

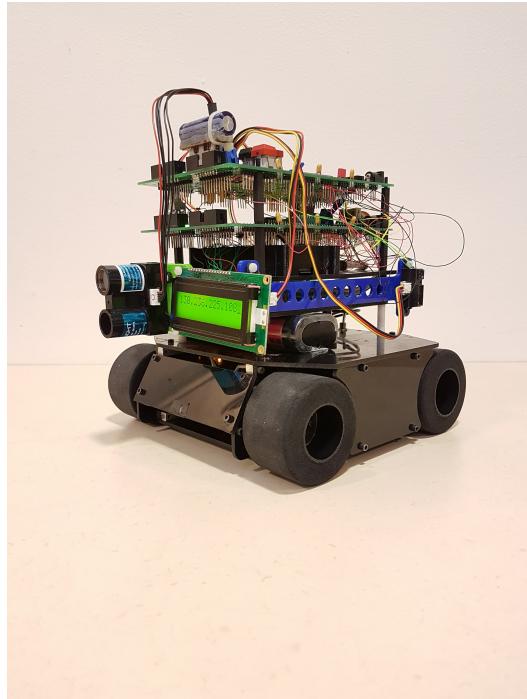
Teknisk dokumentation för kartrobot

Patrik Sletmo

Version 1.0

Status

Godkänd	Patrik Sletmo	2016-12-14
---------	---------------	------------





Projektidentitet

Grupp 3, 16/HT, KarToffel
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Patrik Sletmo	Projektledare	070 783 57 61	patsl736@student.liu.se
Rebecca Lindblom	Utvecklare	073 436 40 79	reqli156@student.liu.se
Matilda Sjöstedt	Utvecklare	070 515 84 11	matsj696@student.liu.se
Sebastian Callh	Utvecklare	073 820 46 64	sebca553@student.liu.se
Anton Dalgren	Utvecklare	076 836 51 56	antda685@student.liu.se
Matilda Dahlström	Utvecklare	070 636 33 52	matda715@student.liu.se

Hemsida: <https://github.com/SebastianCallh/kartoffel-tsea29>

Kund: Mattias Krysander, 013 - 28 2198 , matkr@isy.liu.se

Kursansvarig: Tomas Svensson, 3B 528, +46 (0)13 28 1368, tomas.svensson@liu.se

Handledare: Anders Nilsson, 3B 512, +46 (0)13 28 2635, anders.p.nilsson@liu.se



Innehållsförteckning

1 Inledning	6
1.1 Bakgrund	6
1.2 Syfte	6
2 Produkten	7
3 Teori	9
3.1 Navigering	9
3.2 Kartläggning	10
3.2.1 Positionsbestämning	10
3.2.2 Bestämning av köksö	11
3.3 Reglering	11
3.4 Kommunikation	12
3.4.1 EventBus	12
3.4.2 Bluetooth	13
3.4.3 I2C	14
4 Systemet	16
4.1 Felsökning av mjukvara	16
4.1.1 Felsökning på AVR	17
4.1.2 Felsökning på Huvudenhet	17
4.2 Felsökning av hårdvara	17
4.2.1 Låg batterispänning	17
4.2.2 Felsökning av I2C	18
5 Modulerna	19
5.1 Huvudenhet	19
5.1.1 Navigator	19
5.1.2 Communicator	19
5.1.3 Position	19
5.1.4 Driver	20
5.1.5 IR	20
5.1.6 Laser	20
5.1.7 Gyro	20
5.1.8 Kopplingsschema	21
5.1.9 Komponenter	21
5.1.10 Resurser	22
5.2 Sensorenhet	22
5.2.1 Kopplingsschema	23
5.2.2 Komponenter	23
5.2.3 Resurser	24
5.2.4 Programflöde	24
5.3 Styrenhet	25
5.3.1 Kopplingsschema	25
5.3.2 Komponenter	26
5.3.3 Resurser	26
5.3.4 Programflöde	26
5.4 Mjukvaruklient	27
5.4.1 Presentationsmodul	27



5.4.2	Fjärrstyrningsmodul	27
5.4.3	Programflöde	28
6	Slutsatser	29
6.1	Navigeringsalgoritm	29
6.2	Kartläggning	29
7	Referenser	30
A	Banspecifikation	31



Dokumenthistorik

Version	Datum	Utförda ändringar	Utförd av	Granskad
1.0	2016-12-14	Första version	Grupp 3	Patrik Sletmo



1 Inledning

Systemet beskrivet i det här dokumentet är en kartläggningsrobot som dels kan fjärrstyras och dels släppas in i ett rum och autonomt söka av det för att sedan återvända till ingången.

1.1 Bakgrund

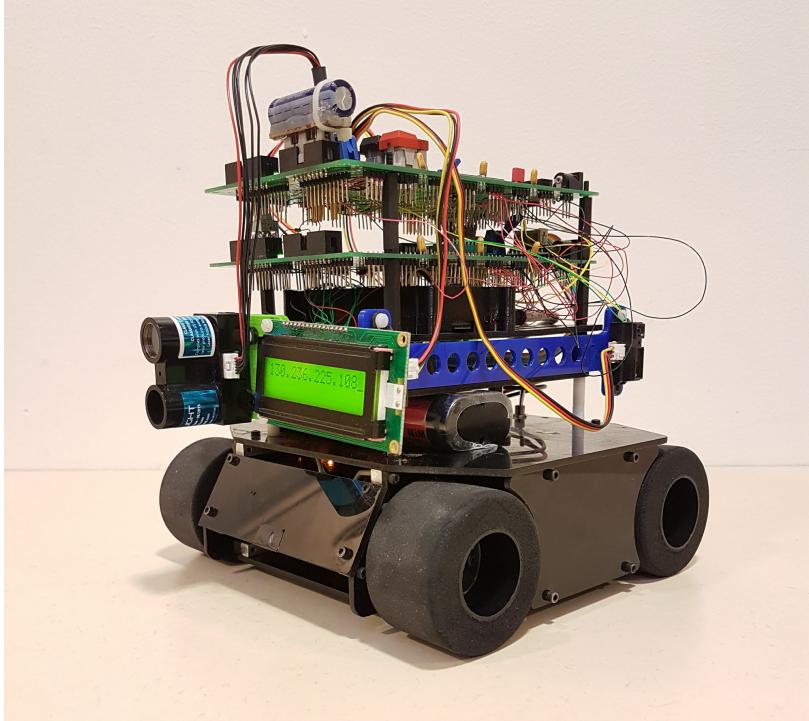
Som en del av utbildningen till civilingenjör i dاتateknik på Tekniska högskolan vid Linköpings universitet har denna projektgrupp i kurserna Konstruktion med mikrodatorer (TSEA29) arbetat mot beställaren Mattias Krysander för att tillverka en robot vars syfte är att kartlägga en sluten bana. Den färdiga produkten tävlar mot andra gruppars robotar och redovisas i samband med kursens avslutande.

1.2 Syfte

Det här dokumentet skall ge en ingående beskrivning av kartrobot-systemet från ett tekniskt perspektiv. Dokumentet ska fungera som konstruktionsunderlag för någon som vill bygga en kopia av den robot som beskrivs. Det ska från instruktioner i dokumentet även vara möjligt att underhålla och felsöka både mjuk- och hårdvaran i systemet.

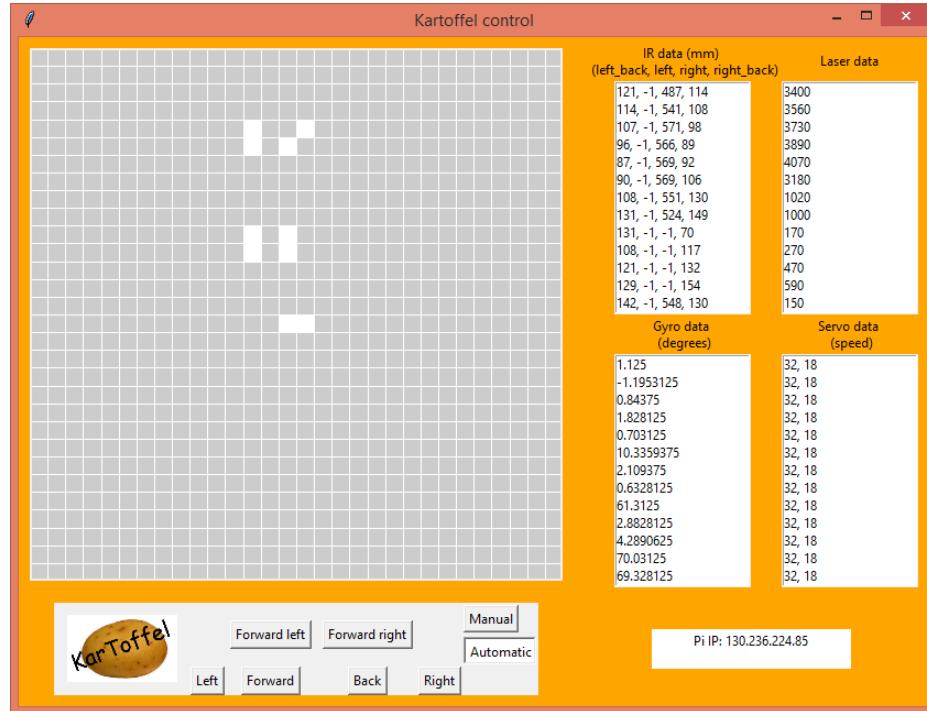
2 Produkten

Systemet består av en fyrfjulig robot samt en mjukvaruklient som kommunicerar med varandra via Bluetooth. Systemets syfte är att roboten ska kunna kartlägga sin omgivning helt autonomt och förmedla kartläggningen till mjukvaruklienten.



Figur 1: En bild på roboten.

För att utföra kartläggandet så är roboten utrustad med diverse IR-sensorer, en lasersensor och ett gyroskop för att kunna navigera och mäta avstånd. Se figur 1 för en bild på konstruktionen.



Figur 2: En bild på mjukvaruklienten.

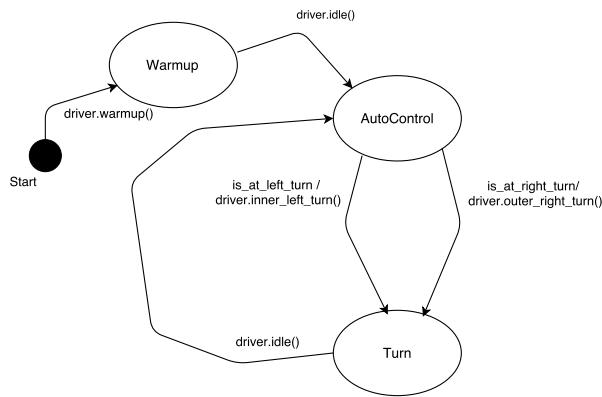
Mjukvaruklienten renderar med hjälp av data från dess Bluetooth-anslutning kartläggningen i realtid och har ett gränssnitt för att kunna fjärrstyrta roboten. Se figur 2 för en bild på mjukvaruklientens gränssnitt.

3 Teori

För att lyckas kartlägga en bana använder sig systemet av en rad olika algoritmer och tekniker som behandlar problemet att läsa in data från omgivningen, applicera det datat på kartläggningsproblemet, samt att uttrycka resultatet tillbaka till den fysiska verkligheten.

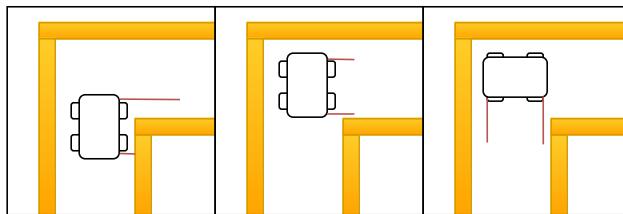
3.1 Navigering

Navigeringslogiken implementeras som en tillståndsmaskin vars transitioner styrs av indata från robotens sensorer. Den antar att omgivningen följer banspecifikationen i bilaga A.



Figur 3: Ett flödesdiagram på navigatorns tillstånd.

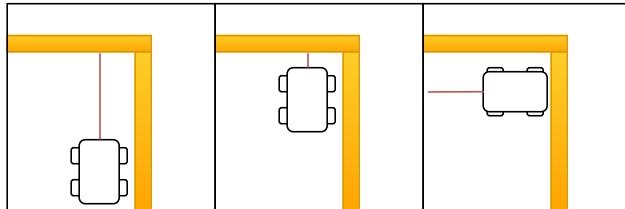
Efter en kort uppvärmningsperiod för att hinna läsa in alla sensorer så börjar roboten följa väggen på sin högra sida och håller ett lämpligt avstånd till den med hjälp av en modifierad PID-reglering (se avsnitt 3.3). Om roboten upptäcker att väggen på höger sida tar slut så innebär det att väggen har svängt av bort från roboten och att den behöver göra en yttersväng till höger för att följa efter. Skulle roboten upptäcka en vägg framför sig betyder det istället att rummet svängt av till vänster, och att roboten behöver göra en innersväng åt vänster. En återvändsgränd består ur robotens perspektiv av två stycken innersvängar till vänster. Ett flödesdiagram över robotens navigeringstillstånd kan ses i figur 3, vilket är all logik som behövs för att den ska kunna följa en vägg. Logiken hanterar yttersväng åt höger, innersväng åt vänster och även återvändsgränder.



Figur 4: En illustration på hur roboten hanterar en yttersväng.

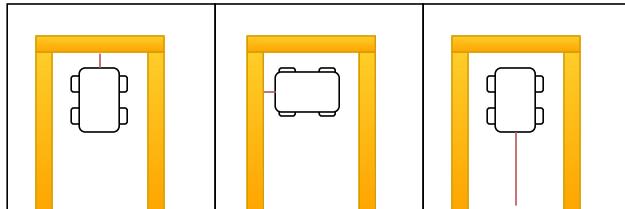
Det första momentet som roboten behöver kunna genomföra är en yttersväng till höger,

vilket illustreras i figur 4. För att detektera en sväng så använder den sina två IR-sensorer monterade fram respektive bak på höger sida. När den första IR-sensorn inte längre detekterar en vägg går den in i ett läge som förbjuder vänstersvängar, och när den bakre gör samma sak utförst en rotation 90 grader åt höger.



Figur 5: En illustration på hur roboten hanterar en innersväng.

För att utföra en innersväng använder sig roboten av lasersensorn monterad fram till och mäter avståndet till väggen. När avståndet underskrider ett visst tröskelvärde så utför roboten en rotation 90 grader till vänster. Sekvensen illustreras i figur 5. Värt att notera är att yttersväng till höger tar prioriteten över att göra en innersväng åt vänster. Det vill säga att även om lasersensorns mätvärde underskrider tröskelvärdet så kommer det att ignoreras ifall robotens främre högra IR-sensor meddelar att den är på väg mot en högersväng.



Figur 6: En illustration på hur roboten hanterar en återvändsgränd.

När roboten tar sig förbi en återvändsgränd faller den i själva verket in i samma rutin som när den utför en innersväng åt vänster upprepats två gånger. Roboten roterar när den närmar sig slutet på återvändsgränden, och direkt efter rotationen upptäcks en vägg framför den och den roterar ytterligare 90 grader för att sedan fortsätta ut ur återvändsgränden. En illustration ses i figur 6.

3.2 Kartläggning

3.2.1 Positionsbestämning

Under kartläggningen av rummet så räknar roboten sektioner som den har passerat och utnyttjar det för att bestämma sina koordinater när en sektion har passerats. Robotens startkoordinater är alltid i origo och startriktningen är alltid "norrut", alltså positivt längs Y-axeln. När roboten enligt navigeringsalgoritmen i avsnitt 3.1 bestämmer sig för att ta en sväng och rotera så kommer kartläggningen att spara den nuvarande positionen. Genom att ta skillnaden mellan högsta och lägsta mätvärdet dividerat med storleken på en ruta

fås antalet rutor som roboten förflyttat sig. Positionsbestämnigen är även den baserad på en tillståndsmaskin. Den är baserad på två tillstånd, väntar och mäter; där väntar är det tillstånd som roboten befinner sig i när den gör en 90 graders rotation. Därefter går roboten in i tillståndet mäter där en ny sektion initieras och mätdata sparar varv i huvudloopen.

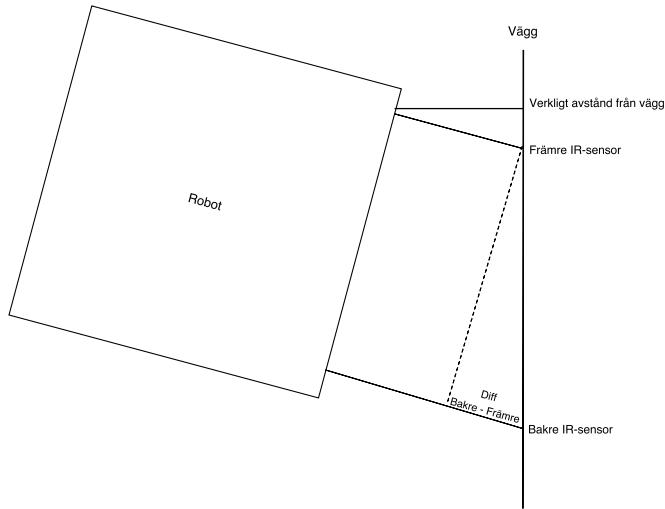
Under positionsbestämningen så kommer roboten att hålla utkik efter väggar på vänstersidan för att kunna lokalisera eventuella väggar på en köksö. Så fort roboten ser något på vänstersidan så kommer den att säga att just denna koordinat kan vara en köksö så länge den inte redan har passerats. Om roboten under första varvet åker förbi en koordinat som kan vara en köksö kommer den att tas bort som eventuell köksö.

3.2.2 Bestämning av köksö

När roboten har åkt ett varv längs ytterkanterna kommer den att gå in i ett läge där den letar efter köksön. Då kommer den att kolla i listan över potentiella köksöar och hitta minst en koordinat som representerar en köksö. När roboten har transporterat sig till en koordinat i den här listan kommer den att ta en vänstersväng och förflytta sig runt ön. Detta sker på samma sätt som beskrivs ovanför i avsnitt 3.2.1. När ett varv har passerats återvänder roboten till garaget genom att slutföra varvet.

3.3 Reglering

Roboten reglerar längs en vägg på sin högra sida med hjälp av två IR-sensorer, en vid robotens främre del och en vid robotens bakre del. Ett önskat värde är bestämt som det avstånd som roboten ska hålla ifrån väggen den reglerar emot. Med hjälp av det önskade värdet tillsammans med värdena ifrån de två IR-sensorerna räknas ett distansfel ut.



Figur 7: En illustration över hur roboten reglerar.

Distansfelet räknas ut genom att dra bort främre IR-sensorns värde ifrån det önskade av-



ståndet. Sedan läggs ett värde proportionellt mot differensen mellan bakre och främre sensorn till för att efterlikna det verkliga avståndet från väggen (se figur 7). Detta då värdet ifrån IR-sensorerna ökar mer än det verkliga värdet desto mer vinklad ifrån väggen roboten står.

Med hjälp av distansfelet och differensen utförs en enklare PID-reglering som istället för I- och D-delen använder sig av differensen och returnerar ett regleringsvärdet u .

$$u[n] = K_p * e[n] + K_a * d \quad (1)$$

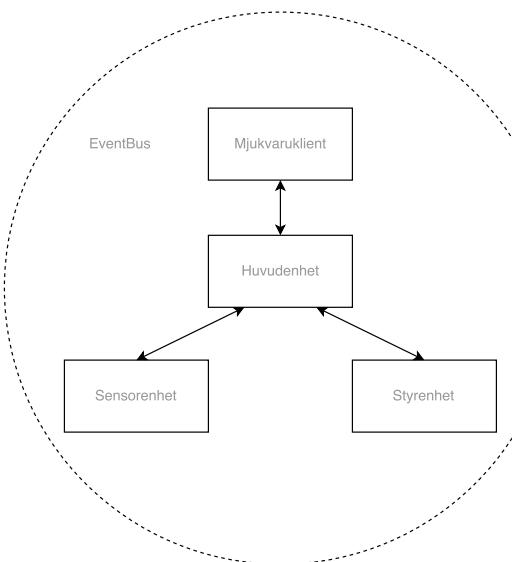
Regleringsvärdet räknas ut enligt (1) där K_p och K_a är konstanter, $e[n]$ är distansfelet vid tid n , och d är differensen mellan värdena på den bakre och främre IR-sensorn. Regleringsvärdet (som ges av (1)) adderas respektive subtraheras sedan på en standard-hastighet och skickas till de båda hjulparen för att räta upp roboten och få den att följa väggen och hålla ett önskat avstånd.

3.4 Kommunikation

Systemet byggs upp av ett antal separata moduler som behöver samarbeta för att utföra robotens funktion då dess unika funktioner är alldelvis för specifika. Kommunikationen som möjliggör detta samarbete sker både via Bluetooth och I2C beroende på modulernas placering i systemet. Då det kan bli förvirrande att hantera dataflödet på två olika protokoll ligger det ett abstraktionslager som kallas EventBus ovanpå dessa överföringsmedier. Detta abstraktionslager samt de underliggande implementationerna beskrivs mer ingående i följande delavsnitt.

3.4.1 EventBus

För att underlätta programmering av logikenheten använder sig systemet av en distribuerad eventbuss kallad EventBus som tillhandahåller funktioner för att skicka data till en annan enhet på bussen och för att lyssna på mottagna meddelanden.



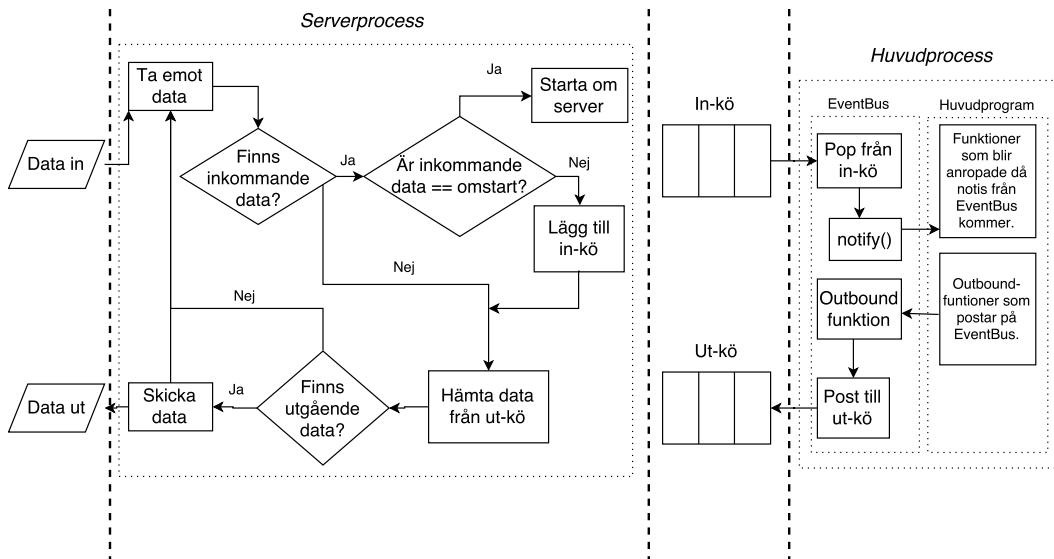
Figur 8: Översikt över eventbussens möjliga kommunikationsriktningar

Eventbussen är delad mellan sensorenheten, styrenheten, huvudenheteren samt mjukvaruklienten men eftersom bussen saknar funktionalitet för att delegera data vidare till andra enheter kan data inte skickas direkt från t.ex. sensorenheten till styrenheten, se figur 8. Eventbussen saknar fysiskt representation i den mening som t.ex. I2C-bussen har utan är bara en abstraktion som med hjälp av de underliggande kommunikationsmedierna realiseras dess distribuerade funktion.

Tanken med eventbussen är att programmeraren ska slippa tänka på att ett funktionsanrop exekveras på en annan enhet eller varifrån data kommer, så länge det läggs åtanke på att funktionerna exekveras asynkront. All mottagna data behöver manuellt behandlas i ett periodiskt funktionsanrop innan de "lyssnande" funktionsanropen körs. För att få en tydligare och mer deterministisk körning av huvudenhets program läses all mottagna data in i början av varje upprepning av huvudloopen. Data ut från huvudenheten skickas till skillnad från inläsningen direkt när anropet sker för att säkerställa att data kommer fram i andra änden så tidigt som möjligt.

3.4.2 Bluetooth

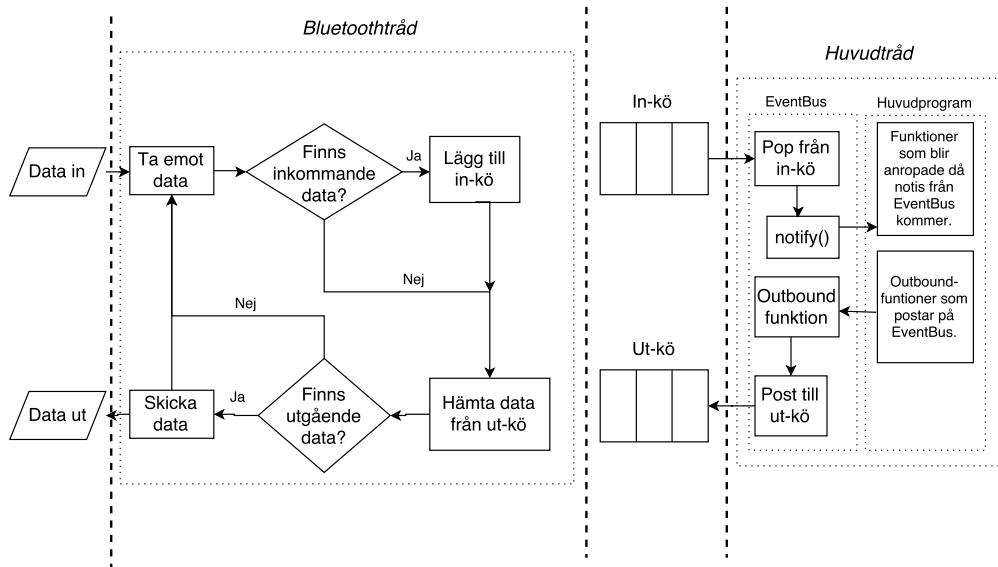
För att roboten ska kunna presentera data används Bluetooth för kommunikation med mjukvaruklienten. Kommunikationen är implementerad i Python och biblioteket Pybluez på både server- och klientsidan. Kommunikationen använder sig även av det allmänna gränssnittet EventBus.



Figur 9: Flödesschema över Bluetoothkommunikationen i huvudenheten.

Figur 9 visar flödet för Bluetoothkommunikationen i huvudenheten. Själva sändningen och mottagningen av data över Bluetooth hanteras av en Bluetoothserver som körs i en egen process skild från huvudprocessen. Servern går igång automatiskt vid start av roboten, och den kontrollerar ständigt om det finns ny data att ta emot respektive ny data att skicka. All hantering av kommandon sker i huvudprocessen, förutom om den tar emot kommandot

omstart som hanteras direkt i serverprocessen. Kommandon till huvudenheten samlas i en kö för inkommande data, där EventBus tittar efter och hämtar ny data om det finns. På samma sätt samlas data som ska till mjukvaruklienten i en kö för utgående data, i detta fall postar EventBus i kön och Bluetoothservern hämtar för att skicka. Dessa in- och utköer är implementerade som textfiler, och det gör det möjligt för serverprocessen och huvudprocessen att kommunicera.

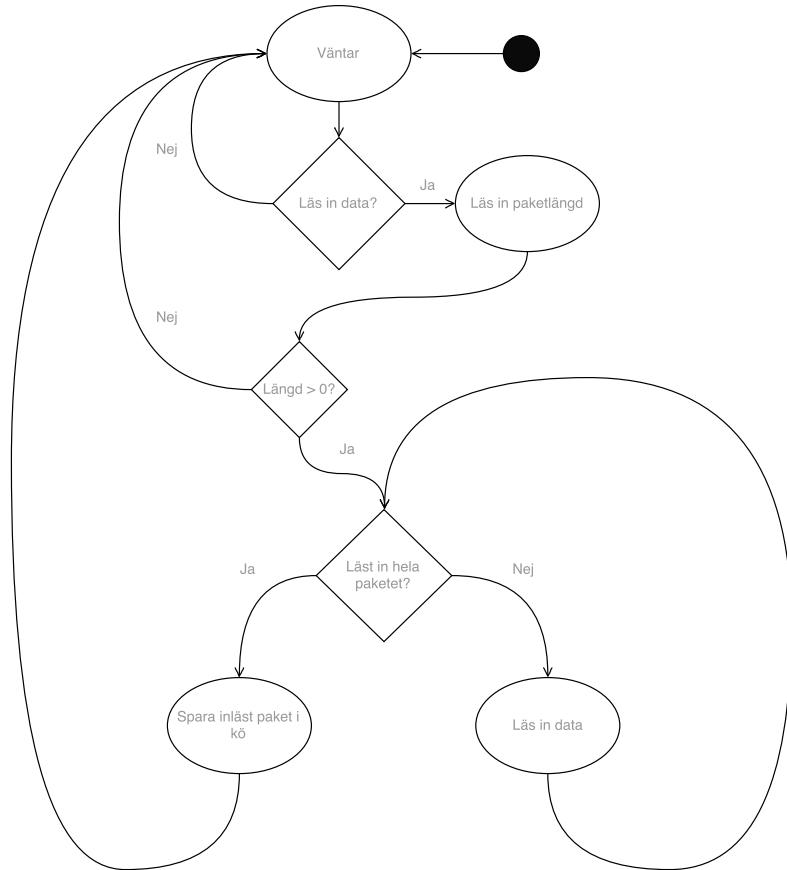


Figur 10: Flödesschema över Bluetoothkommunikationen i mjuvaruklienten.

Bluetoothklientens flöde beskrivs i figur 10 och fungerar på samma sätt som servern, med några mindre skillnader. Klienten körs i samma process men olika trådar, och därför behöver inte heller köerna vara implementerade som textfiler.

3.4.3 I2C

För att kunna kommunicera med olika moduler och sensorer på själva roboten används en I2C-buss som delas mellan huvudenheten, sensorenheten, styrenheten, laser-sensorn samt gyroskopet där huvudenheten agerar master och alla andra enheter slavar. Eftersom kraven som ställs på kommunikationen mellan modulerna förutsätter att data kan skickas i form av anrop räcker inte I2C-bussens protokoll till utan skrivningen och inläsningen till/från adresser behöver utökas med funktionalitet som gör det möjligt att skicka faktiska datapaket.



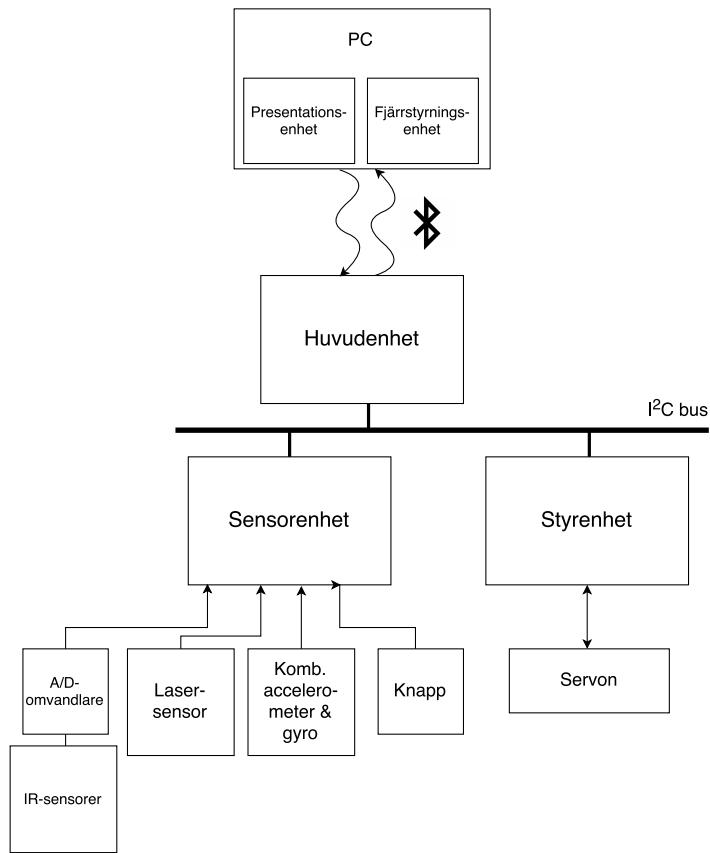
Figur 11: Flödesschema över I2C-kommunikationen

Det påliggande protokollet implementerar köer av paket för båda datariktningarna på AVR:erna som pollas och fylls på från huvudenheten. Varje paket innehåller ett antal bytes för dess data där den första byten innehåller paketets id. Eftersom I2C endast tillåter läsning eller skrivning av en enda byte innehåller varje paket on-the-wire också en inledande byte som specificerar dess längd, vilket ger en maxlängd av 255 bytes. När ett paket lästs in läggs det i en intern kö som hanteras av den modulspecifika koden, se figur 11.

Alla inkopplade enheter på I2C-bussen använder sig inte av det protokoll som beskrivs ovan utan kommunikation med vissa enheter fungerar på det vanliga sättet genom läsning och skrivning till adresser. Dessa enheter är laser-sensorn samt gyroskopet, som adresseras direkt från huvudenheten istället för att passera via sensorenheten till följd av begränsningar som följer från single-master implementationen. Läsning från dessa sensorer sker periodvis i huvudenhetens programloop och sparas undan så att olika delar av programmet kan komma åt det senaste datat utan att behöva läsa multipla gånger.

4 Systemet

Roboten och den tillhörande mjukvaruklienten utgör hela systemet som från början av projektet designats ur ett modulbaserat perspektiv. Tanken med denna separation är att varje modul inte ska vara ansvarig för fler områden än den behöver och på så sätt öka underhållbarheten i systemet. Eftersom varje modul är inkluderad i systemet med hjälp av det gemensamma gränssnitt EventBus (se avsnitt 3.4.1) kan en modul återimplementeras utan att kräva omskrivningar i resten av systemet.



Figur 12: En bild på systemets övergripande konstruktion.

I figur 12 syns systemet i sin helhet med alla enheter och dess respektive anslutningar till huvudenheten. Hierarkin i systemet gör det tydligt vilken enhet som ansvarar för vilka externa komponenter med undantag för lasersensorn och gyroskopet som kommunicerar direkt mot huvudenheten. Anledningen till denna uppdelning är förklarat i avsnitt 3.4.3.

4.1 Felsökning av mjukvara

Redan från tidigt stadio i projektet har det legat högt fokus på testbarhet och felsökning. Det finns enkla gränssnitt både in och ut från robotens alla processorer som gör att data kan manipuleras och inspekteras utan att behöva pausa programmet eller spara undan data i filer.



4.1.1 Felsökning på AVR

Eftersom AVR:erna i sitt grundutförande inte kan ge någon avancerad utdata finns möjligheten att på roboten koppla varje AVR direkt till en dators USB-port med hjälp av en seriell anslutningskabel. Genom att öppna seriellporten i t.ex. PuTTY (Windows) eller screen (UNIX) kan data skickas fram och tillbaka mellan datorn och mikroprocessorn, se listing 1. I den nuvarande konfigurationen stöds bara möjligheten att skicka data från roboten till datorn men gränssnittet stödjer även den motsatta datariktningen. Implementationen bygger på att ersätta stdout-strömmen vilket möjliggör utskrift till dator med det bekanta gränssnitt definierat i stdio.h, se listing 2 för ett komplett exempel.

```
1 $ screen /dev/tty.usbserial
```

Listing 1: Anslutning via screen

```
1 #include "common/debug.h"
2
3 // Initialization, performed during program startup
4 initialize_uart();
5 sei();
6
7 // Debugging, performed at any point during execution
8 int measuredValue = 42;
9 printf("Measured value: %d\n", measuredValue);
```

Listing 2: Exempel av utskrift

4.1.2 Felsökning på Huvudenhet

Huvudenheten består av en Raspberry PI och kör kod skriven i Python, vilket tillsammans ger den utmärkta felsökningsmöjligheter. Enklast är att ansluta sig till huvudenheten med en SSH-uppkoppling över Eduroam men det går även att koppla in tangentbord, mus och skärm direkt till roboten om så önskas. Ifall det trådlösa nätverket är för instabilt går det också att koppla in en ethernet-sladd mellan datorn och roboten. För att ansluta till enheten via SSH så loggar man in med användaren “pi” på den IP-adress som visas på robotens display och anger användarens lösenord. Notera att displayens innehåll uppdateras varje minut-omslag och det kan dröja ett par minuter innan roboten kopplar upp sig mot nätverket första gången.

4.2 Felsökning av hårdvara

På grund av varierande spänningsnivåer så inträffar det ibland fel i hårdvaran där mjukvaran har svårigheter att korrigera fel på egen hand. De vanligaste felet och dess lösningar beskrivs i detta avsnitt.

4.2.1 Låg batterispänning

I stort sett alla fel som uppkommer i hårdvaran beror på en för låg batterispänning. Det finns inget sätt att kontrollera robotens batterispänning från mjukvaruklienten eller huvudenhetens terminal utan batteriet måste avmonteras från roboten för att därefter testas med en multimeter. Lättast sättet att undersöka ifall batteriet har för låg spänning är genom att observera robotens beteende. Vanliga symptom är:

- Trötta servon



- Sena svängar i hörn
- Annorlunda reglering
- Fel mätvärden
- Brutet anslutning över WiFi

Ifall något av dessa symptom observeras bör batteriet omedelbart bytas mot ett nyladdat för att undvika skador på roboten.

4.2.2 Felsökning av I2C

Ibland händer det att I2C-bussen tappar bort någon enhet eller helt enkelt slutar fungera, särskilt vid låga batterinivåer. Det går att verifiera att I2C-bussen fungerar som den ska genom att köra ett kommando från huvudhetens terminal. Se listning 3 för kommando och förväntad utdata för en hälsosam buss. Om bussen är trasig så kommer huvudhetens program att sluta fungera och skriva ut ett felmeddelande i loggen. I de flesta fall går det att lösa problemet genom att bryta och sedan slå på spänningsmatningen till roboten men om batterinivån är låg kan batteriet behöva laddas.

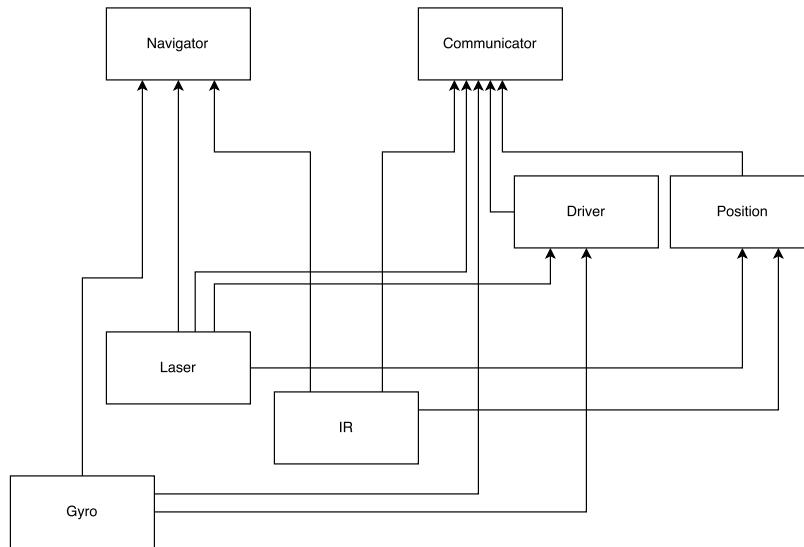
```
1 $ i2cdetect -y 1
2      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
3 00: -----
4 10: --  --  --  --  --  --  --  -- 19  --  --  -- 1e  --
5 20: -----
6 30: 30  --  --  --  --  --  --  --  --
7 40: 40  --  --  --  --  --  --  --  --
8 50: -----
9 60: --  -- 62  --  --  --  --  -- 6b  --  --  --
10 70: --  --  --  --  --  -- 77
```

Listing 3: Kommando och förväntad utdata för en fungerande I2C-buss

5 Modulerna

5.1 Huvudenhet

Huvudenheten realiseras med en Raspberry PI, på vilken operativsystemet Raspbian körs. Den representerar robotens beslutsfattande organ och sköter navigationsbeslut, kartläggning, kommunikation med mjukvaruklienten samt agerar master på robotens I2C-buss.



Figur 13: Ett diagram över logikenhetens mjukvarumoduler.

Robotens logik är skriven i programmeringsspråket Python och är moduluppbryggt. Vid start injiceras alla beroenden i programets klasser enligt diagrammet i figur 13 och en huvudloop startar som uppdaterar relevanta objekt varje iteration. En beskrivning av mjukvarumodulerna finns nedan.

5.1.1 Navigator

Navigator-klassen sköter all logik för förflyttning. Givet viss indata från sensorerna fattar den beslut för att klara av uppdraget. Se avsnitt 3.1 för en beskrivning av navigeringsalgoritmen som används.

5.1.2 Communicator

Communicator tar emot begäran från mjukvarumodulen och innehåller all logik för att leverera korrekt svar. Den skickar både vidare styrkommandon till styrenheten samt returnerar begärda data till mjukvaruklienten.

5.1.3 Position

Position hanterar all logik för hur roboten gör för att bestämma sin position på kartan beroende på vilket tillstånd i kartläggningen roboten befinner sig. Se avsnitt 3.2.1 för en utförligare beskrivning av hur positionsbestämningen fungerar.



5.1.4 Driver

Driver innehåller metoder för att utföra standardiserade förflyttningar, t.ex. en inre högersväng eller en yttre vänstersväng. Vid ett anrop till en metod så ges Driver en uppgift som består av en instruktion, en jämförelsefunktion och ett värde till funktionen som Driver använder för att veta när dess uppgift är klar. Till exempel kan en uppgift (i ord) bestå av ”vänstersväng, mät i grader, 90”, vilket skulle få roboten att svänga vänster, till det att en skillnad i 90 grader uppmäts från gyroskopet.

5.1.5 IR

IR-modulen har en funktion som anropas när sensorenheten har skickat sensordata via I2C-bussen till huvudheten genom att utnyttja designmönstret Observer. De värden som kommer på bussen sparas ner i modulen och kan då användas i andra moduler.

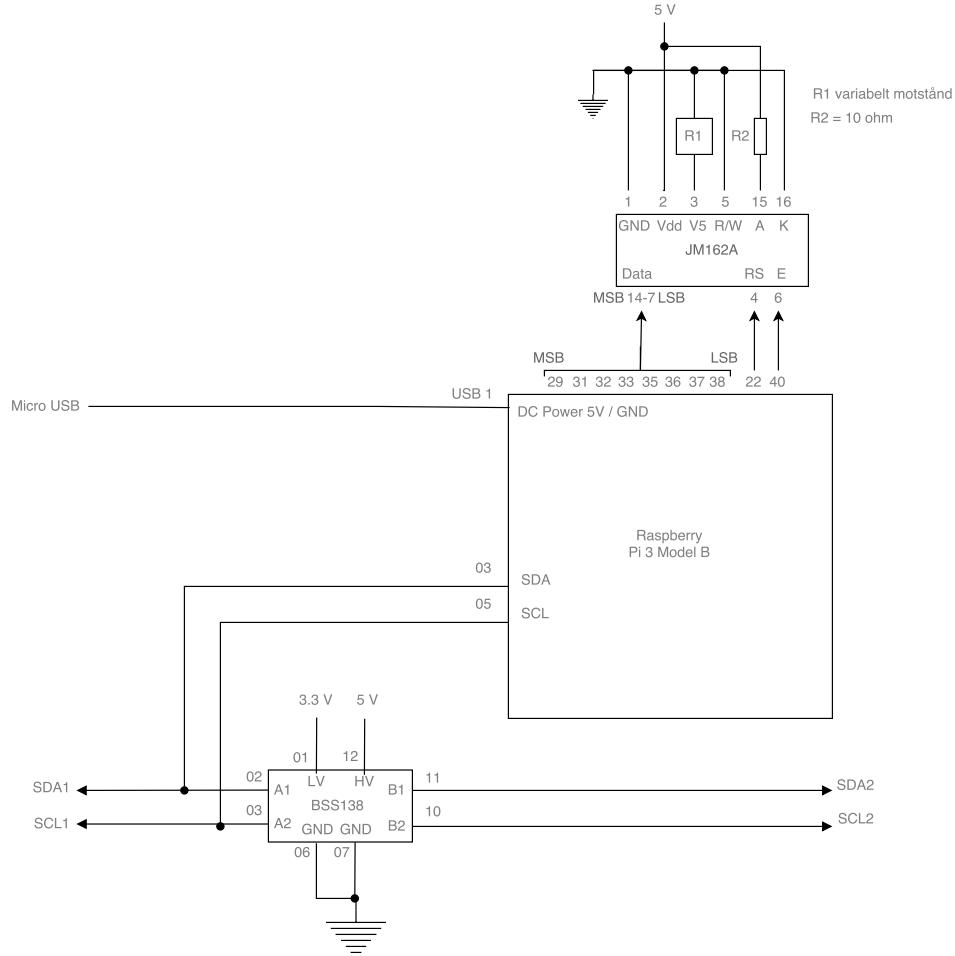
5.1.6 Laser

Laser-modulen läser in mätvärden från lasersensorn och översätter dem till Python-heltal som sedan sparas undan en gång per huvudloop. På så sätt kan övriga moduler läsa det sparade värdet, vilket säkerställer att samtliga av robotens moduler agerar på samma lasersensorvärd.

5.1.7 Gyro

Gyro-modulen fyller samma funktion som Laser-modulen. Den läser in värden från gyroskopet och sparar värdena som övriga moduler kan läsa ifrån.

5.1.8 Kopplingsschema



Figur 14: Kopplingsschema för huvudenheten.

Huvudenheten består av en Raspberry Pi 3 och en LCD-display av modellen JM162A. Utöver kopplingen till LCD:n så kopplas två pins till I2C-bussen. Eftersom Raspberry Pi drivs av 3.3 V matspänning och de övriga ATmega-processorerna i roboten drivs av 5 V så behöver bussen nivåskiftas så att de kan kommunicera med varandra, vilket görs med nivåskiftaren BSS138.

5.1.9 Komponenter

Tabell 1: En tabell över huvudenhetens komponenter.

Komponent	Antal
Raspberry PI 3	1
JM162A	1



Tabell 1 visar huvudenhetens komponenter.

5.1.10 Resurser

Tabell 2: En tabell över tillgänglinga portar på huvudenheten.

Port	Antal	Används
GPIO pins	40	12

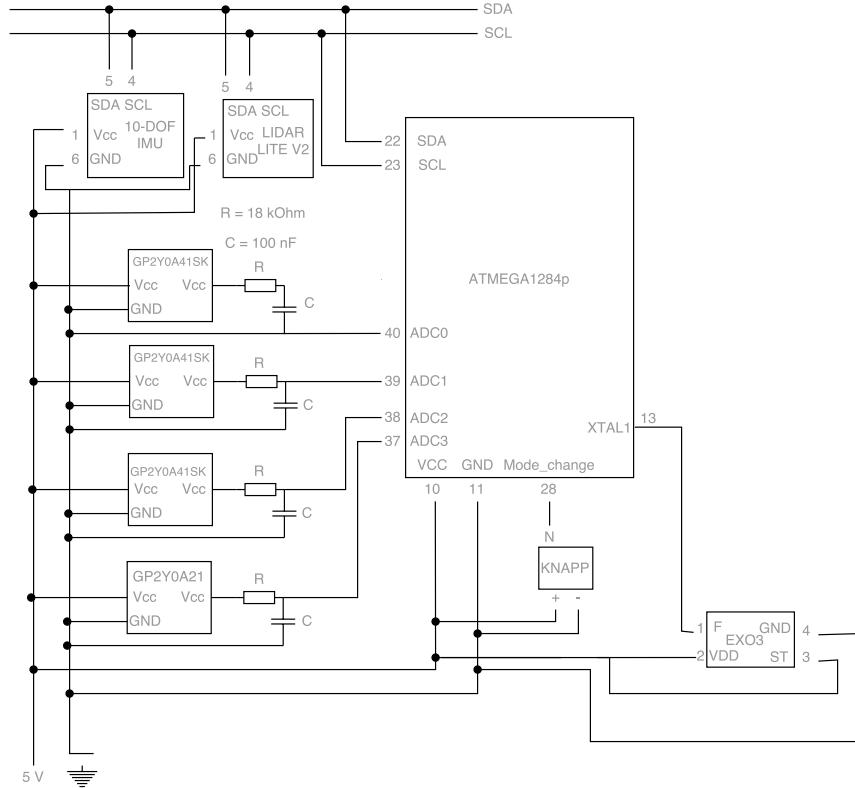
Tabell 2 visar huvudenhetens tillgängliga portar.

5.2 Sensorenhet

Sensorenheten har i uppgift att läsa in värden från robotens sensorer och rapportera värden till huvudenheten. Den består av fyra IR-sensorer, ett gyroskop, en lasersensor och har sin egen beräkningsenhet, en ATmega 1284, vilket låter den arbeta asynkront från andra enheter. Den ser på så sätt alltid till att ha data tillgänglig närhelst huvudenheten begär den. För IR-sensorerna sker det genom att processorn låter AD-omvandlaren köras för en sensor i taget och sparar sedan undan avståndet i minnet, medan gyroskop och laser håller koll på sina egna värden och är kopplade direkt på I2C-bussen. Huvudenheten kan sedan fråga sensorenheten efter ny IR-mätdata, eller fråga gyroskopet/lasern direkt varpå den får sensordatan levererad över I2C.

Avläsningarna från IR-sensorerna är analoga, och konverteras till digitalt med hjälp av ATmegans interna AD-omvandlare. För att sedan omvandla mätdata till en approximation i millimeter används en tabell sparad i ATmegans minne.

5.2.1 Kopplingsschema



Figur 15: Kopplingsschema för sensorenheten.

Som figur 15 visar är IR-sensorerna kopplade till ATmegan via lågpassfilter för att reducera brus. Längst upp i kopplingsschemat ses den gemensamma I2C-bussen, dit gyroskop och laser är anslutna. Notera att EXO3-komponenten är delad mellan sensorenheten och styrenheten trots att den illustreras som separata komponenter i kopplingsschemat.

5.2.2 Komponenter

Tabell 3: Tabell över de komponenter som sensorenheten består av.

Komponent	Antal
ATmega 1284	1
LIDAR-Lite v2	1
Knapp	1
GP2Y0A41SK IR-Sensor	3
GP2Y0A21 IR-Sensor	1
Adafruit 10-DOF IMU	1
EXO3	1 (delad)

I tabell 3 listas komponenterna i sensorenheten.

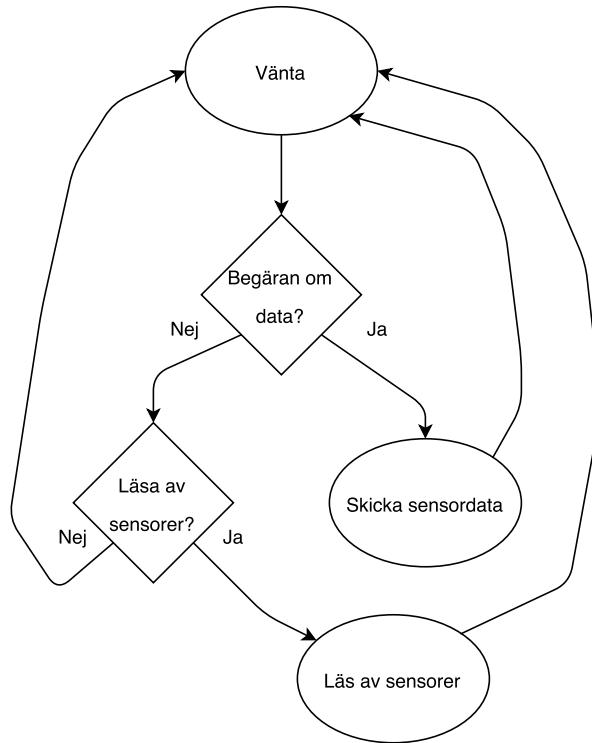
5.2.3 Resurser

Tabell 4: Tabell över tillgängliga portar på processorn.

Port	Antal	Krävs
SDA	1	1
SCL	1	1
PCINT	24	1
A/D	8	4
USART	2	1
JTAG	1	1
CLK	1	1

I tabell 4 listas antalet portar som behövs på ATmegan.

5.2.4 Programflöde



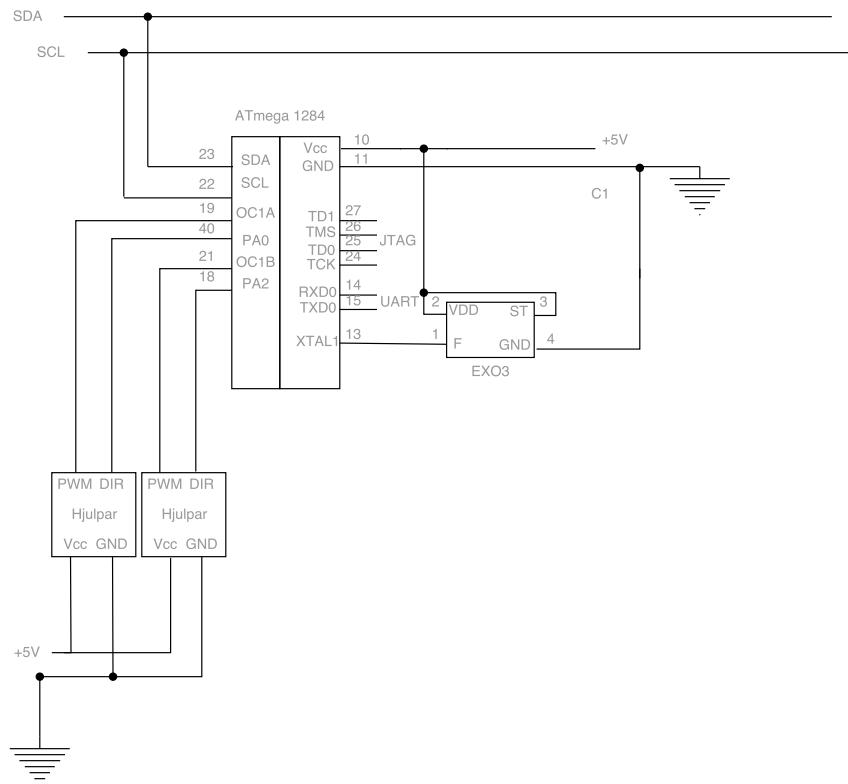
Figur 16: Ett flödesdiagram över sensorenhetens tillstånd

För att kunna tillhandahålla sensordata från IR-sensorerna när huvudenheten ber den om det så läser sensorenheten in data med jämna mellanrum och sparar den lokalt för att sedan skicka datan på kommando från huvudenheten. Om den fått en begäran om sensordata från huvudenheten så tar det prioriteten över att läsa in ny data. Ett flödesdiagram över dess beteende ses i figur 16.

5.3 Styrenhet

Styrenhetens uppgift är att kontrollera robotens förflyttning genom att ge korrekta signaler till dess hjulservon. Hjulen styrs parvis, det vill säga vänster hjulpar styrs med en signal och höger hjulpar med en annan. Hastigheten på hjulen styrs genom pulsbreddsmodulering (PWM, pulse-width modulation) medan hastigheter internt är definierade som heltal. Att hjulen styrs genom PWM innebär att deras hastighet är beroende av vilken pulskvot utsignalen har. Utsignalen växlar mellan hög och låg med vissa intervall, och ju högre kvot av signalen som är hög ju snabbare kommer hjulen att åka. Det är styrenhetens uppgift att översätta heltalshastigheter till korrekt pulskvot.

5.3.1 Kopplingsschema



Figur 17: Kopplingsschema för styrenheten.

Styrenheten klockas av en extern EXO3-klocka som säkerställer en synkroniserad klockfrekvens bland robotens processorer, och är kopplad till de två hjulpare som sitter på robotens kropp. Likt sensorenheten så kommunicerar även styrenheten över I2C-bussen.

5.3.2 Komponenter

Tabell 5: Tabell över styrenhetens komponenter.

Komponent	Antal
ATmega 1284	1
Terminator (bas för fyrfjulingsrobot)	1
EXO3	1 (delad)

Tabell 5 visar styrenhetens komponenter.

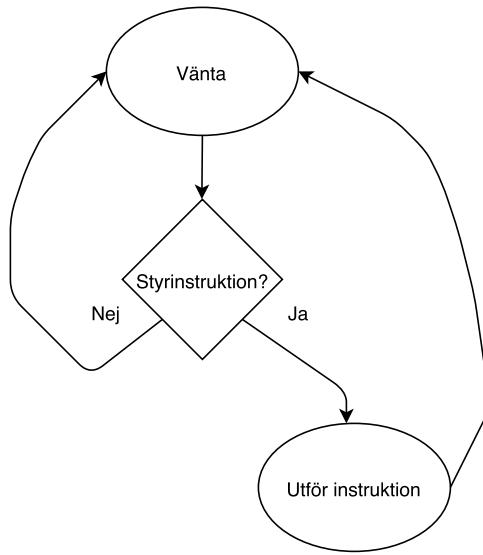
5.3.3 Resurser

Tabell 6: Tabell över tillgängliga portar på processorn.

Port	Antal	Krävs
SDA	1	1
SCL	1	1
PWM	6	2
USART	2	1
JTAG	1	1
CLK	1	1

Tabell 6 visar styrenhetens komponenter.

5.3.4 Programflöde



Figur 18: Ett flödesschema över styrenhetens tillstånd

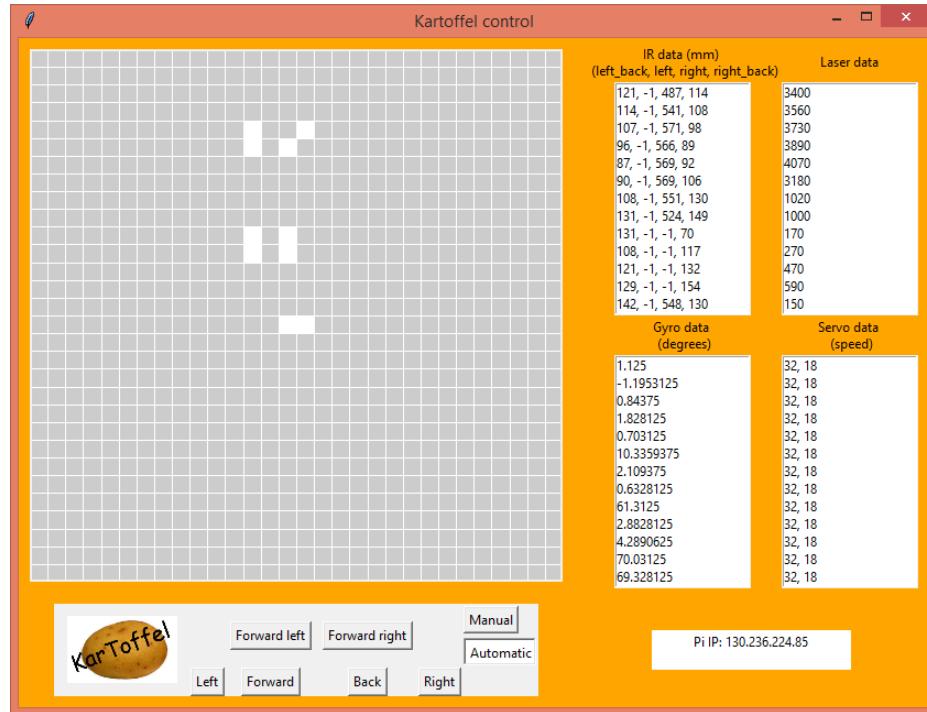
Styrenheten står hela tiden och väntar på instruktioner från huvudenheten, och så snart den



får några utför den dem så snabbt den kan. En instruktion i det här sammamhanget är att sätta nya hastigheter på de båda hjulparen. Ett flödesschema över styrenhetens beteende kan ses i figur 18.

5.4 Mjukvaruklient

För att kommunicera med roboten används en mjukvaruklient. Programmet har två moduler, en för presentation av data från roboten, och en för fjärrstyrning. Klienten är skriven i Python, och biblioteket Tkinter för det grafiska gränssnittet samt Pybluez och EventBus för Bluetoothkommunikation.



Figur 19: Mjukvaruklientens grafiska gränssnitt

5.4.1 Presentationsmodul

När roboten undersöker rummet kommer den att generera flera olika typer av data. Denna data visas av presentationsmodulen. Roboten skickar ingen data självmant, utan får kontinuerligt begäran av presentationsmodulen över Bluetooth. Sensor-, gyro-, servo- och laserdata presenteras till höger i det grafiska gränssnittet, se figur 19. I gränssnittet presenteras också robotens IP-adress om den är uppkopplad på ett trådlöst nätverk.

5.4.2 Fjärrstyrningsmodul

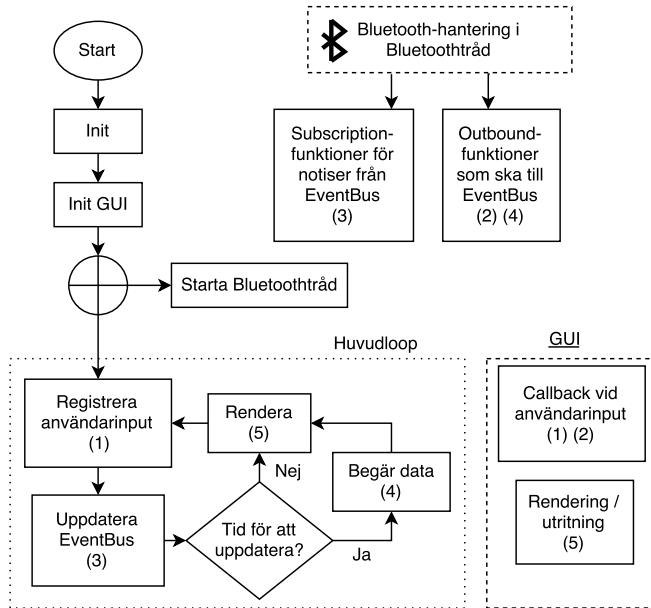
I det grafiska gränssnittet finns en panel med knappar för att skicka styrkommandon till roboten, se figur 19. Dels finns det två knappar för att växla mellan manuellt och autonomt

styrläge, samt knappar med riktningar för att styra i det manuella läget. För styrning kan även följande tangenter på tangentbordet användas:

- Q - fram vänster
- W eller pil upp - framåt
- E - fram höger
- A eller pil vänster - rotera vänster
- S eller pil ner - bakåt
- D eller pil höger - rotera höger

5.4.3 Programflöde

Presentations- och fjärrstyrningsmodulen har ingen tydlig distinktion i klientens programflöde, utan sker gemensamt i klientens flöde.



Figur 20: Flödesschemat för mjukvaruklienten

Figur 20 visar flödesschemat för mjukvaruklienten. Hanteringen för Bluetooth sker i en separat tråd för att det grafiska gränssnittet ska kunna fungera oberoende av väntan på sändning och mottagande av data. Flödet för Bluetoothkommunikationen kan ses i figur 10. Om användarinput registreras (se siffra 1 i bilden) hanteras detta av callbackfunktioner i GUI:t. Om denna input är styrkommandon som ska skickas till roboten skickas detta vidare till outboundfunktionerna (se siffra 2 i bilden). Vid uppdatering av EventBus (se siffra 3 i bilden), anropas subscriptionfunktioner som hanterar inkommande data. Vid begäran av data (se siffra 4 i bilden) anropas outbound-funktioner som skickar kommandon till roboten via EventBus. Rendering av gränssnittet och uppritning av ny data (se siffra 5 i bilden) sker i GUI:t.



6 Slutsatser

6.1 Navigeringsalgoritm

Navigeringsalgoritmen är i nuläget begränsad till att röra sig i ett rutnät och får således bara röra sig i vinklar som är multiplar av 90 grader. Begränsningen beror på att roboten inte har någon bra logik för att bestämma sin position utan att följa en vägg. Om roboten skulle kunna bestämma sin position när den navigerade i godtycklig riktning och på öppna ytor hade den totala sträckan som roboten behövt färdas minskat.

6.2 Kartläggning

Robotens kartläggningsalgoritm kartlägger sin omgivning genom att beräkna raksträckor den åkt när den följt en vägg och sparar start- och slut-koordinaterna. Om roboten kunde observera sin omgivning t.ex. med en svepande laser och beräkna var väggar fanns utifrån den datan så skulle den ej behöva färdas längs med alla väggar för att kartlägga dem. Det här förbättringsförslaget går hand i hand med förbättringarna föreslagna i avsnitt 6.1.



7 Referenser

- Pybluez: <https://github.com/karulis/pybluez>
- Tkinter: <https://wiki.python.org/moin/TkInter>



A Banspecifikation

Ban- och tävlingsspecifikation för kartrobotar 2016

Version 1.0

Gruppsammanssättning

Namn	Gruppnr och namn	E-post
Rebecca Lindblom	3 - KarToffel	reбли156@student.liu.se
Hannes Haglund	1 - Kartrobot	hanha265@student.liu.se
Jacob Lundberg	6 - Dora the Explorer	jaclu010@student.liu.se
Leopold Arrestöm	4 - RobotN	leoar200@student.liu.se
Joakim Argillander	2 - Kartrobot	joaar286@student.liu.se
Ermin Pitarevic	5 - V	ermpis936@student.liu.se

Innehåll

1. Inledning
2. Banutförande
 - 2.1 Givet enligt projektdirektiv
 - 2.2 Tillägg
3. Tävlingsupplägg
4. Diskvalifikation
5. Tidspälägg
6. Exempel på banutförande

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförd av	Granskad
0.1	2016-09-02	Första utkast.	Gruppen för kartrobotregler.	Alla beställare.
1.0	2016-09-08	Strukturerat som en LIPS-mall samt lagt till preliminära strafftider.	Gruppen för kartrobotregler.	Alla beställare

1. Inledning

Detta dokument ämnar att redogöra reglerna och begärnsningarna som används i den avslutande tävlingen för kartrobotar inom ramen för kursen TSEA29. De olika grupperna har gemensamt kommit överens om detta för att kunna följa gemensamma riktlinjer, och en representant för vardera grupp har varit med och utformat detta. På så sätt är grupperna leverantörer, och alla deras beställare detta dokumentets beställare, ty dokumentet ska ingå i respektive grups kravspecifikation.

Dokumentet uppdateras senare efter gemensamt möte mellan alla grupper och dess beställare. Efter överrenskommelse ska tider för tidstillägg och maxtid bestämmas av alla tre beställare gemensamt.

De olika kraven som ställs på en giltig bana, och de olika reglerna definieras i tre-kolumniga tabeller med följande struktur.

Krav-nummer	Förändring	Beskrivning
-------------	------------	-------------

”Krav-nummer” identifierar kravet. ”Förändring” markerar när kravet infördes, och om det är ett tillskott eller en uppdatering. ”Original” betyder då ett tillägg i dokumentets första utgåva. Beskrivning förklarar den faktiska regeln. Alla krav har samma prioritet; dessa regler ska följas av samtliga grupper.

2. Banutförande

2.1 Givet enligt projektdirektiv

1.	Original	Banan som roboten ska utforska är uppbyggd av kartongväggar, hädanefter kallat väggar, max 6x6 meter från vägg till vägg.
2.	Original	Kartongväggarna skall ha längder som är multipler av 40 cm.
3.	Original	Det skall ej vara möjligt att kartlägga kartongvärlden genom att endast följa en vägg.
4.	Original	Alla hörn ska ha en vinkel av 90 grader.
5.	Original	Exakt en ”köksö” skall förekomma.
6.	Original	Den sida på köksön som är närmast ytterväggen ska ha ett avstånd till ytterväggen på max 80 cm.

2.2 Tillägg

7.	Original	Startpunkten ska definieras som ett "garage": en ruta i rutnätet med väggar på tre sidor, 40x40 centimeter.
8.	Original	Målet ska vara detsamma som startpunkten.
9.	Original	Roboten ska få starta och sluta roterad i valfri riktning.
10.	Original	"Garaget" ska vara placerat i ytterväggen (inte i köksön).
11.	Original	Ytan av "garaget" ska inkluderas i de 6x6 meter som utgör banan.
12.	Original	Alla kartongväggar ska vara delar i "väggceller" som upptar en avskärmad yta på 40x40 centimeter. Kartongväggar inuti oåtkomliga områden (ur robotens perspektiv) krävs ej.
13.	Original	Köksöns väggceller ska ej vara förskjutna i rutnätet som bildas av rummets alla celler.
14.	Original	"Diagonaler" ska vara tillåtna, det vill säga väggceller som endast delar ett gemensamt hörn.
15.	Original	Golvet ska utgöras av laminatgolv á la LiU-standard.

3. Tävlingsupplägg

16.	Original	Varje robot ska köra två heat.
17.	Original	Det bästa av de två heaten ska räknas i tävlingen.
18.	Original	Båda heaten ska köras på samma bana.
19.	Original	Roboten ska autonomt köra sitt heat utan yttre påverkan från gruppens medlemmar.
20.	Original	Mellan heaten ska grupperna få chans att laga sin robot vid uppenbara skador, men det ges ej möjlighet att förbättra roboten från dess utförande vid leverans, t.ex. förändra kod, förändra konfiguration eller bygga på fler komponenter.
21.	Original	Tid tas under tävling, och klockan ska stoppas då roboten har läst in ett sammanhängande rum och återvänt till "Garaget", mer precist då roboten stannat med alla fyra hjul inom rutan. Ett sammanhängande rum ska definieras som en yta helt innesluten i väggar.

22.	Original	Ett heat ska avslutas då klockan stoppas.
23.	Original	Resultatet ska vara på formen av ett rutnät . Varje cell ska representera en 40x40 centimeter cell i tävlingsområdet, och ska antingen vara definierad som omsluten av väggar eller öppen.

4. Diskvalifikation

24.	Original	Om robotens höjd överstiger väggarnas höjd ska roboten diskvalificeras från det pågående heatet.
25.	Original	Om roboten överskridet maxtiden 12 minuter ska roboten diskvalificeras från det pågående heatet.
26.	Original	Om gruppens system inte har visat en karta då roboten nått målet ska roboten diskvalificeras från det pågående heatet.
27.	Original	Om en gruppmedlem intar banan eller vidrör en robot innan det pågående heatet är slut så ska gruppens robot diskvalificeras från det pågående heatet.

5. Tidspålägg

28.	Original	Tidspålägg fås därefter vid någon eller fler av följande händelser. Specificerade antal sekunder i tidspålägg bestäms av beställarna.
29.	Original	Vidrörning av vägg ska ge ett tidspålägg på 20 sekunder. Endast de vertikala delarna av kartongväggarna ska räknas som en vägg.
30.	Original	Utritning av felaktig karta. Tidspålägg ska ges per felaktigt ritat block, å 30 sekunder. Vid fler än tre felaktiga block ska det istället för att ges tidspålägg per block ges ett enda tidspålägg på 200 sekunder.

6. Exempel på banutförande

