DESIGNSPECIFIKATION

Tim Fornell

Version 1.0

Status

Granskad	Oscar Frylemo	2016-03-08
Godkänd	Handledare: Anders Nilsson	2016-03-11

PROJEKTIDENTITET

Grupp 8, VT16
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Tim Fornell	Projektledare (PL)	072-394 90 05	timfo734@student.liu.se
Oscar Frylemo	Dokumentansvarig (DOK)	076-209 70 77	oscfr634@student.liu.se
Martin Hinnerson	Designansvarig hårdvara (DHÅ)	072-210 43 72	marhi386@student.liu.se
Jacob Holmberg	Testansvarig (TA)	070-391 50 33	jacho926@student.liu.se
Marcus Homelius	Designansvarig sensorenhet (DS)	070-245 90 96	marho949@student.liu.se
John Stynsberg	Datorenhetsansvarig (DE)	072-057 21 54	johst529student.liu.se

Kund: Tomas Svensson, 581 83 LINKÖPING,
kundtelefon: 013-28 13 68, tomass@isy.liu.se

Kursansvarig: Tomas Svensson, 581 83 LINKÖPING,
telefon: 013-28 13 68, tomass@isy.liu.se

Handledare: Peter Johansson, 581 83 LINKÖPING
telefon: 013-28 13 45, Peter.A.Johansson@liu.se

Handledare: Anders Nilsson, 581 83 LINKÖPING
telefon: 013-28 26 35, Anders.P.Nilsson@liu.se

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2016-03-08	Första utkastet	Alla	OF
1.0	2016-03-11	Första versionen	MH	OF

Innehållsförteckning

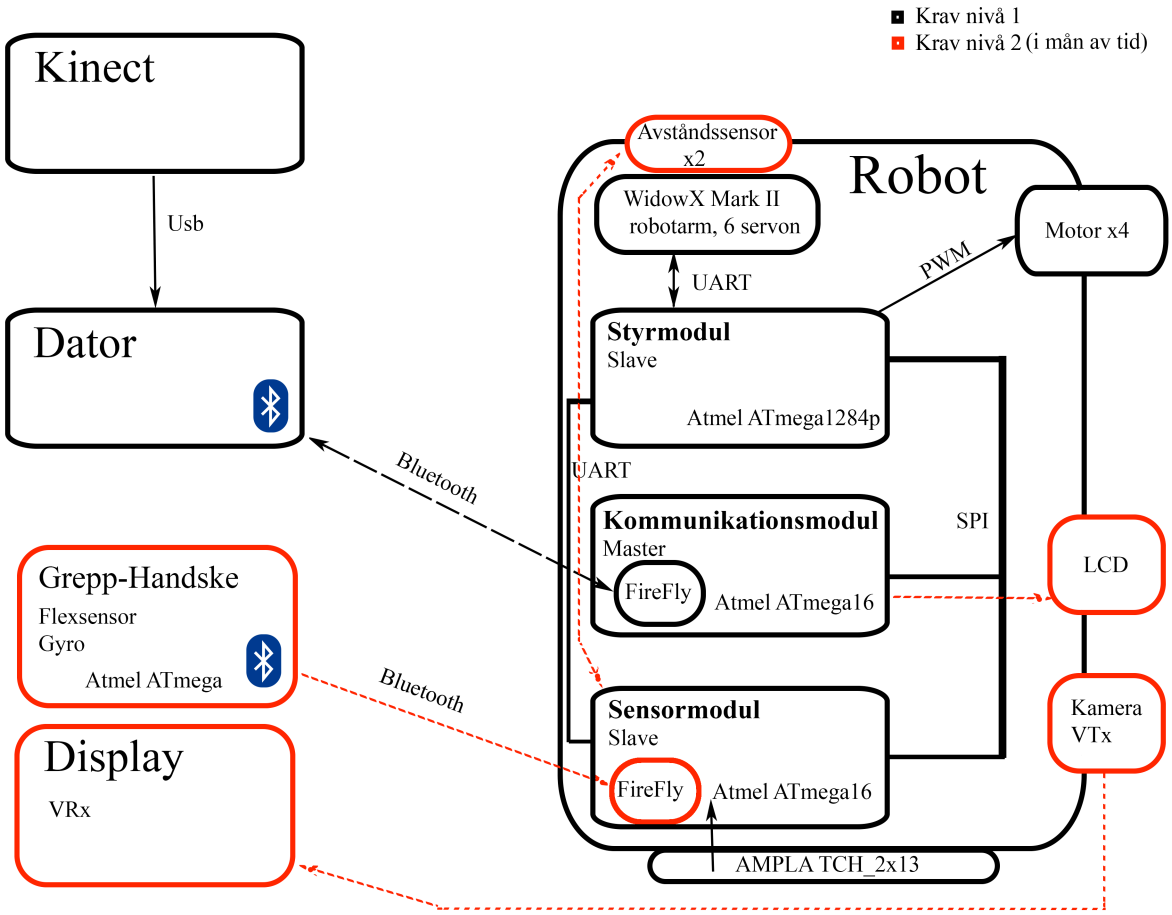
1 Sammanfattning

Detta dokument beskriver hur och vad som behövs för att konstruera en robotplattform med Kinect-styrd arm. Roboten ska kunna arbeta både manuellt och autonomt i en miljö som är skadlig för människor att vistas i. Autonom manövrering sker genom att roboten med hjälp av en linjesensor följer en på förhand markerad väg. När målet, exempelvis ett laboratorieprov, är nått ska robotarmen som styrs manuellt via en Kinect-sensor användas för att utföra önskad uppgift.

Krav **2** kommer inte behandlas i den här versionen av designspecifikationen. Dessa krav behandlas om de blir aktuella i projektet.

2 Systemöversikt

Systemet är uppdelat i tre delsystem; datorenheten, Kinect-enheten och robotenheten. Kinect-enheten och datorenheten är ihopkopplade via USB. Kommunikation mellan datorenheten och robotenheten sker trådlöst via Bluetooth. Systemet illustreras i *figur 1,* där svart indikerar att det är ett krav **1** och rött krav **2** (görs i mån av tid).



Figur 1, en överblick av systemet.

Kinect-enheten har som uppgift att läsa av koordinater för användarens arm och sedan skicka dessa till datorenheten.

Datorenheten används för att skicka kommandon och parametrar, samt att ta emot data (t.ex. styrbeslut och sensorvärden) från robotenheten. Den ska ha ett användargränssnitt där dessa värden illustreras. Datorenheten ska även behandla data från Kinect-enheten för att kunna vidarebefordra detta till styrmodulen vid styrning av robotarmen.

Robotenheten är uppdelad i tre delsystem; kommunikationsmodulen, sensormodulen och styrmodulen. Kommunikationsmodulen har i uppgift att distribuera information till datorenheten, styrmodulen och sensormodulen.

Sensormodulen ansvarar för insamling av data från omgivningen, inledningsvis så görs det med hjälp av en reflexsensormodul. Reflexsensormodulen är placerad i mitten av robotplattformen och består utav 7 reflexsensorer.

Styrmodulen har som uppgift att kontrollera robotenhetens servon, som är lokaliserade i robotarmen och hjulen. Beroende på om det är armen eller hjulen som styrs så kommer styrmodulen att förses kontinuerligt med data antingen från sensormodulen eller datorenheten via kommunikationsmodulen. Data som tas emot ska kunna tolkas och användas för styrbeslut.

2.1 Ingående delsystem

De delsystem som systemet är uppbyggt av är:

- Kinect-enheten, *se kapitel 3*.
- Datorenheten, *se kapitel 4*.
- Robotenheten, *se kapitel 5*.
    - Styrmodul
    - Kommunikationsmodul
    - Sensormodul

3 Kinect-enhet

Kinect-enheten, är en avancerad sensor innehållande kameror, mikrofoner och en accelerometer. Den innehåller också mjukvara som kan bearbeta skelett-data, färg och djup. Kinect innehåller följande hårdvarukomponenter:

• En RGB-kamera med möjlighet att ta bilder i en upplösning på högst 1280x960.

• En IR sändare samt IR djupsensor för att kunna få ett djup i bilderna av rummet.

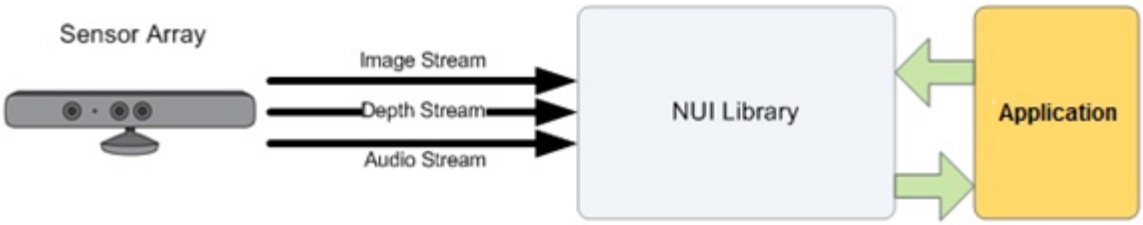
• Fyra stycken mikrofoner. För att bland annat kunna avgöra var ljudkällan i rummet ligger.

• En accelerometer för att kunna avgöra Kinect-enhetens läge.

I projektet kommer Kinect-enheten användas för att få fram koordinater i ett rum för armens leder (axelleden, armbågsleden, handleden och handen), därför behöver endast Kinect-enhetens skelett-data användas. Koordinaterna anges i ett kartesiskt koordinatsystem där Kinect-enhetens position utgör origo. Z-axeln anger avståndet bort från enheten till användaren.

3.1 Kommunikation med andra system

Utvecklings miljön (The SDK, Software Development Kit) för Kinect är uppbyggt enligt *figur 2*. Där sensorn skickar information om bild, djup och ljud till NUI:n (Natural User Interface) som sedan kan användas av applikationer. Datorn måste använda operativsystemet Windows.



Figur 2, översiktlig bild av Kinectsensorn.

Koordinaterna skickas från Kinect-enheten till användargränssnittet i datorn som skickar det vidare till kommunikationsmodulen som skickar det vidare till styrmodulen.

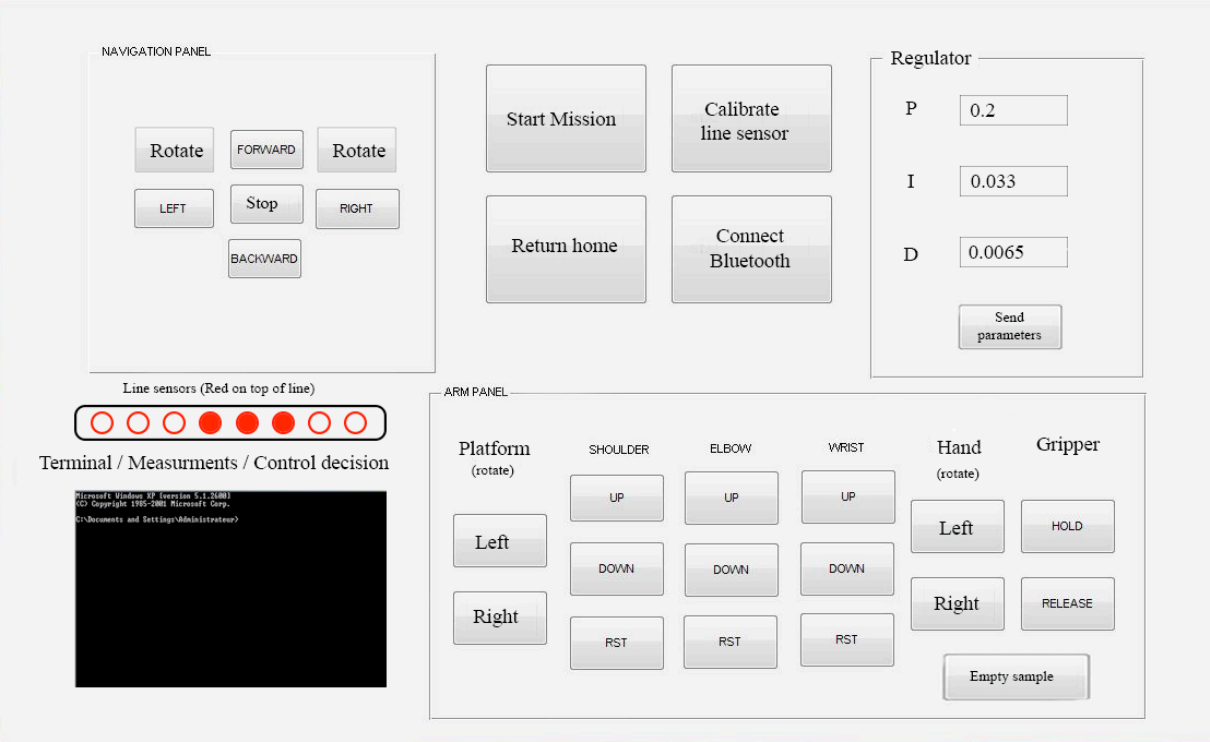
4 Datorenhet

Datorenheten har ett användargränssnitt som kopplar samman Kinect-enheten (via USB) och robotenheten. Datorn kommer att använda operativsystemet Windows. Via användargränssnittet kan användaren skicka kommandon till roboten, styrningen kan ske autonomt eller manuellt.

Vid autonom styrning skickas ett kommando från datorn som säger till roboten att följa en markerad bana (enligt banspecifikation) fram till ett markerat slut.

4.1 Användargränssnitt

Det användargränssnittet som kommer att användas ska utvecklas under projektet och via denna del kan användaren koppla upp sig mot robotplattformen för att skicka och ta emot data, se sensorvärden och kunna kalibrera linjesensorn. Det är även detta program som har i uppgift att tolka data och sensorvärden ifrån Kinect-enheten. Det kommer att utvecklas med programmeringsspråket C++. Användargränssnittet ska se ut enligt följande, *se figur3*.



Figur 3, en översikt av det grafiska gränssnittet.

Roboten ska kunna manövreras i två lägen, autonomt och manuellt. Vid manuell styrning så används ett tangentbord. Vid autonom styrning så följer roboten en markerad bana *(se banspecifikation)* och manövrerar så att den inte lämnar linjen. Autonom styrning initieras via ett kommando i användargränssnittet. I både manuell och autonom styrning så kontrolleras robotarmen med Kinect-enheten samt ett tangentbord.

Vid autonom och manuell styrning av roboten ska användargränssnittet visa sensordata och styrbeslut från roboten. Sensordata representeras i form av lampor där tänd lampa motsvarar att en sensor detekterat tejpen och styrbesluten visas i terminalen. Det ska vara möjligt att kalibrera linjesensorerna samt ändra parametrar för reglering via användargränssnittet.

4.2 Kommunikation

Datorenheten kommunicerar med robotplattformen via Bluetooth och med Kinect-enheten via USB. Vid kommunikation med roboten skickas data i båda riktningarna. Datorenheten skickar kommandon när användaren avser göra detta, roboten skickar kontinuerligt data i form av sensorvärden och styrbeslut till datorn. Mellan datorenheten och Kinect-enheten skickas data endast i en riktning; från Kinect-enheten. Data skickas då Kinect-enheten har identifierat ett objekt som den kan följa med hjälp av sina sensorer. Eftersom FireFly följer standard protokollet 115200 bps, 8-N-1 så kommer den kommunikationen kommer inte behöva begränsas något eftersom datastorleken kommer vara väldigt liten i jämförelse.

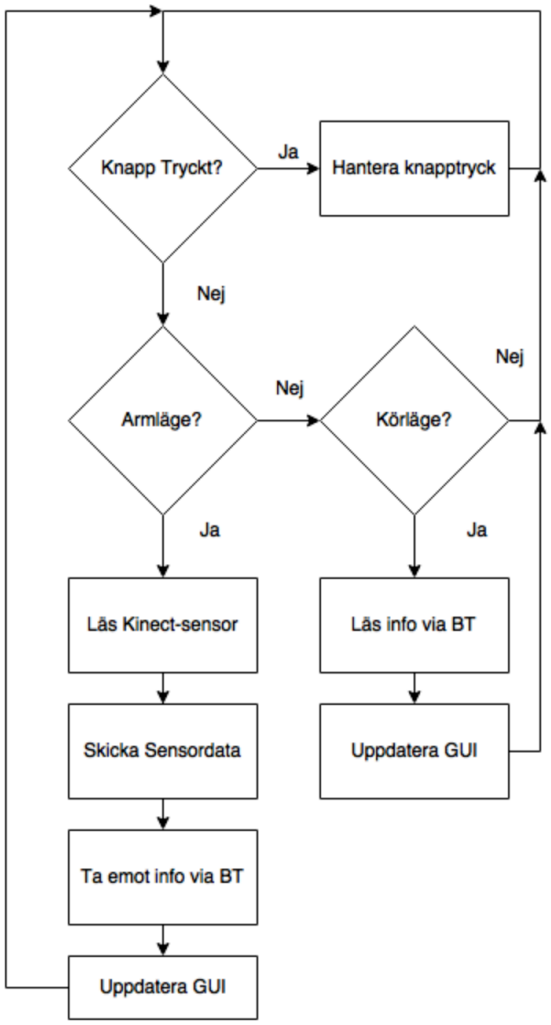
4.3 Implementation

Gränssnittet som ska utvecklas för detta projekt kommer bygga vidare på Johan Hydéns [1] program för att uppnå den funktionalitet som krävs i detta projekt. Funktionerna som denna programvara behöver är följande.

- SendCoordinates – Måste modifieras så att vi delar upp konverteringen och lägger den i en egen funktion så att det är valbart ifall vi skickar vinklar eller koordinater till roboten.
- SendMove – Skickar ett styrkommando till roboten.
- SendGripperCommand – Skickar kommando till klon.
- SendSeries – Skickar en serie av koordinater.
- SendConstants – Skickar regleringskonstanter.
- StartMission – Skickar kommandot ”Starta autonom styrning”.
- GetInfo – Hämta informationen om roboten från COM-porten
- DisplayInfo – Skriv ut styrbesluten och de andra värdena.

Förutom dessa så finns det redan ett antal funktioner i Hydéns program och beroende på om de behövs så kommer dessa att utnyttjas. [1]

Flödesschema för datorenheten är enligt följande, *se figur 4*, på nästa sida.



Figur 4, flödesschema för datorenheten.

4.4 Tester

På datorsidan så kan man testa all funktionalitet kontinuerlig under utvecklingen av programvaran. Varje enskild funktion kan övervakas genom felsökning.

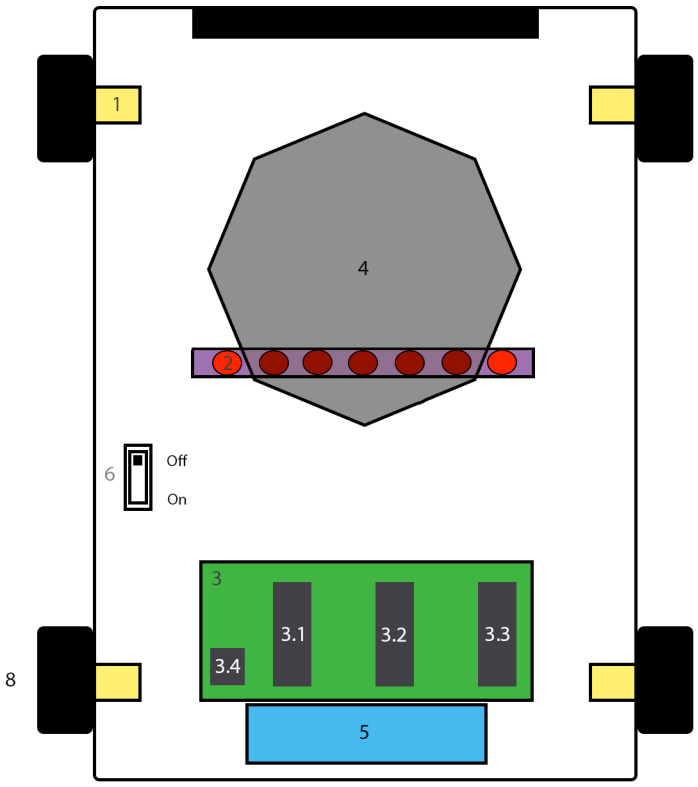
Kommunikationen mellan kommunikationsmodulen bör testas stegvis.

- Test utav koppling mellan datorn och Firefly-enheten. Kontrollera att enkla signaler kan skickas och observeras på båda sidor.
- Test om båda enheterna kan analysera vilken sorts signaler respektive enhet tar emot.

5 Robotenhet

Robotenheten kommer bestå av 3 moduler; styrmodul, kommunikationsmodul och sensormodul. På roboten sitter en robotarm av typen WidowX Mark II, 4 stycken motorer samt reflexsensormodul. Roboten har utvecklingsmöjligheter för LCD skärm, extern kamera och två avståndssensorer på robotarmens klor. Kommunikation mellan robotenheten och datorenheten sker via Bluetooth. Robotplattformen kommer se ut enligt *figur 5*.

Översikt för robotplattform
1 - Hjulservon
2 - Reflexsensor
3 - Virkort
  3.1 - Styrmodul CPU
  3.2 - Kommunikationsmodul CPU
  3.3 - Sensormodul CPU
  3.4 - Bluetooth-modul
4 - Robotarm fäste
5 - Batteri
6 - Av och på-knapp
7 - Bluetooth-modul
8 - Hjul



Figur 5, översikt av robotenheten.

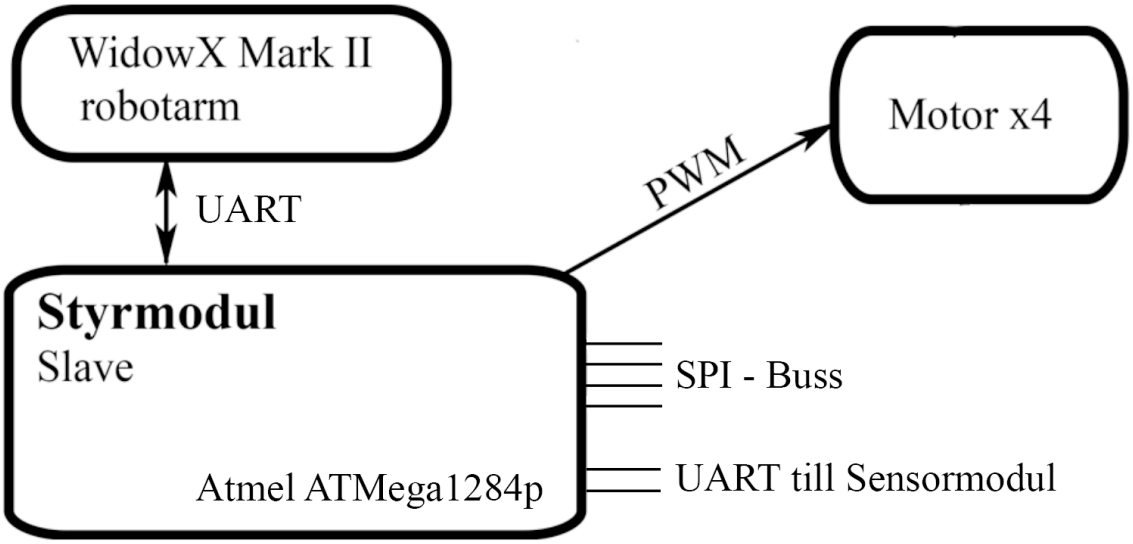
5.1 Styrmodul

Styrmodulen har i uppgift att kontrollera alla robotens rörelser, vilket innebär att den kontrollerar all rörelse av robotarmen samt all rörelse av hjulen.

För att styra robotarmen så ska styrmodulen ta emot data från Kinect-enheten i form av en koordinat dit armen önskas flyttas. Efter detta ska styrmodulen räkna ut med hjälp av invers kinematik hur respektive servo ska ställas in för att armen ska nå denna nya position. Utifrån detta ska den sedan beräkna en rörelseväg för att ta sig till den nya positionen som motsvarar den lämpligaste vägen utifrån robotarmens begränsningar.

För att styra robotplattformens hjul så ska styrmodulen ta emot data från sensormodulen i form av värden från linjesensorn. Utifrån dessa värden så ska styrmodulen fatta beslut om vart robotplattformen befinner sig i förhållande till linjen. För att sedan utifrån detta beräkna hur hjulen ska regleras för att plattformen ska fortsätta följa linjen på önskat vis.

Styrmodulen kommer vara uppbyggd enligt *figur 6*.



Figur 6, en översikt av styrmodulen.

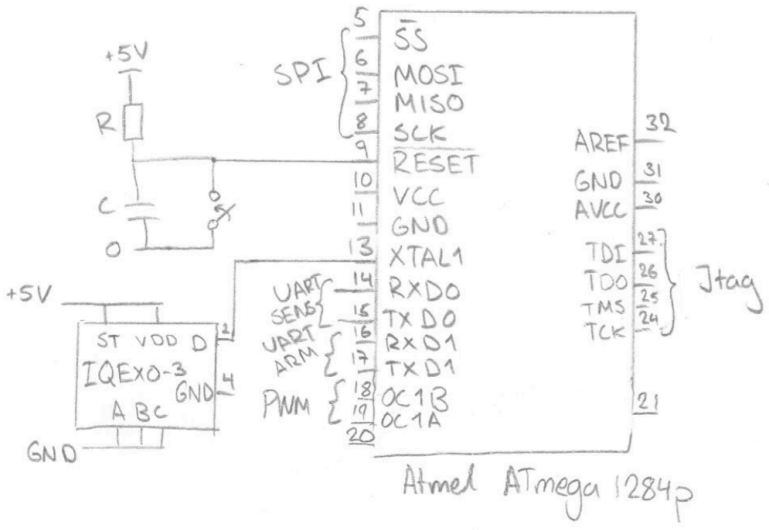
5.1.1 Komponentbudget

Styrmodulen kommer bestå av följande komponenter:

- Atmel ATmega1284p mikrokontroller
- 4 st. PWM-styrda elmotorer
- WidowX Mark II robotarm av Trossen Robotics bestående av 2 st. MX-64, 2 st. MX-28 och 2 st. AX-12A Dynamixel Servon
- IQEXO-3 klockpulsgenerator

5.1.2 Kopplingsschema

I *figur 7* finns ett kopplingsschema för styrmodulen.



Figur 7, kopplingsschema för styrmodulen.

- PD0-1 (RXD0, TXD0), dvs *UART* kopplas till sensormodulen
- PD2-3 (RXD1, TXD1), dvs *UART* kopplas till robotarmen
- PD4-5 (OC1B, OC1A) kopplas till motorernas PWM
- PD6-7 är motorernas riktningssignaler
- PB4-7 (SS, MOSI, MISO, SCK) kopplas till bussen.
- PC2-5 (TDI, TDO, TMS, TCK) kopplas till JTAG-kedjan.
- XTAL1 kopplas till en klockpulsgenerator.
- RESET ska anslutas för att underlätta felsökning.

5.1.3 Mjukvara

Mjukvaran kan delas upp efter styrmodulens två uppgifter, att styra hjulen så att linjen följs och att styra robotarmen. Alla styrbeslut ska också skickas via bussen till kommunikationsmodulen för att kunna vidarebefordras till datorenheten.

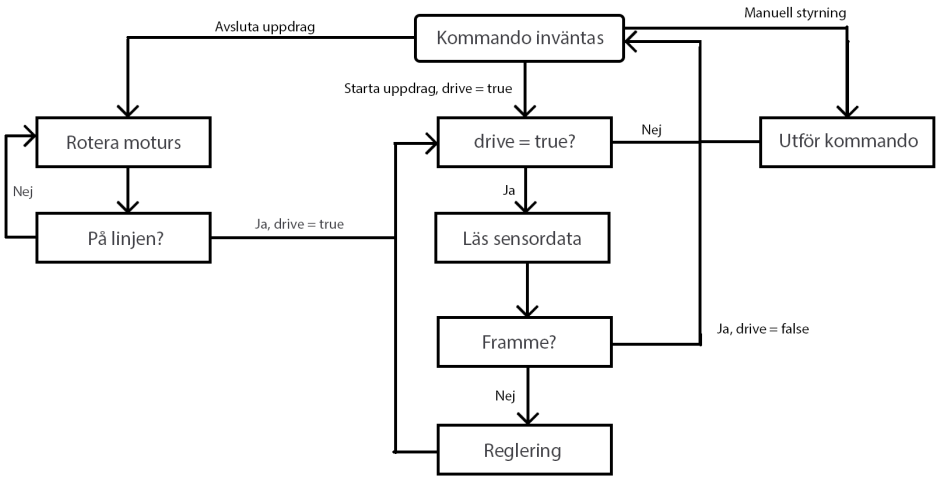
5.1.3.1 Styrning av hjul

Styrning av hjulen kan ske på två sätt; autonomt och manuellt. Autonom styrning initieras genom ett kommando i form av ”starta uppdrag” alt. ”avsluta uppdrag” och manuell styrning initieras genom enstaka kommandon som t.ex. ”kör framåt” alt. ”rotera vänster”.

Vid styrning av hjulen kommer styrmodulen att arbeta enligt flödesschemat i *figur 8*. Styrmodulen kommer att vänta på ett kommando skickas. Om kommandot som fås innebär manuell styrning så ska kommandot tolkas och sedan utföras enligt, på förhand, definierade algoritmer.

Om däremot kommandot som fås är ”starta uppdrag” eller ”avsluta uppdrag” så ska den autonoma styrningen startas. Beroende på vilket av dessa kommandon som fås ska robotplattformen agera olika. Vid ”avsluta uppdrag” ska robotplattformen först rotera ett visst antal grader moturs, detta upprepas till dess att linjen har hittats. Efter detta kommer kommandona ”avsluta uppdrag” och ”starta uppdrag” ha samma flödesschema.

Denna del av flödesschemat ska se till att robotplattformen följer linjen på marken. Detta görs med hjälp av PD-reglering. Styrmodulen kommer först kolla om den har ”klartecken” att köra genom att kontrollera variabeln *drive*. Om den har klartecken att köra så hämtar den data från sensormodulen i form av sensorvärden, sensorvärdena kommer tala om för styrmodulen vart robotplattformen är i förhållande till linjen. Utifrån dessa sensorvärden ska styrmodulen avgöra om robotplattformen är framme vid en start/slut-markering eller om hjulen ska regleras och i sådant fall hur de ska regleras. Om kommandot ”Stopp” fås under körning ska *drive* sättas till ”false” och körningen kommer då att avbrytas.



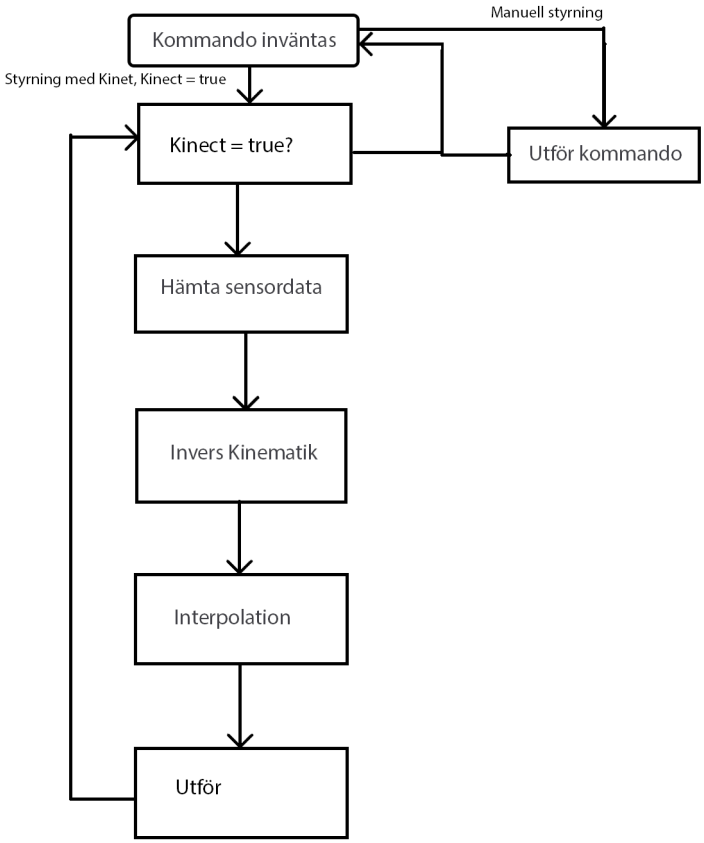
Figur 8, flödesschema för styrning av hjul.

5.1.3.2 Styrning av arm

Styrningen av armen kan ske på två sätt; manuellt via kommandon eller via Kinect. I användargränssnittet är det möjligt att välja i vilket läge robotarmen ska styras.

Om armen önskas styras manuellt via kommandon så kommer styrmodulen att vänta på att ett kommando skickas. Detta kommando kan vara t.ex. ”rotera klo” eller ”vrid arm moturs”, styrmodulen kommer då utföra detta kommando enligt, på förhand, bestämda algoritmer.

Om armen däremot önskas styras via Kinect så kommer styrmodulen arbeta enligt flödesschemat i *figur 9*. Styrmodulen kommer först kontrollera variabeln *Kinect*. Efter detta kommer styrmodulen hämta sensordata från Kinect-enheten (via kommunikationsmodulen) i form av en slutposition för robotarmen. Med denna data ska styrmodulen med hjälp av invers kinematik räkna ut hur servona ska ställas in vid slutpositionen. När detta är gjort ska styrmodulen räkna ut hur den ska flytta armen för att komma till denna slutposition, så att inga onödiga förflyttningar av armen görs. Detta upprepas kontinuerligt till dess att användaren avslutar Kinect-läget för att kontrollera armen.



Figur 9, flödesschema för styrning av robotarm.

5.1.4 Bedömning av prestanda

Då styrmodulen kommer utföra avancerade och tidskrävande beräkningar, i form utav invers kinematik och PD-reglering, så är det svårt att bedöma om ATMega 1284p har tillräcklig prestanda för att klara av beräkningarna. Men då den är kapabel till att utföra 20 MIPS (miljoner instruktioner per sekund) vid en klockfrekvens på 20 MHz och har ett minne på 128 kBytes så bedömer vi att den har tillräcklig prestanda för uppgiften.

5.2 Kommunikationsmodulen

Kommunikationsmodulens huvudsakliga uppgift är att behandla all kommunikation till och från roboten. Kommunikationsmodulen är uppbyggd enligt *figur 10.* Data kommer att skickas till och från kommunikationsmodulen mellan de andra modulerna med ett SPI-protokoll, medans kontakt mellan kommunikationsmodul och dator sker via Bluetooth. Ett krav **2** för kommunikationsmodulen är att den ska kunna skicka sensorvärden till en LCD skärm som finns på robotplattformen.

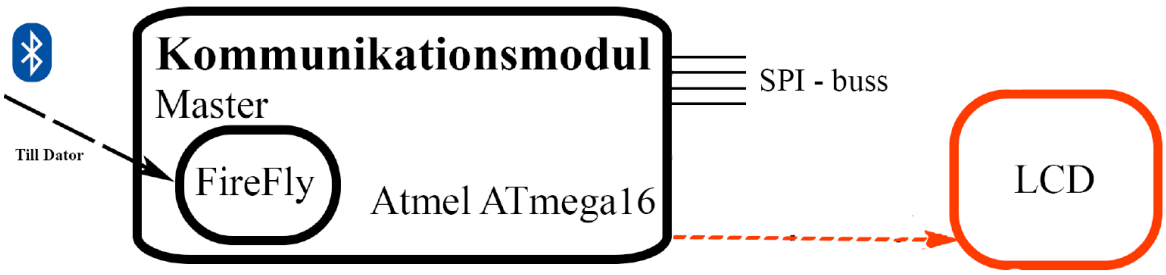
5.2.1 SPI

SPI är ett Master-Slave protokoll, kommunikationsmodul kommer vara masterenhet medans sensor- och styrmodul ska vara slave-enheter. Masterenheten bestämmer vilken modul den vill kommunicera med, detta resulterar i att kommunikationsmodulen blir central i systemet.

När masterenheten vill sända eller ta emot data från en slavenhet sätts tillhörande slave-select till logiskt 0. Sedan placerar masterenheten de bitar som ska skickas i sitt skiftregister, samma procedur sker för slavenheten. En klockpuls genereras från masterenheten vilket resulterar i att bitarna i masterenhetens skiftregister skickas via *Master Output Slave Input* (MOSI) till slavenhetens skiftregister. Bitarna i slavenhetens skiftregister skickas på motsvarande sätt via *Master Input Slave Output* (MISO) till masterenhetens skiftregister. När en överföring är färdig kommer innehållet i masterenhet och slavenhet ha bytt plats med varandra.

5.2.2 Bluetooth

Kommunikationsmodulen kommer kommunicera med datorn vi Bluetooth. Detta möjliggörs via en FireFly-modul, som parkopplas med datorn och är kopplad till kommunikationsmodulens UART, *se figur 11*.



Figur 10, översikt av kommunikationsmodulen.

5.2.3 Komponentbudget

Följande komponenter behövs för kommunikationsmodulen:

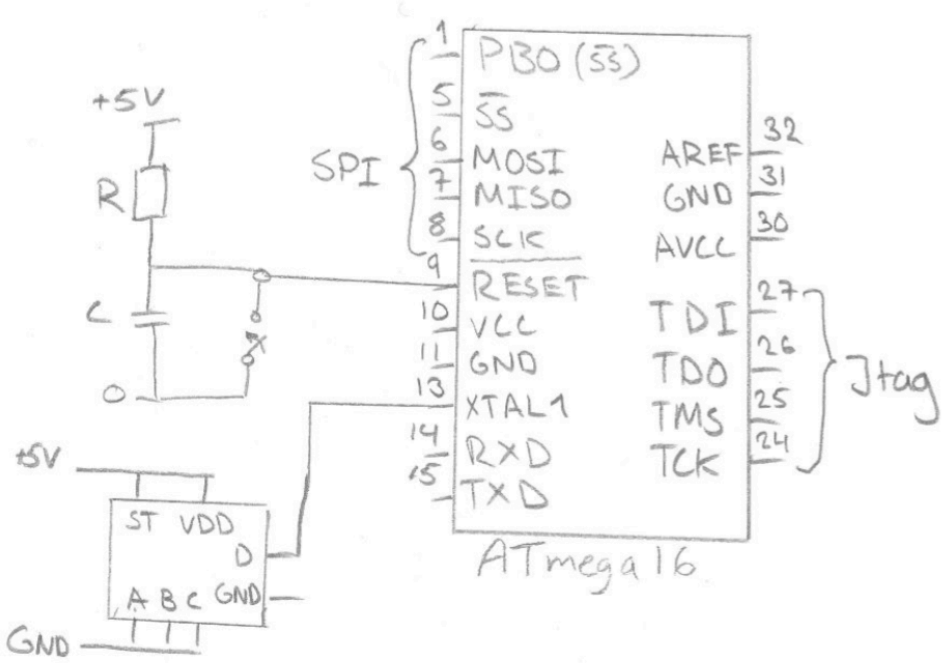
- Atmel ATmega16 mikrokontroller
- FireFly Bluetoothmodul
- IQEXO-3 Kristalloscillator

Krav 2:

- G121C LCD-skärm

5.2.4 Kopplingsschema

Nedan finns ett kopplingsschema för kommunikationsmodulen, *se figur 11*.

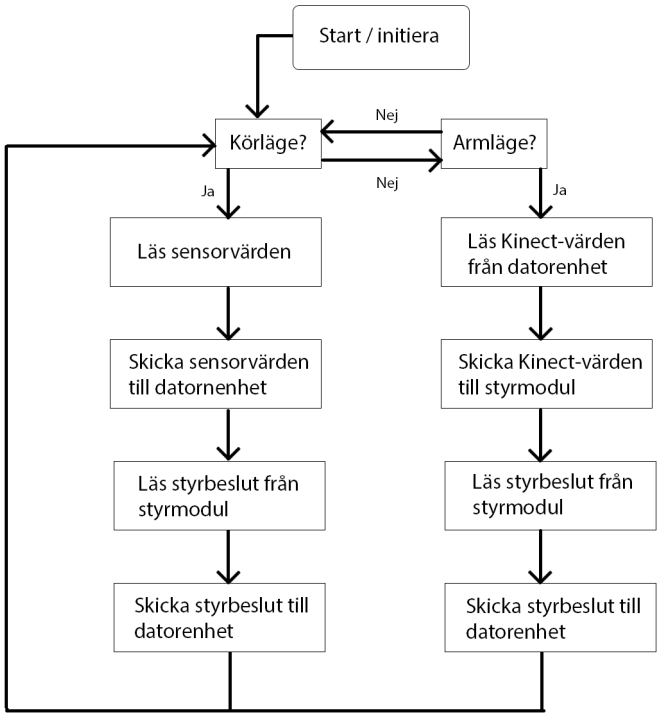


Figur 11, kopplingsschema för kommunikationsmodulen.

- PB4-7 (SS, MOSI, MISO, SCK) kopplas till bussen.
- RXD och TXD kopplas via UART till FireFly-modulen.
- PC2-5 (TDI, TDO, TMS, TCK) kopplas till JTAG-kedjan.
- XTAL1 kopplas till en klockpulsgenerator.
- RESET ska anslutas för att underlätta felsökning.

5.2.5 Mjukvara

Denna modul kommer kontinuerligt med hjälp av huvudloopen och avbrott hantera kommunikation med datorn och övriga moduler. Kommandon från dator ska skickas vidare till respektive modul som t.ex. ”Kalibrera Sensor” eller ”Starta Uppdrag”. Sensorvärden från sensormodulen och styrbeslut från styrmodulen ska skickas vidare till datorenheten, för flödesschema se *figur 12*. Kommandon från datorenheten kommer att generera avbrott och avbrottsrutinen kommer föra vidare kommandot till rätt enhet.



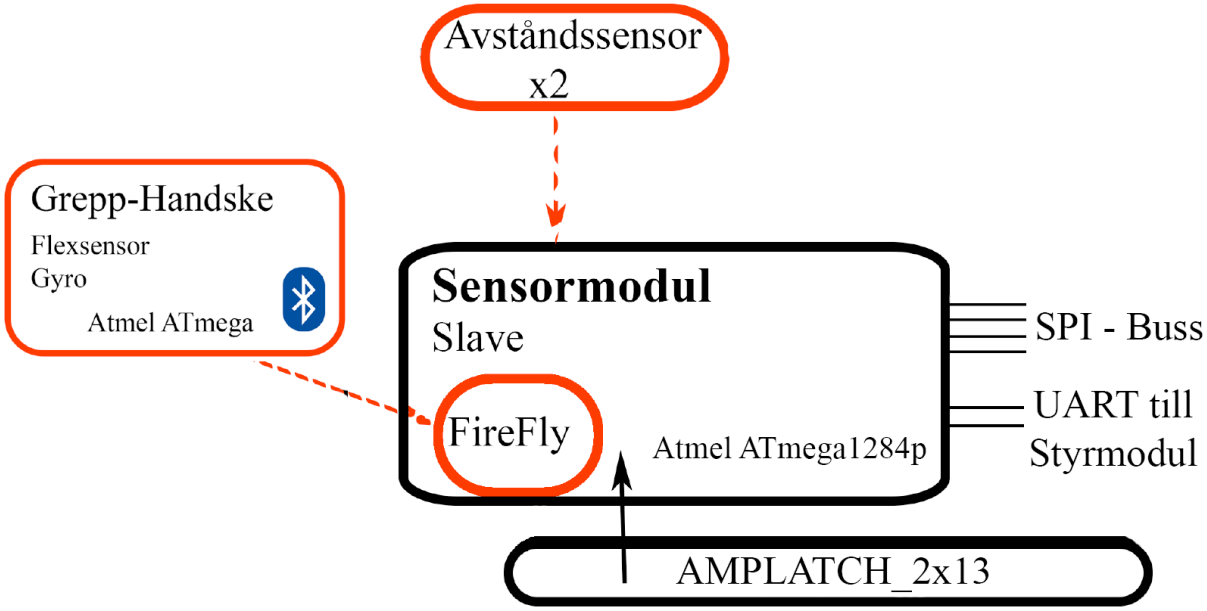
Figur 12, flödesschema för huvudloopen i kommunikationsmodulen.

5.2.6 Bedömning av prestanda

Inga tyngre beräkningar kommer att behöva utföras av kommunikationsmodulen. Dess uppgift är att ta emot och skicka data till och från datorenheten och fördela kommandon till rätt moduler i robotenheten. Det som behövs för detta är tillgång till SPI och UART vilket ATMega16 har. Vi gör bedömningen att mikrokontrollern har tillräckligt med prestanda för att klara av uppgiften.

5.3 Sensormodulen

Sensormodulens uppgift är att att ta emot och behandla sensordata från robotens sensorer. Att behandla sensordata innebär att översätta spänningar till mätdata som kan användas av andra moduler på roboten. Sensormodulen kommer vara uppbyggd enligt *figur 13*.



Figur 13, översikt av sensormodulen.

5.3.1 Kommunikation

Sensormodulen kommer att vara kopplad till kommunikationsmodulen via SPI-bussen. Där kommer sensordata skickas för att sedan skickas vidare till datorn. Sensormodulen kommer även att ta emot kommandon från datorn via denna buss som exempelvis kalibrering av sensorn.
Utöver detta ska även en direktbuss till styrmodulen finnas för att skicka sensordata som ska användas för att följa linjen. Förklaring av detta finns under punkt 6.2.

5.3.2 Komponentbudget

Följande komponenter behövs för sensormodulen:

- Atmel ATmega1284p mikrokontroller
- AMPLATCH_2x13
- IQEXO-3 klockpulsgenerator

Krav 2:

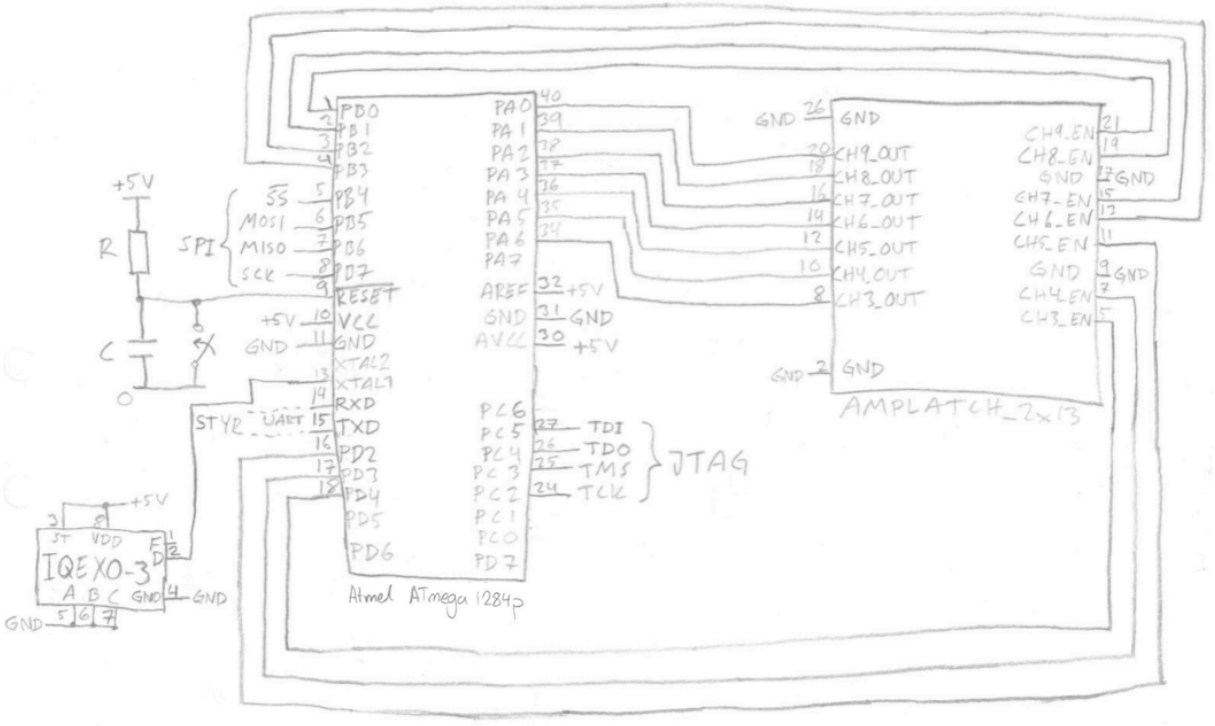
- 2 st. GP2D120 optiska avståndsmätare i intervallet 4-30cm.
- Godtycklig handske
- Flexsensor av Spectra Symbol
- MLX90609 gyro av Melexis
- Atmel ATmega16 mikrokontroller
- 2 st. FireFly Bluetooth moduler

- LiPo 7.4V batteri

AMPLATCH_2x13 är en reflexsensormodul bestående av 11 reflexsensorer varav 7 kommer att användas. Reflexsensorerna skickar ut infrarött ljus och mäter hur mycket ljus som reflekteras från underlaget. Informationen skickas till processorn där resultatet omvandlas till digital data. Data avrundas till antingen 0 eller 1 beroende på kalibreringen av reflexsensorerna.

5.3.3 Kopplingsschema

Nedan finns ett kopplingsschema för sensormodulen, *se figur 14.*

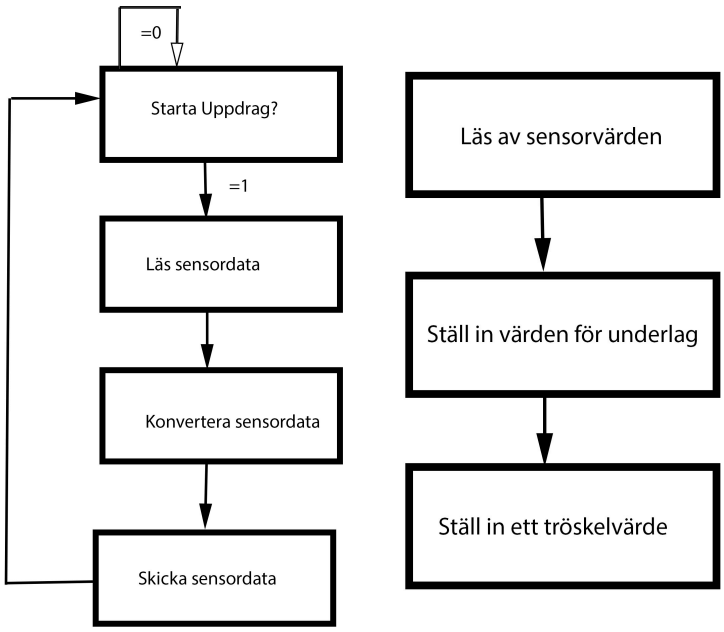


Figur 14, kopplingsschema för sensormodulen.

- PB0-3 samt PD2-4 kopplas till reflexsensormodulens ingångar. Processorn skickar ut logiskt 1 för att aktivera en reflexsensor annars logiskt 0.
- PA0-6 kopplas till reflexsensormodulens utgångar. Processorn tar emot analoga värden från sensorerna. Dessa värden omvandlas till digitala värden inuti processorn.
- PB4-7 (SS, MOSI, MISO, SCK) kopplas till bussen.
- RXD och TXD kopplas via UART till styrmodulen.
- PC2-5 (TDI, TDO, TMS, TCK) kopplas till JTAG-kedjan.
- XTAL1 kopplas till en klockpulsgenerator.
- RESET ska anslutas för att underlätta felsökning.

5.3.4 Mjukvara

Avläsningsloopen för sensormodulen ska under ett uppdrag läsa av sensordata, konvertera sensordata och skicka sensordata, *se figur 15.* Avbrott kommer användas vid A/D – omvandlingen i avläsningsloopen, men funktionen behöver inte utföra något under tiden det sker. Det ska finnas två knappar i användargränssnittet för att kalibrera sensorerna, den ena knappen ska bestämma vad som är underlag och den andra ska bestämma vad som är tejpmarkering. Ett tröskelvärde bestäms sedan för att kunna tolka sensordata och avgöra om signalen ska bli logiskt 1 eller logiskt 0 innan den skickas till styrmodulen.



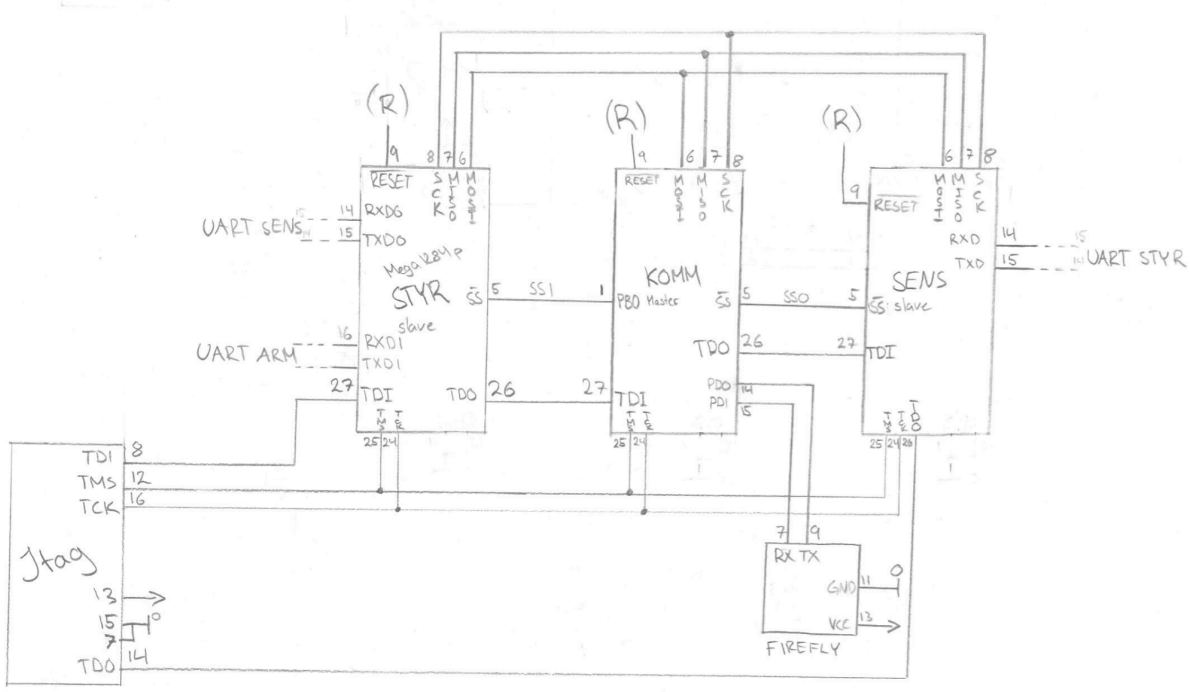
Figur 15, flödesschema för avläsning samt kalibrering.

5.3.5 Bedömning av prestanda

Inga tyngre beräkningar kommer att behöva utföras av sensormodulen. Dess uppgift (*krav 1)* är att ta emot sensordata från reflexmodulen och konvertera denna data, så att styrmodulen kan styra robotplattformen längs linjen. Vi väljer ATmega1284p istället för ATmega16 eftersom vi har ett krav 2 som kräver en extra UART. Vi gör bedömningen att Atmel ATmega1284p mikrokontroller har tillräckligt med prestanda för att klara av uppgiften.

6 Beskrivning av kommunikation mellan delsystem

Här beskrivs hur de olika modulerna i roboten kommunicerar med varandra och även hur robotenheten kommunicerar med datorenheten. Ganska översiktligt kommer det beskrivas vad det är för information som går mellan de olika modulerna.
Robotenheten har en intern SPI-buss där kommunikationsmodulen är master och en direktbuss mellan styrmodulen och sensormodulen. *Se figur 16* för översiktlig bild för systemet.



Figur 16, översiktligt kopplingsschema för systemet.

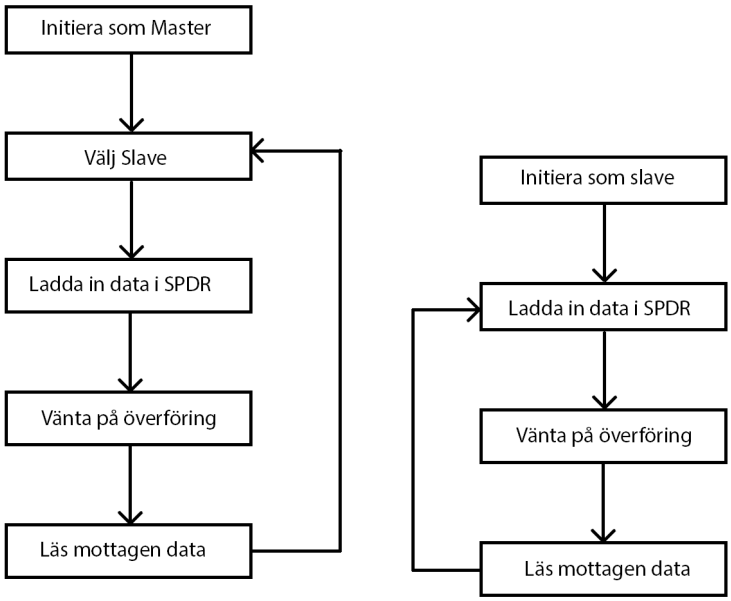
6.1 SPI-Bussen

SPI-bussen används framförallt för att distribuera kommandon som användaren skickar från datorn och för att skicka tillbaka styrbeslut och sensorvärden till datorn. Mellan datorenheten, styrmodulen, sensormodulen och kommunikationsmodulen kommer följande, *se tabell 1,* data skickas.

Tabell 1, översikt av data som överförs mellan modulerna samt datorenheten.

	Datorenhet	**Styrmodul**	**Sensormodul**
Kom.modul	**Data:** Styrbeslut, Sensorvärden, Regulatorvärden **Kommandon:** Start , Stop, Manuella styrkommandon arm/hjul, Kalibrera	**Data:** Styrbeslut, Regulatorvärden **Kommandon:** Start, Stop, Manuella styrkommandon arm/hjul	**Data:** Sensorvärden **Kommandon:** Start, Stop, Kalibrera

SPI-kommunikationen kommer följa flödesschema *figur 17* för master resp. slave. För Pseudokod *se bilaga 9.1.*

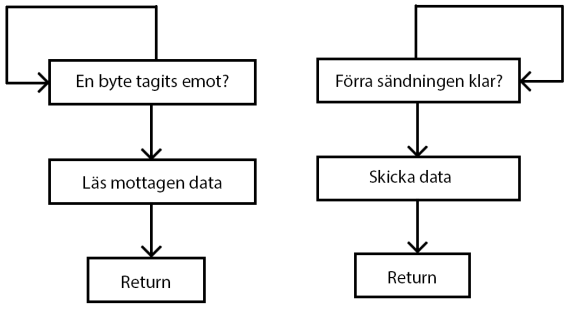


Figur 17, flödesschema för mottagning samt sändning via SPI.

6.2 Direktbussen

Mellan sensormodulen och styrmodulen finns en direktbuss för att skicka sensordata som ska användas för att följa linjen, vilket kommer göras via UART. Detta för att minska risken för eventuella fördröjningar av data från sensormodulen till styrmodulen, då styrmodulen behöver så aktuell sensordata som möjligt för reglering. Utan denna direktbuss hade informationen behövt gå via kommunikationsmodulen för att skickas vidare till styrmodulen.

UART-kommunikationen kommer följa flödesschema enligt *figur 18* för att skicka resp. ta emot data. För Pseudokod *se bilaga 9.2*.



Figur 18, flödesschema för sändning och mottagning via UART.

7 Implementeringsstrategi

För att utvecklingen ska kunna ske optimalt kommer först implementationen av bussen att prioriteras högst. I början av projektet ska också all hårdvara konstrueras. Därefter kommer programmering av delsystemen påbörjas. Med hjälp av en logikanalysator kommer funktionaliteten i varje delsystem till en viss del kunna testas separat. Efter att delsystemen har testats enskilt kopplas de samman och nya test utförs.

8 Referenser

[1] J. Hydén, ”Teknisk Dokumentation,” Linköpings Universitet, Linköping, 2015.

[2] T. Svensson och C. Kryssander, Projektmodellen LIPS, 1:1 red., Lund: Studentlitteratur AB, 2011.

9 Bilagor

9.1 Pseudokod SPI-kommunikation

```
// Initiera Masterenhet
void spi_init_master (void)
{
    // Sätt MOSI, SCK, SS0 och SS1 som
Output
    DDRB = (1<<5)|(1<<7)|(1<<4)|(1<<0);

    // Aktivera SPI, Sätt till Master
    //SCK-frekvens: Fosc/16
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

// Initiera Slavenhet
void spi_init_slave (void)
{
    DDRB = (1<<6);     //MISO till OUTPUT
    SPCR = (1<<SPE);   //Aktivera SPI
}

//Funktion för att skicka och ta emot data
för master och slave
unsigned char spi_tranceiver (unsigned
char data)
{
    //Ladda in data som ska skickas
    SPDR = data;

    //Vänta tills överföring är klar
    while(!(SPSR & (1<<SPIF) ));

    //Returnera mottagen data
    return(SPDR);
}
```

9.2 Pseudokod UART-kommunikation

```
// Definiera BAUD
#define BAUD 9600
// Sätt Baudratevärde för UBRR
#define BAUDRATE ((F_CPU)/(BAUD*16UL)-
// Funktion för att initiera UART
void uart_init (void)
{
// Skifta registret höger 8 bitar
    UBRRH = (BAUDRATE>>8);
// Sätt baudrate
    UBRRL = BAUDRATE;
// Aktivera mottagare och sändare
    UCSRB|= (1<<TXEN)|(1<<RXEN);
// Välj 8-bitars dataformat
    UCSRC|= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);
}
// Funktion för att skicka data
void uart_transmit (unsigned char data)
{
// Vänta tills registret är ledigt
    while (!( UCSRA & (1<<UDRE)));
// Ladda in data i registret
    UDR = data;
}
// Funktion för att ta emot data
unsigned char uart_recieve (void)
{
// Vänta tills data har mottagits
    while(!(UCSRA) & (1<<RXC));
// Returnera 8 bitar data
    return UDR;
}
```