Positioneringssystem för segelbåtstävlingarna ’Gotland Runt’ och ’Nokia OOPS Cup’

MANDUS ENGMAN

KTH Numerisk analys
och datalogi

Examensarbete
Stockholm, Sverige 2005

TRITA-NA-E05104

Numerisk analys och datalogi
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Positioneringssystem för segelbåtstävlingarna ’Gotland Runt’ och ’Nokia OOPS Cup’

MANDUS ENGMAN

TRITA-NA-E05104

Examensarbete i datalogi om 20 poäng
vid Programmet för datateknik,
Kungliga Tekniska Högskolan år 2005
Handledare på Nada var Björn Eiderbäck
Examinator var Stefan Arnborg

Positioneringssystem för segelbåtstävlingarna 'Gotland Runt' och 'Nokia OOPS Cup'

Sammanfattning

Detta examensarbete går ut på att utveckla ett system för positionering av segelbåtar under tävlingar, bl.a. Gotland runt och Nokia OOPS Cup. Syftet med projektet är att tävlingsledning och publik ska få en bättre överblick över båtarnas positioner och rutter under tävlingens gång. Eftersom befintliga positioneringssystem är dyra och otympliga eftersöks ett system byggt på en hårdvara som enkelt och smidigt går att installera på båtarna samtidigt som lösningen måste vara billig.

Eftersom den växande mobiltelefonmarknaden och den snabba tekniska utvecklingen har gjort att priserna på terminaler pressats samtidigt som nya tekniska funktioner medför ökat användningsområdet, kommer system att byggas på en mobiltelefon.

Initialt var en terminal av typ Motorola A920/A925 med inbyggd GPS-sändare aktuell men denna byttes senare till en terminal av typ Nokia 6600 tillsammans med en extern GPS-enhet.

Examensarbetets fokus har lagts på att studera de olika utvecklings- och exekveringsmiljöer de aktuella terminalerna erbjuder för att få en förståelse för vilken funktionalitet som stöds av vilken miljö. Eftersom viss för systemet nödvändig funktionalitet strider mot exekveringsmiljöernas inbyggda säkerhetsmodell har mycket studier och arbete lagts på att finna en lösning på detta.

En slutgiltig systemlösning har utvecklats i javamiljön MIDP 2.0 på terminalen Nokia 6600, där GPS-data hämtas över bluetooth från en extern GPS-enhet.

Systemet användes skarpt på segelbåtstävlingen Gotland Runt sommaren 2004. På 19 av de 252 startande båtarna var systemet installerat och resultatet var mycket bra.

Position tracking for the yacht races of 'Gotland Runt' and 'Nokia OOPS Cup'

Abstract

This document describes the development of a system for yacht position tracking. The system, if successful, is to be used in the yacht races "Gotland Runt" and "Nokia OOPS Cup". The main purpose of this project is letting the audience and staff get a better view of the competitors positions throughout the race.

Tracking systems currently existing on the market are often expensive and at the same time ungainly and therefore a solution that is cheap and easily can be installed on the yachts is of great interest.

The rapid growth of the market of cellular phones combined with the technical development have made cellular phones an interesting hardware to use for tracking systems because of the drop of prices and new interesting features.

Initially the project solution was based on a terminal of the type Motorola A920/A925 with an internal GPS-sender, but this was later exchanged with a terminal of type Nokia 6600 combined with an external GPS-sender.

Much effort has been laid on the study of the numerous different developer and execution environments offered by the different terminals to get an understanding of what feature is supported by which environment. The security model of the environment does not allow some essential features of the tracking system, such as sending SMS-messages continuously without confirmation by the user, and therefore finding a solution has been a main effort throughout the project.

A solution was finally developed in the MIDP 2.0 environment of the terminal Nokia 6600. Because the lack of an internal GPS-sender, the solution was completed with a cordless Bluetooth GPS.

The tracking system was successfully used during the race of 'Gotland Runt' during the summer of 2004.

Innehållsförteckning

1 Inledning 1
1.1 Uppdragsgivare 1
1.2 Bakgrund 1
1.3 Uppgift 1
1.4 Lösningsidé 2
1.5 Avgränsningar 2
1.6 Samarbete 2
2 Trådlös kommunikation 3
2.1 GSM, Global System for Communications 3
2.2 GPRS, General Packet Radio Service 4
2.3 3G, det trådlösa internet 5
2.4 Konkurrensen om bandbredden 6
3 Java och mobiltelefonen 8
3.1 Utvecklingen av Java 8
3.2 Säkerhetsmodellen i MIDP 11
4 GPS, Global Positioning System 13
4.1 Historik 13
4.2 Systemets uppbyggnad 13
4.3 Så fungerar GPS 13
4.4 Tidmätningen 14
4.5 Hur fastställs satellitens exakta position? 16
4.6 Fel som kan påverka en exakt positionering 17
5 Hårdvara och plattform 18
5.1 Specifikation för Motorola A920, A925 18
5.2 Specifikation för Nokia 6600 18
5.3 Specifikation EMTAC CRUXII/BTGPS 18
6 Lämpliga teknologier 19
6.1 Serverkommunikation 19
6.2 Terminal Motorola A920/A925 19
6.3 Terminal Nokia 6600 med extern GPS 20
6.4 Analys och metodbeskrivning 21
7 Lösningsförslag 26
7.1 Trådar 26
7.2 Systemparametrar 26
7.3 Användarvyer 27
7.4 Systemarkitektur 28
8 Utvärdering och slutsats 31
Referenser 32
Bilaga 1 33
Publicitet 33

Bilaga 2 35
Specifikation från NMEA 35

1 Inledning

1.1 Uppdragsgivare

Detta examensarbete är ett samarbete mellan flera parter, NetLight Consulting AB, Oracle, KSSS och Nokia.

NetLight Consulting AB är ett konsultbolag specialiserat på mobila lösningar och systemintegration.

Oracle är ett multinationellt mjukvaruföretag med affärssystem som ett av deras största verksamhetsområden.

KSSS är Kungliga Svenska Segel Sällskapet som årligen håller i tävlingen Gotland runt.

Nokia är en multinationell tillverkare av mobiltelefoner och har flera egna segelbåtstävlingar under namnet Nokia OOPS Cup.

1.2 Bakgrund

Segelbåtstävlingar är ingen ny företeelse och utveckling har gått mot att göra tävlingarna mer publikvänliga genom att öka överblicken för publik, media och sponsorer av de tävlande båtarna under loppets gång. Publiken ska inte längre leva i total ovisshet tills dess att båtarna når mållinjen.

Några av de system som existerar idag är radiobaserade transpondersystem. Dessa är stora, otympliga och dyra, vilket medför att de är besvärliga att installera och tar upp stor plats på de annars avskalade tävlingsbåtarna. Det är även en stor kostnad att låta dem installeras på samtliga båtar. Under tävlingen “Gotland Runt” är det endast några av de större båtarna som är utrustade med transpondrar.

Marknaden för mobiltelefoner har vuxit lavinartat de senaste åren samtidigt som den tekniska utvecklingen tagit ett stort steg framåt. Tack vare utvecklingen av handdator-liknande telefoner med hög prestanda och stor skärm får programvaruutvecklare möjlighet att skapa användbara applikationer inom nya användningsområden för en enormt stor marknad. Ett sådant användningsområde är att med hjälp av GPS-teknik ta fram koordinater för var en terminal befinner sig för att t.ex. kunna markera den på en karta.

1.3 Uppgift

Uppdraget i detta projekt går ut på att utveckla ett positioneringssystem för segelbåtar. Systemet ska utvecklas för drift på konsumenthårdvara, d.v.s. vanliga mobiltelefoner, och ska kontinuerligt skicka positionsdata till en server. Systemets ska vara enkelt att starta och ska klara sig utan underhåll under tävlingens gång.

1.4 Lösningsidé

Lösningen för exjobbet går ut på att kontinuerligt skicka positionsdata från samtliga båtar för central lagring. Centralt kan sedan en mängd tjänster erbjudas för att publik och tävlingsledning under en segelbåtstävling ska få bättre översikt över de tävlande båtarnas positioner och rutter. Systemet erbjuder t.ex. möjlighet att visa ständigt uppdaterade positioner på en storbildsskärm i målområdet, möjlighet att surfa in på en WAP-sida med tillfälligt resultat eller möjlighet till prenumeration av SMS eller MMS-meddelande då en viss båt passerar en viss boj.

1.5 Avgränsningar

Detta examensarbete avgränsas till att endast innefatta utveckling på terminalsidan, d.v.s. all den logik som ska utvecklas för mobiltelefonen. Den logik på serversidan som tar emot positionsdata från mobiltelefonerna ligger utanför detta exjobb och utvecklas av Oracle.

1.6 Samarbete

Under projektets gång har bilden av uppdragsgivare och samarbetspartner förändrats något.

I projektets inledning stod en teleoperatör som projektets uppdragsgivare och samarbetspartner i den mening att de skulle bidra med den hårdvara som skulle användas. Mjukvaruföretaget Oracle var den andra samarbetspartnern med uppgift att ansvara för utvecklingen på serversidan där positionsdata tas emot från LiveTracker, det system som utvecklats i detta exjobb. Oracle ansvarade även för presentationen av de kart- och meddelandetjänster som slutanvändaren kunde använda.

Sex veckor in i projektet då jag fortfarande inte erhållit någon testutrustning från den första uppdragsgivaren fick jag på omvägar veta att uppdragsgivaren valt att dra sig ur projektet. Oracle visade sig dock intresserade av att fortsätta projektet och undrade om även jag kunde tänka mig det, trots de förändrade premisser som det nya samarbetet skulle innebära. Ny samarbetspartner blev Nokia som kunde erbjuda den hårdvara som behövdes.

2 Trådlös kommunikation

2.1 GSM, Global System for Communications

Global System for Communications är en globalt accepterad standard för digital trådlös kommunikation. GSM är namnet på den standardiseringsgrupp som grundades 1982 med syftet att skapa en gemensam europeisk standard för mobil telefoni.[2][9]

2.1.1 Historik

Konceptet med mobiltelefoni är att använda lågspänningssändare där frekvenser kan återanvändas inom ett geografiskt område. Idén med ett cellbaserat radionät föddes i Bell Labs, USA, i början av 1970-talet, men det var i norra Europa som det först introducerades kommersiellt 1981 under namnet, Nordic Mobile Telephone (NMT).

I början av 1980 var de flesta telefonsystemen analoga och inte digitala som dagens system. Ett stort problem med det analoga systemet var att hantera det snabbt växande kapacitetsbehovet på ett kostnadseffektivt sätt. De klara fördelarna med ett digitalt system blev mer och mer tydliga såsom lättare signalering, mindre chans för störningar mellan signaler och ökad möjlighet att hantera ökad kapacitetsefterfrågan.

Genom hela utvecklingen av den mobila telekommunikationen har mängder av olika system skapats utan stöd av en standardiserad specifikation. Detta gav upphov till stora problem med kompatibilitet mellan systemen och utvecklingen av den digitala radiotekniken. För att råda bot på dess problem grundades GSM.

Från 1982 till 1985 diskuterades det om GSM skulle bygga på analog eller digital teknik men efter många tester enades man om en digital grund.

2.1.2 Nätverket

GSM erbjuder rekommendationer, inte krav. GSM-specifikationen definierar funktioner och gränssnittskrav i detalj men ger inga direktiv angående hårdvaran. Anledningen är att begränsa utvecklare så lite som möjligt men ändå göra det möjligt för operatörer att köpa utrustning från olika leverantörer.

GSM-nätverket är uppdelat i tre större system, switchsystemet (SS), basstationssystemet (BSS) och operations- och supportsystemet (OSS).

GSM-nätverket är uppdelat i celler som sköter signalhanteringen. Varje cell täcker in en viss geografisk yta och använder en bestämd frekvens. Frekvensen delas upp i åtta tidsluckor där varje tidslucka kan användas för kommunikation med en mobil slutanvändare, vilket innebär att varje cell har kapacitet att hantera maximalt åtta samtidiga samtal.

När en telefon ska koppla upp till nätverket för att sända data letas en ledig frekvens inom bandet upp och låses så länge samtalet håller på.

2.1.3 SMS

SMS står för Short Message Service och är en integrerad del av GMS specifikationen. SMS är korta textmeddelanden som skickas över kontrollkanalen och användes ursprungligen endast av nätverket för att skicka korta meddelanden till en mobiltelefon. I senare versioner av GSM specifikationen ingår också stöd för att skicka meddelanden från mobiltelefonen och det är på detta sätt som det har blivit en tjänst som är allmänt känd bland användare av GSM.[6]

Användandet av SMS som kommunikationsmedel mellan mobila användare har fullkomligt exploderat under början av 2000 talet. Anledningen till att SMS har blivit så populärt är att det är enkelt och billigt för användaren och nästan utan merkostnad för nätverksoperatörerna, då SMS skickas över den annars outnyttjade kontrollkanalen.

SMS är en GSM standard för att skicka meddelanden från en Short Message Entity (SME), d.v.s. en sändande enhet, till en Mobile Station (MS), d.v.s. en mobil utrustning med tillhörande SIM-kort och en Short Message Service Center (SMSC) som mellanhand.

Ett SMS-meddelande som är skickat mellan olika mobiltelefoner är grovt sett uppbyggt av två delar, ett huvud och en kropp. I huvudet finns information som bl.a. specificerar hur meddelandet ska hanteras, vart meddelandet ska ta vägen och vilken prioritet det har. Kroppen utgörs av det data som ska skickas och är begränsad till 140 bytes.

En av de stora fördelarna med SMS är att det inte är nödvändigt för båda enheterna att vara tillgängliga i nätverket vid samma tillfälle. Om en enhet inte är tillgänglig vid tidpunkten för försändelsen lagras meddelandet i SMSC och skickas vidare till mottagaren då den åter blir tillgänglig i nätverket.

2.2 GPRS, General Packet Radio Service

GPRS är en paketbaserad databärare för GSM och erbjuder GSM operatörer en möjlighet att snabbt starta den förutspådda massmarknaden för trådlösa tjänster.

GPRS erbjuder paketsändning från mobilterminalen och uppåt genom det befintliga GSM-nätet. Med GPRS erbjuds den mobila användare samtidigt utnyttjande av fler tidsluckor hos GSM-nätets celler. Det innebär att en telefon som stöder GPRS kan skicka upp till åtta gånger mer data per tidsenhet än en telefon som inte gör det, detta gäller om alla tidsluckor skulle vara lediga. Uppkoppling sker nästan momentant då data ska skickas, och stängs då överföringen är klar. Användaren debiteras för sänd datamängd och inte efter uppkopplingstid. Först när data verkligen skickas belastas således nätet och överföringshastigheten blir beroende av hur mycket bandbredd som finns ledigt för tillfället vilket innebär att nätet kommer utnyttjas väldigt effektivt.

Alla utbredda datakommunikationsprotokoll som t.ex. TCP/IP stöds av GPRS vilket innebär att det är möjligt att från en mobil terminal med GPRS, var som helst i hela världen koppla upp sig mot en befintlig datakälla.

Introduktionen av GPRS är ett stort och viktigt steg i utvecklingen av GSM-nätverket mot 3G-kapacitet.

2.3 3G, det trådlösa internet

International Telecommunications Union, ITU, som är det internationella FN-organ som hanterar utnyttjandet av frekvensspektrat, har tagit fram två standarder för 3G, UMTS och IMT-2000.

- IMT-2000, International Mobile Telecommunications, som även kallas ITU-2000 är internationellt och använd främst i Japan.
- UMTS, Universal Mobile Telecommunications System, används i Europa.

2.3.1 UMTS

Den mest kända funktionen med UMTS är större bandbredd och därmed ökad överföringshastighet. Med ökad överföringshastighet blir en rad nya tjänster möjliga, som videosamtal och andra tjänster som kräver snabb överföring av data. Ofta finns den efterfrågade informationen på internet vilket kräver en effektiv TCP/IP hantering i UMTS nätet.

Förutom bandbredden är en annan stor skillnad mellan UMTS och andra existerande mobila nätverk som t.ex. GSM att UMTS erbjuder dynamisk förhandling av vissa inställningar hos radiobäraren som i andra nätverk är helt statiska. Dessa inställningar styr t.ex. tillfällig bandbredd, förseningar vid överföring och korrigering för eventuella datafel.[8]

GSM låser alltid en enda ledig frekvens för dataflödet så länge ett samtal pågår medan UMTS upptar flera frekvenser. För att signaler som kan komma att uppta samma frekvens samtidigt inte ska störa ut varandra använder UMTS en funktion som kallas Code Division Multiple Access, CDMA. Med en kod särskiljs de olika sändarna så att varje mottagare bara tar emot signaler från rätt sändare.

Celler hos UMTS klarar fler användare, ca 20 stycken, jämfört med celler hos GSM som bara klarar 8 samtidiga användare.

2.4 Konkurrensen om bandbredden

Det sker ständigt en ökning av radiotrafik runt omkring oss. Det är inte bara det ökande antalet mobiltelefoner eller utvecklingen av de nya näten för mobiltelefoni, 3G, utan även det kraftigt växande antal trådlösa nätverk i hem och på företag som bidrar till denna ökning. Samtidigt går utvecklingen mot att fler och fler nya produkter kommunicerar trådlöst med sin omgivning på ett eller annat sätt.

Med en större mängd radiotrafik på en mindre yta blir det problem för alla signaler att få plats på en ändligt stor bandbredd utan att störa varandra så mycket att de blir oläsbara.

Radiovågor är elektromagnetisk strålning precis som ljus och förflyttar sig som ljuset i 300 000 kilometer per sekund. Vågornas intensitet avtar snabbt i takt med att avståndet mellan sändare och mottagare ökar. Dämpningen är ungefär proportionell mot kvadraten av avståndet vilket innebär att intensiteten minskar till en fjärdedel om avståndet till antennen fördubblas.

Uteffekten från en radiosändare styr intensiteten på radiovågen den sänder ut och därmed dess räckvidd. En vanlig mobiltelefon har en uteffekt på maximalt ca 1 watt, ett vanligt lokalt trådlöst nätverk har en uteffekt på maximalt ca 100 milliwatt och en basstation för GSM har en uteffekt på maximalt några hundra watt för stora basstationer i höga master.

Allt eftersom varje ny generation av mobiltelefoni utvecklas har man ökat frekvensen på radiovågorna. Anledningen till detta är att med ökad frekvens komma åt mer bandbredd och därmed större möjlighet att skicka mer data samt att utrymmet på de lägre frekvenserna redan är fullt.

Bandbredd betyder i radiosammanhang det utrymme i hertz mellan den lägsta och högsta frekvensen i bandet. Enligt standarden för trådlösa nätverk, IEEE 802.11b, är den lägsta tillåtna frekvensen 2,4000 och den hösta tillåtna 2,4835 gigahertz. Bandet som utgörs av frekvenserna mellan dessa ändlägen kallas ISM-banden och har således en bandbredd på 83,5 megahertz. ISM-bandet är fritt för alla att använda. 3]

2.4.1 Trådlösa nätverk

Vid högre frekvenser används ofta begreppet mikrovågor. Dessa höga frekvenser inom ISM-bandet används av mikrovågsugnar, trådlösa nätverk (WiFi) och produkter med bluetooth-kommunikation m.m.

2.4.2 GSM

Mobiltelefoner sänder på flera olika frekvenser beroende på modell och många modeller klarar numera att sända på flera frekvensområden. De vanliga frekvenserna för GSM är 900, 1800 och 1900 megahertz och ligger således långt från ISM-bandet som sänder runt 2,45 gigahertz. Uteffekten på en mobiltelefon är maximal när kontakt med en basstation skall etableras, t.ex. vid inkommande samtal, då effekten kan vara maximalt ca 1 watt men är bara ca 1 milliwatt under ett samtal. Avståndet till närmsta basstation påverkar också vilken uteffekt en telefon har. Ett längre avstånd kräver självklart högre uteffekt.

2.4.3 UMTS

UMTS arbetar med frekvenser mellan 2110 och 2170 megahertz och börjar således närma sig de trådlösa nätverkens arbetsfrekvens men de ligger fortfarande tillräckligt långt från varandra för att inte bli störda. Med UMTS ökar däremot bakgrundsstrålningen då det blir tätare mellan basstationerna.

2.4.4 GPS

GPS-satelliter kan sända med hög frekvens eftersom inga hinder existerar. Detta innebär dock att mottagningen försämras radikalt vid eventuella hinder som tak vid inomhusbruk eller vattendroppar vid regnväder.

3 Java och mobiltelefonen

3.1 Utvecklingen av Java

3.1 .1Bakgrund

Allt eftersom antal mobiltelefonanvändare över hela världen ökar kommer mängden telefoner uppkopplade mot internet öka. Det öppnar en enorm marknad för användare, operatörer och applikationsutvecklare. Användare får möjlighet att personalisera sina mobila enheter genom att ladda ner nya applikationer som spel och nyttoprogram, operatörer får fler användare att sälja tjänster till och utvecklare fler utvecklingsuppdrag. Denna växande marknad ställer högre krav på applikationsutvecklings-plattformen än det gjorts tidigare. Java som är ett programmeringsspråk spritt över hela världen underlättar denna utveckling av mobila applikationer och innehåll.

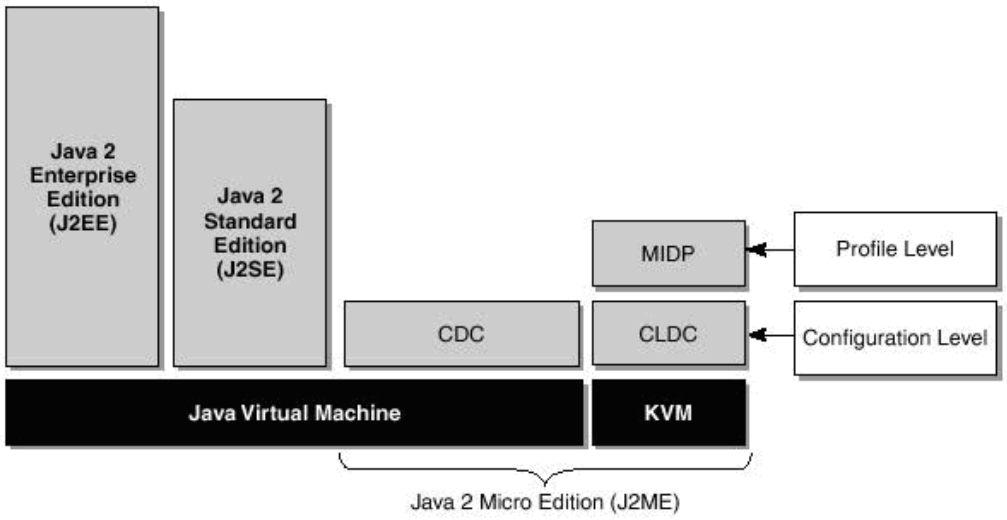
3.1.2 Historik

Java skapades av Sun 1991 för att passa den växande marknaden av applikationer, spel, plug-ins m.m. för olika mjuk- och hårdvaruplattformar. Sun marknadsförde Java som ”write once, run anywhere”, vilket är en bra beskrivning av Javas styrka. Sun baserade Java konceptuellt i två delar, javaapplikationen och javatolken som kallas Java Virtual Machine (JVM). En javaapplikation kan inte exekveras fristående utan kräver en javatolk som översätter koden och exekverar applikationen. Det innebär att en applikation skriven i Java kan användas på flera olika mobila enheter trots att hård- eller mjukvaruplattform skiljer sig.

1999 insåg Sun att ”One size doesn’t fit all” och delade upp Java i tre versioner:

- Java 2 Enterprise Edition (J2EE)
- Java 2 Standard Edition (J2SE)
- Java 2 Micro Edition (J2ME)

Enterprise Edition riktar sig mot helhetslösningar med fokus på applikationer skrivna för serversidan, medan Standard Edition mer riktar sig mot enbart klientsidan såsom applikationer som exekveras på en vanlig pc.[1]



Figur 1. *Javas uppdelning i olika versioner.*

3.1.3 J2ME

J2ME, Java 2 Plattform Micro Edition, blev plattformen för ”mikro” enheter med små processorer och lite minneskapacitet såsom mobiltelefoner och handdatorer.

För att J2ME ska passa en stor mängd produkter kan man i plattformen lägga in en konfiguration och en profil. I konfigurationen definieras en minimal plattform för en familj av produkter med liknande processor- och minneskapacitet. De två typerna av applikationer som kan installeras är beroende av vilken javamiljö som finns i produkten. Två olika typer eller konfigurationer av J2ME finns idag:

- Connected Device Configuration (CDC), i denna miljö används en Java Virtual Machine (JVM). CDC produkter har en minnesstorlek på minst 1 megabyte. Främst handdatorer är uppbyggda på CDC.
- Connected Limited Device Configuration (CLDC), i denna miljö används en kilobyte virtual machine (kVM) som är en lättvikts Java Virtual Machine. CLDC produkter har normalt en minnesstorlek på mellan 128 och 256 kilobytes. De vanligaste CLDC-produkterna är mobiltelefoner och pagers.

3.1.4 PersonalJava och Personal Profile

PersonalJava var en av de första javautvecklingsmiljöerna och används framför allt på handdatorer och handdatorliknande telefoner. Personal Java introducerade funktionalitet som reducerar minnesanvändningen och anpassar applikationer till olika storlekar och upplösningar av displayer och grafiska gränssnitt.

3.1.5 MIDP, Mobile Information Device Profile

MIDP är specialdesignat för mobiltelefoner och handdatorer och erbjuder den kärn-funktionalitet som mobila applikationer kräver med användargränssnitt,

nätverkskommunikation och datalagring. MIDP bygger på en egen javaexekveringsmiljö som minimerar både minnes- och energikonsumtion.

3.1.6 Skillnader mellan PersonalJava och MIDP

PersonalJava bygger på CDC och MIDP är byggt ovanpå CLDC, se kapitel 3.1.3 J2ME.

Applikationer skrivna för PersonalJava överförs till mobila enheter genom kabel från pc medan CLDC/MIDP applikationer kan laddas ner direkt från internet.

Eftersom Javas standardöversättare JVM inte är optimal för enheter med låg minneskapacitet och långsamma processorer utvecklade Sun en lättviktsöversättare, K Virtual Machine (KVM). Denna kräver endast 40-80 kilobytes och kan exekveras på processorer med låg klockfrekvens. PersonalJava använder Javas vanliga översättare, JVM.

PersonalJava har en mer utvecklad applikationsmiljö och kan integrera i större utsträckning med telefonens egen mjukvara än vad CLDC/MIDP kan.

J2ME CLDC använder en säkerhetsmodell uppbyggt av ett antal systemkomponenter som tillsammans säkerställer att endast auktoriserade applikationer kommer åt systemresurser. Javaplattformen använder tre huvudkomponenter, ”the class loader” som laddar och läser in klassfiler, ”the byte-code verifier” som verifierar den översatta bytekoden, och ”the security manager” som handhar säkerheten. [13] Dessa delar har alla en vitig roll och säkerställer att:

- Klasserna är i rätt format
- Endast de rätta klasserna laddas och översätts
- Ej auktoriserade klasser ska inte tillåtas exekverar farliga instruktioner.
- Icke auktoriserade klasser ska inte tillåtas komma åt skyddade systemresurser.

3.1.7 MIDlet

En MIDlet är en applikation som exekveras på en Java-telefon likt en applet som exekveras i en webbläsare eller en servlet som exekveras på en server. En MIDlet laddas ner till en produkt och exekveras ovanpå MIDP-profilen i CLDC. När man skriver en MIDlet kan man välja om den ska vara allmän och portabel genom att använda ett högnivå-API eller produktspecifik, för att utnyttjar produktspecifika funktioner, genom att använda ett lågnivå-API. Teoretiskt kan man exekvera en MIDlet skriven med endast högnivå-API på alla produkter i MIDP-familjen och på de flesta datorer med JVM.

3.1.8 Fördelar med Java

Java medför en öppen utvecklings- och exekveringsmiljö som tillåter applikationer framtagna av tredjepartsutvecklare att laddas ner genom OTA, (eng. ”*Over The Air*”), från internet för att installeras.

För konsumenter innebär det att applikationer kan installeras under en produkts hela livstid och inte bara initialt när produkten köps. Konsumenten kan därmed skräddarsy sin enhet genom att ladda ner nya applikationer och spel när som helst.

För operatörer och företag som erbjuder tjänster och applikationer innebär ökad mängd nerladdningen av konsumenter ökad teletrafik och ökad försäljning av applikationer och eventuella tjänster.

3.2 Säkerhetsmodellen i MIDP

En fungerande säkerhetsmodell är mycket viktigt för MIDPs framtid då den bygger på filosofin att i stort sett vem som helst ska kunna utveckla en MIDlet som vem som helst med en mobiltelefon enkelt kan ladda ner och exekvera. Säkerhetsmodellens uppgift är att skydda användaren mot farliga aktioner en MIDlet annars skulle kunna utföra som t.ex. hämta känslig information från telefonen och skickar den vidare.

Modellen kallas ”sandlådemodellen” då den låter en MIDlet exekvera i en skyddad miljö, en sandlåda, där den kan operera fritt utan att komma åt känslig information eller känsliga funktioner. Känslig information kan t.ex. vara personlig information som användaren sparat i telefonen och känsliga funktioner är t.ex. all nätverkskommunikation eller andra funktioner som kostar användaren pengar.

3.2.1 MIDP 1.0

Säkerhetsmodellen i MIDP 1.0 är uppbyggd så att inga känsliga operationer kan kommas åt överhuvudtaget. Det gör att säkerheten blir stor men användningsområdet för MIDlets blir litet.

3.2.2 MIDP 2.0

I MIDP 2.0 har säkerhetsmodellen förändrats för att öka användningsområdet för MIDlets genom att under kontroll ge dem ökad tillgång till känsliga funktioner. Hela säkerhetsmodellen i MIDP 2.0 bygger på en skyddsdomän, *protection domain,* som styr hur en MIDlet får interagera med sin omvärld och är uppdelad i tillåtna rättigheter, *allowed permissions* och användarrättigheter, *user permissions*. När en MIDlet installeras blir den tilldelad en viss skyddsdomän. [4]

Användarrättigheter

Användarrättigheterna sätts explicit av användaren efter det att en MIDlet är installerad på terminalen och kan ändras när som helst. Dessa rättigheter hanterar hur interaktionen mellan MIDlet och de API som anses känsliga ska se ut genom att för varje sådant API låta användaren styra vad som ska hända då en MIDlet försöker använda det. Beroende på vilken inställning som är gjord för varje känsligt API kan användaren aktivt tvingas att konfirmera när dessa API används. Om användaren väljer att godkänna användandet av ett viss API fortsätter programmet som vanligt i koden men om användaren nekar kastas ett säkerhetsundantag, *securityexception*, som måste fångas för att undvika en programkrasch. Användarrättigheterna kan ha en av tre inställningar, *blanket*, *onshot* eller *session*.

Blanket innebär att en MIDlet ges rättighet att fritt använda det berörda API så länge en MIDlet är installerad utan någon konfirmation av användaren.

Oneshot, innebar att användaren måste konfirmera första gången en MIDlet använder det berörda API men att det sedan kan användas fritt så länge en MIDlet körs.

Session innebär att användaren måste konfirmera varje gång det berörda API används.

Utvecklarens rättigheter

Utvecklarens rättigheter för en MIDlet sätts av utvecklaren och specificerar vilka känsliga API som en MIDlet behöver använda. Dels går det att specificera vilka API som verkligen krävs men man kan också specificera API som är mindre viktiga. Detta gör man genom att

ange MIDlet-Permissions respektive MIDlet-Permission-Opt följt av de klasser man önskar använda i en MIDlets tillhörande JAD-fil. Se exempel nedan:

- MIDlet-Permission: javax.wireless.message.sms.send
- MIDlet-Permission-Opt: javax.wireless.message.sms.recieve

En MIDlet får som sagts sin skyddsdomän satt vid installation på terminalen och eftersom de angivna tillåtna rättigheterna och användarrättigheterna måste vara en delmängd av de rättigheter som finns i terminalens skyddsdomän, styr den hur en MIDlets kommunikation kommer att se ut.

Skyddsdomän

Enligt MIDP 2.0 specifikationen kan en MIDlet ges en av fyra olika skyddsdomäner i skalan från den mest generösa som öppnar upp för användning av alla API till den mest restriktiva som inte ger en MIDlet tillgång till något känsligt API överhuvudtaget. Namnen på de fyra olika skyddsdomänerna är tillverkare *eng. manufacturer*, operatör *eng. operator*, godkända tredjepartsutvecklare *eng. trusted third party* och icke godkända utvecklare *eng. untrusted*, där den förstnämnda är helt öppen och den sistnämnd stängd för kommunikation med alla känsliga API. Tillverkningsdomänen är endast ämnad för utveckling av tillverkare och operatörsdomänen endast ämnad för utveckling av operatören.

En tredjepartsutvecklare kan därmed aldrig erhålla någon annan skyddsdomän än ’godkänd tredjepartsutvecklare’ eller ’icke godkänd utvecklare’.

Signeringsprocessen

Vid signering av en MIDlet krävs ett kodsigneringscertifikat. För en vanlig tredjeparstutvecklare köps detta av en certifikatsmyndighet, CA, *eng. Certificate Authority g*enom att en CSR-fil, *Certificate Signing Request*, skickas till dem. CSR-filen som kan skapas i diverse olika utvecklingsverktyg för MIDP 2.0, innehåller främst information om utvecklaren och används för att skräddarsy det kodsigneringscertifikat som är unikt för varje utvecklare.

Med ett kodsigneringscertifikat är det möjligt att signera obegränsat antal MIDlet under en viss tid då certifikatet innehåller ett sista användningsdatum. Signeringen är möjlig att utföra i de flesta utvecklingsverktyg för MIDP 2.0. Vid signeringen läggs det köpta certifikatet samt krypterad information baserad på den JAR-fil som en MIDlet består av, in som nycklar i form av nya attribut i JAD-filen. Vid installation av en MIDlet i en terminal kommer dessa nycklar att kontrolleras och valideras och kopplas till ett av de rotcertifikat som finns i varje terminal. Bl.a. kommer det kontrolleras att kodsigneringscertifikatet är giltigt samt att JAR-filen är exakt identisk med den vid signeringen då attributen skapades. Fortsättningsvis kontrolleras att de *MIDlet-Permission* attribut i JAD-filen är en delmängd av den skyddsdomän som hör till det rotcertifikat som signeringen gjort anspråk på. Om ett attribut som inte är en delmängd av skyddsdomänen hittas avbryts installationen.

4 GPS, Global Positioning System

4.1 Historik

Att veta exakt var man befinner sig och vilket håll man ska förflytta sig för att nå sitt mål, har genom människans historia varit ett stort problem. Genom tiden har man med hjälp av flera olika teknologier försökt lösa detta problem men alla lösningar har haft stora brister. Till slut bestämde sig U.S. Department of Defense för att den amerikanska militären skulle utveckla ett system som kunde ge exakta positioner över hela världen. Resultatet blev Global Positioning System, GPS. [1] [16]

4.2 Systemets uppbyggnad

GPS använder sig av 25 till 30 satelliter och 5 markstationer. Satelliterna kretsar i en bana ca 20 000 km ovanför jorden med en omloppstid på ungefär 12 timmar. Markstationernas uppgift är att följa satelliterna och hela tiden kontrollera deras hälsostatus och exakta position. En av stationerna, huvudmarkstationen, skickar en rättelse av satelliternas ephemeriskonstanter tillbaks till dem så att de i slutändan kan skicka rätt signaler till alla GPS-mottagare.

4.3 Så fungerar GPS

För att en GPS-sändare ska kunna positionera sig behövs ett antal kända referenspunkter där positionerna i sig är kända och även avstånden mellan dem och sändaren.

Om en referenspunkt är känd liksom avståndet mellan referenspunkten och sändaren, måste sändaren befinna sig på sfären som utgörs av radien lika med det nämnda avståndet.

Vet man att avståndet är 20 000 km mellan sändare och referenspunkt måste sändaren befinna sig på sfären med radien 20 000 km från referenspunkten.

Om en andra referenspunkt är känd och avståndet mellan den och sändaren är 21 000 km, måste sändaren också befinna sig på dess sfär med den nya referenspunkten som centrum och radien 21 000 km.

Eftersom GPS-sändaren måste befinna sig på sfären till både den första och den andra referenspunkten, måste den också befinna sig på den cirkel som utgör skärningen av de både sfärerna.

Om en tredje referenspunkt med tillhörande avstånd är känd, kan det möjliga området för sändarens position till skärningen mellan cirkeln och ännu en sfär minskas. Resultatet blir att de möjliga positionerna för sändaren kommer att utgöras av två punkter, så länge satelliterna inte är för nära varandra. För att avgöra vilken av positionerna som är den riktiga skulle en mätning kunna göras av en fjärde satellit men vanligtvis är en av

positionerna så orimlig p.g.a. att den ligger för långt från jorden eller att den rör sig otroligt snabbt i förhållande till satelliterna, att den kan avfärdas. Däremot är en fjärde mätning viktig av en annan orsak, nämligen tidmätningen.

4.4 Tidmätningen

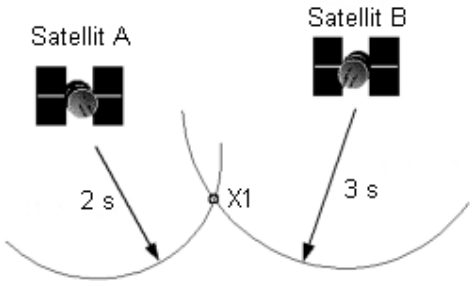
För att räkna ut avståndet mellan GPS-sändaren och en satellit som i hög hastighet svävar runt i rymden, mäts hur lång tid det tar för en signal som skickas från satelliten att nå GPS-sändaren. Genom att multiplicera signalens färdtid med dess hastighet erhålls sträckan. Eftersom signalerna färdas med ljusets hastighet kommer ett mätfel av tiden på en tusendels sekund att resultera i ett mätfel av sträckan på ca 300 km. Därför är det viktift att mätningen blir exakt. I systemet har man löst det genom att både satellit och sändare genererar en unik komplex fyrkantsvåg, som kallas Pseudo Random Code (PRC), med start exakt samtidigt vid ett visst klockslag. Satelliten sänder sin PRC till GPS-sändaren som jämför den med sin egen våg och låter den förskjutas till de matchar perfekt. Genom att mäta förskjutningen blir resultatet den efterfrågade tiden. Pseudo Random Code är en grundläggande del av GPS-tekniken och ser ut som en helt slumpmässigt skapad fyrkantsvåg. Fyrkantskoden är mycket smart och komplex och har flera viktiga funktioner:

- Den är icke upprepande för att jämförelsen inte ska kunna bli fel.
- Den är unik så att varje satellit kan identifieras trots att samma frekvens används av alla satelliter.
- Den är komplex för att inte kunna blandas ihop med andra signaler på jorden.
- Den ska genom filtrering enkelt kunna särskiljas från bakgrundsbrus.
- Den vara läsbar vid låga intensiteter.

För att mätningen ska bli exakt är det som sagt av stor vikt att satellit och sändare startar sina signaler samtidigt. På satelliterna har man löst problemet genom att utrusta dem med atomur som är det mest precisa tidtagningsinstrument som människan än så länge skapat. Av ekonomiska skäl går det inte att utrusta alla GPS-sändaren med atomur, men med några knep har man löst tidtagningen med billiga tidtagningsinstrument utan att förlora atomurens noggrannhet. Genom tre exakta mätningar kan vi räkna ut en precis position för vår GPS-sändare, men med fyra ungefärliga mätningar kan vi eliminera de mätfel som uppstått p.g.a. tidsfel och erhålla den exakta positionen.

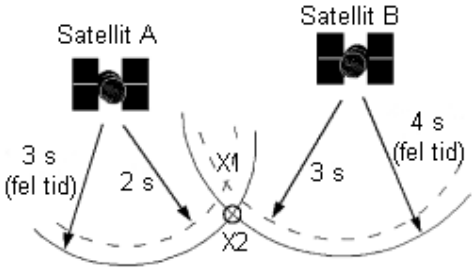
Nedan visas hur elimineringen av mätfelet går till genom ett exempel där GPS-sändarens klocka går en sekund före den exakta universaltiden och därmed ger upphov till ett mätfel på en sekund.

En GPS-sändare befinner sig i X1 och har kontakt med satellit A och B. Avstånden mäter vi för tydlighetens skull i sekunder. Se figur 2.



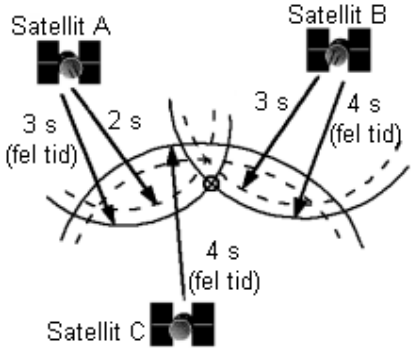
Figur 2. *En GPS-sändare, X1, har kontakt med satellit 1 och satellit 2.*

När satellit och GPS-sändare startar sina PRC-signaler samtidigt på en förut bestämd tidpunkt för att mäta förskjutningen mellan signalerna och därmed tiden, kommer sändaren i själva verket starta en sekund för tidigt p.g.a. dess tidsfel. Följden blir att de uppmätta sträckorna blir en sekund längre än de egentligen är och GPS-sändaren tror då att den befinner sig i X2 i ställer för i X1, se figur 3.



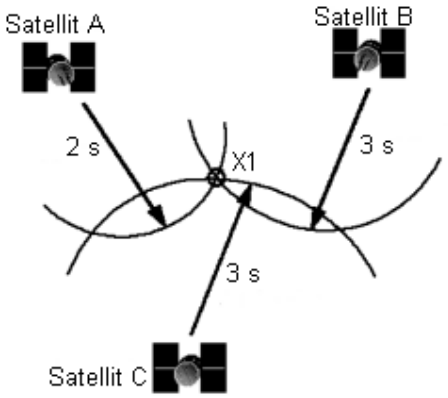
Figur 3. *Ett tidsfel hos GPS-sändaren gör att de uppmätta sträckorna blir felaktiga.*

Om GPS-sändaren använder sig av en tredje satellit kommer även detta uppmätta avstånd att bli en sekund för långt, se figur 4. Det intressanta här är radien till denna tredje satellit, satellit 3, aldrig kommer att kunna skära den punkt där radierna till satellit A och satellit B skär, se figur 4.



Figur 4. *Mätningen till en tredje satellit ger även den en sträcka som är en sekund för lång. Den kommer således inte kunna skära de andra radierna i en gemensam punkt.*

GPS-sändaren hittar sedan sin rätta position genom att korrigera sin interna tid och därmed de uppmätta avstånden till satelliterna tills dess att deras radier skär varandra i en gemensam punkt, X1, se figur 5.



Figur 5. *GPS-sändaren korrigerar sin interna tid tills dess att radierna skär i en gemensam punkt, X1.*

Exemplet ovan gjordes i 2D för tydlighetens skull men fungerar exakt likadant i 3D med data från ännu en satellit.

Genom att utföra en extra mätning med en fjärde satellit som referens, kan således GPS-sändarens exakta position fastställas och dess tid kalibreras med ett atomurs noggrannhet.

4.5 Hur fastställs satellitens exakta position?

I exemplet ovan räknas GPS-sändarens exakta position ut genom enkel algebra där tre punkter i planet samt avståndet mellan dem och sändare, är kända. Hur avståndet mellan sändare och satelliter erhålls förklarades i texten ovan, men hur erhålls satelliternas exakta positioner?

Eftersom alla GPS-sändare behöver kontakt med fyra satelliter för en exakt positionsbestämmelse är antalet satelliter och deras omloppsbanor planerade så att man från alla platser på jorden samtidigt ska kunna få kontakt med minst fem av dem. Tack vare att omloppsbanorna går utanför atmosfären blir det ett mindre problem att hålla satelliterna i sina exakta banor, och det blir därmed även lätt att förutsäga deras framtida positioner.

På marken används marksändare med mycket exakt radarutrustning för att kunna följa satelliternas exakta höjd, position och hastighet. Ephemerisfelet är ett fel som påverkas satelliternas omloppsbanor och uppstår p.g.a. gravitationspulser från månen och solen, och p.g.a. trycket från solens strålning på satelliterna. Ephemerisfelet påverkar inte satelliten särskilt mycket men måste elimineras för att öka mätnoggrannheten i GPS-systemet.

Efter mätning av position och ephemerisfelet skickas dessa nya data tillbaks till satelliterna som använder den vid sin egen utsändning. Satelliterna skickar alltså inte bara ut sin Pseudo Random Code utan även information om sin position, omloppsbana och ephemerisfel. Genom att få sin ephemeriskonstant uppdaterad kan alltså satelliten i förväg räkna ut sin exakta position för varje tidpunkt maximalt 4 timmar framåt i tiden innan den måste bli uppdaterad igen från jorden. Det innebär att satelliterna inte måste ha ständig

kontakt med basstationerna för att kunna veta sin position utan endast behöver bli uppdaterade då och då.

4.6 Fel som kan påverka en exakt positionering

Eftersom satelliterna färdas i vakuum är det enkelt att förutse deras omloppsbanor och därmed framtida positioner. Signalerna som satelliterna sänder ut måste däremot färdas genom atmosfären ner till jordens yta, vilket ger upphov till en hel del fel. Nedan följer några exempel på sådant som kan påverka signalernas gång mellan satellit och sändare:

- Det finns mängder av laddade partiklar och vattenånga i atmosfären som får signalernas hastighet att minska.
- När signalerna närmar sig jordens yta finns det mängder av naturliga och onaturliga föremål som kan få signalerna att ändra riktning, såsom berg och byggda konstruktioner.
- Ephemerisfelet hinner ändra varje satellits bana något innan markstationerna hinner korrigera den.
- Atomuren ger själva upphov till tidsfel, om än väldigt små.

De fel som dessa felkällor ger upphov till korrigeras på flera olika smarta sätt men ger naturligtvis upphov till minskad precision.

5 Hårdvara och plattform

Eftersom de samarbetspartners som ursprungligen var tänkta att tillhandahålla den hårdvara projektet skulle byggas på har ändrats under projektets gång har även den tänkta hårdvaran förändrats. Inledningsvis skulle projektet byggas på en terminal vid namn A920 eller A925 från Motorola med inbyggd GPS. När en partner lämnade projektet och en ny kom ändrades den tänkta hårdvaran till en terminal av typ Nokia 6600 kopplad till en extern GPS.

5.1 Specifikation för Motorola A920, A925

- Operativsystem: Symbian OS v7.0
- Skärm: 65 000 färger
- Skärmstorlek: 41*62 mm, 208*320 pixlar
- Minne: 8 MB inbyggt minne
- Inbyggd GPS
- Endast A925 har inbyggt bluetoothstöd
- Öppna utvecklingsmiljöer: Native C++, Personal Java 1.1.1a, MIDP 1.0.3 [17]

5.2 Specifikation för Nokia 6600

- Operativsystem: Symbian OS v7.0
- Skärm: 65 000 färger
- Skärmstorlek: 41*62 mm, 176*208 pixlar
- Minne: 6 MB inbyggt minne
- Inbyggt bluetoothstöd
- Öppna utvecklingsmiljöer: Native C++, MIDP 2.0 [18]

5.3 Specifikation EMTAC CRUXII/BTGPS

- Inbyggd bluetoothstöd
- NMEA protokoll som stöds: GGA, GSA, GSV, RMC
- Batteritid vid normal drift: 6 timmar [19]

6 Lämpliga teknologier

6.1 Serverkommunikation

Kommunikationen mellan terminal och server kan ske på två sätt.

- Data skickas i ett SMS.
- Data skickas över HTTP .

SMS är en envägskommunikation där ett meddelande skickas iväg utan att ge någon bekräftelse om meddelandet nått sin slutdestination eller inte. I ett SMS går det dock att ange att leveransbekräftelse ska returneras men för att hantera detta krävs ett betydande tillägg till systemet där en egen process lyssnar efter inkommande SMS för att sedan tolka dessa och handla därefter. Det är även omöjligt att förutse hur lång tid det tar innan ett meddelande når sin slutdestination eftersom meddelandet hamnar i en eller flera köer hos någon telefonoperatör innan det behandlas och skickas vidare till rätt slutdestination.

När ett HTTP-anrop används fås alltid ett svar direkt och man vet därmed att anropet kommit fram.

Eftersom systemet är tänkt att skicka positioner då och då spelar det mindre roll om någon enstaka position inte når slutmålet. Vid de tillfällen då det är extra viktigt att positioner når sin slutdestination går det genom att minska tidsintervallet för sändningarna och därmed sända fler meddelande under samma tid att öka chansen att fler positioner kommer fram.

Den andra aspekten till vilken teknik som bör väljas är hur operatörers nättäckning varierar beroende på vald teknik. Till sjöss där systemet är tänkt att användas är GSM-täckningen mer eller mindre befintlig medan GPRS-täckningen i stort sett är obefintlig. SMS skickas genom GSM-nätet och klarar dessutom låg täckningsintensitet, medan HTTP skickas genom GPRS. Det är av stor vikt att systemet kan skicka positioner oavsett var segelbåten befinner sig och därför fås den bästa lösningen genom att skicka data som SMS.

6.2 Terminal Motorola A920/A925

6.2.1.Exekveringsmiljöer

A920/A925 är utrustad med Symbians OS 7.0 som bygger på två exekveringsmiljöer, en javamiljö med stöd för MIDP 1.0 och en lågnivåmiljö, eng. native, som är C++ baserad. Skillnaden mellan dessa miljöer är att man är mycket mer begränsad, se sandlådemodellen, kap 3.2.1 MIDP 1.0, i javamiljön samtidigt som den miljön är mer felförlåtande. Utveckling i lågnivåmiljön kräver att man tar hand om sin egen minneshantering, har mer kontroll på undantag och programflöde medan det detta sköts automatiskt i javamiljön. I javamiljön finns ett gediget klassbibliotek precis som i J2SE men med för J2ME specifika grafiska objekt specialanpassade för telefonutveckling. Sammanfattningsvis kan man säga att utvecklingen i javamiljön går mycket snabbare medan utvecklingen är mycket friare i nativemiljön. Självklart går det snabbare att exekvera kompilerad C++-kod än halvkompilerad javakod som måste tolkas av en javatolk men å andra sidan måste C++-koden kompileras om för varje ny plattform den ska exekveras på medan javakoden direkt kan exekveras på flera plattformar. I detta projekt har exekveringshastighet väldigt liten betydelse eftersom inga tyngre operationer görs.

6.2.2 GPS

A920/A925 har en inbyggd GPS som i nuläget endast går att kommunicera med via ett gränssnitt utvecklat av Motorola för javamiljön. Gränssnittet utgörs av ett klassbibliotek som heter *Location API* och består av några positionsklasser för att kapsla in positionsdata, en positionslyssnare som lyssnar på ny data från GPS-enheten och en klass som hanterar kopplingen till den.

Eftersom A920/A925 stöder MIDP 1.0 och inte den nyare MIDP 2.0 där exekveringen är friare men kontrolleras av rotcertifikat i telefonen, går det inte att skicka SMS från javamiljön. Det går dock att kommunicera utåt med HTTP men det alternativet är redan förkastas eftersom täckningen är markant sämre för GPRS.

6.2.3 Lösning

Sammanfattningsvis är följande känt:

- GPS-data går att hämta från den inbyggda GPS-sändaren genom en MIDlet.
- SMS går att skicka från en applikation skriven i C++ för nativemiljön.

Alltså behövs en en lågninvåapplikation som kan skicka SMS och kommunicera med en MIDlet som i sin tur kan läsa GPS-data. Detta går att lösa med hjälp av sockets som har stöd både hos lågnivåmiljön och hos javamiljön. Eftersom det inte verkar gå att starta en MIDlets från en lågnivåapplikation blir denna lösning problematisk och inte helt uppenbar. För att denna lösning ska fungera måste lågnivåapplikationen fungera som server vad gäller socketkommunikationen. Förfarandet blir som följer:

- Lågnivåapplikation med serversocket startas manuellt av användaren.
- MIDlet startas manuellt av användaren.
- MIDlet får GPS-data från dess positionslyssnare.
- MIDlet skickar GPS-data till serversocket hos lågnivåapplikation.
- Lågnivåapplikation skickar SMS med GPS-data.

6.3 Terminal Nokia 6600 med extern GPS

6.3.1.Exekveringsmiljöer

Nokia 6600 är även den utrustad med Symbian OS v7.0 med en lågnivåmiljö och en javamiljö. Skillnaden är att javamiljön i denna terminal bygger på MIDP 2.0 som ger utvecklaren större frihet men där viss känsliga operationer är skyddade, se kapitel 3.2.2 MIDP 2.0. Terminalen saknar inbyggd GPS så en extern GPS måste i detta fall användas. Eftersom Nokia 6600 har stöd för bluetooth passar en extern trådlös GPS utmärkt då även denna har stöd för bluetooth. MIDP 2.0 stöder både bluetoothkommunikation och SMS vilket gör denna javalösning betydligt enklare än den för A920/A925. Om en lågnivålösning skulle bli aktuell, stöder nativemiljön både sändning av SMS samt kommunikation via bluetooth.

6.3.2 Kända problem

Ett stort problem som kan få hela denna lösning att fallera är att användaren manuellt tvingas konfirmera varje gång ett SMS ska skickas p.g.a. säkerhetslösningen i MIDP 2.0, se kapitel 3.2.2 MIDP 2.0. Ett viktigt krav på projektet är att hitta en smidig lösning som tar liten plats och sköter sig själv så att seglarlaget helt och hållet kan inrikta sig på seglingen. Att skicka SMS är i MIDP 2.0 betraktat som en känslig operation eftersom det dels ger en merkostnad för användaren och dels öppnar en möjlighet för en MIDlet att sprida känslig

personlig information vart den vill. Det som behövs är alltså att ge en MIDlet tillåtelse att skicka SMS utan att användaren ska behöva konfirmera varje gång ett SMS sänds d.v.s. genom signering ge en MIDlet rättigheten att skicka SMS utan konfirmering. För att erhålla dessa rättigheter räcker det tyvärr inte med den skyddsdomän som är möjlig att köpa, se kapitel Signeringsprocessen, utan kräver en av de domäner som öppnar telefonen mer, tillverkardomänen respektive operatörsdomänen. Antagligen är det omöjligt att få rättigheten att installera min MIDlet i någon av dessa domäner vilket skulle göra denna lösning oanvändbar.

6.3.3 Lösning

Eftersom det är oacceptabelt att seglarlaget måste konfirmera varje sändning av SMS finns det två lösningar för den tänkta hårdvaran.

- En javalösning där en MIDlet på något sätt signeras så att tillverkar- eller operatörsdomän erhålls, vilket kan bli svårt.
- En lågnivålösning.

6.4 Analys och metodbeskrivning

Under projektets gång har hårdvaran som tidigare nämnts ändrats och mycket tid har gått åt till att hitta information om hur systemutveckling utförs på de olika terminalerna som varit på tal samt vilka funktionaliteter som stöds av de olika terminalernas exekveringsmiljöer. Det har även varit problem med att hitta och konfigurera utvecklingsverktyg av typen freeware eller shareware för de olika plattformarnas exekveringsmiljöer.

Den första tänkta hårdvaran var en terminal av typ Motorola A920/A925. Mitt arbete började med att i detalj studera och testa vilka funtioner som var möjliga att implementera under de olika exekveringsmiljöer Symbian OS erbjöd. Ganska snart stod det klart att systemutveckling i javamiljön, MIDP 1.0, var betydligt mycket enklare än system utvecklingen i nativemiljön. Sandlådemodellen som javamiljön bygger på ligger som ett skyddande lager runt de javaapplikationer, MIDlet, som exekveras. Detta innebär att allt som är speciellt viktigt för ett inbäddat system, som minneshantering och undantagshantering, hanteras av det skyddande skiktet. I Symbian C++ krävs att utvecklaren själv tar större ansvarar för både minneshantering och undantagshantering vilket ger till följd att man som utvecklare får mer frihet och befogenhet.

För att hjälpa Symbian OS utvecklare finns en mängd kodningsstandarder som bör följas för att undvika systemkrascher och minnesläckor. Några av de större och viktigare punkterna som skiljer utvecklingen mellan de olika miljöerna är[20]:

- I stället för att lägga funktionsanrop där undantag kan kastas innanför en *try/catch*-sats som i vanlig C++-utveckling, skickar man i Symbian C++ dessa funktionsanrop som parameter till en specialmetod, *TRAP*.

  ```
  TInt error;
  TRAP(error, fooFunctionL());
  if (error!=KErrNone)
  {
    // threats exceptions throwed by "fooFunctionL()"
  }
  ```

- För att hantera minneshanteringen används en rensningsstack, *CleanupStack,* där alla objekt som skapas måste läggas. Om ett undantag uppstår kommer TRAP-

funktionen se till att det berörda objektet tas bort från stacken och att dess destruktor exekveras.

- I stället för att använda new ska new(ELeave) användas för att allokera minne till ny objekt. Metoden överlagrar new och tar ElLeave som paramter. Om minnesallokeringen skulle misslyckas kastas ett undantag och en systemkrasch undviks.
- Använd funktionskonstruktorer såsom ConsturktL för konstruktorer som kan komma att kasta ett undantag.

```
void CTest::testFunctionL()
{
  CSomeClass* obj1 = new (ELeave) CSomeClass;
  CleanupStack::PushL(obj1);
  CSomeClass* obj2 = new (ELeave) CSomeClass;
  CleanupStack::PushL(obj2);
  ConstructL();
}
```

Som tidigare redovisats under möjlig lösning, se kapitel, ska SMS skickas från nativemiljön eftersom MIDP 1.0 inte stöder SMS-hantering medan GPS-koordinaterna måste hämtas från javamiljön eftersom nativemiljön saknar bibliotek för kommunikation med den interna GPS-sändaren.

Kommunikationen med GPS-enheten sker med ett bibliotek, Location API, skapat av Motorola. För utveckling av MIDlet som använder GPS-enheten i terminalen måste Location API inkluderas i J2ME Wireless Toolkit projektet. Tyvärr saknas en positioneringssimulator vilket innebär att det inte går att simulera någon GPS-användning utan allt positioneringsutveckling måste testas skarpt på terminal.

Som lösning för kommunikationen mellan nativemiljön och javamiljön har jag i brist på andra alternativ valt socketkommunikation med en serversocket på lågnivåsidan och en socket på javasidan. Sockethanteringen på javasidan utvecklades klart och testades i terminalsimulatorn men hanteringen i nativemiljön gav problem. Kompilatorn vägrade godkänna den serversocketklass i Symbian OS jag behöver använda. Tyvärr hann detta problem inte lösas innan den samarbetspartner som skulle bistå projektet med bl.a. hårdvara drog sig ur projektet. Detta innebar att den nyss nämnda lösningen inte hann testas skarpt.

Eftersom projektet nu stod utan klienthårdvara verkade det som om projektet skulle läggas ner tills Thomas Ericsson på Oracle meddelade mig att de var beredda att bistå med terminaler och därmed ge projektet nytt liv. Den nya klienthårdvaran blev således Nokia modell 6600 samt en extern GPS.

Denna nya fas inleddes med att kontrollera huruvida en javalösning var möjlig på denna terminal. Det visade sig efter undersökning att terminalens javamiljö, MIDP 2.0, hade stöd för alla de operationer den tänkta lösningen skulle kräva. Vid denna tidpunkt insåg jag dock att terminalens säkerhetsmodell kunde bli ett problem för den automatiska sändningen av positions-SMS från terminal till server, men anade inte att problemet skulle bli så svårforcerat som det visade sig bli.

Efter mer informationssökning och djupare studier av javamiljöns säkerhetsmodell och förfarandet vid signering av en MIDlet stod det klart att det skulle bli svårt att lösa problemet med automatisk sändning av SMS genom att gå den vanliga vägen, d.v.s. skicka en CSR-fil, *Certificate Signing Request,* till en CA, *Certificat Authority*, se kapitel 3.2 Signeringsprocessen, som efter godkännande returnerar ett kodsigneringscertifikat för signering av de MIDlet man skapat.

Nokia var angelägna om att projektet skulle lyckas eftersom det är tänkt att en eventuell lösning skall användas i deras egna segelbåtstävlingar som går under namnet Nokia OOPS Cup samt att det gagnar dem men en lösning som använder en terminalplattform byggd av

dem. Enda chansen att få en javalösning att fungera var således att ta hjälp av just Nokia för att på något sätt få dem att hjälpa oss att få en mer öppen skyddsdomän.

Det är väldigt känsligt för Nokia att ge ut ett kodsigneringscertifikat som ger någon tillåtelse att installera i tillverkarens skyddsdomän eftersom ett sådant certifikat kan användas till att signera flera MIDlet. Därför föreslog jag istället att hela MIDlet-koden skickas till Nokia så att de själva skulle kunna kontrollera, paketera och signera koden innan jar-filen och den signerade jad-filen skickas tillbaks till mig. För att skydda ovetande användare som på något sätt exekverar koden utan vetskap om att SMS-meddelande skickas och användaren debiteras kan en varning visas initialt varje gång MIDleten exekveras.

Efter diskussion med Thomas Eriksson på Oracle som i sin tur tog upp problemet med Nokia hänvisade Nokia till deras globala betalhjälptjänst på nätet eftersom de lokalt inte kunde hitta någon lösning på problemet. Problemet är väldigt unikt eftersom det normalt sett varken finns någon möjlighet eller något intresse att få en MIDlet installerad i någon annan säkerhetsdomän än den för "godkända tredjepartsutvecklare", där SMS-meddelande ska kunna skickas utan konfirmation av användaren. Det är precis denna typ av MIDlet som säkerhetesmodellen i MIDP 2.0 har till uppgift att skydda användaren från.

Efter ytterligare påtryckningar på Nokia där vi påpekade att enda sättet att få en javalösning att fungera var med deras hjälp togs problemet då upp på högre ort och ett beslut togs av Nokia Sverige om att min MIDlet skulle få signeras. Det krävdes dock även ett klartäcken från Nokia Finland. Efter ytterligare några dagar kom det positiva beskedet från Nokia Finland om att signeringen skulle ske på Nokias huvudkontor i Finland, det enda stället i världen där en sådan signering är möjlig, med kravet att koden personligen transporterades dit och hem.

Fram till denna tidpunkt var det alltså oklart om en javalösning skulle bli möjlig eller om vissa eller alla delar skulle implementeras i terminalens lågnivåmiljö vilket skulle försvåra och definitivt förlänga utvecklingstiden betydligt. Nu stod det klart att en javalösning åtminstone teoretiskt sett skulle vara möjlig.

6.4.1 Intressant GPS-data

Efter diskussion med Oracle bestämdes det vilken information som skulle vara intressant att få från varje segelbåt. Longitud och latitud är intressanta för att positionera båtarna. Bäring och momentanhastighet är intressanta för att ge en bild av vilken riktning och i vilken hastighet olika båtar seglar. En tidsstämpel för att veta när en viss position kommer från.

Det format de flesta GPS-enheter följer och anger sina data i är en global standard utvecklad av en organisation som heter NMEA, *The National Marine Electronics Association.* Det finns flera olika format i denna standard beroende på vilken sorts data som anges.

De format som stöds av den GPS-enhet som används i projektet, EMTAC CRUXII/BTGPS, är [12]:

- GGA, fix data för position i tre dimensioner och beskrivning av dess riktighet.
- GSA, data som beskriver en satellits status.
- GSV, data som beskriver vilka satelliter sändaren borde kunna få kontakt med baserat på var sändaren befinner sig och satelliternas banor.
- RMC, rekommendera minimal GPS-data.

För mer detaljerad information om de olika formaten se bilaga 2.

Efter studier av de olika format som GPS-enheten stöder har jag valt att använda GGA och RMC för att täcka in de data som valdes intressanta ovan. För att grovt men enkelt kunna verifiera en viss positions riktighet har jag även valt att använda den parameter i GGA

formatet som anger det antal satelliter GPS-enheten har kontakt med. Eftersom en GPS-enhet behöver kontakt med fyra satelliter för att exakt kunna ange sin position, se kapitel 4 GPS, Global Positioning System, kan det vara intressant att veta §hur många satelliter GPS-enheten har haft kontakt med när den lämnat en viss position.

Nedan följer en lista på data som kan vara intressant att skicka till servern.

- Klockslag (UT) – GGA, parameter 1, t.ex. 141648
- Latitud – GGA, parameter 2 och 3, t.ex. 4816.354, N
- Longitud – GGA, parameter 4 och 5, t.ex. 1823.000, E
- Antal satelliter som följs – GGA, parameter 7, t.ex. 06
- Hastighet i knop – RMC, parameter 7, t.ex. 003.8
- Datum (UT) – RMC, paramter 9, t.ex. 060804
- Bäring – saknas i de format som stöds av den GPS-enhet som används. Data om bäring finns t.ex. i formatet RMB, *Recomended minimum Navigation Information.*

6.4.2 Specifikation för sändning av position i SMS-meddeleande.

Specifikationen för de SMS som skickas med GPS-data från terminal till server i systemet har i samarbete med Oracle bestämts till följande:

SMS-prefix [latitud,bäring,longitud,hastighet,datum&tid,terminalid,antal positioner]

När ett meddelande skickas tar SMSC, SMS-Centralen, alltid emot det och avgör beroende på SMS-nummer vart det ska skickas. Beroende på angivet SMS-nummer för meddelandet kan SMSC avgöra om det är ett registrerat kortnummer, vanligtvis tre till fem siffror, eller vanligt nummer. Om ett kortnummer påträffas kan ett prefix läsas från början av meddelandet för att avgöra vilken server eller tjänst meddelandet är riktat till.

Alltså anger SMS-prefixet vart meddelandet ska skickas och vilken tjänst som ska ta emot det medan terminalid är ett id för att identifiera terminalen och måste således vara unikt, t.ex. numret till det SIM-kort som används.

Exempel: oracle gps 4806.121N,0,1605.054,3.7,0406151244,0707301010,04

6.4.3 Positionshantering vid luckor i GSM-nätet

Positioneringssystemet kommer primärt att användas på segelbåtstävlingar till sjöss där GPRS-täckningen kan vara knapp eller totalt obefintlig. Eftersom GPS-nätet dock har mycket bättre täckning än GPRS-nätet, p.g.a. bland annat längre räckvidd, se kapitel 2.1 GSM, Global System for Communications, valdes datakommunikationen i projektet till SMS över GSM-nätet framför HTTP över GPRS-nätet.

GSM-nätet täcker in en större geografisk yta än GPRS-nätet men intensiteten sjunker med avstånd till land och civilisation. Hur täckningen ser ut under en segelbåtstävling blir därför beroende av hur tävlingsrutten går. För Gotland Runt t.ex. kommer båtarna in i en större radioskugga mellan Nynäshamn och Gotland.

Inom alla täckningsområden är det vanligt med radioskuggor beroende på hur topografin ser ut och GSM-masternas placering. På större segelbåtstävlingar är det möjligt att vid eventuella intressanta områden som saknar täckning, placera ut egna temporära GSM-master för att komplettera det befintliga nätet.

För att lösa problemet med att segelbåtarnas positioner inte kan skickas till servern när en terminal saknar täckning i GSM-nätet, sparas alla positioner undan i en vektor. Varje position har förutom den data den hämtat från GPS-enheten, även en flagga som anger om den blivit skickad eller inte. Vid ett schemalagt tillfälle stegas denna vektor genom,

position för position och varje position som ännu inte blivit skickad skickas i ett nytt SMS-meddelande enligt specifikationen ovan.

På detta sätt kommer alla positioner en viss båt haft under en täckningslucka i nätet skickas till servern så fort terminalen åter får täckning och inga positioner kommer att gå förlorade.

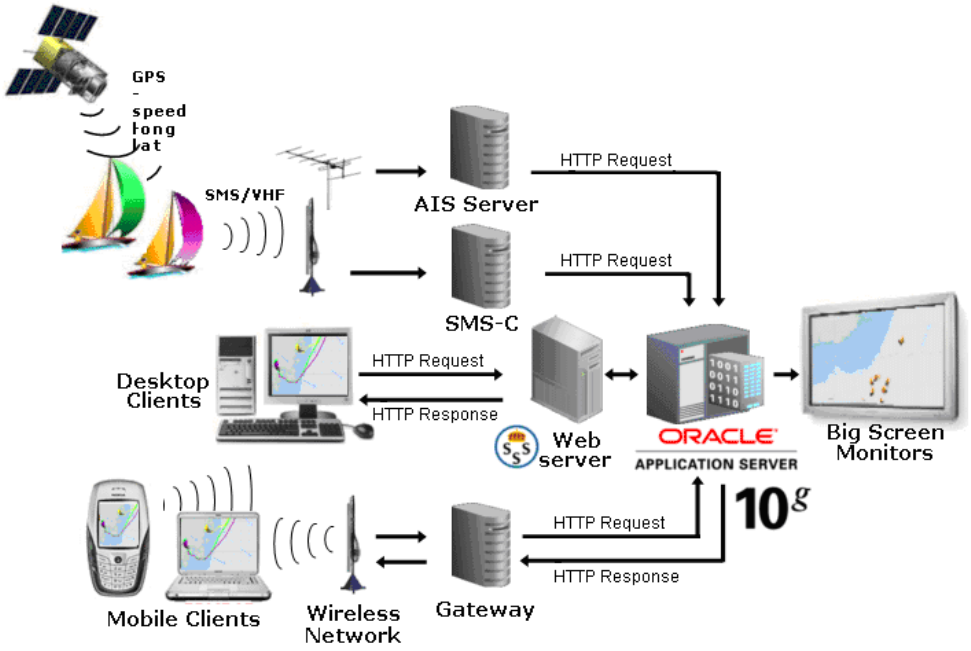
6.4.4 Systemparametrar

Ett antal parametrar kommer att behövas i systemet. Några av dessa redan har nämnts, t.ex. terminalid och sms-prefix. Andra är t.ex. telefonnummer till den SMS-tjänst som ska användas, hur ofta GPS-data ska hämtas från GPS-enheten eller vilken tidsoffset som ska används för systemet ska kunna ange tider i lokal tid och inte UT, universell tid.

Det kommer att finnas ett behov av att ändra dessa parametrar beroende på t.ex. användare, användningsområde eller server och därför måste användaren kunna göra detta i applikationen på någon sätt. För att användaren ska slippa skriva in sina inställningar varje gång applikationen startas behöver de sparas i telefonminnen för att sedan hämtas varje gång MIDleten startas. Detta kan göras med hjälp av J2ME klassen *RecordStore.*

7 Lösningsförslag

Den slutliga lösningen är en MIDlet, javalösning, byggd för en terminal av typ Nokia 6600 kopplad till en extern GPS-enhet. MIDleten har fått namnet LiveTracker, se figur 6.



Figur 6. *Systemöverskit för javalösning med Nokia 6100 och extern GPS-enhet.*

7.1 Trådar

I systemet används ett flertal trådar som har till uppgift att dels oberoende av varandra hantera schemalagda händelser och dels säkerställa att den externa kommunikationen inte låser hela systemet om något skulle gå fel.

7.2 Systemparametrar

I systemet används ett antal parametrar som användaren kan styra systemet med. Dessa går att kontrollera och editera under *inställningar* i systemet. Parametrarna hämtas från telefonminnet vid start av systemet och sparas vid eventuella ändringar för att finnas kvar nästa gång MIDleten exekveras.

- **SMS_NUMBER** är det nummer SMS-meddelandet ska skickas till.

- **TERMINAL_ID** är ett unikt id som särskiljer de olika båtarnas positioner
- **DEVICENAME** är hela eller början på det namn som GPS-enheten har. Detta används vid den initiala handskakningen över bluetooth mellan system och GPS-enhet.
- **TIME_OFFSET** är tidsskillnaden mellan den universella tiden och den lokala där systemet används. I Sverige är denna parameter två under sommartid och 1 under vintertid. Det går inte att i J2ME till skillnad från J2SE kontrollera om en tidpunkt i en viss tidszon är sommartid eller inte.
- **MAX_POSITIONS** är det maximala antal positioner som ska sparas av systemets positionsvektor. När vektorn är fylld kastas den positionen som legat där längst för att ge plats åt den nya.
- **MAX_POS_IN_VIEW** anger hur många positioner som ska visas på terminalens skärm. På skärmen visas positionerna i omvänd kronologis ordning. De positioner som skickats till server markeras med en stjärna och de som ännu ej blivit skickade markeras med ett minus.
- **GPS_INTERVAL** anger det intervall med vilket GPS-data ska hämtas. Valen som finns är var 10:e sekund, var 30:e sekund, varannan minut, var tionde minut och var 30:e minut.

7.3 Användarvyer

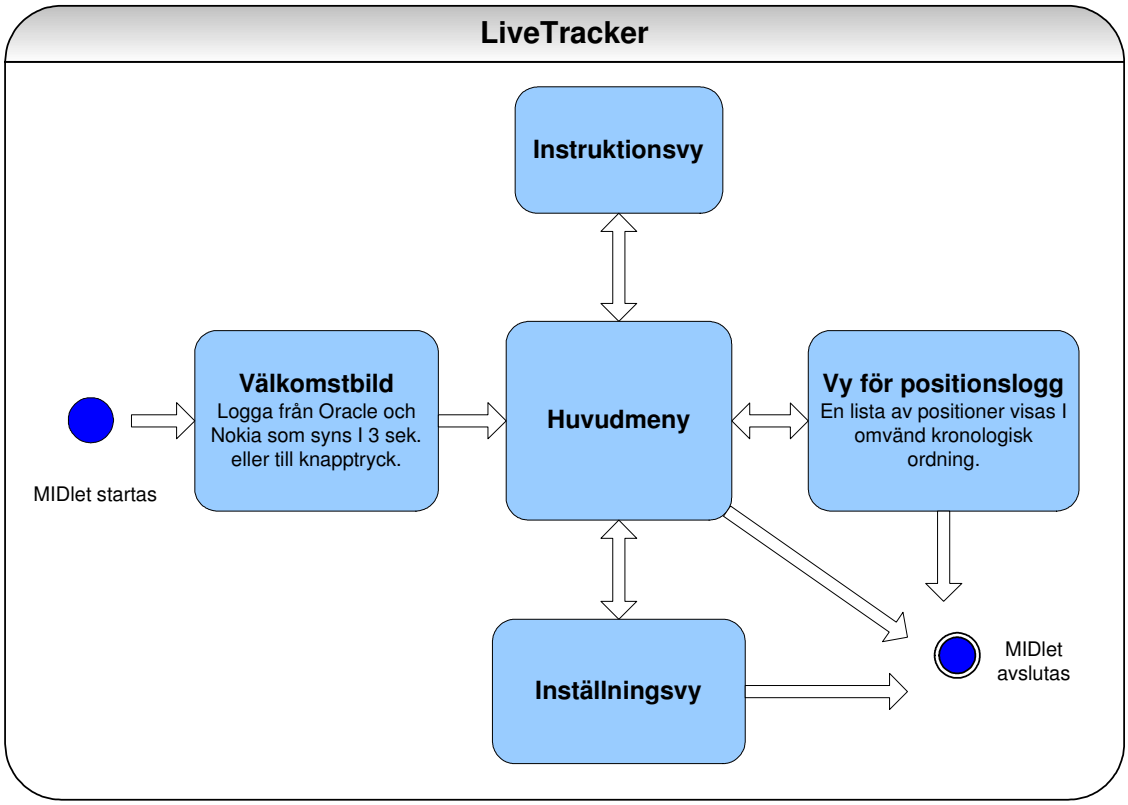
För att ge en bild av hur LiveTracker fungerar följer här en beskrivning av de olika tillstånden systemet kan befinna sig i.

Initialt då systemet startas visas en startbild med två loggor, en från Oracle och en från Nokia. Startbilden visas i tre sekunder eller tills ett knapptryck från användaren mottages. Efter startbilden presenteras huvudmenyn för användaren, se Figur 7. På huvudmenyn kan användaren välja att gå vidare till en instruktionsvy, en inställningsvy eller en vy med positionslogg.

I instruktionsvyn beskrivs kortfattat hur systemet fungerar och vilka inställningar som går att göra.

I inställningsvyn visas alla de inställningar i form av systemparametrar som är gjorda för systemet, se systemparametrar ovan. Det är även här parametrarna kan editeras av användaren. De ändringar av parametrarna som görs får direkt genomslag i hela systemet utan att MIDleten behöver startas om.

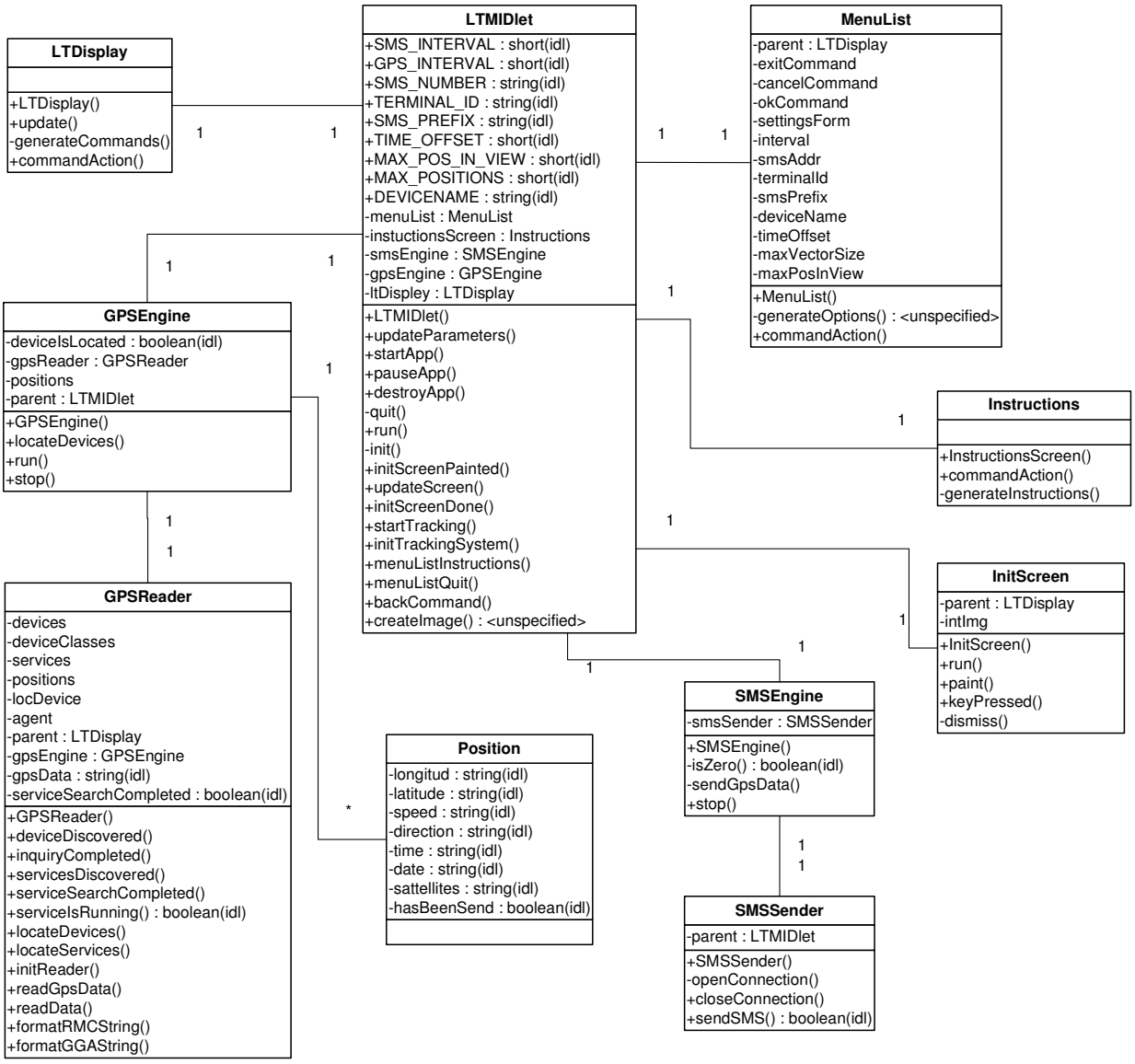
Positionsloggen är den vy där de positioner som hämtas från GPS-enheten presenteras. Här presenteras alla positioner i en formaterad rullningsbar lista. Framför varje position finns en markering i form av ett minus eller en stjärna. Ett minus innebär att positionen ännu inte har skickats vidare till den mottagande servern, p.g.a. att telefonen inte har någon täckning eller att positionen är nyhämtad från GPS-enheten och den schemalagda sändningen inte hunnit exekveras ännu. En stjärna framför en position markerar att positionen redan är skickad till den mottagande servern. Systemparametern MAX_POS_IN_VIEW, bestämmer hur många positioner som ska visas i listan i terminalfönstret. Positionsloggen visar positionerna i omvänd kronologisk ordning och uppdateras så fort en ny position hämtas från GPS-enheten. Från vyn för positionsloggen går det att stänga av och sätta på både sändningen av positioner samt läsningen av positioner från GPS-enheten.



Figur 7. *Användarvyer för LiveTracker.*

7.4 Systemarkitektur

Systemarkitekturen, se Figur 8, by gger på MVC-modellen [5] där presentation, affärslogik och datahantering separeras för att enklare kunna underhålla, återanvända och vidareutveckla ett system.



Figur 8. *Systemöversikt för LiveTracker.*

LTMIDlet

LTMIDlet är systemets huvudklass och ärver J2ME-klassen MIDlet. För att en MIDlet ska kunna exekveras på terminalen måste någon klass ärva javaklassen MIDlet som sköter kommunikationen mellan applikationen och det underliggande lagret i terminalen. Denna klass styr genom metoderna *startApp, pauseApp* och *destroyApp*, skapandet, pausandet och avslutandet av en MIDlet. Eftersom LTMIDlet använder sig av trådar, en för GPS-hanteringen och en för SMS-hanteringen, måste J2ME-gränssnittet *Runnable* implementeras.

I LTMIDlet klassens konstruktor hämtas ett antal systemparametrar lagrade i telefonminnet med hjälp av J2ME-objektet *RecordStore*. Om MIDleten exekveras för första gången finns inget *RecordStore*-objekt utan hårdkodade defaultvärden används.

LTDisplay

LTDisplay är den klass som hanterar presentation av all positionsdata. Denna klass ärver J2ME-klassen *Form* för den grafiska presentationen och J2ME-gränssnittet *CommandListener* för att kunna lyssna efter kommandon från användaren.

MenuList

MenuList är den klass som hanterar alla menyer med lyssnare som används i LiveTracker.

GPSEngine

I GPSEngine hanteras handskakningen med GPS-enheten över bluetooth. Det är även i denna klass som hämtningen av GPS-data schemaläggs och initieras.

En testapplikation med ett enkelt handskakningsförfarande mellan MIDlet och GPS utvecklades av Lasse Ahvenainen på Oracle i Göteborg och skickades till mig. Vissa delar har sedan efter vidareutveckling använts i GPSEngine och GPSReader.

GPSReader

GPSReader hanterar all direktkommunikation med GPS-enheten, dels under handskakningen men även senare när kontakten med GPS-enheten är upprättad och GPS-data ska hämtas.

Handskakningsförfarandet initieras med att signaler från aktiva bluetooth-enheter söks och registreras i en lista med bl.a. namn. Från denna lista plockas sedan den enhet med samma namn som systemparametern DEVICE_NAME, se systemparametrar ovan, ut för att lokalisera de lediga tjänster enheten erbjuder.

När den trådlösa kopplingen väl är upprättad mellan terminal och GPS utförs läsningen av data av GPSReader efter initiering av GPSEngine. De data som hämtas från GPS-enheten följer NMEA:s standard, se kapitel 6.4.2 Specifikation för sändning av position i SMS-meddeleande., och de två format som är intressanta för LiveTracker sorteras ut. Vidare sorteras de intressanta parametrarna ut och görs om till rätt format för att kunna lagras i en instans av klassen Position. Varje gång en ny position ska lagras i positionsvektorn kontrolleras att längden på vektorn inte överstiger systemparametern, MAX_POSITIONS, se kapitel 6.4.4 Systemparametrar, och om så skulle vara fallet raderas den tidigaste inlagda positionen.

Position

Klassen Position är en containerklass för positionsdata, se kapitel *6.4.1 Intressant* GPS-data.

SMSEngine

SMSEngine sköter SMS-hanteringen med hjälp av SMSSender. Varje gång ett SMS-meddelande ska skickas stegas alla sparade positioner genom i SMSEngine. Om en position redan har blivit skickad hoppas denna över och om en position saknar värde för latitud eller longitud, vilket är vanligt då GPS-enheten har dålig eller ingen kontakt med några satelliter, sätts positionen som redan skickad direkt utan att skickas. I annat fall skickas positionen genom SMSReader. SMSReader returnerar sant eller falskt beroende på om kontakt med GSM-nätet kunde fås för att SMSEngine ska kunna flagga positionen som skickad eller ej.

SMSSender

SMSSender öppnar och stänger kopplingen till GSM-nätet samt skickar SMS med positionsdata.

8 Utvärdering och slutsats

Positioneringssystemet LiveTracker som utvecklats inom detta examensarbete användes skarpt med stor framgång under tävlingen Gotland Runt 30 juni – 7 juli 2004. Totalt startade 237 båtar och av dem var 15 båtar utrustade med LiveTracker. Data Communications, segraren av IMS-klassen, samt alla Nokias trimaraner var med bland dessa. Se artiklar från www.gotlandrunt.se i bilaga 2.

Ett problem med systemet är att handskakningen mellan terminal och GPS-sändare ibland misslyckas då flera bluetoothprodukter är aktiverade inom samma område. Min lösning på problemet var att lägga in handskakningen i en loop i en egen tråd och så länge handskakningen misslyckas låta tråden sova några sekunder innan ett nytt försök till handskakning påbörjas. På detta sätt kommer systemet att fungera även om kontakten mellan terminal och GPS-enheten tillfälligt bryts. Då LiveTracker testades skarpt för första gången under tävlingen Gotland Runt med många bluetooth-enheter i närheten misslyckades handskakningen flera gånger för flera båtar och tog därför lång tid. Till nästa version borde därför initieringen av kommunikationen mellan terminal och GPS-enhet ses över.

Detta examensarbete har visat att det är möjligt att använda billig konsumenthårdvara för att positionera segelbåtar under en segelbåtstävling.

Tyvärr beror lösningens utseende mycket på vilken terminalmodell som väljs eftersom de olika utvecklings- och exekverningsmiljöerna som erbjuds skiljer sig åt. Funktionaliteten i ett positioneringssystem måste på några terminalmodeller delas upp på flera miljöer eftersom ingen miljö i dessa stöder implementation av samtliga funktioner. Ett exempel som diskuterats i denna uppsats är terminal Motorola A920/A925 som erbjuder tre miljöer, Symbians lågninvåmiljö för C++-utveckling, MIDP 1.0 och personalJava för javautveckling. Bland dessa miljöer måste implementation av olika funktioner delas upp eftersom t.ex. MIDP 1.0 inte erbjuder implementation av SMS-sändning medan lågnivåmiljön inte erbjuder kommunikation med den interna GPS-sändaren. En viss lösning blir även beroende av om en terminal har en intern GPS-sändare eller inte. Lösningen kommer också att bero av hur de lokala gränssnitten mot GPS-sändaren ser ut och hur kommunikationen med en eventuellt extern GPS-enhet ser ut.

Utvecklingen går mot att fler och fler terminaler utrustas med nyare javamiljöer, t.ex. MIDP 2.0. Denna utveckling kräver en kraftfull säkerhetsmodell som inte hämmar den enkla spridningen av MIDlets genom luften, eng. ”over the air”, samtidigt som den skyddar användaren från de hot en MIDlet kan medföra, såsom virusspridning och informationsstölder.

Den terminal som slutligen användes för positionssystemet var en Nokia 6600 med Symbians lågninvåmiljö samt den nyare javamiljön MIDP 2.0. Javamiljön erbjuder stöd för implementation av alla de funktioner som krävs men dess säkerhetsmodell ställde till problem då en MIDlet inte tilläts skicka SMS-meddelande utan godkännande av användaren. För att komma runt detta problem krävdes en signeringsprocess som i normala fall är otillåten då den ger en MIDlet totala rättigheter och därmed låser upp terminalen helt. Efter godkännande från först Nokia Sverige och sedan Nokia Global i Finland flögs den färdiga MIDleten över till Finland för denna specialsignering.

Positionssystemet som utvecklats i detta examensarbete kräver att segelbåtstävlingar, helt eller delvis befinner sig inom GSM-täckning. Anledningen till att SMS-meddelande används som kommunikationskanal är för att öka det möjliga geografiska användningsområdet eftersom SMS-meddelande använder sig av GSM-nätet som har mycket större räckvidd än både GPRS- och 3G-nätet. När en segelbåt saknar täckning lagras dess positioner tills dess att de kan skickas igen.

Referenser

[1] BEAULIIEU, MARK. Wireless Internet, Applications and Architecture

[2] ERICSSON TELECOM, TELIA OCH STUDENTLITTERATUR, 1988. Understanding telecommunications II, ISBN 91-44-00214-9

[3] FEDRIK OLSSON, 2004. Det börjar bli trångt i de trådlösa näten. *Datormagazin*, Vol. 4-61, 2004, s. 61.

[4] FORUM NOKIA. *MIDP 2.0: Tutorial on Signed MIDlets journal of algorithms*, Version 1.0 May 19, 2004

[5] GAMMA, ERICH & HELM, RICHARD & JOHNSON, RALPH & VLISSIDES, JOHN, 1994. Elements of Reusable Object-Oriented Software, ISBN 0-201-63361-2

[6] GSM 3.40 [ETSI GSM 03:40], GSM standarden för SMS

[7] HARRISSON RICHARD. Symbian OS C++ for mobile phones. ISBN 0-470-85611-4

[8] HOLMA,HARRI, WCDMA for UMTS, ISBN 0-471-48687-6

[9] INTERNATIONAL ENGINEERING CONSORIUM (IEC), 2004. G*lobal System for Mobile Communication (GSM)* http://www.iec.org

[10]JAVA EXPERT GROUP, 2004. Wireless Messaging API (WMA) for J2ME, JSR 120, whitepaper

[11]JAVA EXPERT GROUP, 2004. Location API (WMA) for J2ME, JSR 179, whitepaper

[12] NMEA. *NMEA Data,* http://www.gpsinformation.org/dale/nmea.htm

[13] SYMBIAN DEVELOPER LIBRARY, 2004. *Diffrences between PersonalJava and MIDP Java Environment.* http://www.symbian.com/developer 2004

[14] SYMBIAN, 2004. *UIQ SDK documentaion.* http://www.symbian com

[15] SYMBIAN, 2004. *Symbian developer page.* http://www.symbian com

[16] TRIMBLE. *All about GPS,* http://www.trimble.com 2004

[17] 3 www.tre.se, 2004-04-20

[18] NOKIA www.nokia.se, 2004-05-01

[19] EMTAC www.emtac.com, 2004-05-01

[20] SYMBIAN www.symbian.com, 2004-03-20

Bilaga 1

Publicitet

Artikel 1 från www.gantgotlandrunt.se

Följ båtarna på mobiltelefonen med Oracles positionssystem [KSSS Presscenter i Sandhamn 5/7]

I år kan du följa 20 av båtarna i Gant Gotland Runt direkt i din mobiltelefon och på webben. Genom Oracles avancerade databasteknik skickas kartor, router, resultat och positioner med nya positionssystemet ”Live Tracker” via SMS och webb. Båtarna är utrustade med GSM mobiltelefoner från Nokia och kontantkort från Vodafone.

Sven Lagerberg är seglingsentusiasten som också är produktansvarig för databaser på Oracles Tekniska försäljning. Med 10 Gotland Runt bakom sig, och seger i J/80 förra året, såg han möjligheterna att utnyttja Oracles gedigna kunskap och världsledande teknik så att alla seglingsintresserade enkelt skulle kunna följa båtarna i toppen på Gant Gotland Runt
- Det roliga är att vi använder vanliga konsumentprodukter i kombination med vår avancerade databasteknik i vårt nya positioneringssystem. Det finns en oerhörd potential i den här tekniken, säger Sven.
Oracle har tagit fram en mycket avancerad men lättanvänd applikation speciellt för Gant Gotland Runt och Volvo Baltic Race. 15 av båtarna i Gant Gotland Runt, bland annat ledarbåten i IMS-klassen Data Communication och alla trimaraner, är utrustade med en vanlig Nokia-telefon och ett kontantkort från Vodafone. Från telefonen skickas automatiskt positionen till en GPS-satellit var tionde minut, som direkt förmedlas till Oracles gigantiska HP-servrar där informationen formas till meddelanden och kartor med båtarnas aktuella position, fart och ställning i tävlingen. Även de fem V.O.60-båtarna ingår i Oracles rapportering, men de har utrustats med transpondrar som skickar informationen.
För alla med en mobiltelefon är det enkelt att ta emot den spännande informationen om hur det går i Gant Gotland Runt via SMS och MMS.
Allt som krävs är en modern mobiltelefon med wapteknik och att logga in på följande URL: http://selinrac2.ksss.se:7778/ptg/rm .
- Positionssystemet har fungerat perfekt hela vägen. Academy, som nu leder trimaranklassen, har haft bra täckning runt hela banan. Någon båt har haft problem med strömförsörjning vilket har gjort att informationen inte kunnat skickas. Men vi är mycket nöjda, säger Tomas Eriksson, som är affärsområdesansvarig för Oracle Wireless&Cables.
Oracle har smugit igång Live Tracker för att testa det i samband med Gant Gotland Runt, så sent som på torsdagen fick de alla tillstånd från Nokia för det unika systemet.
- Vi vill visa hur väl vår teknik fungerar och hur enkelt det är för konsumenterna att använda den med sina vanliga mobiltelefoner. Det finns oerhörda utvecklingsmöjligheter, och vi har bara börjat utveckla de här applikationerna.
Enkelt var det inte, för att få utveckla systemet till Gotland Runt behövde Sven och Tomas klartecken från högsta ort i Oracle. Den extremt seglingsintresserade Larry Ellison, ägare och koncernchef i Oracle, gav själv sitt OK till att bygga Live Tracker. Och han har följt utvecklingen av systemet med stort intresse.

Följande båtar i Gant Gotland Runt kan du följa genom Oracles Live Tracker:
Academy, HiQ, Nokia, TietoEnator/L’Oreal, GoreTex (brutit), SonyEricsson, Elanders, Avant, AV-Teknik, JMS Next Generation, Galatea, Edinprogressiva, RPP, Xteam, Data Communication, Gant, Summer Wine, MonAmi (Hans Svedman, Scampi) och Blå Karat.
Logga in på wap-URL, http://selinrac2.ksss.se:7778/ptg/rm .

Alla resultat och positioner kan du hitta på webbsajten: http://www.gantgotlandrunt.se under knappen Resultat & Deltagare.

Artikel från www.gantgotlandrunt.se

Så fungerar tekniken bakom positioneringssystemet

Två olika system har använts för att leverera båtarnas positioner till Oracles GIS-system. Ett levererat av Saab Transpondertech och ett utvecklat av Oracle, Nokia och NetLight baserat på vanliga Nokia telefoner och en blåtands-GPS. Använd muspekaren på båtens linje, aktuell position eller på öarna för att få mer information. För att panorera bilden används pilarna längst ner på menyn.

På kartan visas positionerna för VO.60 båtarna i decimalgrader och inte i grader, minuter och sekunder (WSG84). Det krävs en svg-viewer från Adobe för att kunna se kartorna.

Kom ihåg att installationen på båtarna är temporär och ibland kan innehålla störningar som t ex att strömmen faller ur eller att telefonens placering gör det svårt att få kontakt med en basstation.

Bilaga 2

Specifikation från NMEA

Artikel från http://www.gpsinformation.org/dale/nmea.htm#nmea med utdrag av specifikation från NMEA, *The National Marine ElectronicsAssociation*

GGA - essential fix data which provide 3D location and accuracy data.

```
 $GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                              4 = Real Time Kinematic
                              5 = Float RTK
                               6 = estimated (dead reckoning) (2.3
feature)
                              7 = Manual input mode
                              8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

If the height of geoid is missing then the altitude should be suspect. Some non-standard implementations report altitude with respect to the ellipsoid rather than geoid altitude. Some units do not report negative altitudes at all. This is the only sentence that reports altitude.

GSA - GPS DOP and active satellites. This sentence provides details on the nature of the fix. It includes the numbers of the satellites being used in the current solution and the DOP. DOP (dilution of precision) is an indication of the effect of satellite geometry on the accuracy of the fix. It is a unitless number where smaller is better. For 3D fixes using 4 satellites a 1.0 would be considered to be a perfect number, however for overdetermined solutions it is possible to see numbers below 1.0.

There are differences in the way the PRN's are presented which can effect the ability of some programs to display this data. For example, in the example shown below there are 5 satellites in the solution and the null fields are scattered indicating that the almanac would show satellites in the null positions that are not being used as part of this solution. Other receivers might output all of the satellites used at the beginning of the sentence with the null field all stacked up at the end. This difference accounts for some satellite display programs not always being able to display the satellites being tracked. Some units may show all satellites that have ephemeris data without regard to their use as part of the solution but this is non-standard.

```
  $GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39

Where:
     GSA      Satellite status
     A        Auto selection of 2D or 3D fix (M = manual)
     3        3D fix - values include: 1 = no fix
                                       2 = 2D fix
                                       3 = 3D fix
     04,05... PRNs of satellites used for fix (space for 12)
     2.5      PDOP (dilution of precision)
     1.3      Horizontal dilution of precision (HDOP)
     2.1      Vertical dilution of precision (VDOP)
     *39      the checksum data, always begins with *
```

GSV - Satellites in View shows data about the satellites that the unit might be able to find based on its viewing mask and almanac data. It also shows current ability to track this data. Note that one GSV sentence only can provide data for up to 4 satellites and thus there may need to be 3 sentences for the full information. It is reasonable for the GSV sentence to contain more satellites than GGA might indicate since GSV may include satellites that are not used as part of the solution. It is not a requirment that the GSV sentences all appear in sequence. To avoid overloading the data bandwidth some receivers may place the various sentences in totally different samples since each sentence identifies which one it is.

The field called SNR (Signal to Noise Ratio) in the NMEA standard is often referred to as signal strength. SNR is an indirect but more useful value that raw signal strength. It can range from 0 to 99 and has units of dB according to the NMEA standard, but the various manufacturers send different ranges of numbers with different starting numbers so the values themselves cannot necessarily be used to evaluate different units. The range of working values in a given gps will usually show a difference of about 25 to 35 between the lowest and highest values, however 0 is a special case and may be shown on satellites that are in view but not being tracked.

```
$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*7
5

Where:
      GSV          Satellites in view
      2            Number of sentences for full data
      1            sentence 1 of 2
      08           Number of satellites in view
```

```
      01           Satellite PRN number
      40           Elevation, degrees
      083          Azimuth, degrees
      46           SNR - higher is better
           for up to 4 satellites per sentence
      *75          the checksum data, always begins with *
```

RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data. It is called RMC, The Recommended Minimum, which will look similar to:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6
A

Where:
     RMC          Recommended Minimum sentence C
     123519       Fix taken at 12:35:19 UTC
     A            Status A=active or V=Void.
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     022.4        Speed over the ground in knots
     084.4        Track angle in degrees True
     230394       Date - 23rd of March 1994
     003.1,W      Magnetic Variation
     *6A          The checksum data, always begins with *
```

Note that, as of the 2.3 release of NMEA, there is a new field in the RMC sentence at the end just prior to the checksum. For more information on this field see here.