Teknisk dokumentation

Redaktör: Hannes Snögren

Version 0.1

Status

Granskad		
Godkänd		

PROJEKTIDENTITET

HT2, 2014, Grupp 2
Linköpings Tekniska Högskola, ISY

Gruppdeltagare

Namn	**Ansvar**	**Telefon**	**E-post**
Pål Kastman	Projektledare	0703896295	palka285@student.liu.se
Hannes Snögren	Dokumentansvarig	0706265064	hansn314@student.liu.se
Alexander Yngve	Hårdvaruansvarig	0762749762	aleyn573@student.liu.se
Martin Söderén	Mjukvaruansvarig	0708163241	marso329@student.liu.se
Daniel Wassing	Leveransansvarig	0767741110	danwa223@student.liu.se
Dennis Ljung	Testansvarig	0708568148	denlj069@student.liu.se

Hemsida: http://github.com/ultralaserdeluxe/gloria

Kund: Tomas Svensson
Kontaktperson hos kund: Tomas Svensson
Kursansvarig: Tomas Svensson
Handledare: Peter Johansson

Innehåll

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1				

1 Inledning

Systemet Gloria är en lagerrobot som kan följa en bana, plocka upp paket vid särskilda stationer och sätta ned dem vid nästa tomma station.

Projektet utfördes som ett moment i kursen TSEA29 vid Linköpings Universitet under HT 2014[1]. Syftet med projektet var att ge gruppmedlemmarna övning i konstruktion och utveckling med mikrodatorer och erfarenhet i att jobba enligt en projektmodell, i det här fallet LIPS.

Det här dokumentet dokumenterar hur systemet är designat och fungerar. Syftet är dels att kunden skall kunna lösa uppkomna problem eller vidareutveckla systemet och dels att utomstående part i utbildningsyfte skall kunna förstå hur systemet fungerar.

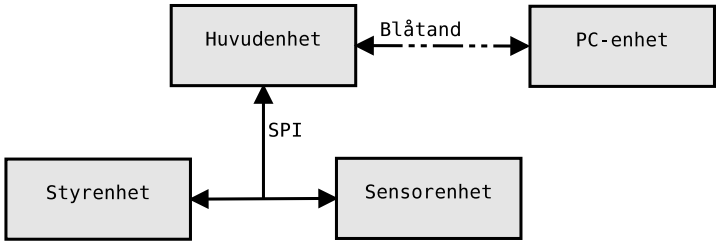
2 Produkten

Systemet Gloria är en fyrhjulig robot med en robotarm. **Den kan massor med saker och skitmycket funktionalitet**

3 Systemet

4 Kommunikation

Överväger att lägga figur 1 i Systemet och protokollen under varje enhet.



Figur 1 – Översikt systemets kommunikationskanaler

Figur 1 ger en översikt av vilka moduler som kommunicerar med varandra och på vilket sätt detta sker. Styrenheten och Sensorenheten använder samma SPI-buss fast med olika Slave Select-pinnar.

4.1 PC-enhet $\longleftrightarrow$ Huvudenhet

PC-enheten och Huvudenheten kommunicerar över Blåtand. **Hur set det här protokollet ut i verkligheten?**

4.2 Huvudenhet $\longleftrightarrow$ Styrenhet

Huvudenheten kommunicerar med Styrenheten över SPI. Storleken på en instruktion är antingen fyra eller sju bytes lång. För att förebygga synkroniseringsproblem skickas först två startbytes bestående av hexadecimalt $0xff$ följt av längden på instruktionen (startbytesen och längdbyten räknas ej med). Därefter följer instruktionsbyten som består av två delar. De fyra högsta bitarna anger vilken instruktion som ska utföras enligt tabell 1, och de fyra lägsta vilket servo eller motorpar instruktionen rör enligt tabell 2. I fallet att instruktionen är *Sätt register A till D* krävs dessutom två databytes.

Då ett motorpar adresseras anger den minsta biten i den första databyten vilken riktning motorparet skall röra sig. 0 anger framåt, 1 bakåt **Stämmer?**. Den andra databyten anger vilken fart vi vill att motorerna skall röra sig i. Notera att motorerna sätts med kommandot.

De servon vi arbetar med har en upplösning på 10 bitar både för position och hastighet. Vi måste alltså ha två databytes när vi vill ändra någon egenskap hos dessa. De två minsta bitarna i den första databyten är de två högsta bitarna och därefter följer den andra databyten.

Tabell för att illustrera bitar hit och dit?

4.3 Instruktionslista

Instruktion	Argument	Beskrivning
0000		Stoppa samtliga servon och motorer **Behöver implementeras**
0001	A, D	Sätt register A till D
0010	A	Utför givna kommandon för A
0011	A, D	Sätt servohastighet för A till D
0100	A	Returnera Status för A **Ska det användas?**

Tabell 1 – Kommandon från huvudenhet till styrenhet **Stämmer tabellen?**

Adress	Beskrivning
0000	Höger hjulpar
0001	Vänster hjulpar
0010	Arm axel 1
0100	Arm axel 2
0110	Arm axel 3
1000	Arm axel 4
1011	Arm axel 5 (*gripklo*)
1100	Samtliga motorer
1101	Samtliga servon
1111	Samtliga motorer och servon

Tabell 2 – Adresser för adressering till styrenhet

4.4 Huvudenhet $\longleftrightarrow$ Sensorenhet

Huvudenheten kommunicerar med Sensorenheten över SPI. Protokollet mellan dessa moduler är väldigt primitivt då det endast finns en instruktion, returnera sensordata för begärd sensor. Instruktionen anges av de 4 högsta bitarna av instruktionsbyten enligt tabell 3. Huvudenhetens begäran är då bara på en enda databyte och Sensorenheten svarar med de 8 högsta bitarna för den efterfrågade sensorn. Vilken sensor som enheten returnerar data för anges av de 4 minsta bitarna i instruktionsbyten enligt tabell 4.

4.5 Instruktionslista

Instruktion	Argument	Beskrivning
0000	A	Returnera sensordata för A

Tabell 3 – Instruktion från huvudenhet till sensorenhet

Adress	Beskrivning
0000	Linjesensor 1
0001	Linjesensor 2
0010	Linjesensor 3
0011	Linjesensor 4
0100	Linjesensor 5
0101	Linjesensor 6
0110	Linjesensor 7
0111	Linjesensor 8
1000	Linjesensor 9
1001	Linjesensor 10
1010	Linjesensor 11
1011	Avståndssensor Höger
1100	Avståndssensor Vänster

Tabell 4 – Adresser för instruktioner till sensorenhet

5 PC-enhet

5.1 Hårdvara

5.2 Mjukvara

6 Huvudenhet

6.1 Teori

Skulle kunna lägga det här som en egen section

6.1.1 Reglering

Hur tolkar vi sensordata? Hur fungerar regleringen. Detta får yngve skriva

6.1.2 Styrning av arm

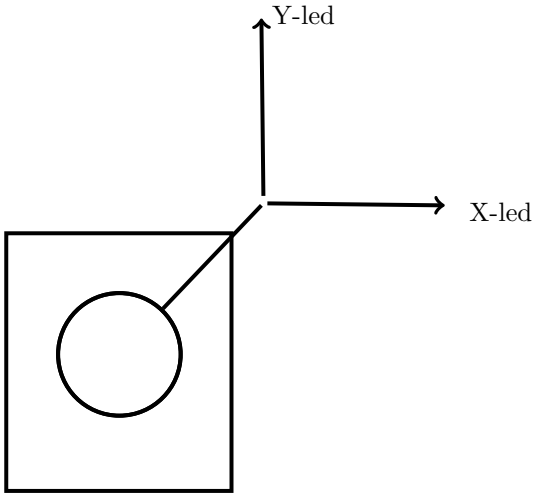
Omvandlingen från (x, y, z)-koordinater till vinklar för armens servon sker på huvudenheten. På huvudenheten finns en pythonmodul `arm.py` som implementerar beräkningarna som beskrivs nedan. Den används i tre steg.

1. Skapa en instans av robotArm som är en klass i `arm.py`.
2. Använd funktionen `setAll` som tar in en lista som argument med följande innehåll $[x,y,z,gripperAngle,gripperRotationsOffset,gripper]$.
3. Använd funktionen `getServoValues` som returnerar en lista med alla servons vinklar i form av 10-bitars heltal.

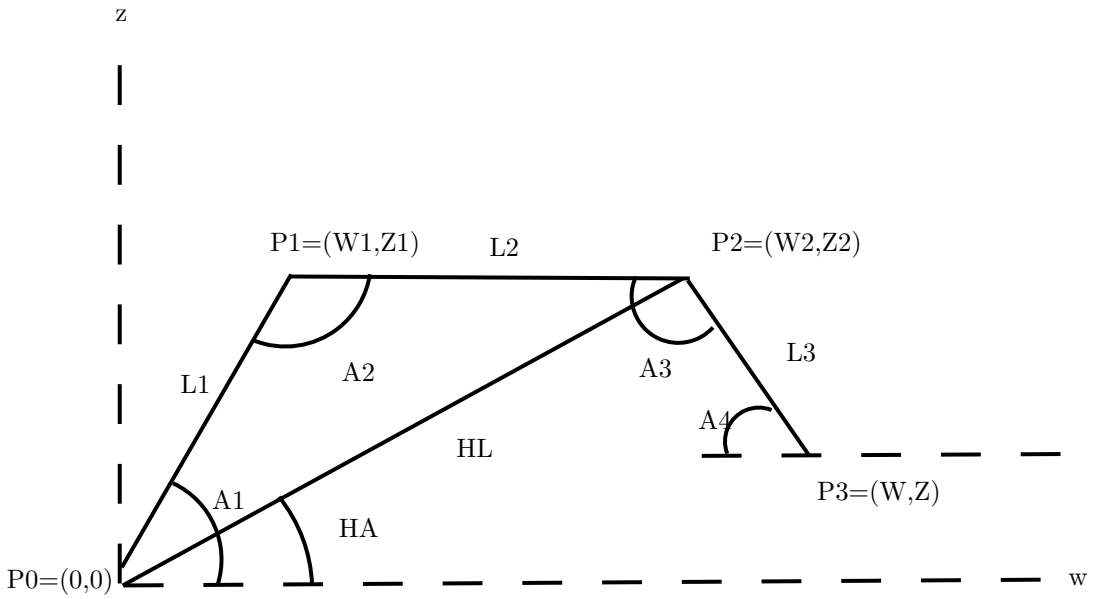
GripperAngle är gripklons vinkel mot marken, gripperRotationsOffset är en offset på gripklons vinkel i förhållande till leden som gripklons är fäst i och gripper är hur mycket gripklon klämmer.

I listan som returneras är första värdet till armens bas, det andra är till $A1$, det tredje är till $A2$, det fjärde till $A3$, det femte till klons rotation och det sjätte till klons grepp.

Omvandlingen sker på följande sätt där gripperRotationOffset och gripper ignoreras. Antag att roboten startar i $(X, Y, Z, GR) = (0, 0, 0, \pi/2)$. Där GR(GripRadian) är vilken vinkel gripklon har till z-axeln.
Vi omvandlar våra rum-koordinater till en vinkel och plankoordinater $X, Y, Z, GR \rightarrow A, W, Z, GR$.



Figur 2 – Armens koordinatsystem i förhållande till roboten. Upp är robotens färdriktning



Figur 3 – Armens vinklar

Där A är armens vinkel mot positiva x-axeln och W är armens utsträckning. **Visualisera omvandling rumkoordinater till plan**

$$A = tan^{-1}(\frac{Y}{X})$$

$$W = \sqrt{X^2 + Y^2}$$

Genom denna omvandlingen så kan vi göra om detta till ett 2-dimensionellt problem istället för ett 3-dimensionellt problem. Vi får in X,Y,Z,GR och ställer in robotens vinkel mot x-axelns positiva del och sedan ser vi till att roboten bara har rätt utsträckning och höjd.

$$P_4 = (W, Z)$$

$$P_0 = (0, 0)$$

$$W_2 = W - cos(GR) * L_3$$
$$Z_2 = Z - sin(GR) * L_3$$
$$H_L = \sqrt{W_2^2 + Z_2^2}$$
$$H_A = tan^{-1}(\frac{Z_2}{W_2})$$

Cosinussatsen ger:

$$A_1 = cos^{-1}(\frac{L_1^2 + H_L^2 - L_2^2}{2 * L_1 * H_L}) + H_A$$
$$W_1 = cos(A_1) * L_1$$
$$Z_1 = sin(A_1) * L_1$$
$$A_2 = tan^{-1}(\frac{Z_2 - Z_1}{W_2 - W_1}) - A_1$$
$$A_3 = GR - D_2 - D_3$$

Då är alla vinklar beräknade men servornas nollpunkter är inte på samma ställen som nollställena i beräkningarna ovanför. Så följande offset ställs in i grader:

- $A0 = A0 + 60$
- $A1 = 240 - A1$
- $A2 = 240 - A2$
- $A3 = 150 - A3$
- $A4 = 240 - A0 + GripperRotationOffset$ (klons rotation beror på basens rotation)
- $A5$ har ingen offset

Dessa vinklar är nu i grader. Dessa multipliceras nu med 3.41 **Varför?** och avrundas nedåt till heltal och kan nu användas direkt av styrenheten.

6.2 Hårdvara

Den hårdvara som ingår i huvudenheten består av en Beagleboard-xm. På denna sitter en Blåtandsenhet av typ **Typ/modell av BT?** monterad för kommunikation med PCenheten. Denna är ansluten till styrenheten och sensorenheten med en flatkabel som innehåller SPI-busen. **Bild på BB?**

6.3 Mjukvara

6.3.1 Allmänt

Mjukvaran är uppdelad i fyra trådar. En maintråd som har hand om all styrlogik, en pctråd som hanterar all kommunikation med PC:n, en sensortråd som kontinuerligt uppdaterar huvudenhetens sensorvärden. Slutligen en regulatortråd som har hand om regleringen så roboten kan följa banan utan problem. Alla dessa kommunicerar med varandra genom ett dictionary som innehållar alla olika variabler som behöver delas. Kommandon från PC:n delas mellan pctråden och maintråden genom en kö.

6.3.2 Kommunikation

För kommunikationen med PC:n så används pythons standard socketpacket för att sätta upp en tcp/ip anslutning med en dator. Detta startas upp vid boot. Skulle en anslutning brytas av någon anledning så startas anslutning om igen och inväntar en ny anslutning. För närvarande så används PAN över bluetooth för att ansluta till huvudenheten. Detta gör att man kan SSH in på den coh konfigurera den trådlöst. Emacs och Nano är installerat för editering av filer. Man kan även ansluta till huvudenhet över ethernetkabel eller låta den ansluta till ett trådlöst nätverk men blåtand är den mest beprövade metoden. För att ansluta en ny dator med Linux till huvudenhetens PAN behövs följande göras: **får uppdatera detta, osäker om detta stämmer**

1. ha tillgång till en terminal på huvudenheten, den enklaste är att ansluta en skärm och tangentbord till den.
   Lösenordet är :temppwd”
2. kör sudo bluez-simple-agent hci0 xx:xx:xx:xx:xx:xxdär du byter ut kryssen mot din blåtandenhets mac-adress
3. paira huvudenheten med din pc från din pc
4. kör sudo bluez-test-device trusted xx:xx:xx:xx:xx:xx yespå huvudenheten
5. du kan nu ta bort pairing från din pc
6. kör sudo pand -n -c 00:19:0E:0F:F0:6F på din pc
7. kör sudo ifconfig bnep0 192.168.99.2 up på din pc

Nu borde du kunna pinga och ssh in på ubuntu@192.168.99.1 vilket är huvudenhetens statiska ip-adress. I framtiden behövs bara de två sista stegen genomföras. **Sista delen bör flyttas till användarhandledning**

6.3.3 Maintråd

Hur tolkar vi sensordata? bäst om dennis skriver denna

6.3.4 PCtråd

PCtråden sätter upp en socket när den skapas och väntar på en inkommande anslutning från en PC. När den får en anslutning så väntar den på ett kommando. Kommandot kommer som en text-sträng på formatet *;kommando=argument1,argument2;*. Om kommandot inte har några argument så ser strängen ut på följande sätt *:”kommando”*. Denna sträng görs om till en lista med följande utseende *: [”kommando”,[ärgument1”,ärgument2”]]*.

Oftas tas flera kommandon emot samtidigt och PCtråden gör om dessa till en lista av kommandon som den går igenom och behandlar en efter en. Får men till exempel in två kommandon om sätta motorhastigheten så ser tillvägagångssättet ut på följande sätt:

1. ”;motorSpeed=speed1,speed2;motorspeed=speed3;speed4;tas emot
2. detta görs om till [[motorspeed”,[speed1”,speed2”]],[motorSpeed”,[speed3”,speed4”]]]
3. det första kommandot behandlas genom att argumenten omvandlas till intergers, motorernas hastighet uppdateras i dictionaryn och kommandot läggs till i kön så maintråd vet att den ska skicka ut kommandot.
4. nästa kommando behandlas på samma sätt men om speed1=speed3 och speed2=speed4 så ignoreras kommandot då det inte påverkar hastigheterna.

6.3.5 Sensortråd

I denna tråd sätts en uppdateringfrekvens och sedan hämtar denna tråd alla värden från sensorenheten och lägger dessa i dictionaryn så ofta.

6.3.6 RegulatorTråd

I denna tråd sätts en uppdateringsfrekvens som för nuvarande är satt till 50Hz och sedan så beräknar denna tråd ut reglerfelet så ofta utgående linjesensorns värden som finns i dictionaryn. Från detta så beräknas hastigheterna ut för vardera motorpar och placeras i en egen plats i dictionaryn som sedan maintråden kan välja att använda eller ej.

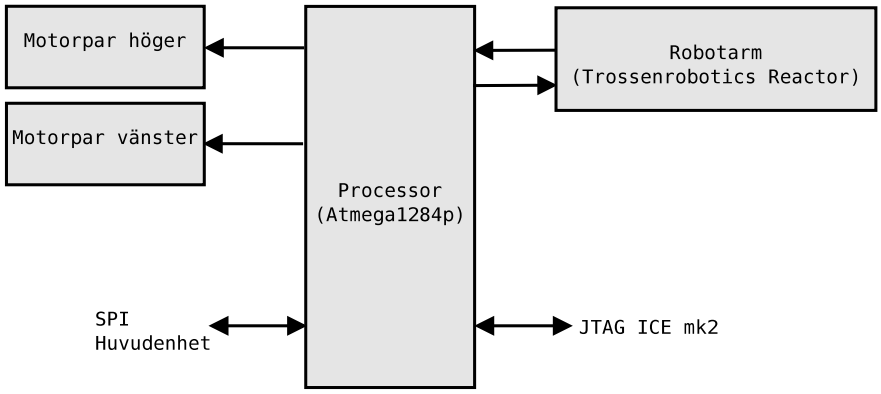
Anledningen till detta hamnade i en egen tråd var för att säkerhetsställa att beräkningarna sker kontinurligt och inte beror på loopen i mainthred.

Schema över asmycket trådar

7 Styrenhet

Styrenheten har till uppgift att ta kommandon från huvudenheten och producera utsignaler som motorer och servon kan förstå.

7.1 Hårdvara



Figur 4 – Översikt av styrenheten

7.2 Mjukvara

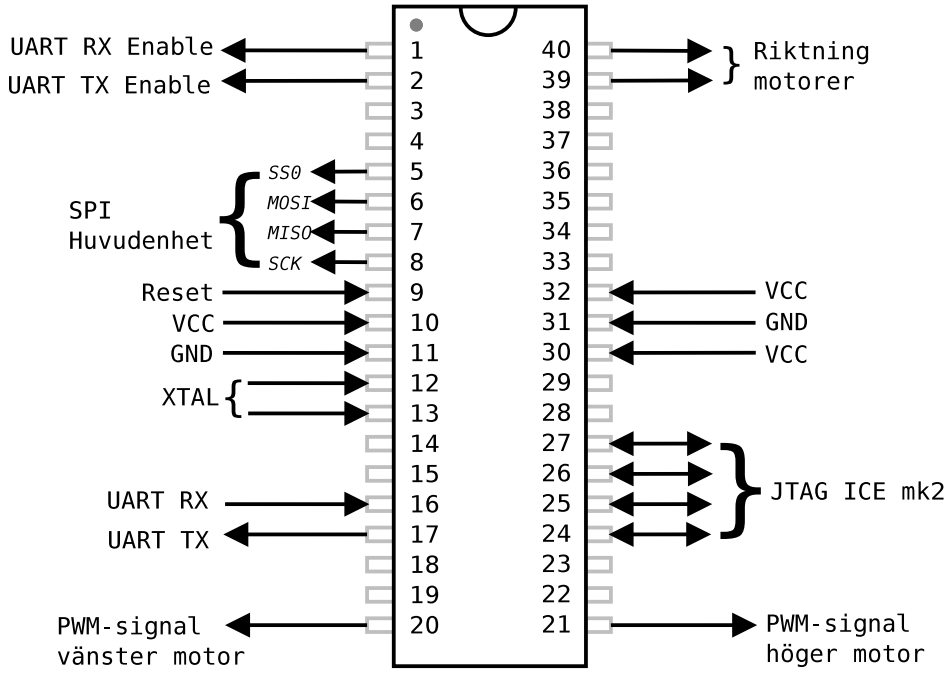
Flödesschema styrenheten

8 Sensorenheten

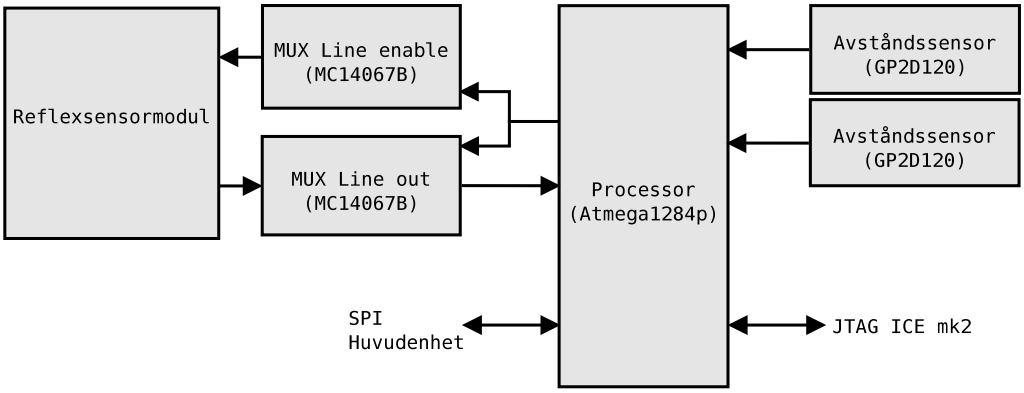
Sensorenheten har till uppgift att förse huvudenheten med sensordata. Sensordatan den returnerar är obehandlad rådata.

8.1 Hårdvara

Sensorenheten består av en Atmega1284p, en reflexsensormodul och två avståndssensorer. Reflexsensormodulen är kopplad via två 16-1 muxar till enkretsdatorn. Den ena muxen används för



Figur 5 – Schema över hur enkretssdatorn i styrenheten är ansluten till övrig hårdvara.



Figur 6 – Översikt av sensorenheten

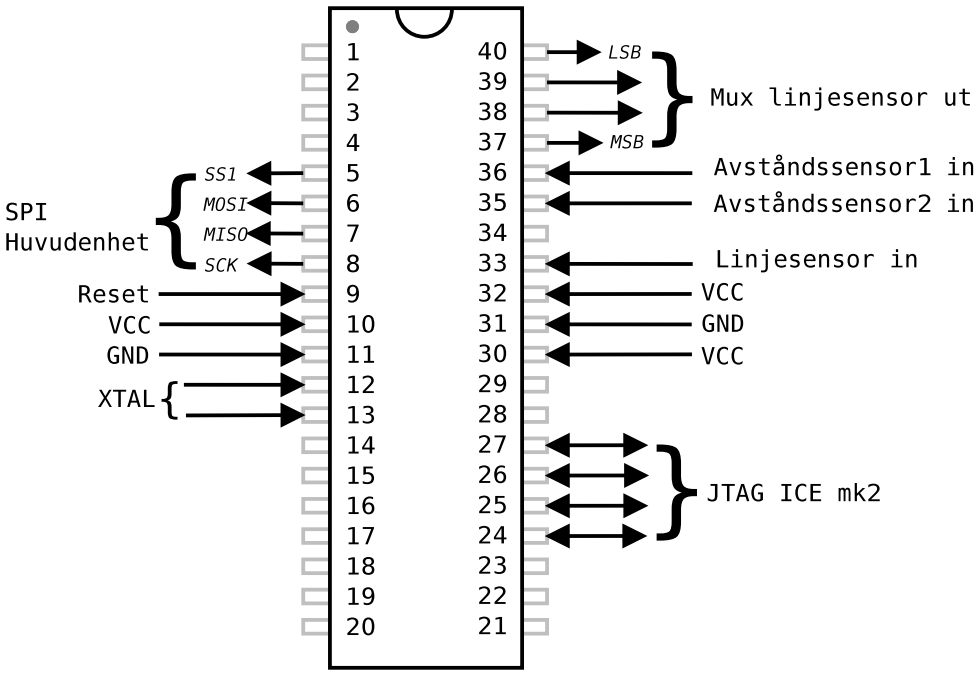
att styra en konstant hög enable-signal till rätt reflexsensor och den andra för att välja rätt utsignal. Båda muxarna styrs av samma styrsignal från processorn. Enkretsdatorn är ansluten till huvudenheten med en **20lol-pinnarskabel?** över vilken de kommunicerar via SPI.

8.1.1 Reflexsensormodul

Reflexsensormodulens syfte är att detektera banan längs vilken roboten skall röra sig. Den består av 11 reflexsensorer som i sig består av en IR-diod och en fototransistor. Utsignalen ligger mellan 0V och 5V. De ger låg utspänning då underlaget reflekterar mycket ljus, och hög utspänning då lite ljus reflekteras.

8.1.2 Avståndssensorer

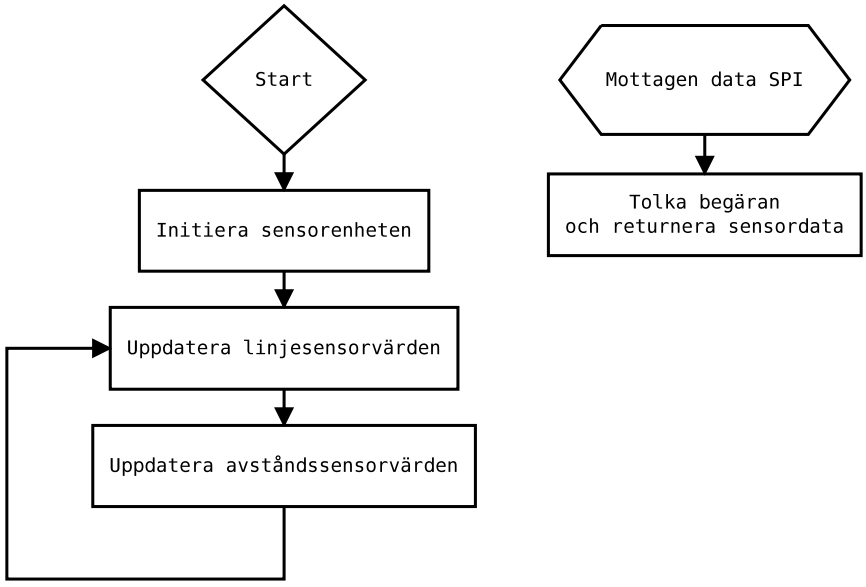
Avståndssensorerna är av typen GP2D120 vilken använder ljus för att detektera avståndet. Dess utsignal är en analog spänning mellan 0V och 3.2V.



Figur 7 – Schema över hur enkretssdatorn i sensorenheten är ansluten till övrig hårdvara.

8.2 Mjukvara

All mjukvara på sensorenheten är skriven i *C* och finns på enkretsdatorn. Den består av två delar, illustrerat i figur 8.



Figur 8 – Till vänster sensorenhetens **huvudslinga**, till höger den avbrottsrutin som körs vid mottagen data över SPI.

För det första körs en mainloop där sensordata kontinuerligt uppdateras. Först itererar vi igenom Reflexsensorerna genom att först styra om enable-signalen till den aktuella sensorn och därefter utföra en AD-omvandling på den signal vi får tillbaks och sedan gå till nästa. Därefter

utför vi i tur och ordning en AD-omvandling på insignalerna från avståndssensorerna. Alla värden sparas i en global datastruktur.
För det andra tar sensorenheten emot begäran om sensordata från huvudenheten över SPI. När en sådan förfrågan inkommer triggas ett avbrott i vilket sensorenheten svarar med det aktuella sensorvärdet. Då inget i avläsningen av sensorerna är tidskritiskt behöver vi inte oroa oss för när dessa avbrott kommer.

9 Slutsatser

Referenser

[1] http://www.isy.liu.se/edu/kurs/TSEA29/, information hämtad 2014-09-**tidigt?**.

[2] http://www.commsys.isy.liu.se/sv/student/kurser/TATA62/lips, information hämtad 2014-09-25.

[3] LIPS, Studentlitteratur, Tomas Svensson, Christian Krysander **Gör rätt**

[4] http://www.trossenrobotics.com/p/phantomx-ax-12-reactor-robot-arm.aspx, information hämtad **hittepå**.

[5] http://support.robotis.com/en/techsupport_eng.htm#product/dynamixel/ax_series/dxl_ax_actuator.htm, information hämtad 2014-10-24.

[6] http://www.atmel.com/images/doc8059.pdf, information hämtad **hittepådatum**.

[7] http://www.sharpsma.com/webfm_send/1205, information hämtad **Hittepådatum**.

[8] http://www.ubuntu.com, **Hur refererar vi ubuntu?**

[9] https://docs.isy.liu.se/twiki/pub/VanHeden/DataSheets/reflexsensormodul.pdf, information hämtad **hittepådatum**.

[10] https://docs.isy.liu.se/twiki/pub/VanHeden/DataSheets/4067b.pdf, information hämtad **hittepådatum**.

[11] https://docs.isy.liu.se/twiki/pub/VanHeden/DataSheets/sn74ls125.pdf, information hämtad **hittepådatum**.

[12] http://www.ti.com/lit/ds/symlink/txb0104.pdf, information hämtad **hittepådatum**.

Resurs/datablad för Beagleboard?
Datablad för kristall/oscillator
Datablad för tryckknapp