SYSTEMSKISS

Jesper Otterholm

Version 1.0

Status

Granskad		
Godkänd		

PROJEKTIDENTITET

Grupp 1, 2015-VT
Tekniska högskolan vid Linköpings universitet, ISY

Namn	Ansvar	Telefon	E-post
Jesper Otterholm	Projektledare (PL)	073 800 03 17	jesot351@student.liu.se
Lage Ragnarsson	Dokumentansvarig (DOA)	073 972 36 35	lagra033@student.liu.se
Erik Sköld		073 905 43 43	erisk214@student.liu.se
Emma Söderström		073 396 21 72	emmso236@student.liu.se
Matilda Östlund Visén		073 817 15 90	matos000@student.liu.se
Filip Östman		072 203 33 07	filos433@student.liu.se

E-postlista för hela gruppen:jesot351@student.liu.se

Kund: Institutionen för systemteknik, Linköpings universitet
Kontaktperson hos kund: Kent Palmkvist, 3B:502, 013-28 13 47, kentp@isy.liu.se

Kursansvarig: Thomas Svensson, 3B:528, 013-28 13 68
Handledare: Olov Andersson, 3B:504, 013-28 26 58, olov@isy.liu.se

Innehåll

Dokumenthistorik

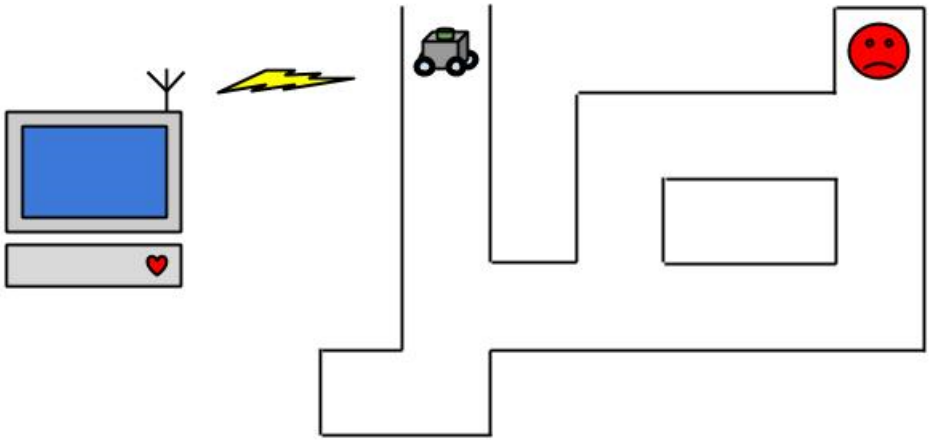
Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2015-02-16	Första utkastet	Alla	
0.2	2015-02-19	Andra utkastet efter kommentar från beställare	JO, ESk	
1.0	2015-02-23	Första versionen		

1 Inledning

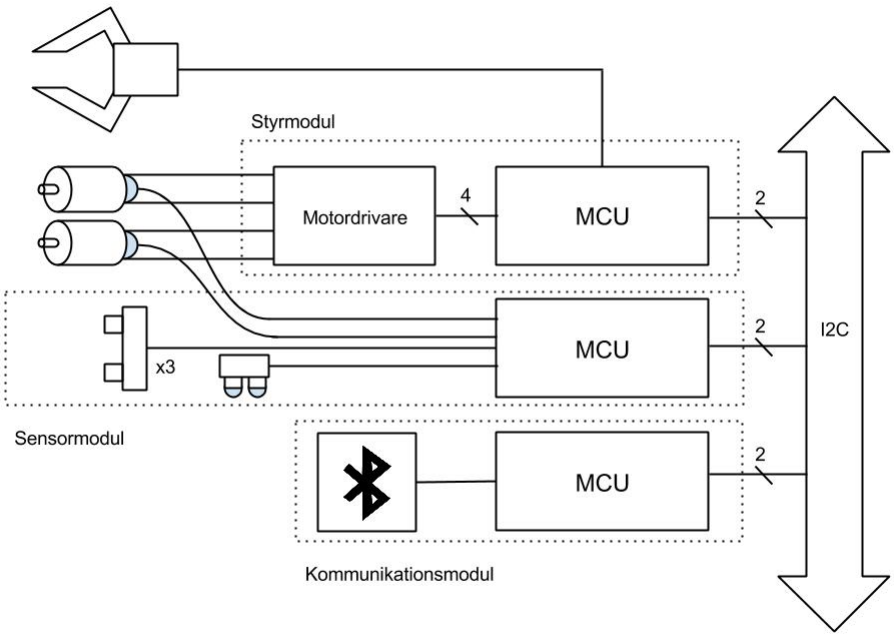
Denna systemskiss beskriver översiktligt hur en undsättningsrobot kan konstrueras med ingående moduler. Roboten ska kunna kartlägga och finna nödställda i en grotta samt hitta snabbaste vägen till och från den nödställde från grottans öppning. Det finns fyra ingående moduler: styrmodul, kommunikationsmodul, sensormodul samt PC-modul. Dokumentet ger olika förslag på ingående komponenter, design och utformning av modulerna samt argument för och emot de givna förslagen.

2 Översikt av systemet

Roboten kommer ha som uppgift att utforska ett grottsystem samt kartlägga denna och hitta en nödställd som befinner sig på godtycklig plats i grottan. Den nödställde ska förses med någon form av proviant som roboten ska hämta från startpunkten genom kortast möjliga sträcka. Robotens utforskade område, position och avlagda sträcka kommer registreras och ritas upp i ett program på en PC.



Figur 1 - Systemöversikt



Figur 2 - Blockschema

3 Chassi

3.1 Alternativ 1

Alternativ 1 är att robotens chassi kommer utgöras av en 5" cirkelskiva färdig med utskurna hål för montering av motorer och hjul samt diverse andra komponenter. Två hjul sitter på motstående sida längs skivans periferi och en, alternativt två, stödkulor används för att hålla balansen. De olika modulernas komponenter viras på 5" runda prototypkort som monteras på höjden.

Fördelen med detta chassi är att man kan svänga utan att slira på hjulen. Man kan då genom att mäta hjulen rotation uppnå mycket exakta rotationer utan att behöva ha en sensor för vinkelmätning, till exempel ett gyro. Det är också enklare att driva endast på två hjul, och för denna robot behövs inte fyrhjulsdrift som de chassin som redan finns tillgängliga.

3.2 Alternativ 2

Alternativ 2 är att använda sig av de fyrhjuliga chassin som redan finns tillgängliga. På dessa kan man montera europakort där man har virat sin konstruktion. Det finns även ytor att montera sensorer och liknande på.

4 Styrmodul

Styrmodulen är den modul som styr robotens motorer och hanterar logik för navigering. Styrmodulen kommunicerar med sensormodulen och kommunikationsmodulen.

4.1 Processor

Styrmodulen ska ha en mikrokontroller och vi tänker använda en ATmega1284. Denna mikrokontroller har relativt gott om minne och portar för kommunikation med andra moduler.

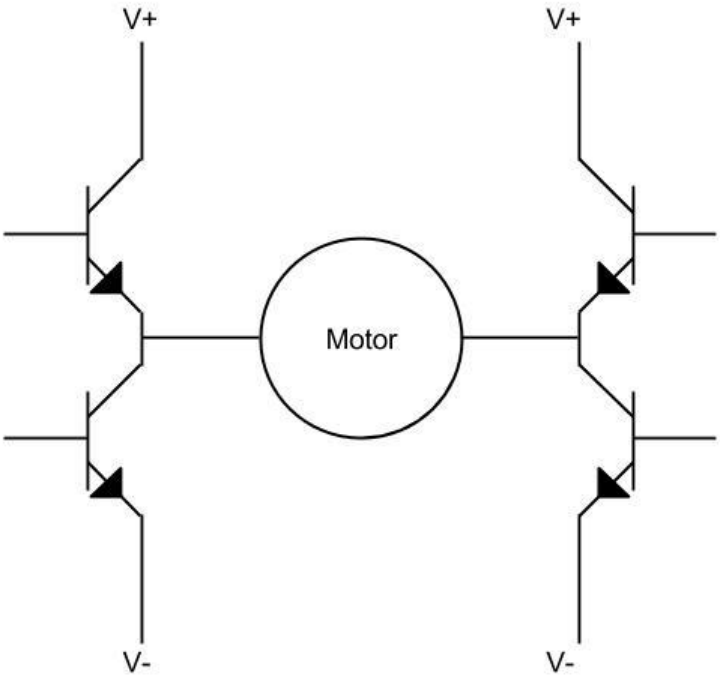
4.2 Motorer

Elmotorer finns i ett par olika utföranden: likströmsmotor (eng. brushed), borstlös likströmsmotor (eng. brushless) och stegmotor (eng. stepper).

En stegmotor har fördelen att den går att styra mycket precist. En stegmotor brukar ha några hundra steg/varv som man kan stega fram ett och ett. På så vis behöver man ingen återkoppling för att mäta hur man har rört sig, förutsatt att man har dimensionerat sin motor rätt. Om man väljer sin motor dåligt, t.ex. väljer en för svag, kan man inte garantera att precisionsstyrningen fungerar som tänkt. En nackdel med dessa är att de är ganska dyra, ofta stora och kräver en drivarkrets för att fungera.

En borstlös motor har fördelen att de är hållbara och lite effektivare än en vanlig likströmsmotor. De kräver dock också speciell drivelektronik för att fungera.

Vi tänker använda vanliga likströmsmotorer (brushed). De är rätt enkla att driva, man behöver bara lägga en likspänning över motorn. Då mikroprocessorn inte kan driva motorerna direkt behövs en enkel drivkrets. Lämpligt är att använda en (dubbel) H-brygga.

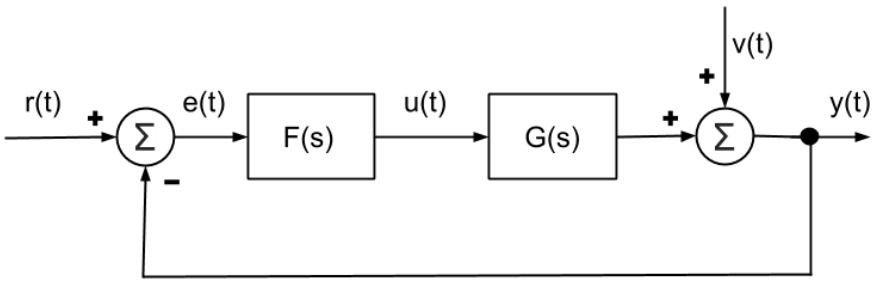


Figur 3 – Motor med H-brygga

Man måste ha tillgång till en lämplig spänningskälla som kan leverera nog med ström för att driva motorerna. Sedan kan man med mikroprocessorns utgångar kopplade till gaten på transistorerna styra strömmen genom motorn. På detta vis kan man kontrollera motorn och beroende på vilka transistorer man aktiverar kan man köra motorn åt bägge hållen. Om man pulsbreddsmodulerar dessa styrsignaler kan man få motorn att gå i olika hastigheter.

4.3 Reglering

För att roboten ska kunna navigera autonomt förutsätts att den kontinuerligt tar hänsyn till, och anpassar sig efter omgivningen. För att åstadkomma detta kommer styrenheten att reglera motorstyrningen utifrån sensormodulens mätvärden. Olika tillstånd för roboten kan behöva egna specialanpassade reglersystem. Ett blockschema över ett förenklat, återkopplat system med beteckningar kan ses i figur 3. I det generella fallet har vi en utsignal, eller mätsignal y(t) som styrs av styrsignalen u(t) genom ett system med överföringsfunktionen G(s). y(t) kan även påverkas av en störsignal v(t). Styrsignalen u(t) är utsignal till vårt reglersystem F(s) som försöker korrigera felet e(t). e(t) är differensen mellan y(t) och referensvärdet r(t).



Figur 4 – Återkopplat system

I vårt fall är flera problem så kallade regulatorproblem vilket innebär att syftet är att hålla y(t) så nära ett konstant r(t) som möjligt under inverkan av störningar i v(t). Men man kan även tänka sig en del servoproblem, det vill säga problem där systemet ska se till att y(t) följer r(t) på ett bra sätt. Till exempel system som reglerar spänningen över motorerna för att hålla en från styrmodulen önskad hastighet eller differensen i spänning över motorerna för att få en önskad vridning.

PID-reglering kan med fördel användas i alla sammanhang med eventuellt utelämnande av en eller två komponenter beroende på användningsområde. En generell beskrivning av styrsignalen beskrivs i ekvation (1).

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(\tau)d\tau + K_D \frac{de(t)}{dt} \quad (1)$$

Nedan beskrivs tre tänkbara reglersystem och konstruktionstekniska överväganden att beakta.

4.3.1 Korridorsreglering

Det är kritiskt att roboten håller sig i mitten av en korridor av flera skäl. En svängig och oförutsägbar körbana gör positionsbestämning och kartering mycket svårare, dessutom är det långsammare. För att hålla kursen i en korridor med konstant bredd kan differensen av sidosensorernas mätvärden användas som mätsignal y(t) och referensvärdet r(t) sätts då till 0. Det betyder att felet e(t) = -y(t). I detta fall kan PD-reglering användas då det statiska reglerfelet är 0 enligt slutvärdesteoremet, förutsatt att systemet är stabilt.

Om roboten utrustas med två avståndssensorer på var sida kan även vinkeln till väggen mätas och ingå i regleringsalgoritmen.

För att klara av att navigera i miljöer som inte enbart har korridorer kan man tänka sig att man istället låter avståndet till en vägg vara y(t) och reglerar detta avstånd med ett nollskilt r(t). I detta fall skulle PID-reglering kunna användas om det statiska felet anses påverka i tillräckligt stor grad. Styrsignalen för korridorsregleringssystemet beror på motorstyrningsabstraktionen, se avsnittet om drive-by-wire för exempel.

4.3.2 Finpositionering

För att kunna närma sig till exempel ett föremål som ska plockas upp kan man reglera avståndet från den framåtriktade sensorn med referensvärdet satt till det avstånd då gripklon når föremålet. I detta fall är det extra viktigt att inte få någon översläng och parametervalet får ske därefter.

4.3.3 Drive-by-wire

Det kommer att krävas någon typ av abstraktionslager mellan styrmodulens beslutsfattande och motorstyrningen. Det skulle kunna vara tillräckligt att styra genom att ange den andel av maxspänningen som önskas för respektive motor. Detta kräver att man experimentellt undersöker vilka effekter dessa har på robotens fart och rotation och skapar något i stil med ett look-up-table för olika hastigheter och vinkelutslag.

Ett annat sätt är att skapa ett så kallat drive-by-wire-system som i grunden är ett eller flera reglersystem vilka kan reglera fart eller vinkelutslag till angiven referens. Det enklaste av de två skulle vara ett reglersystem som reglerar hastigheten med en skalning av spänningen för respektive motor som styrsignal u(t) och uträknad fart som mätsignal y(t). Referenssignalen r(t) är då önskad fart. Den kanske mest användbara men också svåra delen av ett drive-by-wire system av den här typen är styrningen. Ett möjligt sätt att realisera ett steer-by-wire-system skulle vara genom att låta robotens rotation vara y(t) vilket skulle kunna räknas ut från sensormodulens pulsgivare eller med ett gyroskop. u(t) skulle då kunna vara differensen mellan spänningen över de två motorerna och r(t) önskad rotation (ej relativ). Detta skulle tillsammans med systemet för att hålla en viss konstant fart styra motorerna.

Tillsammans med korridorsregleringen skulle drive-by-wire-systemet låta navigationsdelen av styrmodulen fokusera på vart roboten ska istället för hur den ska styras dit. Till exempel kanske

roboten behöver hålla en viss fart vid kartering och en annan vid navigering av redan utforskad terräng. En hastighet kan då specificeras frikopplat från styrning och korridorsregleringssystemet kan använda steer-by-wire-systemets referenssignal som styrsignal för att svänga in sig i mitten av korridoren. Vidare skulle 90°-svängar och annat kunna abstraheras som förändringar i just denna referenssignal.

4.3.4 Parameterval för PID-regulatorer

För att uppnå önskat beteende på respektive reglersystem bör de ingående reglerparametrarna väljas med omsorg. En experimentell metod för att ta fram adekvata parametrar är Ziegler-Nicholsmetoden som går ut på att först sätta K_I och K_D till noll och sedan ändra K_P tills y(t) oscillerar med konstant amplitud. Sedan använder man detta värde på K_P och den periodtid y(t) har för att beräkna K_I och K_D. Det finns samband för att ta fram parametrarna för PI, PD och PID med olika beteende för systemet så som med minimal översläng.

4.3.5 Sampling och tidsdiskretisering

I praktiken kommer vi inte att manipulera någon typ av tidskontinuerliga signaler. Eftersom systemet är digitalt kommer både r(t) och u(t) förändras i diskreta hopp, vi kan beteckna dem som r[n] och u[n]. På sammas sätt kommer mätsignalen y(t) vara en samplad tidsdiskret signal y[n].

Förslagsvis kommer derivatan av e[n] tas fram med linjärapproximation mellan två på varandra följande mätpunkter. Här får man gå balansgång mellan att få tillförlitliga derivatatermer med säkrare, medelvärdesbildade mätpunkter och att kunna uppdatera styrsignalen tillräckligt ofta för en given hastighet. Samplingstiden T_s är i detta fall tiden sensormodulen behöver för att producera en ny mätpunkt. Beräkningen skulle kunna ske enligt ekvation (2).

$$\frac{de(t)}{dt} \approx \frac{e[n] - e[n-1]}{T_s} \tag{2}$$

Integraltermen kan tas fram med samma typ av approximation genom att senaste bidraget räknas ut med trapetsmetoden och sedan adderas till totala värdet sedan t_0 enligt ekvation (3).

$$\int_{t_0}^{t} e(\tau)d\tau \approx \int_{t_0}^{t-T_s} e(\tau)d\tau + T_s\frac{e[n-1] + e[n]}{2} \tag{3}$$

I båda uträkningarna är det viktigt att inte bara räkna utan vidare reflektion. Till exempel kan derivatan bli väldigt stor om roboten roterar på vissa sätt, framförallt i samband med hörn och korsningar i labyrinten. Av den anledningen bör man vara restriktiv med när en viss reglering sker. En stor förändring i mätvärden kanske ska förkastas eller leda till en tillståndsförändring. På samma sätt får man se upp så att integraltermen inte växer okontrollerat. Man kan tänka sig att den kan nollställas under vissa förutbestämda omständigheter.

4.4 Kartering

Roboten skall kunna rita en karta över den omgivning den kör i. Att avbilda sin omgivning och navigera efter detta kan vara ett svårt problem generellt sett, men då vi vet mycket om vår omgivning går det att förenkla. Vi tänker använda en form av så kallad "occupancy grid mapping". Denna metod bygger på att man representerar världen med ett antal rutor av en viss storlek. När man sedan åker runt i världen får man mätvärden från sina sensorer om hur långt det är till något objekt. Man kan då räkna ut vilken ruta denna mätning motsvarar och i sin karta markera den som okörbar. Beroende på rutstorlek i sin karta får man då olika upplösning.

Då vi vet mycket om vår omvärld kan vi göra logiken för kartering relativt enkel och möjlig att köra på en ganska enkel mikrokontroller. Vi kan till exempel använda samma rutstorlek som banan kommer vara uppbyggd av, då högre upplösning än så endast ökar minnesanvändningen och beräkningskomplexiteten.

En möjlig strategi för att kartera en okänd omgivning skulle kunna vara:

1. Börja med en tom karta.
2. Kör i korridoren och uppdatera kartan efter vad du ser.
3. När en korsning dyker upp, fatta ett vägvalsbeslut. Detta kan vara alltid åt ett håll, slumpmässig riktning eller något annat.
4. Om man hamnar i en återvändsgränd eller hamnar på ett ställe man redan varit, lista ut snabbaste vägen till en outforskad ruta och fortsätt där.
5. Fortsätt tills alla rutor i labyrinten är utforskade.

För att denna karteringsmetod ska fungera måste man på något vis kunna beräkna sin position i omgivningen med ganska bra noggrannhet. Att positionera sig mitt i en korridor kan ett reglersystem sköta och att mäta hur långt hjulen har rullat anger när man har flyttat sig från en ruta i kartan till en annan.

4.5 Kortaste vägen

Vid ett antal tillfällen kommer vi vilja beräkna kortaste vägen från en punkt i labyrinten till en annan. Detta kan vara då vi karterar och vill åka till ett outforskat område, när vi ska tillbaka till målet efter att ha hittat den nödställde och eventuellt en ny beräkning för att åka tillbaka till den nödställde.

Kartan kan representeras som en graf och kommer att vara i storleksordningen 15*15 noder. För att beräkna kortaste vägen finns ett antal välkända algoritmer att undersöka. Den kanske mest kända är Dijkstras algoritm, andra vanligt förekommande alternativ är A* och D*-lite. Under avsökningsfasen kan man tänka sig att algoritmen får köras allt eftersom nya noder avsöks och att outforskade noder antas vara tillgängliga för körning tills motsatsen är bevisad.

Eftersom att den kortaste vägen inte nödvändigtvis behöver vara den snabbaste för roboten bör man undersöka om grenkostnader kan genereras utifrån antagandet att det går långsammare att svänga än att köra rakt fram. Detta bör beaktas både i avsöknigsfasen och transportfasen.

4.6 Autonom körning

Ett vanligt sätt att hantera beslutsfattandet vid autonom körning är att implementera en finit tillståndsmaskin som kan konstrueras som en Moore- eller Mealy-maskin. Beslutsfattande kommer då att ske utifrån aktivt tillstånd och sensordata. Till exempel kan vissa typer av reglering kopplas av och på i olika tillstånd, kartering kan aktiveras och inaktiveras, hastighet kan förändras med mera.

Följande sammanhang skulle kunna vara tillräckligt unika för att utgöra ett eget tillstånd:

- Korridorskörning under utforskning
- Korridorskörning i redan utforskad miljö
- Svängning i korsning under utforskning
- Svängning i korsning i redan utforskad miljö
- Vändning i återvändsgränd under utforskning
- Körning rakt fram i korsning under utforskning
- Körning rakt fram i korsning i redan utforskad miljö
- Har upptäckt nödställda
- Plocka upp förnödenheter
- Lämna av förnödenheter

4.7 Gränssnitt

Styrmodulen ska kunna ta emot meddelanden från kommunikationsmodulen vid manuell styrning samt för justering av reglerparametrar. Styrmodulen kommer skicka meddelanden till kommunikationsmodulen om sin aktuella kartrepresentation och tänkta väg under autonom styrning. Från sensormodulen kommer styrmodulen att ta emot mätvärden från alla sensorer kontinuerligt. Man kan tänka sig att den skulle kunna ställa in samplingsfrekvens och liknande parametrar för sensormodulen också.

5 Sensormodul

Sensormodulen är den del som läser av systemets samtliga sensorer och bearbetar erhållen data så att den utefter ett specificerat gränssnitt kan kommuniceras till och användas av övriga delsystem.

De sensorområden som är aktuella i vårt fall är positionsbestämning, avståndsmätning samt identifiering av tejpremsor på underlaget.

5.1 Mikrokontroller

Då omvandling av rå sensordata till mätresultat för användning av de andra modulerna kan komma att involvera stora look-up-tables väljs ATMega1284p som mikrokontroller på grund av dess större flashminne. ATMega1284p har dessutom fler externa avbrottspinnar vilket kan komma väl till hands då flera av sensorerna kommer samplas med hjälp av avbrott.

5.2 Positionsbestämning

För att avgöra robotens position krävs både beräkning av avlagd sträcka och kännedom om robotens orientering under färdens gång. Vi listar här två aktuella alternativ för beräkning av dessa storheter.

5.2.1 Pulsgivare

En pulsgivare genererar pulser då motoraxeln den monteras på roterar. Två pulser, 90 grader ur fas, används för att bestämma både rotationshastighet och rotationsriktning. Här finns två alternativ att tillgå: en magnetisk och en optisk variant. Den magnetiska pulsgivaren har en inbyggd komparator med hysteres vilket gör att den genererar en digital utsignal. Den optiska saknar detta och ger istället en sinusformad signal vilket betyder att vi själva kommer behöva montera/konstruera en schmittrigger. De digitala pulserna kommer användas för att generera avbrott hos processorn som då kan uppdatera robotens position.

Utförliga beräkningar kan göras för att integrera fram en absolut position och orientering, men då detta är högst känsligt för ackumulerande fel är troligen en enklare beräkning kombinerat med successiv kalibrering med hjälp av avståndsmätning till omgivningen ett lämpligare alternativ.

Med givare som genererar 12 pulser per varv, motorer med utväxling 50:1 och hjul med en omkrets på ca 10 cm ges en puls ungefär sex gånger per millimeter. Detta är troligen mer än vad som behövs och för att stabilisera beräkningarna bör positionsuppdateringen göras först efter att ett antal steg från någon av givarna har registrerats.

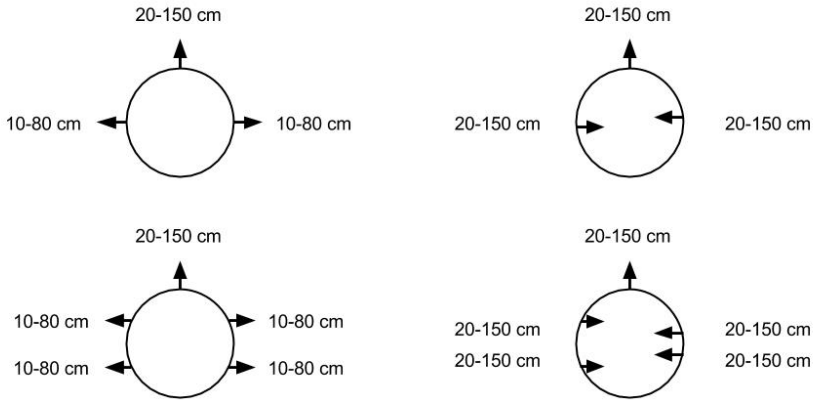
5.2.2 Tröghetsnavigering

Ett alternativt sätt för positionsbestämning är tröghetsnavigering. Här används accelerometrar och gyroskop för att bestämma sensorns, och därmed robotens, läge. Sensorn mäter förändringar i acceleration och för att erhålla en absolut position måste således mätresultaten integreras två gånger. För att minimera felet i beräkningarna måste mätningarna från accelerometrarna och gyroskopen

sammanvägas i så kallad sensorfusion. Denna operation kan ta tid att implementera och är dessutom relativt beräkningsintensiv, vilket ställer högre krav på processorkraft än alternativet med pulsgivare.

5.3 Avståndsmätning

För navigering och kartering mäts avstånd till omgivande väggar. För detta planerar vi använda tre eller fyra IR-sensorer, med olika arbetsområden, riktade åt olika håll. Några tänkbara konfigurationer illustreras i figur 4.



Figur 5 – Tänkbara sensorkonfigurationer

IR-sensorerna genererar analoga spänningar som först måste AD-omvandlas och därefter konverteras till avstånd i millimeter. Konverteringen från spänning till avstånd är gravt olinjär [1], [2] och med tanke på den begränsade beräkningskraft som är tillgänglig skapas lämpligen så kallade look-up-tables från vilka aktuella värden interpoleras fram.

Sensorerna har en uppdateringsfrekvens på ca 25 Hz [1], [2] och de samplas lämpligen från något slags timer-baserat avbrott.

5.4 Identifiering av tejpremsor

För att identifiera tejpremsor utlagda på underlaget planeras en reflexsensor att användas. Denna sensor, bestående av en infraröd lysdiod och en ljuskänslig transistor, placeras på robotens framkant och alldeles ovanför underlaget. Utsignalen är en analog spänning som blir lägre ju mer ljus som reflekteras.

För att säkerställa att tejpen upptäcks måste sensorn samplas med en frekvens större än 100 Hz (förutsatt en maximal körhastighet på 1 m/s och en 1 cm bred tejp). Detta kan lösas med timer-baserade avbrott men kan även tyckas vara slöseri med beräkningskraft då det är extremt sällsynt att en tejp påträffas. Ett alternativ kan vara att utnyttja en analog komparator för att generera ett avbrott. För att sensorn ska kunna kalibreras för aktuella förhållanden behöver den analoga referensspänningen till komparatorn kunna väljas från mjukvara. Detta kan till exempel åstadkommas med en filtrerad pulsmodulerad signal.

5.5 Gränssnitt

Sensormodulen ska sammanställa avståndsmätningar till omgivningen och avlagd sträcka, och skicka denna data, mätt i millimeter, till övriga moduler.

Kommando för att kalibrera reflexsensorn ska kunna tas emot varpå en ny referensspänning sätts.

6 Kommunikationsmodul

Den trådlösa kommunikationen mellan PC-modulen och robotens olika delmoduler går genom kommunikationsmodulen på roboten. Detta skulle kunna genomföras med en Bluetoothanslutning. PC-modulen har en Bluetooth-dongle alternativt inbyggd Bluetooth i en bärbar PC och kommunikationsmodulen har en Bluetoothmodul.

6.1 Bluetooth

Bluetoothmodulen som kommer användas heter FireFly och har en räckvidd upp till 100m [3].

6.2 Mikrokontroller

ATmega168 är ett lämpligt val för denna modul. Mikrokontrollern har tillräckligt med pinnar för att ansluta Bluetoothmodulen och har I2C anslutning. Den är även fysiskt mindre än dess syskon ATmega1284 och ATmega16 och tar således inte upp lika stor yta på virkortet.

6.3 Data som går genom modulen

Från	Till	Data
Sensormodulen	PC-modulen	Bearbetade sensorvärden
Styrmodulen	PC-modulen	Positionering, kartdata
PC-modulen	Styrmodulen	Manuell styrdata från användaren. Ex. styrning
PC-modulen	Sensormodulen	Kalibrera

Data som kommer från robotens andra moduler skickas vidare till PC-modulen via Bluetooth. Data som kommer från PC-modulen skickas vidare till robotens övriga moduler. Se intern kommunikation för datatransport mellan modulerna. Eventuellt kommer kommunikationsmodulen inte skicka något direkt till sensormodulen utan istället skickas till styrmodulen för att sedan skickas vidare. Detta gäller även åt andra hållet. Data från sensormodulen kommer troligtvis skickas från styrmodulen eftersom den ändå tar del av sensorvärdena.

6.4 Gränssnitt

Kommunikationsmodulen blir en mellanhand mellan PC-modulen och styrmodulen och ser till att all information kommer dit den ska. Här kommer inte data bearbetas utan ansvaret med sammanställning och tolkning av data får PC- respektive styrmodul göra.

7 PC-modul

PC-modulen är den modul som kommer hantera GUI, skicka styrsignaler, kalibrera sensorer och ta emot sensorvärden. PC-modulen tar emot data från sensormodul och styrmodul samt skickar information till styrmodulen i manuellt läge (se 6.3). PC-modulen kommer vara utrustad med en Bluetooth USB adapter för kommunikation med roboten.

7.1 Programmeringsspråk

Det finns flera alternativ till programmeringsspråk för utveckling av GUI och bakomliggande (övriga?) funktioner. Det mest fördelaktiga programspråket att använda är ett projektgruppen har kunskaper och erfarenhet inom. Nedan beskrivs två alternativ som båda är relativt kända för projektgruppen.

7.1.1 C/C++

Ett alternativ är att skriva koden i C++. Fördelarna är att det är ett mycket kraftfullt språk med mycket finesser som kan komma till användning och att koden exekveras fortare. Det kan däremot finnas så mycket finesser att det blir svårt att programmera i.
Det finns flera olika bibliotek att välja mellan. Bland annat

- QT
- SDL
- SFML
- GTK
- Visual Studio (MFC)

7.1.2 Matlab

Koden kan även skrivas i Matlab, och fördelen med detta är att ett GUI-bibliotek redan finns inkluderat i installationen. Tröskeln för att komma igång är lägre, men en stor nackdel är att GUI program körs mycket långsammare och önskad realtidsuppritning av kartan skulle därmed troligtvis lida.

7.2 GUI

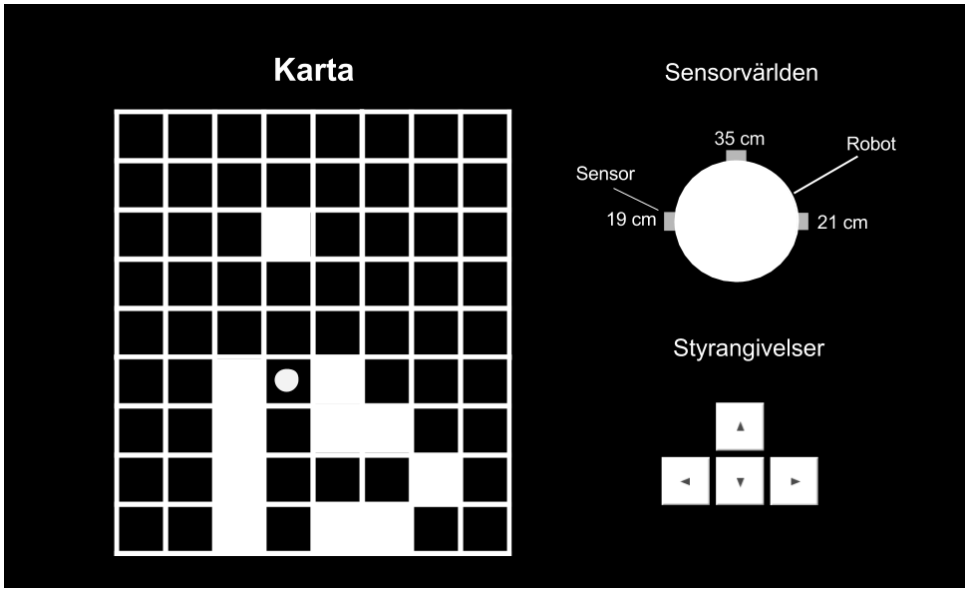
Det grafiska användargränssnittet är det fönster som användaren interagerar i medan programmet körs.

Det som måste finnas i programfönstret är:

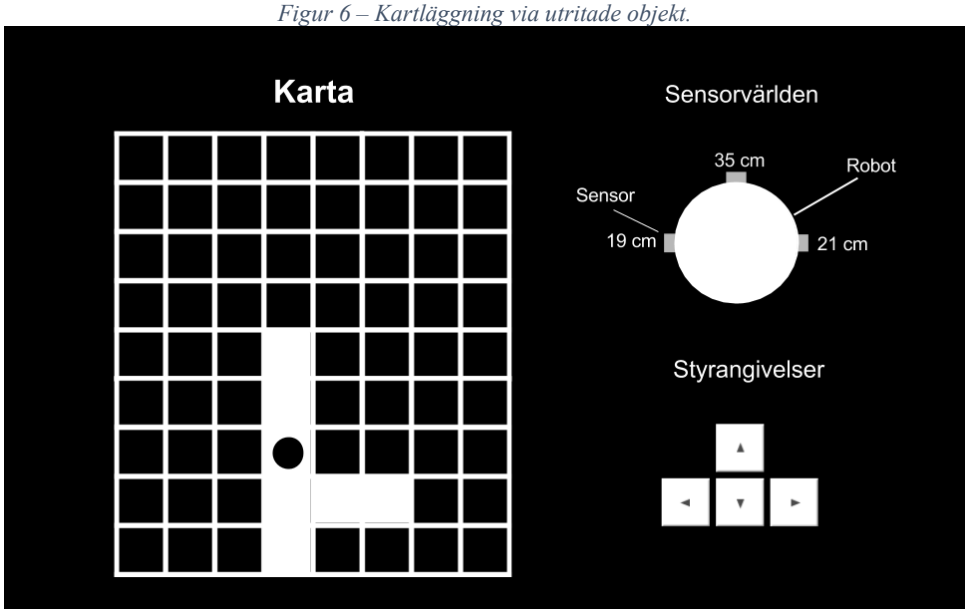
- Ett område som fylls i allteftersom roboten utforskar grottsystemet för att representera väggar och utforskat område.
    - Start och mål ska markeras tydligt
    - Den nödställde ska markeras på rätt plats tydligt.
- Sensorvärden för de sensorer som finns på roboten.
- Möjlighet till manuell styrning via antingen tangenter eller mustryckningar. Det ska klart framgå om roboten befinner sig i autonomt eller manuellt läge.
- Möjlighet att ta del av de värden som samlats in av de andra modulerna i felsökningssyfte.

-

7.2.1 Grafiska exempel



Figur 6 – Kartläggning via utritade objekt.



Figur 7 – Kartläggning via utritade väg/ruttt.

I figur 6 och 7 visas två olika GUI förslag på hur kartläggningen av robotens framfart i labyrinten sker. De skiljer sig ganska markant i hur kartan ritas ut. I den första GUI:n i figur 6 anges det att objekt som sensorerna känner av, exempelvis väggar, är det som markeras. Upptäckta väggar markeras med vita rutor medan fria eller okända rutor visas med svarta rutor. I figur 7 anges det istället att det är robotens utforskade område som målas upp eftersom datan från sensorerna bearbetas. Labyrinten kommer alltså växa fram i form av vita rutor medan väggar och okända områden kommer markeras med svarta rutor.

Fördelen med att använda sig av kartläggningen i figur 6 är att när själva objekten målas ut eftersom, blir labyrintens väggar snabbt synliga vilket tydliggör ej möjliga vägar. Nackdelen med detta är att robotens framfart, upptäckta fria områden, blir inte lika tydligt om man jämför med kartläggningen i figur 7. Där robotens väg och utforskade områden lättare blir synliga.

Roboten kommer att markeras med en vit eller svart prick på kartan samt kommer den nödställde fördelaktigt att markeras med en röd prick (eller en röd ruta).

Uppe i högra hörnet ritas roboten ut med de olika avståndsangivelser som sensorvärdena ger på respektive sensor. Sensornas avståndsangivelser ska också visas i realtid och anges i centimeter. För övrigt finns också piltangenter i det nedre högra hörnet, vilka visar körriktningen/knapptryckningarna i manuellt läge genom att bli gråaktiga.

7.2.2 Ytterligare funktioner

Ytterligare funktionella delar i det grafiska användargränssnittet skulle kunna vara:

- Visa robotens nuvarande hastighet
- Möjlighet att ta del av samt plotta sensor- och reglerdata som skickas mellan modulerna.
- Markering av något slag om roboten befinner sig närmare en vägg än önskat. Exempelvis att motsvarande sensor som ger det korta avståndet blinkar rött eller att den sidan på roboten blinkar rött i programmet.
- Möjlighet att ändra reglerkonstanter.
- Nodkostnader för sökalgoritmen
- Sträck som visar robotens kortaste väg tillbaka till start.
- Möjlighet att visa bussaktivitet

En del av dessa funktioner skulle då fördelaktigt kunna visas genom att via huvudfönstret (med kartan, sensorvärlden etc.) ta sig vidare via en knapp "extra funktioner", alternativt en ny flik. Dessa funktioner i programmet skulle kunna vara av intresse vid exempelvis felsökning samt optimering av robotens rörelse.

7.3 Manuellt och autonomt läge

I autonomt läge ska alla funktioner som kan påverka styrning och dylikt inte gå att ändra via PC:n. Karta och sensorvärden ska däremot visas och det ska tydligt framgå för användaren att roboten befinner sig i autonomt läge.

Manuellt läge innebär att roboten styrs från PC:n via piltangenterna. Styrsignalerna skickas till roboten via Bluetooth

7.3.1 Designförslag för visning av manuellt och autonomt läge

Ett alternativ är att det i autonomt läge endast visas själva kartan och sensorvärden för roboten i programmet, det innebär att man inte visar piltangenterna alls just i autonomt läge medan i manuellt läge så visas allt.

Ett annat alternativ är att man bara skriver i fönstret att roboten befinner sig i autonomt/manuellt läge utan att ändra något i layouten.

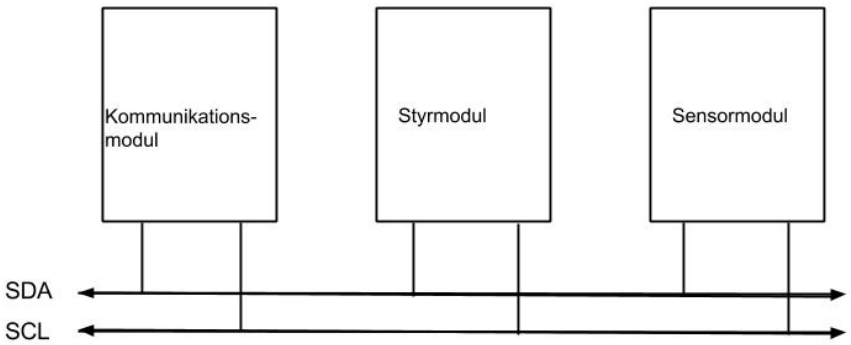
Ett tredje alternativ är att man både skriver ut autonomt läge och enbart visar karta och sensorvärden i autonomt läge och på liknande vis i manuellt läge fast man visar riktningar också.

8 Intern kommunikation

För den interna kommunikationen mellan robotens moduler finns ett antal tillvägagångssätt, bl.a I2C SPI och UART.

8.1 I2C

AVR processorn har gott stöd för I2C som består av en tvåtrådsbuss där den ena tråden består av klockan (SCL) och den andra innehåller den data som ska skickas (SDA). Data skickas seriellt. De olika modulerna ansluts med trådarna enligt figur 11.



Figur 11 - Illustration över hur modulerna delar information med I2C

För att börja skicka data på bussen finns ett startkriterium, SDA går låg när SCL är hög. Efter detta kommer adressen till inblandad modul följt av data. Stoppkriteriet är sedan när SDA går hög när SCL är hög. Det förekommer utöver detta fler bitar mellan start och stopp som ser till att all data kommer fram. Vilken modul som bestämmer över trådarna kan ske genom att det finns en eller flera masters.

8.1.1 En master

En av modulerna innehar rollen master och de andra modulerna kallas slavar. Masterns uppgift är att bestämma över klockan och vilken av slavarna som ska prata med mastern. Varje slav har en adress så mastern kan adressera dem.

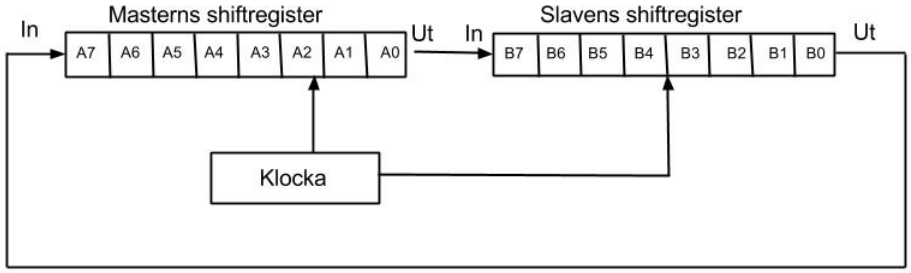
Det kommer gå mycket data från sensormodulen till styrmodulen men inte så mycket data till sensormodulen från övriga moduler. Ett förslag är då att göra styrmodulen till master som då främst får säga till sensormodulen att det är OK att skicka och sedan ta del av datan för att reglera roboten. Då skulle styrmodulen även skicka data till kommunikationsmodulen eftersom sensorvärden ska till PC-modulen.

8.1.2 Multimaster

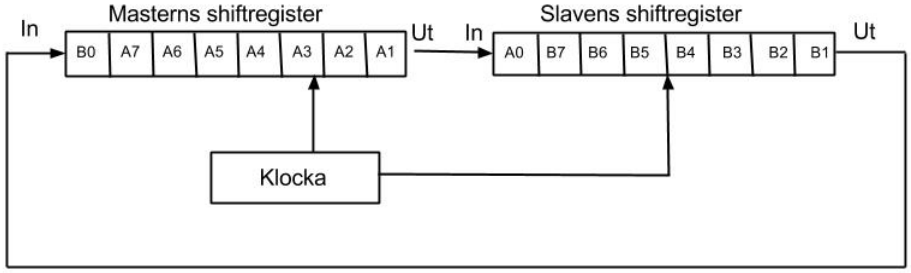
En annan variant är multimaster där ett flertal moduler är master och kan styra bussen. Ett stort problem då är att en master som vill skicka/läsa från någon modul måste kolla att bussen är ledig. Det kan göras genom att kolla att stopkriteriet har genomförts vilket signalerar att bussen ledig.

8.2 SPI

En annan möjlighet är SPI (Serial Peripheral Interface). Även denna teknik bygger på en master/slav funktion där mastern meddelar önskad slav att data ska skickas. Sedan skiftas registerna önskat antal bytes tills datan har kommit fram, se figur 12.



Figur 12 – Ursprungliga värden i masterns och slavens register



Figur 13 – Masterns och slavens register efter en skiftning

9 Referenser

9.1 Datablad

[1] Sharp Corporation, ”GP2Y0A21YK Optoelectric Device”, GP2Y0A21YK datablad, 2005

[2] Sharp Corporation, ”GP2Y0A02YK Long Distance Measuring Sensor”, GP2Y0A02YK datablad

9.2 Digitala källor

[3] ISY, ”FireFly”, *vanheden.isy.liu.se*, tillgänglig på https://docs.isy.liu.se/twiki/pub/VanHeden/DataSheets/firefly.pdf [Hämtad 20150219]