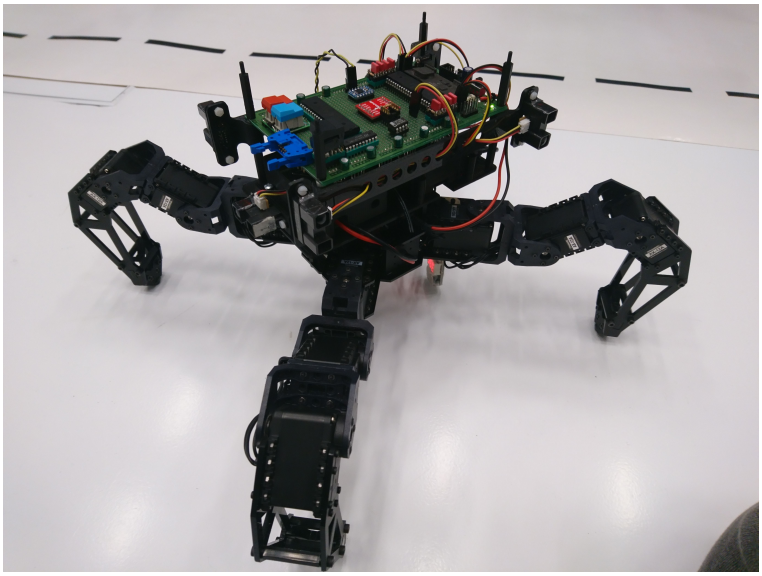
Teknisk Dokumentation
Quadrapod

Redaktörer: Viktor Blidh, Tobias Fellegi, Aron Gosch,
Alfred Hagberg, Mustaf Musse, Niclas Byrsten

23 maj 2018

Version 1.1



Status

Granskad		
Godkänd		

Projektidentitet

Namn	Ansvar	Telefon	E-post
Viktor Blidh	Projektansvarig (PL)	070-6941166	vikbl327@student.liu.se
Tobias Fellegi	Dokumentansvarig (DOK)	070-4282800	tobfe318@student.liu.se
Niclas Byrsten	Designansvarig (DES)	072-8525164	nicby889@student.liu.se
Mustaf Musse	Testansvarig (TST)	076-2633149	musab250@student.liu.se
Aron Gosch	Kvalitetssamordnare (QS)	073-0744622	arogo305@student.liu.se
Alfred Hagberg	Implementationsansvarig (IMP)	076-1459838	alfha306@student.liu.se

E-postlista för hela gruppen: vikbl327@student.liu.se, tobfe318@student.liu.se, nicby889@student.liu.se, musab250@student.liu.se, arogo305@student.liu.se, alfha306@student.liu.se

Kund: Tomas Svensson, tomas.svensson@liu.se

Kursansvarig: Tomas Svensson, tomas.svensson@liu.se
Handledare: Anders Nilsson, anders.p.nilsson@liu.se

Sammanfattning

Denna tekniska dokumentation beskriver konstruktionen av en 4-bent autonom robot med mjukvara. Roboten är uppdelad i tre olika moduler som alla samverkar för att göra roboten autonom, dessa moduler kallas styr-, sensor- samt kommunikationsmodulen. Styr- och sensormodulerna har båda en ATmega1284 mikrokontroller som kärna, medan kommunikationsmodulen använder en Raspberry Pi 3.

Styrmodulen ansvarar för styrandet av servon (12st AX-12A) i benen och därmed också gången. Servon är fördelade med 3 st på vardera ben och genom att samordna rörelsen i dessa så kan den få ett ben att röra sig och med hjälp av alla ben tillsammans få roboten att gå.

Sensormodulen hämtar in information om omvärlden med hjälp av 5 st IR-sensorer av modellen SHARP GP2Y0A02YK och tolkar dessa som distanser till väggar runt omkring roboten. Dessa värden skickas sedan vidare till kommunikationsmodulen.

Kommunikationsmodulen är den centrala enheten i roboten, där den sköter många beräkningar samt gör alla val i sin labyrintlösningsalgoritm. Den kommunicerar med de andra modulerna genom SPI och med datorn genom Wifi.

Till projektet följer också en applikation med ett GUI som enkelt låter användaren skicka kommandon till roboten och läsa värden som exempelvis sensordata från sensorenheten.

Programmet för datorn är skrivet i C++ med hjälp av QtCreator med dess inbyggda fönsterhantering, detta betyder också att programmet fungerar både på Linux- och Windows-baserade system. Mjukvara för styr- och sensorenhet är skriven i C i programmet Atmel studio och kod för kommunikationsmodulen är skriven i Python 3.

Innehåll

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
1.0	2017-12-14	Första versionen	Gruppen	2017-12-18
1.1	2017-12-18	Utfört kompletteringar	TF, AG	

1 Inledning

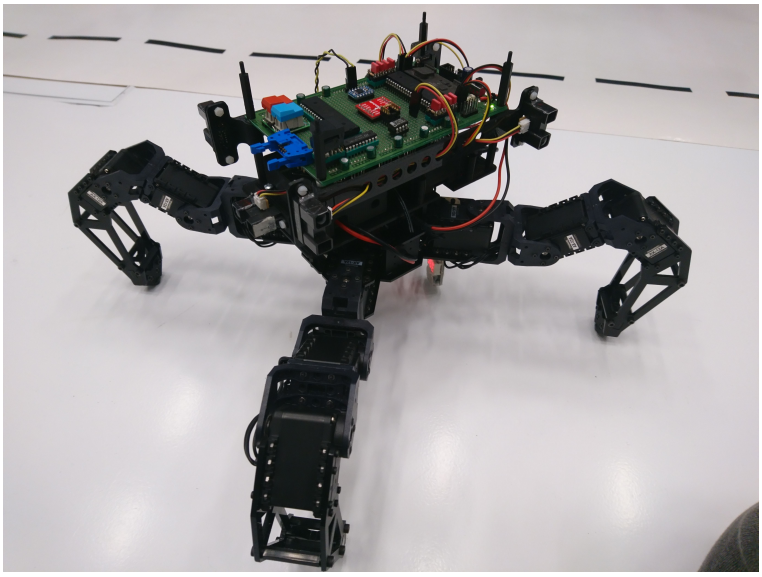
1.1 Bakgrund

Detta dokument är en beskrivning av den tekniska lösningen för en 4-bent labyrintrobot som gjorts inom kursen *TSEA29 - Konstruktion med mikrodatorer* på Linköpings universitet. Detta projekt har utförts under höstterminen 2017.

1.2 Syfte och mål

Syftet med detta dokument är att ge en ingående beskrivning av hur roboten är konstruerad samt förståelse för hur roboten skulle kunna rekonstrueras. Målet bakom dokumentet är även att ge en djupare förståelse för robotens uppbyggnad så att underhåll och felsökning av roboten blir enklare.

2 Produkten



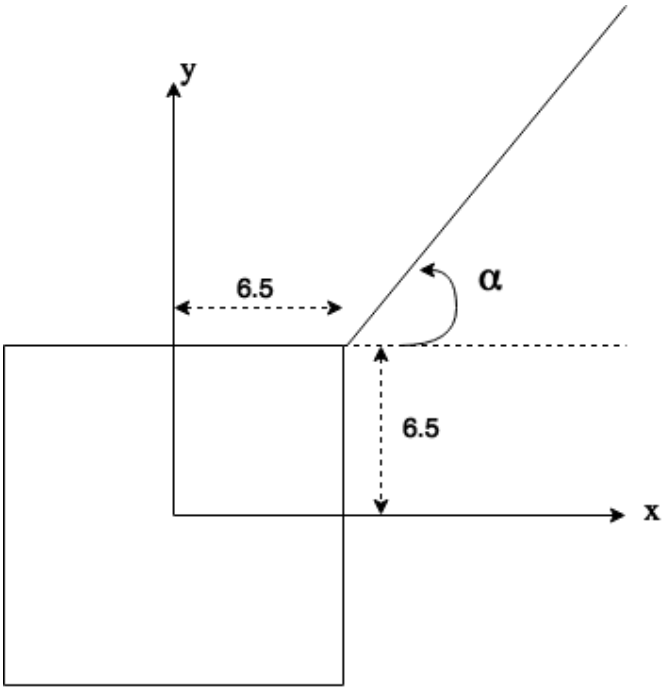
Figur 1: Bild på 4-bent robot.

Roboten, vid namn Brobot, är en 4-bent vandrande robot designad för att kunna styras manuellt med hjälp av en dator kopplad till roboten över Wifi samt att den ska kunna autonomt ta sig ut ur en labyrint byggd enligt banspecifikationerna medföljande kravspecifikationen. Roboten blir lätt att kontrollera om man läser den tillhörande användarhandledningen vilket förklarar hur mjukvaran till datorn kontrollerar det manuella läget samt autonomt läge.

3 Teori

3.1 Invers kinematik

Inom invers kinematik så utgår man från slutpositionen där ett ben ska sättas ned och räknar sedan ut de ledvinklar som krävs för att benet ska hamna på denna punkt. För att bestämma vinklar för samtliga tre leder behöver vi tre olika ekvationer och dessa kan man sedan lösa med hjälp av trigonometriska samband. I figur 1 ser vi hur det ser ut ovanifrån när roboten förflyttar ett ben framåt. Axelleden förskjuts då med en vinkel α i xy-planet för att foten längst ut på benet ska nå en viss punkt $P = (x, y, z)$.



Figur 2: Roboten sett ovanifrån.

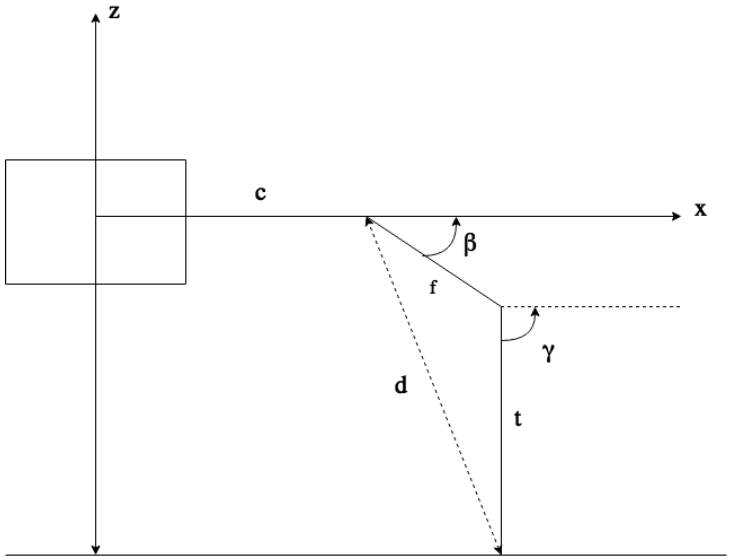
Den första vinkeln α får vi genom:

$$\frac{y - 6.5}{x - 6.5} = \tan \alpha \tag{1}$$

$$\alpha = \arctan \frac{y - 6.5}{x - 6.5} \tag{2}$$

Om man nu studerar roboten från sidan, som i figur 2, kan man ställa upp ytterligare ett problem för de två andra ledvinklarna β och γ. I figur 2 ser vi

hur lederna COAX, FEMUR och TIBIA bildar trianglar som vi kan utnyttja för att beräkna de ledvinklar som söks. FEMUR, TIBIA och fotspetsen utgör här en triangel där vinkeln för femur-leden. För enkelhetens skull kommer lederna att skrivas som c, f och t i de matematiska formlerna.



Figur 3: Roboten sett från sidan.

För att kunna beräkna vinklarna i figur 3 med cosinussatsen behöver vi först bestämma avståndet d mellan femur-leden och fotspetsen.

$$d^2 = (x - 6.5 - \frac{cx}{\sqrt{x^2+y^2}})^2 + (y - 6.5 - \frac{cx}{\sqrt{x^2+y^2}}y)^2 + z^2 \tag{3}$$

Detta ger att β blir:

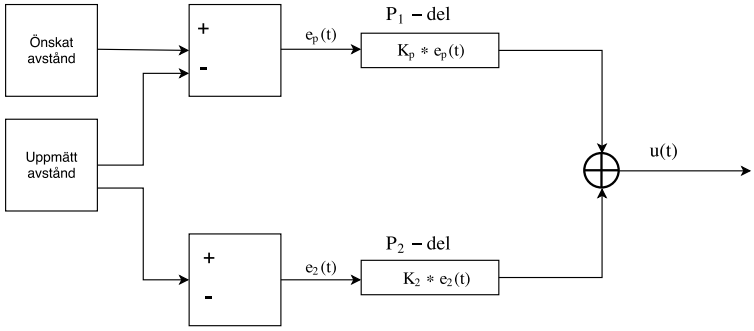
$$\beta = \arccos \frac{-z}{(x-6.5)^2 + (y-6.5)^2} - 5 - \arccos \frac{-t^2 + f^2 + d^2}{2fd} \tag{4}$$

Slutligen erhålls den sista vinkeln γ genom:

$$\gamma = \arccos \frac{d^2 - f^2 - t^2}{2tf} \tag{5}$$

3.2 P-regulator

En P-regulator är en variant på den vanligt förekommande PID-regulatorn inom reglertekniken. Förkortningen PID kommer från dess tre olika element: en proportionerlig del, en integrerande del och en deriverande del. En bild på P-regulatorn som används vid reglering av roboten finns nedan.



Figur 4: Modifierad P-regulator.

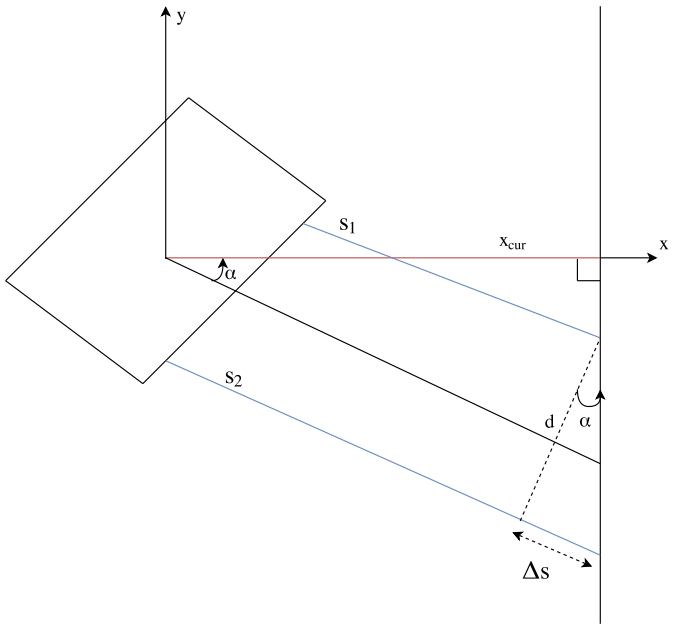
Som man ser i figur 4 så är detta en variant av P-reglering som har modifierats för att bättre passa situationen för vår robot med två sidosensorer. Regleringen delas upp i en P_1-del som korrigerar robotens position i x-led och en P_2-del som korrigerar dess lutning. Styrsignalen $u(t)$ ges således genom formeln:

$$u(t) = K_p \cdot e_p(t) + K_2 \cdot e_2(t) \tag{6}$$

P_1-delen utgörs av en proportionalitetskonstant multiplicerat med differensen i x-led från vårt önskade värde och P_2-delen utgörs av en annan proportionalitetskonstant multiplicerat med förhållandet mellan framsensor och baksensor. När roboten går ifrån ursprungsläget och närmar sig en vägg blir P_1-delen dominerande och motverkar förändringen. På samma sätt tar P_2-delen över när roboten befinner sig i mitten och ser till att den då håller sig rakt i banan. Detta kan sammanfattas som att:

- u(t) = 0: Roboten fortsätter rakt fram.
- u(t) > 0: Roboten svänger höger.
- u(t) < 0: Roboten svänger vänster.

För att beräkna termerna i P-regulatorn behöver vi först rita upp en bild på roboten i banan.



Figur 5: Reglering mot högervägg.

Förklaring till variablerna i figur 5 följer nedan:

- x_{cur}: Nuvarande avstånd från mitten av robot till vägg.
- α: Vinkel från robot till vägg.
- s_1: Uppmätt avstånd till vägg för höger sensor fram.
- s_1: Uppmätt avstånd till vägg för höger sensor bak.
- d: Avstånd mellan sensorer.

Vinkeln till högerväggen ges av:

$$\alpha = \arctan \frac{s_1 - s_2}{d} \tag{7}$$

Vi kan ta fram det nuvarande avståndet till väggen x_{cur}, men man behöver även ta hänsyn till monteringsavståndet på sensorerna som är 8 cm från mitten av roboten.

$$x_{cur} = (\frac{s_1 + s_2}{2} + 8) \cos \alpha \tag{8}$$

För att beräkna felet i P_1-del och P_2-del kan man nu ställa upp följande ekvationer.

$$e_p(t) = x_{desired} - x_{cur} \tag{9}$$

$$e_2(t) = \frac{s_1 - s_2}{d} \tag{10}$$

Slutligen kan vi uppdatera vårt uttryck i ekvation 6 med insättning av framtagna värden:

$$u(t) = K_p \cdot (x_{desired} - x_{cur}) + K_2 \cdot \frac{s_1 - s_2}{d} \tag{11}$$

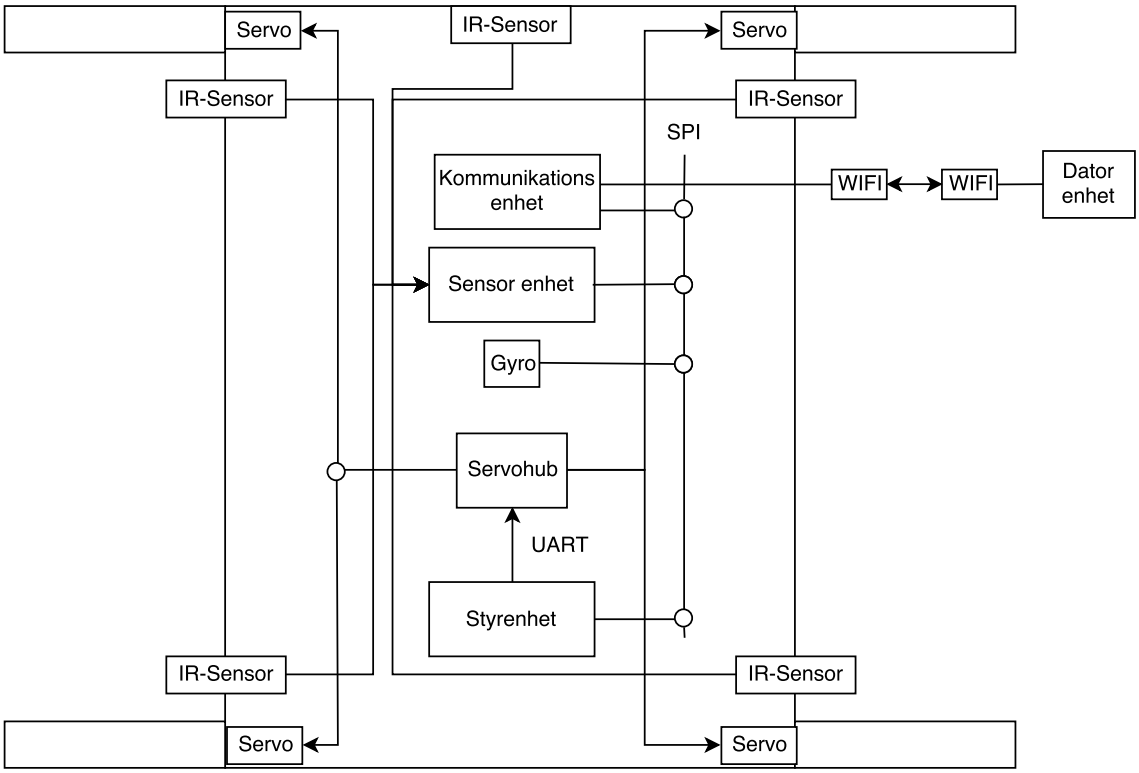
Med samma resonemang kan man naturligtvis reglera mot en vänstervägg och både höger och vänster samtidigt om man tar medelvärdet mellan de två formlerna.

4 Systemet

Systemet består av tre olika enheter: kommunikationsenhet, styrenhet och sensorenhet. Kommunikationsenheten är vår huvudenhet, som hanterar all data som skickas till de andra enheterna. Detta förklaras mer ingående i sektion 5.

4.1 Blockschema

I figur 6 nedan kan man få en enkel översikt på systemet och hur de tre enheterna är sammankopplade.



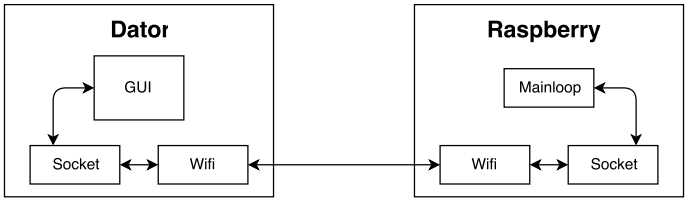
Figur 6: Översikt blockschema.

4.2 Kommunikation

Roboten använder sig av tre olika sätt för att kommunicera, den använder SPI mellan modulerna, Wifi för att koppla samman kommunikationsmodulen med datorn som används för manuellt läge samt UART för att kommunicera med servon i benen.

4.2.1 Wifi

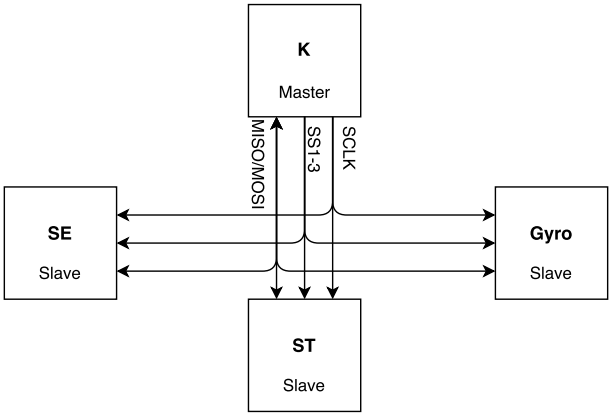
Datorn ansluts till Kommunikationsmodulen med hjälp av en Wifi-anslutning, mjukvaran som styr roboten skapar en klient-socket och ansluter till en server-socket som väntar på kom-modulens sida. När denna kontakt är färdig så kan datorn skicka kommandon till kom-modulen och ta emot sensor- och servo-data. Denna socket använder sig av TCP/IP protokoll. Detta illustreras i figur 7. Kommandon och information som sänds mellan dator och kom-modulen beskrivs närmare i sektion 5 Kommunikationsmodul.



Figur 7: Kommunikation mellan Raspberry och GUI.

4.2.2 SPI

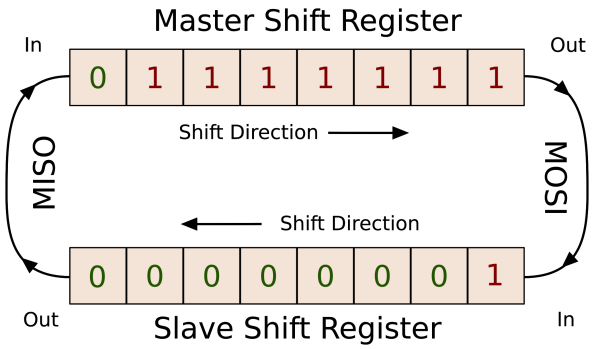
Kommunikation internt i roboten sker över en SPI-buss där Kommunikationsmodulen agerar master och gyro, sensor- samt styrmodulen agerar slaves. Detta betyder att kom-modulen är enheten som styr när data ska utbytas mellan sig själv och de andra enheterna. Figur 8 visar en simpel SPI-koppling mellan enheterna, det borde dock förtydligas att MOSI, MISO samt de tre olika Slaves selects (SS1-3) har var sin kabel och inte delar koppling med varandra.



Figur 8: Kommunikation med SPI.

En slave select var behövs för att kom-modulen ska kunna välja en slave

genom att sätta den SS-signalen låg. MOSI (Master Output, Slave Input) är kabeln som data kommer ut från kom-modulen till slaves bara den valda slaven kan läsa av MOSI. MISO (Master Input, Slave Output) är kabeln för att skicka bitar från slaven till mastern.

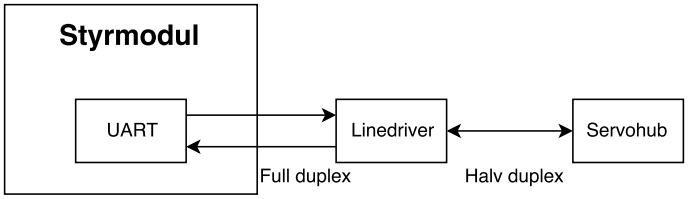


Figur 9: Visualisering av skift-register för SPI. [11]

I figur 9 ser man en bild på hur datautbyte med SPI fungerar. Man skiftar ut en byte från master till en slave och får en byte tillbaka från slaven, så både master och slave har ett skift-register som data skiftas ut från och in i samtidigt. Detta sker i omgångar av en byte. För att reagera på att mastern har skrivit så används avbrott i slave-modulerna som triggas när en byte har skrivits till skiftregistret. Då informationen och kommandona som skickas mellan de olika enheterna skiljer sig åt så tas det i modulernas egna sektioner.

4.2.3 UART

För att styra robotens servon, som är av typen dynamixel AX-12A, så använder sig styrenheten av ett UART-protokoll samt en line-driver som gör om den fullt-duplexa UART från ATmega1284 till en halvt-duplex UART som benens servon använder. Detta illustreras i figur 10. Kommandon och information som skickas över UART beskrivs närmare i sektion 6 Styrmodul.



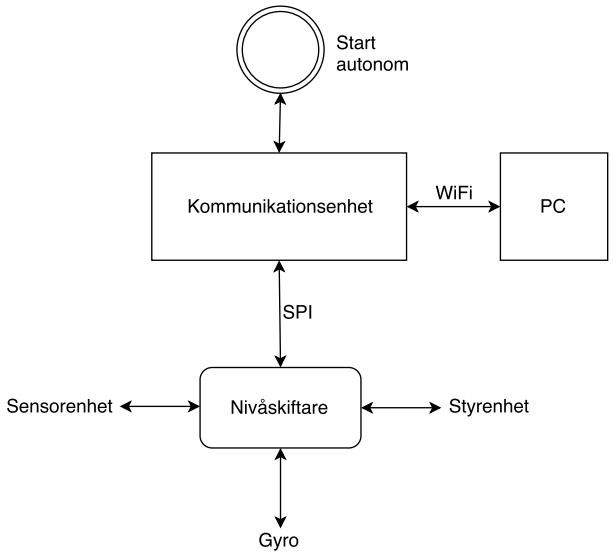
Figur 10: Kommunikation mellan styrmodul och servo.

5 Kommunikationsmodul

Huvudenheten för kommunikation på roboten består av en Raspberry Pi 3 som hanterar de mer prestandakrävande momenten för roboten, som t.ex. algoritmberäkningarna för att navigera i labyrinten. Man kan se denna enhet som ”hjärnan” för roboten och den används för att behandla extern kommunikation med andra enheter samt intern kommunikation mellan olika enheter i systemet. Kommunikationsmodulen är kopplad mot gyro, sensor- och styrmodul med en SPI-buss. På SPI-bussen agerar kommunikationsmodulen *master* medan de övriga enheterna blir *slaves*. Detta innebär att det är kommunikationsenheten som bestämmer vilken modul som ska skicka och ta emot information samt sätter bussens klockfrekvens. Kommunikationsmodulen kommunicerar också till en dator via WiFi och den kan ta emot samt skicka data till en GUI-applikation på den anslutna datorn. I figur 11 kan man få en enkel överblick över delsystemet med kommunikationsmodulens kopplingar till datorn, styr- och sensormodulen samt de knappar som används.

5.1 Översiktligt blockschema

Nedan i figur 11 ser man ett översiktligt blockschema för delsystemet.



Figur 11: Blockschema kommunikationsenhet

5.2 Hårdvara

Kommunikationsenheten består av en *Raspberry Pi 3 Model B* [9], en nivåskiftare av typen TXB0104 [10] vilken är full duplex nivåskiftare för SPI. Genom denna nivåskiftare kan Raspberry Pi kommunicera med andra enheter i systemet

eftersom att Raspberry pi kräver +3.3V och andra komponenter, t.ex. ATMega1284 [2], kräver +5V. För att inte skada vår Raspberry Pi använder man därför nivåskiftare för att säkerställa att varje enhet får rätt mängd ström. Datorn, däremot, är kopplad till Raspberry Pi via Wifi.

5.2.1 Komponentlista

Komponenterna som kommer användas i kommunikationsenheten är som följer.

Komponent	Antal
Raspberry Pi 3 Model B	1
TXB0104	1
Knapp	1

5.3 Kommunikation

För att använda Raspberryn så behövs ett operativsystem och det vi använder oss av heter *Raspian* som är ett speciellt framtaget Linuxsystem för Raspberry Pi. För att kunna kommunicera med de andra enheterna så har vi inkluderat biblioteken *Spidev* som ger oss kontroll över Raspberryns SPI-buss samt *Socket* som låter oss programmera sockets för kommunikationen över ethernet eller Wifi anslutningar. Vid start så kör Raspberryn två Python skript, ett som startar en socket server som väntar på en anslutning från GUI samt huvudprogrammet som innehåller det manuella och autonoma läget för roboten. Detta huvudprogram har access till alla delskript som behövs för att styra bl.a. spi-bussen samt regleringsberäkningar. Den startar även upp wifi-access punkten som sänder ut en wifi-signal med ett SSID vid namn *Brobot*. För att hantera kommunikation med flera enheter samtidigt använder sig kommunikationsmodulen av multitrådning, där varje utbyte görs på en egen tråd.

5.3.1 Socket server

Socket-skriptet skapar en *socket*, en åtkomstpunkt/kommunikationsport, som på lägsta möjliga nivå skapar en port som man kan ansluta till samt skicka och ta emot information från. Skriptet väntar på att GUI ska ansluta till den socket som servern skapat med ett visst portnamn och kod. Denna socket agerar sedan brygga mellan GUI och Raspberryn och den skickar informationen över den wifi-anslutning som Raspberryn har satt upp.

5.3.2 SPI

SPI-bussen styrs från kommunikationsmodulen som agerar master över de andra modulerna som alla agerar slaves. Detta betyder att kommunikationsmodulen själv bestämmer vilken modul den vill prata med genom tre olika Slave-select (SS) signaler, en för varje ansluten slave-modul, sensor- styr- och gyromodulen. Ingen av dessa signaler behöver kopplas med en nivåskiftare då 3.3V tolkas som en logisk etta i både styrenheten och sensorenheten. Kommunikationsmodulen

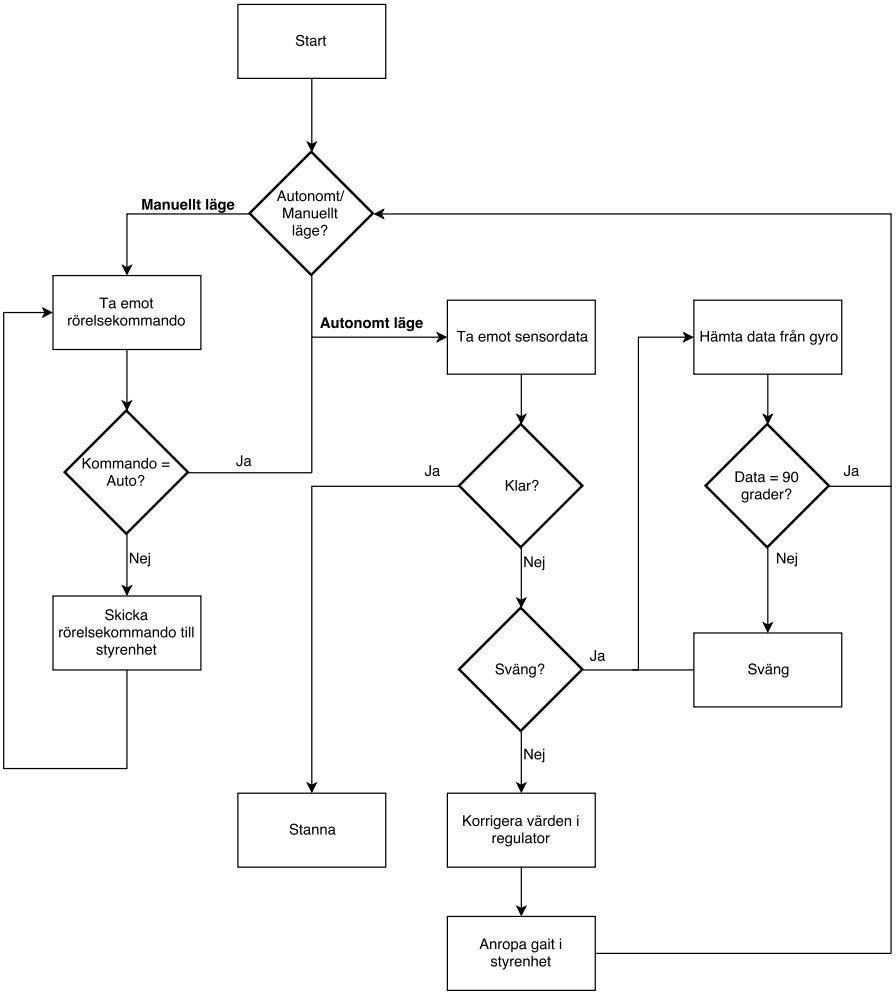
styr också bussens hastighet. Från SPI-bussen hämtar kommunikationsmodulen kontinuerligt in data från sensorenhet och gyro och skickar ut rörelsekommandon samt reglerdata till styrenheten.

5.4 Reglering

Reglering av robotens rörelse sker med en modifierad P-regulator (se sektion 3.2) som justerar robotens avstånd och vinkel mot en eller två väggar på höger respektive vänster sida beroende på situation. Eftersom GP2Y0A02YK-sensorerna ger noll i utslag både då de är för nära en vägg och då de inte har något framför sig skapar vi en matris för de senaste tio sensorvärdena. Modulen kollar sedan på medelvärdet av dessa och beslutar om en nolla innebär att vi inte har något framför roboten eller om den är för nära en vägg. Detta sensorvärde ersätts då med gränsvärdena 20 om den är nära en vägg eller 150 om den inte ser något framför sig. Utifrån detta tar den sedan ett beslut om den ska reglera mot höger, vänster eller båda sidor. Om roboten bara har en vägg åt vänster kommer den reglera mot vänster, om den bara har en vägg mot höger så kommer den reglera mot höger och om den har väggar på båda sidor så kommer den att reglera mot både höger och vänster.

5.5 Mjukvara

Kommunikationsmodulen arbetar mot en huvudloop som kommunicerar med en användardator eller GUI. Det finns två olika lägen som roboten kan befinna sig i: manuellt läge eller autonomt läge. I manuellt läge tar kommunikationsmodulen emot rörelsekommandon från GUI och skickar dessa vidare till styrmodulen som beräknar och utför önskad rörelse. I autonomt läge startas ett program som får roboten att börja navigera en labyrint konstruerad enligt given banspecifikation (se appendix C).



Figur 12: Flödesschema kommunikationsmodul.

I figur 12 ser vi flödesschemat för huvudloopen. I manuellt läge tar kommunikationsmodulen emot ett kommando som kan skickas vidare till styrmodulen eller kan byta till autonomt läge. I autonomt läge utgörs av algoritmen som roboten använder sig av när den navigerar i banan. Roboten kollar först och främst om den har kommit ut ur labyrinten, samtliga sensorer ska då ge största möjliga värde, dvs. 150, eftersom det inte längre finns några väggar runt roboten. Roboten ska då stanna och inte fortsätta röra sig. Om robotens främre högersensor indikerar att det inte finns någon vägg åt höger och roboten kan se att det vägen framåt tar slut, dvs. framsensorns värde är mindre än 150, så kommer roboten att svänga höger och det fungerar på samma sätt åt vänster. När roboten har valt att svänga kopplas regleringen bort och roboten fortsätter att svänga tills gyrot säger att den har vridit sig 90 grader. Om ingen sväng

ska utföras fortsätter roboten framåt. Genom att enbart tillåta svängar om den får sensorvärden större än 80 cm undviker man att roboten svänger in i en återvändsgränd. Detta sammanfattas i tabellen nedan:

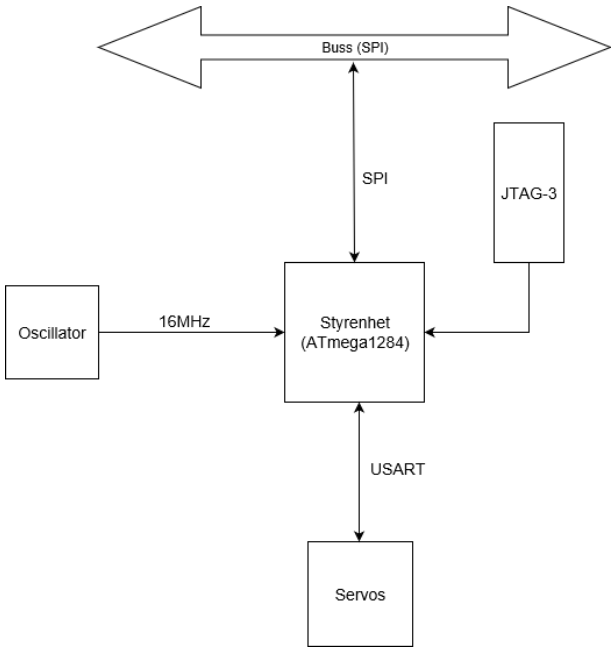
Situation	**Konsekvens**
Inga väggar runt robot	Stanna
Vägg framåt och ingen vägg till vänster	Sväng vänster
Vägg framåt och ingen vägg till höger	Sväng höger
Annars	Fortsätt framåt med reglering

6 Styrmodul

Styrmodulen har som uppgift att hantera rörelse för roboten. När ett rörelsekommando tas emot från kommunikationsmodulen så ska styrmodulen anropa benens servon för att producera önskad rörelse hos roboten. Roboten ska vid start ställa sig upp och sedan kunna röra sig framåt, bakåt samt diagonalt åt både höger och vänster. Den ska även kunna vända sig om 90 grader på en viss punkt åt höger och vänster. Robotplattformen som används har fyra ben uppdelade i tre leder; Coax, Femur och Tibia som skiljs åt av tre servon. Sammanlagt har plattformen tolv stycken servon. Ett servo av typen AX12-A kan rotera från 0 till 300 grader. De servon som sitter på robotplattformen har dock en viss offset, p.g.a. monteringsvinklar och liknande som man måste ta hänsyn till när man vill ställa in ett servo i en viss vinkel.

6.1 Översiktligt blockschema

I figur 13 kan man få en enkel överblick av styrenheten.



Figur 13: Överblick av styrenhet

6.2 Hårdvara

Styrmodulen består av en ATmega1284, en 8-bitars AVR mikrokontroller som är kopplad till övriga enheter i systemet med en SPI-buss. Via SPI kan styrmodulen ta emot kommandon och reglerdata från kommunikationsmodulen. AVR-kortet

kopplas i en daisy-chain med en ATMEL-ICE [3] för att den ska kunna programmeras i Atmel-studio. Kommunikationen med servon i benen sker över en halv-duplex UART, då detta är den kommunikation som stöds av Dynamixel AX12-A [4]. En line-driver av typ 74ls241 [1] kopplas som tri-state buffer mellan servona för att styra kommunikationsvägen och en AVR-kortet kopplas till en extern klocka EXO3 [5] på 16MHz för att få rätt frekvens på 1MBaud i kommunikation med servona. Reset-porten kopplas till en extern knapp som kan användas för att återställa enheten.

6.2.1 Komponentlista

Komponenterna som används i styrmodulen är dessa.

Komponent	Antal
ATmega1284	1
JTAG Ice 3	1
EXO3	1
74LS241	1
AX-12A (servon)	12
Avstudsad knapp	1

6.3 Kommunikation

6.3.1 SPI

I styrmodulen finns funktioner för att konfigurera AVR-kortet och för att hantera kommunikation via SPI och UART. I fallet med SPI ska styrmodulen förhålla sig som en slave till kommunikationsmodulen som är master i vårt system. Styrmodulen ska kontinuerligt uppdatera aktuellt rörelsekommando och reglerdata. När data skickas från kommunikationsmodulen triggar detta ett avbrott i styrmodulen som uppdaterar två globala variabler för vilket kommando som ska utföras och vilken reglerdata som ska användas i rörelsen. När data har tagits emot av enheten skickar den tillbaka ett konstant värde ACK = 0x0A för att visa att givna data har tagits emot.

6.3.2 UART

UART-kommunikation med AX12-A dynamixel servon måste också konfigureras. Eftersom halv-duplex UART använder samma sladd för kommunikation i båda riktningar. Mikrodatorn saknar dessa egenskaper och förutsätter dubbelriktad kommunikation (full duplex UART). För att lösa detta måste vi bestämma vilken enhet som ska sända eller ta emot data. Detta gör vi genom att skicka en riktningssignal till vår tri-state buffer och vi kallar dessa olika lägen för TRANSMIT-MODE och RECIEVE-MODE. Samtliga servon i robotplattformen är kopplade till en servohub i mitten av roboten. Detta innebär att alla meddelanden man skickar ut via UART alltid tas emot av alla servon.

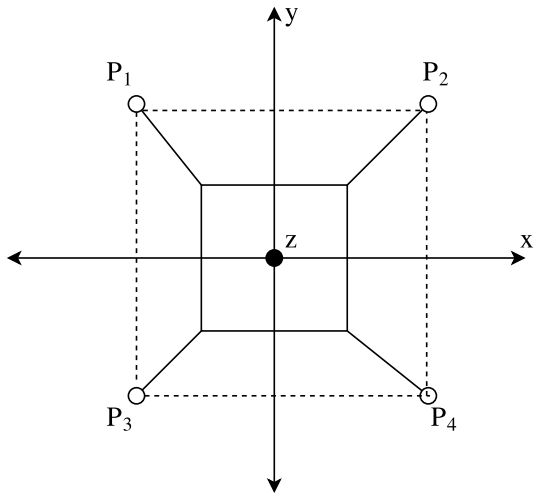
Varje servo har dock ett hårdkodat ID och behandlar endast instruktioner som är adresserade till sig.

0xFF	0xFF	ID	LENGTH	INSTRUCTION	PARAMETERn	CHECKSUM

Först skickas två start-bytes 0xFF följt av servo ID, adress, instruktion, längd, parametrar och sekvensen avslutas med en Checksum-byte som är summan av samtliga bytes efter servo-ID. För de flesta instruktioner returnerar AX12-A ett statuspaket där man kan läsa av status för servot som adresserats. Speciellt viktigt är att man med detta kan läsa av om något fel har skett i servot genom att kontrollera dess Error-byte för att se vilken typ av fel som har skett. I styrenheten utnyttjar vi oss framförallt av instruktionen WRITE för att skriva positioner som vi vill att ett servo ska anta och READ för att läsa registerdata i servon. Men det finns även stöd för ett antal fler kommandon vid kommunikation med AX12-A som man skulle kunna använda sig av om man har behov för det. För en mer ingående beskrivning av Dynamixel AX-12A hänvisas läsaren till dess datablad. [4]

6.4 Rörelse

Vid rörelse av roboten utgår man från ett koordinatsystem med origo i centrum av roboten. Genom att använda dessa punkter i formler för invers kinematik (se sektion 3.1) kan man beräkna vinklarna som varje led i servo behöver ställa in sig i för att nå punkten.



Figur 14: Koordinatsystem för robot sett från ovan.

I figur 14 ser man roboten i dess startposition. Fotspetsarna bildar då en kvadrat och roboten står stabilt. I samtliga rörelser kan man korrigera robotens höjd från marken, steghöjd, steglängd och bredd mellan benen. Det är dock viktigt att förhållandet mellan punkterna förblir detsamma för att inte rörelsen ska bli instabil.

6.4.1 Servorörelse

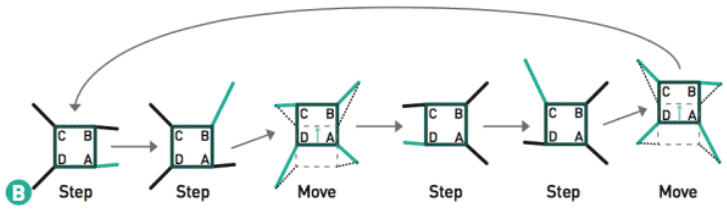
När man vill förflytta ett servo i ett ben till en viss vinkel behöver man ta hänsyn till en offset som uppstår pga. monteringsvinklar och liknande. För att generera en jämn rörelse tar styrmodulen hänsyn till servots nuvarande vinkel och beräknar därefter en lämplig hastighet. Detta gör det möjligt att synkronisera förflyttningar så att rörelser i ett ben med flera servon avslutas samtidigt. Om servot står i sitt ursprungsläge rör det sig med en standardhastighet på 200 och sedan kan den röra sig snabbare respektive långsammare beroende hur långt den är ifrån sitt startläge.

6.4.2 Rörelsekommandon

Styrmodulen har ett antal funktioner för rörelse:

Kommando	Förklaring
FORWARD	Rakt framåt
FORWARD-LEFT	Diagonalt åt vänster
FORWARD-RIGHT	Diagonalt åt höger
BACKWARDS	Rakt bakåt
TURN-RIGHT	Rotation 90 grader åt höger
TURN-LEFT	Rotation 90 grader åt vänster
STOP	Robot ställer sig i startläge

I rörelsesekvenserna FORWARD och BACKWARDS rör sig roboten med en s.k. ”creep gait”. Creep gait är en stabil gångstil där roboten håller sitt masscentrum inom den triangel som formas av de tre ben som ej används. Om dess masscentrum förflyttas utanför detta triangelområde under en alltför lång tid så kommer roboten att kollapsa. I figur 15 ser vi ett exempel på rörelsemönstret som används för creep-gait.



Figur 15: Creep-gait rörelsemönster.[7]

1. Roboten ställer sig i start-position med två ben utsträckta på ena sidan och de övriga invikta på den andra sidan.
2. Det främre högra benet lyfts upp och sträcks ut långt framför roboten.
3. Alla ben förflyttas bakåt och kroppen skjuts framåt.
4. Det bakre vänstra benet rör sig framåt längs kroppen.
5. Det främre vänstra benet lyfts upp och sträcks ut långt framför roboten.
6. Återigen förflyttas alla ben bakåt och kroppen skjuts framåt.
7. Det bakre högra benet rör sig in mot kroppen och tar oss tillbaka till startpositionen i steg 1.

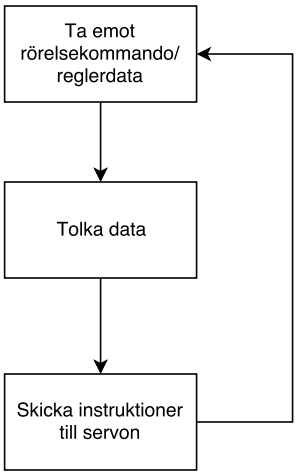
Vid rotation med en viss vinkel, som i TURN-RIGHT och TURN-LEFT, multipliceras benens nuvarande koordinater med en rotationsmatris och en viss vinkel θ som man vill vrida roboten med.

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Detta utnyttjas även i regleringen av roboten då man kan korrigera gaiten och få roboten att styra upp om den avviker från mitten av banan. Till skillnad från den vanliga gaiten så är FORWARD-LEFT och FORWARD-RIGHT speciellt designade för att gå i 45 graders riktning åt höger eller vänster och kan inte regleras.

6.5 Mjukvara

Styrmodulen har ingen komplex huvudloop utan tolkar endast indata, gör beräkningar och skickar instruktioner till servon.



Figur 16: Loop för styrenhet

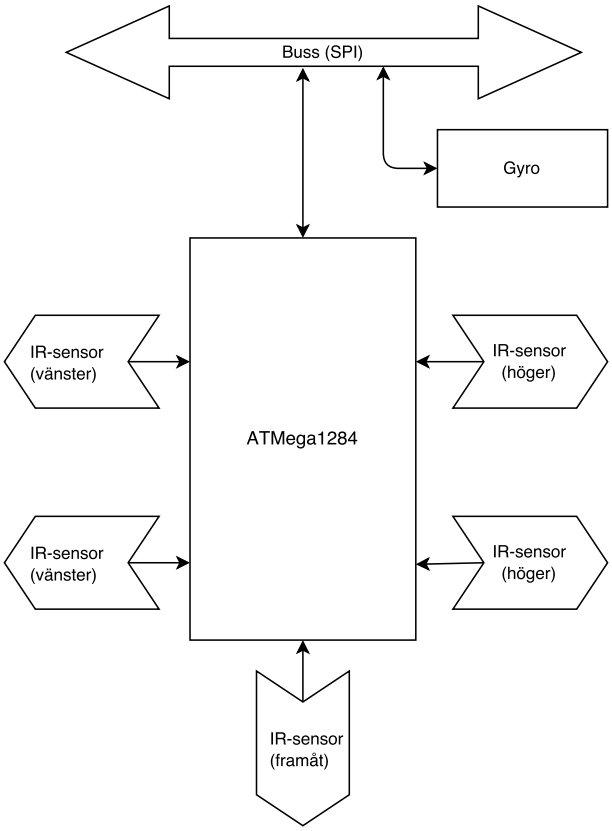
I figur 16 ser man hur styrmodulen arbetar. Vid SPI-avbrott uppdateras två globala variabler, kommando och reglerdata. I huvudloopen tolkas denna data och styrmodulen startar det rörelsekommando som den har fått in alternativt korrigerar data som behandlas i en redan startad sekvens. Modulen utför sedan beräkningar för servovinklar och skickar dessa till rätt servo, vilket leder till önskad rörelse.

7 Sensormodul

Sensormodulen ger roboten en uppfattning om dess omgivning genom att polla värden från de anslutna IR-sensorerna, dessa värden måste sedan A/D-omvandlas så att de kan beräknas om till distanser. Denna information skickas sedan vidare till kommunikationsmodulen över SPI-bussen när den blir givet ett kommando.

7.1 Översiktligt blockschema

I figur 17 kan man få en enkel överblick av sensormodulen.



Figur 17: Överblick av sensorenhet

7.2 Hårdvara

Sensormodulen består av en ATmega1284, på samma sätt som styrenheten, och en ATMEL ICE används för att programmera ATmega1284 i Atmel-studio. Enheten tar in data från ett antal sensorer av typ GP2Y0A02YK samt ett

gyro MLX90609. Den är kopplad till samma klocka EXO3 som styrmodulen och använder samma reset-knapp. En mer ingående beskrivning av komponenternas egenskaper finner man under respektive sektion nedan.

7.2.1 Komponentlista

Komponenterna som används i sensorenheten är dessa.

Komponent	Antal
ATmega1284	1
JTAG Ice 3	1
EXO3	1
GP2Y0A02YK	5
MLX90609	1
Avstudsad knapp	1

7.3 Kommunikation

7.3.1 SPI

Sensormodulen kommunicerar via SPI med kommunikationsmodulen. SPI-bussen skapar ett avbrott i sensormodulen som stoppar den från att hämta värden från IR-sensorerna och skickar istället sensordata till kommunikationsmodulen om ett korrekt kommando skrivits till buffern, kommandot är 0x02 i hexadecimal bas. Detta betyder att en korrekt efterfrågan av sensordata är ett kommando 0x02 följt av fem inläsningar, en för varje sensor. Vid ett inkorrekt givet kommando så ignorerar sensormodulen given data och skickar inte heller någon sensordata.

7.4 Gyro

Gyrot används för att försäkra att en 90 graders sväng har utförts och den kommunicerar med kommunikationsmodulen genom SPI-bussen där den agerar slave. För mer genomgående beskrivning av gyrot och vilka kommandon den använder så hänvisar vi till gyrots egna datablad. [8]

7.5 IR-Sensor

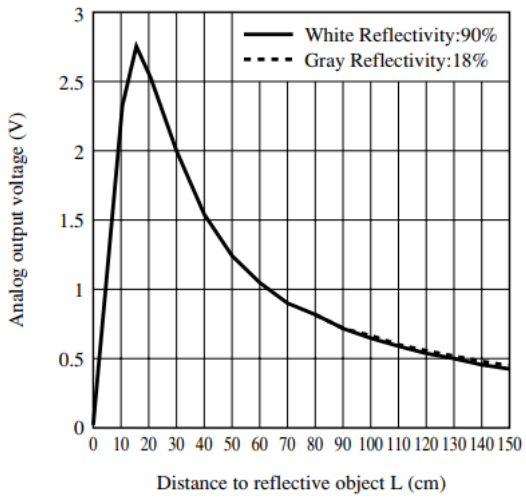
Roboten använder sig av fem stycken IR-sensorer med en placerad framåt, två åt höger samt två åt vänster, detta ger oss möjligheten att göra vinkelberäkningar till väggarna utöver att beräkna distansskillnaden mellan sidorna. IR-sensorerna kan avläsa distanser mellan 20 och 150 cm. Därför kan vi avgöra vilket håll den ska gå åt i en trevägskorsning, eftersom en återvändsgränd enligt banspecifikationen endast får vara 80 cm djup. För mer avancerad information av hur IR-sensorerna funkar själva så hänvisar vi till deras datablad. [6]

7.6 Gränssnitt

Mätvärden från samtliga IR-sensorer skickas till AVR-kortet som analoga volt-signaler där volt-påslaget motsvarar en distans, de filtreras separat med varsitt lågpassfilter och A/D-omvandlas på ATmega1284. Denna information skickas sedan vidare till kommunikationsenheten när kommando om det kommer på SPI-bussen. Gyrot är direkt kopplad till SPI-bussen och efterfrågas direkt av kommunikationsenheten.

7.7 A/D-omvandling

Processorns A/D-omvandlare tar in en analog spänning och omvandlar det till ett 10-bitars tal genom successiv approximation. Man väljer vilken insignal som ska tolkas med hjälp av en mux som har åtta ingångar att välja mellan. För att tolka volstyrkan på den ingående signalen så måste A/D-omvandlaren ha referenser till 0V och någon positiv volt som då blir 100 procent. För våra ändamål så använder vi en 5-volts referens som ges från virkortets spänning. I nedanstående bild, figur 18, tagen från sensorns datablad, så kan man se hur voltstyrkan motsvara en distans:

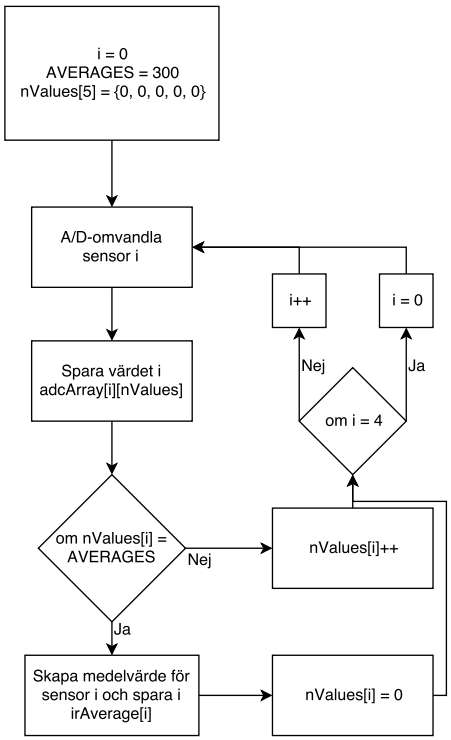


Figur 18: SHARP sensorns analoga signal samt distansen den motsvarar. [6]

7.8 Mjukvara

Sensormodulen arbetar i en loop som A/D-omvandlar spänningarna från de fem olika sensorerna i tur och ordning. Data från A/D-omvandlaren tolkas sedan om från ett digitala spännings-tal till cm. Informationen sparas sedan i en lista av listor med ett flertal värden från de olika sensorerna, när tillräckligt många värden har sparats så beräknas ett medelvärde som tar bort påverkan som

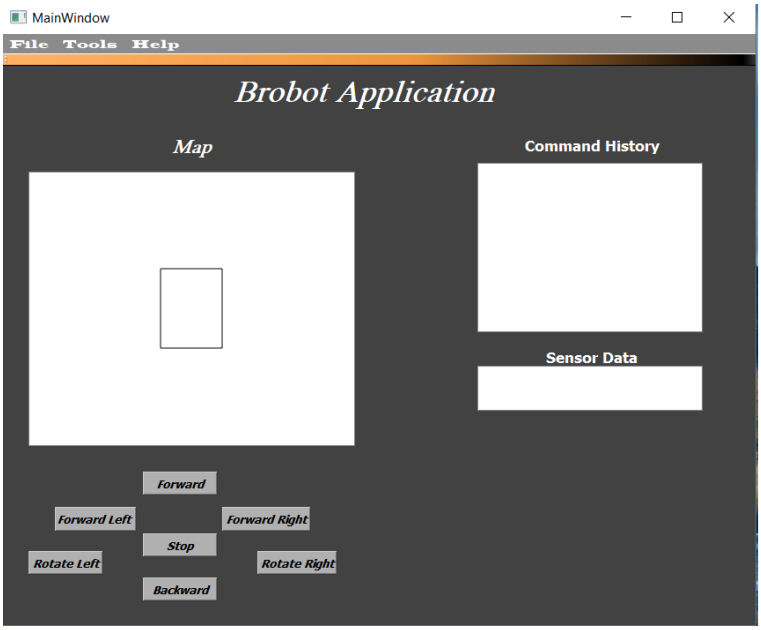
störningar i signalerna skulle kunna orsaka på individuella mätningar. Dessa värden sparas sedan i en lista med medelvärden som också är listan som skickas till kommunikationsmodulen när den får kommandot. Nedan, i figur 19, följer en simpel visuell representation av loopen:



Figur 19: Sensormodulens loop

8 PC

Roboten kan kontrolleras från ett GUI på användarens dator. I detta interface kan man interagera med roboten genom att skicka kommandon och ta emot sensordata.



Figur 20: GUI till systemet.

Man kan i figur 20 se hur GUI för systemet ser ut. Tabellen nedan visar vad varje knapp motsvarar för funktionalitet.

1. Map: Visar en bild på robotens läge.
2. Styrknappar: Varje knapp svarar mot ett rörelsekommando för roboten
3. Kommandohistorik: Här kan man se vilka tidigare kommandon som har givits till roboten.
4. Sensordata: Visar mätvärden från robotens sensorer.

8.1 Mjukvara

Datorns mjukvara för kontroll av Brobot är skrivet i C++ med hjälp av utvecklingsmiljön QtCreator. QtCreator valdes då den innehåller färdig fönsterhantering samt tillåter visuell konstruktion av fönsterna. För att starta GUI så krävs endast att man har den exekverbara filen på sin dator. Vi har utvecklat systemets GUI på Windows version av QtCreator men den kompilerade filen gör ingen skillnad på OS, den kräver endast en Wifi-anslutning till Brobot för att kunna ansluta till dess server-socket.

8.2 Kommunikation

På datorns sida så skapar GUI en client-socket som ansluter till Raspberry Pi server-socket över Wifi-anslutningen. GUI kräver alltså att datorn är ansluten med WIfi till kommunikationsmodulen innan kommunikation kan upprättas.

9 Slutsatser

Vi har byggt en vandrande 4-bent robot som kan utföra en rad uppgifter både autonomt och manuellt. Även om systemet fungerar så finns det många förbättringar som skulle kunna göras; autonomt läge skulle kunna utökas för att hantera mer avancerade labyrinter. Det finns även ett par mindre fel som man skulle kunna lösa, t.ex. felmarginaler vid kommunikation med SPI. Exempel på förbättringar som skulle kunna göras:

1. Utveckling av autonomt läge: Med en bättre labyrintalgoritm skulle roboten kunna hantera en mer komplex bana med t.ex. trevägskorsningar med mer än en möjlig väg eller fyrvägskorsningar.

2. Styrning från app: Hanteringen av Wifi-kommunikation gör det möjligt att ansluta andra enheter än endast en laptop. Kontroll av robot skulle t.ex. kunna ske från ett GUI på en telefon.

3. Utveckling av GUI: En mer avancerad GUI-design, med ökad funktionalitet skulle kunna ge en användare mer information om hur roboten rör sig och t.ex. rita upp en karta av vägen den navigerar i labyrinten.

4. Utveckling av gait: En möjlig förbättring skulle kunna vara en mer komplex rörelsesekvens som t.ex. Trot gait, som ger en snabbare och mer smidig rörelse än creep-gait.

5. Utveckling av sensoranvändning: En utökad användning av sensorer skulle kunna användas för att hantera hinder i banan eller t.ex. följa en tejpad linje.

Referenser

[1] *74ls240*. MOTOROLA. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/sn74ls240.pdf.

[2] *ATmega1284*. Atmel. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/atmega1284p.pdf.

[3] *ATMEL-ICE*. ATMEL. URL: http://www.atmel.com/Images/Atmel-42330-Atmel-ICE_UserGuide.pdf.

[4] *AX-12*. Dynamixel. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/AX-12.pdf.

[5] *EXO-3*. ELFA. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/exo3.pdf.

[6] *GP2Y0A02YK*. Sharp. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/gp2y0a02_e.pdf.

[7] Oscar Liang. *Inverse Kinematics Basics Tutorial*. Jan. 2012. URL: https://oscarliang.com/inverse-kinematics-and-trigonometry-basics/.

[8] *MLX90609*. Melexis. URL: https://docs.isy.liu.se/pub/VanHeden/DataSheets/MLX90609_datasheet.pdf.

[9] *Raspberry Pi manual*. Raspberry Pi Foundation. URL: https://www.raspberrypi.org/documentation/.

[10] *TXB0104 Bi-Directional Level Shifter*. Adafruit. URL: https://cdn-shop.adafruit.com/datasheets/txb0104.pdf.

[11] Elliot Williams. *What Could Go Wrong: SPI*. Juli 2016. URL: https://hackaday.com/2016/07/01/what-could-go-wrong-spi/.

Appendix

A Detaljerat kopplingsschema

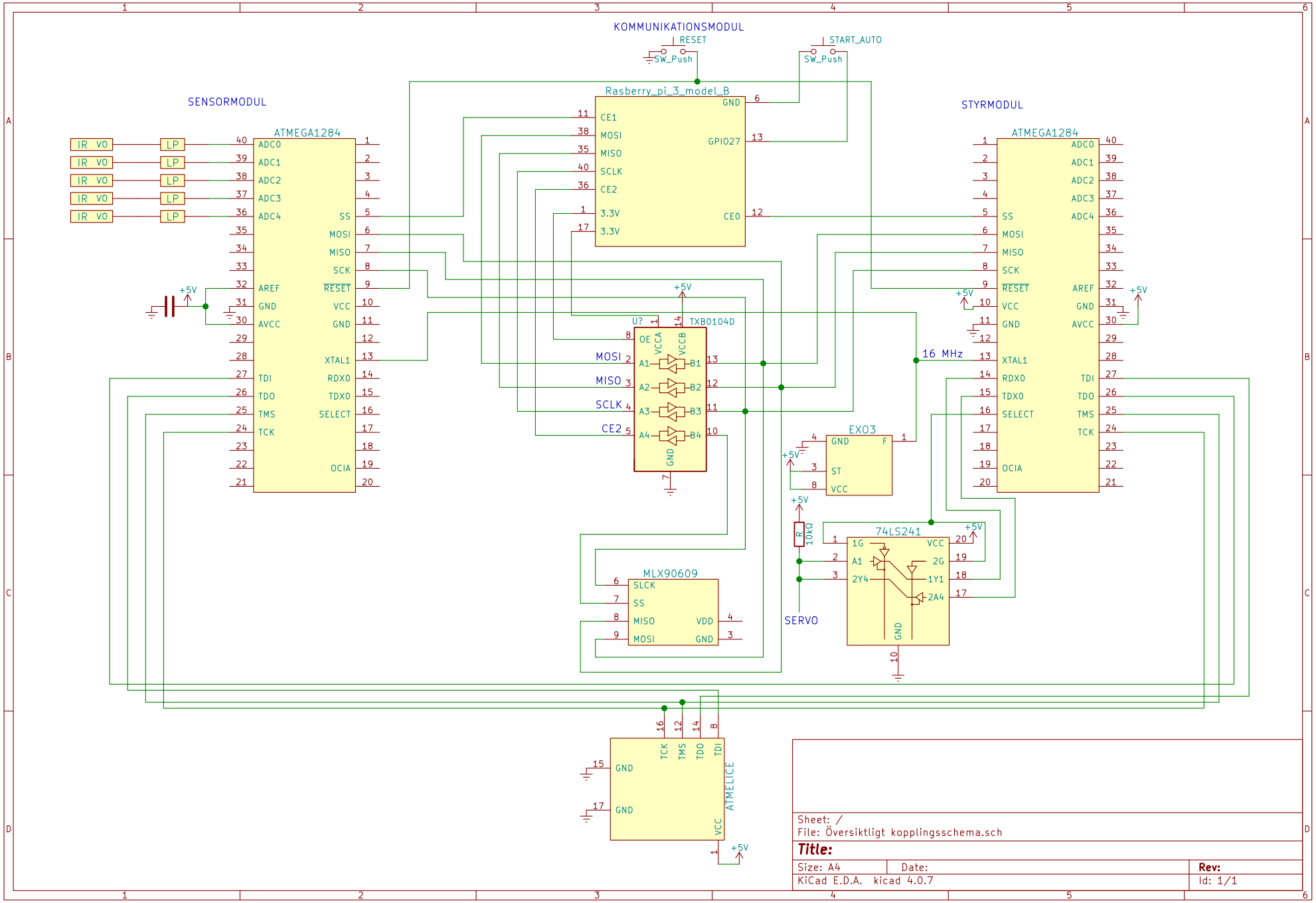
Detaljerat kopplingsschema över hela systemet kan ses på nästa sida.

B Programlistning

Mjukvara till systemet bifogas i separat arkivfil.

C Banspecifikation

Banan ska bestå av en enkel labyrint med kartongväggar. Avståndet mellan väggarna är 80 cm. Banan har endast 3-vägskorsningar med återvändsgränder. Återvändsgränderna är 80 cm djupa. Banan ryms inom ett område på maximalt 10*10 meter. Roboten startar i ett hörn och målet är när den kommit ur labyrinten, dvs. när den inte har några väggar kring sig varvid den ska stanna.


KOMMUNIKATIONSMODUL
RESET
SW_Push
START_AUTO
SW_Push
Rasberry_pi_3_model_B
SENSORMODUL
STYRMODUL
ATMEGA1284
ATMEGA1284
IR VO
LP
+5V
U?
TXB0104D
MOSI
MISO
SCLK
CE2
16 MHz
EXO3
74LS241
MLX90609
SERVO
10kΩ
ATMELICE
Sheet: /
File: Översiktligt kopplingsschema.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 4.0.7
Id: 1/1