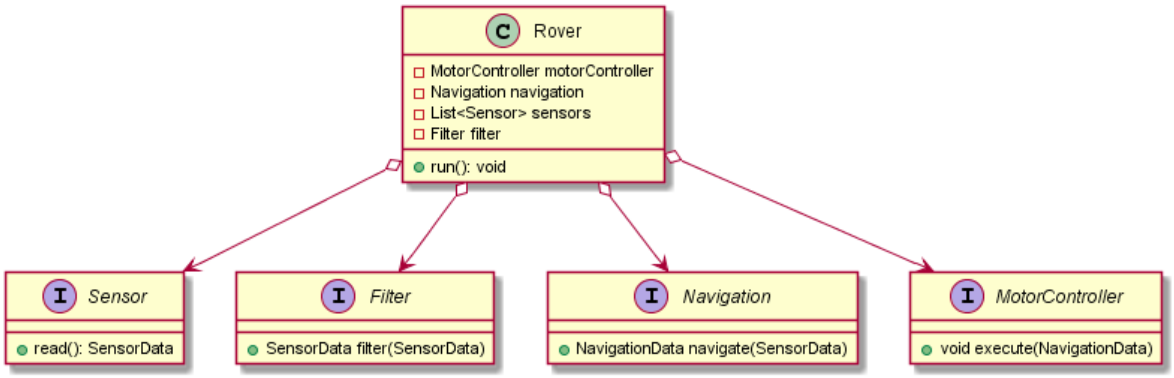
I modellen visas systemets olika delar samt hur de hänger ihop. Det visas hur robotgräsklipparen har komponenter vilka är

- motor
- sensorer
- strömtillförsel
- nätverk
- dator
- basstation.

De här komponenterna har, som visas i modellen, underkomponenter som alla återfinns på robotgräsklipparen, förutom *Basstation*:en, som är placerad på ett tak. Datorn som finns på robotgräsklipparen kör *Mjukvara* som kan delas in i tre olika delar, vilka är

- navigationslogik
- RTKLIB
- webbserver.

Navigationslogiken är den delen av mjukvaran som beräknar hur robotgräsklipparen ska köras beroende på värden som beräknas av robotgräsklipparens sensorer. RTKLIB är den del av mjukvarusystemet som hanterar användningen av en *Basstation* för att öka precisionen på positionsbestämningen. *Webbserver* är den applikation som användaren kan använda för att styra och kontrollera robotgräsklipparens drift.



Figur 9: Ett UML-diagram över gränssnitt som agerar grundstenar för gräsklipparens funktionalitet. Vissa metoder är borttagna för att förenkla diagrammet.

Utifrån domänmodellen var det möjligt att bygga de grundstenar som möjliggör gräsklipparens funktionalitet samtidigt som en hög modularitet uppnåddes. Se figur 9 för en överblick av de centrala gränssnitten. Huvudobjektet som gräsklipparen består utav är en *Rover* vilket har en koppling till en *MotorController*, *Filter*, *Navigation* och en lista av *Sensor*-objekt. Varje gång en cykel av gräsklipparens logik förväntas köras kallas funktionen *run()* i klassen *Rover* vilket resulterar i att *SensorData* läses av från listan av *Sensor*-objekten och skickas in i *Filter*-objektet, värderna från filtret skickas därefter in i *Navigation*-objektet där ett *NavigationData*-objekt skapas som definierar hur gräsklipparen bör köra. Slutligen skickas *NavigationData*-objektet in i *MotorController* vilket får de kopplade motorerna att rotera i den förväntade riktningen.

Fördelen med mjukvaruarkitekturen som beskrivs ovan är att den är modulär. Modulariteten har varit användbar under utvecklingen eftersom det har varit enkelt att skapa olika sensorer samt filter och testa dem i olika konfigurationer. Det har också gjort det möjligt att skapa en simulator som använder samma logik som det verkliga systemet eftersom det går att skapa simulerade instanser av både sensorer och motorer. Simulatorn beskrivs närmre i avsnitt 4.2.3.

4.2.1 Filtrering

För att kombinera samt filtrera accelerationsdata från IMU-modulen och positionsdata från GPS:en har pythonbiblioteket *pykalman* använts som beskrivs i avsnitt 2.3.3. En pythonklass har utvecklats vilket skapar ett sigmapunktspyklamanfilter, definierar nödvändiga matriser och funktioner, samt implementerar *pykalmans* realtidsfilteringsfunktion *filter_update*. Dessutom innehåller klassen funktioner för att spara och rita ut filtrerad samt ickefiltrerad data för jämförelse.

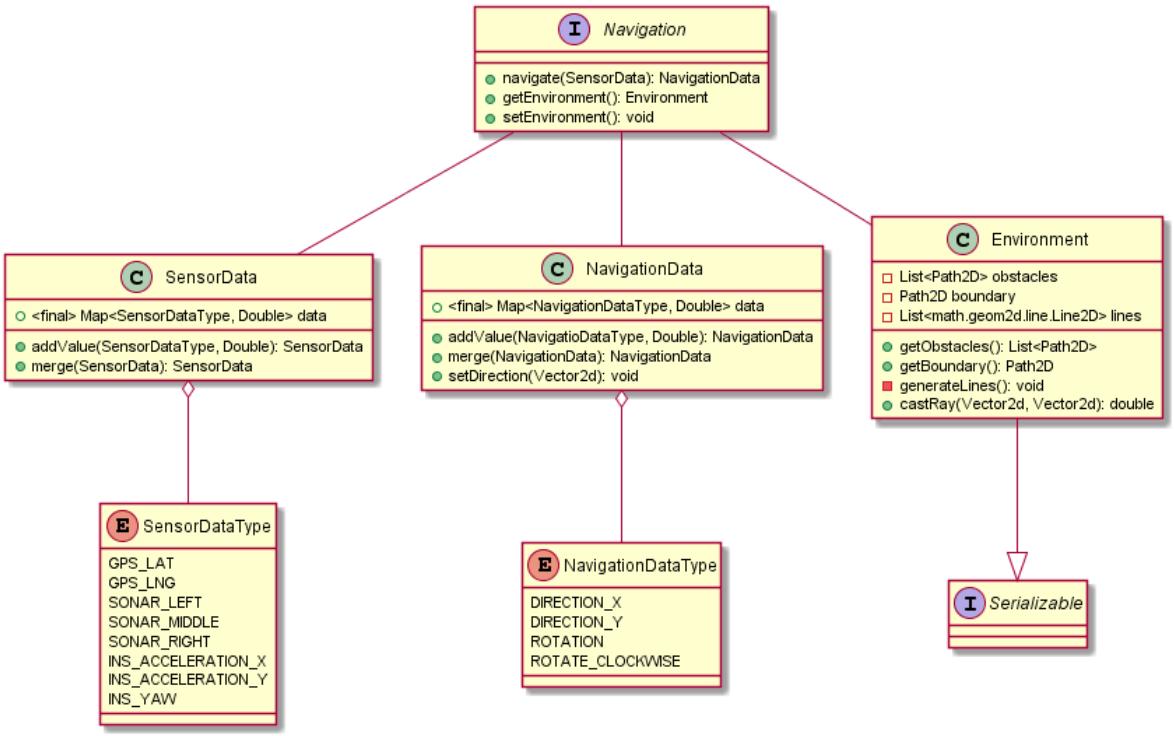
Initialvärdena ansattes godtyckligt medan transformations- och observationskovariansen bestämts genom kalibrering av filtret. Positions- och accelerationsdata samlades in för att uppskatta avvikelsernas storleksordning och därifrån valdes transformations- och observationskovariansen för att uppnå önskad viktning av observationerna och uppskattningar. För att kunna definiera transformations- och observationssmatriserna och funktionerna behövdes ett antal transformationsfunktioner skapas för att kunna relatera datan, på grund av att IMU:en och GPS har två olika koordinatsystem. Därför definierades ett gemensamt koordinatsystem utifrån den initiala positionen med x- och y-axeln i östlig respektive nordlig riktning till vilken positions- och accelerationsdata transformerades för att kunna tillämpa den kända relationen mellan en position och acceleration. För transformation mellan latitud, longitud och xy-koordinater användes pythonbiblioteket *distance* från *GeoPy*. x- och y-positionen erhölls genom projektion av avståndet på latituden respektive longituden, för tester påvisade att felen blev försumbara inom de avstånd filtret tänkts tillämpas. Felet uppskattades till 10^{-6} m vid en avståndmätning på 100

m vid Chalmers latitud och longitud. Övriga transformationer kunde göras med hjälp av IMU-data.

Klassens filtreringsfunktion är den funktion som använts för själva filtreringen. Den tar observationerna (latituden och longituden, accelerationen, rotationen och tiden) som parametrar. Den sätter klassens tid och riktning vilket behövs till transformations- och observationsfunktionerna samt koordinatransformationerna. Därefter transformerar den observationsdatan innan den filtreras med *filter_update*. Koordinattransformationen har placerats här istället för i transformations- och observationsfunktionerna på grund av att det inte gick att ansätta tillräckligt låga värden på kovariansen för filtrering i latitud och longitud utan att *SciPy* genererar fel. Felet genereras då det skapas negativa egenvärden och där med en negativt definit matris vilken inte går att faktorisera till en undertriangulär matris och motsvarande konjugerade transponat. Filterfunktionen transformerar tillbaka och sparar därefter data innan den returnerar de filtrerade positionerna.

Den ovannämnda koden är skriven i Python men resten av gräsklipparsystemet är skrivet i Java. Därför har biblioteket *Jep* används, vars förkortning står för **J**ava **E**mbedded **P**ython [32]. Det har används för att kunna exekvera pythonkod i en Java-miljö. Det finns därför ett *KalmanFilter*-objekt som implementerar *Filter* gränssnittet, när *filter(SensorData)* kallas kommer relevanta sensorvärden att skickas genom *Jep* till pythonkoden. När resultatet från pythonfiltret kommer tillbaka returneras det i form av ett nytt *SensorData*-objekt.

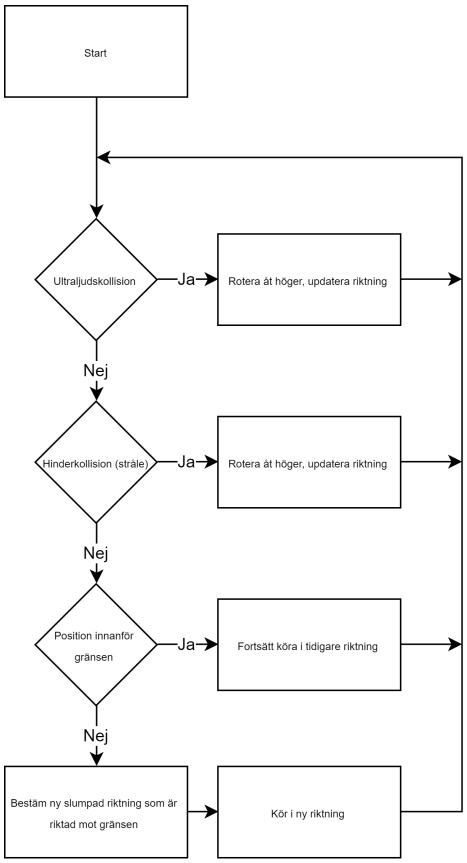
4.2.2 Navigationslogik



Figur 10: Ett UML-diagram över klasser kopplade till navigationslogik.

Navigationslogiken bestämmer utifrån olika villkor och parametrar hur gräsklipparen ska styra motorerna och på så sätt vilken rutt som ska köras. Logiken består av en klass som implementerar gränssnittet *Navigation* där den erhåller funktionen *navigate(SensorData)* och möjligheten att få information om

Environment. *SensorData* beskriver värden som senast lästes in från robotgräsklipparens sensorer så som position och rotation. Innan *SensorData*-objektet når *Navigation* implementationen skickas datan igenom en filteringsprocess, se avsnitt 2.3. *Environment* beskriver hur området navigationen behöver ta hänsyn till ser ut i form av användardefinerade hinder och en avgränsning bestående av polygoner. Utifrån de här två datakällorna är det möjligt att besluta hur gräsklipparen bör köra i nästa tidpunkt i form av *NavigationData* som skickas till en *MotorController*. En mer detaljerad koppling visas i figur 10.

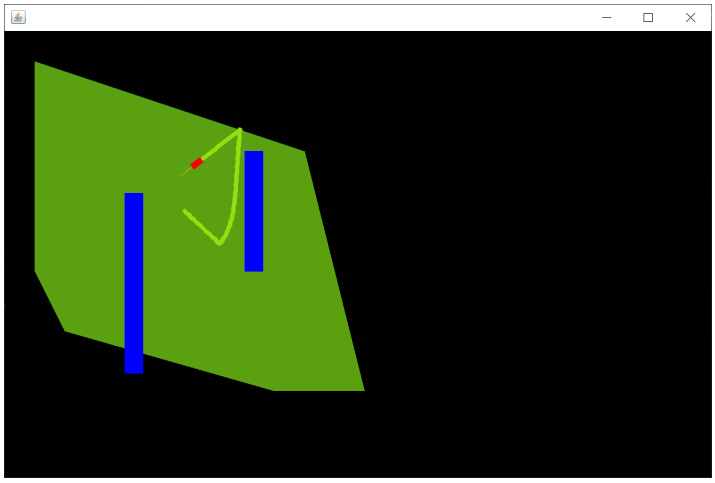


Figur 11: Flödesdiagram över navigationslogik.

En överblick över hur navigationslogiken fungerar ges i figur 11. Det första som görs är att kolla om ultraljudssensorerna registrerar ett avstånd på mindre eller lika med 25 cm. Om villkoret uppfylls kommer gräsklipparen att rotera åt höger tills det inte längre är fallet. Samma logik tillämpas till de hinder som är definierade i *Environment*, fast istället för fysiska ultraljudsvågor skickas virtuella strålar i olika riktningar som beräknar avstånd till närmaste hinder. Nästa steg är att avgöra om gräsklipparen befinner sig inom avgränsningen. Om gräsklipparen är inom området kommer den fortsätta köra i den tidigare definierade riktningen. Annars kommer en ny riktning beräknas som resulterar i att gräsklipparen kör in i området igen. Riktningen beräknas genom att slumpa fram positioner i en växande radie runt om gräsklipparen för att sedan kolla om de är innanför området. Om en av de slumpade positionerna är innanför området kommer en vektor till den utifrån gräsklipparens position sättas som den nya körriktningen. Den här proceduren fortsätter under en såpass lång tid att hela gräsmattan antas ha blivit klippt, på ett liknande sätt som många av dagens robotgräsklipparen fungerar exempelvis Robotic MowerRX50 Pro S, RX50u,RX20 Pro, RX20u, och RX12u [33].

4.2.3 Simulering

Det utvecklades en mjukvarutestmiljö som gjorde det möjligt att testa navigationssystemet utan att använda hårdvaran. Det sågs som fördelaktigt då det tillät utveckling av navigationssystem parallellt med arbete för att få hårdvaran att fungera. Den tidigare nämnda mjukvaruarkitekturen är modulär vilket möjliggjorde testkörning av hela körlogiken genom att enbart skapa simulatorobjekt av sensorer och motorer.



Figur 12: Simulering av navigation inom ett hårdkodat område. Den röda rektangeln representerar gräsklipparen, det ljusgröna strecket den sträcka den kört och det gula strecket den nuvarande riktningen

Simulatorn fungerar genom att låta gräsklipparen representeras av en punkt. Positionen på punkten skickas via ett *Senor*-objekt som simulerar en GPS. Sedan simuleras även motorn vilket får punktens position att ändras när den tar emot navigeringskommandon. Positionsändringen baseras på en modell för differentialdrift, se ekvation:

$$\begin{aligned} \Delta x &= v \cos(\theta) \\ \Delta y &= v \sin(\theta) \\ \Delta\theta &= \omega \end{aligned} \tag{2}$$

I modellen är v hastigheten i nuvarande riktning och ω vinkelhastigheten.

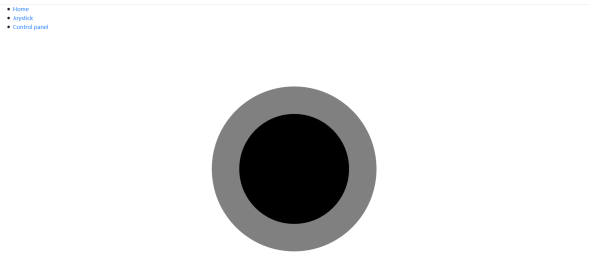
Ultraljudssensorerna simulerades genom att skicka virtuella strålar på precis samma sätt som nämns i navigationslogiken för att upptäcka fördefinerade hinder.

4.2.4 Webbapplikation

Webbapplikationen användaren interagerar med drivs av JavaScript-ramverket React [34]. Webbapplikationen kommunicerar med en REST-server, vilket är en server som kan utföra maskin-till-maskin-kommunikation genom URL-kommandon, för att på olika sätt kontrollera gräsklipparen. Under utvecklingen kompilerades hemsidan med Webpack [35].

För att manuellt kunna styra robotgräsklipparen, genom att skicka instruktioner till motorerna via Arduino-kortet, implementerades en joystick-komponent i webbapplikationens gränssnitt, se figur 13. Komponenten ger ut X- och Y-värden baserat på hur joysticken dras. De här värdena ger dock inte motorerna nog med information för att få gräsklipparen att röra sig som förväntat. För att få en önskad styrning av gräsklipparen bearbetades joystickens värden för att få motorerna att arbeta på rätt sätt.

Ett exempel är att om joysticken hålls rakt fram så går båda motorerna på full kraft, vilket leder till att gräsklipparens båda hjul snurrar framåt, vilket leder till att den går rakt fram. Bearbetningen av de nya värderna sker dock inte i webbapplikationen, utan i REST-servern.



Figur 13: Webbapplikationens joystick-komponent.

Ett av målen med webbapplikationen var att låta användaren kontrollera och få översikt över robotgräsklipparens position i realtid. För att ge användaren en översikt av navigationen användes GoogleMapsAPI [36]. Med hjälp av API:et implementerades en karta där användaren kan skapa en polygon som representerar området som användaren vill ha klippt. Det här området kan vid ett knapptryck skickas till REST-servern som finns på den RPI3 som finns på gräsklipparen. Vidare utvecklades också funktionalitet för att spara den senaste polygonen så att den också laddas upp och används vid nästa uppstart, vilket innebär att användaren inte behöver markera sin trädgård om och om igen. Utöver funktionaliteten som innebär att skicka den markerade polygonen till REST-servern implementerades även en knapp som tar bort det markerade området. Ytterligare implementerades även en knapp som användaren kan använda för att direkt stoppa gräsklipparen.

Slutligen lades det även till funktionalitet som gör det möjligt att placera ut flera olika zoner där gräsklipparen inte får åka. De zonerna representeras av polygoner och ger användaren möjlighet att markera ut objekt som dammar eller rabatter där robotgräsklipparen inte ska åka. Precis som med den polygon som representerar det området som ska klippas, skickas polygonerna genom ett knapptryck till REST-servern som hanterar navigationen.

4.2.5 REST-server

REST-servern drivs av Spring Boot, som är en del av Spring-ramverket [37]. Den tar emot kommandon och data från webbapplikationen, och utför olika åtgärder som påverkar navigationen av gräsklipparen. Den har även en koppling till ett *Rover*-objekt och dess motorer och sensorer. Det tillåter servern att med hjälp av ett navigationssystem manövrera gräsklipparen genom att skicka värden till robotgräsklipparens motorer som får hjulen att snurra.

En funktionalitet som utvecklats är att de zoner man placerat ut, både område att klippa och som inte ska klippas vilket representeras av ett *Environment*-objekt, sparas i en .ser-fil som gör det möjligt för robotgräsklipparen att ”minnas” områden mellan omgångar av klippning. Det gör att användaren slipper definiera områdena på nytt varje gång som gräset ska klippas.

4.3 Konfiguration och installation av GPS-system

Då GPS-systemet som skulle användas i projektet var ett RTK-system med två moduler var det mycket som behövde installeras och konfigureras för att det skulle fungera. Hur de olika modulerna installeras beskrivs i det här avsnittet.

4.3.1 Installation av RasPiGNSS

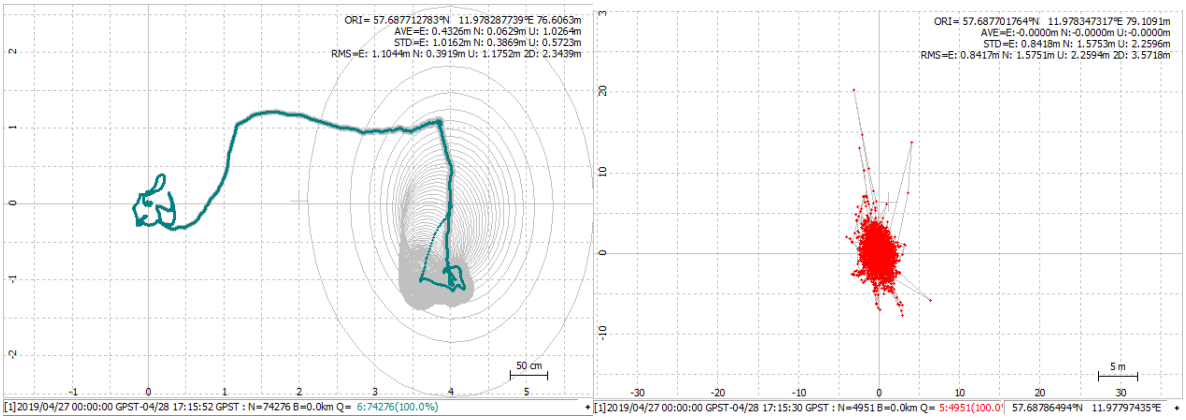
RasPiGNSS monteras på GPIO-pinnarna som finns på en RPI3. Efter det måste operativsystemet konfigureras och drivrutiner installeras för att möjliggöra läsning från GPS-modulen. Instruktioner för installationen finns på hemsidan för RaspPiGNSS [11]. Svårigheten låg i att instruktionerna är primärt gjorda för en Raspberry Pi 1, med små tillägg för hur det ska göras på en RPI3. Dock var tilläggen inte tillräckliga då det uppstod problem vid installationen av perl-biblioteket för BCM2835 [38]-modulen. Den modulen är vad som kontrollerar GPIO pinnarna. Det visade sig vara på grund av att C-biblioteket för BCM2835 inte var installerat. Biblioteket gick dock att ladda ner och bygga vilket ledde till en lyckad installationen av perl-bibloteket.

4.3.2 RTKLIB

Basstationen består av en RPI3 vilken kör programmet STR2STR som startar en TCP-server där GPS-rådata i form av NVS-BINR skickas till anslutna klienter. Programmet gör inga beräkningar utan vidarebefordrar enbart data till robotgräsklipparens RPI3 som är ansluten till dataströmmen via en TCP-klient i programmet RTKCV. RTKCV är ett program som processar GPS-data i realtid utifrån ett antal olika datakällor och konfigurationer. De här konfigurationerna beror primärt på vilket *Positioning Mode* som används, i projektet används *Kinematic* vilket motsvarar RTK med en bas och rover, som i detta projekt är robotgräsklipparen. Den beräknade positionen som bestäms av RTKCV skickas sedan vidare via en TCP-server till Java-programmet.

RTKLIB har utöver programmen ovan ett antal ytterligare program som har använts i projektet. Vid felsökande och testande av olika inställningar har RTKNAVI varit användbart då det är en grafisk version av RTKCV vilket snabbt ger en överblick över GPS:ens status. I de fallen RTKNAVI används skickar både rovern och bastationen rådata till RTKNAVI via STR2STR. Dessutom finns RTKPOST vilket fungerar som RTKNAVI och RTKCV, dock inte i realtid då det här programmet använder loggad rådata som datakälla. RTKPOST har varit användbart för att testa olika konfigurationer på samma dataset för att sedan jämföra resultaten. Med RTKPOST går det även att lägga till ytterligare datakällor från andra GPS-stationer samt information om satelliternas omloppsbanor.

4.3.3 Basstation

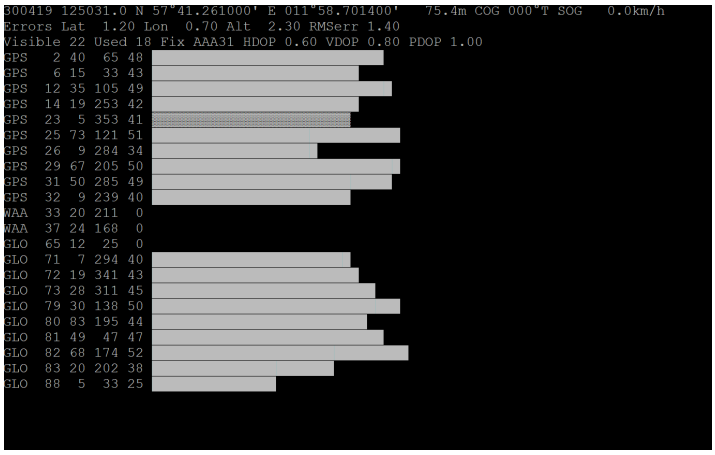


(a) Latitud och longitud med *Positioning Mode*: *PPP Static*. (b) Latitud och longitud med *Positioning Mode*: *Single*.

Figur 14: Plottar över efterprocesserad GPS-data med olika *Positioning Mode* över ett dygn.

För att få ökad precision vid användning av GPS användes tekniken RTK, vilket beskrevs i avsnitt 2.1.1. För att använda tekniken krävdes det att en basstation installerades på ett tak. Därefter behövdes en exakt position bestämmas för att möjligöra absoluta, istället för relativa, värden i RTK-beräkningarna. En sådan position bestäms genom att låta basstationen logga rådata under en längre tid. Sedan används rådatan för att bestämma en exakt position i RTKPOST, se figur 14.

För att *Positioning Mode*:*PPP Static* ska optimalt bestämma en position krävs uppkoppling till så kallad WAAS- eller SBAS-satelliter. Under utvecklingen och testningen av RTK-systemet visades det att de nödvändiga satelliterna inte var tillgängliga, se figur 15, vilket gjorde att positionen på basstationen inte kunde fastställas utan driftade runt istället vilket kan ses i figur 14. På grund av det här användes *Positioning Mode*: *Single* istället vilket gav en medelposition som verkade bättre. Det ska noteras att när positionen kollas upp på Google Maps så stämmer den inte överens med var den är i verkligheten, utan är några meter skiftad.

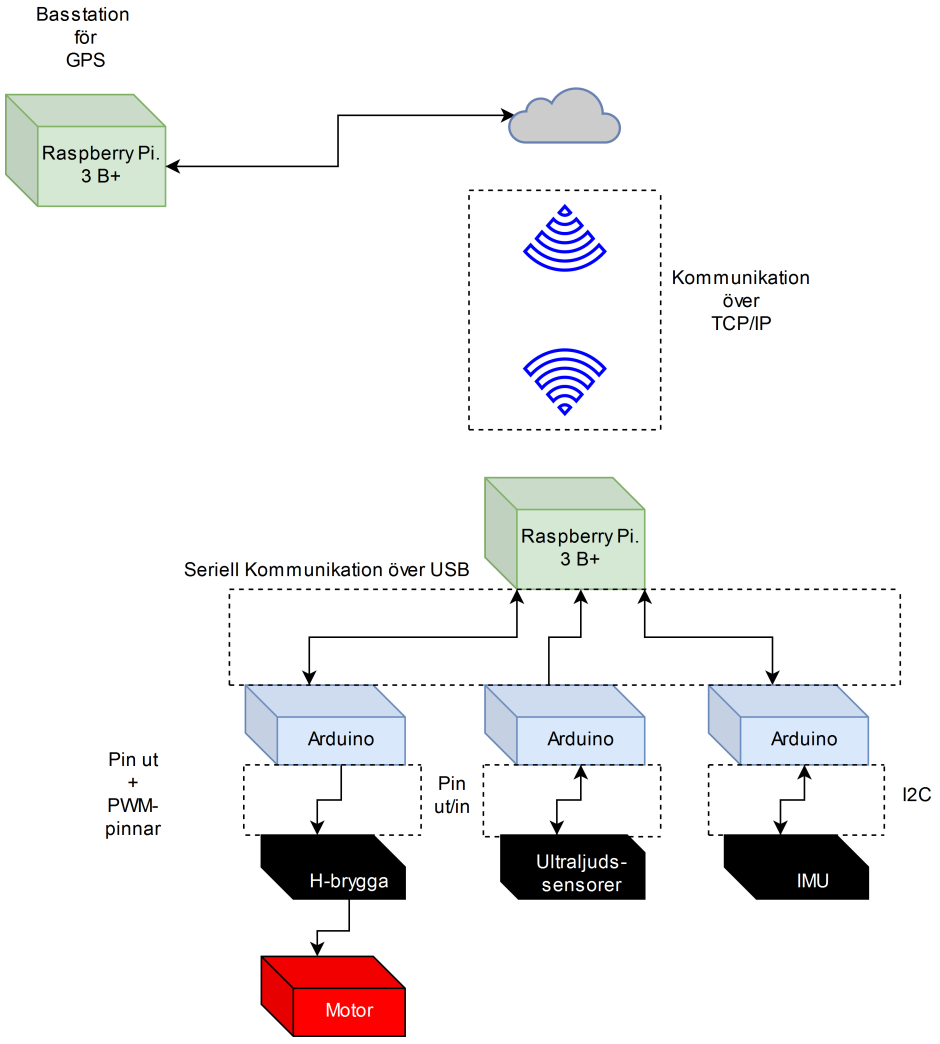


Figur 15: Bild på terminalen som visar uppkoppling med satelliterna. De gråa linjerna visar signalstyrkan; notera att WAA-satelliterna inte har någon signal med GPS-modulen.

4.4 Kommunikation

Eftersom flera oberoende delar kommunicerar med varandra samt att arbetet delades upp parallellt var det viktigt att ha ett väletablerat kommunikationssystem för att säkerhetsställa hela systemets funktionalitet. Ett diagram över systemets slutgiltiga kommunikation finns i figur 16.

Då GPS-modulen tar upp flertal pinnar från RPI3-modulen valdes seriell kommunikation över USB mellan RPI3-modulen och Arduino-korten. Det uppstod då problem i och med att flera Arduino-kort använder samma namn på sina seriella portar vilket gav konflikter vid uppstart. Lösningen på problemet var att länka specifika namn till specifika enhetsnoder genom att lägga till nya regler för enhetshanteraren i operativsystemkärnan.



Figur 16: Kommunikation inom roboten samt mellan roboten och basstationen.

Kommunikationen mellan Arduino-korten och sina respektive moduler är relativt simpla. MPU-6050 är byggd för att kommunicera över I2C, ultraljudssensorerna kräver endast positiva och negativa flankar, och H-bryggan behöver PWM-signaler för att styra motorerna i olika hastigheter. Utöver PWM-signaler skickar även Arduino-kortet aktivationssignaler för att styra hjulens riktning.

För att kunna använda RTKLIB användes en basstation som tar emot rådata från satelliterna. Eftersom basstationen var monterad uppe på ett tak skickades datan till roboten genom TCP/IP som sedan använde datan för positionsberäkningar.

5 Testning

Navigationssystemet som utvecklats under projektet testades på tre olika sätt. De olika testen utfördes på samma område, för att kunna jämföra de olika resultaten. Området ritades även ut med gatkritor för att lättare kunna se hur robotgräsklipparen förhåller sig till området i verkligheten. Alla tre test utfördes genom att låta gräsklipparen köra inom det markerade området tills den åker ut, eller i 30 minuter. För att kunna utvärdera filtrets funktionalitet samt för att kunna få en referenspunkt att jämföra testdatan med genererades simulerad data som filtret tillämpades på. Det möjliggör en bättre överblick av filtrets funktion på optimal data jämfört med brusig data.

Målet med testerna var att se huruvida de olika användningarna av navigationssystemet klarar av att hålla den inom ett markerat område (se figur 17), vilket var syftet med projektet. Testerna utformades på följande vis:

1. Det första testet var att låta gräsklipparen köra utan att använda det utvecklade Kalmanfiltret, utan istället endast använda positionsdata från RTKLIB och rotationsdata från IMU-modulen.
2. Det andra testet var att använda det utvecklade Kalmanfiltret under hela testet.
3. Det tredje testet var att använda det utvecklade Kalmanfiltret, men att en gång i minuten stänga av GPS-signalen i 10 sekunder, vilket gjordes för att simulera att robotgräsklipparen körde under ett träd.

Figur 17: Området där gräsklipparens förmåga att hålla sig inom området testades. Markören på kartan representerar robotgräsklipparens position och pilen representerar riktningen som den är vinklad i, hämtad från IMU-modulen.

Utöver förmågan att hålla sig inom ett utmarkerat område innehåller robotgräsklipparens navigationssystem två sätt att undvika hinder. Det ena är att undvika fördefinierade hinder, så som rabatter, genom att markera ut dem på kartan i webbapplikationen. Det andra är att robotgräsklipparens ultraljudssensorer ger utslag när ett oförutsett objekt är i vägen för robotgräsklipparens färdriktning. De här olika teknikerna testades på två olika sätt.

Det första sättet som förmågan att undvika hinder testas är att i webbapplikationen skapa ett område med ett tydligt hinder inuti området. Sedan fick robotgräsklipparen köra i området i 10 minuter. Slutligen jämfördes hur robotgräsklipparen kört (som plottades kontinuerligt av RTKLIB) med hur

området såg ut, för att jämföra om den faktiskt undvikit hindret.

Den andra funktionen som testades var robotgräsklipparens förmåga att undvika oförutsedda hinder med hjälp av ultraljudssensorerna. Det testades genom att ett område markeras ut där robotgräsklipparen fick köra. Medan robotgräsklipparen körde ställde sig en person i robotgräsklipparens färdriktning. Huruvida robotgräsklipparen lyckades undvika hindret observeras och noteras. Det upprepades 10 gånger och resultatet sammanställdes.

6 Resultat

Projektets slutliga resultat kan delas in i två huvuddelar. Den första huvudelen är webbapplikationen som skapats, medan den andra huvuddelen är navigationssystemet. De resultat som uppnåddes i de två delarna diskuteras i följande kapitel.

6.1 Webbapplikation

Webbapplikationen uppfyllde slutligen alla mål som var satta, då den funktionalitet som planerades förverkligades. Dessa funktioner var följande:

- Förmåga att visa robotgräsklipparens position på en karta.
- Förmåga att markera ett område på kartan som representerar ett område som ska klippas.
- Förmåga att stoppa robotgräsklipparen.
- Förmåga att markera ut flera hinder.

Figur 18 visar hur det ser ut när ett område, samt hinder, har markerats ut. Den visar också de olika knappar användaren kan använda för att markera ut dessa områden och hinder. Figur 19 visar de olika knappar som finns i webbapplikationens kontrollpanel. Funktionaliteten hos den vänstra knappen är att stänga av gräsklipparens självkörning, vilket gör att den stannar, medan den högra stänger av GPS-mottagningen, vilket användes för att testa navigationssystemets robusthet.

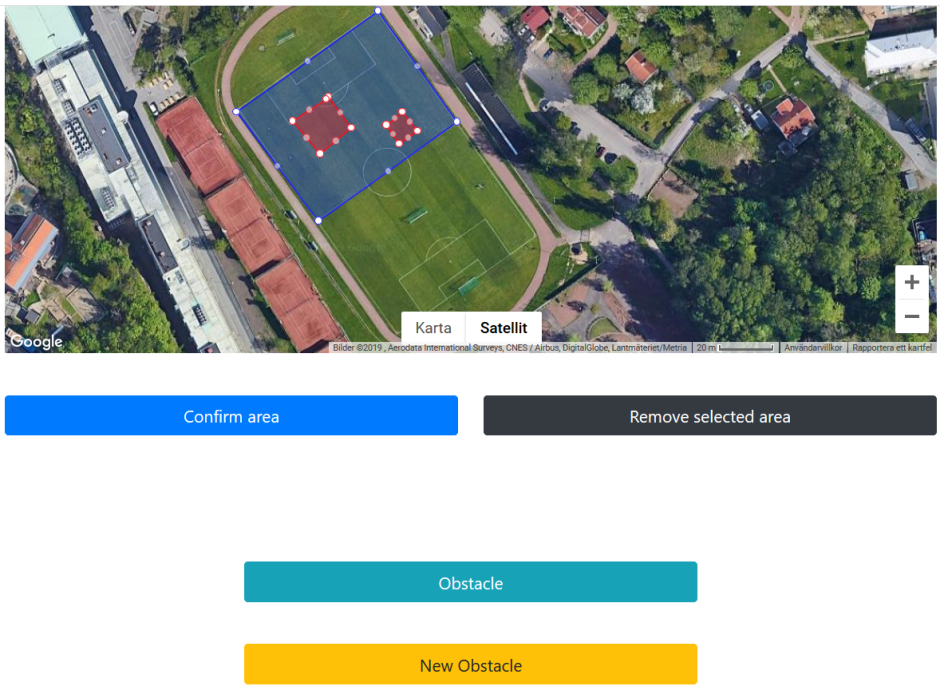
6.2 Resultat av tester

I det här avsnittet presenteras resultatet av de tester som utfördes. De olika testerna bestod av tre tester av navigationssystemet samt två tester av undvikande av hinder.

6.2.1 Test 1 - Endast GPS

Resultatet av det första testet var att robotgräsklipparen höll sig inom det markerade området under hela testet, vilket var i 30 min. I figur 20 syns mönstret som bildats av plottningen av hur den kört under testet. Mönstret har samma form och proportioner som det markerade området, vilket innebär att den har följt de begränsningar som markerats ut. Enstaka undantag uppstod dock, då den vände först någon decimeter utanför området.

Området som markerades ut valdes i linje med vissa objekt som syntes i verkligheten och markerades ut med gatkrita, men när robotgräsklipparen observerades i verkligheten syntes det att den följde ett område som var skiftat ungefär en meter från området markerat på kartan. Exempelvis var den ena gränsen av området längs med ett tak, men robotgräsklipparen åkte alltid förbi den gränsen och vände alltid ungefär en meter efter. På motsvarande sätt vände den en meter innan gränsen på motstående

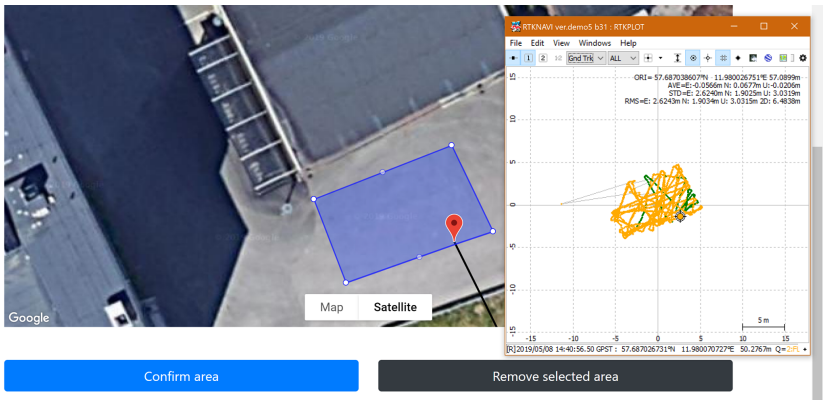


Figur 18: En bild från webbapplikationen. En karta där ett område, i blått, där robotgräsklipparen ska köra har markerats. Områdena i rött representerar hinder som robotgräsklipparen inte får köra i. Det finns knappar för att bekräfta ett markerat område, för att ta bort de markerade områdena, för att välja om hinder eller område ska markeras ut, samt om ett nytt hinder ska skapas.

Figur 19: En bild från webbapplikationen. Knappar som kan stoppa robotgräsklipparen, samt stänga av GPS-mottagningen hos robotgräsklipparen, vilket användes vid testning.

sida, vilket tyder på att navigationssystemet uppfattade det markerade området som förskjutet ungefär en meter från hur det såg ut i verkligheten.

Om endast det som visas på webbapplikationen tas i beaktning såg det hela tiden ut som att den höll sig inom området, förutom enstaka gånger då den körde någon decimeter utanför. Det tyder på att om positionsbestämningen hade varit mer exakt hade navigationssystemet fungerat som förväntat, då själva navigationslogiken fungerar.



Figur 20: Det blåa området är området där robotgräsklipparen ska köra, markören är robotgräsklipparen och pilen representerar den riktning dit robotgräsklipparen är vänd. Bilden till höger representerar hur den har kör under testet. Där strecket är grönt har GPS-mottagningen varit god och positioneringen träffsäker, medan det gula strecket representerar när GPS-mottagningen varit sämre och positioneringen varit lite mer osäker.

6.2.2 Test 2 - Navigation med Kalmanfiltret

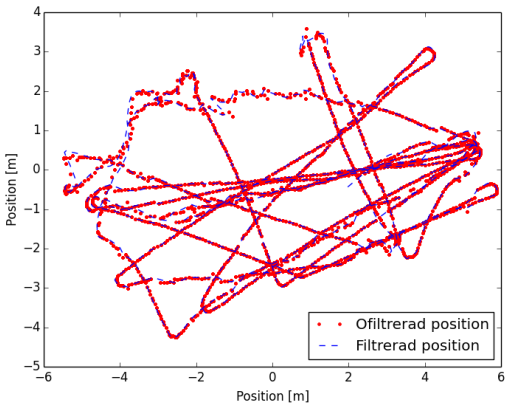
Robotgräsklipparen kunde köra i 19 minuter och 20 sekunder innan den åkte utanför området och inte lyckades komma in igen. Det berodde på att den fastnade. På samma sätt som i det första testet följde den det markerade området enligt vad navigationssystemet självt trodde att den var, det vill säga att om endast webbapplikationens visualisering beaktades såg det ut som att den höll sig inom området. Undantaget var att den ett fåtal gånger inte vände tidigt nog och därför hamnade en bit utanför, innan den vände in igen.

Positionen skiftade flera gånger under testets gång, till skillnad från test 1, då den var skiftad på samma sätt hela tiden. I verkligheten lyckades robotgräsklipparen alltså inte hålla sig inom det med krita utmarkerade området, men inom skiftningar av det.

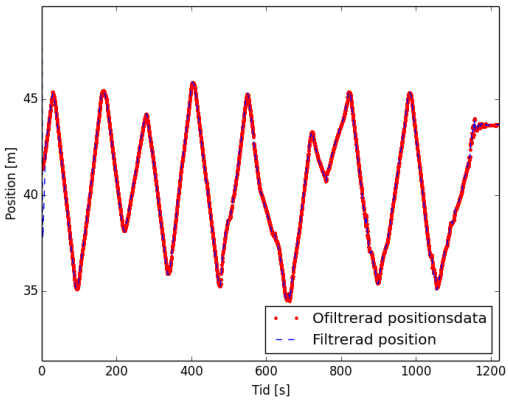
Den filtrerade positionen från test av filtret inom området i figur 17 utan blockering av GPS-signalen presenteras i figur 21a och 21b. GPS-signalen var stabil under testningen vilket innebär att den filtrerade och ofiltrerade datan i figur 21a och 21b överensstämmer. För att därför säkerhetställa filtets funktion adderades ett brus till den ofiltrerade datan vilket presenteras i figur 22a och 22b. Accelerationsdatan, presenterad i figur 23 däremot följer inte något tydligt mönster.

6.2.3 Test 3 - Navigation med Kalmanfiltrer vid förlust av GPS-signal

Test av navigation med förlust av GPS-signal inom området i figur 17 presenteras i figur 24a, 24b och 24c. Robotgräsklipparen kunde köra i 7 minuter och 27 sekunder innan den åkte utanför och inte lyckades komma in igen. Figurerna påvisar, i synerhet figur 24b och 24c, att resultatet beror på gräsklipparens rörelse innan GPS-signalen går förlorad samt att felen divergerar med tiden. Testet utfördes även på simulerad testdata vilket presenteras i figur 25 vilken påvisar ett bättre resultat det presenterat i figurerna 24b och 24c

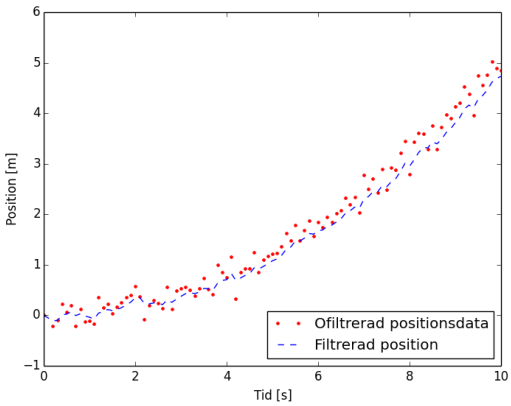


(a) Positionen i x- och y-led.

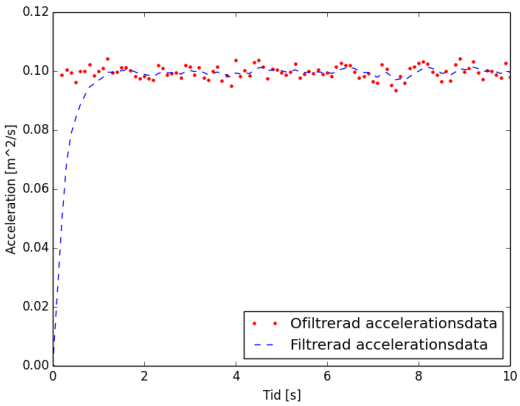


(b) Positionen i x-led över tiden.

Figur 21: Plottar över det filtrerade och ofiltrerade positionerna på insamlad mätdata.



(a) Positionen i x-led över tiden.

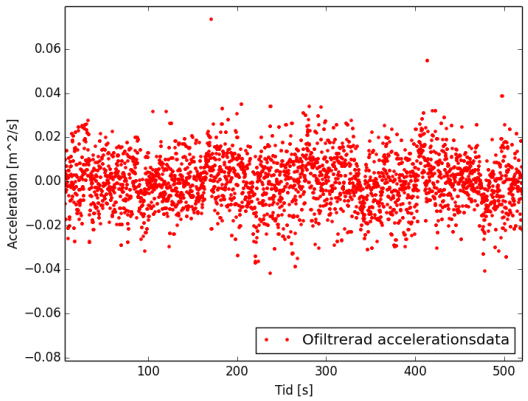


(b) Accelerationen i x-led över tiden.

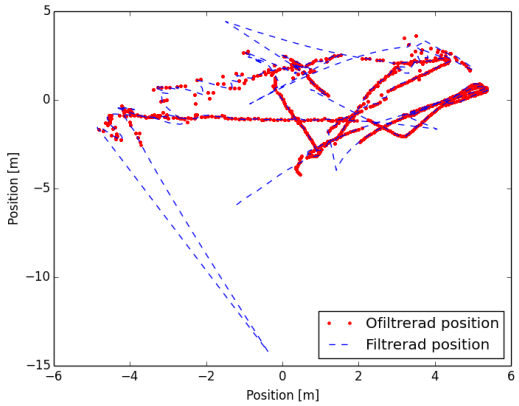
Figur 22: Plottar över det filtrerade och ofiltrerade position och acceleration på sjävgenererad test data för test av filtets förmåga att minska brus. Observera att initaltillståndet är stillastående.

6.2.4 Test av undvikande av hinder

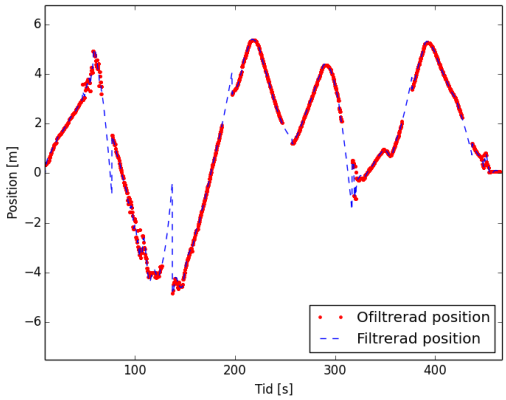
Det första testet för att undvika hinder innebar att roboten skulle köra inom ett visst område med en extra polygon inom området som roboten skulle undvika. Resultatet av testet visade att funktionaliteten inte fungerar som förväntat. Enligt det som syntes på webbapplikationens karta, alltså det som också robotgräsklipparen tror att dess position är, reagerade robotgräsklipparen när den nådde kanten av ett fördefinierat hinder, på så vis att den började rotera mot en ny riktning. Problemet var att riktningen den valde ibland gick rakt igenom hindret, vilket ledde till att robotgräsklipparen ignorerade hindret och körde rakt igenom det. Det andra testet för att undvika hinder innebar att robotgräsklipparen skulle, med hjälp av ultraljudssensorerna, undvika oförutspådda objekt som var i vägen för robotgräsklipparens bana. Resultatet av testet visade att robotgräsklipparen alla gånger uppfattade och lyckades undvika hinder som placerades framför den, vilket pekar på att sensorerna fungerar som de ska, samt att logiken som instruerar robotgräsklipparen att svänga också fungerar.



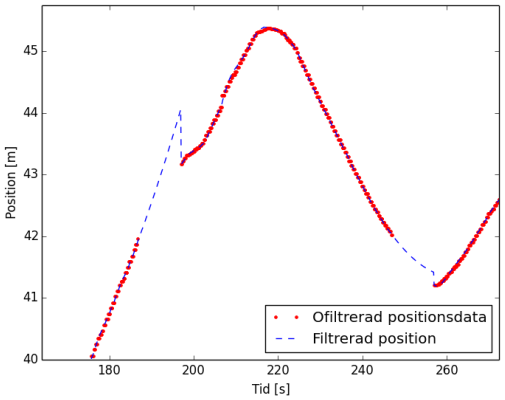
Figur 23: Ofiltrerad accelerationsdata från IMU-modulen i modulens x-riktning.



(a) Positionen i x- och y-led.

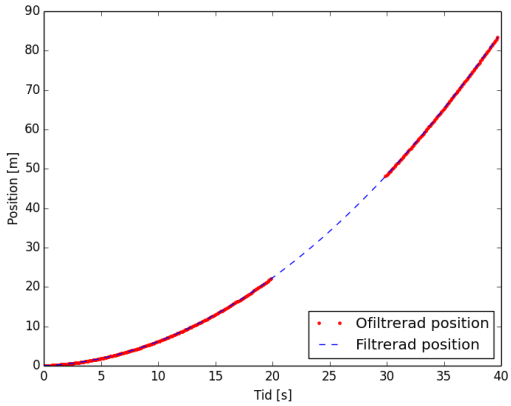


(b) Positionen i x-led över tiden.



(c) Positionen i x-led över tiden.

Figur 24: Plottar över det filtrerade och ofiltrerade positionerna vid förlust av GPS-signal.



Figur 25: Plott från test vid av saknad av GPS-signal på simulera testdata. Observera att samma data som i figur 22a och 22b har använts.

7 Diskussion

Det här kapitlet innerhåller gruppens reflektioner kring det resultat som slutligen uppnåddes, hur arbetet som ledde till resultatet utfördes och hur fortsatt utveckling av projektet skulle kunna se ut. Slutligen en diskussion kring etiska dilemman som kan uppstå om den produkt som utvecklats skulle bli kommersiell och användas i stor skala.

7.1 Diskussion av testning

Att testa det utvecklade navigationssystemet var en stor utmaning. Vid första testet gick det att åstadkomma en ganska bra initial precision på GPS-signalen, vilket är otroligt viktigt för det utvecklade systemet, då RTK-tekniken går ut på att förändringen från initialpositionen i förhållande till basstationen är grunden till var navigationssystemet uppfattar sig ha sin position. Däremot visade sig det att GPS-signalen var inkonsekvent; samma inställningar kunde ge olika bra precisioner även under liknande omständigheter vilket medförde svårigheter att utföra de olika testerna. Det ledde till att när testandet skulle fortsättas på andra dagar var det svårt att få uppnå en lika bra initialposition, vilket ledde till att testandet var svårt att utföra konsekvent, något som också pekar på att navigationssystemet inte alls är så robust som målet var.

I det första testet fanns ett litet fel i positionen, då GPS-positionen var lite skiftad åt ett håll, men skiftningen positionen var konsekvent. Det innebär att positionen under stora delar av testens gång var ungefär lika fel som initialpositionen var, vilket tyder på att om initialpositionen hade varit bra, till exempel genom kalibrering under en längre tid eller att en fast position hårdkodas in, hade systemet med stor sannolikhet fungerat mycket bättre vad det gäller att hålla sig inom ett markerat område.

Skiftningen som uppstod under det första testet kan bero på olika saker. Det första skulle kunna bero på är att kartan som Google Maps visar eventuellt inte helt stämmer överens med hur verkligheten ser ut, då den är lite förvrängd (byggnader ses inte alltid rakt ovanifrån, utan en del av väggen kan ses). Det andra som det skulle kunna bero på är att när navigationssystemet avgör var robotgräsklipparen befinner sig baserar den det till stor del på var GPS-basstationen är belägen. Om basstationens uppfattade position är skiftad en meter från var den i verkligheten skulle det kunna göra att robotgräsklipparens position får samma fel. Positionen på basstationen var något som enkelt skulle tas fram genom att låta den kalibreras under en längre tid, vilket var något som utfördes. Däremot gick det inte att få fram en

bra precision på basstationen på grund av de problem som beskrevs i avsnitt 4.3.3.

Det som talar emot att felet skulle vara att en skiftning uppstår på grund av att basstationen är felkalibrerad, att initialpositionen är fel eller att Google Maps visar en karta som inte stämmer helt med verkligheten, är att i test 2 och test 3 verkade skiftningen förändras under testets gång. Det var dock svårt att uppskatta hur mycket skiftningen förändrades, då det fick göras genom observation. Det är därför svårt att fastställa huruvida skiftningen faktiskt förändrades eller om det bara var ett tillfälligt hopp i den av robotgräsklipparen uppfattade positionen. Skulle det bara varit ett hopp så skulle de anledningar som angavs i föregående stycke kunna stämma, men om det var att skiftningen faktiskt förändrades så är de anledningarna inte troliga.

Det var svårt att kontrollera att den faktiskt i verkligheten höll sig inom det område som markerades ut, men detta försökte uppnås genom att markera ut området på ett sådant sätt att det följde tydliga objekt, exempelvis tak, kanter av trappor, en kant där en ränna bytte färg och kanten av ett staket. De markerades också ut med krita för att enklare kunna identifiera hur robotgräsklipparen förhöll sig till det markerade området, men det var svårt att få det exakt.

Ett problem som uppstod i alla test var att robotgräsklipparen vid ett fåtal gånger hann lämna området, även enligt webbapplikationen, innan den hann vända. Det berodde troligvis på att navigationssystemet inte fick positionsdata ofta nog, vilket orsakat att den vid en mätning var precis innanför och sedan hann åka utanför innan positionen uppdaterades. Så fort som positionen uppdaterades vände den som förväntat, vilket tyder på att navigationslogiken fungerar som den ska. Det här problemet hade kunnat lösas genom att öka frekvensen av uppdateringar av position, vilket hade lett till att den inte hinner åka ut innan den vänder.

Det uppstod även en del problem under testningen, exempelvis fastnade robotgräsklipparen sporadiskt där det låg småsten och pinnar samt var det ena hjulet på robotgräsklipparen en aning skevt. Det ledde till att roboten förflyttades lite åt vänster hela tiden, något den behövde korrigera med jämna mellanrum genom att svänga åt höger eftersom robotgräsklipparen alltid försöker hålla sig till en riktning innan en ny riktning ges.

Under den period av projektets gång då navigationssystemet testades var vädret väldigt ostadigt. Då robotgräsklipparens komponenter, framförallt kretskort av olika slag, är vattenkänsliga gjorde vädrets osäkerhet det svårt att utföra tester under en längre tid. Det ledde till att möjligheterna att testa begränsades kraftigt. Testningen belyste problem som inte hittades innan. Då vädret fördröjde testen innebar det att dessa problem upptäcktes för sent för att hinna åtgärdas, vilket innebar att projektets slutresultat inte uppnådde den nivå som det hade kunnat göra om testen hade kunnat göras tidigare.

7.1.1 Test 1 - Endast GPS

Resultatet av det första testet var på ett sätt väldigt positivt, och på ett annat sätt väldigt negativt. Det var positivt i den mån att gräsklipparen enligt användargränssnittet verkligen höll sig inom området, förutom några få tillfällen då den hann någon decimeter utanför innan den vände in igen. Det var negativt i den mån att hur det såg ut i användargränssnittet inte helt stämde överens med hur det såg ut i verkligheten.

När robotgräsklipparen sedan körde följde den i verkligheten, dock inte de kanter som var utmarkerade på kartan i webbapplikationen, samt på marken med krita. Det som upptäcktes var dock att den var konsekvent med hur långt från kanterna den körde, till exempel körde den lika långt utanför den ena kortsidan som den hade kvar till den andra kortsidan när den vände.

7.1.2 Test 2 - Med filter

Precis som i det första testet höll sig robotgräsklipparen inom området om bara webbapplikationens visualisering tas i beaktning. Problemet var att det, precis som i test 1, inte stämde överens med verkligheten, utan var skiftat. Till skillnad från i test 1 höll sig dock inte skiftningen konstant under testets gång, utan förändrades flera gånger.

Det erhållna resultatet från testning med navigation med Kalmanfiltret påvisar att filtet hade en minimal inverkan på positioneringen av robotgräsklipparen. Detta berode med stor sannolikhet på att GPS-signalen inte var särskilt brusig i och med att test av filtet på den brusiga simulerade testdatan gav önskat resultat.

7.1.3 Test 3 - Med filter och temporär GPS-förlust

Det erhållna resultatet vid testning av navigation med Kalmanfiltret och förlust av GPS-signal visar att robotgräsklipparen klarar att navigera utan GPS-signal men att noggrannheten på positioneringen är varierande. Precisionen påverkas troligtvis av robotens hackiga rörelse, vilken blir svår att förutsäga, samt de stora variationerna i accelerationsdata som postionbestämmningen baseras på då GPS-signalen förloras. I och med att accelerationsdatan varierar så pass mycket och inte heller följer något tydligt mönster blir för det första accelerationen svårfiltrerad, huvudsakligen för att det inte finns någon referens att kalibrera filtret utifrån, och för det andra får den filtrerade accelerationsdatan en stor kovarians. Det resulterar i en sämre förmåga att förutspå positionen vilket också framkommer vid jämförelse av testresultatet i figur 24b och simuleringen i 25 då den förutspådda positionen vid förlust av GPS-signal motsvarar den exakta positionen. På de 10 sekunder som GPS-mottagningen stängdes av uppkom maximala fel uppmot 2 meter i x-led men på halva tidsintervallet ligger dock de maximala felen under 0,5 meter. Värt att observera är dock att enligt figur 24a ligger filtrerade positioner vid GPS-förlust utanför området vilket borde stoppat gräsklipparen. Vid felsökning upptäcktes att webbapplikationen uppfattade signalförlusten som ett stop vilket förklarar felet och gör den uppmätta tiden missvisande.

Förbättring av positionsbestämningen vid förlust av GPS-signalen skulle kunna uppnås genom att förbättra robotsgräsklipparens rörelsemönster så det blir mer förutsägbart. Det skulle till exempel kunna göras genom att byta hjul, då de hjul som användes nu var hårda och icke-stötdämpande, för att få en bättre rörelse, samt genom att öka robotsgräsklipparens hastighet så den inte fastnar. En ökad hastighet skulle även medföra större accelerationer vilket kanske skulle ge accelerationsdata av större storleksordning, vilket hade kunna separera den egentliga accelerationen gentemot bruset bättre. Det hade även varit intressant att prova en annan IMU-modul för jämföra accelerationsdatan.

Mest problematiskt är förlust av GPS-signal i vändningarna då filtret inte kan förutspå vändningen utifrån föregående position i och med att filtret inte vet var områdets gräns är definierat. I och med att robotgräskilpparen inte vänder så fort har vändningen inte så stor inverkan på testningen vid förlust av GPS-signal i 10 sekunder men i verkligheten då objekt som orsakar signalförlust, såsom träd, inte är bundna till något tidsintervall kan det bli ett problem.

Huruvida total förlust av GPS-signal är en rimlig simulering för att representera träd och dåliga väderomständigheter kan ifrågasättas. Inom det tänkta användningsområdet är troligtvis störningar i form av brus från träd och liknande mer troligt än en total förlust. I och med tidsbrist delvis på grund av dåligt väder testades inte det.

7.1.4 Test av undvikande av hinder

Det första testet av undvikande av hinder var att undvika ett fördefinierat hinder, vilket robotgräsklipparen inte klarade av. Logiken för att undvika fördefinierade hinder fungerade i simulatorn men inte i verkligheten. Det är troligtvis på grund av att avståndet som de virtuella strålarna bestämmer

definieras som magnituden av en vektor bestående av latitud och longitud. Om magnituden är mindre eller lika med 0,000003 enheter anser programmet att det är en kollision, och det är möjligt att avståndet är för litet för biblioteket som används för att skicka strålarna att hantera. På grund av tidsbrist och prioritering valdes det inte att försöka åtgärdas.

Den andra funktionen som testades var att undvika hinder med hjälp av ultraljudssensorerna som var fastmonterade på robotgräsklipparen. Då funktionen var väldigt konkret var det enkelt att testa huruvida den fungerade. Resultaten som uppnåddes var entydiga, alla observationer visade på att funktionen fungerade som den ska.

7.2 Gruppens upplevelser

Projektet som helhet var en utmaning för gruppen. Då gruppmedlemmarna hade bakgrund i olika program fanns det stora skillnader i vad de hade kunskap om, och det märktes tidigt att det var svårt att strukturera upp uppgifter som passade allas kompetenser. I efterhand kan det konstateras att det hade varit fördelaktigt att utforma projektet mer brett och verkligen försökt få in arbetsområden som passar alla i gruppen.

På samma spår kan det också nämnas att det var svårt att uppskatta hur mycket arbete som skulle krävas för att få all hårdvara att fungera. Det ledde till att de gruppmedlemmar som hade tidigare kunskap om hårdvaran fick lägga ner betydligt mer tid än andra, då det var mycket felsökande och installation som krävdes för att få robotgräsklipparens basfunktionalitet så som att få motorerna att fungera. Det hade underlättat om fler gruppmedlemmar tidigt insett att det skulle bli mycket mer arbete med hårdvaran än att bara installera det, och därmed kunnat sätt sig in i det mer för att kunna bidra i felsökandet som varit en stor del av arbetet i slutskedet av arbetet.

7.2.1 Montering av basstation

Basstationens montering var något som borde ha inletts tidigare i projektet eftersom det var först efter monteringen felsökning av GPS möjliggjordes. Det krävdes dock väldigt mycket administrativt arbete med att få tillstånd och anordna montering, vilket gjorde att det drog ut på tiden.

När basstationen skulle installeras krävdes det mycket administrativt arbete. Då basstationen skulle monteras på ett tak högt upp innebar det att många människor behövde kontaktas och ge tillåtelse, rådgivning och hjälp av montering. Det första som utfördes var att kontakta två av skolans vaktmästare. De kontaktade i sin tur Chalmers säkerhetsamordnare, vars tillåtelse behövdes för att få vistas på taket, samt den lokalansvarige från Akademiska Hus. Möten med de sistnämnda två personerna gav en klarhet i frågorna som fanns och till slut bestämdes det att basstationen skulle få monteras utanför ett personalrum i EDIT-huset. Det sista som krävdes var en internetanslutning, vilket ordnades genom kontakt med Chalmers IT-support.

7.3 Vidareutveckling i framtiden

Projektet skulle kunna utvecklas vidare på flera olika sätt. De olika förslagen kan sammanfattas i att robotgräsklipparen skulle kunna förbättras genom att:

- implementera en ruttplanerare som hittar den bästa rutten för att klippa gräset implementeras.
- implementera redskap som behövs för att faktiskt klippa gräs.
- implementera en algoritm som kartlägger GPS-täckningen i området för att kunna underlätta navigation.

- uppfinna ett sätt som eliminerar behovet av att ha en basstation för att ha bra precision
- utveckla systemets användning av Kalmanfiltret genom att skicka in fler värden i filtret vilket kan öka precision och robusthet.

En tydlig förbättring som skulle kunna utvecklas är att implementera logik som låter gräsklipparen beräkna den optimala rutten för att klippa allt gräs inom området på så kort sträcka och tid som möjligt. Det här skulle spara på tid och el, samt minska slitaget, vilket skulle innebära att gräsklipparen förblir användbar under en längre period. Det finns forskning på hur logiken som det skulle kräva kan implementeras, vilket beskrivs i en rapport från 2017 [39].

En annan möjlighet att utveckla det som uppnåtts under det här projektet är att implementera hårdvara som krävs för att faktiskt klippa gräs. Den uppgiften skulle kunna ses som ett möjligt kandidatarbete för mer hårdvarufokuserade utbildningar så som *Maskinteknik*, men även för utbildningar med designinriktning så som *Teknisk Design*.

Ännu ett område som kan utvecklas är navigationen. Det som hade kunnat implementeras är att kontinuerligt spara positioner där GPS-signalen är dålig (av exempelvis träd). På så sätt kan då robotgräsklipparen vara medveten om ungefär hur lång tid det kommer ta innan roboten får en stabil GPS-signal och därmed justera sin rutt efter det, då INS:ens felvärden divergerar med tiden.

Dessutom det finns utrymme för förbättring av GPS-användningen. Ett exempel skulle vara att utveckla ett system som är mer flexibelt genom att ta bort behovet av basstation, eller att utveckla en mer portabel och lättanvänd basstation som användare själva skulle kunna montera och flytta runt vid behov, för att kunna dela på användandet av gräsklipparen med andra. Det finns även möjlighet till att använda en tredjepart som kan ge tillgång till basstationer, exempelvis SWEPOS [40].

Slutligen utvecklingsområde är att öka antalet sensorer som används som inmatning i Kalmanfiltret. På Gy88-kortet är även en magnetometer fastmonterad vilken hade kunnat ge en referenspunkt åt norr och därmed öka robustheten i positioneringen genom att tillåta en extra jämförelse i vinkeln mellan magnetometerns värde och värdet i vinkeln från IMU-modulen. Däremot hade magnetometern behövts isoleras eller kalibreras noggrant på grund av all närliggande elektronik som skapar magnetfält som kan interferera med magnetometerns position enligt jordens magnetfält.

7.4 Etiska och samhälleliga aspekter

De etiska och samhälleliga aspekter som behövs fundera kring vad det gäller det här projektet kan delas upp i fyra tydliga delar, vilka är: **integritet**, **säkerhet**, **ekonomi** och **miljö**. Det här avsnittet utgår från att den färdiga produkten faktiskt hade fungerat som tänkt.

När det kommer till integritet handlar eventuella problem främst om hantering av GPS-data. Det finns ett ansvar hos utvecklare och ansvariga att se till att den data som kan avslöja information om den potentiella kunden och dess gräsklippare inte hamnar i fel händer. Ett exempel på problem som skulle kunna uppstå är att en illvillig aktör lätt skulle kunna lokalisera gräsklipparen för att stjäla den eller sabotera den.

Vad det gäller säkerhet ligger faran i eventuella fel i gräsklipparens programvara som skulle kunna leda till att gräsklipparen åker till områden där den ej ska vara, vilket i sin tur skulle kunna leda till att människor eller djur kommer till skada. Ytterligare ett exempel på en säkerhetsrisk är att det finns en risk att en illvillig aktör olovligt skulle kunna ta sig in i och kontrollera navigationssystemet för att få gräsklipparen att åka till positioner där den kan orsaka stor skada för människor, till exempel om den åker ut på en väg och orsakar en trafikolycka.

En ekonomiska aspekt som projektet berör är att människor som arbetar med parkskötsel riskerar att förlora sina jobb om ett navigationssystemet som skapats utvecklas och blir bra nog för att användas på stor skala. Det då en robotgräsklippare lätt kan flyttas från park till park och klippa gräset, något som

inte är möjligt om gräsklipparen kräver en avgränsningskabel som måste grävas ner. Den här aspekten har dock en annan sida, vilken är att kommuner sparar in pengarna som hade gått till att betala lönen åt parkskötare och dylikt.

Om navigationssystemet utvecklas vidare skulle det kunna innebära att en gräsklippare smidigt och enkelt skulle kunna användas på flera olika gräsmattor, exempelvis parker som beskrevs i föregående syfte. Det skulle kunna ha goda effekter för miljön, eftersom antalet gräsklippare som produceras skulle kunna minskas om grannar skulle kunna dela på en gemensam gräsklippare. Det här kan dessutom ses som en ekonomisk aspekt, då det skulle innebära att det skulle vara billigare för människor att ha tillgång till en robotgräsklippare om kostnaden delas mellan alla brukare. Ett alternativ till att dela på en gräsklippare med grannskapet är företag som skulle kunna syssla med uthyrning av gräsklippare, vilket även det skulle minska kostnad och miljöpåverkan.

7.4.1 Åtgärder

Hur åtgärdas då eventuella problem? Många av problemen är på en sådan nivå att de inte var aktuella att arbeta med inom det här projektet, men de är ändå värda att nämna i diskussionssyfte.

Integritetsproblemet är inte något som behövdes ta särskild hänsyn till då programvaran och hårdvaran ej distribueras till verkliga användare, och därmed finns det ingen platsdata som måste skyddas.

Säkerhetsproblemet är något som är viktigare att ta hänsyn till. Sättet som det hanterades är att robotgräsklipparen inte hade anordningar som skulle kunna användas för gräsklippning, till exempel knivar eller vassa blad. Det här minskade risken som roboten skulle medföra för människor och djur till att bli så gott som negligerbar. Utöver det här fanns det ett ansvar att se till att systemet var säkert och inte var sårbart för illvilliga aktörer. Det hade kunnat dras till sin spets, till exempel genom att se till att all data som skickades fram och tillbaka var krypterad. Det här ansågs dock inte utgöra en stor risk, då det ansågs vara orimligt att någon illvillig aktör känt till och velat sabotera projektet. Det här baserades på att det är ett icke-kommersiellt arbete som dessutom kommer att offentliggöras, vilket gjorde det onödigt för en eventuell illvillig aktör att försöka erhålla information. Med den motiveringen lades det därför inte inget fokus på att utveckla ett försvar mot cyberattacker, men i ett projekt som syftar till att utveckla en användarprodukt skulle det behöva hanteras och motverkas.

De ekonomiska problem som eventuell kan uppstå vid ett scenario där denna produkt används i parkskötsel anses vara små relativt till de fördelar som kunde nåtts med produkten. Vad det gäller miljö förutspåddes inga negativa konsekvenser av användandet av produkten som utvecklats.

8 Slutsats

Projektets mål var att robotgräsklipparen skulle kunna röra sig inom ett område i 15 minuter, och att den inte skulle åka utanför även om den förlorade GPS-mottagning i upp till 10 sekunder. Robotgräsklipparen lyckades inte i något av testerna hålla sig inom det utmarkerade området i verkligheten, dock så höll sig robotgräsklipparen inom området enligt webbapplikationens visualisering med Google Maps. Det var på grund av att GPS-positionen var skiftad åt ett håll, vilket innebar att navigationssystemet uppfattade robotgräsklipparens position som innanför området. Däremot fungerade navigationslogiken baserad på de positionsdata som den fick, men eftersom positionsdata inte var precisa nog kunde inte heller hela navigationssystemet fungera enligt målsättningen. Därav uppnåddes inte målet för navigationssystemet. I och med att precisionen på GPS-systemet inte var precis och konsekvent var det också svårt att dra en slutsats kring huruvida INS-systemet var till hjälp vid förlust av GPS-signaler.

Webbapplikationen som utvecklades under projektet nådde alla de mål som satts i början av projektet. Den användes mycket under testningen av navigationssystemet och inga problem upptäcktes under

rigorös användning.

Slutsatsen av det här kandidatarbetet är att ett navigationssystem som i grunden baseras på GPS-navigation, med ett utmarkerat området som skapas i Google Maps, inte är robust i den mening att gräsklipparen inte åker utanför det angivna området. Dessutom är det svårt att dra en slutats angående potentialen av att använda RTK-teknik för robotgräsklippare då flera SBAS-satelliter inte var tillgängliga under testningens gång vilket försämrade precisionen av RTK-systemet.

9 Referenser

[1] J.M. Derander, P. Andersson, E. Wennerberg, A. Nitsche, E. Moen och F. Labe, "Smart Robot Lawn Mower," kandidatarbete, Avdelningen för Datavetenskap och Datateknik, Chalmers tekniska högskola, Göteborg, Sverige, 2018.[Online].

[2] Self-propelled random motion lawnmower, S. L. Bellinger (1972, 17 oktober). *US3698523A*[Online]. Tillgänglig: `https://patents.google.com/patent/US3698523#patentCitations`, hämtad: 2019-02-11.

[3] Elsäkerhetsverket, "Robotgräsklippare," 2015.[Online]. Tillgänglig: `https://www.elsakerhetsverket.se/privatpersoner/sakra-elprodukter/produkter/robotgrasklippare/`, hämtad: 2019-02-04.

[4] SLU, *Nya beviljade innovationsprojek*, apr. 2018.[Online]. Tillgänglig: `https://www.slu.se/ew-nyheter/2018/4/nya-beviljade-innovationsprojekt/`, hämtad: 2019-02-04.

[5] M. Baerveldt, P. Helmersson, S. Ingemarsson, V. Ohm, N. Uusitalo och A. Wessman, "Satellitnavigerad robotgräsklippare med RTK," kandidatarbete, Institutionen för Produkt- och Produktionsutveckling, Produktionssystem, Chalmers tekniska högskola, Göteborg, Sverige, 2016.[Online].

[6] K. M. Ng, J. Johari, S. A. C. Abdullah, A. Ahmad och B. N. Laja, "Online complete coverage path planning using two-way proximity search," i *2018 18th International Conference on Control, Automation and Systems (ICCAS): Performance Evaluation of the RTK-GNSS Navigating under Different Landscape*, PyeongChang, GangWon, Korea, 2018, ss. 1424-1428.[Online].

[7] A. Noureldin, *Fundamentals of inertial navigation, satellite-based positioning and their integration,* Heidelberg : Springer, 2013.[Online].

[8] K. An, S. Xie och Y. Ouyang, "Reliable Sensor Location for Object Positioning and Surveillance via Trilateration," *Transportation Research Procedia*, vol. 23, ss. 228-245, jul. 2017.[Online].

[9] Rossi, "Trilateration3," 2006.[Elektronisk bild]. Tillgänglig: `https://upload.wikimedia.org/wikipedia/commons/5/50/Trilateration3.png`, hämtad: 2019-03-11.

[10] Lantmäteriet, "RTK."[Online]. Tillgänglig: `https://www.lantmateriet.se/sv/Kartor-och-geografisk-information/GPS-och-geodetisk-matning/GPS-och-satellitpositionering/Metoder-for-GNSS-matning/RTK/`, hämtad: 2019-03-25.

[11] K. M. Fauske, "Example: Inertial navigation system," 2016.[Elektronisk bild]. Tillgänglig: `http://www.texample.net/tikz/examples/inertial-navigation-system/`, hämtad: 2019-02-20.

[12] F. Fasching, "Dr. Franz Fasching Information - Telecommunications - Technology Products - Services - Solutions," 2016.[Online]. Tillgänglig: `https://drfasching.com/products/gnss/raspignss.html`, hämtad: 2019-03-26.

[13] Arduino, "Arduinos products."[Online]. Tillgänglig: `https://www.arduino.cc/en/Main/Products`, hämtad: 2019-04-10.

[14] Arduino, "Arduino Uno"[Online] Tillgänglig: `https://cdn-learn.adafruit.com/assets/assets/000/003/199/original/learn_arduino_arduinounotop.jpg?1396793561`, hämtad: 2019-05-16.

[15] Electrokit, "Avståndsmätare ultraljud HC-SR04 2 – 400cm."[Online]. Tillgänglig: `https://www.electrokit.com`, hämtad: 2019-04-30.

[16] Adafruit, "HC-SR04 Ultrasonic Sonar Distance Sensor + 2 x 10K resistors."[Elektroniskbild]. Tillgänglig: `https://search.creativecommons.org/photos/47163df6-0962-4deb-aeed-394dfdf17167`, hämtad: 2019-05-02.

[17] Raspberry Pi Foundation, "What is a Raspberry Pi?"[Online]. Tillgänglig: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/, hämtad: 2019-02-18.

[18] NVS Technologies AG, "NV08C-CSM."[Online]. Tillgänglig: http://www.nvs-gnss.com/products/receivers/item/2-nv08c-csm.html, hämtad: 2019-05-02.

[19] RTKLIB, "RTKLIB: An Open Source Program Package for GNSS Positioning."[Online]. Tillgänglig: http://www.rtklib.com/, hämtad: 2019-02-11.

[20] Tallysman, "TW2410/TW2412 Magnetic Mount GPS/GLONASS Antenna."[Online]. Tillgänglig: http://www.tallysman.com/index.php/gnss/products/antennas-gpsglonass/tw2410-tw2412/, hämtad: 2019-03-25.

[21] M. S. Grewal och A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*, vol. 3, Wiley-IEEE Press, 2008.[Online], hämtad: 2019-03-25.

[22] M. Zarchan, *Fundamentals of Kalman Filtering: A Practical Approach, Fourth Edition*, AIAA, 2015.[Online].

[23] J. N. Gross, Y. Gu, M. B. Rhudy, S. Gururajan och M. R. Napolitano, "Flight-Test Evaluation of Sensor Fusion Algorithms for Attitude Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, nr.3, ss. 2128-2139, jul. 2012.[Online]. Tillgänglig: https://www.researchgate.net/publication/260657990_Flight-Test_Evaluation_of_Sensor_Fusion_Algorithms_for_Attitude_Estimation, hämtad: 2019-05-12.

[24] A. Chiu, T. Jones och C. E. van Daalen, "A comparison of linearisation and the unscented transform for computer vision applications," i *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, Stellenbosch, Sydafrika, 2016.[Online].

[25] F. Gustafsson and G. Hendeby, "Some Relations Between Extended and Unscented Kalman Filters," *IEEE Transactions on Signal Processing*, vol. 60, nr.2, ss. 545-555, feb. 2012.[Online].

[26] J. Wendel, J. Metzger, R. Moenikes, A. Maier och G. F. Trommer, "A Performance Comparison of Tightly Coupled GPS/INS Navigation Systems based on Extended and Sigma Point Kalman Filters," *Navigation*, vol. 53, nr.1, ss. 21-31, apr. 2006.[Online]. Tillgänglig: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2161-4296.2006.tb00368.x, hämtad: 2019-05-12.

[27] D. Duckworth, "Pykalman," 2019.[Online]. Tillgänglig: https://pykalman.github.io/, hämtad: 2019-05-01.

[28] *NEMO inertial module: 3D accelerometer and 3D gyroscope with digital output for industrial applications*, STMicroelectronics, 2018.[Online]. Tillgänglig: https://www.st.com/resource/en/datasheet/ism330dlc.pdf, hämtad: 2019-04-30.

[29] *MPU-6000 and MPU-6050 Product Specification Revision 3.4*, InvenSens, 2013.[Online]. Tillgänglig: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf, hämtad: 2019-04-28.

[30] M.NU, "IMU 10DOF MPU6050+HMC5883L+BMP085," 2018.[Online]. Tillgänglig: https://www.m.nu/sensorer-matinstrument/imu-10dof-mpu6050-hmc5883l-bmp085, hämtad: 2019-04-28.

[31] *3-Axis Digital Compass IC HMC5883L*, Honeywell, 2013.[Online]. Tillgänglig: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf, hämtad: 2019-04-28.

[32] *BMP085 Digital pressure sensor: Data Sheet*, BOSCH, 2009.[Online]. Tillgänglig: https://www.sparkfun.com/datasheets/Components/General/BST-BMP085-DS000-05.pdf, hämtad: 2019-04-28.

[33] Python Packaging Authority, "Jep - Java Embedded Python," 2019.[Online]. Tillgänglig: https:

//spring.io/, hämtad: 2019-05-02.

[34] *User Guide: Original Operating Instructions*, Robomow.[Online]. Tillgänglig: https://www.robomow.com/wp-content/uploads/2018/11/DOC9043C_EN-Robomow-RX_User_guide_V02_X-EN_2019A_Web-2.pdf, hämtad: 2019-05-01.

[35] Facebook OpenSource, "React - A JavaScript library for building interfaces," 2019.[Online]. Tillgänglig: https://reactjs.org/, hämtad: 2019-05-06.

[36] T. Koppers, "webpack," 2019.[Online]. Tillgänglig: https://webpack.js.org/, hämtad: 2019-05-06.

[37] Google Maps Platform, "Google Maps Platform Documentation," 2019.[Online]. Tillgänglig: https://developers.google.com/maps/documentation/, hämtad: 2019-05-06.

[38] Pivotal Software, "Spring," 2019.[Online]. Tillgänglig: https://spring.io/, hämtad: 2019-05-03.

[39] Adafruit, "BCM2835"[Online]. Tillgänglig: https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/README.md, hämtad: 2019-05-16

[40] A. Khan, I. Noreen, H. Ryu, m.fl., "Online complete coverage path planning using two-way proximity search," *Intelligent Service Robotics*, vol. 10, nr. 3, ss. 229-240, mar. 2017.[Online]. Tillgänglig: https://doi.org/10.1007/s11370-017-0223-z, hämtad: 2019-02-05.

[41] Lantmäteriet, "SWEPOS är Lantmäteriets stödsystem för satellitpositionering i Sverige."[Online]. Tillgänglig: https://swepos.lantmateriet.se, hämtad: 2019-05-03.