Teknisk Dokumentation

Version 1.0

Status

Granskad	Jennifer Santos	2023-12-11
Godkänd	Anders Nilsson	2023-12-19

PROJEKTIDENTITET

G06, HT2023, charty
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Hugo Nilsson	kundansvarig (KUN)	073-429 33 26	hugni385@student.liu.se
Jennifer Santos	dokumentansvarig (DOK)	070-863 59 08	jensa682@student.liu.se
Edvard Wetind	designansvarig (DES)	072-717 51 15	edvwe024@student.liu.se
Elin Rydebrink	testansvarig (TST)	070-315 69 83	eliry213@student.liu.se
Elliot Norlander	kvalitetssamordnare (QS)	070-719 90 17	ellno907@student.liu.se
Jacob Sjölin	implementationsansvarig (IMP)	070-861 16 57	jacsj573@student.liu.se
Elliot Norlander	projektledare (PL)	070-719 90 17	ellno907@student.liu.se

Kursansvarig: Anders Nilsson, anders.p.nilsson@liu.se

Kund: Anders Nilsson, anders.p.nilsson@liu.se
Handledare: Theodor Lindberg, theodor.lindberg@liu.se

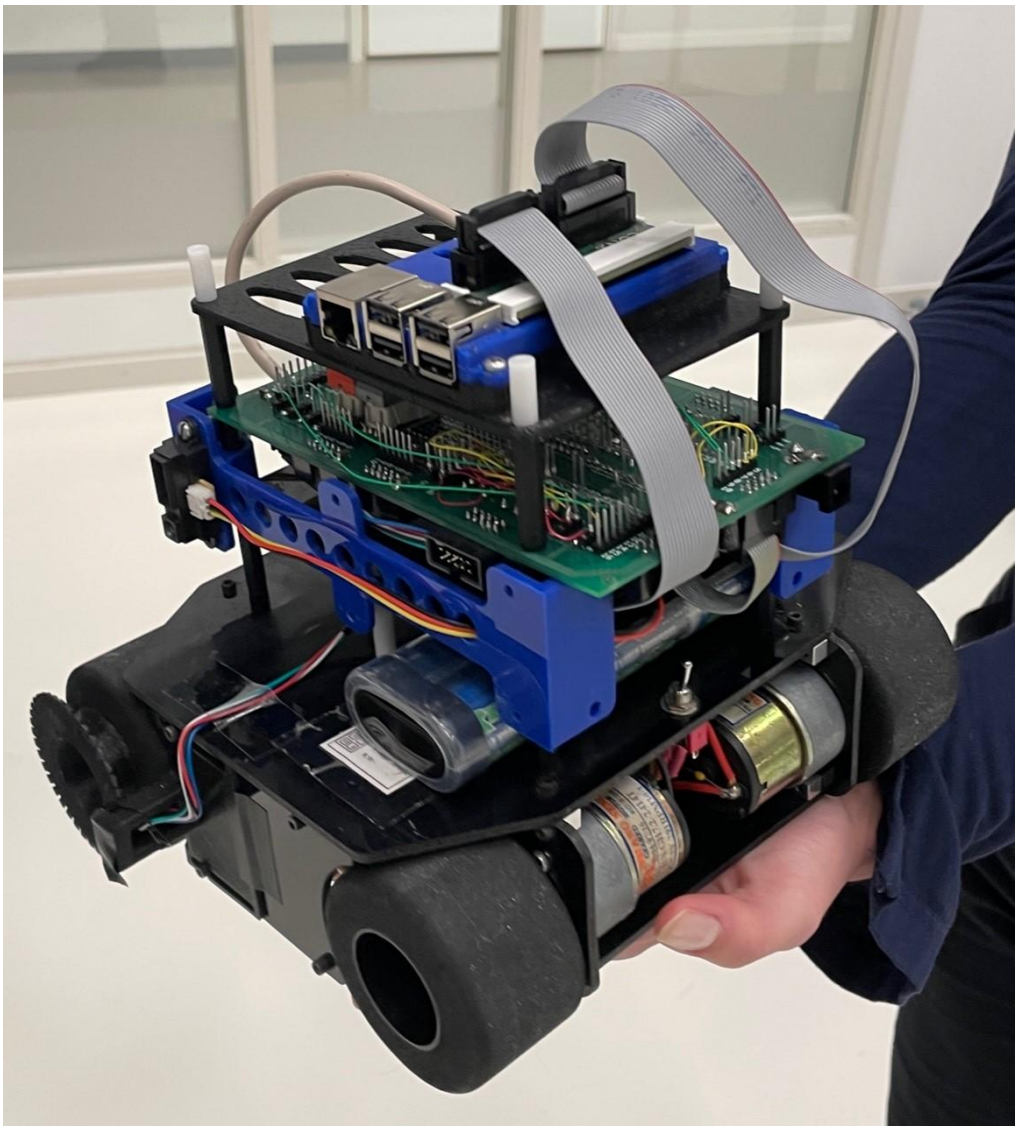
1. Inledning

Projektet som beskrivs i följande sidor utfördes som en del av kursen *Konstruktion med Mikrodatorer* (TSEA29) under den femte terminen av civilingenjörsprogrammet inom datateknik vid Linköpings universitet. Studenterna delade in sig i grupper om maximalt sex personer, varefter varje grupp fick ett särskilt uppdrag att fullfölja. Projektet som utfördes av grupp G06 var att konstruera en robot med självkörande förmåga som även kunde kartlägga världen runt omkring sig. Syftet med projektet var att tillämpa de kunskaper och färdigheter som tidigare lärts ut i kurser inom elektronik och hårdvara samtidigt som studenterna fick möjligheten att jobba i enlighet med en vedertagen projektmodell. Modellen som användes var *LIPS* och gav studenterna erfarenhet av hur det kan vara att jobba med projekt inom yrkeslivet efter examen.

2. Produkten

2.1 Produktbeskrivning

Den färdiga produkten är en konstruktion på fyra hjul, varav ett av hjulen är utrustat med en odometer. Odometern är synlig i *Figur 1* på det övre hjulet i det synliga hjulparet. Roboten är även utrustad med tre IR-sensorer, där de två som är placerade på sidorna mäter avstånd inom ett kortare intervall och den som är framåtriktad mäter avstånd som befinner sig större avstånd. Dessa sensorer mäter avståndet till närmsta vägg vilket är absolut nödvändigt för den självkörande funktionen. Vidare finns där även ett gyroskop, beläget under det gröna kortet som syns i *Figur 1*. Gyroskåpet tillgodoser att mjukvaran med data för att räkna ut vinkeln hos en svängning, vilket är nödvändigt för att kunna göra korrekta svängar.



Figur 1. Bild över roboten CHARTY.

2.2 Funktion

Charty är designad för att vara enkel för vem som helst att använda. Anslut till enhetens bluetooth och starta programmet “code/main.py”. Placera Charty på valfri plats på kartongkartan. Tryck på röda knappen som är placerad på det gröna vir-kortet eller starta via gränssnittet. Charty startar av sig själv och börjar kartlägga världen omkring sig. När roboten hittat kartlagt hela kartan, kör den tillbaka till startpositionen.

3. Teori

3.1 Regleringsalgoritm

För att roboten skall kunna reglera sin position i förhållande till väggar på sidorna har vi implementerat PD-reglering. Denna regleringsalgoritm mäter avstånd mellan eventuella väggar och tar dessutom förändringar i vinklar i gyroskopet i beaktning.

Formel enligt följande: u[n] = KP * e[n] + KD * Φ där KP och KD är konstanter, e[n] är skillnaden i avstånd mellan väggar och Φ är vinkelförändring från tidigare mätpunkt.
Konstanterna har justerats för att skapa en rimlig reglering. Resultatet som tillhandahålls från algoritmen ställer in hastigheter för de båda hjulparen.

3.2 Autonom styrningsalgoritm

All kod för den autonoma styrningen och kartan är skriven i python.

3.2.1 Kartan

3.2.1.1 Kartans sturktur

Kartan är ett koordinatsystem i form av en lista med listor för varje rad som representerar y-värdena. Koordinat 0,0 är placerad högst upp i vänstra hörnet och till exempel en 4x4 stor karta kommer ha koordinaterna enligt figur 2.

0,0	1,0	2,0	3,0
0,1	1,1	2,1	3,1
0,2	1,2	2,2	3,2
0,3	1,3	2,3	3,3

Figur 2. Koordinater.

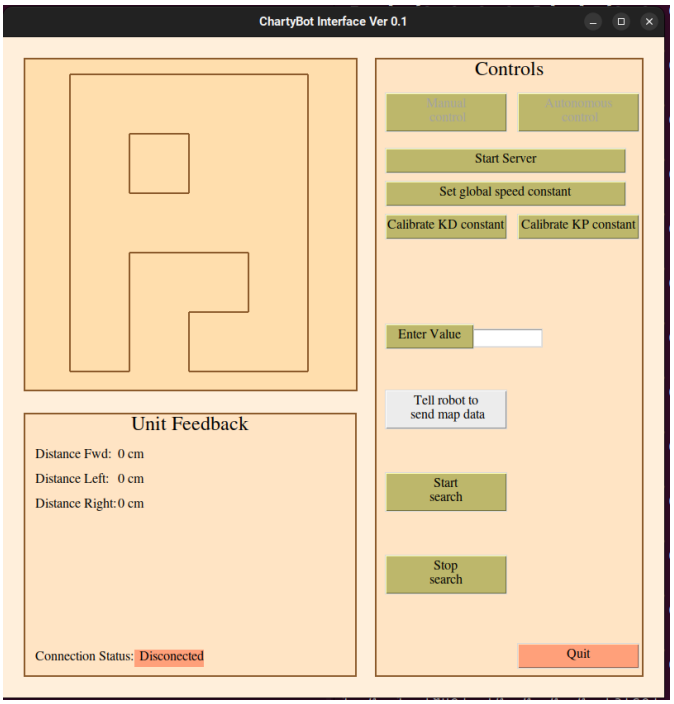
Varje koordinat har en lista med information om hur väggarna runt den rutan ser ut samt en sant eller falskt variabel som talar om huruvida rutan är avsökt eller inte. Listan har fem

element på formen [övre, högra, nedre, vänstra, avsökt] där alla riktningar läses av från kartläsarens perspektiv. Om det står en etta på någon av de fyra första platserna betyder det att det finns en vägg där och en nolla innebär att det inte finns en vägg. Ett exempel på ett 2x2 koordinatsystem som är delvis avsökt är:

Koordinater = [[[0,1,0,1,True], [0,1,1,1,True]],
[[0,0,0,0,False],[0,0,0,0,False]]]

3.2.1.2 Uppritning av kartan

Kartan ritas upp i GUI:t med hjälp av “tkinter” och uppdateras kontinuerligt när roboten kör genom kartan [6]. Datan om hur kartan ser ut skickas från roboten till datorn över bluetooth. Varje ruta på kartan har kvadratisk form vars storlek beror på hur kartan är just då. Om kartan ökar i storlek minskar rutornas storlek så att det ska vara lagom stort i fönstret med en viss marginal runt kartan. Storleken på rutorna begränsas av antingen höjden eller bredden på kartan beroende på vilken av dem som är störst. GUI:t ritar då upp alla väggar på den grafiska kartan utifrån robotens karta, se bild 1.



Figur 1. Gränssnittet.

3.2.2 Algoritmen

3.2.2.1 Söka av första rutan

Det automona styrningsalgoritmen börjar med att läsa av väggarna i den första rutan som har koordinaterna (0,0) genom att använda sensorerna på höger och vänster sida samt sensorn på framsidan av roboten. Roboten mäter sedan av väggen bakom sig genom att vrida sig 90

grader år höger och mäta av sensorn på höger sida. Roboten lägger sen till väggarna och alla grannar i listan och rör sig framåt på samma sätt som i alla andra rutor, se 3.2.2.3-3.2.2.8.

3.2.2.2 Uppdatera x- och y-koordinaterna

För att hålla koll på hur roboten har förflyttat sig används en variabel som vet i vilken riktning roboten står i och vet då hur roboten förflyttat sig i den förra iterationen. Utifrån robotens riktning uppdateras då robotens x- och y-koordinat. Eftersom kartan i slutet av varje iteration utökas efter vilka grannar som finns, hamnar roboten aldrig utanför det satta koordinatsystemet.

3.2.2.3 Lägga till väggar i koordinatsystemet

Roboten söker av med sensorerna på höger och vänster sida samt sensorn som mäter framåt och avgör om det finns väggar på dessa platser. Det antas alltid att det inte finns någon vägg bakom roboten då den inte kan ha en vägg från rutan den körde ifrån. Värdena sparas i en lista som läggs in i koordinatsystemet på x- och y-koordinaten som roboten står på. Om det är en vägg placeras en etta på den motsvarande platsen i listan annars placeras en nolla där. Variabeln som säger om rutan är avsökt eller inte sätts till “True”. Det kan antas att x- och y-koordinaterna finns i koordinatsystemet då rutan lades till som granne i iterationen innan.

3.2.2.4 Lägga till grannar

Om det inte finns väggar vill algoritmen lägga till grannar för att komma ihåg att det är platser som roboten måste besöka. Granne läggs till som en plats i koordinatsystemet med avsökt variabel satt till “False” och alla väggar satta till noll. Om en granne läggs till ovanför eller under koordinatsystemet, dvs att x värdet är för stort eller för litet, så läggs en hel lista till över eller under med alla koordinater i listan satta till [0,0,0,0,False]. Samma sak gäller för om y-koordinaten är för stor eller liten. Då läggs ett element till slutet eller början av alla listor för att fylla ut så koordinatsystemet fortfarande är korrekt. Om det läggs till en granne över eller till vänster om systemet, dvs om x eller y är mindre än noll, måste systemet skiftas så att en ny ruta har koordinaten (0,0) och koordinaterna för rutan som roboten står på ändras. X- och y-koordinaterna för roboten uppdateras då för att de ska ha rätt plats i det uppdaterade koordinatsystemet.

3.2.2.5 Kör till nästa ruta

Om det finns någon granne till robotens nuvarande position så förflyttar sig roboten till en av dem med prioritering höger, fram, vänster. Det betyder att om det finns en ruta till höger som roboten inte har besökt åker roboten dit genom att rotera 90 grader åt höger och sedan köra framåt i en rutas längd. Om roboten inte kan eller behöver åka till höger men kan åka rakt fram så kör den framåt en ruta. Om den inte kan göra något av det men det kan köra åt vänster så roterar roboten 90 grader åt vänster och kör sedan framåt i en rutas längd. Om inget av detta är möjligt följer roboten vägen tillbaka, se 3.2.8

3.2.2.6 Lägg till vägen tillbaka i en lista

Om roboten förflyttat sig till en ny ruta läggs riktningen som roboten måste köra tillbaka till i en lista. Det är då motsatt riktning till den som roboten precis körde.

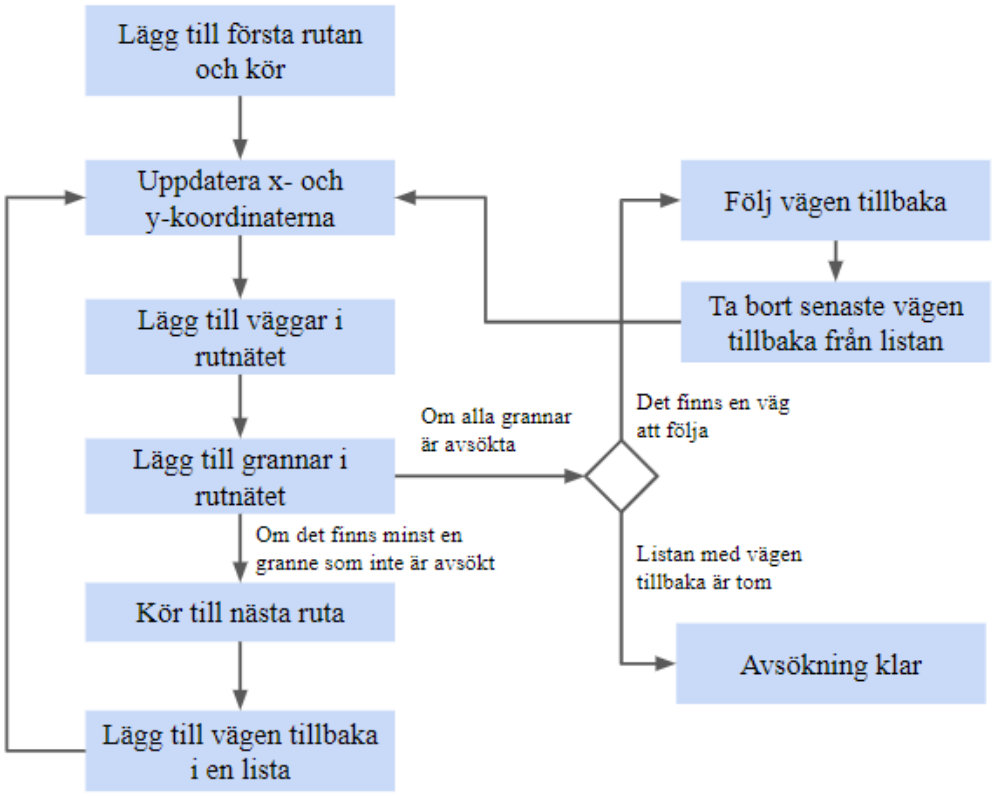
3.2.2.7 Följ vägen tillbaka

Om det inte finns någon granne att besöka vänder sig roboten så att den står i riktningen som ligger sist i listan, dvs motsatt riktning som roboten körde för att komma till den nuvarande rutan, och kör en rutas längd framåt. Den tar sedan bort det senaste elementet i listan.

3.2.2.8 Avsökning klar

Om det inte finns några grannar att besöka och att listan med vägen tillbaka är slut så betyder det att hela kartmiljön är avsökt. Då avslutas programmet.

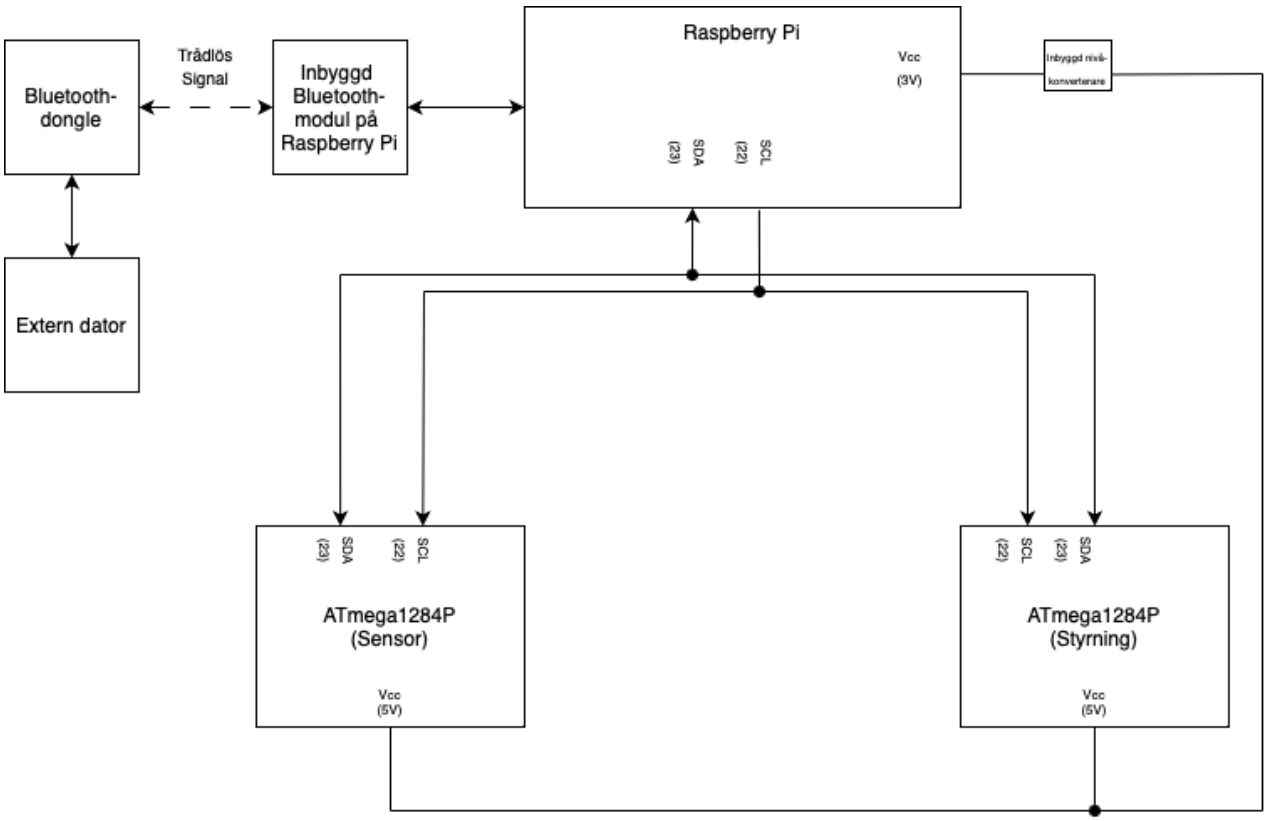
3.2.3 Blockschema för den autonoma styralgoritmen



Figur 3. Blockschema för autonoma styralgoritmen.

4. Systemet

Systemet består av fyra moduler: kommunikation, styrning, sensor och centralenhet, se figur 4. Koden är skriven i två olika språk: Python och C.



Figur 4. Blockschema för systemet.

5. Modulerna

Innehåller mer detaljerade blockscheman och beskrivningar av varje modul.

5.1 Kommunikationsmodulen

Kommunikationsmodulens syfte är att skicka och motta information mellan de olika delsystemen.

5.1.1 Trådlös kommunikation

Bluetooth används för att kommunicera med datorn och Raspberry-Pi. Detta har implementerats i Python med paketet “socket” [1]. Datan som skickas sinsemellan är strängar som skickas byte för byte. Servern skickar strängar som delas upp i två delar: ID och data enligt tabell 1. Beroende på vilket styr-läge roboten har kommer det skickade ID:t tolkas olika. Klienten skickar enbart kartläggningen.

GUI-knapp	Styr-läge	ID
Stopp	Manuellt	0
Kör fram	Manuellt	1
Kör bak	Manuellt	2
Kör höger	Manuellt	3
Kör vänster	Manuellt	4
Styr-läge till autonomt	Manuellt	5
Kör fram och rotera höger	Manuellt	6
Kör fram och rotera vänster	Manuellt	7
Uppdatera karta	Autonomt	0
Starta kartläggning	Autonomt	1
Avsluta kartläggning	Autonomt	2
Styr-läge till manuellt	Autonomt	3
Ändra Kp	Autonomt	4
Ändra Kd	Autonomt	5
Ändra avståndsförskjutning	Autonomt	6

Tabell 1. Signaler från servern.

5.1.1.1 Server och klient

För att skapa en sambindning mellan datorn och raspberry:pi finns två separata Python-filer: “server” och “client”, som körs av antingen på datorn eller raspberry-Pi. Server-sidan kommer ta emot socket-information när en inkommande förfrågan är accepterad. Klient-sidan kommer att skriva socket-information när den öppnat en RFCOMM-kanal till servern. Servern och klienten kommer att vara sammankopplade när de båda är kopplade till samma socket i RFCOMM-kanalen. Då kan data börja överföras.

5.1.1.2 Server

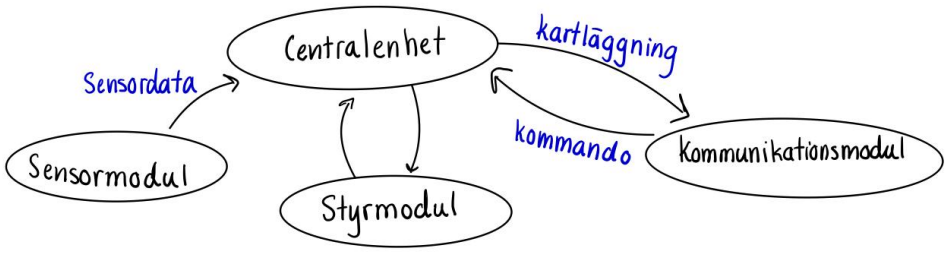
Server-filen kommer att lyssna efter en eventuell förfrågan om sambindning från en klient och därefter acceptera. När en kanal skapats kommer servern att kunna skicka data till klienten genom att skicka meddelanden. Servern lyssnar också efter eventuell data, detta genom att skicka “ID 0” till klienten. Den mottagna datan kommer att sparas i en lista. Servern kan även avsluta förbindelsen.

5.1.1.3 Klient

Klienten kommer att försöka att ansluta till servern med hjälp av MAC-adressen till datorns bluetooth-enhet. Vid förbindelse kommer klienten att kunna motta ett datapaket från servern. Klienten lyssnar kontinuerligt i autonomt läge efter “ID 0” från servern, och när det mottas så skickas data.

5.1.2 Kommunikation mellan delsystem

Kommunikation mellan delsystemen sker, bortsett från kommunikation mellan extern dator och robot, över I2C enligt figur 5. Den färdiga konstruktionen består av fyra delsystem: centralenhet, sensorenheten, kommunikationsenheten och styrenheten,



Figur 5. Kommunikationsväg mellan delsystem.

5.1.2.1 I2C

För att få en kommunikation mellan de olika modulerna behövde vi något protokoll som var möjlig att implementera på en enchipsdator. På processorn Atmega1284p finns det redan inbyggt stöd för SPI och I2C kommunikation. Båda två skickar data seriellt men har små skillnader. I2C, vilket var den som användes, består av endast två signaler: SCL och SDA. SCL är en klocksignal som ser till att alla slavar läser datan likadant. SDA är vår data som

skickas, första biten bestämmer ifall master vill läsa eller skriva och sedan skickas den data man vill skicka.

Sensormodul I2C

Till våran sensormodul skickar vi endast en “läsa” bit då vi inte behöver skicka några värden till sensormodulen. När vi skickat första biten sätter sig mastern och lyssnar tills sensormodulen har skickat all relevant data.

Styrmodul I2C

Till styrmodulen finns det flera kommandon som kan skickas. Det är ingen data som mastern vill få från styrmodulen så alla signaler mellan styr- och kommunikationsmodulen inleds med en “skriva” bit. De kommandon som kan skickas visas i tabell 2.

Kommando	ID
Stop	0
Fram	1
Bakåt	2
Rotera höger	3
Rotera vänster	4
Kör och sväng höger	5
Kör och sväng vänster	6
Ändra hastighet, höger hjul	7, data
Ändra hastighet, vänster hjul	8, data

Tabell 2. Kommandon från kommunikationsmodul till styrmodul-

5.4 Gränssnitt

Projektet består av ett användargränsssnitt skrivet i Python med paketet tkinter [6]. Tkinter är ett standard-bibliotek som tillåter snabb utveckling av grafiska gränssnitt med kraftfull funktionalitet. Se figur 1.

ChartyBot Interface har tre huvudsakliga paneler. *Controls* hanterar saker såsom att uppdatera regler-konstantervärden, sätta på och stänga av manuell styrning eller ändra hastigheten. Fönstret *Unit Feedback* visar feedback från roboten. Detta är exempelvis avstånd från respektive sensor till vägg eller uppkopplingsstatus. I det tredje fönstret visas kartan som roboten ritar upp.

Vid kalibrering av enheten så trycker användaren på önskad modul att kalibrera, skriver in ett nytt värde, och trycker *Enter Value*. Detta skickar värdet från gränssnittet till Python-filen *server.py* som hanterar den trådlösa kommunikationen mellan laptopen och roboten. Samma princip används för alla kontroller i användargränssnittet.

5.4.1.1 Manuell styrning

Användargränssnittet erbjuder användaren möjligheten att styra roboten manuellt med tangentbordets piltangenter. Det manuella läget sätts enkelt på genom att trycka på *Enable Manual Control*. Det stängs av med motsvarande knapp. Inbyggd funktionalitet i Tkinter tillåter klienten att lyssna efter avbrott från tangentbordet. Denna funktion bygger på att varje piltangent är bunden till en specifik funktion vilket skickar unik data till roboten då den trycks ned, enligt tabell 1.

5.4.1.2 Kalibrering

Genom användargränssnittet kan användaren kalibrera robotens olika reglertekniska konstanter. Dessa är PD- och KD-konstanterna. Via justeringar av dessa konstanter kan användaren ändra hur känsligt roboten reagerar på in-sensordata. Exempelvis, hur mycket den kommer svänga ifrån en vägg ifall den skulle komma för nära.

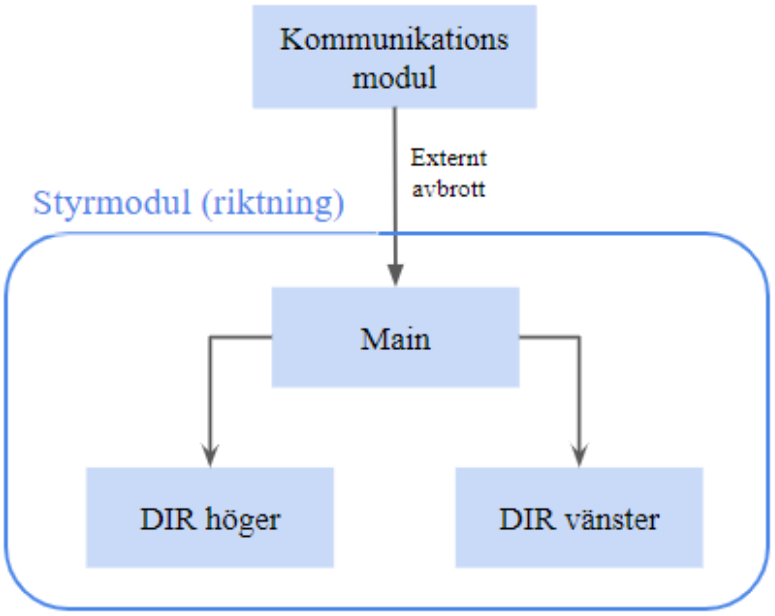
Kalibrering av dessa variabler är av stor vikt för att roboten framgångsrikt ska kunna navigera kartongvärlden.

5.2 Styrmodul

Styrmodulens syfte är att skicka information till robotens motorer om riktningen samt rotationshastigheten på hjulen.

5.2.1 Bestämma riktning

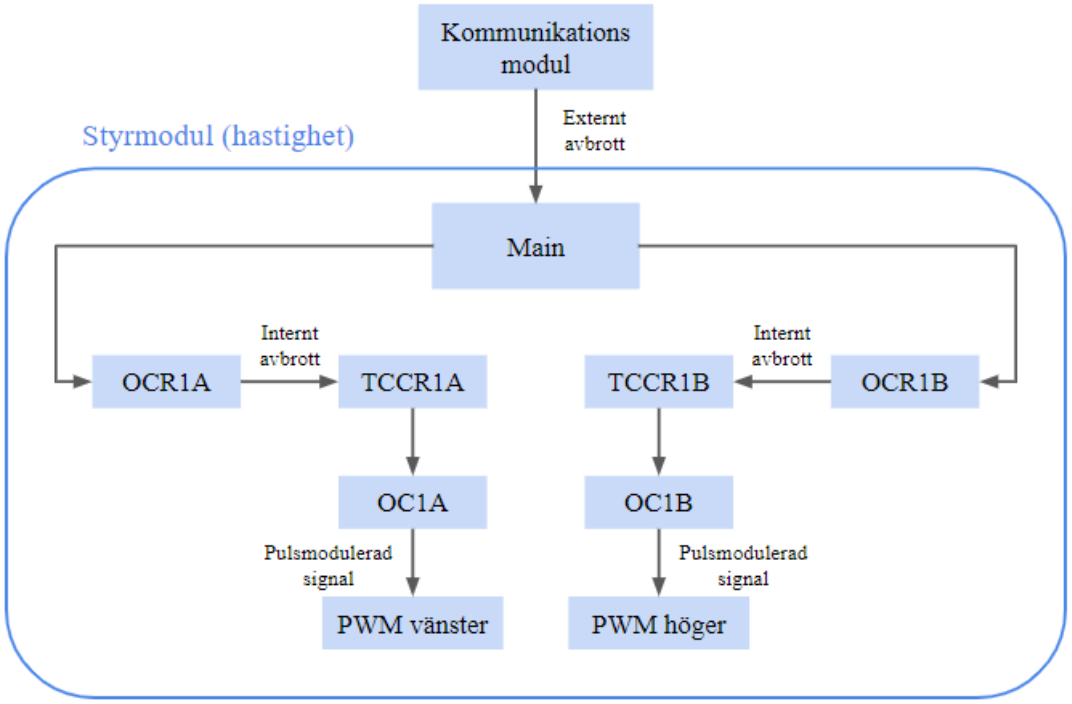
Riktningen på hjulparen bestäms genom att antingen skicka en etta eller en nolla till robotens separata hjulpar på DIR ingången. Om en etta skickas till det högra hjulparet kommer de att rotera framåt och om en nolla skickas kommer de att rotera bakåt. Samma sak gäller för det vänstra hjulparet. Riktningen skickas via kommunikationsmodulen som via ett externt avbrott får styrmodulens main fil att ändra riktning på hjulen, se figur 6.



Figur 6. Blockschema för styrmodulen.

5.2.2 Ändra hastighet

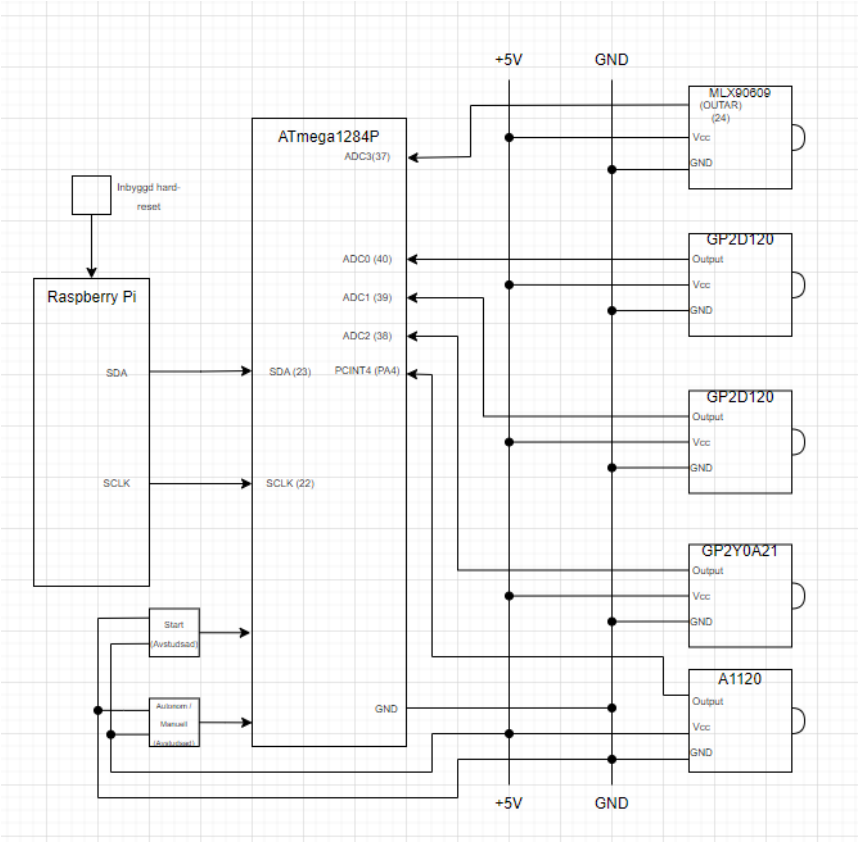
För att rotera hjulen skickas en pulsbreddsmodulerad signal. Det skickas en ny puls varje 255:e klockpuls. Hastigheten på hjulparen beror på längden på dessa pulser. Ju längre pulsen är desto snabbare kommer hjulen att rotera. Längden på pulsen bestäms med hjälp av en timer som räknar upp till 255 och sätter en utgång till ett under en del av den tiden. Timerna för de olika hjulparen är TCC1x där x är A för det vänstra hjulparet och B för det högra. Timerna är kopplade till utgångarna OC1x där det skickas ut en etta från att timern startar tills värdet som är satt på OCR1x nås. OCR1x sätts till ett värde mellan 0 och 255 som då reglerar hur långa pulserna blir och på så sätt bestäms hastigheten på hjulen. Om OCR1x är satt till noll kommer ingen puls skickas ut och hjulen kommer att stå stilla. Om OCR1x är satt till 255 kommer OC1x vara konstant hög och hjulen kommer ha maximal hastighet. OCR1x värde sätts då kommunikationsmodulen skickar ett externt avbrott till main filen för styrmodulen som ändrar värdet på OCR1x så att roboten kör i önskad hastighet, se figur 7.



Figur 7. Blockschema för styrmodulen (hastighet).

5.3 Sensormodul

För att kartlägga områden behövs sensorer som kan lokalisera saker i robotens närvaro. Därutöver behövs sensorer som kan avgöra hur långt roboten har rört sig samt någon sensor som kan avgöra i vilken riktning som roboten är riktad åt. Se figur 8.



Figur 8. Blockschema för sensormodulen.

5.3.1 Avståndssensorer

Syftet hos dessa sensorer är alltså att avgöra avståndet till väggar som både kartläggs men är också en vital del inom regleringen. Som tidigare nämnts används sammanlagt tre stycken IR-avståndsmätare av olika sorter. Den som är riktad framåt är av modell GP2Y0A21 och denna är kapabel att mäta avståndet på objekt som befinner sig inom intervallet 10 - 80 centimeter [2]. De avståndssensorer som är placerade på höger och vänster sida är båda av modell GP2D120 och dessa har kapaciteten att mäta ut avståndet till objekt som är inom intervallet 4 - 30 centimeter [3].

5.3.1.1 Avkodning av avståndssensorer

Genom A/D-omvandling av den analoga signalen som är en spänning och har tillhandahållits av avståndssensorerna, genereras ett tio-bitarstal som representerar ett avstånd. Detta tio-bitarstal behöver översättas till det avstånd som spänningen representerar. Eftersom två olika sorters avståndssensorer används behövs två olika översättningsfunktioner och dessa har framtagits genom att mäta vilket värde sensorerna ger för ett intervall på 5 centimeter från maximalt avstånd till minimalt. Sedan plottades punkterna i Excel, och lyder enligt följande:

Avkodning för GP2Y0A21: $y = 3876 * x^{-1,27}$

Avkodning för GP2D120: $y = 42,6 * e^{-0,0248*x}$

Där *y* är värdet i centimeter och *x* är värdet från A/D-omvandlingen.

5.3.2 Gyroskop

Gyroskopet är en av robotens fyra sensorer och möjliggör exakta rotationer vilket är nödvändigt för att göra svängar av samma storlek varje gång [4]. Gyroskopet är av modellen MLX 90609 och innehåller en i r-led oscillerande vikt, som vid rotationer generarar corioliskrafter. Se avsnitt *5.3.2.1 Avkodning av Gyroskop* för detaljerad beskrivning om hur datan avkodas.

5.3.2.1 Avkodning av Gyroskop

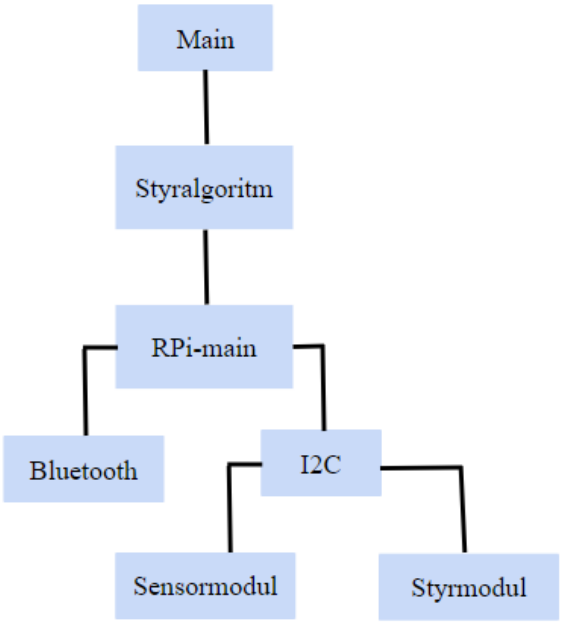
Denna sensor kommer att skicka en analog signal som är en spänning mellan 0.5V till 4.5V till A/D omvandlaren där värdet 0.5V innebär en maximal rotation moturs, 2.5V innebär ingen rotation och 4.5V representerar en maximal rotation medurs. Den analoga signalen ändras linjärt vilket möjliggör att vi kan räkna ut hur mycket den har roterat på ett enkelt sätt med ekvationen $\theta = U_{in}\omega_{max}\Delta t$.

5.3.3 Odometer

I syfte att avgöra avståndet på robotens förflyttning används en odometer [5]. Denna sensor består dels av en kugghjulsskiva som är placerad på ett av robotens fyra hjul och dels en läsgaffel som antingen ger värdet 0 eller 1 beroende på om skivan täcker läsgaffelns mätpunkt eller inte. Genom att räkna ut omkretsen på kugghjulet och dividera det med antalet "hål" i det kan man bestämma hur långt roboten åker vid varje enskild kugge. Genom summering av detta kan man bestämma hur långt roboten har åkt under en viss tidsperiod.

5.4 Centralenhet

Centralenheten är konstruerad enligt figur 9. Dess huvudsakliga syfte är att fungera som en mittpunkt, för att kunna hantera inkommande information och skicka lämplig data till de olika modulerna.



Figur 9. Blockschema för centralenhet.

5.4.1 Main

Denna fil hanterar alla kommandon från bluetooth-servern och gör de ändringar som kommandot begärt enligt tabell 1.

5.4.2 RPi-main

RPi-main använder Bluetooth för att ta emot data från klienten och I2C för att kunna kommunicera med sensor- och styrmodulen. Här finns alla funktioner som får roboten att köra enligt styralgoritmen. Följande funktionaliteter är implementerade:

- Stanna roboten.
- Värde från odometer omvandlat till centimeter.
- Kör fram.
- Justera robotens riktning.
- Uppdatera robotens vinkel.
- Roterade grader.
- Rotera höger.
- Rotera vänster.
- Sväng höger.
- Sväng vänster.
- Ändra hastighet på höger hjulpar.
- Ändra hastighet på vänster hjulpar.
- Upptäckt av vänster vägg.

- Upptäckt av höger vägg.
- Upptäckt av vägg framför.
- Initiering av I2C.

6. Slutsatser

De största förbättringsmöjligheterna ligger i optimering av koden.

Första förbättringen är att sökalgoritmen hade kunnat förenklats och förbättrats. Vid implementation valdes en egengjord sökalgoritm över en välkänd såsom A*, BFS eller DFS. Detta designval medförde svårigheter då det är svårare att implementera en egen algoritm. En fördel med algoritmen är att den möjliggör dynamisk storlek hos kartan, vilket tillåter stor flexibilitet när det kommer till testning av mjukvaran. Fortsättningsvis möjliggör det ett större användningsområde för kunden.

Mjukvaru-sidan av projektet består av en stor mängd filer som är uppdelade i olika mappar, som i sin tur ligger på tre olika enheter. Här hade en bättre struktur kunnat användas, då koden kan vara svår att navigera igenom. En bra filstruktur är därför av stor vikt för förståelse av implementationen.

En annan tydlig förbättring är att bluetooth-server borde initieras hos Raspberry Pi:n istället för hos den externa datorn. Detta hade möjliggjort att roboten kunnat avsluta förbindelsen smidigare. I slutsats, så uppfyller projektet de krav som ställts enligt kravspecifikationen. Förbättringsmöjligheterna har prioriterats bort på grund av tidsbrist.

7. Referenser

[1] https://docs.python.org/3/library/socket.html

[2] https://da-proj.gitlab-pages.liu.se/vanheden/pdf/gp2y0a21.pdf

[3] https://da-proj.gitlab-pages.liu.se/vanheden/pdf/sharp-GP2Y0A41SK0F.pdf

[4] https://da-proj.gitlab-pages.liu.se/vanheden/pdf/mlx90609.pdf

[5] https://da-proj.gitlab-pages.liu.se/vanheden/pdf/opb990t51z.pdf

[6] https://docs.python.org/3/library/tkinter.html