

# Designspecifikation

Redaktör: Patrik Nyberg

Version 0.1

## Status

Granskad		
Godkänd		

## PROJEKTIDENTITET

Grupp 1, VT14, Lagerrobot  
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Karl Linderhed	Projektledare (PL)	073-679 59 59	karli315@student.liu.se
Patrik Nyberg	Dokumentansvarig (DOK)	073 -049 59 90	patny205@student.liu.se
Johan Lind		070-897 58 24	johli887@student.liu.se
Erik Nybom		070-022 47 85	eriny778@student.liu.se
Andreas Runfalk		070-564 23 79	andru411@student.liu.se
Philip Nilsson		073-528 48 86	phini326@student.liu.se
Lucas Nilsson		073-059 42 94	lucni395@student.liu.se

**E-postlista för hela gruppen:** tsea56-2014-grupp-1@googlegroups.com

**Kontaktperson hos kund:** Tomas Svensson, 013-28 13 68, tomass@isy.liu.se

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

**Handledare:** Anders Nilsson, 3B:512, 013-28 26 35, anders.p.nilsson@liu.se

# Innehåll

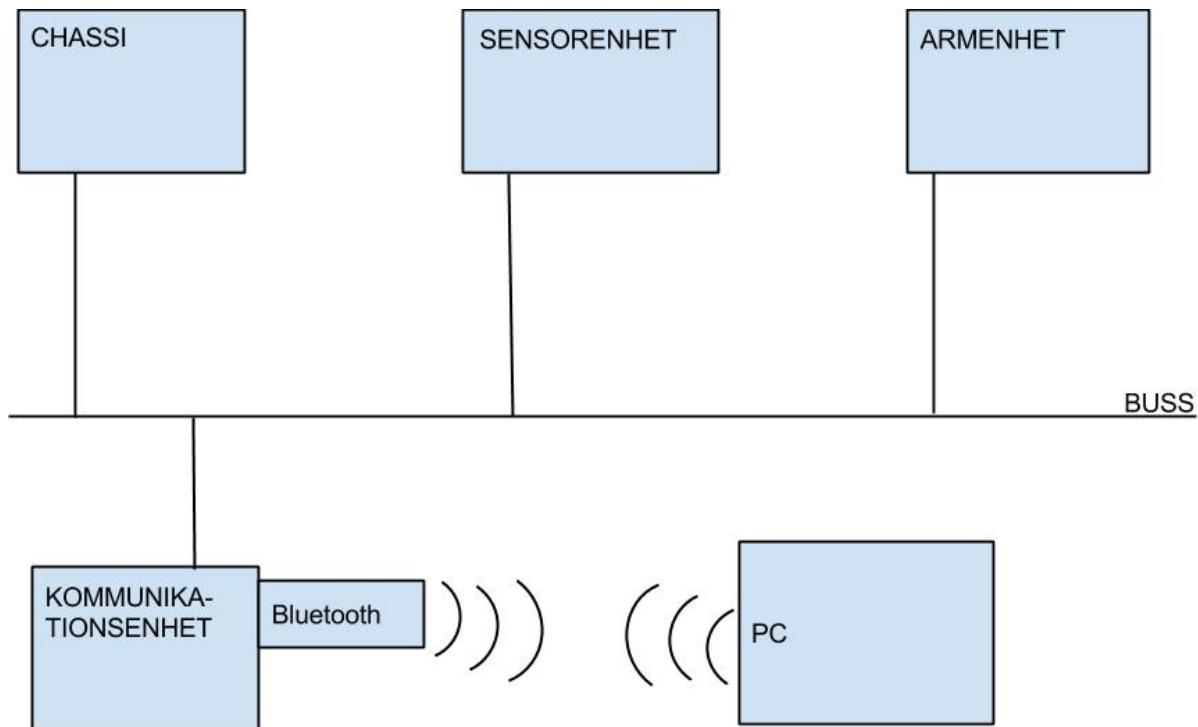
<b>1 Systemöversikt</b>	<b>1</b>
1.1 Delsystemsöversikt . . . . .	1
1.2 Sensorer och sensorplacering . . . . .	2
<b>2 Delsystem kommunikation</b>	<b>3</b>
2.1 Funktion . . . . .	3
2.2 Kopplingsschema . . . . .	3
2.3 Komponenter . . . . .	4
2.4 Analys av prestanda och resurser . . . . .	4
2.5 Översiktlig beskrivning av programmet . . . . .	4
<b>3 Delsystem sensorenhet</b>	<b>6</b>
3.1 Funktion . . . . .	6
3.2 Kopplingsschema . . . . .	6
3.3 Komponenter . . . . .	6
3.4 Sensorer . . . . .	7
3.4.1 Avståndssensorer . . . . .	7
3.4.2 Linjesensor . . . . .	7
3.4.3 RFID-läsare . . . . .	7
3.5 Sidoskanner . . . . .	7
3.6 Översiktlig beskrivning av programmet . . . . .	8
3.6.1 Sidoskanner . . . . .	8
3.6.2 Inläsning från avståndssensorer . . . . .	9
3.6.3 Inläsning från Linjesensor . . . . .	9
3.6.4 Korsning, avbrott och plockstationsdetektering . . . . .	10
3.6.5 Tyngdpunktsberäkning . . . . .	11
3.6.6 Detektering av objekt vid plockstation . . . . .	11
3.6.7 RFID-inläsning . . . . .	12
3.7 Analys av prestanda . . . . .	12
<b>4 Delsystem chassis</b>	<b>13</b>
4.1 Funktion . . . . .	13
4.2 Kopplingsschema . . . . .	13
4.3 Komponenter . . . . .	14
4.4 Översiktlig beskrivning av programmet . . . . .	14
4.4.1 Följa linje samt motorkontroll . . . . .	16
4.4.2 Identifiering av plockstationer samt trippmätning . . . . .	16
4.4.3 Styrkommando . . . . .	16
4.4.4 Autonomt läge och start-/stoppkommando . . . . .	16
4.5 Analys av prestanda och resurser . . . . .	17

<b>5 Delsystem arm</b>	<b>18</b>
5.1 Funktion . . . . .	18
5.1.1 Manuellt läge . . . . .	18
5.1.2 Autonomt läge . . . . .	18
5.1.3 Inverterad kinematik . . . . .	18
5.1.4 Förprogrammerade rörelser . . . . .	19
5.2 Kopplingsschema . . . . .	20
5.3 Komponenter . . . . .	20
5.4 Översiktlig beskrivning av programmet . . . . .	20
5.5 Analys av prestanda och minne . . . . .	21
<b>6 Programvara – PC</b>	<b>22</b>
6.1 Funktion . . . . .	22
6.1.1 Styrning . . . . .	22
6.1.2 Armrörelse . . . . .	23
6.1.3 Sensordata . . . . .	23
6.1.4 Grafer . . . . .	23
6.1.5 Debugfönster . . . . .	23
6.1.6 Loggfönster . . . . .	23
<b>7 Kommunikation mellan delsystem</b>	<b>24</b>
7.1 Protokollbeskrivning . . . . .	24
7.2 Analys av prestanda . . . . .	24
7.3 Informationsflöde . . . . .	25
<b>8 Implementeringsstrategi</b>	<b>26</b>
<b>A Master transmitt</b>	<b>27</b>
A.1 Master transmitt . . . . .	27
A.2 Master receiver mode . . . . .	28
A.3 Slave receiver mode . . . . .	28
A.4 Slave transmitter mode . . . . .	29
<b>B Bluetooth initiering och konfigurering</b>	<b>29</b>

**Dokumenthistorik**

<b>Version</b>	<b>Datum</b>	<b>Utförda förändringar</b>	<b>Utförda av</b>	<b>Granskad</b>
0.1	2014-03-10	Första utkast.	PaN	LN

# 1 Systemöversikt



Figur 1: Schematisk beskrivning av systemet

Roboten ska vara uppbyggd av fyra olika delsystem (chassi, sensor, arm och kommunikation) som vart och ett har sin specifika uppgift att sköta. Målet är att ett delsystem inte ska behöva känna till hur de andra delsystemen fungerar och att ett delsystem ska kunna bytas ut utan att andra delsystem behöver modifieras. Förutom robotens fyra delsystem kommer också en programvara till en dator att utvecklas för övervakning och styrning. Den kommer att både kunna operera i ett fjärrstyrts och i ett autonomt läge. I första hand ska autonomin endast innefatta linjeföljning och några förutbestämda armrörelser (t.ex. arm ut, arm in, lämna av föremål åt höger). Om det finns tid i projektet kommer autonom lokalisering och upplockning av föremål också implementeras.

## 1.1 Delsystemsöversikt

Sensorenheten ska kontinuerligt övervaka de olika sensorerna roboten har och omvandla dess primärdata till systemanvändbara enheter för att sedan kunna skicka dessa vidare till andra delsystem.

Kommunikationsenheten ska fungera som en förbindelseväg mellan omvärlden och roboten. Den ska kunna bli tillfrågad och skicka bekräftelse om de beslut som roboten gör samt status över de olika delsystemen och omgivningsobservationer.

Chassienheten fungerar som det handlingsbeslutande organet på roboten. Den ska vid autonomt läge veta hur roboten ska agera vid olika situationer. Den styrs också hjulen på roboten, det vill säga att den genom sensordata och reglering beräknar hastighet på hjulen för att kunna svänga och hålla kursen längst banan.

Armenheten ska vid en avhämtningsstation, utifrån identifiering av var ett objekt befinner sig, kunna plocka upp detta med robotarmen som är monterad på chassit. Det kommer ha kontroll över samtliga servon och vet hur dessa ska röras vid kommando. Med hjälp av inverterad kinematik ska den kunna, vid automatiskt läge, beräkna hur vardera led skall röra sig för nå objektet och för att plocka upp det.

PC-programvaran ska visa upp ett instrumentbräde för användaren att både se robotens beslut samt statusinformation. Här ska också roboten kunna styras med olika sorters kommandon som finns.

## 1.2 Sensorer och sensorplacering

En linjesensor ska sitta längst fram på roboten, nära marken. En RFID-läsare ska sitta parallellt med marken under roboten, då den endast har 7 cm läsavstånd till taggarna som ligger på marken. På vardera sida av roboten kommer en avståndssensor sitta placerad på ett servo, för att skanna efter objekt att plocka upp. På armen kommer det också att sitta en avståndssensor för att kunna göra upplockningen av föremål mer exakt.

## 2 Delsystem kommunikation

Kommunikationsenheten ska agera som ett gränssnitt mellan roboten och omvärlden.

### 2.1 Funktion

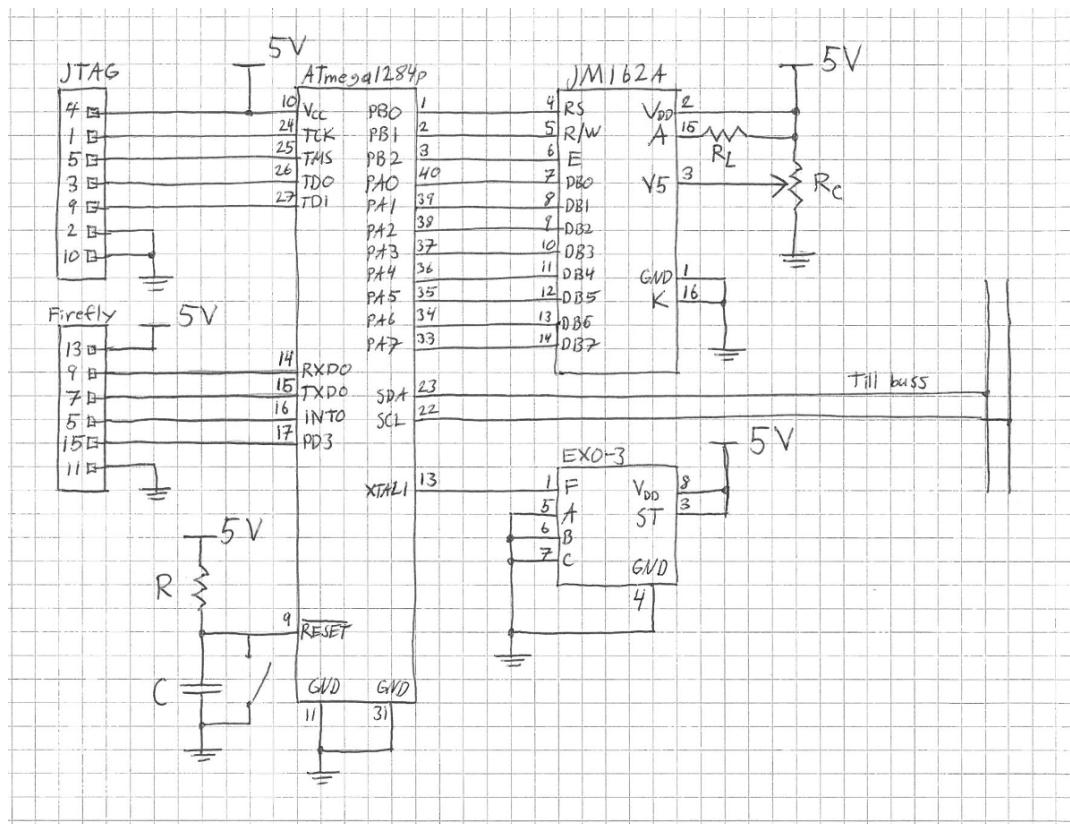
Kommunikationsenheten ska ha två uppgifter, att styra bluetoothkommunikationen och att hantera LCD-displayen. Kommunikationsenheten agerar helt passivt och initierar inga förfrågningar på egen hand, utan är bara en mellanhand för robotens kommunikation med datorn och LCD-displayen.

Bluetoothenheten som är kopplad till kommunikationsenhetens UART ska fungera som en virtuell seriell länk mellan PC och robotsystemet. Här ska information som roboten får in från sensorer samt vilka beslut roboten väljer att göra skickas via BT till PC.

LCD-displayen kan visa två rader med 16 tecken på varje. Det ska finnas ett enkelt gränssnitt där alla delsystem kan mata ut information på displayen med ett kommando. Kommunikationsenheten kommer att lagra vad respektive delsystem "säger" för närvarande, och roterar/växlar med någon sekunds intervall vilken enhets information som visas på displayen.

### 2.2 Kopplingsschema

I figur 2 visas ett kopplingschema över allt som ska finnas på kommunikationsenhetens virkort.



Figur 2: Kopplingsschema för delsystem kommunikation

## 2.3 Komponenter

- 1 x ATmega1284p - huvudprocessor
- 1 x EXO-3 oscillator (18.432 MHz, delas med 160 i USART-klockan för 115200 baud)
- 1 x JM126A LCD-display - för utmatning av debug/besluts/sensorinformation
- 1 x BlueSMiRF Gold Blåtandsmodem, "Firefly"
- 1 x tryckknapp för reset
- 2 x resistorer - pullup för reset-linan och strömbegränsning till displayens backlight
- 1 x potentiometer - för att justera kontrastspänningen till displayen
- 1 x kondensator - fördröjer pullup av reset vid strömtillslag

## 2.4 Analys av prestanda och resurser

I kopplingsschemat framgår att pinnarna på processorn räcker till och inte behöver användas till flera saker samtidigt.

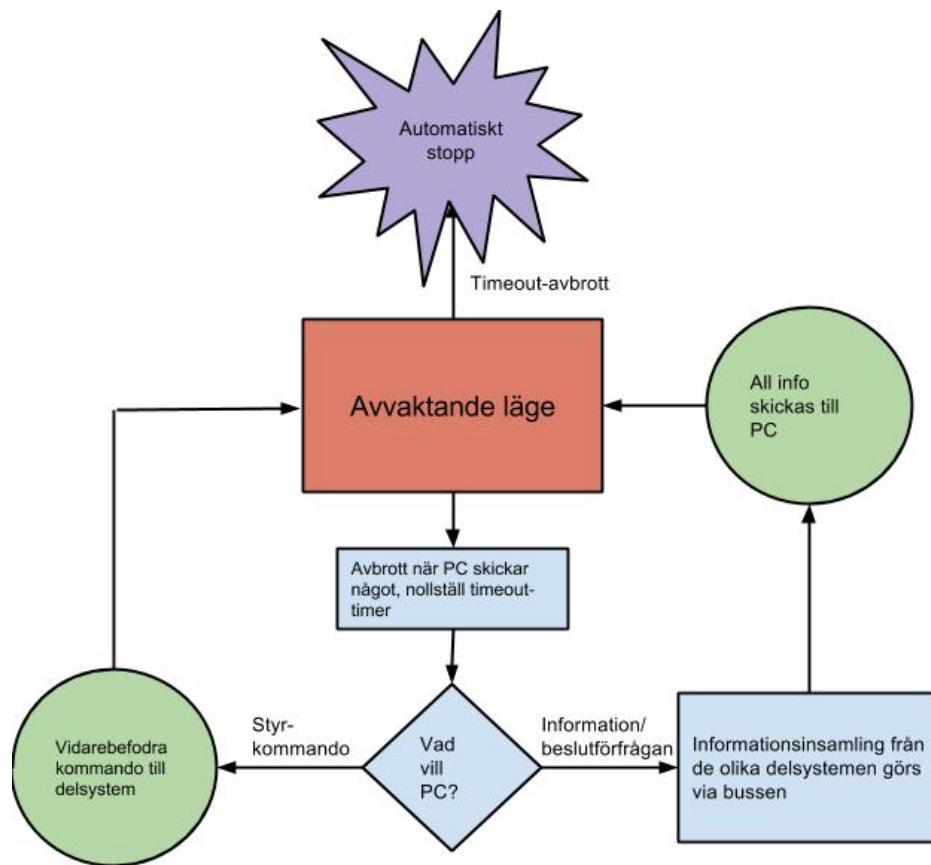
Kommunikationsenheten kommer att behöva använda två timers, en för att upptäcka om uppkopplingen till PC:n har förlorats och en för att växla mellan olika delsystems utmatning på displayen. ATmega1284p har två 8-bitars timers och två 16-bitars, så det finns inga begränsningar där.

## 2.5 Översiktlig beskrivning av programmet

För initiering och konfigurering av kommunikation mellan bluetoothenhet och processor, se bilaga B.

Kommunikationsenheten kommer att vara avvaktande tills dess att PC har skickat något kommando via BT till den. Från PC kommer den antingen få en informations/statusförfrågan eller ett styrkommando. Enheten har sedan till uppgift att antingen samla på sig den informations som frågas efter och skicka det till PC, eller att vidarebefodra det styrkommando som kommer in till lämplig enhet på bussen. Se figur 3. Datorn kommer även, om den inte håller på att skicka något, att skicka en "ping" till roboten. När roboten får någon data, kommandon eller ping, kommer en timer att nollställas. Om länken bryts av någon anledning så att timern inte nollställs kommer den att ge ett avbrott när den når ett visst värde, vilket leder till att kommunikationsenheten skickar ett general call på bussen om att alla enheter ska avbryta all rörelse.

När kommunikationsenheten får en begäran om att visa information på LCD-displayen kommer den att lagra informationen i minnet. Det är först när en timer går ut som texten på displayen kommer att växlas till nästa delsystem i ordningen och informationen visas.



Figur 3: Flödesschema över hur kommunikation med BT sker.

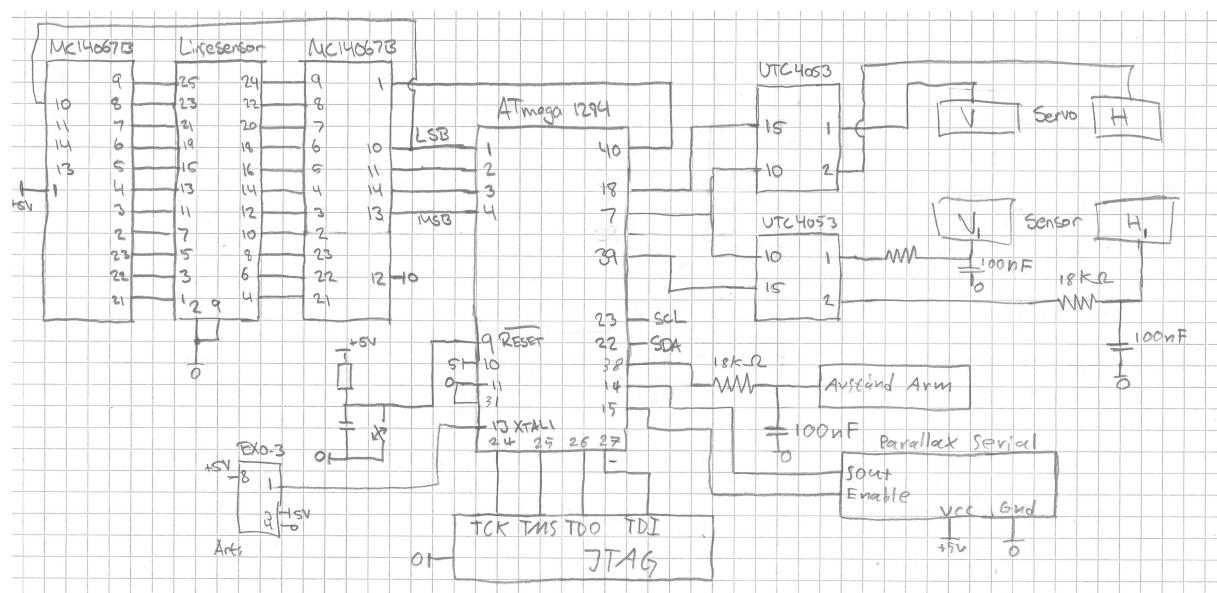
### 3 Delsystem sensorenhet

#### 3.1 Funktion

Sensorenhetens uppgift är att samla in rådata från de olika sensorerna och formatera denna så att den på begäran kan skickas ut till de andra delsystemen via bussen. Sensorenheten är utrustad med en linjesensor vars uppgift det är att ge styrdata som chassimodulen kan använda för att roboten ska kunna följa linjen. Vidare är roboten bestyckad med två stycken sidoskannrar som används för att lokalisera de objekt roboten ska plocka upp, samt en avståndssensor monterad på armen för finjusteringar när objekt plockas upp.

Eftersom att A/D-omvandlaren sannolikt kommer att vara en flaskhals i systemet så är det viktigt att optimera användningen av denna. Detta görs genom att använda sensorerna en och en, beroende på vilken uppgift det är som roboten för tillfället utför. När roboten följer linjen så kommer enbart linjesensorn att vara aktiverad medan endast en av de båda sidoskannrarna kommer att användas då roboten väl stannat. Efter att objektet har lokaliseras så slutar sidoskannern att jobba och den avståndssensor som är monterad på robotarmen kopplas istället in.

#### 3.2 Kopplingsschema



Figur 4: Kopplingsschema till delsystem sensorenhet.

#### 3.3 Komponenter

- 1 x ATmega1284P, huvudprocessor
- 1 x reflexsensormodul
- 2 x 16-kanals multiplexerar (MC14067B)
- 2 x 2-kanals multiplexerar (UTC 4053)
- 1 x RFID-läsare (Parallax Serial)
- 3 x avståndssensorer (GP2D120)

- 3 x resistorer ( $18\text{ k}\Omega$ ), för lågpassfilter
- 3 x kondensatorer ( $100\text{ nF}$ ), för lågpassfilter

## 3.4 Sensorer

Sensorenheten använder tre olika sensor typer: avståndssensor, linjesensor och RFID-läsare. Dessa beskrivs nedan.

### 3.4.1 Avståndssensorer

Avståndssensorer kommer användas dels till sidoskannrarna och dels till armen. Dessa mäter avstånd med hjälp av IR. Eftersom att avståndssensorerna först mäter ett digitalt värde och sedan skickar ut ett analogt värde så uppstår brus på utsignalen. Detta brus är relativt högfrekvent och kommer därför filtreras bort med hjälp av ett lågpassfilter enligt kopplingsschema i figur 4.

### 3.4.2 Linjesensor

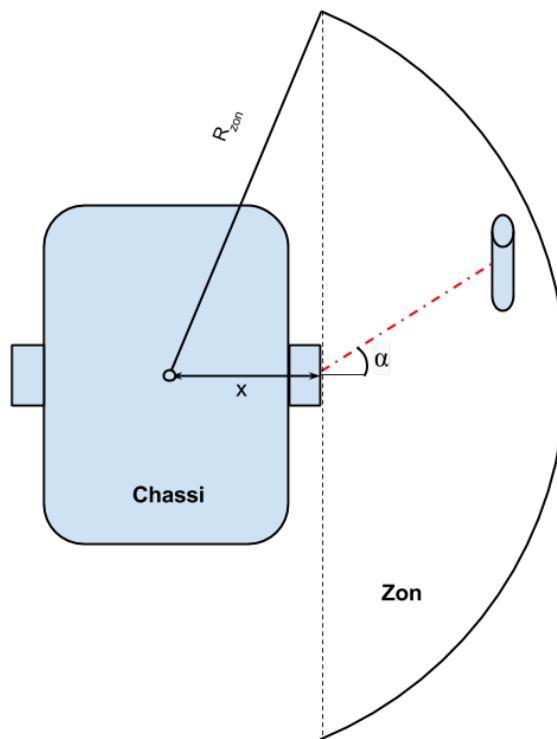
Reflexsensormodulen använder en uppsättning IR-dioder och fototransistorer för att mäta hur ljus en yta är. Eftersom reflexsensormodulen består av 11 stycken separata IR-dioder och fototransistorer kommer en multiplexer och en demultiplexer användas för att styra denna sensor, demultiplexern kommer att driva IR-dioderna, och multiplexern kommer att användas för att ta in insignalerna från fototransistorerna. Linjesensorn kommer att vara kopplad i enlighet med kopplingsschemat i figur 4.

### 3.4.3 RFID-läsare

Linjesensorn kommer detektera plockstationer. När detta sker anropas en subrutin som läser av RFID-sensorn och skickar ut RFID-värdet till chassienheten. Den RFID-läsare som kommer att användas är en ”Parallax Serial”, denna kräver två pinnar på processorn enligt kopplingsschema i figur 4.

## 3.5 Sidoskanner

Sidokannerns uppgift är att hitta det objekt som ska plockas upp när roboten har stannat vid en plockstation.



Figur 5: Översiktlig figur av sidoskanner.

Sidoskannern består av två avståndssensorer, på höger respektive vänster sida av roboten, monterade på varsitt servo. Två multiplexrar används för att välja huruvida det är höger eller vänster sida som ska avsökas. Servona styrs med pulsbreddsmodulering. Servot ska få en puls var 20:e millisekund och pulsbredeten avgör vilket läge servot antar. En pulsbredd på 1 ms motsvarar servots ursprungsläge och en pulsbredd på 2 ms motsvarar max vinkelutslag. Enheten ska ha en tabell över vilka pulstider som motsvarar vilka vinklar på servot. För att ställa servot i en vinkel som ligger mellan de vinklar som finns i tabellen så kommer enheten beräkna en approximerad pulsbredd med hjälp av linjärinterpolering.

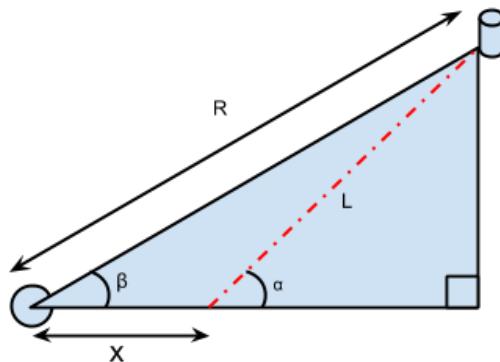
## 3.6 Översiktlig beskrivning av programmet

Sensorenheten ska initieras vid uppstart. Detta innefattar att konfigurera in- respektive utgångar, aktivera avbrott, initiera A/D-omvandlaren samt pulsbreddsmoduleringen. Utöver detta behöver även muxarna som är kopplade till linjesensorn initieras för att läsa av den första reflexsensorn. Efter detta kommer en A/D-omvandling startas. Sensoravläsning kommer sedan skötas kontinuerligt via avbrott som genereras av A/D-omvandlaren vid färdig avbrottssomvandling. Avbrotttsrutinen kollar sedan vilken kanal A/D-omvandlaren är inställd på, och uppdaterar på så sätt den för tillfället aktuella sensorn.

### 3.6.1 Sidoskanner

Sensorenheten ska med hjälp av sidoskannern kunna ta fram en koordinat för objektet som ska plockas upp. Koordinaterna bestäms utifrån ett koordinatsystem där origo ligger i mitten på roboten. När sidoskannern får in ett avstånd som är mindre än plockstationens radie ska

sidoskannerns vinkel noteras. Med hjälp av det uppmätta avståndet för ett visst vinkelutslag beräknas sedan objektets koordinat.



Figur 6: Visar vad koordinatberäkning för objektet begrundar sig på.

I figur 6 betecknar  $x$  avståndet från armens fäspunkt, alltså origo i koordinatsystemet, till avståndssensorn. Vidare betecknar  $R$  avståndet från robotarmens fäspunkt till det objekt som ska plockas upp. Utifrån vinkeln  $\alpha$  och avståndet  $L$  som fås från sensorenheten kan föremålets koordinat i planet med armens fäspunkt som origo beräknas genom:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X + L\cos(\alpha) \\ L\sin(\alpha) \end{pmatrix}$$

### 3.6.2 Inläsning från avståndssensorer

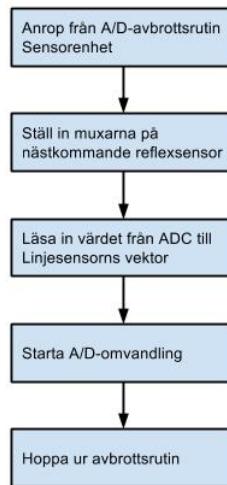
En avläsning från en avståndssensor görs genom att omvandla sensorns analoga utspänning till ett digitalt värde. Värdet jämförs sedan med en tabell över kända avstånd och spänningar för att översätta den A/D-omvandlade spänningen till ett avstånd. Värden som inte finns i tabellen kommer beräknas med linjärinterpolering.

### 3.6.3 Inläsning från Linjesensor

Inläsning från linjesensorn kommer ske genom ett funktionsanrop (via avbrott från A/D-omvandlaren) till linjesensorinläsningsfunktionen som finns på sensorenheten. När denna funktion anropas uppdateras en uppsättning variabler som är lagrade i en vektor. Vektorn kommer vara 11 element lång, där varje element representerar en reflexsensor.

När uppdateringen av linjesensorn startas för första gången kommer muxarna som styr linjesensorn ställas in så reflexsensorn längst till vänster väljs. Efter detta startas A/D-omvandling på kanal 0. När A/D-omvandlingen är färdig kommer ett avbrott (ADIF) att genereras vilket triggar en avbrottsrutin.

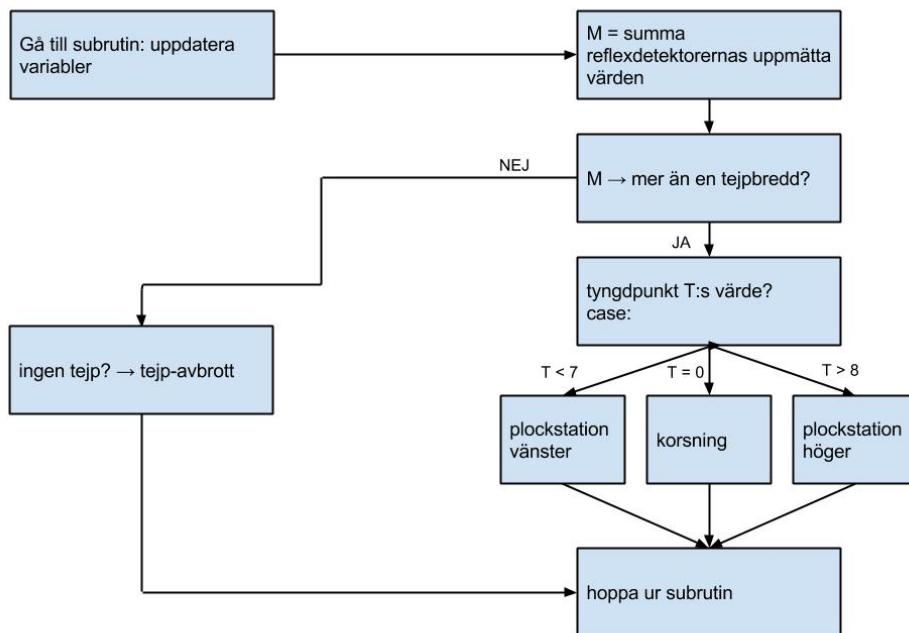
I denna avbrottsrutin skrivs det A/D-omvandlade värdet in på korrekt plats i linjesensorns datavektor. Denna plats motsvarar den valda linjesensorns position på sensormatrisen. Därefter uprepas samma process för nästkommande sensor. Figur 7 illustrerar processen för att uppdatera linjesensorns data.



Figur 7: Programflöde vid uppdatering av linjesensorn

### 3.6.4 Korsning, avbrott och plockstationsdetektering

Varje gång linjesensorn läses av uppdateras viktvektorn och en ny tyngdpunkt beräknas utifrån denna. Det lagras också olika flaggor för vilken status tejpen ligger i, som kommer att returneras med tyngdpunkten när chassisenheten begär sensordata. Det kommer att sättas en flagga för om roboten har kommit till en korsning eller inte (se figur 8), en för eventuellt avbrott i tejpen, om det har kommit en plockstation eller inte och en för vilken sida plockstationen befinner sig på.



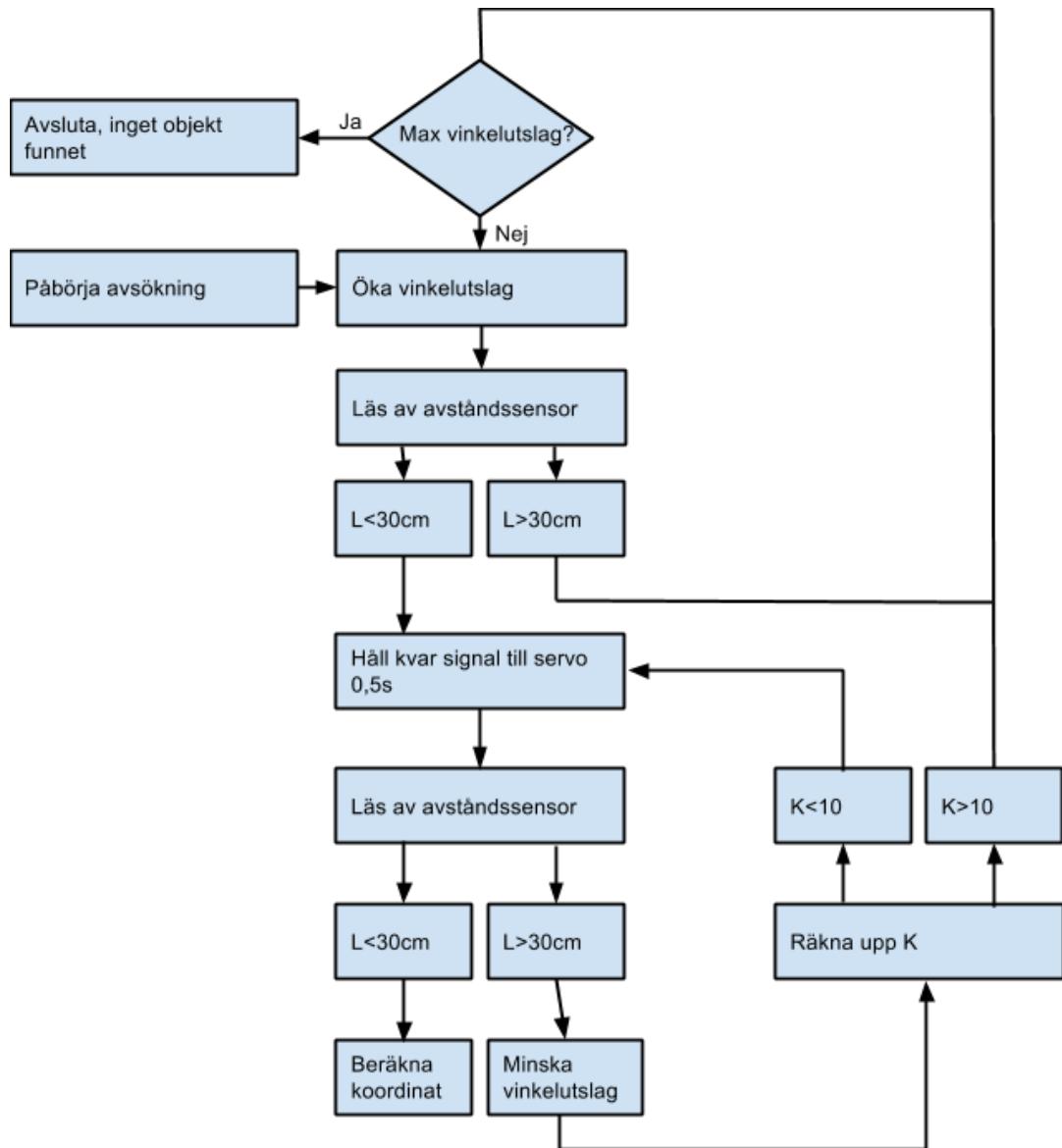
Figur 8: Programflöde för att upptäcka korsning eller plockstation

### 3.6.5 Tyngdpunktsberäkning

Tyngdpunktberäkning sker i en separat subrutin. Varje gång rutinen körs omvandlas sensorvärdena till en byte innehållande ett värde mellan -127 och 128. -127 innehåller att tyngdpunkten ligger längst till vänster på sensorn och 128 innehåller att den är längst till höger. Om värdet är 0 ligger tyngdpunkten precis i mitten av linjesensorn.

### 3.6.6 Detektering av objekt vid plockstation

Det program som kommer att koordinatbestämma objektet kommer att fungera enligt figur 9 programflöde:



Figur 9: Programflöde för detektion av objekt vid plockstation

I denna figur är  $L$ , som tidigare beskrivet, det av avståndssensorn uppmätta avståndet till det föremål som ska plockas upp. Ifall avståndssensorn av någon anledning returnerar ett felaktigt

värde där sensorn antingen detekterar ett föremål, trots att det inte finns där, eller sveper förbi det föremål som sensorn letar efter så använder den sig av parametern  $K$ . Till att börja med stegas servot tillbaka i små steg för att se ifall föremålet missats. Om fler än 10 steg hinner tas innan ett föremål hittas så startas objektsökningen om. Observera att just värdet 10 kan komma att justeras, beroende på hur stora steg vinkelutslaget ändras.

### 3.6.7 RFID-inläsning

När sensorenheten upptäckt att roboten är på en station körs en rutin där RFID-läsaren aktiveras och ett värde läses in och sparas i minnet tills dess att chassienheten efterfrågar det.

## 3.7 Analys av prestanda

A/D-omvandlaren går att klocka i upp till 200 kHz utan att tappa upplösning. RFID-läsaren kommunicerar dock seriellt med en baud-rate på 2400 bps, för att få maximal hastighet på A/D-klockan och samtidigt kunna läsa med exakt 2400 bps hade en processorklocka på 11.0592 MHz önskats. Denna gick dock inte att få fram med de oscillatorkretsar som finns tillgängliga i projektet. Detta innebär att klockfrekvens väljs till 20 MHz istället på systemklockan. Då fås en kompromiss som gör att kommunikationen med RFID-läsaren fungerar samtidigt som A/D-omvandlarens frekvens maximeras inom ramen för vilka frekvenser processorn klarar av. Resultatet blir att A/D-omvandlaren får en klockfrekvens på ca 150 kHz. Ytterligare en fördel med att klocka processorn i 20 MHz är att processorn då hinner med så många instruktioner som möjligt medan A/D-omvandlaren är upptaget med att omvandla sensordata.

Samplingstakten av de olika sensorerna kommer alltså att bli beroende dels av hur snabbt A/D-omvandlarens klocka går, samt hur många klockcykler det tar för A/D-omvandlaren att omvandla ett analogt värde. Detta gör att en A/D-omvandling, och således en sensorsampling, kommer att ta i storleksordningen  $10^{-4}$  sekunder.

De timer som kommer att användas är dels till för att åstadkomma pulsbreddsmodulering till sidoskannrarna samt för att ställa servona i önskat läge. Vidare är resterande pinnar och funktioner på processorn redogjorda för i figur 4.

## 4 Delsystem chassis

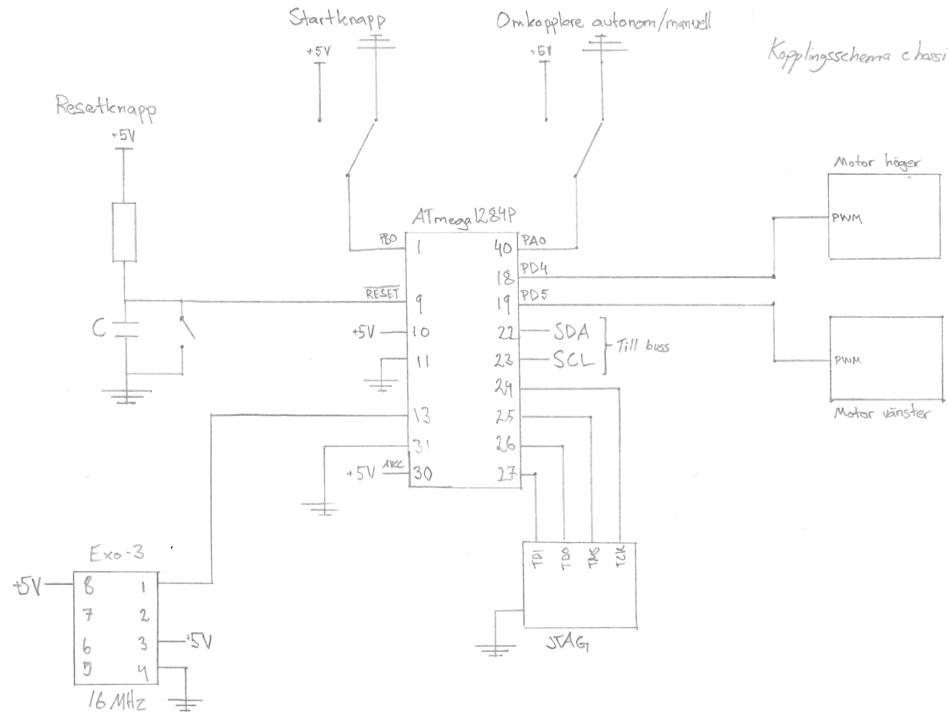
Chassienheten tar alla övergripande beslut om vad som ska göras, t.ex. är det chassienheten som bestämmer när armen ska plocka upp objekt. Den kommer att sköta regleringen av styrningen med hjälp av den tyngpunkt som fås av sensorenheten via bussen. Detta för att kunna följa en tejpad linje på marken. På chassienheten kommer det även att finnas två omkopplare, en för att informera roboten om att den ska påbörja sin tävlingsprocedur, och en som anger om roboten ska operera autonomt eller styras via fjärrkommandon.

## 4.1 Funktion

Chassit ska uppfylla flera funktioner, som anges i punktlistan nedan.

- Kunna ta beslut och styras både autonomt och genom order från dator via kommunikationsenheten.
  - Styra de två motorerna med PWM-styrning.
  - Använda regleralgoritm för att kunna följa linjen utan att slingra sig fram.
  - Hantera plockstationer genom att stanna och skicka kommandon till arm- och sensorenheten.
  - Kunna vända autonomt.
  - Mäta tid mellan stationer och m.h.a. detta ta beslut om att vända eller inte.
  - Skicka styrbeslut till dator via kommunikationsenheten.

## 4.2 Kopplingsschema



Figur 10: Kopplingsschema för delsystem chassis

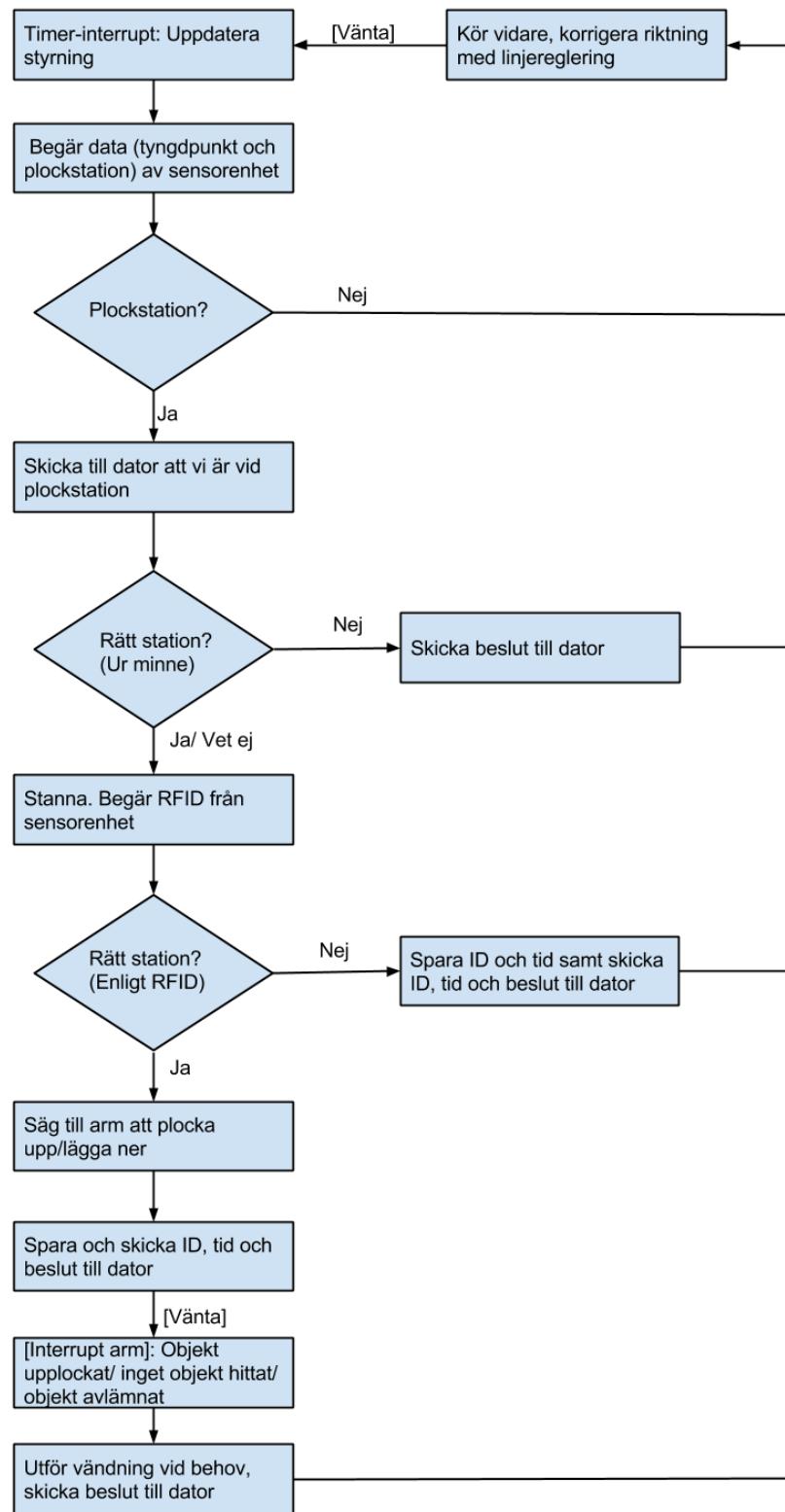
## 4.3 Komponenter

- Färdig plattform med batteri, två motorer som driver varsitt hjul och två "kundvagnshjul"
- 1 x Atmega1284P, huvudprocessor
- 2 x tryckknappar för reset samt start/stopp.
- 1 x omkopplare, för autonomt/manuellt läge.
- 1 x EXO-3, kristalloscillator (16 MHz)

## 4.4 Översiktlig beskrivning av programmet

Programmet på chassit kommer med hjälp av tyngdpunkten att beräkna styrriktning. Den kommer hämta nya värden från sensorenheten med jämna tidsintervall och med dessa uppdatera motorstyrningen. Enheten kommer också mäta tid mellan plockstationer, deras ordning samt lyssna på styr- och stoppkommandon.

Chassit får information om att roboten är på plockstation av sensorenheten i samma paket som den får tyngdpunkten. Vid plockstation skall roboten stanna och en begäran om att få RFID-tag skickas till sensorenheten. Efter detta skickas instruktion till armenheten att antingen plocka upp eller lämna av föremål. När det är dags att köra vidare tar chassit beslut om ifall den ska vända eller inte. Alla dessa beslut skickas via kommunikationsenheten till datorn.



Figur 11: Flödesdiagram för huvudprogram till chassienheten.

#### 4.4.1 Följa linje samt motorkontroll

För att kontrollera hastighet till motorerna kommer snabb PWM att implementeras i roboten. Denna kommer att vara kopplad till en H-brygga (inbyggd i den givna plattformen) som sköter matningen till själva motorerna. Motorerna kommer att uppdateras med en uppdateringsfrekvens på 1 kHz.

Eftersom systemets klocka kommer gå i 16 MHz måste timern ställas in så att den kan uppnå en period på 1 kHz i uppdateringsfrekvens. Detta uppnås genom att prescalern ( $N$ ) till timern sätts till 1 samt att timern räknar till 15999 innan en ny period ska starta.

För att kunna göra linjeföljning utan att slingra sig fram måste en regleralgoritm implementeras. Den reglering som kommer att göras är en PD-reglering, enligt formlen:

$$u[t] = K_p e[t] + K_d \frac{d}{dt} e[t]$$

Om värdet är positivt/negativ ska motor på vänster/höger sida att bromsas medan den andra kör med full fart. För att beräkna motorkontrollen till PWM kommer följande formel användas.

$$P[t] = 15999 - K_{total} u[t]$$

Där alltså  $P[t]$  är värdet räknaren till vänster/höger motor ska räkna till. Den andra motorn kommer att gå på full hastighet, alltså är  $P[t] = 15999$  för den motorn. Av dessa värden kommer  $K_p, K_d$  och  $K_{total}$  att kunna ändras för att optimera regleringen.

#### 4.4.2 Identifiering av plockstationer samt trippmätning

Varje gång roboten stannar vid en plockstation kommer chassis att, via sensorenheten, läsa av plockstationens RFID-tagg. Värdet hos dessa RFID-taggar kommer sparas i en tabell innehållandes dels taggarnas värden och dels hur långt det är mellan taggarna tidsmässigt. Detta gör att roboten kan vända om vägen till plockstationen som den ska till blir kortare då.

#### 4.4.3 Styrkommando

Chassis ska reagera på styrkommandon från bussen om brytaren är satt i manuellt läge. Ett styrkommando består av ett rattutslag och ett gaspådrag. Ett negativt rattutslag innebär vänstersväng och ett positivt innebär högersväng. Gasståndet kan vara positivt och negativt, där positivt för roboten framåt och negativt för roboten bakåt.

#### 4.4.4 Autonomt läge och start-/stoppkommando

När brytaren är i autonomt läge ska chassis reagera på startknappen som sitter på chassis. Startknappen påbörjar linjeföljning med hantering av plockstationer beskrivet i 4.4.1.

## 4.5 Analys av prestanda och resurser

Chassienheten kommer behöva 3 timer. En för att mäta tid för trippmätaren, en för PWM-styrning av motorerna och en som ger avbrott för att begära linjesensordata med jämn intervall. En av ATmega1284:ans 8 bits-timers kommer att användas för att begära sensordata, och de två 16 bits-timrarna kommer att användas till trippmätare respektive PWM. I övrigt kommer chassit att beräkna gaspådrag till de olika motorerna med en regleralgoritm. Eftersom värdet på linjesensorn kommer uppdateras med intervall på 2 ms kommer det finnas tillräckligt med tid för att göra denna beräkning.

## 5 Delsystem arm

Robotens arm är av modell PhantomX Reactor från Trossen Robotics, som är en servostyrd arm med 4 rotationsleder och en griphand varpå det sitter totalt 7 st AX-12A-servon. Armen kontrolleras av en microprocessor, ATmega 1284, genom att parallellkoppla servona till en seriell UART-port, via en tri state buffer så att styrningen till servona sker med half duplex. Armen har en räckvidd på 38 cm och en rotationsfrihet på 300 grader.

### 5.1 Funktion

Armen ska att kunna köras i antingen manuellt eller autonomt läge. Den kommer vara avvaktande tills dess att ett kommando skickas på bussen till armenheten. Det kan antingen vara ett visst styrkommando eller en koordinat där objektet som ska plockas upp befinner sig.

#### 5.1.1 Manuellt läge

Vid manuellt läge får armenheten kommandon skickade från PC via kommunikationsenheten, som armen rör sig utefter. Ett kommando kommer se ut på det viset att den talar om vilken riktning armen ska röra sig, i djupled (Y), i höjd (Z), rotationen runt Z eller klogrip. Därefter rör den på sig tills dess att annan instruktion tilldelas eller tills att den nått max-läge. Den ska också kunna få en instruktion om att den automatiskt ska röra sig till ett startläge genom inverterad kinematik (se 5.1.3).

#### 5.1.2 Autonomt läge

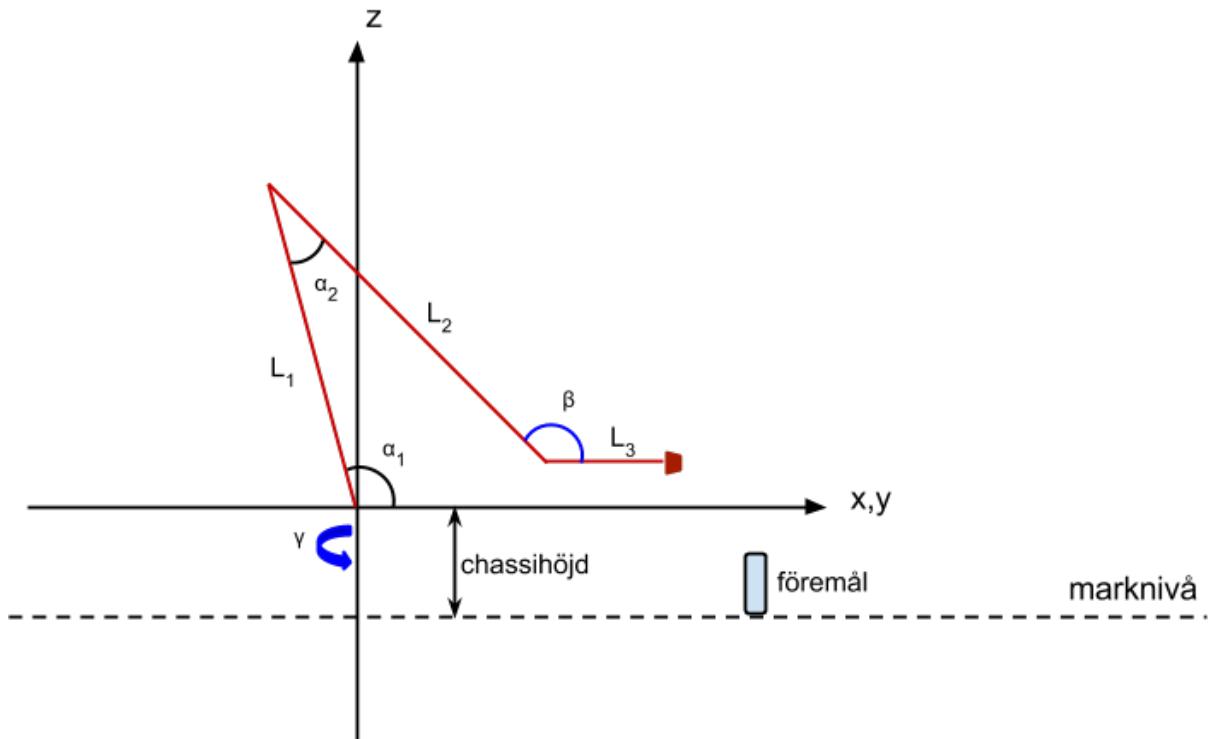
Vid autonomt läge ska armenheten befina sig i ett avvaktande läge tills dess att den får ett kommando från chassienheten att den befinner vid en plockstation. Armen kommer då fråga sensorenheten om objektets position. Som svar kommer den få en vinkel, avstånd från origo, och position i höjdled. Detta svar innehåller också information om ifall det finns ett objekt på plockstationen. Utifrån detta skall enheten göra beräkning på hur armen behöver röra sig för att nå koordinaten med klon. Detta görs genom inverterad kinematik (se 5.1.3).

För att kunna förutsätta att armen plockar upp ett föremål med säkerhet och försiktighet så rör sig armen först till ett läge en bit framför slutkoordinaten för att sedan tillsammans med respons från en avståndssensor på klon sakta röra sig fram till objektet och greppa det. När den rör sig närmare objektet frågar armenheten koninuerligt sensorenheten om avståndet till objektet för att utefter det besluta om att fortsätta. Efter greppning av föremål rör den sig tillbaka till startläget på samma sätt. När armen är i utgångsposition igen, eller om inget föremål hittades, skickas kommando till chassienheten om föremål plockades upp eller inte. Därefter kan chassienheten välja att köra igen.

#### 5.1.3 Inverterad kinematik

Positioner anges i ett koordinatsystem relativt armens bas. X anger avstånd i sidled, Y anger avstånd i djupled och Z anger avstånd i höjdled. Armens gripklo hålls horisontell mot underlaget. Då armens rotation i XY-planet endast styrs av armbasens rotation kan det inverterade kinematikproblemet reduceras till ett tvådimensionellt problem. Eftersom griplederna ska hållas

horisontell kan även denna led tas ur ekvationen, då den anpassas efter de övriga två ledernas position.



Figur 12: Figur över koordinatsystem och vinklar som definierar armens läge.

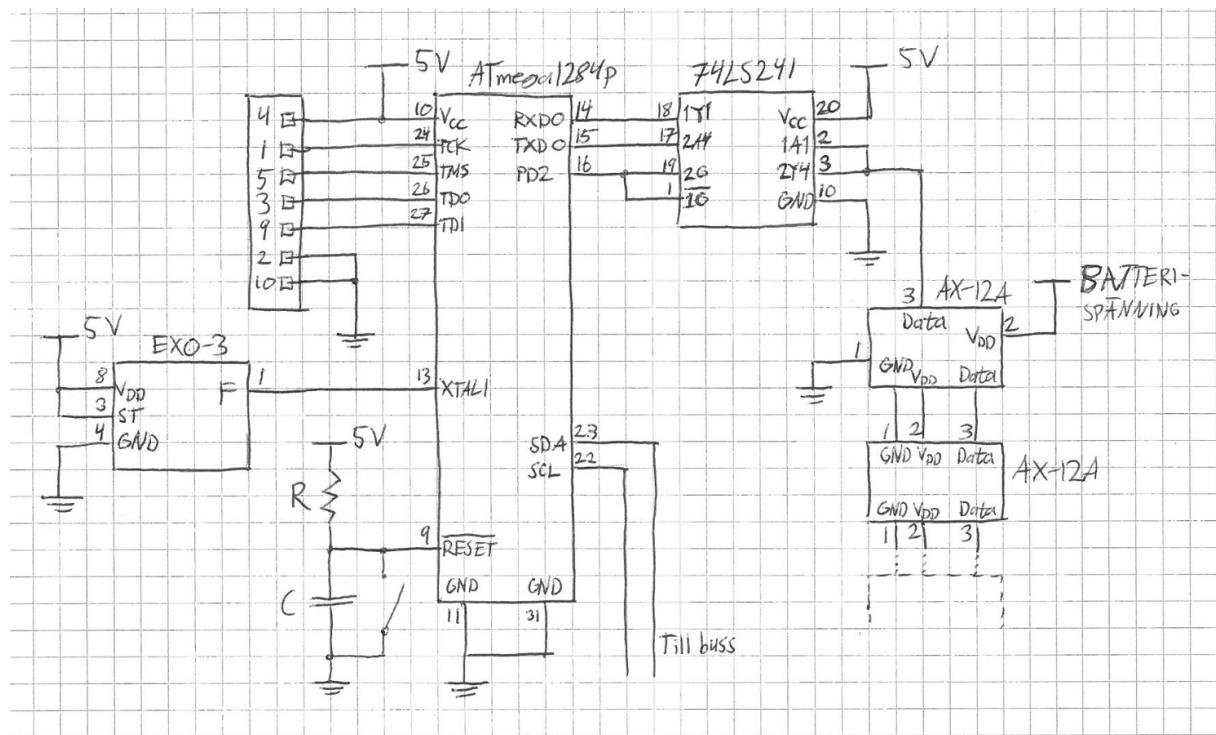
I figur 12 ses att origo ligger över marknivå.  $\gamma$  kommer vara inställd i den riktning som armen ska röra sig i för att nå föremålet.  $\beta$  kommer se till att hålla  $L_3$  horisontell mot marknivån.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 \cos(\alpha_1) + L_2 \cos(\alpha_1 + \alpha_2 - \pi) \\ L_1 \sin(\alpha_1) + L_2 \sin(\alpha_1 + \alpha_2 - \pi) \end{pmatrix}$$

#### 5.1.4 Förprogrammerade rörelser

När gripklon har fått grepp om föremålet kommer armen att kunna återgå till sitt ursprungsläge genom ett förprogrammerat rörelsemönster. Armenheten kommer även, vid avlämningsstationer, kunna ta emot kommandot "lämna av föremål". Detta kommando kommer starta ett förprogrammerat beteende hos armens servon för att lämna av föremålet autonomt.

## 5.2 Kopplingsschema



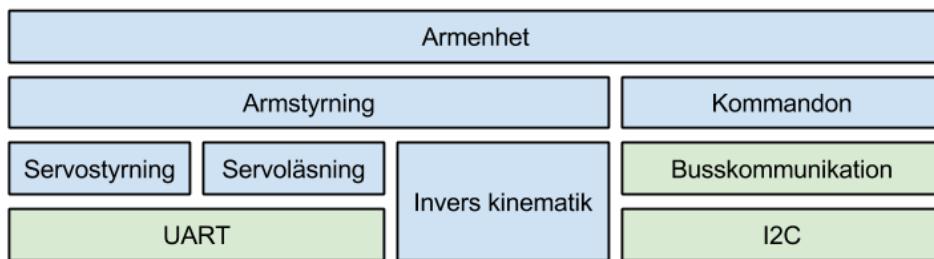
Figur 13: Kopplingsschema över delsystem arm

## 5.3 Komponenter

- 1 x PhantomX Reactor arm med 7st AX-12A-servon
- 1 x 3-state buffer (74LS241)
- 1 x ATMega1284P, huvudprocessor
- 1 x EXO-3, kristalloscillator (16 MHz)
- 1 x Resistor, för pullup av reset
- 1 x Kondensator, för fördräjning av resetpullup

## 5.4 Översiktlig beskrivning av programmet

Programmet kommunicerar med servona över UART. Med hjälp av inverterad kinematik kan programmet beräkna vilka vinklar armens servon ska anta för att armens gripklo ska anta en given position. Nedanstående figur visar vilka bibliotek som programmet ska bestå av. Grönmarkerade lådor innebär att de kan delas med andra delar av roboten. Lådor som ligger ovanför en annan låda bygger på den lådans funktioner. Exempelvis är servostyrning och servoläsning beroende av funktioner för att kommunicera över UART.



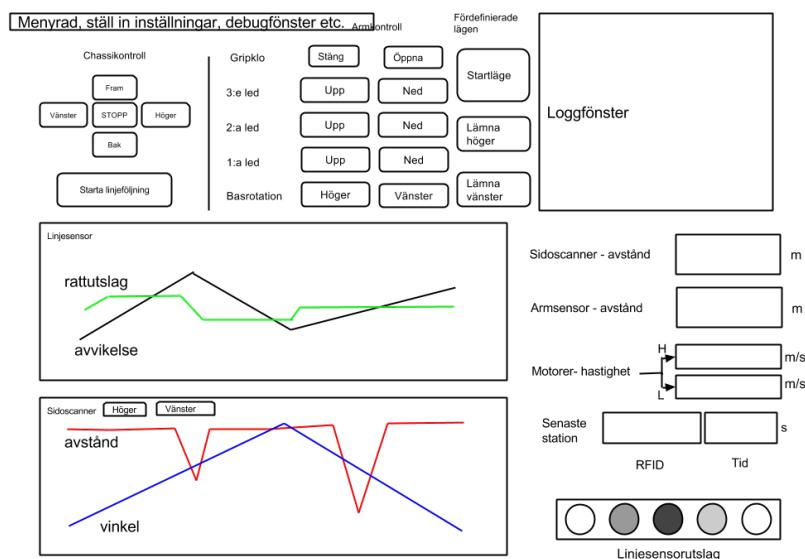
Figur 14: Övergripande bild över armenhetens programbibliotek.

## 5.5 Analys av prestanda och minne

Av kopplingsschemat framgår att antalet pinnar räcker till på processorenheten. Vald klockfrekvens garanterar att rätt baudrate kan användas för den seriella kommunikationen med armens servon och även över bussen. Enheten behöver inte använda några av processorns AD-omvandlare eller timers.

## 6 Programvara – PC

Gränssnittet ska användas för att en människa ska kunna kommunicera med roboten. Detta innebär att i realtid redovisas alla relevanta styrbeslut som roboten tar, så som fart och riktning. Utöver detta skall gränssnittet även representera världen ur robotens synvinkel, alltså ska det på ett förståeligt sätt uppvisa och formatera de sensordata som sensorenheten samlar in för att ge en bild av lagerrobotens omgivning. För att kunna testa delsystems beteende i grunden finns ett debug-fönster där en förfrågan kan skickas till ett delsystem för att se vilket svar som returnerades.



Figur 15: Användargränssnitt för PC-programvara

Programvaran kommer att ha två huvudsakliga uppgifter. För det första så ska den kontinuerligt visa upp relevant data och information från roboten oavsett om roboten opererar i autonomt eller manuellt läge. För det andra så ska programvaran kunna skicka styrbeslut till roboten när den befinner sig i manuellt läge. All kommunikation sköts trådlöst via Bluetooth. Kommunikationsenheten skickar vidare datorns kommandon till rätt enhet. Programvaran ska också periodiskt skicka ett enkelt paket till roboten som kommer att nollställa en timeout-räknare, för att möjliggöra att roboten kan upptäcka om kontakten förloras.

### 6.1 Funktion

Programvaran skall möjliggöra manuell styrning av robotens arm, manuell styrning av robotens framfart och åskådliggöra mätdata från sensorer samt de beslut roboten tar.

#### 6.1.1 Styrning

Med knappar i det grafiska instrumentbräde och/eller tangentnedtryckningar kan användaren styra roboten. Genom att trycka ner en styrknapp så ökar farten i den önskade riktningen, det

vill säga, för att åka framåt följt av en högersväng så ska först farten ökas framåt sedan i högerriktnings. Det betyder också att farten sänks i någon riktning genom att öka farten i motsatt riktning. Om höger- och vänsterstyrning görs när roboten står stilla, kommer roboten rotera på plats. En stoppknapp finns för att stanna roboten. Det finns också en linjeföljningsknapp som gör att när roboten kör så kommer den automatiskt att följa linjen på banan. I linjeföljningsläge kommer höger- och vänsterstyrning endast fungera när roboten står stilla och då kommer roboten att rotera.

### 6.1.2 Armrörelse

På instrumentbrädet skall också användaren kunna kontrollera armens rörelse genom knappar eller tangentnedtryckningar. Armens rörelse kommer kunna styras genom rotation av basen och tre leder i själva armen. Gripklon ska kunna öppnas och stängas. Dessa styrningar kommer fungera så att roboten rör på sig så länge du håller i knappen för önskad styrning. Det kommer också finnas ett gäng olika fördefinierade lägen för armen som den automatiskt ska kunna röra sig till genom att användaren trycker på önskat läge. Några exempel på fördefinierade lägen är:

- Startläge, ett infällt läge där roboten kommer vara redo att köra vidare.
- Avlämning höger/vänster, ett mönster där roboten kommer att lämna av ett föremål på någon sida.

### 6.1.3 Sensordata

I programvaran ska en rad olika sensordata finnas representerade. Dessa ska uppdateras i realtid så att användaren lätt kan se kvantitativa uppgifter om hur roboten rör sig samt sensordata och stationsinformation.

### 6.1.4 Grafer

Ett antal utvalda sensordata kommer att lagras över tid och visas på ett diagram som kontinuerligt uppdateras med tiden. Detta ska möjliggöra för användaren att tydligt kunna se och förstå hur roboten har rört sig och vad som observerats den senaste tiden.

### 6.1.5 Debugfönster

Det kommer finnas ett debugfönster där en detaljerad förfrågan kan specificeras bit för bit till ett visst delsystem. När fönstret stängs kommer kommunikationseenheten att skicka denna förfrågan över bussen och få ett svar, som returneras direkt i sin helhet till datorn och visas i loggfönstret.

### 6.1.6 Loggfönster

Loggfönstret kommer fungera som en systemomfattande statusmeddelare. Här kommer ett flöde av beslut och statusrapporter från roboten, bekräftelse av kommando och errormeddelande visas.

## 7 Kommunikation mellan delsystem

För att alla delssystem ska kunna kommunicera med varandra kommer en multimaster I<sup>2</sup>C buss att implementeras Alla delsystem kommer att sitta kopplade på denna och all kommunikation mellan olika delssystem måste gå via denna.

Fysiskt kommer bussen bestå av två kablar, en Serial Data Line (SDA) samt en Serial Clock Line (SCL).

### 7.1 Protokollbeskrivning

När bussen är ledig kommer alla system att vara master. När ett system vill ta kontroll över bussen skickar det en start-bit. Detta kommer tala om för alla andra system att bussen nu är upptagen och de antar då slave-mode.

Alla system kommer att ha en egen unik adress (SLA). Vilken adress olika system har finns i tabell 1. General call kommer finnas och alla system kommer då vara adresserade slavar.

Delsystem	Adress
Chassi	0000 001
Arm	0000 110
Sensor	0000 100
Kommunikation	0000 101

Tabell 1: Adresser över samtliga delsystem

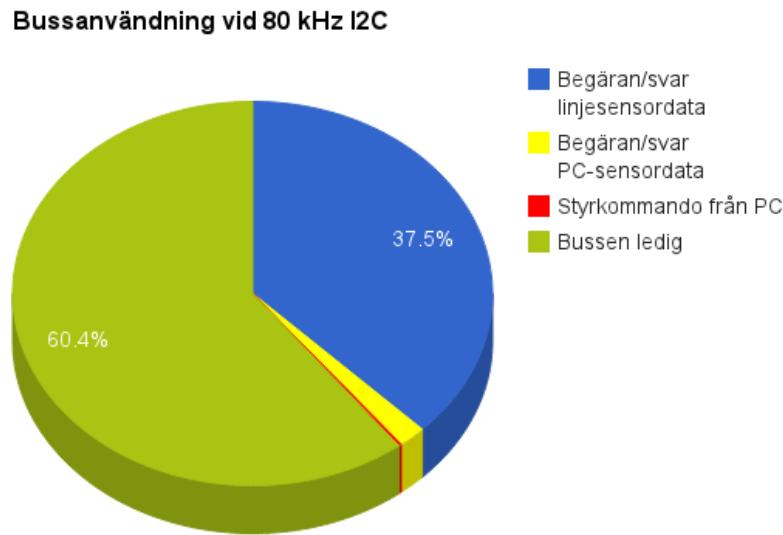
Varje dataöverföring kommer att vara två byte lång. Efter detta kommer alltid repeatedstart att vara tillåtet. Efter varje byte som överförs skickar system som tog emot acknowledgement (ACK) eller no-acknowledgement (NACK) tillbaka. NACK kommer endast att skickas då överföringen misslyckades eller i specialfallet då master är mottagare och den sista byten i överföringen har mottagits.

Vid fallet att två eller flera system försöker prata på bussen samtidigt kommer de system som förlorar arbitration att vänta på en stop-signal på bussen och sedan försöka igen.

Programflöden med beskrivningar om vilka register som ska ändras för att få önskad funktion finns i bilaga A.

### 7.2 Analys av prestanda

Figur 16 beskriver hur bussens tid fördelas på olika funktioner. Grafen har framställts genom att uppskatta hur lång tid överföringar av olika funktioner tar och hur många gånger de sker under en sekund.



Figur 16: Relativ tidsbelastning på bussen från olika ärenden. De flesta överföringar sker så sällan att de inte syns i diagrammet

### 7.3 Informationsflöde

Delsystem kommer både att kunna efterfråga data från andra delsystem över bussen, och skicka kommandon och information på eget initiativ. Viktigt att observera är att en förfrågan endast ska göras när datan redan finns redo och förberedd på delsystemet som får förfrågan, eftersom datan ska returneras omedelbart efter att förfrågan har kommit in. Om datan vid en förfrågan inte är redo eller inte finns ska en förutbestämd felkod returneras.

Data	Sändarenhet	Metod	Mottagarenhet
Tyngdpunkt och flaggor för tejpstatus	Sensor	Får förfrågan från	Chassi
Värde på RFID-tag	Sensor	Får förfrågan från	Chassi
Koordinater till plac-kobjekt	Sensor	Får förfrågan från	Arm
Avstånd från grip-klon till föremål	Sensor	Får förfrågan från	Arm
All sensordata	Sensor	Får förfrågan från	PC via Komm. e.
Styrkommandon	PC via Komm. e.	Skickar själv till	Arm och Chassi
Stoppkod	Komm. e. eller PC via Komm. e.	Skickar själv till	Alla via general call
Styrbeslut och händelser	Chassi	Skickar själv till	PC via Komm. e.
Kommando för upplockining	Chassi	Skickar själv till	Arm
Kommando för avlämning	Chassi	Skickar själv till	Arm
Rattutslag och gaspådrag	Chassi	Får förfrågan från	PC via Komm. e.
Statusmeddelanden	Arm	Skickar själv till	Chassi och PC via Komm. e.

Tabell 2: Informationsflöde mellan delsystem.

## 8 Implementeringsstrategi

För att kunna arbeta optimalt så kommer första prioritet för utveckling att vara att färdigställa den interna busskommunikationen fullständigt. Med hjälp av logikanalysatorer och andra standardkretsar som använder I<sup>2</sup>C kommer funktionaliteten att kunna testas i varje delsystem för sig till viss del, innan alla kopplas samman och testas.

Bland det första som ska ske i projektet är också att bygga ihop hårdvaran till virkorten och verifiera att den fungerar. För att kunna undersöka hur delsystem reagerar på förfrågningar över bussen finns funktionalitet i programvaran för att skicka förfrågningar till delsystem från PC:n och observera deras svar. Detta ska användas för att verifiera att delsystem fungerar som de ska. I så stor utsträckning som möjligt ska alla funktioner testas i samband med att de färdigställs.

## A Master transmitt

Pseudokod samt vilka register som ska ändras i vilka steg för att bussen ska få önskad funktion. Denna kommer bestå utav 4 olika program, enligt följande:

### A.1 Master transmitt

Programflödet för det program som krävs för master transmitt ska fungera De register som måste modifieras och hur de ska modifieras i de olika punkter finns i följande tabell.

1. För START ska TWINT samt TWSTA sättas till 1 i TWCR. Processorn kommer då att vänta på att bussen ska bli ledig och sedan skicka START på bussen.
2. Vänta på att TWINT ska gå hög.
3. Läs TWSR, kontrollera innehåll, koderna är som följer:
  - (a) 0x08 START lyckades.
  - (b) 0x10 REPEATED START lyckades.
4. Skriv SLA+W till TWDR.
5. Skriv TWINT till 1 i TWCR.
6. Vänta på att TWINT ska gå hög.
7. Läs TWSR, kontrollera innehåll, koderna är som följer:
  - (a) 0x18 SLA+W lyckades, ACK mottagen.
  - (b) 0x20 SLA+W lyckades, NACK mottagen.
  - (c) 0x38 arbitration förlorad.
8. Skicka data till TWDR. Beroende på värde i TWSR ska följa aktion tas:
  - (a) 0x18: TWINT = 1, TWSTA = TWSTO = 0. Skriver data på bussen.
  - (b) 0x20: TWINT = TWSTA = 1, TWSTO = 0. REPEATED START, tillbaka på punkt 2.
  - (c) 0x38: TWINT = TWSTA = 1, TWSTO = 0. START kommer sändas då bussen är ledig, tillbaka på punkt 2.
9. Vänta på att TWINT till 1 i TWCR.
10. Läs TWSR, kontrollera innehåll, koderna är som följer:
  - (a) 0x28 Data byte skickad, ACK mottagen.
  - (b) 0x30 Data byte skickad, NACK mottagen.
11. Beroende på kod samt om mer data ska skickas eller inte kan följande aktion tas:
  - (a) 0x28:
    - i. Mer data skickas i samma paket? Om ja, TWINT = 1, TWSTA = TWSTO = 0. Tillbaka på 8 a.
    - ii. TWSTA = TWINT = 1, TWSTO = 0. REPEATED START, om mer data ska skickas, tillbaka på punkt 1.
    - iii. TWINT = TWSTO = 1, TWSTA = 0. STOP skickas, överföring klar.
  - (b) 0x30:
    - i. TWINT = TWSTA = 1, TWSTO = 0. REPEATED START skickas, tillbaka på punkt 2.
    - ii. TWINT = TWSTO = 1, TWSTA = 0. STOP skickas, överföring klar.

## A.2 Master receiver mode

Punkt 1 till 6 samma som för master transmitt mode med skillnaden att SLA+R skrivs till TWDR istället för SLA+W. Sedan följer flödet:

5. Läs TWSR, kontrollera innehåll, koderna är som följande:
  - (a) 0x38 arbitration förlorad eller NACK mottagen.
  - (b) 0x40 SLA+R skickad, ACK mottagen.
  - (c) 0x48 SLA+R skickad, NACK mottagen.
6. Beroende på värdet i TWSR ska följande aktion tas:
  - (a) 0x38: TWINT = TWSTA = 1, TWSTO = 0. Vänta till bussen är ledig och skicka START igen, tillbaka på punkt 2.
  - (b) 0x40: Vänta på att TWINT ska bli hög, fortsätt på punkt 9.
  - (c) 0x48: TWINT = TWSTA = 1, TWSTO = 0. REPEATED START kommer att skickas. (Finns risk för oändlig loop om adressen inte finns)
7. Läs TWSR, kontrollera innehåll, koderna är som följer:
  - (a) 0x50 Data mottagen, ACK skickad.
  - (b) 0x58 Data mottagen, NACK skickad.
8. Läs data i TWDR.
9. Beroende på värdet i TWSR ska följande aktion tas:
  - (a) 0x50: Ta emot en byte till? Om ja ska TWINT = TWEA = 1, TWSTA = TWSTO = 0. ACK skickas. Om nej ska TWINT = 1, TWEA = TWSTA = TWSTO = 0. NACK skickas.
  - (b) 0x58: Starta en ny överföring? Om ja, sätt TWINT = TWSTA = 1, TWSTO = 0. REPETADE START kommer skickas. Om nej, TWINT = TWSTO = 1, TWSTA = 0.

## A.3 Slave receiver mode

För att en slav ska kunna ta emot data behövs följande programflöde.

1. Vänta på TWINT FLAG går hög (hanteras via avbrott)
2. Kontrollera TWSR, möjliga koder är:
  - (a) 0x60 Egen SLA+W har tagits emot, ACK skickats.
  - (b) 0x68 Förlorat arbitration som master, egen SLA+W/R har mottagits. ACK skickats.
  - (c) 0x70 general call adress mottagen, ACK skickad.
  - (d) 0x78 Arbitration förlorad som master. General call mottaget, ACK returnerad.
  - (e) 0x80 Data har mottagits, ACK returnerad.
  - (f) 0x88 Data har mottagits, NACK returnerad.
  - (g) 0x90 Data har mottagits efter general call, ACK returnerad.
  - (h) 0x98 Data har mottagits efter general call, NACK returnerad.'
  - (i) 0xA0 STOP eller REPETATED START har mottagits medans fortfarande adresserad som slav.

3. Beroende på värde i TWSR ska följande aktion tas:
  - (a) 0x60: TWINT = TWAE = 1, TWSTO = 0. Data tas emot, ACK returneras.
  - (b) 0x68: Samma som för 0x60.
  - (c) 0x70: Samma som för 0x60.
  - (d) 0x80: Samma som för 0x60.
  - (e) 0x88: Läs från TWDR (kan vara koorupt?) TWINT = TWEA = 1, TWSTO = 0. Ändra inte TWSTA, detta görs i en annan rutin om vi vill skicka på bussen när den blir ledig.
  - (f) 0x90: Läs från TWDR. TWINT = TWEA = 1, TWSTO = 0. Data kommer tas emot, ACK returneras.
  - (g) 0x98: Samma som 0x88.
  - (h) 0xA0: Samma som 0x88.

## A.4 Slave transmitter mode

För att en slav ska kunna skicka data behövs följande programflöde.

1. Vänta på att TWINT FLAG går hög. (hanteras via avbrott)
2. Kontrollera TWSR, möjliga koder är:
  - (a) 0xA8 Egen SLA+R har tagits emot, ACK skickats.
  - (b) 0xB0 Arbitration förlorad i SLA+R/W som master, egen SLA+R mottagen, ACK skickad.
  - (c) 0xC0 Data i TWDR skickas, NACK mottagen.
  - (d) 0xC8 Sista byte i TWDR skickad, ACK mottagen.
3. Beroende på värde i TWSR ska följande aktion tas:
  - (a) 0xA8: Ladda data till TWDR. TWINT = TWEA = 1, TWSTO = 0. Data skickas.
  - (b) 0xB0 Samma som 0xA8.
  - (c) 0xB8 Ladda data till TWDR. TWINT = 1, TWSTO = TWEA = 0. Sista data byte skickas.
  - (d) 0xC0 TWINT = TWEA = 1, TWSTO = 0. Går tillbaka till grundläge. Ändra inte TWSTA, detta görs i en annan rutin om vi vill skicka på bussen när det blir ledig.
  - (e) 0xC8 Samma som för 0xC0.

## B Bluetooth initiering och konfigurering

För att inleda kommunikation så behövs följande göras på BT enheten och på PC:

1. Följa uppkopplingsguide till BT-enheten på följande websida:  
<https://docs.isy.liu.se/twiki/bin/view/VanHeden/BlueTooth>
2. Skicka '\$\$\$' till BT när enheten är Idle ('stat'-lampa blinkar 1 gång per sek).
3. Konfigurera efter preferens (skicka SF,1 för att gå tillbaka till default).
4. Skicka '—' för att gå ur kommandoläge
5. Nu är det bara att skicka/läsa den data du behagar till/från den port som enheten är kopplad till.

Några bra konfigurationskommandon (ändras endast med välmotiverade och genomtänkta anledningar):

- SF,1 ;Gör att enheten går tillbaka till defaultinställningar
- SN ;Sätter namnet på BT-enheten
- SM ;Sätter mode på BT
- SP ;Sätter pinkod (default är 1234)
- SU ;Sätter baud rate (default 115k)

För att processorn ska kunna läsa och skriva data till BT-enhet:

1. Koppla BT med processorn enligt kopplingschema.
2. Initiera USART genom att konfigurera alla register enligt:
  - (a) UCSRnB
    - i. RXENn (bit4) = 0x1, slår på Recieve
    - ii. TXENn (bit3) = 0x1, slår på Transmitt
    - iii. RXCIEn (bit7) = 0x1, gör att det sker ett avbrott när processorn fått en recieve.
  - (b) USCRnC
    - i. UMSEL = 0x0 asynkront läge (ingen extern klocka som styr)
    - ii. UPMn = 0x00, ingen paritet
    - iii. USBSn = 0x0, vi kör med 1 stopbit
  - (c) Baudrate-registrerna konfigureras med formeln,  $BaudValue = \frac{F_{CPU}}{16USARTBAUDRATE - 1}$  enligt nuvarande konfigurationer sätts:
    - i. UBRRH = 0x00
    - ii. UBRL = 0x09
3. Skicka görs genom att skriva data på UDRn-registret (8 bitar i taget) så görs resten automatiskt. Läsning görs genom att läsa från UDRn-registret (8 bitar), detta då en receive mottagits och inladdning i registret är klart, enligt konfiguration så sker ett avbrott när detta händer.
4. Se flödesschema för hantering av dataflöde.