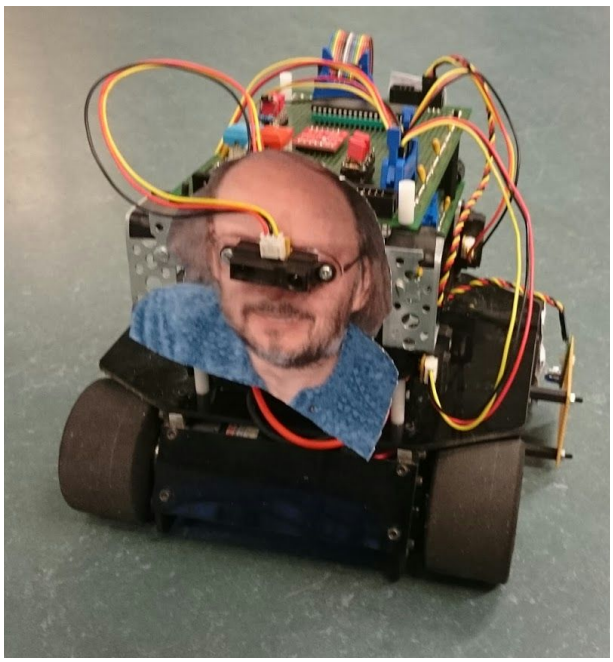
Teknisk dokumentation

Victor Tranell

Version 0.1



Status

Granskad		
Godkänd		

M/S Sea++

Projektgrupp 2 , HT15
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Anton Rooth		070 369 01 40	antro937@student.liu.se
Erik Rönmark		076 818 78 26	eriro331@student.liu.se
Michael Sörsäter	Dokumentansvarig (DOK)	076 142 70 99	micso554@student.liu.se
Mikael Ångman		073 843 15 00	mikan972@student.liu.se
Peter Tullstedt		073 714 45 66	pettu298@student.liu.se
Victor Tranell	Projektledare (PL)	073 680 71 09	victr593@student.liu.se

E-postlista för hela gruppen: tsea29.grupp2@gmail.com
Hemsida: https://github.com/nullacid/grupp2robot

Kund: Instutitionen för systemteknik, Linköpings Universitet, 581 83 LINKÖPING,
telefon 013-28 10 00, fax: 013-13 92 82
Kontaktperson hos kund: Tomas Svensson, 013-28 13 68, tomas.svensson@liu.se

Kursansvarig: Tomas Svensson, B-huset, rum 3B:528, 013-28 13 68, tomas@isy.liu.se
Handledare: Peter Johansson, B-Huset, rum 3D: 541, 013-28 13 45, peter.a.johansson@liu.se

Innehållsförteckning

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
1.0	2015-12-16	Första versionen	Victor Tranell	

1. Inledning

Systemet M/S Sea++ är en kartrobot som kan placeras i ett okänt rum, undersöka omgivningen och rita ut rummets väggar.

Projektet utfördes som ett moment i kursen TSEA29 vid Linköpings Universitet under hösten 2015. Syftet med projektet var att projektmedlemmarna skulle få erfarenhet av att jobba enligt en projektmodell samt att ge övning i konstruktion och utveckling av mikrodatorer. Projektet utfördes enligt projektmodellen LIPS.

Syftet med detta dokument är att beskriva hur systemet är designat och fungerar dels i utbildningssyfte och dels för att kunden ska kunna felsöka problem vid vidareutveckling av systemet.

2. Översikt av systemet

Systemet är uppbyggt av tre moduler som är monterade på ett chassi. Utöver det används en extern dator.

- Den externa datorn ska med ett grafiskt gränssnitt visa sensordata samt realtidsuppritning av kartan. Den externa datorn kommunicerar med kommunikationsmodulen via Bluetooth.
- Styrmodulen ansvarar för all styrlogik, tar emot kommandon och data från övriga moduler och styr hjulen utifrån detta.
- Sensormodulen samlar in sensordata som bearbetas för att få värden som de andra modulerna kan hämta.

2.1. Gränssnitt

Kommunikationen i systemet sker seriellt och asynkront (USART) mellan alla moduler. Eftersom vi inte vill att kommunikationsmodulen ska vara en flaskhals så lyssnar den bara på USART-porten uppåt för att sedan skicka vidare meddelanden nedåt. De andra modulerna kontrollerar en flagga i sin huvudloop för att se om det finns ett meddelande att hämta och behandlar meddelandet först då.

Kommunikationen mellan den externa datorn och kommunikationsmodulen sker via Bluetooth. Mellan kommunikationsmodulen och styrmodulen samt mellan styrmodulen och sensormodulen sker kommunikationen med USART. När kommunikation sker nedåt i systemet skickas ett funktionskommando, se Bild 1. När kommunikationen sker uppåt i systemet så skickas databytes med bl.a. sensordata, vilket enbart sker på förfrågan uppifrån. I funktionskommandot så bestämmer de 2 mest signifikanta bitarna längden i bytes på datasträngen som ska skickas tillbaka. Dessa är 0 när ingen data ska hämtas (styrkommandon).

Bild 1. Översikt av bitarnas betydelse i ett funktionskommando.

2.1.1. Kommandon

I Bild 2 visas hur ett kommando skickas genom systemet och vilken information som skickas i vilket steg. Funktionskommandon skickas bara nedåt i systemet och datakommandon skickas bara uppåt i systemet. När en modul har skickat ned ett kommando för att hämta data så kommer den att vänta på att datan returneras innan den kan fortsätta med sitt arbete.

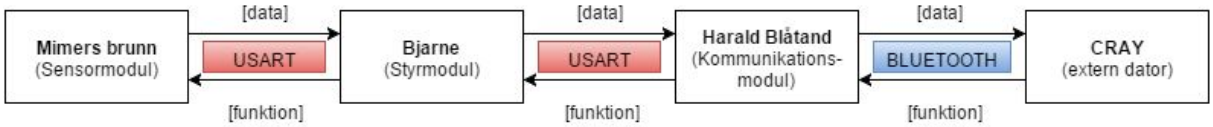


Bild 2. Hämtning av sensordata.

När data ska hämtas kommer enheten som frågade nedåt att vänta på att få tillbaka data nedifrån. Eftersom den har kvar funktionskommandot som den skickade nedåt vet den även hur många bytes den ska vänta på innan all data har hämtats. Den modul som är ansvarig för originalförfrågan kommer av samma anledning även att veta vad datan som tas emot hör till. I Tabell 1 och Tabell 2 nedan listas de olika funktionskommandona som används.

Tabell 1 - Meddelandedata för styrningen

Tangentbordshändelse	**Beskrivning**	**OP-kod (hexadecimal)**
Knapp W trycks ner	Kör framåt	00
Knapp W släpps	Sluta köra framåt	01
Knapp A trycks ner	Kör framåt vänster	02
Knapp A släpps	Sluta köra framåt vänster	03
Knapp S trycks ner	Kör bakåt	04
Knapp S släpps	Sluta köra bakåt	05
Knapp D trycks ner	Kör framåt höger	06
Knapp D släpps	Sluta köra framåt höger	07
Knapp Q trycks ner	Rotera vänster	22

Knapp Q släpps upp	Sluta rotera vänster	23
Knapp E trycks ner	Rotera höger	24
Knapp E släpps upp	Sluta rotera höger	25

Tabell 2 - Meddelandedata för hämtning av data.

Datatyp att hämta	**Beskrivning**	**Data (hex), svarets storlek (antal bytes)**	**OP**
Sensordata IR framåt	Avstånd i cm till vägg framåt	48, 1	08
Sensordata IR höger fram	Avstånd i cm till vägg	49, 1	09
Sensordata IR höger bak	Avstånd i cm till vägg	4A, 1	0A
Sensordata IR vänster fram	Avstånd i cm till vägg	4B, 1	0B
Sensordata IR vänster bak	Avstånd i cm till vägg	4C, 1	0C
Tillryggalagd sträcka	Antalet åkta rutor sedan start	8D, 2	0D
Tokenvärde framåt	Utifrån IR-data räkna ut antalet rutor till vägg	4F, 1	0F
Tokenvärde parallell höger	Differens mellan de högra IR-sensorerna	50, 1	10
Tokenvärde parallell vänster	Differens mellan de vänstra IR-sensorerna	51, 1	11
Tokenvärde Gyro	Anger när rotationen är klar.	52, 1	12
Tokenvärde vägg höger fram	Utifrån IR-data räkna ut antalet rutor till vägg	53, 1	13
Tokenvärde vägg höger bak	Utifrån IR-data räkna ut antalet rutor till vägg	54, 1	14
Tokenvärde vägg vänster fram	Utifrån IR-data räkna ut antalet rutor till vägg	55, 1	15
Tokenvärde vägg vänster bak	Utifrån IR-data räkna ut antalet rutor till vägg	56, 1	16
Tokenvärde Reflexsensor	Antalet åkta segment	57,1	17
Förändringskö karta dequeue	x, y och info om tilen. Se Bild 11	98, 2	18

Senaste styrdata	Senaste styrbeslut	99, 2	19
Systemets position i kartan	x och y	9A, 2	1A
Position i algoritm	Vad roboten just nu utför (följa vägg, utforska unexplored, hitta hem, o.s.v)	5B, 1	1B
Begär tempKartdata	Begär den senast begärda tilen på nytt.	9C, 2	1C
Begär all sensordata	Begär all sensordata i sekvens. Används endast mellan styrmodulen och sensormodulen.	1D,0	1D
Sync	Synkar kommandon med svaren. Används om bluetoothkopplingen bryts.	26,0	26

2.1.2. SPI

SPI används i sensormodulen, mellan sensorenheten(master) och gyrot(slave).

SPI står för Serial Peripheral Interface. Det finns en masterenhet och en slavenhet och de båda enheterna innehåller varsitt skiftregister. I SPI används dessa olika signaler:

- SS, slave select, används för att välja vilken slavenhet som ska användas
- MOSI, Master Output Slave Input, Datalinje där masterenheten skickar information till slavenheten
- MISO, Master Input Slave Output, Datalinje där slavenheten matar masterenheten med information
- SCLK, Klocka för synkronisering

I vår implementation behöver vi aldrig ha mer än en slavenhet per masterenhet, vår implementation ser ut som Bild 3.

Bild 3. Vår implementation av SPI

2.1.3. Bluetooth

Bluetooth används mellan kommunikationsmodulen (Harald) och den externa datorn (CRAY). Läs mer om implementationen i kapitel 4.

2.1.4. USART

ATmega1284p har inbyggt stöd för USART i form av ett register och följande pinnar (per port):

- RxD: Reciever
- TxD: Transmitter
- XCK: Klocka

Våra ATmega-kort körs i asynkront normalt läge och använder sig då inte av XCK klockan. Istället så har baud-raten satts till 115 200 bps för processorerna i mjukvaran. Det finns två USART-portar på ATmega1284, den första används för kommunikation uppåt och den andra används för kommunikation nedåt.

3. PC - CRAY™ Super Computer

CRAY uppfyller tre funktioner:

- Hämta data från systemet
- Presentera datan från systemet
- Skicka styrkommandon till systemet

Det externa programmet (CRAY) är skrivet i Python 3.1 på grund av att de två externa biblioteken som har använts båda fungerar till version 3.1. Datan och framförallt kartan presenteras i ett GUI. När systemet är i sitt manuella läge kan systemet styras med knapptryck i CRAY.

3.1. Bluetooth → Kommunikationsmodul

CRAY används för att kommunicera med M/S Sea++ genom Bluetooth. För att enkelt kunna implementera bluetoothkommunikationen så används pythonbiblioteket Pybluez[1]. De kommandon som används för att kommunicera med systemet finns i Tabell 1 och Tabell 2.

3.2. Implementation av kommandon

Pythonbiblioteket pygame[2] har använts för att enkelt kunna implementera knapptryck (nedtryckningar och uppsläppningar). Nedtryckning av en knapp skickar ett kommando för att styra motorerna. När knappen sedan släpps upp så stänger systemet av motorutslaget som hör till det kommandot (t.ex. om man håller ned A och W och sedan släpper A kommer systemet att sluta svänga, men fortfarande åka framåt). Dessa kommandon finns i Tabell 1.

Hämtning av data sker autonomt i CRAY, och använder sig av de kommandon som finns i Tabell 2. När datan kommer tillbaka till systemet så sätts det till rätt variabel så att den tydligt kan presenteras på skärmen.

3.3. GUI

CRAYs GUI är skrivet med hjälp av biblioteket pygame. Det presenterar sensordata, styrdata och kartan från M/S Sea++. En bild av GUI:t finns att hitta i Bild 4, där kartan tar upp vänstra delen av skärmen och sensordatan är upplistad på högerdelen.

[1] https://github.com/karulis/pybluez
[2] http://pygame.org/news.html

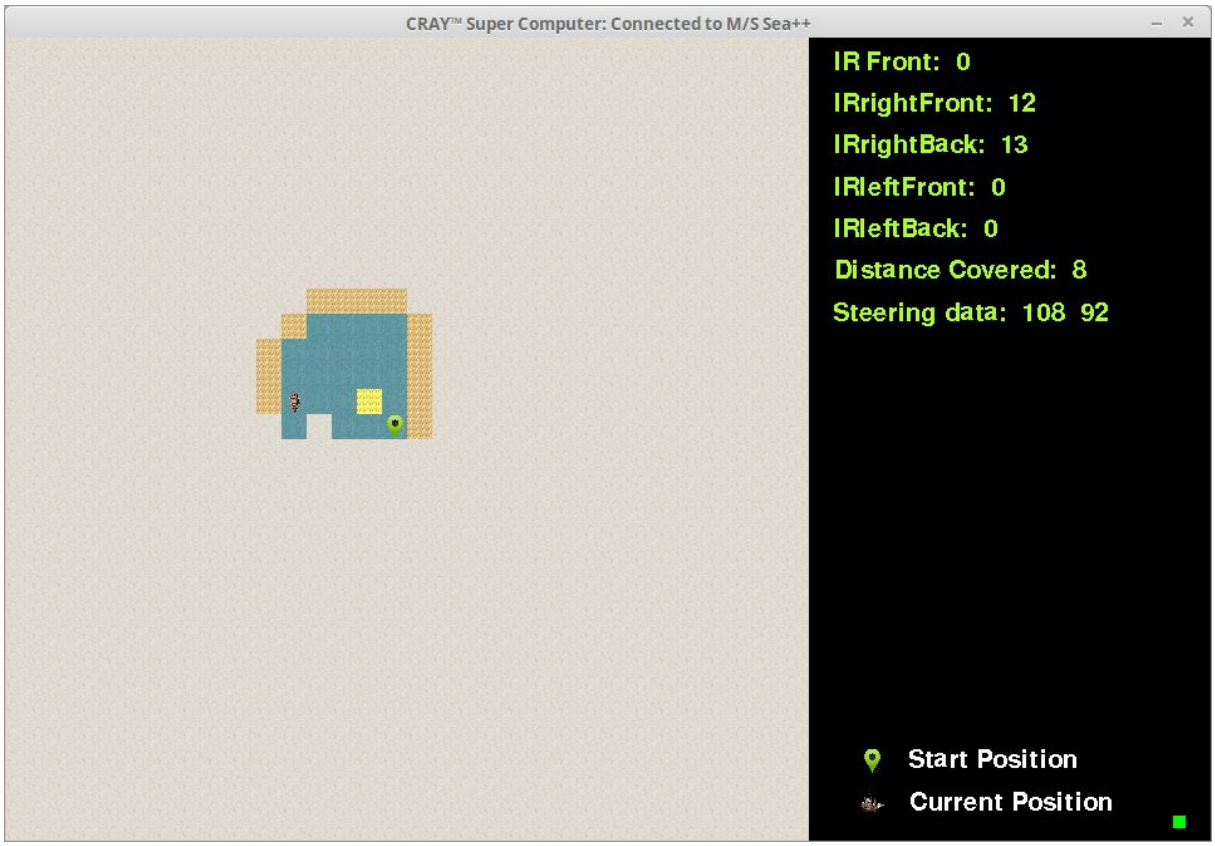


Bild 4. GUI:t för CRAY.

3.3.1. Karta

Kartan i systemet representeras av en 32 x 32 array med värden för om varje ruta är outforskad, öppen, innervägg eller yttervägg. När CRAY begär kartuppdatering från systemet så skickas den senaste kartuppdateringen i systemet upp till datorn. I kartan visas också systemets startposition och nuvarande position (nuvarande position hämtas också från systemet).

4. Kommunikationsmodul - Harald Blåtand

Harald Blåtand hanterar kommunikationen mellan systemet och den externa datorn. Modulen har en dator av typen ATmega1284p. Modulen väntar på ett kommando från den externa datorn, när den tar emot ett kommando skickar den vidare det via USART till styrmodulen. Om kommandot var att hämta data väntar den på en respons och skickar sedan vidare datan upp till den externa datorn. Om datan som den ska få tillbaka av styrmodulen innehåller fler bytes väntar kommunikationsmodulen tills den fått alla bytes innan den skickar vidare datan till den externa datorn. Efter det lyssnar den igen efter ett kommando att ta emot från den externa datorn. En tydligare bild av kommunikationsmodulens arbetsgång visas i Bild 5. Till kommunikationsmodulen sitter en resetknapp inkopplad. Denna knapp återställer kommunikationsmodulen till sitt ursprungliga läge och PCn behöver då återansluta sig till fireflykortet.

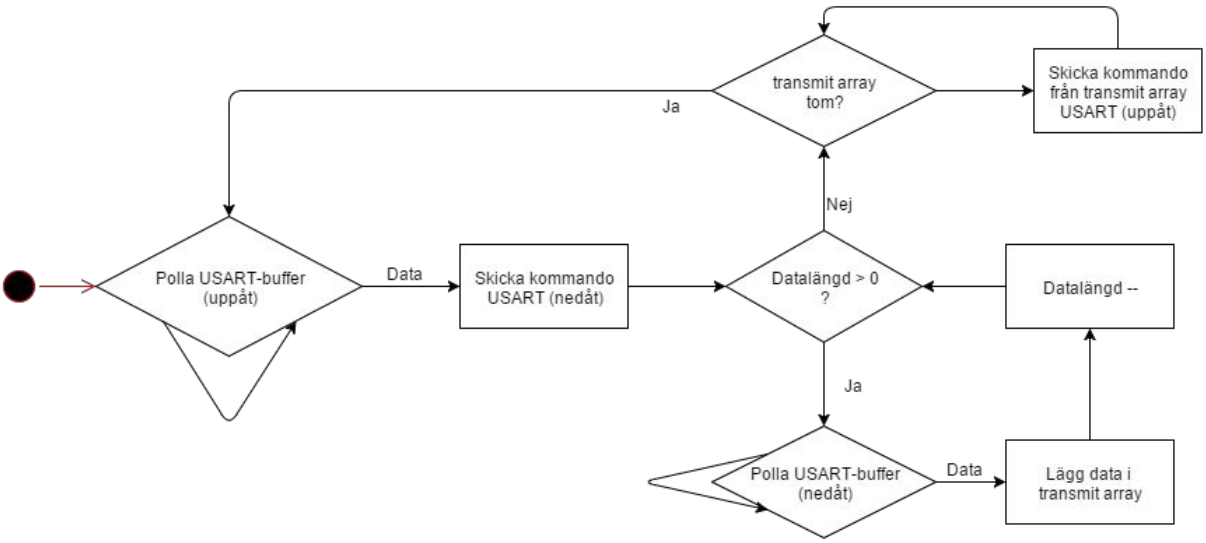


Bild 5. Flödesschema över kommunikationsmodulens arbetsgång

4.1. Firefly (Bluetooth)

Kommuikationsmodulens USART(upp)-port är kopplad till en FireFly-modul. Den översätter bluetooth-kommunikation till USART-kommunikationen som används i systemet. Så när kommunikationsmodulen skickar data till USART(upp)-porten översätts det av FireFly-modulen till bluetooth. FireFly-modulen tar även emot data från den externa datorn som den översätter till USART och skickar vidare till kommunikationsmodulen.

4.2. USART → Styrmodul

Kommunikationen mellan kommunikationsmodulen och styrmodulen använder sig av USART.

5. Styrmodul - Bjarne

Bjarne har till uppgift att styra och kontrollera motorerna som driver hjulen. Bjarne består av en AVR av typen ATmega1284p. AVRen är ansluten till sensormodulen via USART och till kommunikationsmodulen via USART. Instruktioner om styrning eller datahämtning hämtas från kommunikationsmodulen. Bjarne kör även algoritmen för autonomt läge. En knapp och en brytare är inkopplade till Bjarne. Knappen är en resetknapp som används för att återställa styrmodulen till ursprungligt tillstånd. Samma knapp är även inkopplad till kommunikationsmoduelns resetingång. Brytaren bestämmer om roboten ska vara i autonomt eller manuellt läge.

5.1. Motordrivning

Styrenheten innehåller två motorpar kopplade till höger respektive vänster hjulpar vilket gör roboten differentialstyrd. Motorerna kopplas in till styrmodulen via en IDC-kontakt, se Bild 6.

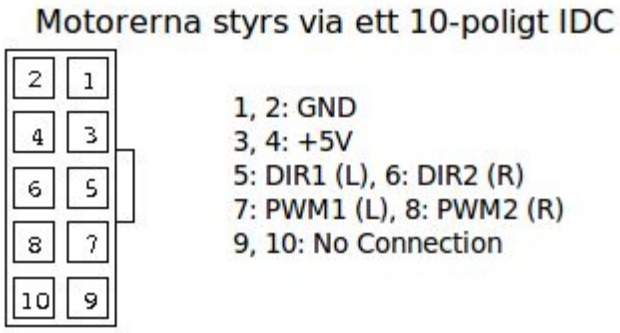


Bild 6. De olika pinnarna till motorstyrningen

Dessa motorer styrs via två signaler var; DIR och PWM. DIR styr motorenas riktning och PWM är en pulsbreddsmodulerad signal som kontrollerar motorernas hastighet, se Bild 7. PWM fungerar genom att fyrkantsvågor kontinuerligt skickas, i vårt fall från styrenheten till motorerna. Genom att slå av och på spänningen varierar man effekten hos motorerna, längre pulser ger högre hastigheter.

För att generera PWM-signalen används 16-bits-räknare inkluderad i ATMega1284p. Dessa räknare jämför sina nuvarande tal med ett referenstal. När dessa två tal är lika sätts utsignalen till logisk 0. Räknaren fortsätter sedan räkna uppåt tills maxtalen nås och den börjar om igen. När räknaren börjar om sätts även utsignalen till logisk 1, vilket alltså resulterar i en fyrkantsvåg där referenstalen bestämmer förhållandet mellan $t_{hög}$ och $t_{låg}$.

PWM-signalen styrs via två olika register; TCNT0 och OCR0. TCNT0 styr räknarens funktion och OCR0 är referenstalet. PWM-signalens frekvens styrs genom att räknarens egna frekvens delas med ett heltal (prescaling). Uttrycket för PWM-frekvensen ser ut på följande sätt:

$$F = CLK/(N * 1024)$$

Där CLK är datorns klockfrekvens (20 MHz) och N är heltalet. I vårt fall väljs N till 64 vilket ger en frekvens på 1,221 kHz.

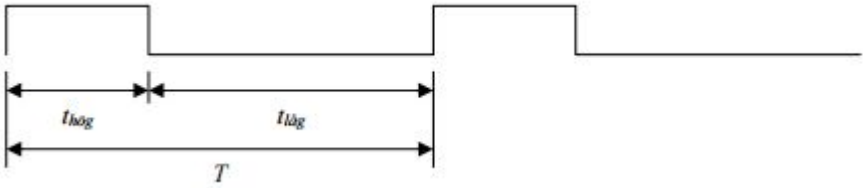


Bild 7. Större $t_{hög}$ ger högre hastighet på motorerna.

5.2. Karta

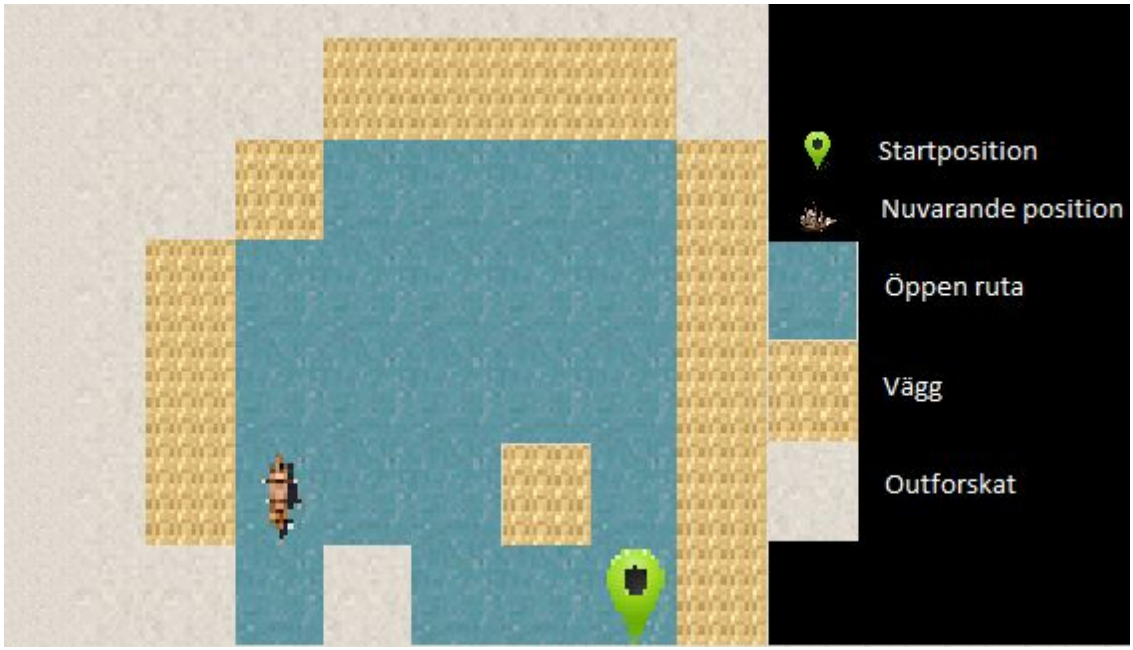


Bild 8. Representation av karta

Roboten har en representation av den insamlade kartdatan. Representationen är en 2D-array vilken är 34x34 rutor där varje ruta representerar varje möjlig position för roboten. Roboten utgår från att den startar i mitten av den (16,16, 0-indexerad). Initialt är varje ruta markerad som outforskad. Allt eftersom roboten upptäcker sin omgivning markerar den olika rutor i kartan som golv, yttervägg eller innervägg, se Bild 8.

Direkt vid uppstart börjar roboten markera utforskat/vägg mot sidorna. Sensorerna på högersidan markerar direkt vägg medan sensorerna på vänstersidan markerar innervägg vilka kan bli överskrivna av en högervägg. Detta gör att när roboten åkt ett varv kan den enkelt veta om det finns outforskade köksöar eller inte.

När ny kartinformation upptäcks uppdateras kartan med den nya informationen. Samtidigt läggs den nya kartinformationen till i en förändringskö. Där lagras 2 bytes med information om koordinaterna och tillståndet rutan är i. Se Bild 9 för hur datan är placerad i de 2 bytesen. När den externa datorn uppdaterar sin karta så frågar den efter en uppdatering från förändringskön, så att den inte behöver ladda över hela kartan varje gång.

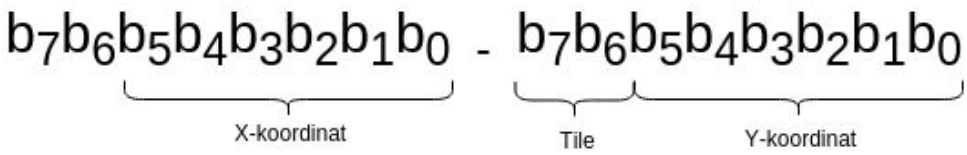


Bild 9. En förändring i kartan lagras med 2 bytes.

5.3. Autonoma läget

När systemet startar i det autonoma läget så börjar roboten följa högerväggen. Hur M/S Sea++ följer högerväggen visas i Bild 10. Medan M/S Sea++ undersöker högerväggen så ritar den upp en karta över omgivningen så fort sensorerna ser något. Kartan består av rutor som kan vara av typ utsida, outforskad, golv, innervägg eller yttervägg. Längst ut som en ram runt kartan ligger utsida för att underlätta för algoritmen. De väggar som läses av de högra sensorerna markeras som yttervägg och de som läses av de vänstra sensorerna markeras som innervägg. När M/S Sea++ har åkt en ruta frammåt görs en djupet-först-sökning efter en ruta av typ utsida. Om en sådan ruta hittas, innebär det att ytterväggen inte är sluten. Om djupet-först-sökningen inte hittar någon utsida-ruta så betyder det att den har ritat klart ytterväggarna och kan börja leta efter köksöar.

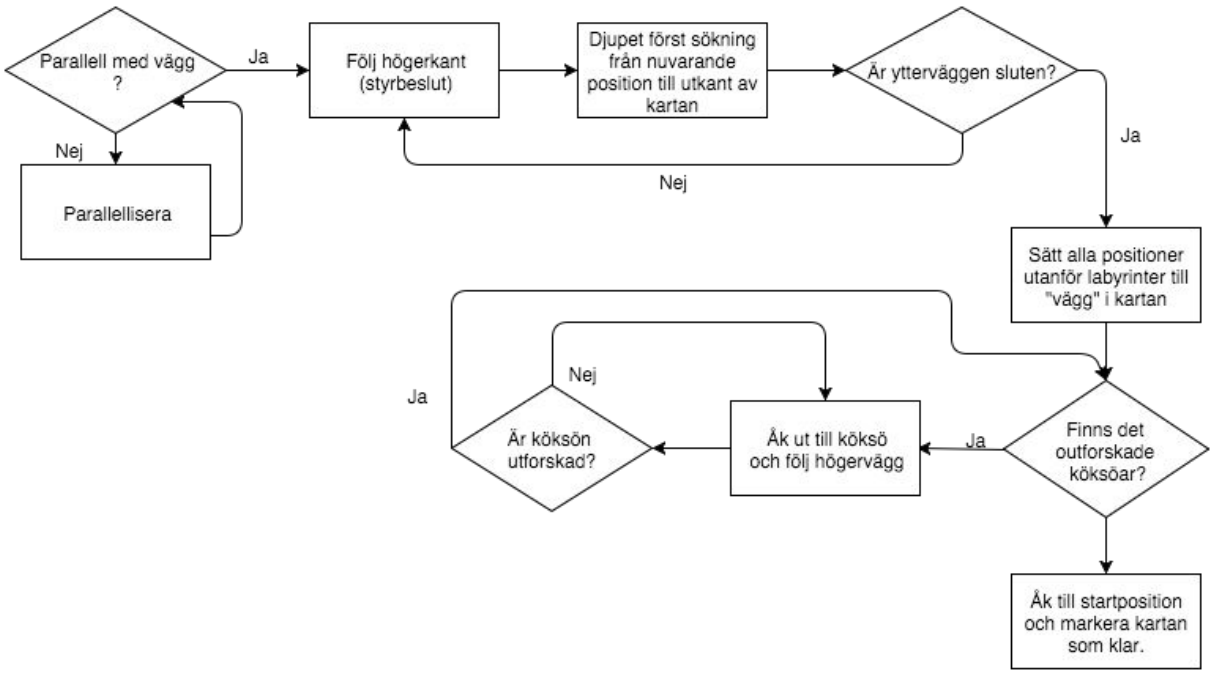


Bild 10. Flödesschema för styralgoritmen.

När den har ritat ut ytterväggen så ska den hitta köksöar i banan. I och med att alla innerväggar vi sett har markerats i kartan så görs helt enkelt en sökning efter närmsta innervägg. M/S Sea++ följer sedan högerväggen tills den ser en vänstervägg igen. Därefter traverserar M/S Sea++ avståndet till köksön för att sedan följa köksöns högervägg ett helt varv. Sedan återupprepas processen med att hitta nästa innervägg och följa den tills alla köksöar är undersökta och det därmed inte finns några innerväggar kvar. Då återstår bara att följa ytterväggen tillbaka till startpositionen. Algoritmen visualiseras i Bild 11 nedan.

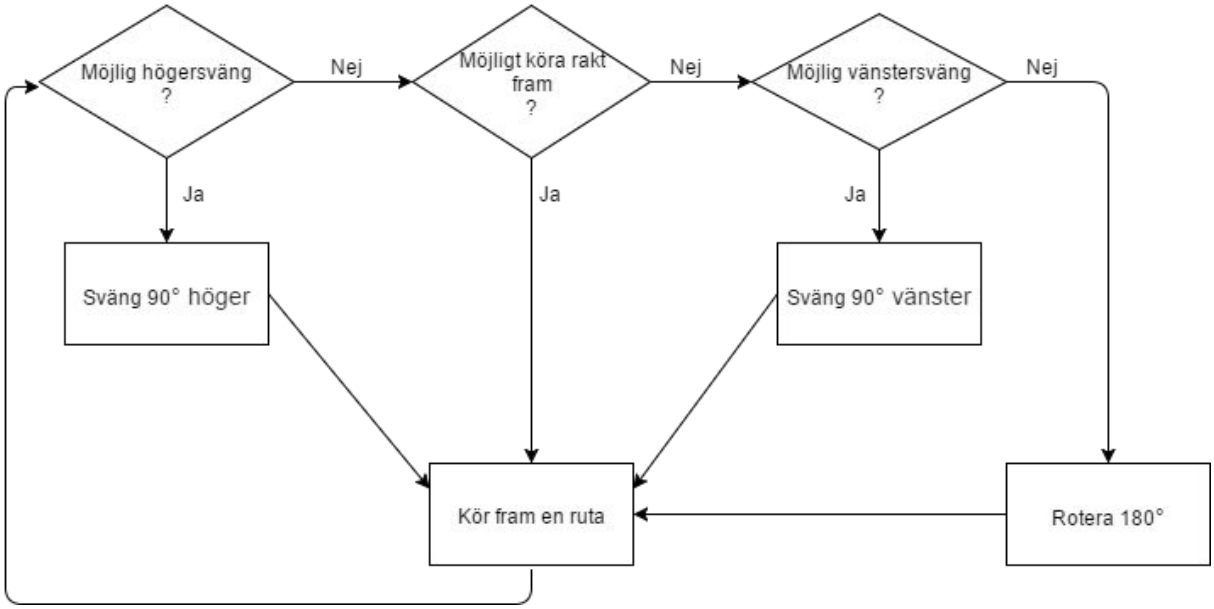


Bild 11. Flödesschema över styrbeslut för "följ högerkanten".

5.4. Reglering

Styrenheten implementerar reglering av styrningen så att roboten rör sig med mjuka och jämna rörelser när den åker längst med en vägg. För att uppnå detta används PD-reglering. Den proportionerliga delen fungerar som så att Bjarne multiplicerar skillnaden mellan det nuvarande avståndet från väggen och det önskade avståndet med en konstant. Den deriverade delen kollar skillnaden mellan den främre ir-sensorn och den bakre och multiplicerar detta med en annan konstant. Det är dessa konstanter som bestämmer hur regleringen ska fungera och valet av konstanterna utgör utmaningen i att få en bra regleralgoritm. Sedan adderas dessa tal och beroende av dess värde så adderas eller subtraheras denna summa från motorernas hastighet.

Från sensorerna i sensormodulen kommer tokendata som bedömer ifall systemet står parallellt med väggen, se avsnitt 6.1.1, vilket används vid start och efter alla vänster rotationer. Styrmodulen hämtar också sensordata för tillryggalagd sträcka från sensormodulen för att bedöma när systemet åkt 40 cm, då ska det göras ett nytt styrbeslut.

5.5. Styrlogik

Ett gränssnitt är implementerat för att enkelt kunna översätta inkommande styrkommandon till motorrörelser. Detta gränssnitt tar fyra argument som indata; spd_left, spd_right, dir_left, dir_right. Båda spd-argumenten är heltal mellan 0 och 100 vilka sedan mappas mot 0-1024 och används sedan som referenstal till PWM-signalen. DIR är true/false-värden och styr vänster respektive höger motors riktning.

- För att åka rakt framåt (100,100,1,1)
- För att snurra åt vänster (100,100,0,1)

5.6. Knappar

Två knappar och en brytare har monteras på systemet. Brytaren bestämmer om systemet körs autonomt eller manuellt. De två knapparna är resetknappar som kommer vara låga tills en stund efter systemstart. Detta är implementerat enligt Bild 12. Komponentvärden är 100 nF på kondensatorn och 4,7 kΩ på resistorn.

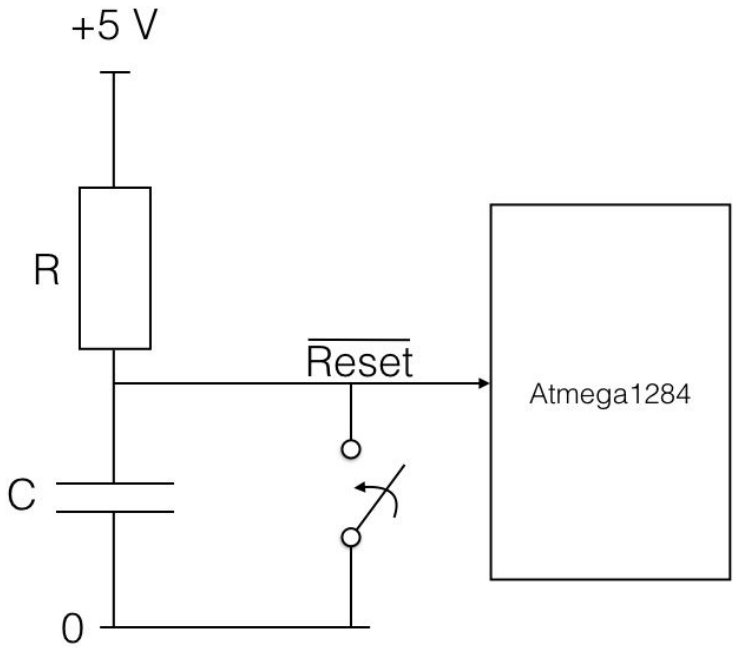


Bild 12: Bild på implementationen av resetkanppen.

6. Sensormodul - Mimers brunn

Mimers brunn har till uppgift att samla in data från de olika sensorerna och filtrera/omvandla datat till ren/användarvänlig form. Sensorenheten består av ett ATmega1284p-kort.

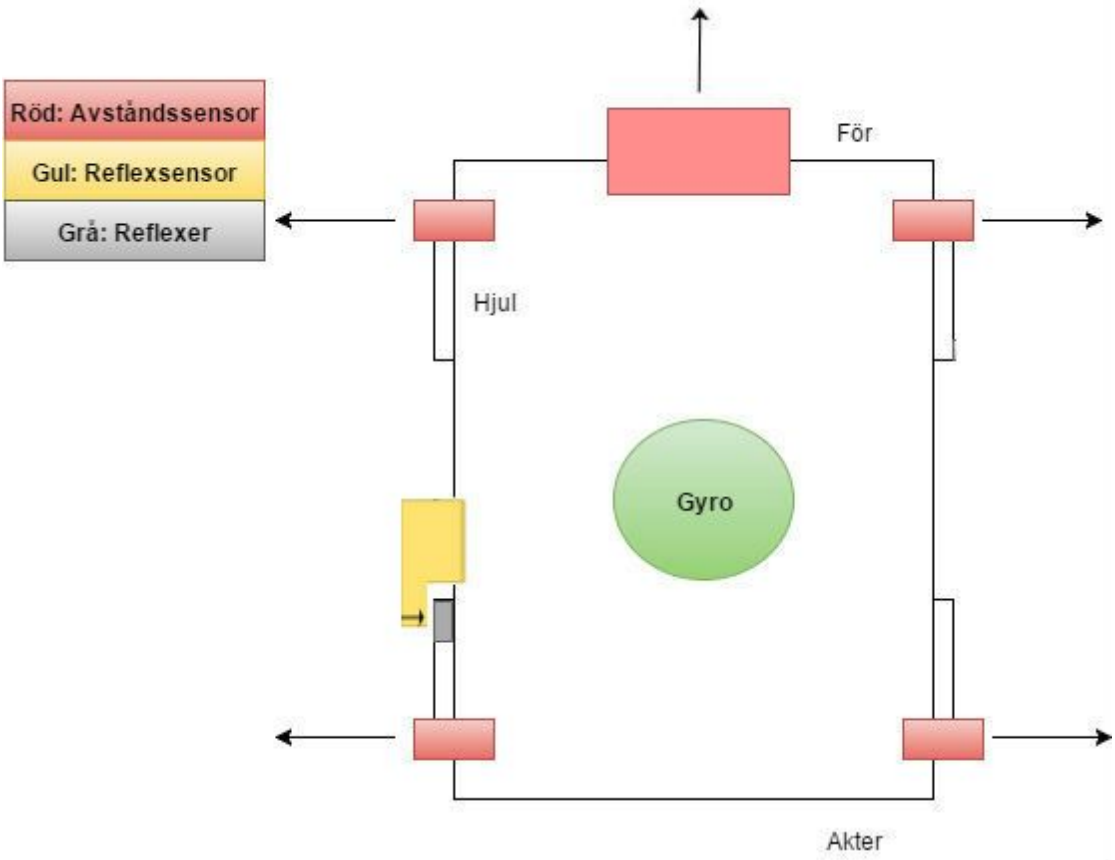


Bild 13. Sensorerna och deras placering på roboten

6.1. Sensorer

I modulen används tre olika typer av sensorer, se Bild 13:

- Hugin och Munin, IR-sensorer som mäter avståndet till väggarna
- Gyroskop, som mäter rotationen av roboten
- Hel, reflexsensor för att mäta tillryggalagd sträcka

6.1.1. Hugin och Munin (IR-sensorer)

Avståndet till väggarna mäts med hjälp av 5 st IR sensorer av typ GP2Y0A21. Sensorerna är monterade så att två är åt vänster, två åt höger och en rakt framåt. Dessa sensorer lämpar sig väl för avståndsmätning mellan 10 och 80 cm. Sensorerna matas med 5 volt till Vcc och de skickar ut en analog signal V_0. Sensorerna har en uppdateringsfrekvens på 39 ms.

6.1.2. Gyro

För att kunna veta när roboten svängt 90°, utan att behöva bry sig om eventuellt olika slitage på motorerna och slirning etc, används gyroskopet MLX90609. MLX90609 skickar ut den A/D omvandlade vinkelhastigheten till processorn via SPI, se avsnitt 2.1.1 för mer information om detta.

Vid uppstart kalibreras gyrot för att beräkna den referensspänning gyrot har i vila. Det görs genom att sampla gyrot 16 gånger och beräkna medelvärdet.

För att svänga samplar vi data från gyrot med bestämd frekvens och beräknar differensen mellan det aktulla värdet och referensvärdet vilket summeras tills en konstant är uppnådd. Konstanten varierar beroende på om det är höger, vänster eller 180°. När rotationen är klar meddelar sensormodulen styrmodulen att rotationen är klar.

6.1.3. Hel (Reflexsensor)

För att kunna mäta tillryggalagd sträcka finns en reflexsensor monterad på vänster bakhjul. På hjulet finns svarta och vita segment utsatta så reflexsensorn kan upptäcka ljusförändringar. Utsignalen är en analog spänning som motsvarar hur stor reflektion underlaget ger. Då segmenten är vita/svarta blir det stora skillnader mellan dem och det är lätt att få ett bra resultat. Segmenten summeras och översätts till cm.

6.2. IR och Filtrering

Signalen från IR-sensorerna skickas genom ett lågpassfilter, för att filtrera bort störningar, bestående av följande komponenter: 1 resistor á 18 kohm och 1 kondendsator á 100 nF, detta ger en brytfrekvens på ungefär 88.42 Hz. Bild 14 visar kopplingsschemat över lågpassfiltret.

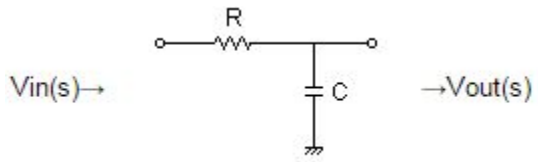


Bild 14. Lågpassfilter.

Den filtrerade signalen skickas sedan vidare genom ATmega1284p-kortets A/D-omvandlare.

I fallet med signalen från IR-sensorerna behöver processorn omvandla den icke-linjära signalen från volt till cm. Detta görs genom att vi har en tabell med voltvärden och dess motsvarande avstånd och använder den för omvandlingen.

6.3. Sensorenhetens implementation

Sensorernas värden läses ut kontinuerligt av sensormodulen i ordning. Detta gör så att begärd data kan sändas direkt vid begäran från styrmodulen. Detta sker via USART, se avsnitt 2.1.4. De olika datavariablerna listas i Tabell 2.

6.4. Tokenfiering

För att enkelt tolka IR-sensorernas data görs den om till två bitar, antingen 10 om vägg närmast, 01 om vägg ett steg bort, annars 00 vilket innebär två rutor med golv. Bitarna läggs i en byte. För att skapa token om hur parallellt roboten står tas värdet för främre sensorn subtraherat på det för bakre sensorn. Är differensen tillräckligt nära 0 anses systemet vara parallellt med väggen. Styrmodulen använder denna data för att reglera längs väggen. Antalet cm för de olika värdena är olika för sensorn fram och sensorerna mot sidorna för att optimera algoritmen.

Reflexsensorns data kontrolleras och tokenvärdet adderas med 1 ifall värdet ändrades. När styrmodulen har åkt önskad sträcka nollställs antalet åkta segment.

Gyroskopets token är normalt 0 och sätts till 1 när rotationen är klar. När styrmodulen fått kommandot att rotationen är klar skickar den ner en återställningssignal som nollställer gyrotoken.

7. Komponentlista

Tabell 3. Komponentlista

Namn	Antal	Beskrivning
Allmänt		
Chassi	1	4-hjuligt Terminatorchassi till roboten
Virkort	2	Ett virkort per modul
Tryckknappar	2	Avstudsad tryckknappsmodul för att återställa modulerna se Bild 12
Brytare	1	För att växla mellan det manuella och autonoma läget
Kommunikation		
AVR-ATmega1284	1	Mikrokontroller
Firefly kort	1	För Bluetoothkommunikationen
Sensorer		
AVR-ATmega1284	1	Mikrokontroller
Reflexsensor	1	För att mäta åkt sträcka genom att mäta ljusskillnader på ett hjul
IR sensor	5	För att mäta avståndet till väggarna
Lågpassfilter	6	Filtrera signalen från IR-sensorerna och reflexsensorn
GYRO	1	Mäter förändring i rotation, för att kunna svänga korrekt
Styrning		
AVR-ATmega1284	1	Mikrokontroller

8. Slutsatser

Vi har byggt en robot som autonomt kan kartlägga ett rum och åka tillbaka till sin startposition samt styras manuellt. Systemet fungerar idag men många förbättringar skulle kunna göras. Mjukvarumässigt kan en hel del kod optimeras för att systemet ska flyta på bättre.

Exempelvis kan reglering, positionering, framställning av sensorvärden, rotationer, svängar och liknande finjusteras. En stor förbättring är att göra den autonoma algoritmen bättre. Exempelvis genom att kunna följa vänsterväggar eller åka utan att följa vägg för att därmed kunna kartlägga mitten av kartan snabbare.

Hårdvarumässigt skulle mer precisa IR-sensorer, snabbare processor samt snabbare A/D-omvandlare bidra till en bättre robot.

9. Referenser

1. PyBluez, hämtat 2015-12-16: https://github.com/karulis/pybluez
2. Pygame, hämtat 2015-12-16: http://www.pygame.org/wiki/about

Appendix A - Kopplingschema