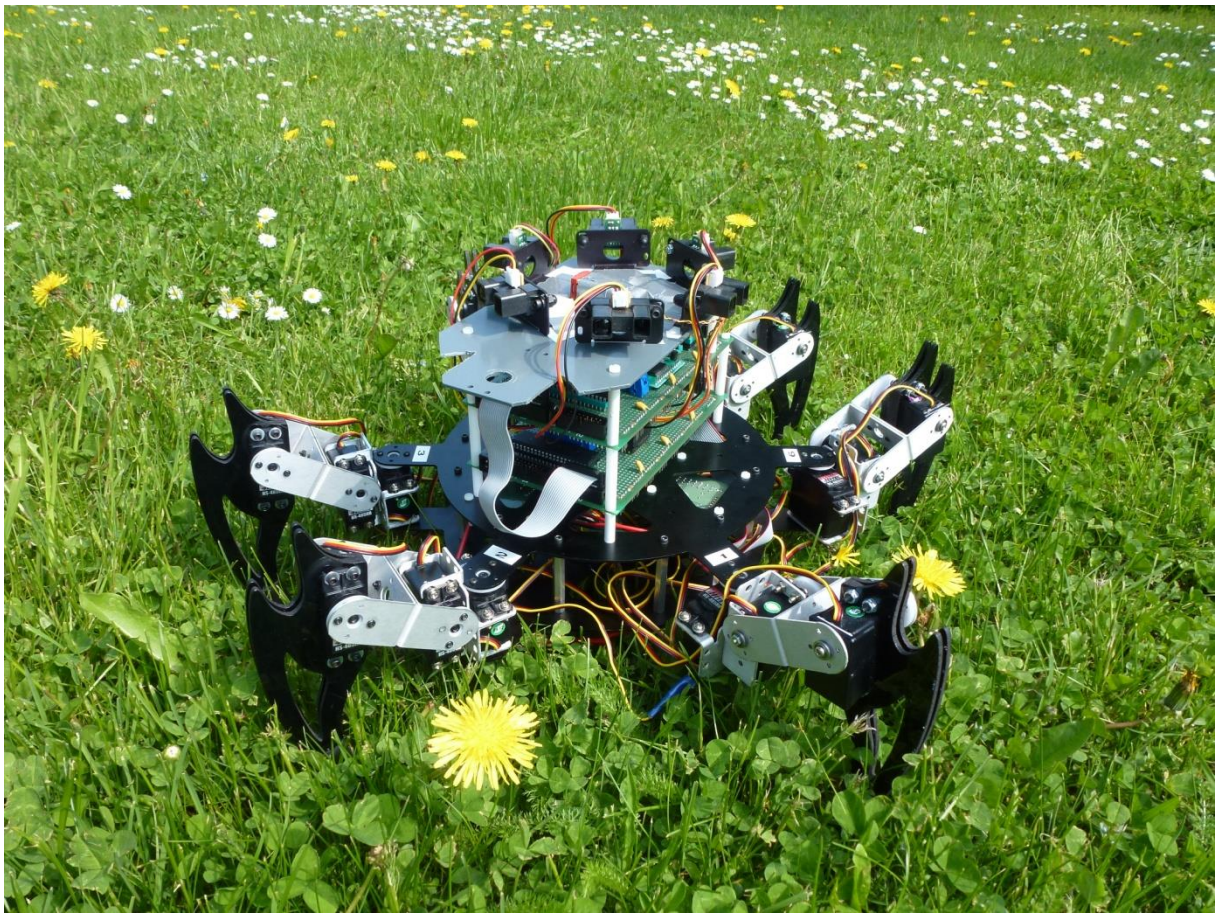
# Teknisk dokumentation

Markus Örn

**Version 1.1**



## Status

| | | |
|---|---|---|
| Granskad | 2012-05-11 | FS, AS |

# PROJEKTIDENTITET

Projektgruppsnummer 15, VT 2012, HexBot
Linköpings tekniska högskola, ISY

| Namn | Ansvar | Telefon | E-post |
|---|---|---|---|
| Markus Örn | Projektledare (PL) | 076-807 92 35 | maror634@student.liu.se |
| Fredrik Sandvik | Dokumentansvarig (DOK) | 070-482 45 37 | fresa911@student.liu.se |
| Igor Vojvodic | | 070-356 69 89 | igovo722@student.liu.se |
| Kristian Sinivaara | | 070-754 66 27 | krisi745@student.liu.se |
| Gustav Svanfeldt | | 070-659 07 95 | gussv645@student.liu.se |
| Alexander Sjöholm | | 076-225 11 74 | alesj050@student.liu.se |

**Kund:** Tomas Svensson, ISY,
kundtelefon 013-28 13 68, tomass@isy.liu.se

**Kursansvarig**: Tomas Svensson, 3B:528, 013-28 13 68, tomass@isy.liu.se

# Innehåll

Dokumenthistorik

| version | datum | utförda förändringar | utförda av | granskad |
|---|---|---|---|---|
| 0.1 | 2012-05-08 | Första utkast | MÖ, IV, FS, AS | AS, FS |
| 1.0 | 2012-05-11 | Färdig version | MÖ | AS, FS |
| 1.1 | 2012-05-18 | Förändringar efter kommentarer | Alla | |

# 1 Inledning

Projektgruppen har konstruerat en sexbent robot, som på så kort tid som möjligt presenterar en karta över sin omgivning. Kartan som ska presenteras är specificerad i *Appendix A.* Ett exempel på en tillåten bana finns i Figur 1.

Vid uppritning av karta är roboten självgående och skickar kontinuerligt data, via Blåtand, till en dator där själva uppritningen sker. Roboten går även att styra manuellt.

**Figur 1 Banexempel**

# 2 Systemöversikt

Systemet är uppbyggt av fyra delsystem; en kommunikationsenhet, en styrenhet, en sensorenhet och en PC.

### 2.1 Grov beskrivning av produkten

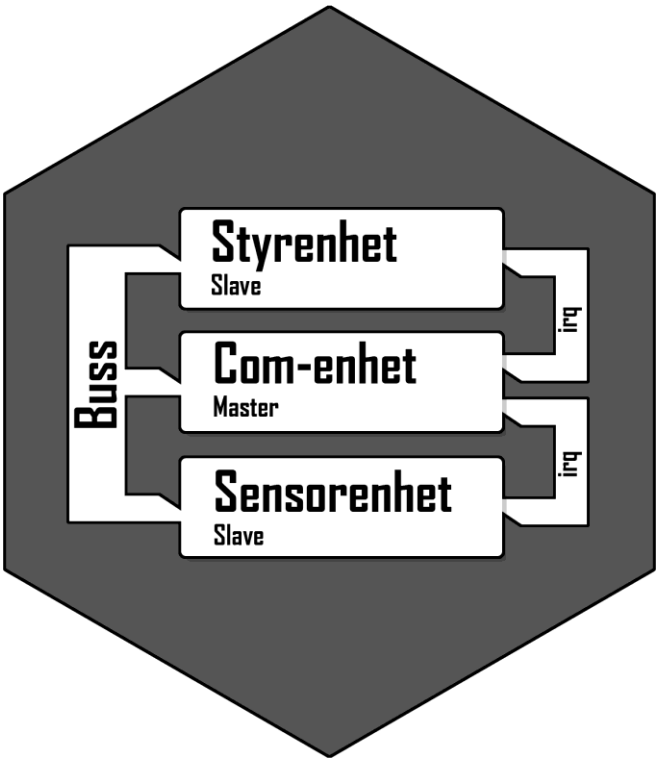
Kommunikationsenheten fungerar som talman och samordnare på roboten. Den agerar master på robotens buss och när man vill nå roboten via Blåtand från en dator, kommer kommunikation gå genom denna enhet.

Styrenheten samordnar på order av kommunikationsenheten de sex benen så att dessa lyckas förflytta roboten, i önskad riktning.

Sensorenheten mäter kontinuerligt av robotens omgivning, i autonomt läge, dels för att inte gå in i väggar men även för att kunna rita den önskade kartan.

Sensorenheten ritar även upp kartan under tiden som roboten går, och med jämna mellanrum skickar den kart- och sensordata till PC-enheten via kommunikationsenheten.

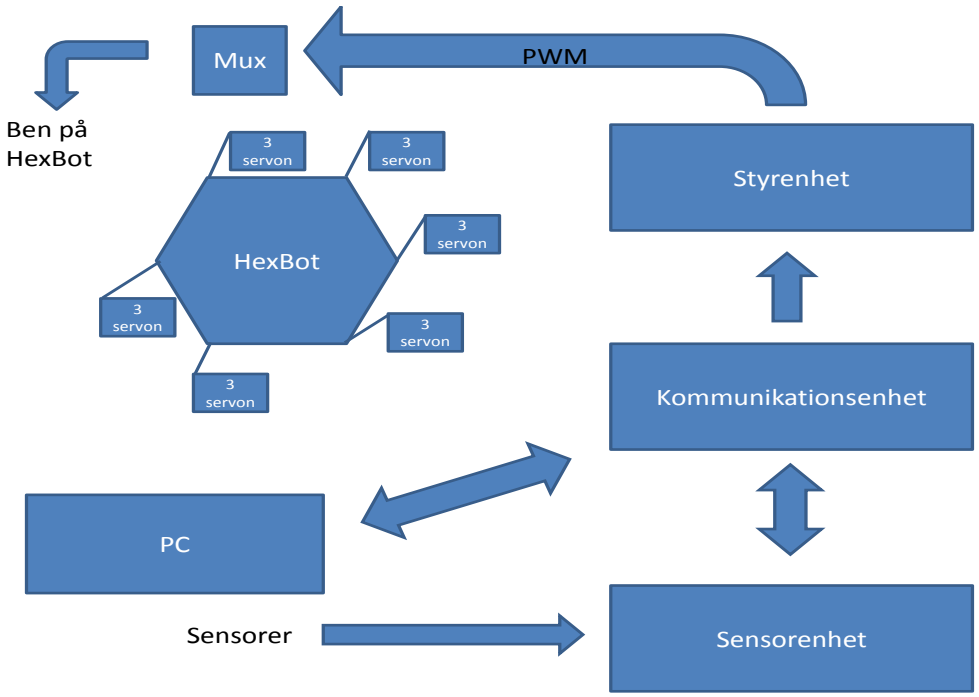
PC-enheten tar emot, tolkar och visar de mätdata som skickas från roboten för att kunna rita upp den karta som den får skickad till sig. Den skickar även, vid manuell styrning, ut de styrsignaler som leder roboten. En översiktlig skiss av systemet finns i Figur 2.



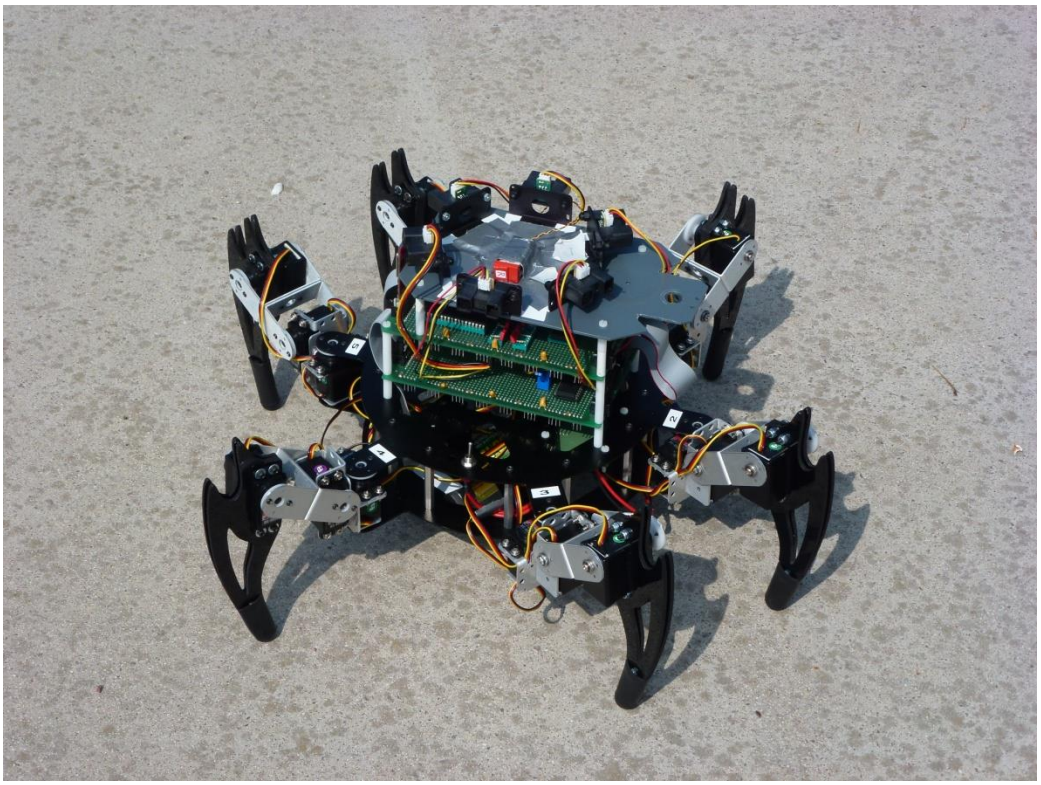

**Figur 2 Systemskiss**

### 2.2 Moduluppbyggnad

Roboten är uppbyggd av tre stycken moduler; kommunikationsenhet, styrenhet och sensorenhet. De tre modulerna går att byta ut mot en annan modul, av samma slag enligt krav, givet att den inbytta modulen fungerar på samma sätt som den ursprungliga. Ett översiktligt flödesschema finns i Figur 3.




**Figur 3 Flödesschema hela systemet**



**Figur 4 Bild på roboten**

# 3 Produkten

## 3.1 Beskrivning av produkten

Produkten är en sexbent kartritande robot. Roboten har tre processorer, sex IR-sensorer samt en CPDL. En bild på roboten finns i Figur 4.

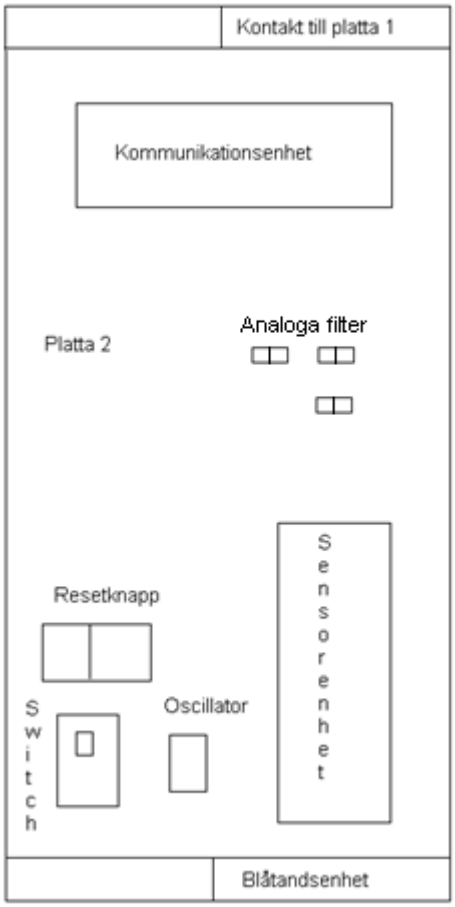
## 3.2 Användningsområden

Roboten har två lägen, ett manuellt och ett autonomt. I manuellt läge styrs roboten via en dator, i ett medföljande program (HexBot) vars användarhandvisning kan läsas i medföljande produktblad ”*Användarhandvisning för HexBot”*. Roboten kan i manuellt läge gå framåt, bakåt, rotera åt höger eller åt vänster. I manuellt läge ritas ingen karta.

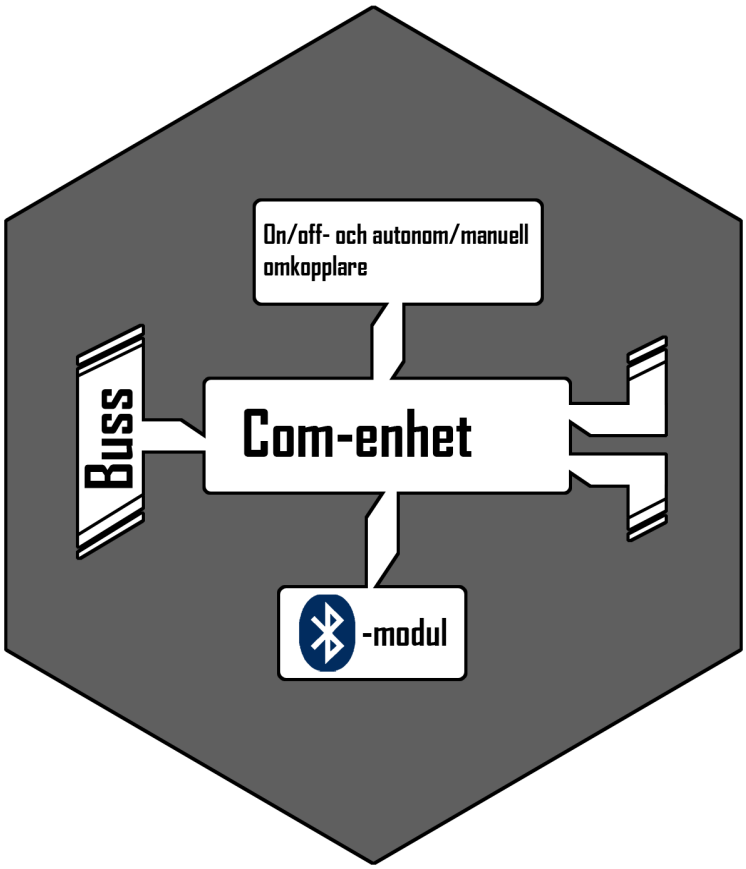
I autonomt läge är robotens uppgift att röra sig i en värld, uppbyggd av kartongväggar enligt *Appendix A.* Samtidigt som roboten rör sig i en okänd miljö, så kartlägger den även miljön. Efter att den ritat färdigt kartan kommer roboten att stanna.

# 4 Delsystem 1: Kommunikationsenhet

Kommunikationsenheten fungerar som talman och samordnare på roboten. Den agerar master på robotens buss och när man vill nå roboten via Blåtand från en dator går det genom denna enhet. Vid autonom styrning beslutar sensorenheten om hur man ska gå, dessa beslut kommer även dem skickas till styrenheten via kommunikationsenheten. När det kommer till bussen kommer enheten växla mellan att läsa data från sensorenheten, och skicka styrinstruktioner till styrenheten. Kommunikationsenheten sitter tillsammans med sensorenheten på platta 2, vilket kan ses i Figur 5. En översiktlig bilda kommunikationsenheten finns i Figur 6.




**Figur 5 Komponenter på platta 2**




**Figur 6 Blockschema över kommunikationsenheten**

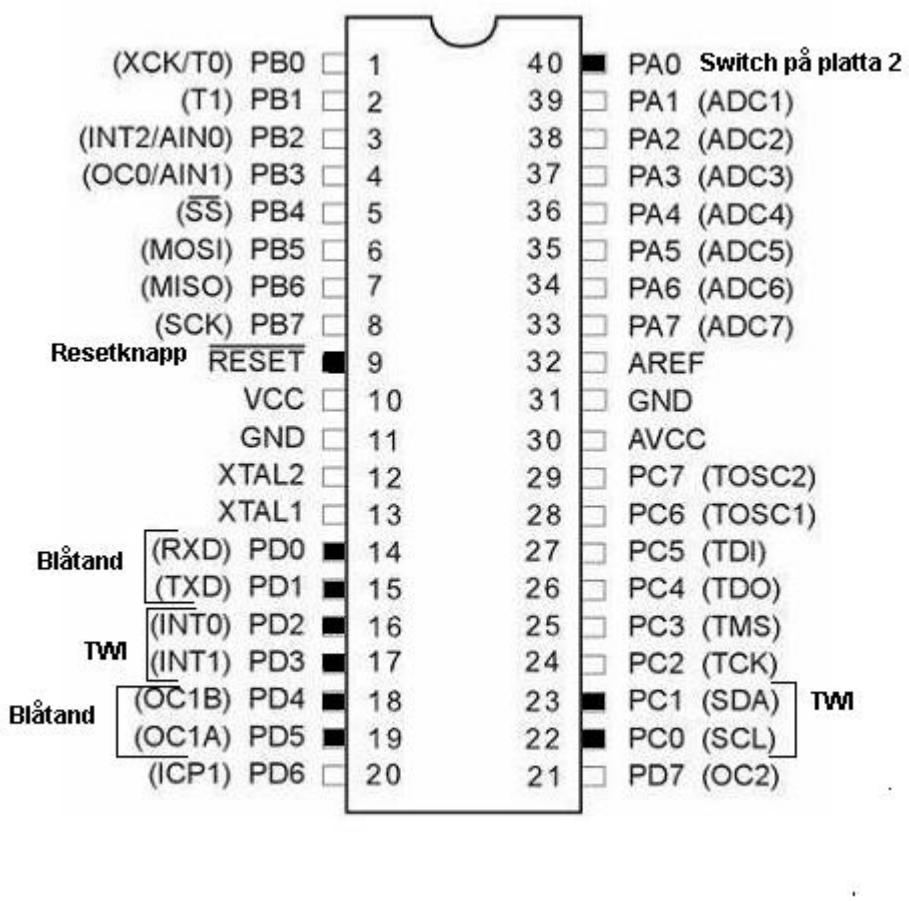
### 4.1 Komponenter

Kommunikationsenheten finns på platta 2 och består av följande komponenter:

- Processor, ATmega16
- Blåtandmodul
- Extern oscillator, 14,745MHz
- Brytare för val mellan autonom och manuell styrning

### 4.2 Processorn

Processorn sköter kommunikationen mellan de olika enheterna. Detta görs via TWI om det är kommunikation med sensorenhet eller styrenhet. Kommunikation med PC sker via USART och överförs med hjälp av blåtandsenheten. Kommunikationsenheten styr programloopen, som bland annat instruerar sensorenheten att läsa av sensorerna, och i steget efter att skicka datan till kommunikationsenheten. Programloopen ser olika ut beroende på om roboten ska arbeta i autonomt eller manuellt läge. De använda benen på kommunikationsenhetens processor kan ses i Figur 7.

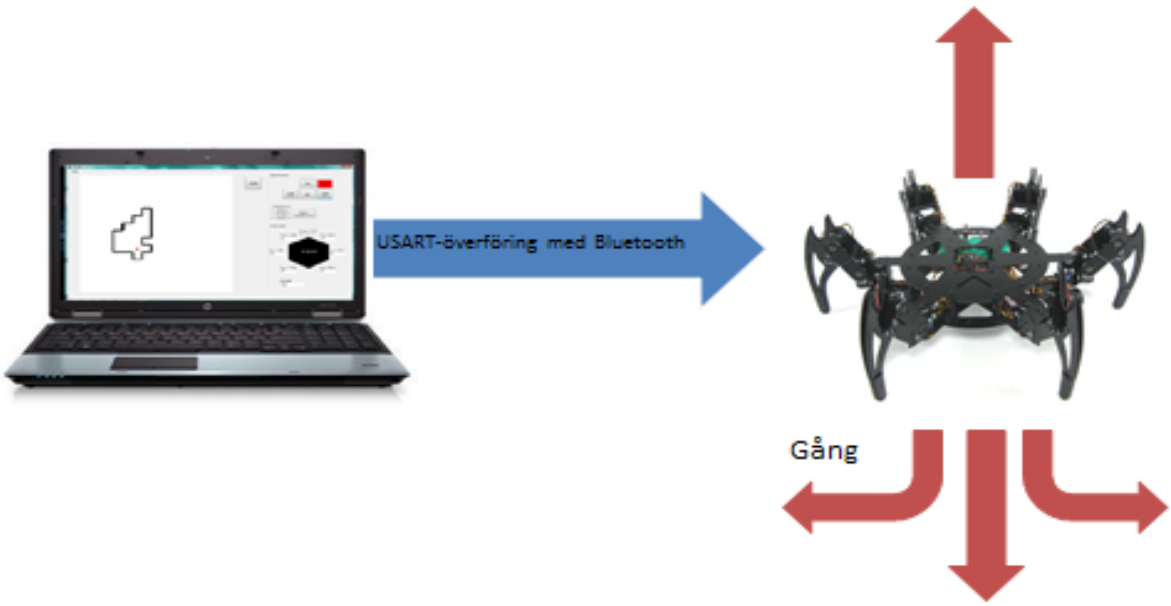



**Figur 7 Använda ben på kommunikationsenhetens processor**

### 4.3 Blåtandsenheten

Kommunikationsenhetens I/O-port D används för de fyra signaler som hanteras av Blåtandsenheten. PD0 och PD1 används för dataöverföring, och PD4 (CTS, Clear To Send) och PD5(RTS, Request To Send) används för handskakning.

Överföring mellan robot och dator börjar med en byte som kallas startbyte. Därefter kommer en byte där de fyra mest signifikanta bitarna talar om vilken datatyp det är och de fyra minst signifikanta bitarna talar om hur många efterföljande databytes som kommer.
Vidare kommer 1-9 bytes (beroende på vad tidigare byte angett) data.
Sist i sekvensen kommer en stoppbyte som meddelar att överföringen är klar.
Mottagaren tolkar och utför instruktionen alternativt sparar datan, givet att sekvensen följde ovan beskrivet mönster. Kommunikationen mellan roboten och datorn åskådliggörs i Figur 8. Specifikation av kommunikationen via Blåtand finns i *Appendix B*.




.

**Figur 8 Översiktlig bild över hur roboten kommunicerar med datorn.**

### 4.4 TWI

Busskommunikationen sköts via processorns port C, pinne 0 och pinne 1. Dessa är dedikerade till ett Two-wire Serial Interface. Kommunikationsenheten är master på bussen,
detta betyder att om någon av de andra enheterna (slavar) ska skicka data på bussen så måste Mastern ha instruerat om detta innan. Styrinformationen paketeras i paket om 8 bitar. De data som tar mest plats på bussen är kartdata, av den anledningen skickas kartdata från föregående position samtidigt som roboten förflyttar sig. Specifikation av kommunikationen via bussen finns i *Appendix B*.

### 4.5 Autonomt läge

I autonomt läge agerar kommunikationsenheten som datafördelare. Vid autonomt läge skickas kart-, position - och sensordata från sensorenhet till PC, via kommunikationsenheten. Kommunikationsenheten erhåller data från sensorenheten med TWI, och skickar ut data till

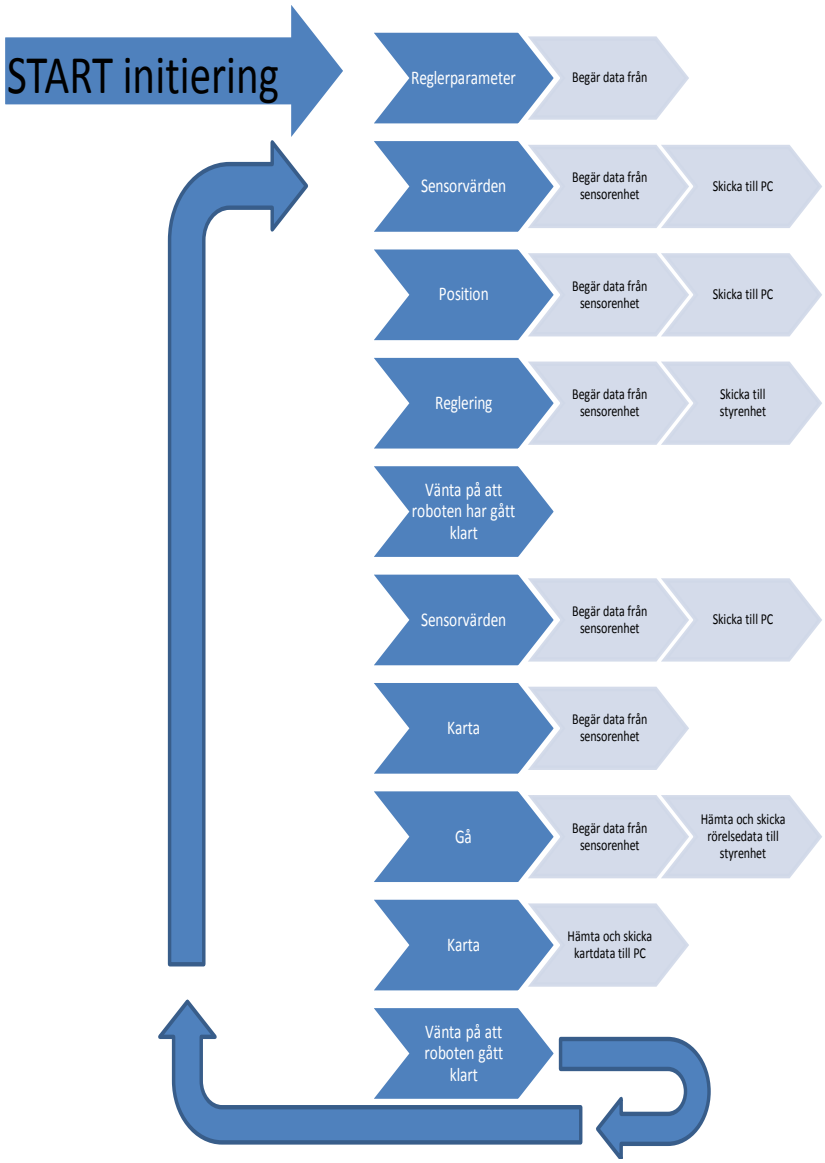
PC med USART via Blåtand. Styrdata hämtas också från sensorenheten men denna skickas, via kommunikationsenheten, till styrenheten med TWI. Från PC:n skickas reglerdata med Blåtand och USART till kommunikationsenheten, varvid dessa data skickas till sensorenheten med TWI. Ett flödesschema för programmet som körs i autonomt läge finns i Figur 9.

### 4.6 Manuellt läge

I manuellt läge skickas styrdata till kommunikationsenheten med USART och Blåtand varvid detta skickas vidare till styrenheten. Sensordata hanteras likadant som i autonomt läge.

### 4.7 Brytare

En huvudströmbrytare sitter monterad på robotchassit, utöver denna krävs en ”masterreset” som sätter alla processorer i giltiga startpositioner. En brytare för val mellan autonom och manuell styrning finns.

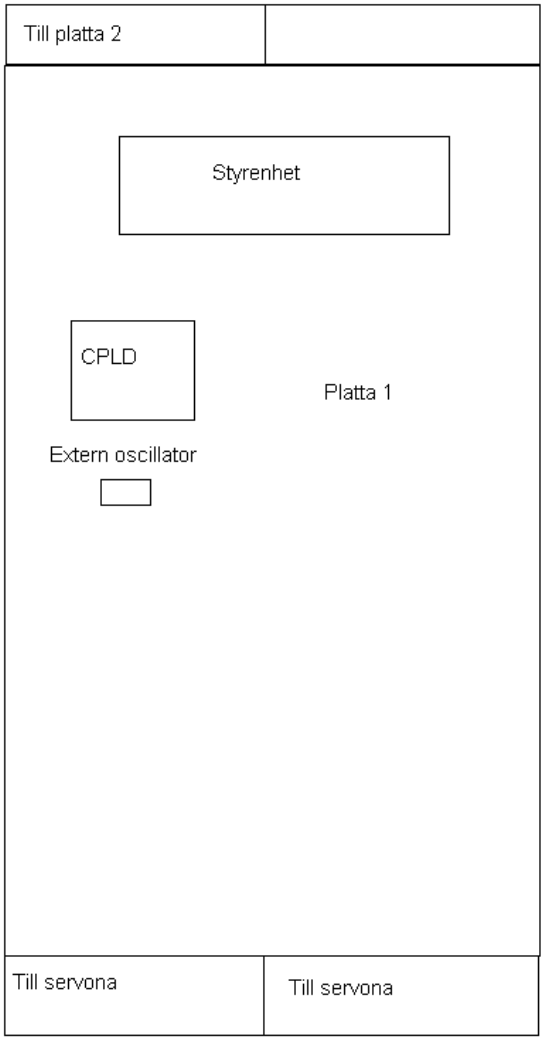



**Figur 9 Flödesschema, programmet i kommunikationsenheten**

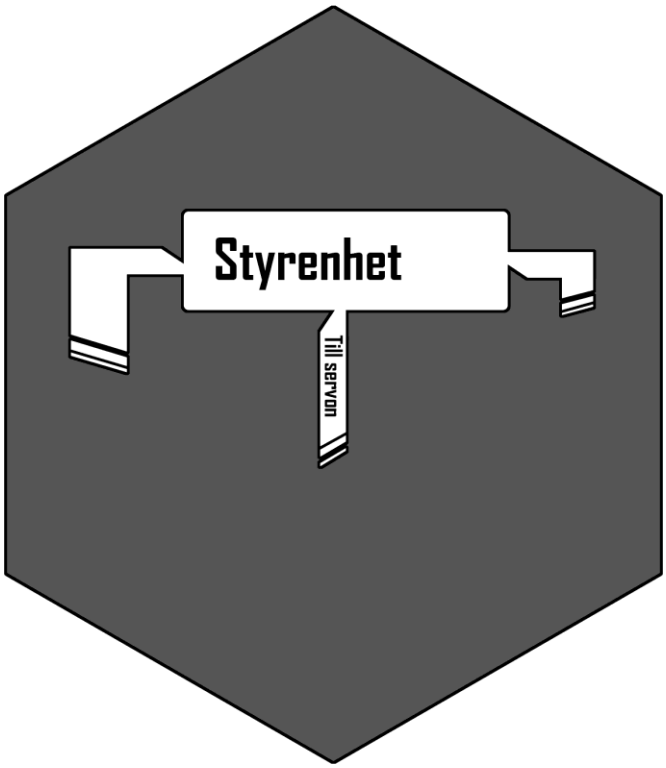
# 5 Delsystem 2: Styrenhet

Styrenheten samordnar på order av kommunikationsenheten de sex benen så dessa lyckas förflytta roboten i önskad riktning. Roboten går på det sättet att den har två ben framåt, två ben bakåt och ett balansben på varje sida. När den går flyttas endast två ben åt gången för att servona ska orka hålla roboten uppe. Detta är uppenbart långsammare än att lyfta tre ben åt gången, men stabilitet prioriteras högre än hastighet.

När kommunikationsenheten har beräknat vilken riktning roboten skall gå skickas detta via bussen. Styrenheten tar emot och tolkar de instruktioner den skall utföra och skickar sedan två styrsignaler åt gången genom en demultiplexer som styrs av styrenheten. Signalerna slussa nu till respektive servo som intar önskade position. Styrenheten ligger ensam på platta 1, vilket kan ses i Figur 10. Ett blockschema över styrenheten finns i Figur 11.




**Figur 10 Komponenter på platta 1**




**Figur 41 Skiss över styrenheten**

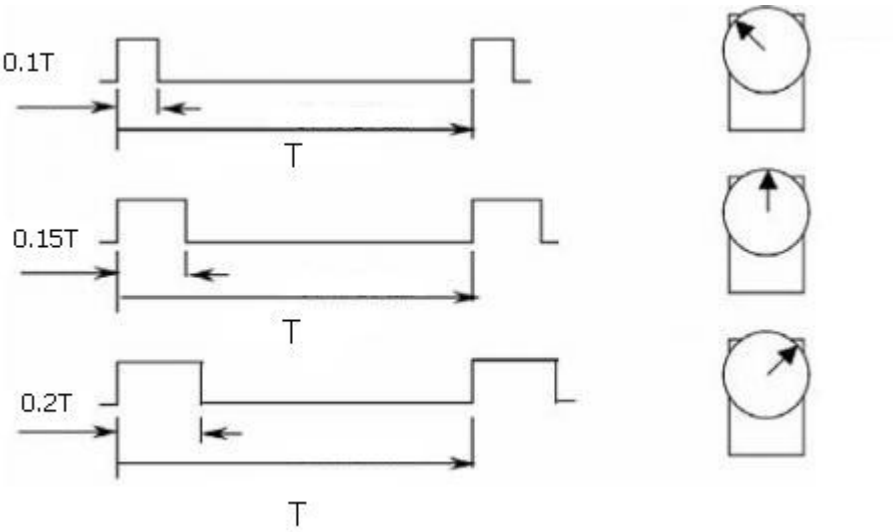
### 5.1 Komponenter

Styrenheten består av följande komponenter:

- Processor, ATmega16
- Extern oscillator, 14,745MHz
- CPLD för distribution av PWM:signaler till servon

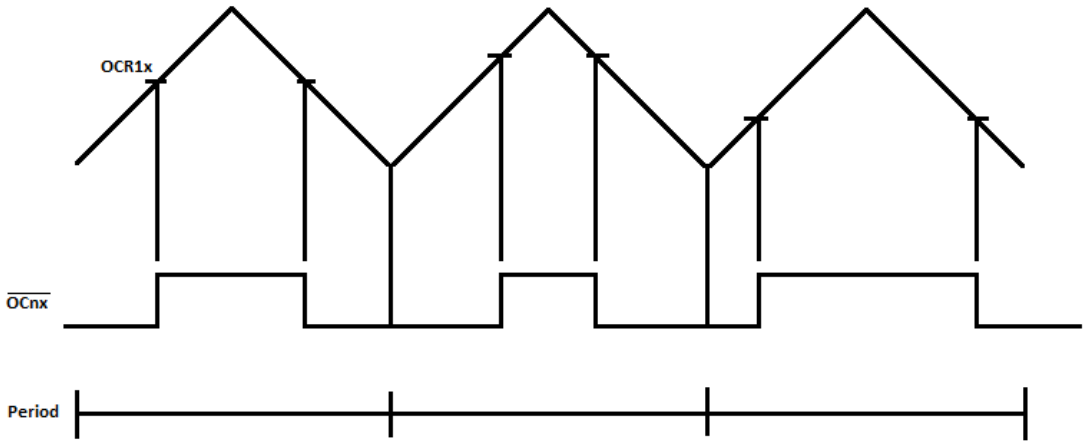
### 5.2 Processorn

För varje uppdrag, så som gå eller rotera, finns i styrenheten en korresponderande tabell som innehåller den sekvens av servolägen som måste genomlöpas för uppdraget skall utföras. Servona styrs med hjälp av Pulse Width Modulation (PWM). Detta fungerar som så att en alltid 20 ms lång styrsignal skickas till varje servo 50 gånger per sekund och det som avgör vilket läge servot skall inta är förhållandet mellan hög- och lågflank, duty cycle. Till exempel ger 7,5 % högflank ett servo i dess mittläge. För att täcka alla lägen ett servo skall inta krävs endast att maximalt 11 % styrsignalen skall vara hög, resten får vara låg. Det är detta som utnyttjas då processorn egentligen endast har två PWM utgångar men 18 servon skall styras. Om man klipper av signalen strax efter att 11 % har skickats så kan processorn börja styra nästa servo och hinner på så vis precis skicka en tillräcklig högflank till varje servo innan det är dags att styra det första servot igen för att behålla den krävda 20 ms periodtiden. Exempel på styrpulser till ett servo finns i Figur 12.




**Figur 12 Servostyrpulsexempel**

Processorns inbyggda fas- och frekvenskorrekta PWM-mod användes för att generera en centrerad duty-cycle. Centreringen ger en säkerhetsmarginal när PWM-signalen passerar CPLDn som är nödvändig för att undvika att spikar uppstår. I Figur 13 syns ett timing-diagram hur mjukvaruprogrammet ser till att varje signal får önskad duty-cycle. En inbyggd timer räknar upp till ett värde som bestäms i programkoden och börjar sedan räknas ner. Samtidigt bestämmer värdet i register OCR1x när utsignalen OCnx ska växla mellan noll och ett, vilket resulterar i en PWM-puls med centrerad duty-cycle.

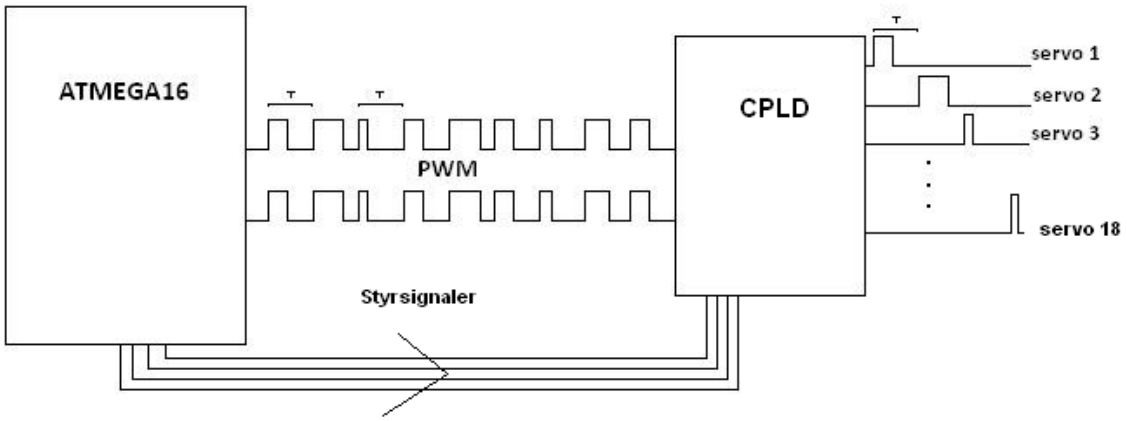



**Figur 13 Framkallning av PWM-pulser**

### 5.3 CPLD

CPLD:n (Complex Programmable Logic Device) är programmerad som en demultiplexer med 6 ingångar och 18 utgångar. En översiktlig skiss över hur detta fungerar finns i Figur 14.

Två av insignalerna är processorns PWM-signaler som fördelas ut till de 18 servona med hjälp av de andra 4 insignalerna (styrsignaler). PWM-signalen som kommer in till muxen har en frekvens som är 9 gånger så hög som den signal som går ut till servona har. Det betyder att den inkommande signalen innehåller styrsignalerna till 9 servon.




**Figur 54 Servostyrning via CPLD**

### 5.4 Mjukvara

Programmet för styrenheten innehåller en styrmatris som lagrar alla de önskade positioner som programmeraren vill att de sex benen ska hålla vid ett givet tillfälle. Varje ben har 3 servon och det finns 6 ben, men antalet positioner är godtyckligt. Detta blir naturligtvis en väldigt stor 3-dimensionell matris, men eftersom den sparas i programminnet så blir storleken inget problem. Många servolägen återkommer i styrmatrisen varför alla lägen definieras innan det att matrisen sätts samman. Själva bygget av matrisen påminner sedan lite om att bygga LEGO™. Man har en påse med bitar som ska plockas ihop så att resultatet liknar en vandrande spindel. Designen gör att det ”lätt” går att programmera olika gångstilar, rotationer och benlägen. Ett utdrag ur en styrmatris finns i Figur 15.

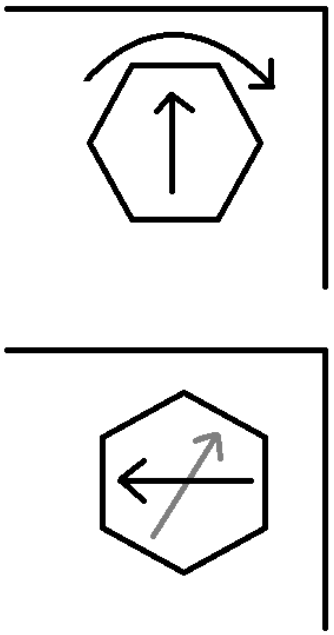
Efter det att styrenheten har kört några initierande funktioner styrs den av två olika avbrottsrutiner. Det ena avbrottet är ett TWI-avbrott som körs varje gång kommunikationsenheten anropar den via bussen. Avbrottet innehåller kommandotolkning som bestämmer vilket ”uppdrag” roboten skall utföra och hur den ska reglera. Det andra avbrottet är PWM-räknarens avbrott som genereras varje gång den räknat klart, ungefär varannan millisekund. Denna avbrottsrutin plockar ut nästa element i matrisen som avgör hur nästa styrsignal skall se ut. Den räknar även upp ett antal variabler för att se till att matrisen genomlöps på rätt sätt.

```
const int styrmatris[45][6][3] PROGMEM = {
        //Position0
        {
                {l1, m2, brl3},
                {m1, l2, brr3},
                {sh1, sl2, h3},
                {l1, m2, frl3},
                {m1, l2, frr3},
                {sh1, sl2, h3}
        },
...
        //position3 - Grundläge vid gång
        {
                {l1, m2, brl3},
                {m1, l2, brr3},
                {sm1, sl2, h3},
                {l1, m2, frl3},
                {m1, l2, frr3},
                {sm1, sl2, h3}
        },
        //position4 - Lyft udda
        {
                {h1, m2, brl3},
                {m1, l2, brr3},
                {h1, sl2, h3},
                {l1, m2, frl3},
                {h1, l2, frr3},
                {sm1, sl2, h3}
        },
...
```

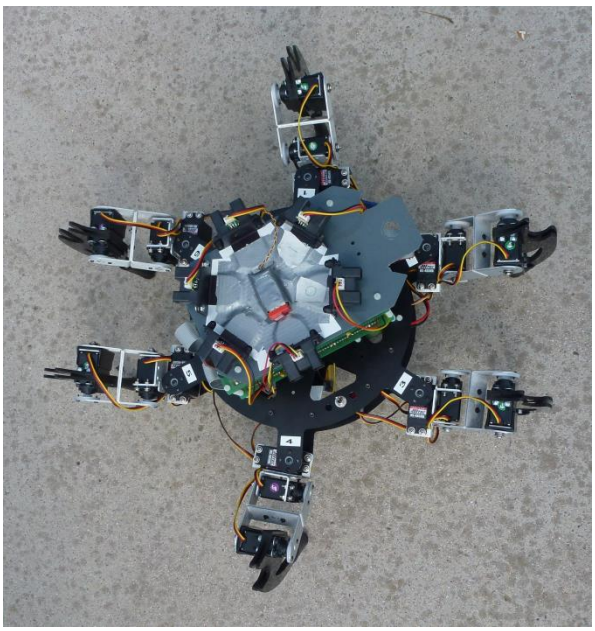
**Figur 6 Utdrag ur styrmatris**

### 5.5 Gångstil

Roboten har en gångstil där den har två ben framåt som drar roboten och två ben bakåt som skjuter ifrån samt två sidoben som för det mesta hjälper till att hålla balansen. Detta syns i Figur 17. Här flyttas endast två ben i taget för att servona ska orka. När samtliga ben flyttats framåt drar alla ben slutligen kroppen framåt varpå allt börjar om. Tack vare robotens sexfaldiga symmetri kan denna gångstil använda i sex olika riktningar. Det betyder att om roboten behöver göra en 90 graders sväng så behöver den endast rotera 30 grader för att få två nya ben pekandes framåt i den nya önskade riktningen. Detta visas i figur 16. När roboten ska rotera lyfter den tre ben och roterar dessa i luften och gör sedan lika dant med de tre övriga benen. Först när detta är klart roteras kroppen med alla sex ben samtidigt för att alltid ha tillräckligt med stöd från benen.

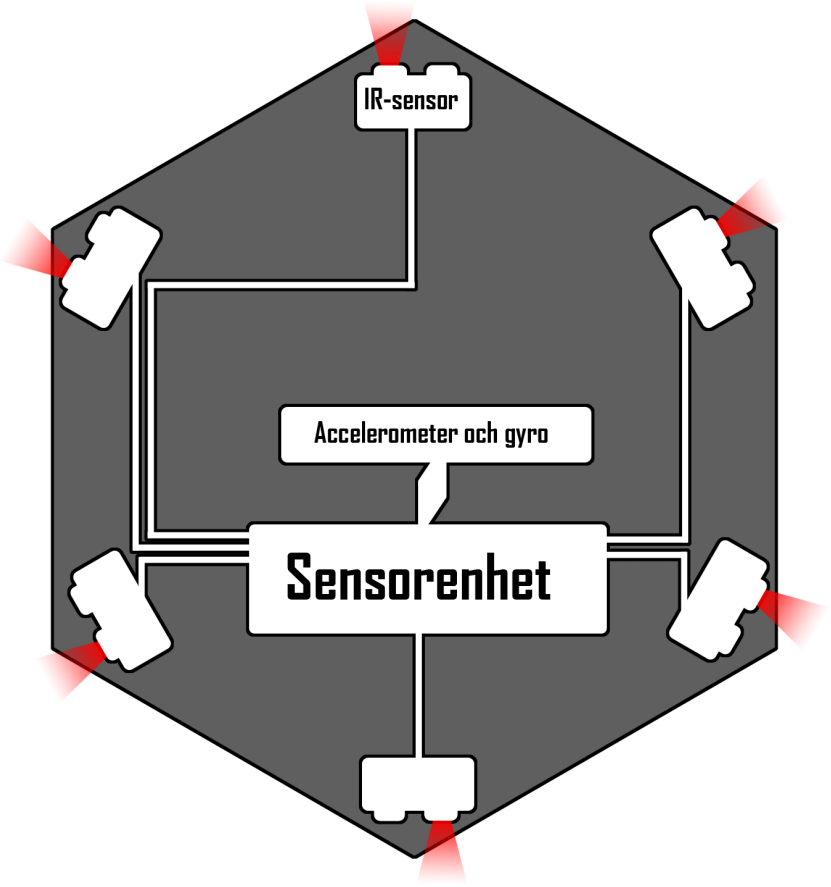


**Figur 7 Bild över robotens beteende vid sväng**



**Figur 87 Illustrativ bild över robotens gångstil.**

# 6 Delsystem 3: Sensorenhet




**Figur 98 Skiss över sensorenheten**

## *6.1 Hårdvara*

Sensorenheten består av en atmega16 och sex stycken IR-sensorer för att mäta avstånd.

IR-sensorerna sitter monterade en åt varje håll på det översta lagret av kretskort på roboten. IR-sensorerna är kopplade till atmegans ADC-ingångar noll till fem. En översikt av sensorenheten finns i Figur 18.

IR-sensorerna börjar mäta avstånd ca 30ms efter strömpåslag och mäter sedan kontinuerligt. Sensorenhetens MCU kommer att läsa av IR-sensorerna en efter en hela tiden, då AD-omvandlingen tar förhållandevis lång tid. Avläsningen går till så att en AD-omvandling initieras, varvid MCUns övriga uppgifter börjar utföras. När konverteringen är klar körs en avbrottsrutin och sedan startas nästa AD-omvandling, för nästa sensor. Detta fortsätter tills alla sensorer har gåtts igenom, och då börjar allt om.

### 6.2 Mjukvara

IR-sensorerna kommer att vara begränsade att mäta mellan 25 till 150 cm, då dessa inte är tillräckligt exakta utanför detta område. För att inte mäta närmare än 25cm så måste man se till att roboten aldrig går närmare än just detta avstånd.

För att ta sig runt labyrinten används sensorenheten för att avgöra var väggar finns. Dessa data skickas sedan till kommunikationsenheten som avgör vad roboten skall göra härnäst. Detta indelas i två faser, en orienteringsfas och en utforskningsfas.

Orienteringsfasen handlar om att ställa in roboten så att den står parallellt intill en vägg, bestämma vilken sida på roboten som blir framåt, och sedan rita in det man vet om väggarna i den interna kartan. När detta är gjort så börjar utforskningsfasen.

I utforskningsfasen delas IR-sensorerna i grupper med lite olika funktionalitet. De sensorer som är riktade framåt och bakåt används nu för att få en ungefärlig uppskattning om hur långt man har gått.

De två sensorer som sitter placerade åt den sida som en vägg finns används för att reglera så att roboten går parallellt med en vägg. De kommer även användas för att upptäcka när roboten har gått förbi ett hörn.

Sensorenheten kommer även att använda sina mätdata för att rita upp den karta som den skickar till PC-enheten via kommunikationsenheten. Sensorenheten kommer även att skicka styrbeslut till styrenheten via kommunikationsenheten vid autonom styrning.

### 6.3 Komponenter

Sensorenheten består av följande komponenter:

- Processor, ATmega16
- Optisk avståndsmätare
- Motstånd

### 6.4 Karta

Bankraven gör att man oavsett startruta aldrig kan ha mer än 6 rutor, åt något håll, till yttervägg. Om roboten då i den interna representationen av världen startar i mittenrutan, av ett 13x13 rutsystem, så kommer kartan alltid att få plats. Detta utvidgas dock till ett 15x15 eftersom det då blir lättare att hantera ytterväggar.

### 6.5 Styrbeslut

Utifrån den interna kartan och information från sensorerna skall roboten styras så att hela omgivningen kartläggs. När detta är klart skall roboten stanna.

## 6.6 Styrscenario

### 6.6.1 Start

För att kartritningen ska fungera korrekt ska kartan vara uppbyggd enligt *Appendix A*. En ruta på planen som har minst en angränsande vägg väljs, varpå användaren själv får placera roboten i denna ruta. Roboten kommer med hjälp av sensorerna ställa in sig så rakt den kan innan den börjar gå.

### 6.6.2 Väggföljning

När roboten anser att den är tillräckligt nära väggen så roterar den 30° medurs så att en sida blir parallell med väggen. Nu börjar loopen på sätt och vis om och roboten går framåt för att hitta en ny vägg. Denna algoritm fungerar så länge banan är rektangulär, men om det skulle förekomma en återvändsgränd eller att den skulle behöva svänga åt höger behöver den även använda andra sensorer för att kunna avgöra sin väg.

### 6.6.3 Högersväng/högeråtervändsgränd

Därför måste de två sensorer som pekar in mot den vägg som roboten följer hela tiden kollas så att de inte går från att visa säg 40 cm till helt plötsligt 140 cm. I det läget ska den inte tro att den hamnat snett utan förstå att det är frågan om ett hål åt höger. Om den i detta läge kan detektera en vägg framåt som är närmre än cirka 120 cm så vet den att hålet i fråga måste vara en återvändsgränd och behöver då aldrig bege sig i där utan fortsätter fram till väggen och fortsätter enligt huvudalgoritm.

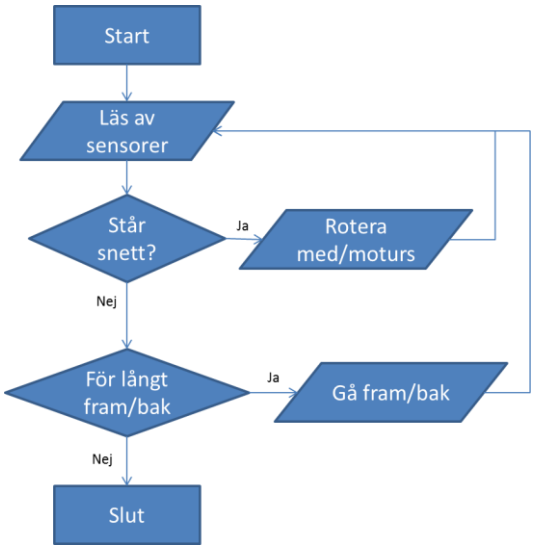
Om den inte kan känna en vägg framför sig på tillräckligt kort avstånd ska den fortsätta tills dess att den tror att den befinner sig mitt i rutan framför vänsterhålet och rotera 30° moturs. Den fortsätter sedan in där och fortsätter enligt huvudalgoritm.

## 6.7 Reglering

Regleringen används för att motverka ett antal typer av fel i gången, dels att roboten roterar och dels att roboten glider i sidled och går olika långt.

Den första delen av regleringen, som är av diskret typ, sker då roboten tror att den är mitt i en ruta och skall läsa av omgivningen med sensorerna. Den har till uppgift att justera in så att den står mitt i rutan och parallellt med omgivningen, så att avläsningen blir rätt.

Om den då har en vägg till höger som den inte står parallellt med justeras roboten så att den är så pass parallell så att avläsningen blir rätt. Om den här har väggar framför eller bakom sig så kommer den även justera så att den står mitt i rutan. Denna process illustreras i figuren bredvid.




**Figur 10 Flödesschema över den diskreta regleringen**

Den andra delen av regleringen är en kontinuerlig PD-reglering som styr roboten under gång, för att försöka hålla roboten på ett lagom avstånd från väggen. Regleringen är implementerad så att roboten hela tiden skall stå parallellt med en sidovägg om sådan finns. För att göra detta används både fram- och bakben för att justera avståndet från väggen och bakbenen för att justera vinkeln mot väggen.

Båda dessa regleringar använder väggen till höger som referensvägg, och kan då alltså inte reglera om en vägg till höger ej existerar.

Ett flödesschema över hur regleringen fungerar finns i Figur 19.

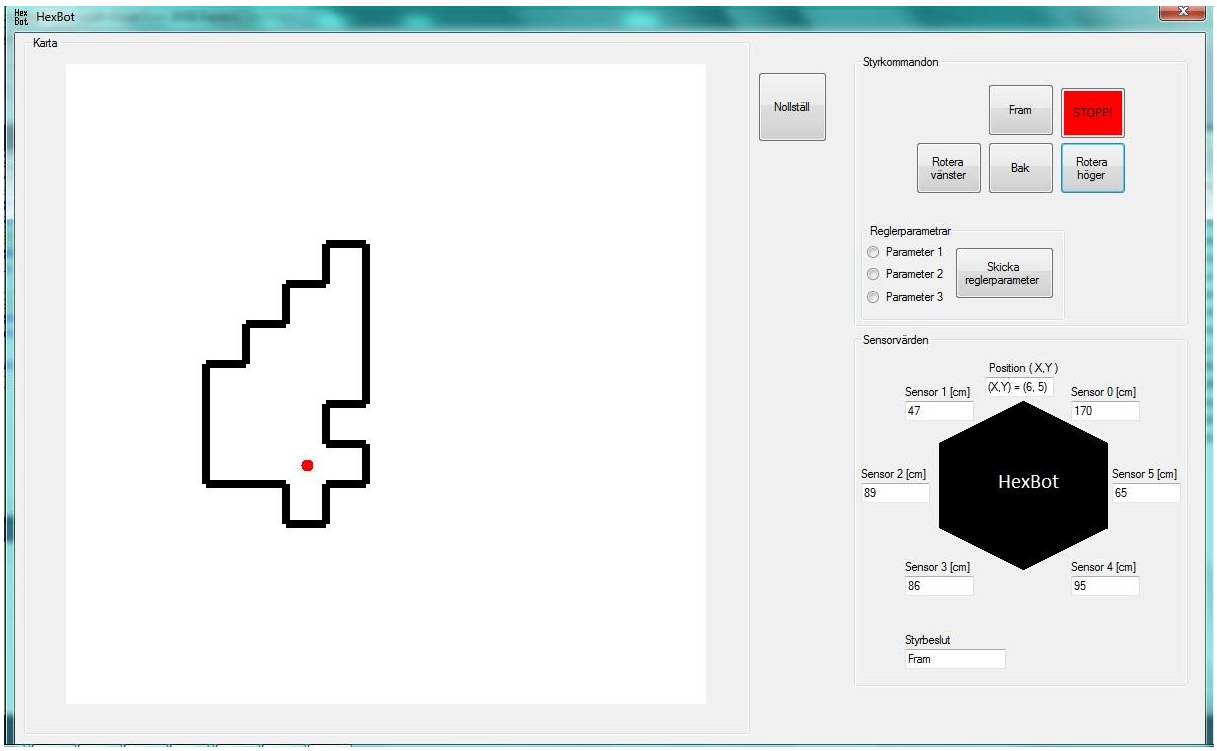
### *6.8 Kartoptimering*

Då roboten läser av sina IR-sensorer och upptäcker väggar använder den en algoritm för att optimera fram väggar som den inte kan se. Algoritmen följer banspecifikationens regler för att sätta ut de väggar som sensorerna inte kan se, men som måste finnas där.

# 7 Delsystem 4: PC-enhet

PC-enheten har två olika lägen, autonomt och manuellt. Vid autonomt läge är roboten en kartritande robot. PC-enheten tar då emot och tolkar de mätdata som skickas från roboten för att kunna rita upp kartan. På skärmen ska framförallt kartan skådas, men även sensor- och positionsvärden kan ses.

Vid manuell styrning skickar PC-enheten ut de styrsignaler som ska leda roboten. All data, både för manuellt och autonomt läge, mellan PC och roboten skickas seriellt med Blåtand och det är kommunikationsenheten som tar emot datan först. Det finns knappar för att gå roboten att utföra önskad gång i riktning eller rotation.




**Figur 20 Bild över programmet med uppritad karta**

## *7.1 Komponenter*

Kommunikationsenheten består av följande komponenter:

- PC
- Blåtandmodul
- Windows Visual Studios

### 7.2 Programmet

In- och utdata kommer i port 4. Programmet består av två .h-filer och en .c-fil( main-fil). Språket som används är c++ och programmet är Windows visual studios. Det som programmet ska göra är att ta emot data från roboten och bearbeta det. Det ska även skicka styr- och reglerkommandon vid manuellt läge. En startbyte, en datatypbyte, 1-6 databytes och en stoppbit. Detta är endast mellan robot och PC. I Windows Visual studios används Form för att rendera gränssnittet. Projektet är uppbyggt av en main-fil som kör igång Form. I övrigt används två h-filer för att skriva resten av koden. En av h-filerna innehåller definierade namespaces.

PC-programmet ska skicka styrbeslut, via kommunikationsenheten, till styrenheten samt reglerparameter, via kommunikationsenheten, till sensorenheten. Det tar emot sensorvärden och position från sensorenheten. Vid autonomt läge visas även styrbesluten från sensorenheten.

För användning av PC-programmet hänvisas användaren att läsa ”*Användarhandledning HexBot”* som följer med denna dokumentation. Det finns även en skärmdump från programmet i Figur 20.

# Appendix A – Banspecifikation

Version 1.2

Banspecifikationen utgör de krav som finns på banan.

| | | | |
|---|---|---|---|
| **Krav nr 1** | **Original** | Banan ska bestå av ett rutsystem bestående av högst 7 x 7 rutor | 1 |
| **Krav nr 2** | **Original** | Alla rutor ska vara 80cm x 80cm och samtliga hörn ska vara $90^{o}$ | 1 |
| **Krav nr 3** | **Original** | Samtliga väggar ska vara en multipel av 80cm | 1 |
| **Krav nr 4** | **Original** | Väggarna ska placeras efter rutsystemet | 1 |
| **Krav nr 5** | **Original** | Alla ytterväggar ska bilda en sluten, enkel slinga | 1 |
| **Krav nr 6** | **Utgått** | Ytterligare en sluten slinga (då en kvadrat på 80cm x 80cm) får förekomma, en så kallad köksö. Dock endast inom 80cm från minst en yttervägg | 2 |
| **Krav nr 7** | **Uppdaterat** | Återvändsgränder ska vara exakt 80 cm djupa. Med en återvändsgränd menas att man måste gå igenom en och samma ruta två gånger för att ta sig runt hela banan efter väggarna. | 1 |
| **Krav nr 8** | **Original** | Beslut om något krav på prioritetsnivå 2 eller 3 ska genomföras, ska tas senast tre veckor innan leverans. | 1 |

# Appendix B- Definition av gränssnitt mellan modulerna

### *Från sensorenhet:*

**Data:**

Karta
Framåtriktning
Position
Sensordata
Reglering
Styrbeslut

**Avbrott:**

Data finns att hämta
Stanna
Reglera

### *Från PC:*

**Data:**

Gå framåt
Gå bakåt
Rotera vänster
Rotera höger
Stanna
Regleralgoritmens initiala styrparametrar

**Avbrott:**

Inga

### *Från Kommunikationsenhet:*

**Data:**

Sensordata
Styrbeslut
Gå framåt
Gå bakåt
Rotera vänster
Rotera höger
Stanna
Regleralgorimens initala styrparametrar

Reglering
Kartdata
Positionsdata

**Avbrott:**
Stanna

***Från Styrenhet:***

**Data:**
Inga

**Avbrott:**
Inga

***Dataöverföring:***
2-17 Byte

Byte 1:
Datatyp (3 bitar) Antal databyte (3 bitar)

Byte 2-16:
Data

## Via Blåtand :

| ***Datatyp: PC till robot(kom-enhet)*** | ***Binär representation*** |
|---|---|
| Gå framåt | 0010 xxxx |
| Gå bakåt | 0011 xxxx |
| Rotera vänster | 0100 xxxx |
| Rotera höger | 0101 xxxx |
| Stanna | 0110 0000 |
| Regleralgoritmens initiala styrparametrar | 0111 xxxx |
| Start av data | 1000 0000 |
| Slut på data | 1001 0000 |

| *Datatyp: Robot(kom-enhet) till PC* | *Binär representation* |
|---|---|
| Karta | 0001 xxxx |
| Styrbeslut | 0010 xxxx |
| Position | 0011 xxxx |
| Sensordata | 0111 xxxx |
| Start av data | 1000 0000 |
| Slut på data | 1001 0000 |

***Via TWI:***

**Allmänt TWI:**

Start: 0x00
Stop: 0x02
Data: 0x01

**Slavadress**
Sensorenhet : 0x02
Styrenhet : 0x01

| *Datatyp: Från och till kommunikationsenhet* | *Hex.-representation* |
|---|---|
| Ta emot sensordata | 0x01 |
| Läs sensordata | 0x10 |
| Ta emot kartdata | 0x02 |
| Läs av väggar | 0x20 |
| Ta emot regleringsdata( diskret) | 0x03 |
| Ta emot kontinuerlig regleringsdata | 0x30 |
| Ta emot rörelsedata | 0x04 |
| Ta emot positionsdata | 0x05 |
| Ta emot regleringsdata | 0x06 |
| Ta emot klartecken att roboten har gått klart | 0x70 |

| *Datatyp : Från och till sensorenheten* | *Hex.-representation* |
|---|---|
| Skicka sensordata | 0x01 |
| Läs sensordata | 0x10 |
| Skicka kartdata | 0x02 |
| Läs av väggar | 0x20 |
| Skicka regleringsdata (diskret) | 0x03 |
| Skicka kontinuerlig regleringsdata | 0x30 |
| Skicka rörelsedata | 0x04 |
| Skicka positionsdata | 0x05 |
| Spara regleringsparameter | 0x06 |

| ***Datatyp : Till styrenhet*** | ***Hex.-representation*** |
|---|---|
| Gå framåt | 0x20 |
| Gå bakåt | 0x30 |
| Rotera vänster | 0x40 |
| Rotera höger | 0x50 |
| Stanna | 0x60 |
| Gå lite framåt | 0x0A |
| Gå lite bakåt | 0x0B |
| Rotera lite vänster | 0x80 |
| Rotera lite höger | 0x0D |
| Regleringsdata | 0xC0 |

***Karta:***

Föregås av positionssändning.
Genomsöka väggar (4 bitar)
Vägg eller inte (4 bitar)
ORa vänster/höger/framtåt och bakåt för att få all information om rutan.
Om väggen inte är genomsökt ska den inte behandlas.

| Kartväggar | Binär representation |
|---|---|
| Vänster | 0000 0001 |
| Framåt | 0000 0010 |
| Höger | 0000 0100 |
| Bakåt | 0000 1000 |
| Vänster genomsökt | 0001 0000 |
| Framåt genomsökt | 0010 0000 |
| Höger genomsökt | 0100 0000 |
| Bakåt genomsökt | 1000 0000 |

**Framåtriktning:**

Riktningar är numrerade 1-6 där respektive riktning är normal till den sida som är till höger om varje ben med samma numrering.

| Framåtriktning | Binär representation |
|---|---|
| 1 | 0000 0001 |
| 2 | 0000 0010 |
| 3 | 0000 0011 |
| 4 | 0000 0100 |
| 5 | 0000 0101 |
| 6 | 0000 0110 |

***Position:***

X-position (4 bitar) Y-position (4 bitar)

| Position (X,Y) | Binär representation |
|---|---|
| -,1 | 0000 0001 |
| -,2 | 0000 0010 |
| -,3 | 0000 0011 |
| -,4 | 0000 0100 |
| -,5 | 0000 0101 |
| -,6 | 0000 0110 |
| -,7 | 0000 0111 |
| -,8 | 0000 1000 |
| -,9 | 0000 1001 |
| -,10 | 0000 1010 |
| -,11 | 0000 1011 |
| -,12 | 0000 1100 |
| -,13 | 0000 1101 |

***Gå i riktning:***

| Riktning | Binär representation |
|---|---|
| 1 | 0000 0001 |
| 2 | 0000 0010 |
| 3 | 0000 0011 |
| 4 | 0000 0100 |
| 5 | 0000 0101 |
| 6 | 0000 0110 |

***Rotera:***

Rotera går nu för tiden att göra åt alla håll varför detta skall fixa lite.

| Rotationsriktning | Binär representation |
|---|---|
| Medurs | 0000 0001 |
| Moturs | 0000 0010 |
| Mikro medurs | 0000 0011 |
| Mikro moturs | *0000 0100* |

***Stanna:***

| Stanna | Binär representation |
|---|---|
| Vänta på ny instruktion | 0000 0001 |

***Regleralgoritmens initiala styrparametrar:***

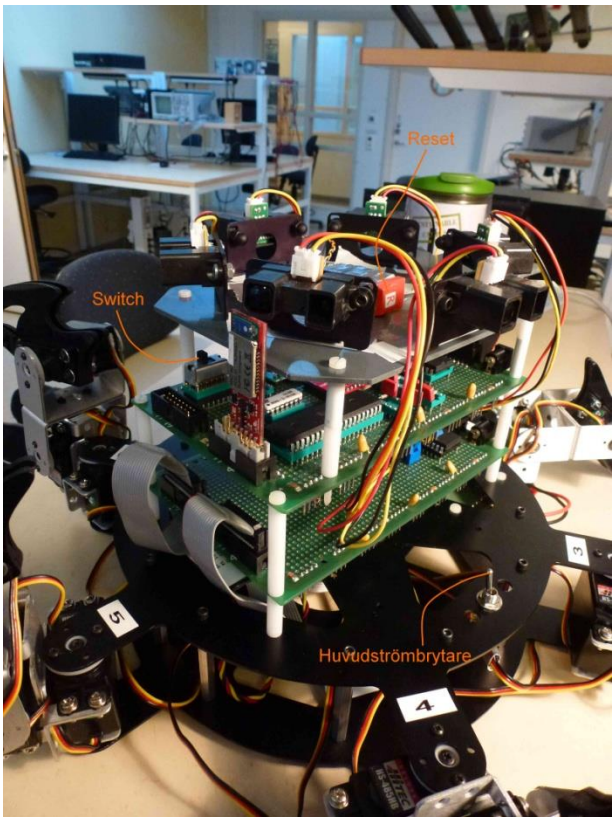
| Styrparameter | Binär representation |
|---|---|
| Minavstånd till vägg, i multipler av 5cm | 0000 0001 |
| Segerdans | 0000 0010 |

**Sensordata:**
Det är BARA IR-sensorernas värden som skickas
*Överföringsordning:*
Sensor i MS till Sensor i LS , i = 0,1,...,4,5
MS = Most significant
LS = Least significant.
Alltså 6 st överföringar totalt-

# Bilaga 1 - Användarhandleding

## Användarhadlednig HexBot

Användarhandledningen ska ge instruktioner för hur HexBot ska användas. Den beskriver både hur roboten styrs manuellt och hur man kan använda den i autonomt läge för att rita upp en karta över en miljö som uppfyller kraven i *Appendix A*.




Figur 1. Bild för att visa plats för knappar.

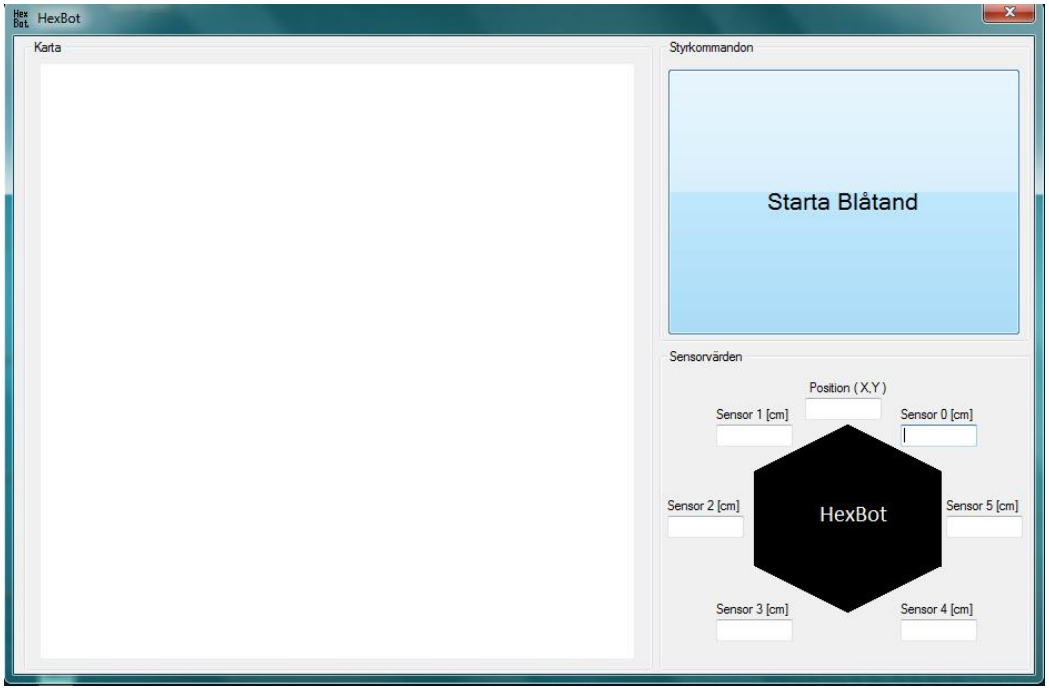
## Manuellt läge:

***Uppstartssekvens:***

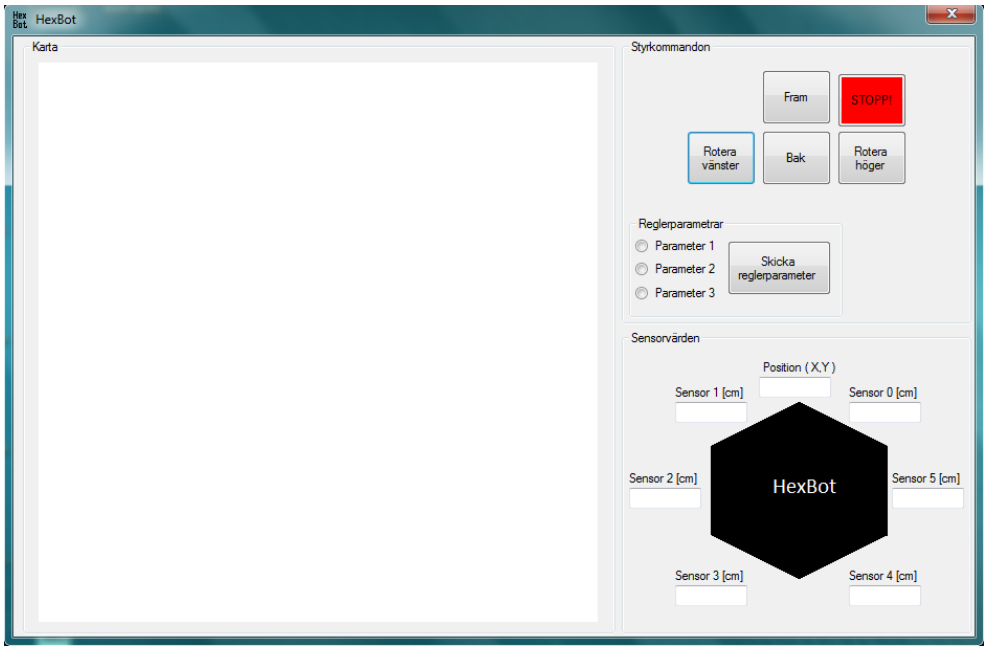
| Steg | PC | Robot |
|---|---|---|
| **1** | Starta datorn | Koppla in batteriet |
| **2** | Starta Blåtanden på datorn | Ställ brytaren i manuellt läge (Se figur 1) |
| **3** | Starta “HexBot.exe” | Koppla in Blåtandstickan |
| **4** | ---------- | Slå till huvudströmbrytaren (Se figur 1) |
| **5** | ---------- | Tryck på Resetknappen (Se figur 1) |
| **6** | Tryck på “Starta Blåtand” | ---------- |

### *Styrning:*

Efter att uppstarten är klar så styr man roboten genom att trycka på önskad funktionsknapp i “HexBot.exe”. För att förhindra oönskat beteende hos roboten, bör man låta roboten genomföra sin rörelse innan nästa instruktion ges, undantaget är att “Stopp!”- knappen kan användas för att stanna roboten då den rör sig framåt eller bakåt. Vid manuell styrning visas sensorvärdena, men inte kartväggar. Vid manuell styrning kan inte reglerparameter väljas.




Figur 2. Vid uppstart av programmet kommer denna ruta upp. För att starta styrningsläge klicka på “Starta Blåtand”.



Figur 3. Den ruta som är uppe under manuell styrning.

### *Nedstängning:*

Nedstängning sker genom att slå av huvudströmbrytaren på roboten och stänga av “HexBot.exe”.

# Autonomt läge:

***Uppstartssekvens:***

| Steg | PC | Robot |
|---|---|---|
| **1** | Starta datorn | Koppla in batteriet |
| **2** | Starta Blåtanden på datorn | Ställ brytaren i önskat läge (Se figur 1) |
| **3** | Starta “HexBot.exe” | Koppla in Blåtandstickan |
| **4** | ---------- | Slå till huvudströmbrytaren (Se figur 1) |
| **5** | ---------- | Tryck på Resetknappen (Se figur 1) |
| **6** | Tryck på “Starta Blåtand” | ---------- |
| **7** | Välj önskat reglervärde | ---------- |
| **8** | Tryck på “Skicka reglerparamter” | ---------- |

***Styrning:***

I autonomt läge går det inte att styra roboten via datorn, mer än att initiera reglerparametern.
Medan roboten utforskar världen så genereras kartan med robotens position och väggar samtidigt som sensorvärdena presenteras.

Roboten är utformad för att klara av en karta som uppfyller kraven i *Appendix A.*

# Varning!

Om roboten helt plötsligt vrider ihop sig beror detta på att batteriet är slut. Stäng av roboten på huvudströmbrytaren fortast möjligt för att undvika trasiga servon. Batteriet måste laddas innan fortsatt användning.

Robotens ben bör vara riktade utåt vid uppstart, man bör även hålla den en bit över marken. Detta för att undvika att den ligger på sina ben från start, eftersom även då riskerar servona att gå sönder.

# Bilaga 2 – Programlistning

Kodexempel från kommunikationsenheten:

```
void TIMER0init()
{
		//ställ in timer-prescaler till clk/1024
		TCCR0 |= (1<<CS00)|(1<<CS02);

		//enable interruptflagga ifall timern får overflow.
		TIMSK |= (1<<TOIE0);
}

ISR(TIMER0_OVF_vect)			//Här är "huvudloopen" för roboten
{
		switch(timersteg)
				{
					case 0: // TILLFÄLLIG STEGNING FÖR FELSÖKNING
						if(move_enable)
						{
						timersteg = 1;
						move_enable = 0;
						}
					break;

					case 1: // Begär att sensorenheten läser av sensorena

						data[0] = READ_SENSOR_DATA;
						if(i2c_writebyte(0x02))
						{
						timersteg = 2;
						}
					break;

					case 2: // Begär sensordata från sensorenheten

						data[0] = RECEIVE_SENSOR_DATA;
						if(i2c_writebyte(0x02))
						{
						timersteg = 3;
						}
					break;
```

```
                case 3: // Ta emot och skicka sensordata till PC

                                if(i2c_readbyte(0x02))
                                {
                                for(uint8_t i= 0; i < 6; i++)
                                                {
                                                send_data[i] = data[i];
                                                }

USART_Transmit(send_data, sensor_datatyp, 6);
timersteg = tillbakahopp; //Sätter återhoppsadress
                                }
                break;

                case 4:         // Kolla om styrenheten har gått klart

                                if(i2c_readbyte(0x01))
                                {
                                                if(data[0] == 1)
                                                {
                                                //om roboten gått klart
                                                timersteg = 5;
                                                }
                                                if(data[0] == 0)
                                                {
                                                // Hoppa till kont.reglering
                                                timersteg = 21;
                                                }
                                }
                break;
```

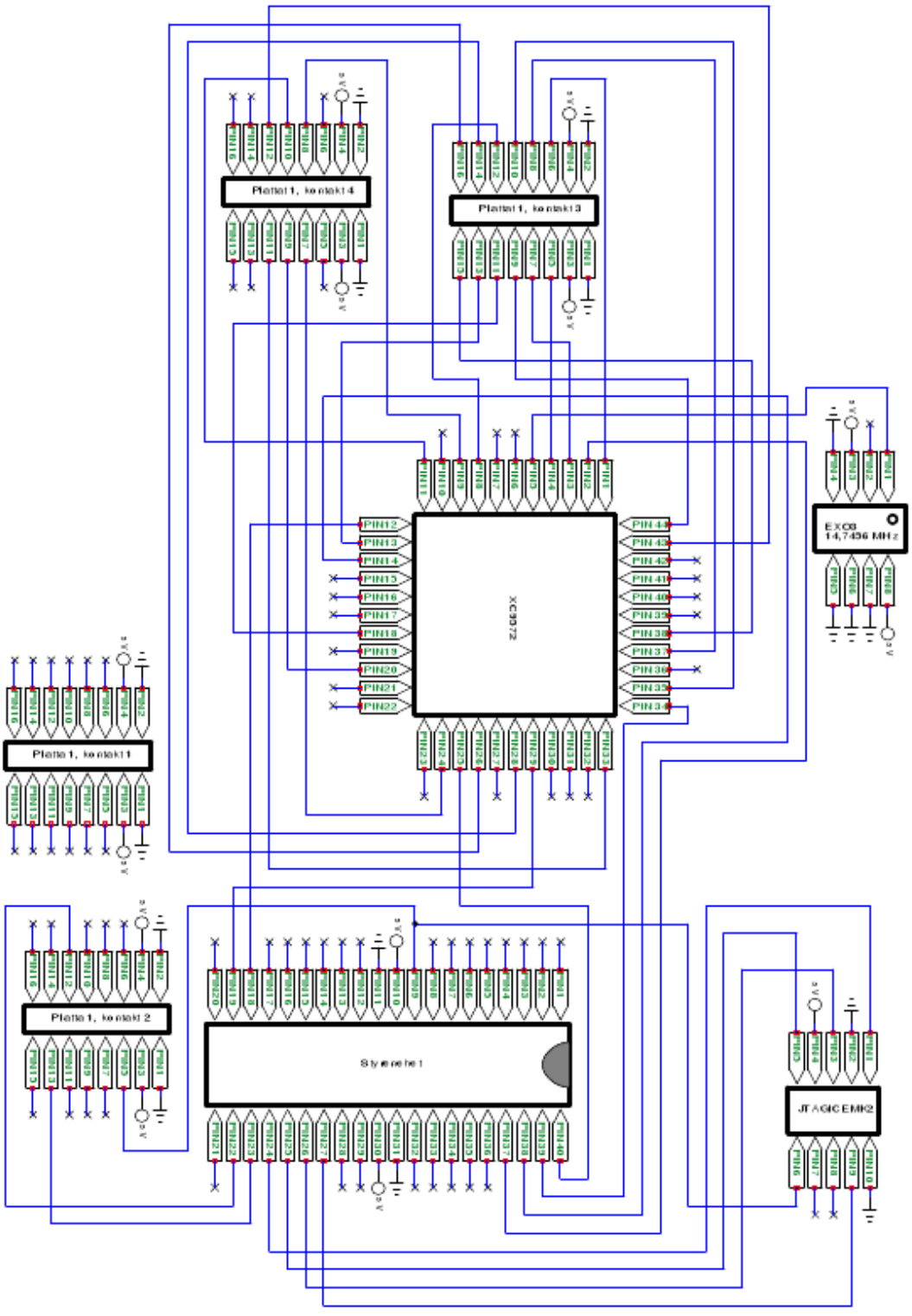
Kodexempel från sensorenheten:

```
//Den diskreta regleringen som endast används då roboten står mitt
//i en ruta
uint8_t regulation()
{
		//De senaste mätvärdena från de två sensorerna som är
		//riktade till höger, och de två som är fram och bak
		uint8_t ovre_sensor_r = get_measurements(direction_to_sensor(5));
		uint8_t undre_sensor_r = get_measurements(direction_to_sensor(4));
		uint8_t fram_sensor = get_measurements(direction_to_sensor(0));
		uint8_t bak_sensor = get_measurements(direction_to_sensor(3));


		//Om båda sensorerna ser väggar till höger skall den reglera om så krävs
		if(check_tolerance(ovre_sensor_r, side_wall_dist_1,0)
		 & check_tolerance(undre_sensor_r, side_wall_dist_1,0))
		{
				//Kollar om roboten pekar för mycket från väggen
				if(undre_sensor_r > (ovre_sensor_r + tolerance))
				{
						//rotera lite åt höger
						return 0x80;
				}
				//Kollar om roboten pekar för mycket mot väggen
				else if(ovre_sensor_r > (undre_sensor_r + tolerance))
				{
						//rotera lite åt vänster
						return 0x90;
				}
		}

		//Tittar om en vägg finns framför roboten i samma ruta men
		//är för långt ifrån
		if((fram_sensor < 0x53) & (fram_sensor > 0x35))
		{
				//Gå framåt
				return 0xA0;
		}

		//Tittar om en vägg finns bakom roboten i samma ruta men
		//är för nära
		if(bak_sensor > 0x6C)
		{
				//Gå framåt
				return 0xA0;
		}

		//Tittar om en vägg finns bakom roboten i rutan bakom roboten men
		//är för nära
		if((bak_sensor > 0x20) & (bak_sensor < 0x24)) // (Väggen finns i rutan bakom)
		{
				//Gå framåt
				return 0xA0;
		}
```

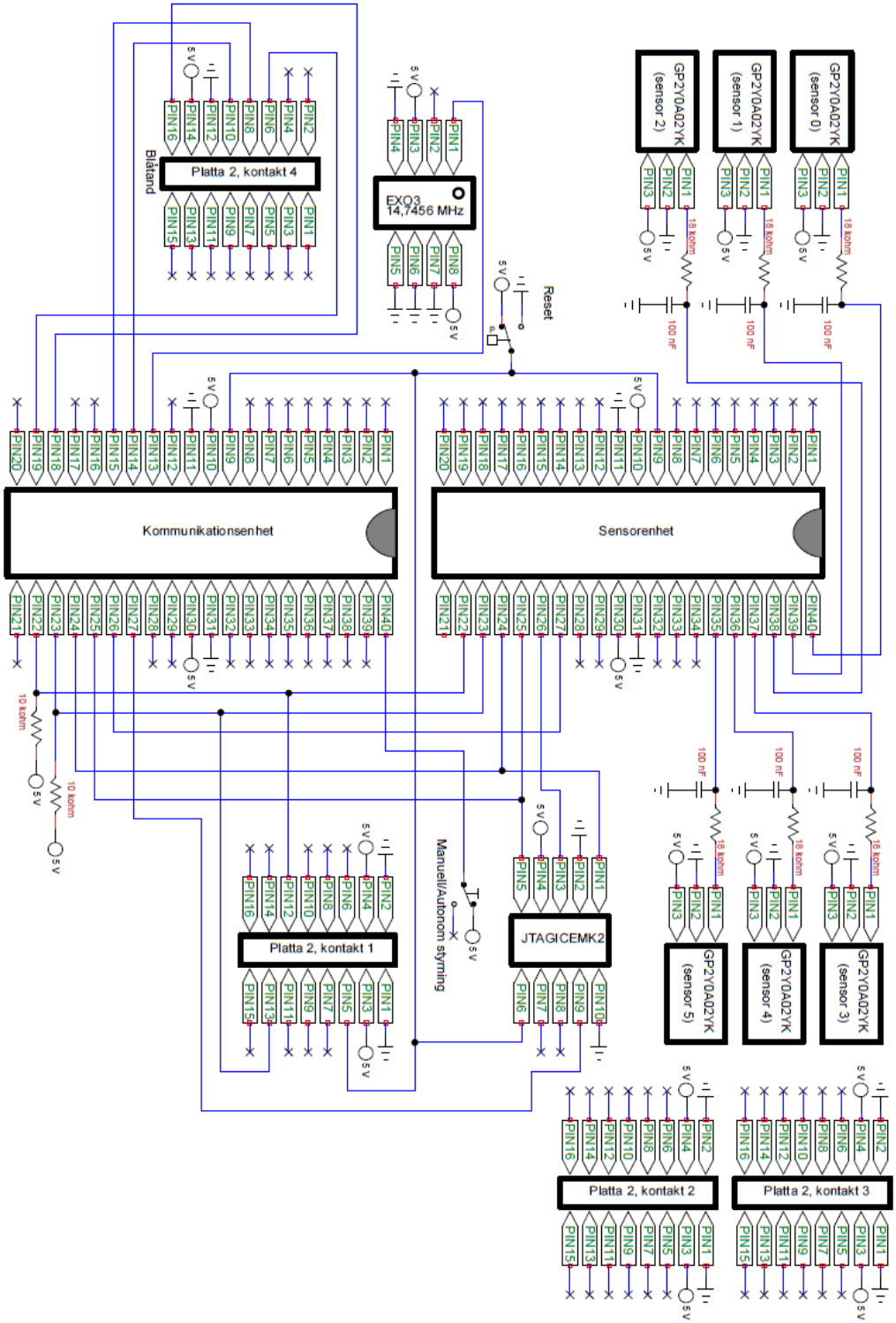
```
		//Tittar om en vägg finns framför roboten i samma ruta men
		//är för nära
		if(fram_sensor > 0x6C)
		{
			//Gå bakåt
			return 0xB0;
		}

		//Tittar om en vägg finns bakom roboten i rutan bakom roboten men
		//är för långt ifrån
		if((fram_sensor < 0x22) & (fram_sensor > 0x1B))
		{
			//Gå bakåt
			return 0xB0;
		}

		// Reglera inte
		return 0x60;
}
```

# Bilaga 3- Kopplingsschema

Platta 1



Platta 2


Kommunikationsenhet
Sensorenhet
Platta 2, kontakt 4
Platta 2, kontakt 1
Platta 2, kontakt 2
Platta 2, kontakt 3
Blåtand
EXO3
14,7456 MHz
Reset
JTAGICEMK2
Manuell/Autonom styrning
GP2Y0A02YK
(sensor 2)
GP2Y0A02YK
(sensor 1)
GP2Y0A02YK
(sensor 0)
GP2Y0A02YK
(sensor 5)
GP2Y0A02YK
(sensor 4)
GP2Y0A02YK
(sensor 3)
18 kohm
10 kohm
100 nF
5v