CHALMERS

Robust och noggrant positioneringssystem baserat på avståndsmätningar mellan mobila noder

– Kandidatarbete inom Data- och informationsteknik

Alexander Altby
Tobias Boström
Timur Sibgatullin
Karl Stjärne

Institutionen för Data- och informationsteknik

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2012
Kandidatarbete nr 2012:01

Förord

Kandidatarbetet utfördes vid Institutionen för Data- och informationsteknik, Chalmers Tekniska Högskola, våren 2012. Vi vill tacka vår handledare Elad Schiller för hans tid och hängivenhet till projektet, Benjamin Vedder och Chalmers robotförening för allt jobb med hårdvaruplattformen, Henk Wymeersch och Gabriel Garcia för deras hjälp och åsikter angående Kalmanfiltret, kandidatgrupp DATX02-12-61 för deras synpunkter på projektet och slutrapporten samt Mohamed Mustafa och Mitra Pahlavan för deras hjälp med MICAz-modulerna.

Sammanfattning

Det existerar ett flertal applikationer där kännedom om användarens position används, dessa tillämpningar är i många fall baserade på GPS-systemet. Problem kan dock uppstå då GPS-signalen inte alltid är tillgänglig, exempel på detta kan vara i stadsmiljöer med höga byggnader eller inomhus i parkeringshus eller lagerbyggnader.

Eftersom GPS inte lämpar sig i realtidsapplikationer där krav sätts på noggrannhet i avseende på positionsuppskattning, syftar detta arbete till att implementera ett positioneringssystem baserat på inbördes avståndsmätningar mellan noder.

Litteraturen inom området antar att ett protokoll för *medium access control* (MAC) existerar och därmed att inga datakollisioner uppstår, på så sätt kan alla noder i systemet utföra avståndsmätningar samtidigt. Dessa implementationer använder sig oftast av ett MAC-lager som har en oförutsägbar fördröjning. Oförutsägbar fördröjning är ett stort problem för positioneringssystem som ska fungera i realtid. Kandidatarbetet utvecklade två MAC-algoritmer anpassade för olika hårdvarukonfigurationer, båda är självstabiliserande, självkonfigurerande samt självorganiserande.

De testresultat som presenteras visar att den implementerade algoritmen endast beror på antalet mobila noder i systemet och därmed är helt deterministisk. Vidare har även ett filter som bestämmer den mobila nodens position, genom att kombinera sensordata med information från avståndsmätningarna, skapats.

Dessa resultat kommer att underlätta för framtida utveckling inom realtids styrsystem som exempelvis obemannade fordon eller räddningsenheter som behöver veta exakt position av dess personal.

Abstract

Numerous applications require the knowledge of the user location. Existing implementations assume GPS accessibility, however GPS is not always available due to loss-of-signal failures. Some examples of this are in urban canyons and indoor facilities such as parking lots, warehouses and shopping malls.

Because GPS is not suitable in real-time applications where accuracy is required, in according to positioning estimation, the purpose of this project is to implement a positioning system based on the use of peer to peer ranging.

The literature in this area assumes the existence of a medium access control protocol in which collisions cannot occur, and that ranging can be performed instantaneously. These implementations mostly consider a non-real time probabilistic MAC that have an unbounded random delay. This doesn't go very well with localization. When in the context of localization, unbounded random delay presents a great obstacle for the implementation of real time localization. This project presents a self-stabilizing, self-configuring and self-organizing MAC algorithm.

This project presents experiments that prove that the implemented algorithm has a delay that only depends on the number of mobile nodes, and is fully deterministic. A filter that determines the mobile nodes position by weighing the sensor data to the ranging information has also been implemented.

This project will contribute to the future development of real time control systems such as unmanned vehicles and rescue units in need of knowing the exact position of its personnel.

Innehållsförteckning

Figur- och tabellförteckning

Ordlista

Ad hoc All kommunikation sker direkt mellan noder utan att passera en central accesspunkt.

Ankare Statisk nod med fördefinierad position. Används som referensnod vid avståndsmätningar.

CRE Coarse Range Estimate. En metod för att uppskatta avståndet till en nod baserat på dess signalstyrka.

Firmware Mjukvara förprogrammerad i hårdvara.

GPS Global Positioning System. Satellitbaserat positioneringssystem där avståndsmätningarna görs med hjälp av klocksynkronisering.

Gulliver Plattform under utveckling för testning, utveckling och demonstration med hjälp av miniatyrfordon.

LIDAR LIght Detection And Ranging. Metod för att mäta avstånd med hjälp av ljus.

MAC Media Access Control. Ett sätt att tillåta enheter i ett nätverk att få access till mediet.

MICAz Programmerbar mikrokontroller med trådlöst gränssnitt som möjliggör kommunikation med andra MICAz-moduler.

PII Pulse Integration Index. Används i RCM för att koda om symboler till pulser.

PRE Precision Range Estimate. Den nuvarande versionen av avståndsmätning baserat på TOF för varje enskild nod. Namnet används för att skilja denna metod från CRE i framtida versioner av RCM:en.

RCM Ranging and Communication Module. UWB Radiomodul som används för att mäta avstånd från en RCM till en annan.

Self-* En gemensam benämning på system med egenskaper som självstabiliserande, självkonfigurerande och självorganiserande.

TDMA Time Division Multiple Access. En metod för schemaläggning genom tidsuppdelning.

TOF Time Of Flight. Tiden det tar för en signal att färdas mellan två noder.

UWB Ultra-wide band. Teknik för att överföra mycket information trådlöst över korta sträckor med en bandbredd på minst 500 MHz.

1 Inledning

I alla tider har människan haft behovet av att veta vart hon befinner sig. Kartor har funnits i flera tusen år, och genom historien har dessa kartor samt tekniker för att navigera och bestämma sin position förbättras. I slutet på 1960-talet sjösattes ett satellitbaserat positioneringssystem, *Global Positioning System* (GPS) [1], som senare skulle komma att revolutionera sättet mänskligheten orienterar sig på. GPS har dock en del stora svagheter, systemet har exempelvis inte bättre noggrannhet än ett tiotals meter. I stadsmiljöer med höga hus förlorar signalen sin styrka och noggrannheten försämras ytterligare. Det finns även fall då signalen helt försvinner, som till exempel i tunnlar eller inuti parkeringshus. Det finns även andra tekniker för att bestämma sin position via radio, med hjälp av signalstyrkan från ett antal basstationer är ett exempel.

De ovan nämnda positioneringssystem utgår från ett kollisionsfritt MAC-lager. Implementationerna innehar dock en icke-deterministisk tidsfördröjning, vilket är opassande i tillämpningar med mobila noder då det inte enbart är positionen som är intressant utan även vid vilken tidpunkt denna är aktuell.

För att uppfylla de brister som GPS och liknande positioneringssystem har, syftar detta arbete till att skapa ett noggrant positioneringssystem som fungerar såväl för inomhus- som utomhusmiljöer. Systemet ställer också krav på samtida positionsbestämmelse av flera noder, därför har även ett MAC-lager med deterministisk fördröjning implementerats.

Det system skapat av kandidatarbetet kommer att implementeras i forskningsprojektet *Co-opnet*[1] samt forskningsprojektet *Gulliver* [2]. *Gulliver* har som syfte att tillhandahålla en smidig plattform för testning, utveckling och demonstration med hjälp av miniatyrfordon. Kandidatgruppen har även blivit kontaktad av *Time Domain*, företaget som utvecklat RCM:erna, då de tänkte implementera en MAC-algoritm mycket lik vår i sin hårdvara. Det diskuterades även fördelar med filter och hur dessa kan förbättra prestandan på RCM:erna.

1.1 Syfte

Kandidatarbetet syftar till att tillhandahålla ett positioneringssystem baserat på avståndsmätningar via radio. Positioneringssystemet ska vara högpresterande i termer av noggrannhet samt ha stöd för ett flertal samtida noder. Syftet är även att skapa ett robust kollisionsfritt MAC-lager med deterministisk fördröjning.

Denna rapports syfte är att utvärdera områden som hur och varför filter används vid positionering samt hur man undviker datakollisioner då flera noder kommunicerar på samma frekvens. Dess syfte är också att utvärdera hur ett MAC-lager som ska vara självstabiliserande, självkonfigurerande och självorganiserande (self-*) ska konstrueras samt hur

[1] https://sites.google.com/site/erccoopnet/

positioneringssystemet är implementerat. Rapporten tjänar även till att ge en överblick av systemet och dess möjliga tillämpningar.

1.2 Problembeskrivning

Nedan listas de problem som måste lösas för att kandidatarbetet ska uppnå sitt syfte.

1.2.1 Kravspecifikation

1. **Noggrann positionering**
   Medelvärdet på osäkerheten för positioneringssystemet bör inte vara större än 100 mm.
2. **Hög uppdateringsfrekvens**
   Avståndsmätningarna ska genomföras minst tre gånger per sekund.
3. **Ad hoc**
   All kommunikation ska ske ad hoc, vilket innebär att kommunikationen sker direkt mellan noderna i systemet utan att passera en central accesspunkt.
4. **Self-***
   Det ska vara möjligt att ansluta eller koppla bort noder från systemet utan att systemet behöver konfigureras om.
5. **Deterministisk fördröjning**
   Fördröjningen på positionering med RCM:erna ska vara deterministisk.

1.2.2 Positionsbestämning

Alla noder i systemet måste kunna bestämma sina positioner. Den vanligaste metoden för att göra detta är med hjälp av triangulering. En lösning på trianguleringsproblemet presenteras i kapitel 2.1. Problemet ligger i att skapa en algoritm som utför detta.

1.2.3 Filtrering

Att endast utföra en positionsbestämning baserat på data från avståndsmätningar ger upphov till en rad problem. Dessa beskrivs i kapitel 2.2. En utmaning var att lösa dessa problem genom att implementera ett filter. Det filter som användes var en specialvariant av ett Kalmanfilter. Kalmanfiltret lämpar sig bra då det gör en uppskattning av systemets nästkommande tillstånd genom att väga systemets nuvarande tillstånd med extern data, i detta fall i form av avståndsmätningarna. [3] Kalmanfiltret kräver kunskaper om normalfördelningar, ickelinjära system, ickelinjära transformer och tillämpad signalbehandling, något medlemmarna i kandidatgruppen saknade kunskap om. Ett problem att lösa var således att inhämta kunskap om Kalmanfiltret och skapa en egen förenklad variant anpassad för den hårdvara som fanns att tillgå.

1.2.4 Förutsägbar fördröjning på avståndsmätningar

För att få en fungerande positioneringssystem för stora system krävs ett *self-** MAC (*Media access control)* som styr radiomodulernas åtkomst till radiofrekvensen.

1.2.5 Kollisionsfria avståndsmätningar

Avståndsmätningarna behöver vara kollisionsfria för att det ska fungera för fler än en nod. Detta löses även med hjälp av MAC-lagret.

1.3 Avgränsningar

Ranging and Communication Modules (RCM:er) kommer att användas för att utföra avståndsmätningarna. RCM:erna mäter *Time Of Flight* (TOF) i pikosekunder tur och retur mellan två noder och kan på så sätt beräkna avståndet, detta beskrivs mer ingående i kapitel 2.2. Anledningen till TOF via radio valdes istället för klocksynkronisering via radio är att det är svårt att uppnå den nivå på synkronisering, maximalt en nanosekunds osäkerhet, som krävs för att systemet ska fungera på önskvärt sätt.

Systemet har även begränsats på så sätt att endast upp till fem aktiva noder tillåts. Detta för att klara av kraven för hög uppdateringsfrekvens eftersom RCM:erna kräver 25 ms per avståndsmätning. Den maximala ytan som systemet kommer att testas på är 30x30 m där bilarna håller en maximal hastighet på två meter per sekund.

1.4 Rapportens upplägg

Kapitel 2 - *Teoretisk bakgrund*
Här presenteras övergripande teori om de komponenter som legat till grund för kandidatarbetet. Tanken är att introducera läsaren i den terminologi som används i rapporten samt även att ge förståelse för de tekniker som använts.

Kapitel 3 - *Metod och genomförande*
Här förklaras vilken utvecklingsmetod och utvecklingsmiljö som använts samt hur hela systemet verifierades.

Kapitel 4 - *Implementering och resultat*
Här presenteras hur systemet är implementerat samt resultatet från den i projektets slutskede genomförda testomgången av systemets samtliga komponenter, alla integrerade i plattformen.

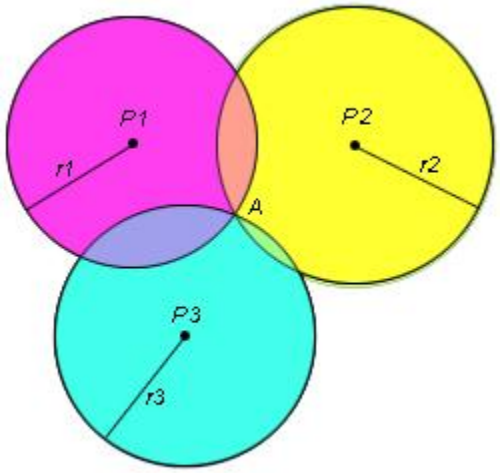
Kapitel 5 - *Slutsats och diskussion*
Här diskuteras resultaten med utgångspunkt från syftesformuleringen i inledningen.

2 Teoretisk bakgrund

I detta kapitel presenteras övergripande teori om de komponenter som legat till grund för kandidatarbetet. Tanken är att introducera läsaren i den terminologi som används i rapporten samt även att ge förståelse för de tekniker som använts.

2.1 Triangulering

Triangulering [4] är en metod som används för att göra positionsskattningar baserat på avståndsmätningar från givna referenspunkter. Antalet referenspunkter måste vara en fler än det antal dimensioner man önskar utföra trianguleringen i för att resultatet skall vara entydigt. I Figur 1 visas ett exempel på triangulering i två dimensioner, radien på de olika cirklarna (benämnda r1, r2 och r3) är det uppmätta avståndet till respektive referenspunkt och skärningspunkten (benämnd A) mellan de tre cirklarna är aktuell position.



Figur 1. *Triangulering i två dimensioner.*

2.2 Avståndsmätningar

Avståndsmätningar kan göras med en mängd olika tekniker, LIDAR, ultraljudssensorer, TOF via radio och klocksynkronisering via radio för att nämna några. Alla med sina för- och nackdelar. LIDAR och ultraljudssensorer mäter relativa avstånd till närliggande föremål med hög precision och hög frekvens, vanliga tillämpningar är exempelvis backsensorer på bilar eller i självgående dammsugare. Nackdelen med dessa tekniker är att det kan uppstå problem att räkna ut den absoluta position i rummet då olika föremål kan skugga varandra.

TOF och klocksynkronisering via radio mäter absoluta avstånd mellan noder. Metoderna har lägre precision och längre initieringstid än LIDAR och ultraljudssensorer. Fördelen med att mäta absoluta avstånd istället för relativa är att man undviker risken med att referensytan skuggas av andra föremål. Att mäta TOF via radio ställer höga krav på hårdvaran, då signalen måste samplas otroligt snabbt eftersom den färdas med ljusets hastighet. TOF mäts ofta tur och retur mellan två noder, den totala tiden delas med två och multipliceras med ljusets hastighet för att

ge avståndet. Samplingsfrekvenser i storleksordningen 10^9 Hz ger en upplösning på ungefär 30 cm.

Klocksynkronisering via radio är den teknik som till exempel GPS-systemet [5] bygger på. Alla noder i systemet har synkroniserade klockor. För att genomföra en avståndsmätning sänder referensnoden en tidsstämpel på vad klockan var när meddelandet skickades. När en nod tar emot meddelandet jämför denna vad klockan var när meddelandet skickades och vad klocka var när meddelandet togs emot. Skillnaden i tid multipliceras med ljushastigheten för att ge avståndet. Klocksynkronisering ställer höga krav på både hård- och mjukvara. Hårdvaran då samplingsfrekvensen måste vara hög av ovan nämnda anledning och mjukvaran då det krävs smarta algoritmer för att utföra själva klocksynkroniseringen, det vill säga att alla noder i systemet ska enas om en global klocka.

2.3 Filtrering

Att enbart basera positionen på data från avståndsmätningar ger upphov till en rad problem, detta då hård- och mjukvaran som genomför avståndsmätningarna oftast returnerar ett värde som är av högre upplösning än avståndsmätningen egentligen är. Resultatet kan till exempel bli att den beräknade punkten som man befinner sig på flyttas omkring, trots att ens verkliga position är statisk. Befinner man sig i rörelse kan den beräknade sträckan man färdats bli längre och även hastigheten högre än vad den egentligen är. I Figur 2 visas en ofiltrerad GPS-baserad positionsuppskattning då mottagaren rör sig med konstant hastighet längs en rät linje. I applikationer där tajming är kritiskt kan detta få stora konsekvenser, till exempel då föremål kan befinna sig på andra positioner och med andra hastigheter än vad man tror. För att få bukt med problemet används olika typer av filter, där den vanligaste typen är ett Kalmanfilter [6]. I positioneringsfallet viktar Kalmanfiltret intern med extern data. I tillämpningen som kandidatarbetet utgör består den interna datan av position, hastighet samt riktningsvinkel och den externa datan av avståndsmätningar. Viktningen görs baserat på kvalitén på datan, representerade av standardavvikelser och de är dessa som utgör viktfaktorerna. Kalmanfiltret returnerar inte bara en punkt i vilken man befinner sig utan även en standardavvikelse på den uträknade positionen.

Figur 2. *Bild på ofiltrerad GPS-baserad positionsuppskattning då mottagaren rört sig med konstant hastighet längs en rät linje.*

2.4 MAC-lager

Paketkollision kan uppstå då två eller fler enheter i samma nätverk försöker att kommunicera vid samma tidpunkt. Då trådlös kommunikation sker i ett öppet medium, till skillnad från trådad kommunikation krävs också att frekvensen hos radiovågorna, som utgör meddelandet för enheterna är densamma.

För att säkerställa att den data som skickas kommer fram till mottagaren kan man använda sig av olika metoder. *Collission avoidance,* vilket är den metod som använts i detta projekt, går ut på att undvika att paketkollision uppstår. Däremot går *Collission detection* ut på att upptäcka ifall paketkollision har uppstått och därmed vidta eventuella åtgärder.

2.4.1 Time Division Multiple Access - TDMA

För att möjliggöra kommunikation av flera enheter i ett nätverk krävs ett systematiskt sätt att schemalägga alla enheter så att inte paketkollision uppstår. Ett sätt att uppnå detta är att dela upp tiden så att varje enhet får en specifik tid då just den får använda radiofrekvensen. Detta kan uppnås genom att dela upp en dynamisk tidsram i känt antal konstanta tidsluckor där varje enhet tilldelas en tidslucka. Ett alternativt sätt är att låta storleken på tidsramen vara konstant och storleken på tidsluckorna dynamiska. Ett tredje sätt är att låta storleken på både tidsramen och tidsluckorna vara konstanta, detta medför också att antalet tidsluckor i en tidsram är konstant, vilket innebär att antalet möjliga noder i systemet är konstant. Så är fallet för Chameleon-MAC, se kapitel 2.4.2.

2.4.2 Chameleon-MAC

Chameleon MAC algoritmen har utvecklats av en forskargrupp med medlemmar från Chalmers Tekniska Högskola och University of Geneva. Algoritmen är avsedd för användning inom mobila ad hoc nätverk där höga krav på prestanda och stabilitet ställs [7].

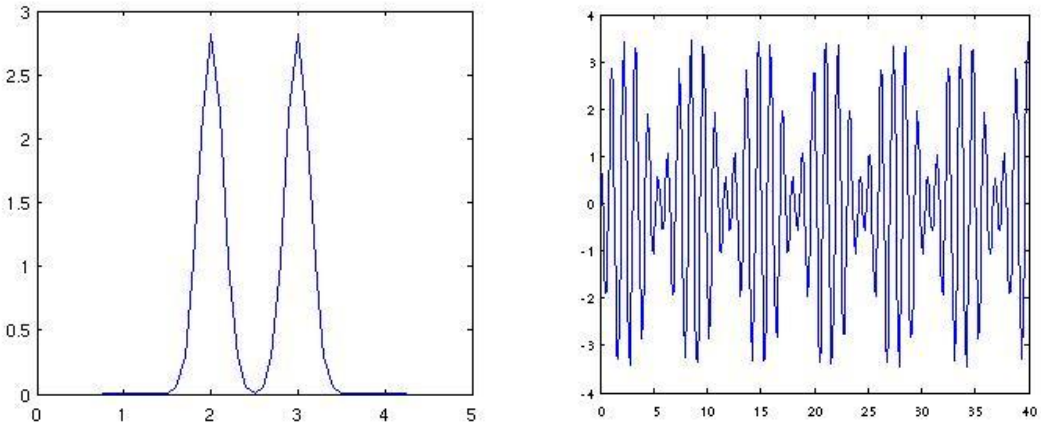
Chamelion MAC går ut på att noderna tävlar om tidsluckor genom att slumpmässigt välja ut en ledig tidslucka. Inom vald tidslucka väntar noden ut slumpmässigt vald tidsfördröjning innan sändning. Tidsluckan markeras som upptagen om noden lyckades sända data. Ifall det skulle inträffa så att två noder ska komma in i systemet och att de skulle välja samma tidslucka att sända i, kommer den senaste noden meddelas av underliggande hårdvara att radiokanalen är upptagen. Då skall noden vänta ut tiden motsvarande några tidsramar innan den försöker igen. Givetvis är det ett krav att hårdvaran har stöd för *carrier sensing*, det vill säga den tekniken som känner av om radiokanalen är ledig.

Tidiga implementationer av Chameleon MAC tillåter endast tidsramar med konstant storlek *n* i ett nätverk med maximalt *n* noder. Ifall det finns färre aktiva noder i systemet än den var konfigurerad för, kommer outnyttjade tidsluckorna att vara tomma.
Slumpmässiga algoritmer som används i Chameleon MAC medför att systemet har en konvergenstid efter vilken alla noder kommer att ha en tidslucka [7].

Chameleon MAC har implementerats i Gulliver projektet i form av exekverbar kod för MICAz modulerna. I kandidatprojektet har den varit en inspirationskälla när det gäller förståelse av *self-* * tekniker.

2.5 Ultra-wideband

Ultra-wideband (UWB) [8] är en speciell variant av radio. UWB bygger på att korta impulser skickas ut istället för längre kombinationer av sinussignaler som i traditionell radio. På grund av att signalen är uppbyggd av korta impulser krävs det många frekvenser för att modellera signalen, därav namnet *Ultra-wideband*. En av fördelarna med UWB är att det är väldigt låg effekt men ändå hög dataöverföringshastighet. Detta leder till att signalen inte stör andra lågbands-radiosignaler, det innebär dock också att räckvidden inte blir speciellt lång. En annan stor fördel med UWB är att man alltid vet vilken signal som reflekterats och vilken som inte gjort detta. För att göra detta kollar man på vilken impuls som kommer fram först, eftersom det då måste vara den icke reflekterade signalen. I vanlig lågbandsradio hade detta aldrig fungerat i och med att signalerna där är lågbands sinussignaler. En kombination av reflektion och den direkta signalen blir med UWB två impulser och med lågbandsradio ännu en sinussignal där det har skett en fasförskjutning. Som vi ser i Figur 4 i den vänstra delen är det enkelt att mäta avståndet mellan de två impulserna medan fasskillnaden i den högra delen av figur 4 inte går att mäta i och med att man inte har någon referens att mäta mot.



***Figur 3**. Kombination av två impulser till vänster och två lågbandssignaler till höger, y-axel anger amplitud och x-axel anger tid.*

2.6 Systemplattform

Systemplattformen, som positioneringssystemet och MAC-lagret är implementerade på, utgörs av en, i följande text kallad, robot. Roboten är byggd på en färdig kaross där komponenter såsom givare, mikroprocessorer, kommunikationsmoduler, modul för avståndsmätning med mera har monterats för att göra det möjligt att simulera olika trafiksituationer som kan uppstå i den verkliga trafiken. Chalmers robotförening har varit ansvariga för montering av roboten samt utveckling av diverse kretskort.

2.6.1 Plattformens komponenter

Nedan följer en beskrivning av ingående komponenter för systemplattformen. Se Figur 4 för en schematisk bild över hur komponenterna är sammankopplade.

Den huvudsakliga programkoden med diverse beräkningar körs på ett egenutvecklat moderkort som kallas för *Main-board*. För att möjliggöra positionering har det köpts in ett kretskort, från företaget *Time Domain*, som heter *Ranging and Communication Module* **(**RCM). Mer ingående information om denna komponent kan läsas kapitel 2.6.2. RCM är sammankopplad med moderkortet via ett lokalt seriellt gränssnitt. *Main-board* består bland annat av en mikroprocessor, *AT-XMEGA-32A3-AU* med en klockfrekvens på 32MHz. Det är på denna mikroprocessor de båda MAC-lagren är implementerade samt Kalman-filtret.

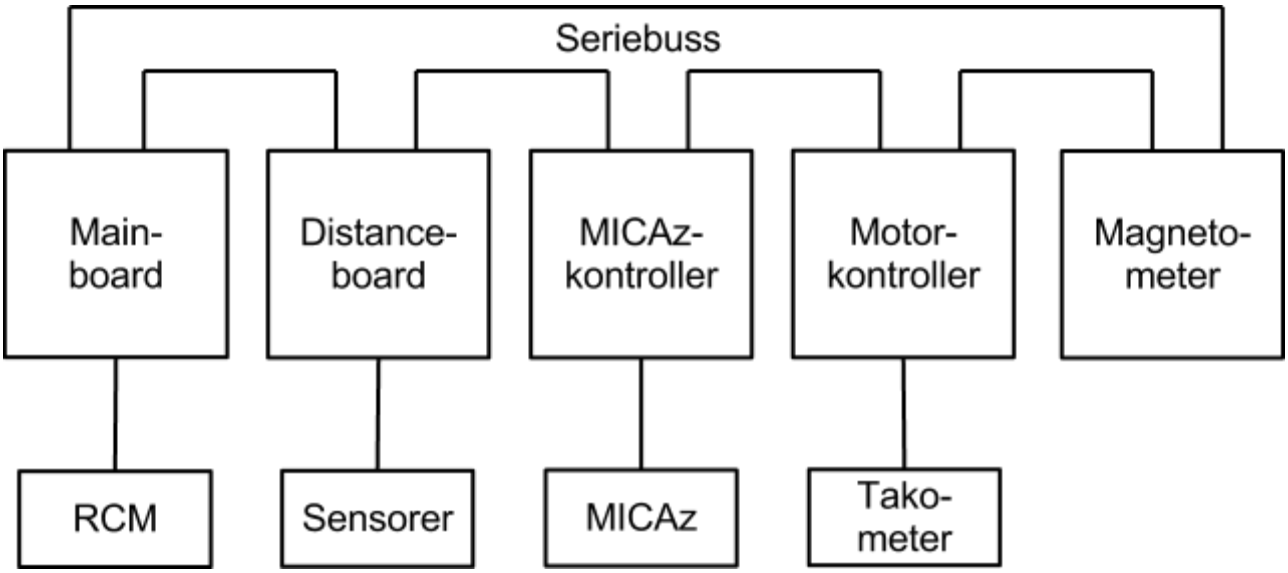
På robotens front sitter ultraljudssensorer för att mäta avstånd till framförvarande objekt. Dessa används för att förhindra att roboten kör in i föremål. Sensorerna är kopplade via ett lokalt seriellt gränssnitt till ett kretskort kallat *Distance-board*.

Robotens hastighet regleras via ett motorkontrollerkort. Motorkontrollern får information från en takometer angående motorns varvtal vilket gör det möjligt att ange hur långt roboten har färdats. I motorkontrollern anges dessutom vilken spänning som skall sättas på styrservon för att uppnå en viss styrvinkel på framhjulen (relativt en tänkt axel parallellt med robotens långsida). Med kännedom om både styrvinkel och den sträcka roboten färdats kan den uppskatta sin position i ett koordinatsystem. Denna uppskattning sker i *Main-board*.

För att kommunicera med andra robotar används en MICAz-modul. MICAz är kopplad till ett kretskort som kallas MICAz-*controller* vilket möjliggör för övriga komponenter i systemet att kommunicera med MICAz, och således med andra robotar.

Roboten har också en magnetometer för att kunna avgöra åt vilket håll den står. Denna komponent utnyttjas även i positioneringssystemet.

Main-board, *Distance-board*, MICAz-*kontroller* samt magnetometern är kopplade till en seriell buss så att alla komponenter kan kommunicera med varandra. Det protokoll som används för kommunikation på bussen är UART.

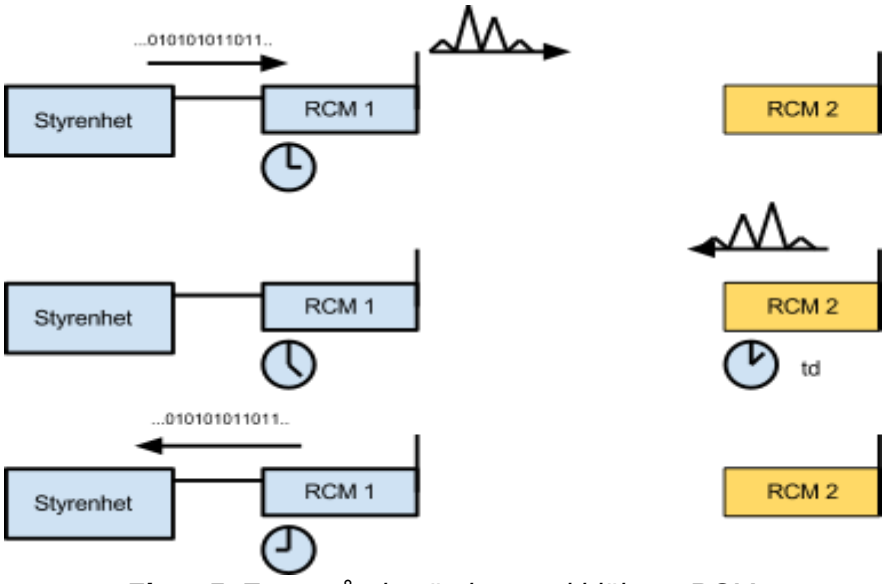


Figur 4. *Schematisk bild över hårdvaran på Gulliverroboten.*

2.6.2 Ranging and Communication Module

Som det har nämnts tidigare i kapitel 1.3 så bygger positioneringssystemet på högpresterande avståndsmätare RCM från företaget Time Domain. Dessa styrs via ett seriellt gränssnitt, men stödjer även Ethernet. Figur 5 beskriver hur en avståndsmätning går till. Styrenheten anropar RCM modulen (RCM1) med ett datapaket, som har bland annat mottagarens id (RCM2) i sin struktur. RCM1 kodar om datapaketet till ett pulståg och sänder ut den på radiokanalen. När första pulsen mottas startas en timer med en upplösning på tiotals pikosekunder. Den mottagande enheten (RCM2) tar emot pulståget och väntar ut determinerad tid (*td*) som beror på konfigurationen av RCM. Sedan sänds ett pulståg med RCM1 id som mottagare. När det andra pulståget är mottaget stoppar RCM1 sin timer. Tiden subtraheras med *td*, divideras med två och multipliceras med ljusets hastighet (*c*), enligt formel (1). Sedan bäddas information om avståndsmätningen in i ett datapaket som skickas till styrenheten.

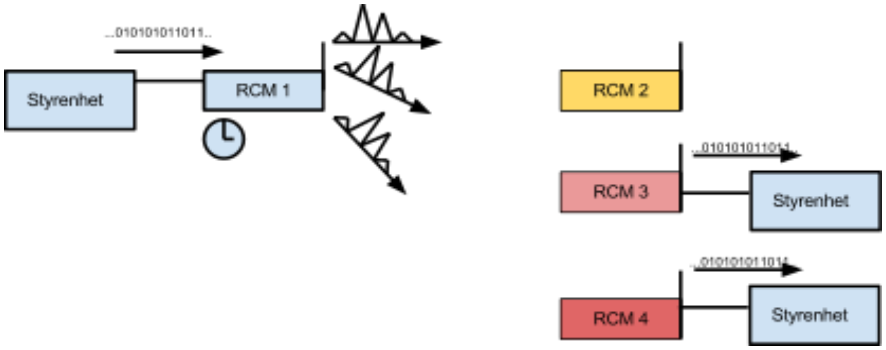
$$Distance = \frac{time - td}{2} \cdot c \qquad (1)$$



Figur 5. *En avståndsmätning med hjälp av RCM.*

Anledningen till att RCM väntar ut tiden *td* är att pulståget kan bli väldigt långt om det bär mycket information. Varje symbol i datapaketet kodas i PII^2 pulser där PII (*Pulse Integration Index*) kan ställas in mellan 6 och 10. Högre PII värden ger bättre signal till brus förhållande samt längre maximala avstånd mellan RCM. Dock kommer detta på bekostnad av längre tidsfördröjning *td* som växer polynomiskt. Projektets mål är att skapa ett högpresterande positioneringssystem för flera mobila noder, därför har RCM:erna konfigurerats med lägsta PII=6, vilket ger lägsta möjliga tidsfördröjning på ca 25 ms. Det är en hårdvarubegränsning som påverkar tidsluckornas storlek som beskrevs i kapitel 2.4.2.

Radiomodulerna har även stöd för dataöverföring, vilket utnyttjas för att skapa ett ad hoc *self-** system. Distribution av nodernas klockor och medlemslistor sker i samband med avståndsmätningar vilket tillåter högre kapacitet av radiofrekvensen. Figur 6 visar dataöverföring via RCM.



Figur 6. *Dataöverföring i samband med en avståndsmätning.*

3 Metod och genomförande

I detta kapitel förklaras det hur projektet lagts upp, hur planeringen gått till och hur projektet genomförts. Det förklaras även hur projektet har förändrats när förutsättningarna har ändrats under projektets gång. Testscenarion både under projektet gång och de slutgiltiga testningsmetoderna gås igenom.

3.1 Förstudier

En förstudie har genomförts där ett antal artiklar och rapporter lästes. Redan existerande kod för Gulliver-plattformen har studerats för att få en bra uppfattning av systemet. Större delen av denna kod har varit odokumenterad eller väldigt dåligt dokumenterad vilket har lett till att mycket tid har fått läggas på detta. Kandidatgruppen har haft löpande veckomöten med Henk Wymeersch och Gabriel Garcia för diskussion om positionsuppskattning och filtrering, med Elad Schiller för diskussion om projektet som helhet. Vidare har även Mohamed Mustafa och Mitra Pahlavan kontaktats för diskussion angående *Media Access Control*.

3.2 Utvecklingsmetod

I början av projektet sattes ett antal milstolpar upp, tydligt definierade både vad det gällde innehåll och tidsfrist. Det visade sig dock ganska snabbt att vissa av dessa var svåra att uppnå då den hårdvara som används saknar viss funktionalitet. Då detta uppdagats togs beslutet att använda en iterativ utvecklingsmetod. Detta underlättar då lösningen för projektets problemställning har vuxit fram under projektets gång och det har varit en prioritet att ha frihet att kunna ändra tekniker för att nå målet. För att bekanta oss med hårdvaran skapades förenklade versioner av vissa funktioner. Exempelvis skapades en *master-slave*-baserad *time division multiple access* (TDMA) algoritm innan det självstabiliserande MAC-lagret utvecklades. Detta gav oss insikt i hur seriekommunikationen till och från RCM:en fungerar, lämpligt uppdateringsintervall för avståndsmätningarna samt systemets latens. Det sistnämnda är viktigt för att uppskatta hur hög grad av synkronisering som är möjlig mellan olika noder.

3.3 Utvecklingsmiljö och programmeringsspråk

C lämpar sig bra att programmera i då man vill utveckla program till bland annat mikroprocessor då detta språk möjliggör bättre kontroll över hur data skickas mellan olika mikroprocessorer. Därför var det inget tvivel när frågan kom till vilket programmeringsspråk som skulle användas. C++ har även används vid ett fåtal tillfällen då konfigurering av RCM:erna var nödvändigt.

All programutveckling har skett i utvecklingsverktyget Eclipse. Eclipse har förvisso inte stöd för programmeringsspråket C i dess originalform, men tack vare plugin som AVRDUDE är det möjligt att utveckla program i C för ATMEL-processorer.

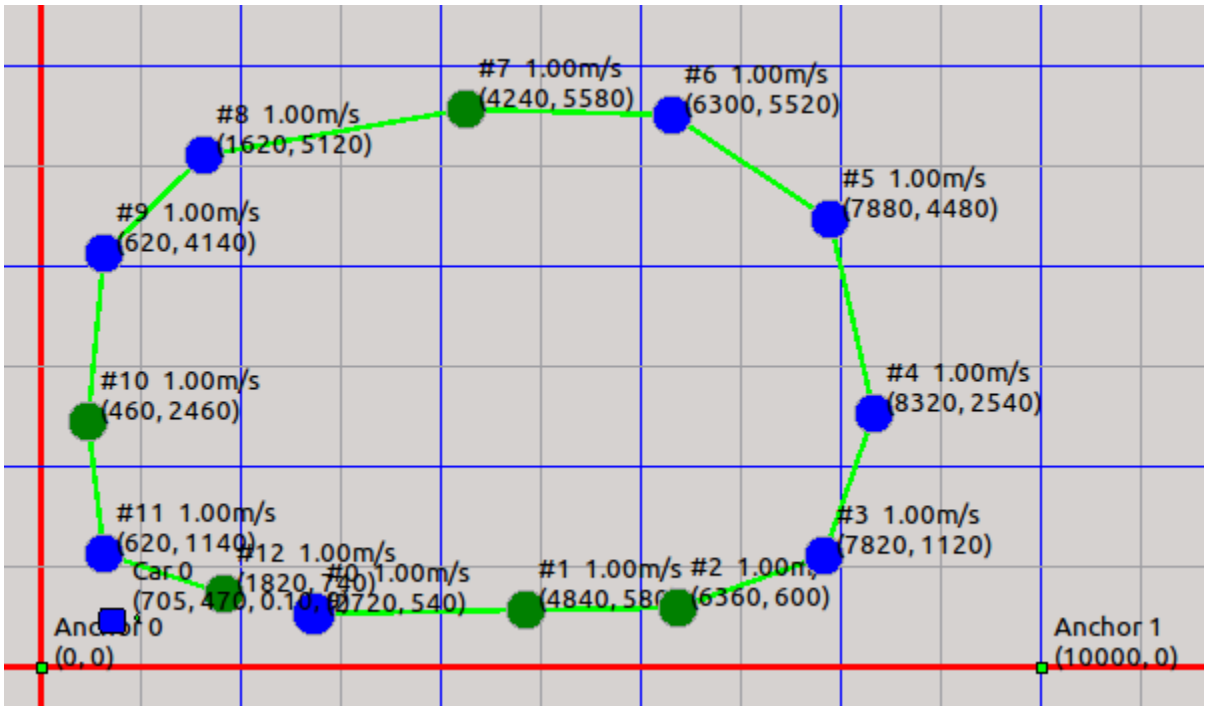
3.4 Testning av funktionalitet

Kontinuerligt under projektets gång genomföredes tester av de ingående komponenterna, ofta genom modultestande. Modultestande innebär att programmet delas upp i mindre delar som testas fristående av varandra. För att verifiera att programmet betedde sig på önskvärt sätt användes ofta en serieterminal för utskrifter eller det mer enkla sättet att blinka de lysdioder som är monterade på kretskortet.

I projektets slutskede genomfördes ett antal testomgångar av systemets samtliga komponenter, alla integrerade i plattformen. Detta genomfördes för att verifiera att kraven var uppfyllda och att systemet som levererats fungerade på önskvärt sätt. Metoder för att testa de olika delarna presenteras nedan.

3.4.1 Positionering

En simpel rutt definierades i den karteditor skapad för plattformen *Gulliver* [9]. Rutten består av ett antal förbundna koordinater och presenteras i Figur 7, ruttens längd är cirka 25 meter. Fyra koordinatpunkter jämnt fördelade längs rutten markerades på golvet i den lokal där testerna utfördes, varpå roboten kördes upprepade varv på rutten. Varje gång roboten passerade en av punkterna på kartan noterades hur många mm roboten avvek från denna punkt i verkligheten. Samtliga av de nedan presenterade testfallen utfördes oberoende av varandra.



Figur 7. *Visualisering av den rutt som skapades.*

Det genomfördes tre testkörningar:

1. Roboten bestämmer sin position baserat på endast intern data. Det vill säga data från styrservo, takometer och magnetometer. Positionen kommer således inte att filtreras.

2. Roboten bestämmer sin position baserat på endast extern data. Det vill säga data från avståndsmätningarna. Positionen kommer således inte att filtreras.
3. Roboten filtrerar sin position baserat på både intern och extern data.

De olika testfallen utfördes på en öppen yta, som synes i Figur 8, och bör ge en tydlig bild över hur bra system fungerar. Testfallen bör även belysa skillnaden i att utföra positionsbestämningen med och utan filtrering.

Figur 8. *Platsen där positioneringssystemet testades.*

3.4.2 MAC-lager

Validering av MAC-lagret utfördes i en interferensfri miljö för att urskilja paketkollisioner som uppstår då flera närvarande noder misslyckades med TOF-mätningar på grund av reflektioner från omgivningen. Vid varje test har det genomförts 10000 avståndsmätningar. Testerna där MAC-lagret styrde avståndsmätningarna, genomfördes efter stabilisering av nätverket och godkännande av alla noder.

Fyra scenarion har testats:

1. En nod utan MAC-lager. Visar kvalitén på kommunikationsmiljön. Används som referens.
2. Två noder utan MAC-lager. Visar om paketkollisioner uppstår och därmed behovet av ett MAC-lager.
3. En nod med MAC-lager. Användes för att visa prestandan hos MAC-lagret.
4. Två noder med MAC-lager. Visar om funktionskraven har uppfyllts samt prestandan hos MAC-lagret.

Testerna bör visa huruvida MAC-lagret uppfyller ställda krav, det vill säga kollisionsfri kommunikation med linjär och konstant tidsfördröjning i ett nätverk med minst två noder.
MAC-lagret inför längre tidsfördröjning mellan avståndsmätningarna i ett nätverk med flera noder. Då antalet noder växer kommer detta att leda till sämre precision mellan trianguleringsrundorna. Därför har ett extra test utförts för att visa hur mycket systemet påverkas av fördröjningen. I detta testfall kommer roboten att filtrera sin position baserat på både intern och extern data, likt testfall 3 i kapitel 3.4.1 med enda skillnaden att MAC-lagret simulerar närvaro av fem bilar.

4 Implementering och resultat

Kandidatarbetet resulterade i ett positioneringssystem med två versioner av kollisionsfria MAC-lager. Det ena MAC-lagret förlitar sig på en extern enhet, MICAz, för att synkronisera klocka och hantera ingående noder i nätverket medan det andra MAC-lagret sköter detta internt. MAC-lagren är självstabiliserande på så sätt att nya enheter kan läggas till eller tas bort utan att systemet behöver konfigureras om.

Positionen bestäms genom att filtrera robotens interna data, såsom körd sträcka och styrvinkel, med den externa datan i form av aktuell vinkel med hjälp av magnetometern samt avståndsmätningar till fördefinierade ankare med hjälp av ett Kalmanfilter.

I följande kapitel presenteras hur de olika systemen har implementerats.

4.1 Positioneringssystem

Ett filter har implementerats på roboten som väger data från avståndsmätningarna med robotens interna data. Med hjälp av den interna datan beräknas robotens troliga position. Denna position tenderar dock att driva med avståndet roboten kört. För att få en god positionsuppskattning korrigeras därför positionen varje gång en avståndsmätning genomförs.

Robotens osäkerhet representeras av en standardavvikelse som växer med avståndet roboten kört. Avståndsmätningarna levereras i form av en normalfördelning, $Z \in N(\mu_z, \sigma_z)$, där väntevärdet är det levererade avståndet och standardavvikelsen är mätosäkerheten. Vårt filter fungerar på så sätt att roboten beräknar avståndet till ankaret den gjort avståndsmätning till och använder detta som väntevärde för att skapa normalfördelningen $Y \in N(\mu_y, \sigma_y)$.

Beräkning av robotens uppdaterade normalfördelning: $X \in N(\mu, \sigma)$ går till enligt följande[10]:

$$\sigma = \frac{\sigma_z \sigma_y}{\sigma_z + \sigma_y} \tag{2}$$

$$\mu = \frac{\mu_z \sigma_y + \mu_y \sigma_z}{\sigma_z + \sigma_y} \tag{3}$$

Ekvation (2) visar hur den nya standardavvikelsen uträknas och ekvation (3) visar hur det nya väntevärdet, det vill säga avståndet till ankaret, beräknas.

4.2 MAC-lager med extern synkroniseringskälla

Detta kapitel redogör för hur det MAC-lager som förlitar sig på MICAz är implementerat.

Den funktion som utgör MAC-lagret är implementerade i *Mainboard* och anropas varje gång en positionsuppdatering ska utföras. MAC-lagret returnerar då 1 eller 0 beroende på om den aktuella Gulliverroboten befinner sig i sin tidslucka eller ej. Ett blockschema för MAC-lagret kan betraktas i Figur 9.

Varje Gulliverrobot har en MICAz-modul till förfogande. Eftersom MICAz redan har ett fungerande MAC-lager har dessa moduler använts vid framtagande av det MAC-lager som beskrivs i detta kapitel. MICAz-modulen på den aktuella Gulliverroboten förmedlar information angående antal robotar i nätverket samt global synkroniserad klocka vidare till *Mainboard. Det* är därför möjligt att räkna ut vilken robots tur det är att tillgå mediet och på så vis bestämma när den aktuella roboten har möjlighet att uppdatera sin position.

Den funktion på MICAz som möjliggör att tillhandhålla information angående ingående robotar i nätverket bygger på Heartbeat-protokollet [11]. Noder som överhörs under pågående tidsram läggs till i en lista med aktiva noder, *active list*. Noder som inte var aktiva under senaste *x* tidsramar läggs till i en lista med misstänkta noder, *suspicious list*. Noder som var misstänkta under *y* tidsramar och inte fanns i andras listor utesluts från nätverket. Det tar minst *x+y* tidsramar innan en nod kan uteslutas från nätverket, vilket tillåter att noderna som har tillfälliga kommunikationsproblem kan återhämta sig. MICAz använder sig av *converge-to-MAX* [12] för att synkronisera klockan internt med andra MICAz-moduler. Det går till på så vis att alla ingående MICAz-moduler i nätverket annonserar aktuellt värde på sina klockor under deras tidsluckor och anpassar dessa till den största klockan.

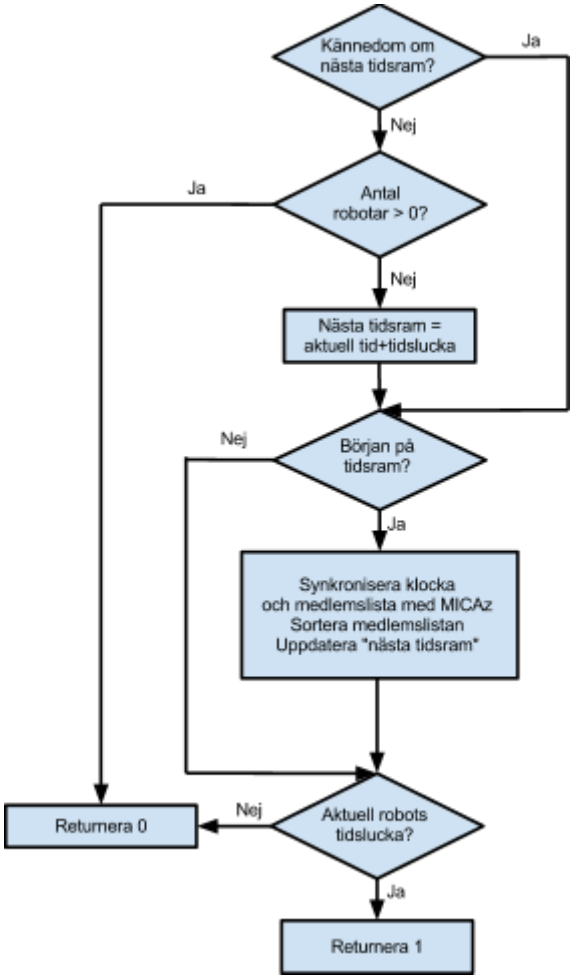
För MAC-lagret är det i förväg bestämt hur stor en tidslucka är. Detta bestäms utifrån hur lång tid det tar att genomföra en avståndsmätning till varje ankare. Varje robot har en intern klocka som håller ordning på tiden samt en intern räknare som håller koll på vid vilken tid nästa tidsram börjar. Den interna kockan synkroniseras med jämna mellanrum med den klocka som erhålles från MICAz.

Eftersom att alla robotar i nätverk har kännedom om synkroniserad klocka och medlemslista finns det ett trivialt sätt att räkna ut när nästa tidsram börjar. Genom att beräkna klockans värde modulo storleken på tidsramen fås på så vis hur lång tid in på den aktuella tidsramen klockan är. Ett problem med detta är att eftersom storleken på tidsramen är beroende av antalet medlemmar i medlemslistan kan en ickedeterministisk tidsfördröjning uppkomma. Låt säga att klockan vid ett tillfälle visar 92260 och antalet medlemmar i nätverket vid samma tillfälle är 5. Om det antas att tidsluckan för varje robot är 150 ms kommer storleken på en tidsram då att vara 750 ms. 92260 modulo 750 = 10. Alltså 10 ms in i aktuell tidsram. Detta medför att det är 740 ms kvar på aktuell tidsram. Det vill säga att nästa tidsram börjar 92260 + 740 = 93000. Om det nu har tillkommit en robot i medlemslistan blir storleken på tidsramen istället 900 ms. 93000 mod 900 = 300. Alltså beräknas man vara 300 ms in i tidsramen när tidsramen för 5 noder egentligen precis började. Därför kommer den nod som har sin tidslucka först att få vänta tills nästa tidsram innan det är dess tur.

Därför har istället följande metod använts för att räkna ut början på nästa tidsram. Genom att låta varje robot, som är etablerad i nätverket, annonsera när nästa tidsram börjar under sin tidslucka, kan nytillkomna robotar som ej har kännedom om tidsramen på så vis räkna ut denna. Om antalet robotar i nätverket (borträknat sig själv) är 0 och det saknas kännedom om när

nästa tidsram börjar, antar man sig vara den första etablerade roboten i nätverket. Därför definieras nästa tidsram till aktuell tid plus storleken på en tidslucka.

För att hålla reda på alla robotar tilldelas de ett unikt id. Om antalet robotar i nätverket (borträknat sig själv) är större än 0, sorteras listan med ingående robotars id. Eftersom alla robotar sorterar listan efter robotarnas id kommer den sorterade listan att se samma ut för alla robotar. Dessvärre finns aktuell robot inte med i listan av övriga etablerade robotar i nätverket, därför får denne jämföra sitt id med robotarna i listan och på så vis räkna ut sin position. Då roboten vet sin position i listan, när nästa tidsram börjar samt hur lång en tidslucka är kan den på så vis räkna ut vid vilken tid dess tidslucka är.



Figur 9. *Blockschema för MAC-lager med hjälp av MICAz.*

4.3 MAC-lager med hjälp av RCM

Vid design av MAC-lager för positioneringssystem kan utvecklaren föredra en lösning med externa synkroniseringskällor på grund av dess tillgänglighet. Dock när feltoleransen ska in i bilden har interna kommunikationstekniker en klar fördel: intern kommunikation bekräftar anslutning av RCM modulerna. Ett exempel är Gulliverroboten, där MICAz modulen och

RCM:en har olika strömkällor. Det kan inträffa att batteriet som strömförsörjer RCM:en laddas ur, och radiomodulen stängs av, medan MICAz fortfarande rapporterar om bilens närvaro i nätverkets medlemslista. I detta avsnitt presenteras ett MAC-lager som använder RCM:ernas interna kommunikationstekniker och som garanterar att RCM:erna fungerar.

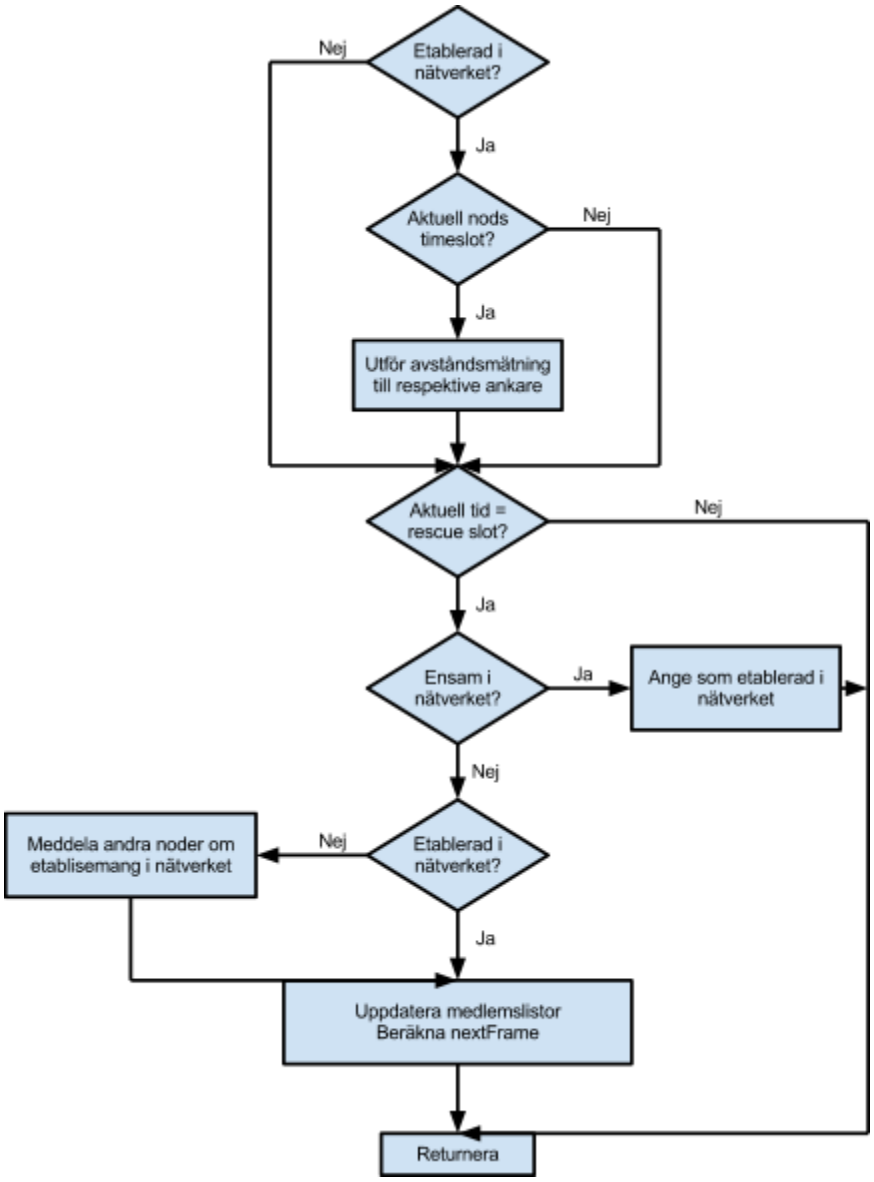
RCM-MAC är namnet på en uppsättning av olika algoritmer som används i systemet för att åstadkomma ett TDMA-liknande beteende hos schemaläggaren som styr RCM:erna. Huvudmålet var att implementera ett MAC-lager med varierande storlek på tidsramen, som använder radiomodulernas interna kommunikationstekniker. Detta bidrar till att systemet kan installeras ad hoc och fungera oberoende av externa synkroniseringskällor. RCM-MAC har även *self-** egenskaper, vilket tillåter noderna att komma in och lämna nätverket, utan att behöva konfigurera om systemet.

Se blockschema för följande kodbeskrivning i Figur 10. I likhet med algoritmerna som exekveras på MICAz modulerna, så använder RCM-MAC sig av Heartbeat-protokollet [11] för tillhandahållande av medlemslistor.

För att tillåta nya noder att komma in i systemet läggs en extra tidslucka mellan två tidsramar, vidare kallad *rescue slot.* Den kan användas enbart av nytillkomna noder och noder som har uteslutits från systemet på grund av radiointerferenser. En nod som ska etablera sig i nätverket sänder ut ett datapaket via radiomodulen under pågående rescue slot.

Aktiva noder använder *rescue slot* vid beräkning av nästa tidsrams storlek. På detta sätt kommer systemets medlemslistor att uppdateras under samma tid. *Rescue slot*-storleken var satt till *50 ms*. Storleken på tidsramen bestäms av antalet noder (*n*) i systemet och räknas ut enligt *150 ms x n*. På så vis blir fördröjningen mellan trianguleringsrundorna deterministisk i ett stabiliserat nätverk. Anledningen till de valda tidskonstanterna är de begränsningar som finns i hårdvaran och mjukvaran, delvis i RCM modulerna (PII-indexet som beskrivs i kapitel 2.6.2) och delvis i Gulliver-roboten, då positioneringskoden exekveras i en loop med andra funktioner som inför en vis fördröjning.

För att synkronisera nodernas klockor använder systemet sig av Grasshopper algoritmen [13]. Istället för att justera klockan mot ett gemensamt värde, som görs i converge-to-MAX alternativt converge-to-MEDIAN [12], observerar algoritmen mönstret på inkommande pulser. Med kännedom om tidsramens storlek kommer algoritmen att räkna ut när nästa serie av pulser beräknas tas emot och justera den interna klockan så att den blir synkroniserad med nätverket.



Figur 10. *Blockschema för MAC-lager med hjälp av RCM.*

4.4 Resultat från testning av systemet

Nedan presenteras resultatet från den i projektets slutskede genomförda testomgången av systemets samtliga komponenter, alla integrerade i plattformen.

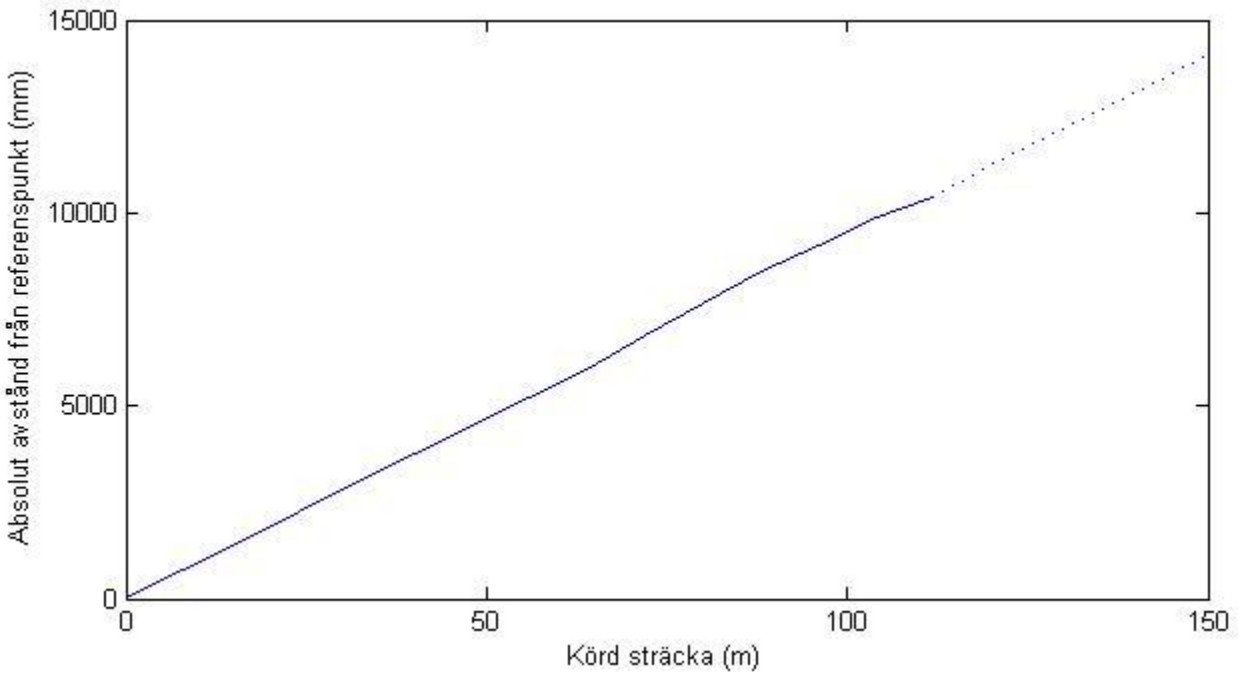
4.4.1 Testning av positioneringssystemet

Resultatet från det i kapitel 3.4.1 beskrivna testförfarandet av positioneringssystemet återfinns i följande kapitel. Tabell 1 visar avvikelse från skapad rutt i testfall 1 (positionsbestämningen baserades enbart på intern data, det vill säga data från takometer, styrservo och kompass). Som synes ökar avvikelsen för varje varv, detta då roboten inte får någon data om dess absoluta position i rummet utan endast relativt den position som beräknades i föregående iteration. Detta innebär att felet aldrig korrigeras utan hela tiden kommer att öka då felet byggs

på i varje iteration. I Figur 11 visas en graf över hur robotens position avvek från den skapade rutten i förhållande till hur lång sträcka den hade kört. Tyvärr fick testningen avbrytas efter knappt fyra varv då roboten avvikit så mycket från rutten att den hotade att krocka med kringliggande objekt.

Tabell 1. *Absolut avvikelse i mm från referenspunkt då bilen körde flera varv på banan definierad i karteditorn. Testfall 1, positionsuppskattningen baserad på intern data.*

Varv\Testpunkt	1	2	3	4
1	0	734	1487	2245
2	3012	3773	4476	5242
3	5938	6801	7648	8490
4	9153	9876	10420	-
5	-	-	-	-



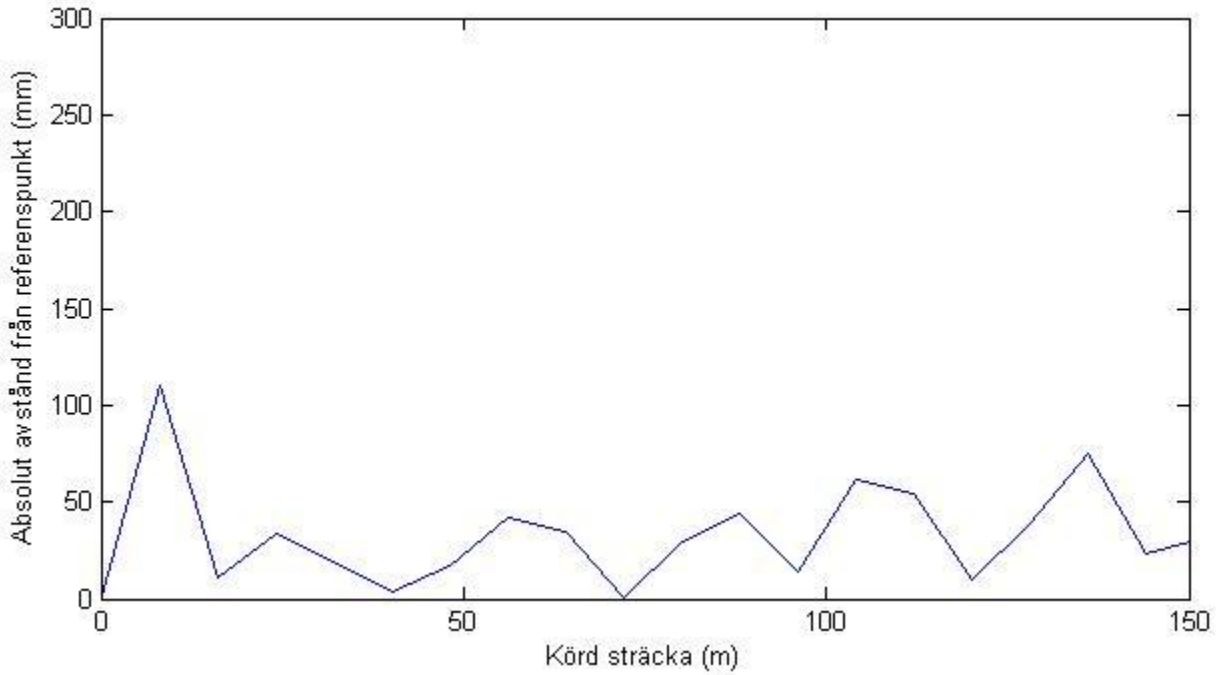
Figur 11. *Graf över hur felet växer med avståndet roboten kört då positionsuppskattningen endast baseras på intern data.*

I testfall 2, då positionsbestämningen gjordes enbart baserat på avståndsmätningarna, avvek roboten inte med mer än maximalt 110 mm från sin fördefinierade rutt. Vad som inte framgår av tabellen är att den bestämda positionen fluktuerade väldigt mycket, även när roboten var

stillastående. Hade linjer mellan de skattade positionerna dragits hade resultatet snarare liknat kurvan i Figur 2 i kapitel 2.3 än en jämn kurva. I Tabell 2 presenteras mätdata från testfall 2. Medelvärdet på avvikelserna från referenspunkterna är 33 mm.

Tabell 2. *Absolut avvikelse i mm från referenspunkt då bilen körde flera varv på banan definierad i karteditorn. Testfall 2, positionsuppskattningen baserad på extern data.*

Testpunkt\Varv	**1**	**2**	**3**	**4**	**5**
1	0	110	11	34	19
2	4	17	42	35	1
3	30	44	14	62	54
4	10	39	75	23	31

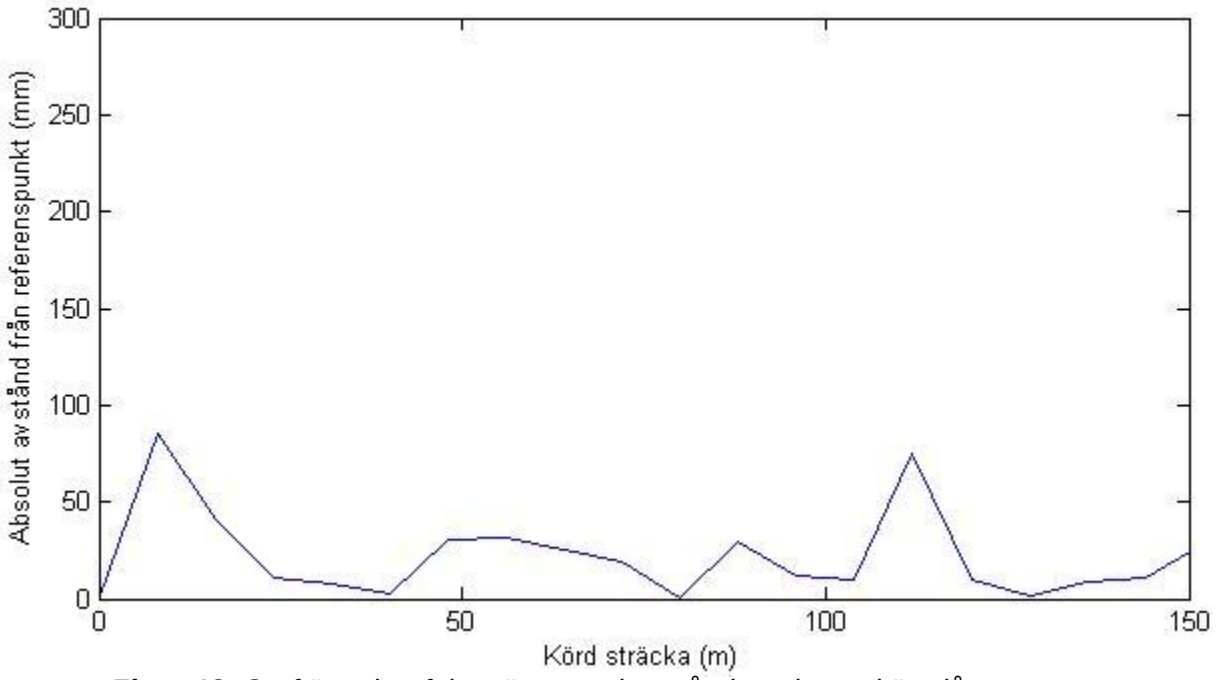


Figur 12. *Graf över hur felet växer med avståndet roboten kört då positionsuppskattningen endast baseras på extern data.*

Skillnaden i mätdata då positionsuppskattningen gjordes baserad på filtrerad intern och extern data (testfall 3) kontra endast extern data (testfall 2) är marginell. Vad som inte framgår av tabellerna är dock att roboten följer en mycket jämnare bana i detta testfall än då positioneringen utfördes baserat endast på extern data. Mätdata från testfall 3 presenteras i Tabell 3. Medelvärdet på avvikelserna från referenspunkterna är 22 mm.

Tabell 3. *Absolut avvikelse i mm från referenspunkt då bilen körde flera varv på banan definierad i karteditorn. Testfall 3, positionsuppskattningen baserad på filtrerad intern och extern data.*

Testpunkt\Varv	1	2	3	4	5
1	0	85	41	11	8
2	3	31	32	25	19
3	1	29	12	10	75
4	10	2	9	11	29



Figur 13. *Graf över hur felet växer med avståndet roboten kört då positionsuppskattningen baseras på filtrerad data.*

4.4.2 Mätdata från testning av MAC-lagret

Tabell 4 och Figur 14 visar den effektiva kapaciteten hos positioneringssystemet, det vill säga antalet lyckade avståndsmätningar per tidsenhet. Det framgår att antalet fel växer drastisk då flera noder som saknar MAC-lager mäter avstånd samtidigt. Det syns även att det tar längre tid att utföra 10000 mätningar för en nod utan MAC-lager än för den som har det.

Då det inte fanns tillräckligt med robotar att tillgå för att kunna testa MAC-lagret för fler robotar togs beslutet att simulera antalet robotar genom att öka storleken på tidsramen från 200 ms till 800 ms. Effekten av detta blir att antalet positionsbestämningar per sekund minskar från 5 till 1,25, vilket motsvarar 5 robotar. Resultatet blev, som förväntat sämre än tidigare, men inom ramen för de mål som sattes upp. Varför *absolut avstånd från referensnod*, i Figur 14, varierar

så mycket har att göra med att det är beroende av när positionsbestämningen utfördes. Detta då positionsbestämningen görs så sällan i jämförelse med när tidsramen var mindre. Om en positionsbestämning utförs precis efter en referenspunkt kan resultatet därför bli lidande om det interna systemet har drivit för mycket.

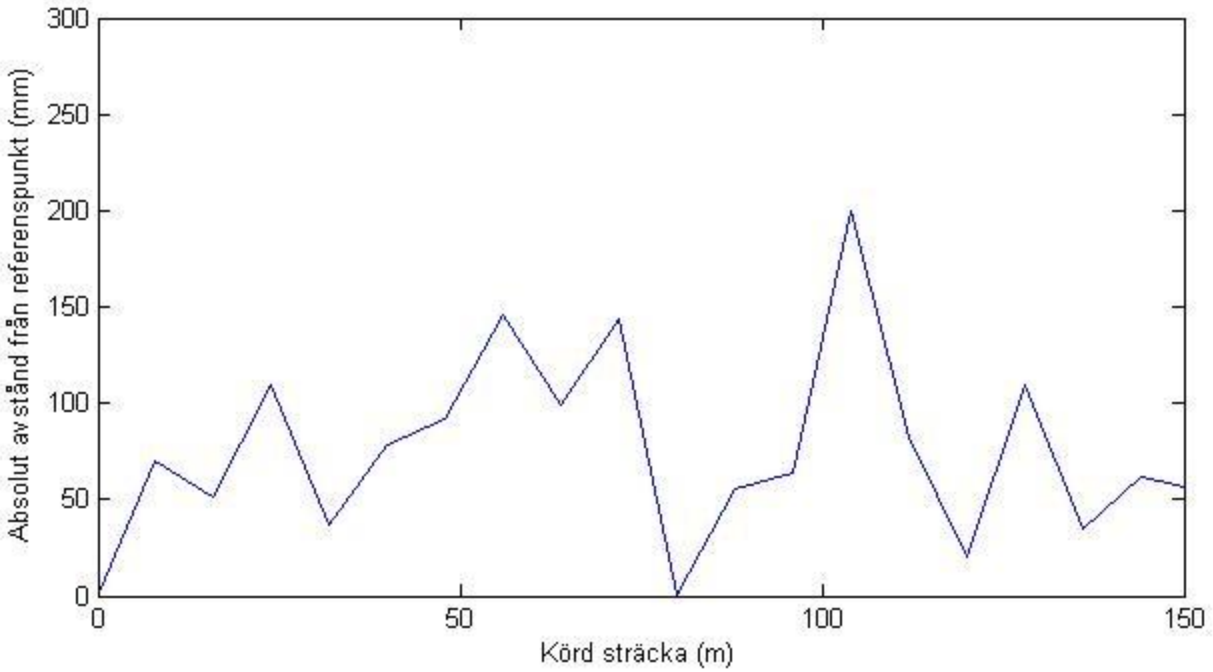
Tabell 4. *Effektiv uppdateringsfrekvens för testfallen presenterade i Kapitel 3.4.2*

Typ av test	Antal mätningar	Antal misslyckade försök	Tid (ms)	Effektiv uppdateringsfrekvens (ggr/s)
En nod utan MAC-lager	10000	30	494508	20
Två noder utan MAC-lager (nod 1)	10000	7281	481588	7.7
Två noder utan MAC-lager (nod 2)	10000	5199	480581	10
En nod med MAC-lager	10000	27	428673	23
Två noder med MAC-lager (nod 1)	10000	63	751233	13
Två noder med MAC-lager (nod 2)	10000	74	753578	13

I tabell 5 presenteras avvikelse från referenspunkten då bilen körde flera varv på banan definierad i karteditorn. Tidsramens storlek var satt till 800 ms vilket motsvarar fem bilar. Medelvärdet på avvikelserna från referenspunkterna är 63 mm.

Tabell 5. *Absolut avvikelse i mm från referenspunkt. Positionsuppskattningen baserad på filtrerad intern och extern data. MAC-lager simulerar närvaro av fem bilar.*

Testpunkt\Varv	**1**	**2**	**3**	**4**	**5**
1	0	70	51	109	37
2	1	56	64	200	83
3	78	92	146	99	144
4	20	110	35	62	55



Figur 14. *Graf över hur felet växer med avståndet roboten kört då positionsuppskattningen baseras på filtrerad data, MAC-lager simulerar närvaro av fem bilar.*

5 Slutsats och diskussion

I detta kapitel diskuteras de resultat som uppnåtts samt vilka slutsatser som kan dras utifrån mätdata i kapitel 4.

Som i inledningen nämns finns det ett behov av positioneringssystem för mobila noder. För att kunna realisera ett effektivt sådant system krävs ett MAC-lager med deterministisk tidsfördröjning. Testresultaten visar att MAC-lagret är ett måste om systemet ska uppfylla ställda krav. Antalet misslyckade positioneringsförsök kommer bara att växa i takt med att flera noder ansluter till systemet. Det som inte framgår i tabell 4 är den kvantitativa fördelningen av avståndsmätningar till varje ankare, det kan mycket väl vara att antalet lyckade avståndsmätningar till enskilda ankare inte är tillräcklig för den mobila noden att bestämma sin position. Det implementerade MAC-lagret har en deterministisk tidsfördröjning. Det har även visats indirekt att det har *self-** egenskaper, då mer avancerade testfall utgår från att nätverket är stabiliserat.

Ursprungligen bestod positioneringssystemet enbart av intern data från takometer och styrvinkel, med viss inverkan av magnetometern. Eftersom att det inte går att kalibrera styrvinkeln exakt resulterar detta i att robotens position kommer att driva mer och mer åt ett håll beroende på hur långt den kört. Detta syns tydligt i testfall 1 då man enbart förlitar sig på intern data. Robotens position drev då ca tre meter för varje varv. Att enbart basera positioneringen på intern data ställer alltså till problem.

Det gjordes också ett test då robotens position endast baserades på avståndsmätningarna, testfall 2. Detta fungerade över förväntan och testresultaten blev väldigt bra. Vad som inte syns i testresultaten är dock att för vissa områden på kartan märktes det att roboten inte följde den linje som var avsedd. Detta berodde troligen på störningar i form av reflektioner från ett glasparti vid testplatsen, vilket försämrade mätresultatet. Att basera positioneringssystemet enbart på extern data gör systemet väldigt känsligt för störningar.

Genom att kombinera intern och extern data med ett Kalmanfilter erhålles ett system som inte är lika känsligt för störningar som i det externa fallet, och som kompenserar för de fel som uppkommer i det interna fallet. Resultatet i Figur 12 visar att de krav som satts upp i kravspecifikationen, i kapitel 1.3.1, uppfylls. De spikar som kan skådas i Figur 13 vid 10 och 110 meter har att göra med störningar i magnetometern som troligtvis beror på tillfälliga magnetfält vilket orsakas av elektroniken som sitter på roboten.

Detta projekt visar, för första gången[2], att positionsbestämning med deterministisk tidsfördröjning för flera mobila noder i realtid kan uppnås.

[2] Konversation med Elad Schiller

5.1 Potentiell vidareutveckling

Projektet har skapat en gedigen grund för att enkelt kunna vidareutveckla systemet med ytterligare funktionalitet. Med funktioner som kollisionsfritt MAC-lager och fungerande filter har ett robust system skapats. Tack vare de funktioner som skapats under projektet kan man nu lätt göra vidarutvecklingar som utan detta projekt hade tagit lång tid. Nedan följer fyra områden för potentiell vidareutveckling.

5.1.1 Cooperative localization

Cooperative localization [14] är en teknik som bygger på de noder som befinner sig i systemet inte enbart gör avståndsmätningar till de fasta ankare som finns utan även till alla mobila noder. Fördelen med detta är till exempel att då noder befinner sig utom räckhåll från ett ankare istället kan göra en avståndsmätning till en närliggande nod. Det ger även fördelen att två noder kan få reda på deras relativa avstånd från varandra utan fasta ankare.

5.1.2 Frequency-division multiple access

I dagsläget finns det möjlighet att låta RCM-modulerna fungera på olika frekvensband. Med tre ankare inställda på olika kanaler är det möjligt att låta Gulliver-robotarna turas om mellan frekvenser inom samma tidslucka. I teorin innebär detta, i kombination med nuvarande MAC-lager, tredubbling av kapaciteten. Dock saknar existerande *firmware* stöd för tillräckligt snabbt kanalbyte. Med uppgradering av *firmware* blir det möjligt att få RCM modulerna att byta kanaler på några millisekunder[3], vilket öppnar vägen för nya implementationer av MAC-lager.

5.1.3 CRE

Coarse Range Estimate (CRE) används för att mäta avstånd till ankarna med RCM:en med $O(n)$ istället för $O(n^2)$ som fås om man använder sig av det nuvarande sättet att göra avståndsmätningar, detta då alla noder endast behöver göra en avståndsmätning var för att en nod ska få en uppskattning av avståndet till alla andra noder. Anledningen att CRE har lägre komplexitet är att endast signalstyrka används för att uppskatta avståndet. Detta gör att en RCM kan skicka ut en puls och sedan kan alla närliggande noder räkna ut avståndet till RCM:en. När sedan alla ankare har skickat ut en puls vet alla noder var de är. Ett problem med CRE är dock att det endast fungerar bra på korta avstånd. Därför bör man istället använda sig av båda PRE och CRE för att sedan filtrera de båda signalerna med ett Kalmanfilter. Detta skulle snabbt kunna implementeras men det går inte att använda CRE i den nuvarande versionen av RCM:en, utan endast PRE.

5.1.4 Multivarianta filter

Med multivarianta filter går det att få ett mer exakt filter. Detta på grund av att avvikelsen i både x- och y-led används för att räkna ut normalfördelningen för positionen. Utöver detta krävs även en vinkel för att kunna representerade den tvådimensionella normalfördelningen helt och hållet. När tvådimensionella normalfördelningar används går det att använda sig av olika standardavvikelser i sidled och i färdriktning, vilket är en fördel eftersom det oftast inte är

[3] Konversation med Brandon Dewberry, Time Domain

samma mätfel på till exempel styrvinkeln och antalet varv hjulen har roterat. Även vid filtrering av en mätning från RCM:en och senaste uppskattade positionen av bilen har multivarianta filter en fördel. Detta då avståndsmätningens normalfördelning egentligen är torusformad med centrum på RCM:en. Denna torusformade normalfördelning kan approximeras med en oändligt lång, lika bred normalfördelning just i den punkt där bilen befinner sig.

Under projektet har delar av detta filter utvecklas men eftersom det visade sig att det skulle bli för prestandakrävande för hårdvaran lades det ner. Med bättre algoritmer och hårdvara kan detta dock bli aktuellt i framtiden.

5.2 Tillämpningar

Det finns en rad tillämpningar för det positioneringssystem som tagits fram. Bland annat kommer systemet att integreras i Gulliverprojektet som startades hösten 2011 på Chalmers Tekniska Högskola. Dess målsättning är att tillhandahålla en smidig och billig testmiljö för forskning inom intelligenta transportsystem. Projektet har till syfte skapat ett toppmodernt och effektivt system för testning. Detta testsystem innefattar ett flertal miniatyrfordon, så kallade Gulliverrobotar, och är ett mångfasetterat system för att skapa prototyper till- och demonstrera nya fordonsystem. [2]

Kandidatgruppen har blivit kontaktade av *Time Domain*, företaget bakom RCM:erna, och de är intresserade av implementera en MAC-algoritm mycket lik vår i sin hårdvara. Det diskuterades även fördelar med filter och hur dessa kan förbättra prestandan på RCM:erna.

Systemet kan även användas för att höja säkerheten i miljöer där tunga maskiner används, till exempel i gruvor. Om alla tunga maskiner samt alla arbetare är utrustade med RCM:er kan systemet varna så att ingen blir klämd eller överkörd.

Referenser

[1]: Moore, P.; Crossley, P; “GPS applications in power systems. I. Introduction to GPS” Power Engineering Journal, vol.13, no.1, pp.33-39, februari 1999

[2]: Pahlavan, M; Papatriantafilou, M; Schiller, E; “Gulliver: A Test-bed for Developing, Demonstrating and Prototyping Vehicular Systems.” Proceedings of the 9th ACM International Workshop on Mobility Management & Wireless Access, MOBIWAC 2011, October 31-November 4, 2011, Miami Beach, USA, ss. 1-8

[3]: Bishop, G; Welch, G; “An Introduction to the Kalman Filter.”, Chapel Hill: University of North Carolina at Chapel Hill, 2004

[4]: Loera, J. A.; Rambau, J.; Santos, F., “Triangulations”. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010

[5]: National Research Council (U.S.). Committee on the Future of the Global Positioning System, National Academy Of Public Administration, “The Global Positioning System: A Shared National Asset : Recommendations for Technical Improvements and Enhancements”, National Academies Press, 1995

[6]: Kalman, R. E.; “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45, Mars 1960

[7]: Leone, P; Papatriantafilou, M; Schiller, E.M; Zhu, G; “Chameleon-MAC: Adaptive and Self-⋆ Algorithms for Media Access Control in Mobile Ad Hoc Networks”, Stabilization, Safety, and Security of Distributed Systems - 12th International Symposium, SSS 2010, September 20-22, 2010, New York, ss. 468-488.

[8]: Zafer Ðsahinoæglu, Sinan Gezici, Ismail Güvenç; “Ultra-wideband positioning systems : theoretical limits, ranging algorithms, and protocols.” Cambridge University Press, 2008.

[9]: Dahlgren, E; Grundén, J; Gunnarson, D; Holtryd, N; Khazal, A; Swantesson, V; “Gulliver En plattform för testning, utveckling och demonstration med hjälp av miniatyrfordon.”, Göteborg: Chalmers tekniska högskola. (Kandidatarbete inom Institutionen Data- och informationsteknik), 2012

[10]: Råde, L; Westergren, L; ”Mathematics Handbook for Science and Engineering”, Studentlitteratur AB, Lund 2004 EJ FORMATERAD!!!

[11]: Gouda, M; McGuire, T; ”Accelerated heartbeat protocols.” In Proceedings of the International Conference on Distributed Computing Systems, 1998.

[12]: Herman, T; Zhan, C; “Best Paper: Stabilizing Clock Synchronization for Wireless Sensor Networks.”, Stabilization, Safety, and Security of Distributed Systems: Lecture Notes in Computer Science vol. 4280/2006 ss. 335-349, 2006

[13]: Mustafa, M; Papatriantafilou, M; Shiller, E; Tohidi, A; Tsigas, P; “Autonomous TDMA Alignment for VANETs.” Kommer att publiceras i “Vehicles and Technology” under hösten 2012.

[14]: Wymeersch, H.; Lien, J.; Win, M.Z.; , "Cooperative Localization in Wireless Networks," *Proceedings of the IEEE* , vol.97, no.2, pp.427-450, Feb. 2009

Bilagor

A Bidragsförteckning:

Namnen är sorterade efter bokstavsordning och inte prestation.

Bidrag till utvecklingsområden

- Genomförande
    - TDMA - Master/Slave baserad
      Alexander, Karl, Timur
    - TDMA - “TimeSlotManager”
      Alexander, Karl, Timur
    - RCM-MAC
      Timur
    - MICAz-MAC
      Alexander, Karl, Timur
    - Kalmanfilter
        - Skapa Kalmanfilter med singelvarianta normalfördelningar.
          Alexander, Karl
        - Skapa Matlab- och C-kod som räknar på multivarianta normalfördelningar i 2D.
          Tobias

Bidrag till problemlösning, syntes och analys

Alla ingående medlemmar i gruppen har bidragit inom olika områden vilket gör det svårt att urskilja och bedöma individuella bidrag.

Korrigering av rapport efter opponering

Alexander, Karl, Timur

Huvudansvarig författare av avsnitt

Alexander:

2.4, 2.4.1, 2.6, 2.6.1, 3.3, 4.2, 5

Karl:

1, 1.1, 1.2, 1.2.1, 1.2.2, 1.2.3, 1.3, 2, 2.1, 2.2, 2.3, 3.2, 3.4, 3.4.1, 4, 4.4.1, 5.3

Timur:

1.2.4, 2.4.2, 2.6.2, 3.4.2, 4.3, 4.4.2, 5.2.2

Tobias:

1.2.5, 2.5, 3.1, 4.1, 5.2, 5.2.1, 5.2.3, 5.2.4

Muntliga redovisningar

Mittredovisning: Karl, Timur
Slutredovisning: Alexander, Tobias
Opponering: Karl, Timur

B Vidarebefordrad kunskap

I kandidatarbetet har det varit en viktig del att samarbeta med forskningsprojektet Gulliver samt forskningsprojektet Co-opnet och att införliva projektets arbete till dessa. För att möjligöra ett sådant samarbete har kandidatgruppen haft regelbundna veckomöten där kunskap och information har utbytts med forskningsprojektens medlemmar. Utöver dessa möten har gruppen även förklarat och visat hur kandidatarbetets nyskapade system fungerar. Genom systemet har gruppen positivt bidragit till och påverkat forskningsprojektets framtagning. Detta i bemärkelse att gruppen har undersökt och införskaffat kunskap samt hittat flertal tekniska lösningar på existerande samt nyfunna problem, vilket har underlättat för projektets mål.

Vid de regelbundna mötena har bland annat Henk Wymeersch, Gabriel Garcia, Elad Schiller och Benjamin Vedder varit närvarande. Efter kandidatarbetets avslutande hölls ett överlämningsmöte där representanter för de olika forskningsprojekten deltog. Blockscheman över de algoritmer som skrivits har skapats för att ge en enkel överblick över systemet, dessutom är all kod kommenterad för att underlätta framtida vidareutveckling.

C Förändringar gjorda efter opponering

Projektgruppen vill tacka DATx02_12_43 för den tiden de lagt på att skapa en bra och noggrant utförd opponering. Vi tog till oss många av åsikterna och gjorde ändringar i vår rapport. Nedan följer dock en lista på de punkter vi inte håller med om, samt kommentarer på varför.

- Det kommer inte att ges utförligare förklaringar under huvudrubrikerna på vad varje kapitel kommer att behandla. Vi anser att nuvarande text att fullt tillräcklig.
- Kap 1.4 “Rapportens upplägg” kommer att behållas då vi anser att det underlättar för läsaren att hitta relevant information i rapporten.
- Vi väljer att inte presentera Gulliverprojektet mer ingående då detta inte är enda möjliga tillämpningen till systemet utan endast plattformen som vi byggt systemet på.
- Eventuella felkällor till resultatet behandlas i diskussionen och kommer inte att tas upp i teorikapitlet.
- Det kommer inte att undersökas eller diskuteras huruvida annan hårdvara hade förbättrat resultatet. RCM:erna är *State of the art* enligt Elad Schiller och Henk Wymeersch.
- Referenserna är inte formaterade efter Vancouversystemet då det är IEEE standarden som används.