Designspecifikation

Jonatan Gezelius, Jacob Eriksson, Valentin Johansson, Rasmus Oliv,
Christoffer Johansson, Arshen Anwia

Version 0.2

Status

Granskad		

PROJEKTIDENTITET

Grupp 10, HT 2018, Explorer
Linköpings tekniska högskola, ISY

Namn	Ansvar	E-post
Jonatan Gezelius	Projektledare (PL)	pl@kartrobot.se
Jacob Eriksson	Dokumentansvarig (DOK)	dok@kartrobot.se
Valentin Johansson	Informationsansvarig (INFO)	info@kartrobot.se
Rasmus Oliv	Testansvarig (TST)	tst@kartrobot.se
Christoffer Johansson	Mjukvaruansvarig (MV)	mv@kartrobot.se
Arshen Anwia	Hårdvaruansvarig (HV)	hv@kartrobot.se

E-post arbetsgrupp: projektgrupp@kartrobot.se
Hemsida: https://www.kartrobot.se/

Kund: Kent Palmkvist

Kontaktperson hos kund: Kent Palmkvist, kent.palmkvist@liu.se

KURSANSVARIG: ANDERS NILSSON, ANDERS.P.NILSSON@LIU.SE
HANDLEDARE: OLOV ANDERSSON, OLOV.ANDERSSON@LIU.SE

Innehåll

Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2018-10-20	Första utkastet	Alla	
0.2	2018-10-30	Justerat struktur på kapitel 3. Rättat kontaktuppgifter till handledare. Kompletterat kapitel 3.5, 3.6. Justerat huvudenhetens kretsschema. Justerat styrenhetens kretsschema och JSP diagram. Justerat sensorenhetens kretsschema.	Alla	

1 Inledning

Systemet ska bestå av en robot som autonomt ska kunna kartlägga ett rum på ett effektivt sätt samtidigt som den presenterar miljön för användaren i ett uppritningsprogram på en separat enhet, se figur 1. Roboten skall även gå att styra manuellt ifrån den separata enheten. Den kommer sedan att delta i en tävling mot andra robotar för att utvärderas av kunden [1].

Figur 1 – En illustration av kartroboten som söker igenom rummet. PC:n presenterar kartan.

2 Systemöversikt

Systemet ska bestå av fyra delsystem enligt Figur 2. Delsystemen är:

- Huvudenhet
- Sensorenhet
- Styrenheten
- Styr och uppritningsprogrammet

2.1 Definitioner

SOUP -- Styr-och uppritningsprogrammet, körs på en PC.

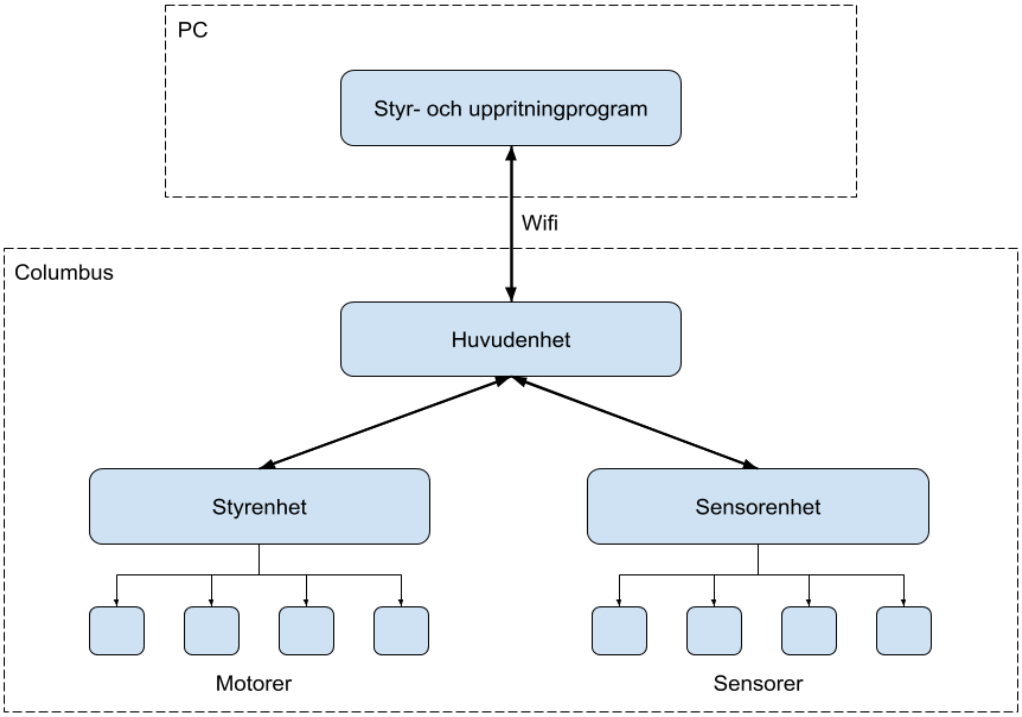
Columbus – Namnet på roboten som utforskar banan.

Raspberry Pi 3 – En enkortsdator [2]

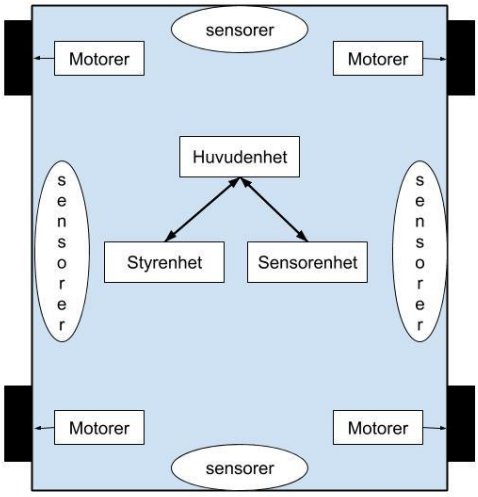
RPI – Se Raspberry Pi 3

2.2 Kommunikation

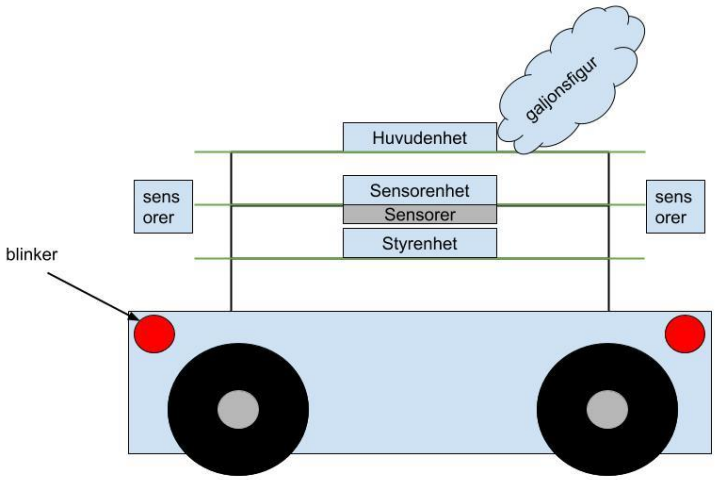
Kommunikationen mellan PC och Colombus kommer ske via TCP-sockets över Wifi. De tre enheterna på Colombus kommer kommunicera via UART mellan varandra. Huvudenheten kommer att använda en USB till UART adapter för att möjliggöra kommunikationen. Se specifikationen för bifogat kommunikationsprotokoll för mer detaljer angående kommunikationen mellan enheterna, Bilaga 1.



Figur 2 – Systemet består av Columbus och SOUP. Columbus består i sin tur utav en huvudenhet, en styrenhet och en sensorenhet.



Figur 3 – Övergripande bild på roboten ovanifrån. Sensorernas placering trimmas in vid utprovning för att erhålla ideal funktion.



Figur 4 – Övergripande bild på roboten från högra sidan. Galjonsfiguren designas i ett senare skede i mån av tid.

3 Delsystem 1 – Huvudenhet

Huvudenheten har till uppgift att autonomt styra Columbus med hjälp av styrenheten, baserat på data ifrån sensorenheten.

3.1 Delsystemets funktion

Huvudenheten kommer att bestå av en Rasbery Pi 3 som kör operativsystemet Linux. Styrningen kommer att ske antingen manuellt eller autonomt. Manuell styrning av Columbus sker från styr- och uppritningsprogrammet via en TCP-socket över en Wifi-uppkoppling. Huvudenheten kommer att agera accesspunkt för Wifi-nätet. Den autonoma styrningen sköts av mjukvaran i huvudenheten.

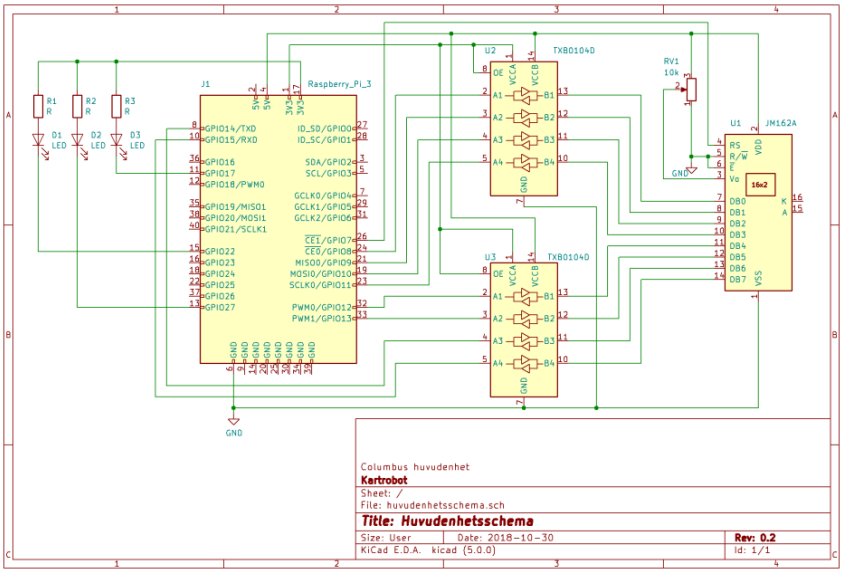
3.2 Extern hårdvara

För att kunna visa status finns det en 2x16 alfanumerisk display kopplad till enheten. Det finns även tre lysdioder som visar status för bland annat WiFi-anslutningen. Kommunikationen mellan styrenheten, sensorenheten och huvudenheten kommer att ske via USB med hjälp av två USB till UART omvandlare.
RPI:n har 28st in- och utgångar tillgängliga. Samtliga arbetar med 3.3 V logik, vilket gör det viktig att skydda ingångarna ifall 5 V logik används. Displayen som används arbetar med 5 V, för att skydda RPI:n används nivåskiftare mellan alla utgångar från displayen och RPI:n. Kopplingsschemat kan ses i figur 5.

Tabell 1: De komponenter som behövs för huvudenheten.

Komponenter	Beskrivning	Antal
J1	Raspberry Pi 3, Enkortsdator	1st

U1	LCD JM162A , Alfanumerisk display 2x16 tecken	1st
D1-D3	LED	3st
R1-R3	560 Ohm Resistor	3st
RV1	10k Linjär potentiometer	1st
U2,U3	3.3 – 5 V nivåskiftare	2st



Figur 5 – Huvudenhetens kopplingsschema. RPI:n är skyddad från displayens 5 V med hjälp av nivåskiftare.

3.3 Mjukvara

Huvudenhetens mjukvara kommer att bestå av flera delar som har olika ansvarsområden, se figur 7. Varje del kommunicerar med någon eller några av de andra delarna med specificerade interface.

En del kommer att initiera programmet, hantera programflödet och sköta kommunikationen mot Pc:n med hjälp av det kommunikationsprotokoll som fastställts enligt Bilaga 1.

3.4 Simultaneous Localization And Mapping

SLAM är ett koncept som går ut på att en agent som gör sig i ett okänt rum kartlägger detta samtidigt som den använder kartan till att beräkna sin position i rummet. Detta görs utifrån den sensordata som agenten kan samla in i kombination med tidigare kända positioner. Agenten använder därefter vissa speciella punkter för att analysera sin position och hitta tidigare kända punkter i kartan. Dessa punkter kallas landmarks och används även till att kunna stänga loopar i rummet. SLAM kräver av agenten att dess sensordata är filtrerad för eventuella fel med en tillräckligt hög precision.

3.5 Styralgoritmen

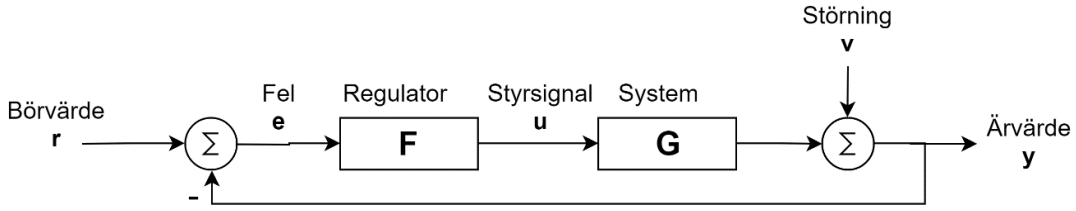
Systemet styrs antingen via manuella kommandon eller en autonom algoritm. I manuellt läge fås styrsignalerna från styr- och uppritnings programmet som ger Columbus instruktioner om vilken riktning och i vilken hastighet som en förflyttning ska ske. När Columbus är i manuellt läge kommer

SLAM–algoritmen fortfarande att uppskatta Columbus position i förhållande till rummets utseende men den kommer att följa användarens instruktioner utan att ta hänsyn till väggar och andra hinder. I autonomt läge använder Columbus informationen om position- och kartdata som fås av SLAM-algoritmen. Utifrån den informationen kan Columbus söka av den interna kartan efter okända punkter. Dessa punkter kommer att viktas utifrån de handlingar som Columbus måste utföra för att kunna ta sig till den plats där en okänd punkt kan avsökas. När en okänd punkt har hittats kommer de förflyttningar som behöver utföras för att ta sig till en punkt där det okända området kan avsökas att köas. Under tiden som förflyttningarna utförs kommer den position som Columbus förväntar sig att vara på jämföras med den position som SLAM uppger som den faktiska positionen för att kunna avbryta vägen och rensa kön om dessa inte överensstämmer. Columbus kommer sedan att korrigera sin position och räkna ut en ny väg för att avsöka målet.

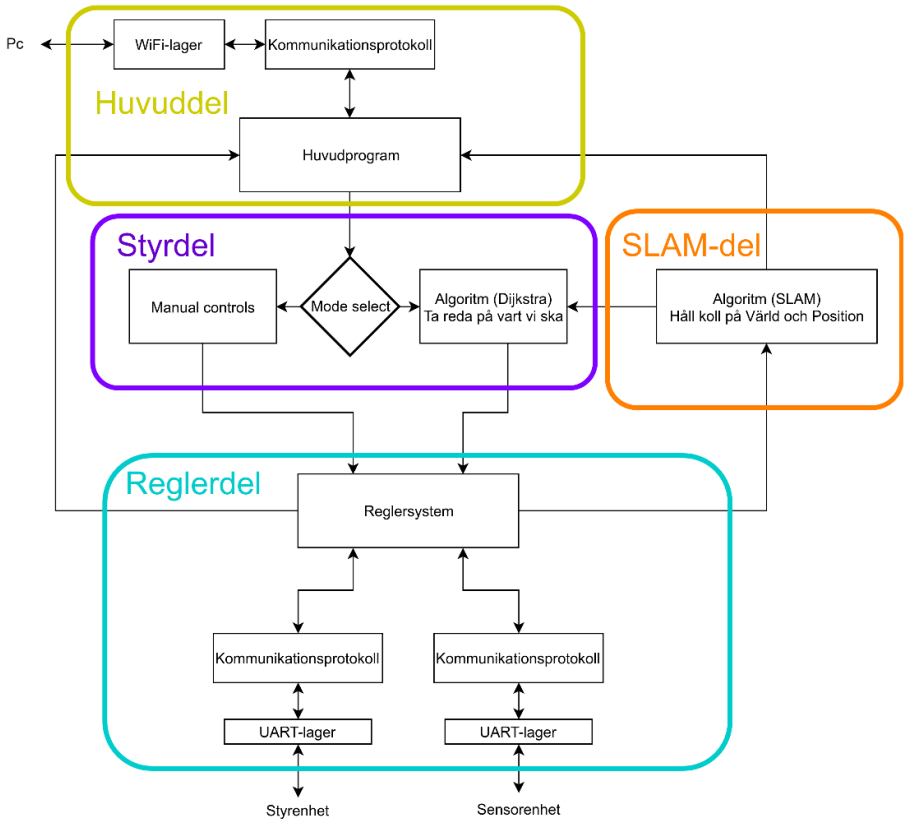
3.6 Reglersystem

Reglersystemet kommer vara den del som har direkt kontakt mot styr- och sensorenheterna via UART. Det har till uppgift att baserat på direktiv från styrdelen se till att roboten rör sig enligt dessa direktiv. Reglersystemet kommer kontinuerligt att jämföra ärvärde mot börvärde och korrigera styrsignalerna utifrån reglerfelet, ett så kallat *closed loop system*, se figur 6.

Eftersom Columbus skall kunna röra sig på olika sätt så kommer olika sensorer användas vid de olika rörelserna. Detta kommer att lösas genom att ha ett antal olika reglerloopar som hanterar olika insignaler med hjälp av olika regulatorer att användas. Reglersystemet skall ha ett läge för att köra rakt i en bestämd hastighet, ett för att vrida roboten på stället och ett för att svänga och köra samtidigt. En regulator kommer att vara någon variant av PID regulator i de fall där detta är en rimlig lösning, till exempel då roboten skall köra rakt längs med en vägg.



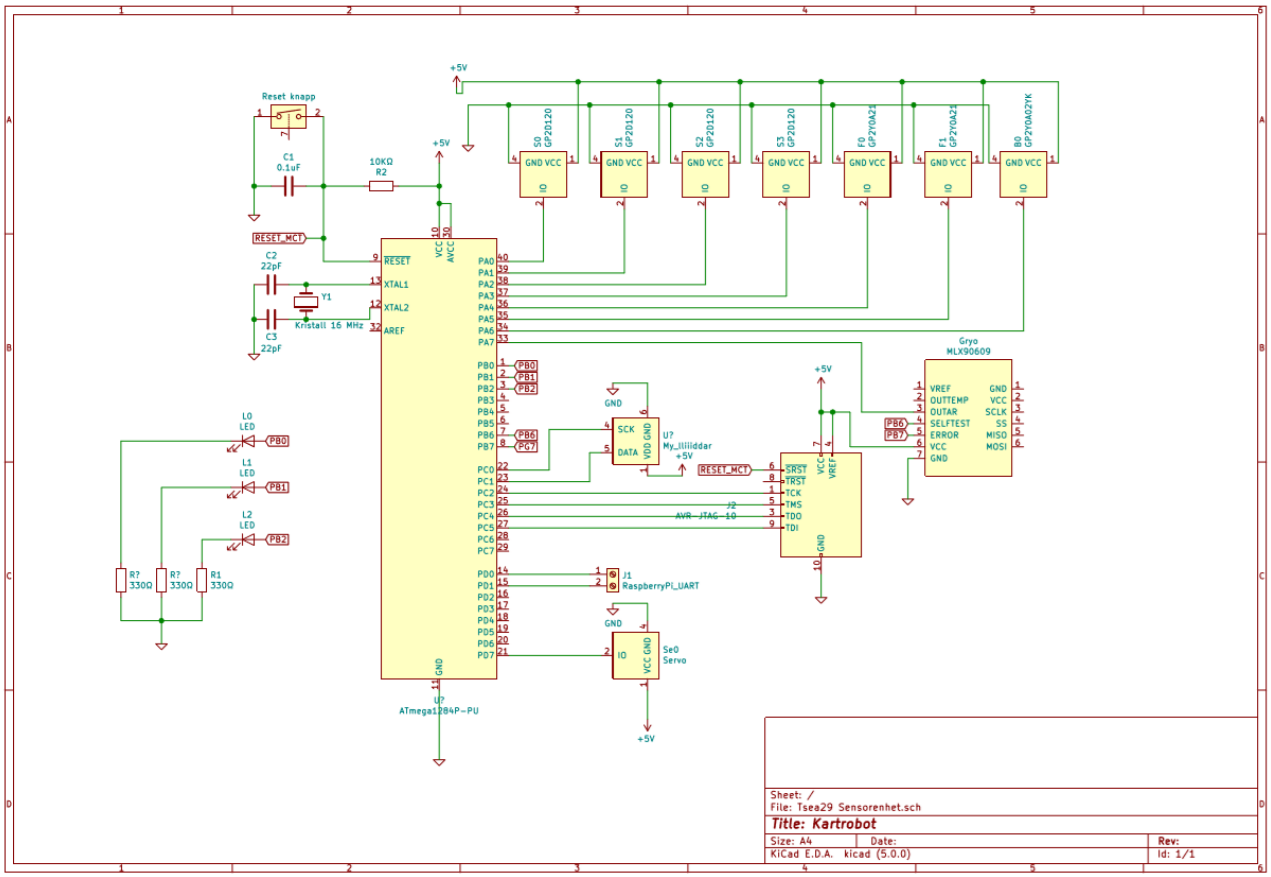
Figur 6 – Reglersystemen som används kommer i möjligast mån att vara återkopplade, och reglerade med PID.



Figur 7 – Struktur över huvudenhetens mjukvara.

4 Delsystem 2 – Sensorenhet

Detta delsystem har som uppgift att hantera och behandla sensorernas data samt kommunicera med huvudmodulen. Kopplingsschema för enheten illustreras i figur 8.



Figur 8 – Kretsschema över sensorenheten.

4.1 Delsystemets funktion

Sensorenheten kommer bestå av en atmega1284 processor samt nio sensorer. Åtta av sensorerna är avståndssensorer som används för att mäta avstånd till närliggande väggar, den sista är ett gyro som används för att mäta vinkelhastigheten. Informationen från samtliga sensorer behandlas sedan, innan det skickas vidare till huvudenheten. Kommunikationen med huvudenheten beskrivs Bilaga 1.

4.2 Sensorer

Sensorer som systemet ska använda:

GP2D120 4st: Optiska avståndsmätare som ska sitta på sidan av roboten, används för att kunna hålla roboten på rak kurs längs en vägg. Analog utsignal som beskriver avståndet med en olinjär kurva.

Lidar Lite V3 1st: Laseravståndsmätare som ska sitta på ett 180° servo riktad framåt på roboten, används för att se väggar längre ifrån roboten. Kommunicerar via I2C, olinjär under 1 meter.

GP2Y0A21 2st: Optiska avståndsmätare ska sitta riktade framåt, används för att komplettera Lidar sensorn vid kortare avstånd. Analog utsignal som beskriver avståndet med en olinjär kurva.

GP2Y0A02YK 1st: Optisk avståndsmätare som ska sitta bakåt. Analog utsignal som beskriver avståndet med en olinjär kurva.

MLX90609 1st: Vinkelhastighetssensor(gyro) som ska sitta i mitten av roboten, används för att mäta hur snabbt roboten roterar. Den analoga och linjära utsignalen kommer användas.

4.3 Sensordata

Avståndssensorernas data kommer att läsas av ett antal gånger för att sedan ta fram ett medelvärde innan de omvandlas till längdenheter. Vid omvandlingen till längdenheter måste varje sensors olinjäritet tas hänsyn till. Detta görs för att minska felmarginalen på avståndet. Den hanterade informationen skickas sedan vidare till huvudenheten. Mätvärden ifrån gyrot läses kontinuerligt av för att på begäran skickas vidare till huvudenheten.

4.4 Strömbrytare

På sensormodulen ska det finnas strömbrytare som skickar informationen till huvudenheten. Dessa strömbrytare skall göra följande:

- Startknapp, som startar igång roboten.
- Auto/man, här ska man kunna välja mellan ett manuellt- eller autonomt läge.

Signalerna skickas vidare till huvudenheten som tar beslut baserat på detta.

4.5 Lampor

Sensorenheten kommer att ha statusindikationer i form av lampor, för att se ifall allt fungerar som det ska.

4.6 Komponentlista

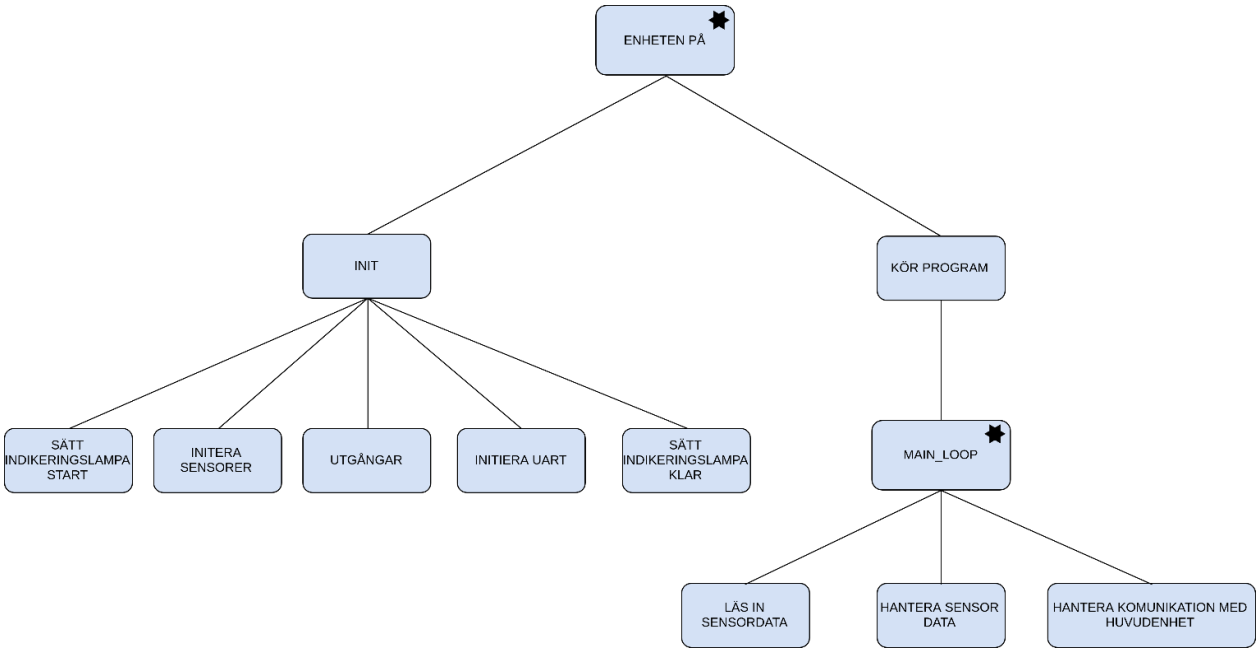
Tabell 2: De komponenter som behövs för sensorenheten.

Komponenter	Beskrivning	Antal
ATmega1284P-PU	Mikrokontroller	1st
Kristall	16 MHz	1st
Kondensator	22 pF	2st
Kondensator	0,1 uF	1st
Resistor	1 KΩ	1st
Resistor	330 Ω	3st
Återfjädrande tryckknapp		1st
Virkort		1st
Röd diod		1st
Grön diod		2st
Robotplattform Terminator		1st

GP2D120	Optisk avståndsmätare, 4–30 cm	4st
GP2Y0A21	Optisk avståndsmätare, 10–80 cm	2st
GP2Y0A02YK	Optisk avståndsmätare, 20–150 cm	1st
Lidar Lite V3	Laseravståndsmätare, 0-40m	1st
Servo		1st
MLX90609	Vinkelhastighetssensor (gyro), 300 grader/s	1st

4.7 JSP-diagram

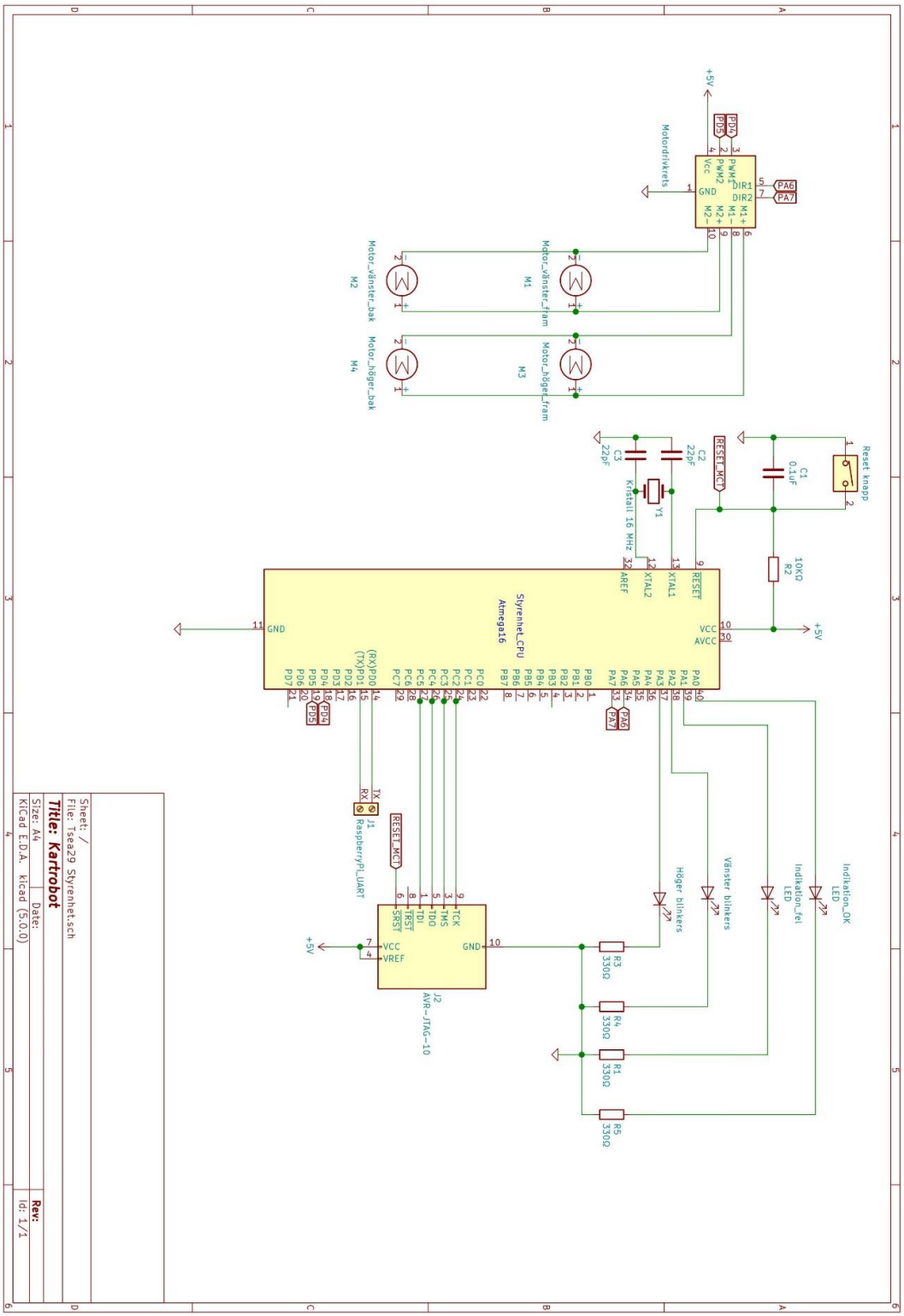
Här beskrivs hur sensorenhetsprogrammet ska vara uppbyggt enligt figur 9.



Figur 9: Ett översiktlig JSP-diagram över sensorenhetsprogrammet.

5 Delsystem 3 – Styrenhet

Styrenheten skall ta emot information från huvudenheten och med hjälp av det styra motorerna och de lampor som finns i systemet, kopplingsschema för styrenheten illustreras i figur 10.



Figur 10 – Kretsschema över styrmodulen.

5.1 Delsystemets funktion

Styrenheten kommer bestå av en ATmega16 processor som är kopplad till en motordrivkrets, som i sin tur är kopplad till motorer. Från huvudenheten kommer paket med kommandon att skickas till styrenheten, som exempelvis meddelar hur motorerna ska styras. Alla paket som går att skicka finns i ett givet protokollsdokument, Bilaga 1.

5.2 Motorer

Systemets fyra 7,2 V DC-motorer kommer att styras parvis. För att ändra hastighet på robotens hjul på höger sida skickas signal till motorparet på höger sida och samma sak gäller för vänster sida. Genom att ha olika hastighet på höger respektive vänster motorpar kommer roboten kunna svänga. Motorernas hastighet kommer att regleras med pulsbreddsmodulering och deras riktning kommer att ställas in med en styrsignal för vardera motorpar på motordrivkretsen.

5.3 Drivelektronik

För att kunna styra motorerna med en mikrokontroller krävs drivelektronik, eftersom mikrokontrollen inte klarar av att leverera tillräckligt med ström till motorerna. Dessutom använder motorerna och mikrokontrollerna olika spänningsnivåer. Drivelektroniken för detta sitter redan i robotplattformen som projektet kommer byggas utifrån.

5.4 Lampor

Systemet kommer att innehålla lampor för att indikera systemets status och för att signalera när roboten ska svänga. Roboten kommer även utrustas med en del dekorativ belysning.

5.5 Komponenter

Tabell 3: De komponenter som kommer behövas för styrenheten.

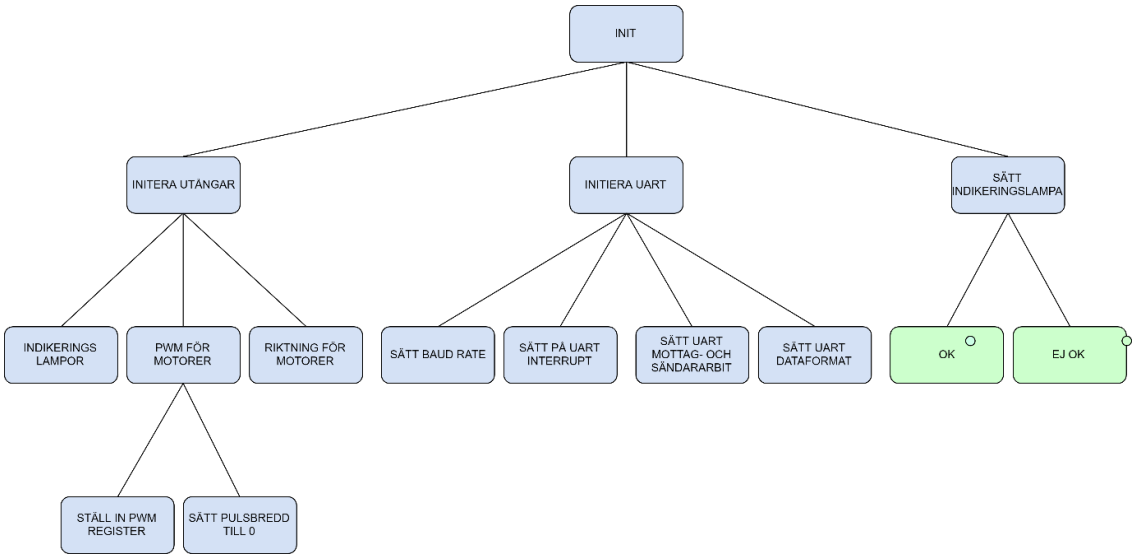
Komponenter	**Antal**
ATmega16	1st
16 MHz kristall	1st
Kondensator 22 pF	2st
Kondensator 0,1 uF	1st
Resistor 1 KΩ	1st
Resistor 330 Ω	4st
Återfjädrande tryckknapp	1st
Virkort	1st
Röd diod	1st
Grön diod	3st
Robotplattform Terminator	1st
Kopplingsplint med 2 anslutningar	1st

5.6 JSP-diagram

Under det här kapitlet beskriv hur styrenhetensmodulensprogram kommer var uppbyggt med hjälp av JSP-diagram.

5.6.1 Initiering

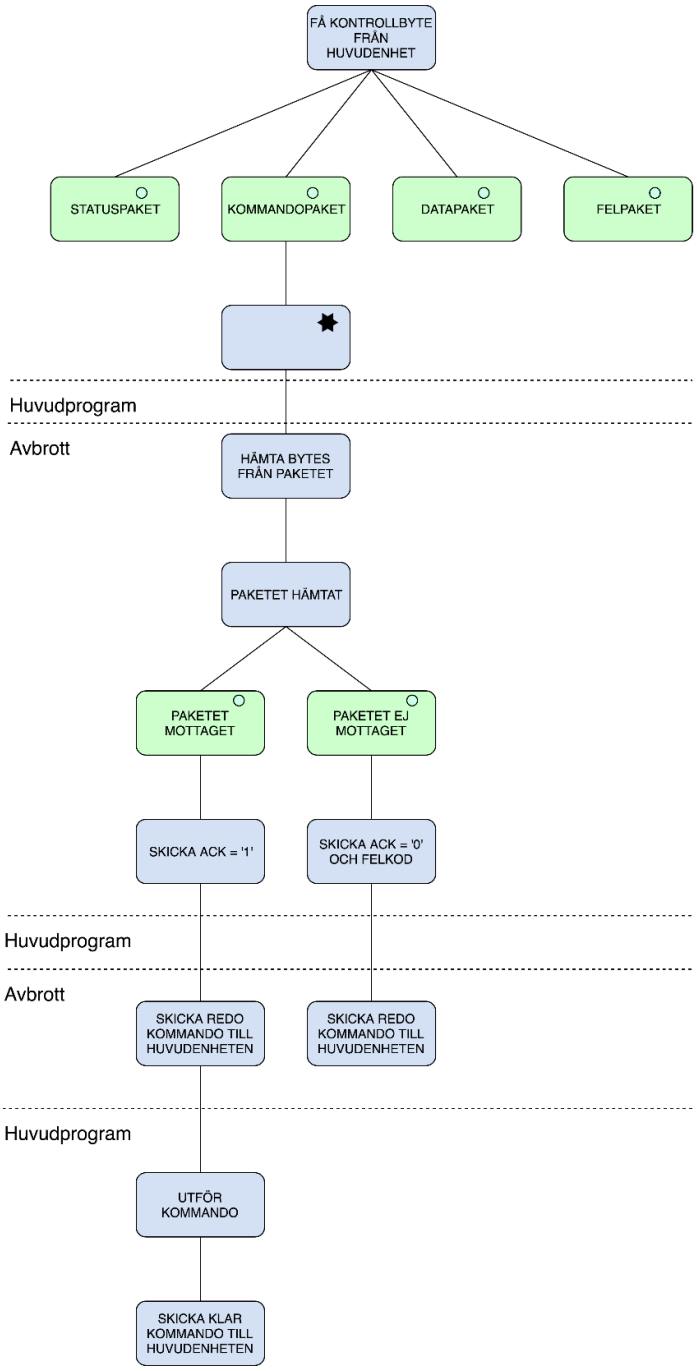
JSP-diagrammet i figur 11 visar hur initieringen för styrenhetens hårdvara ska utföras.



Figur 11: JSP-diagram över initiering för styrenheten

5.6.2 Ta emot data från huvudenhet

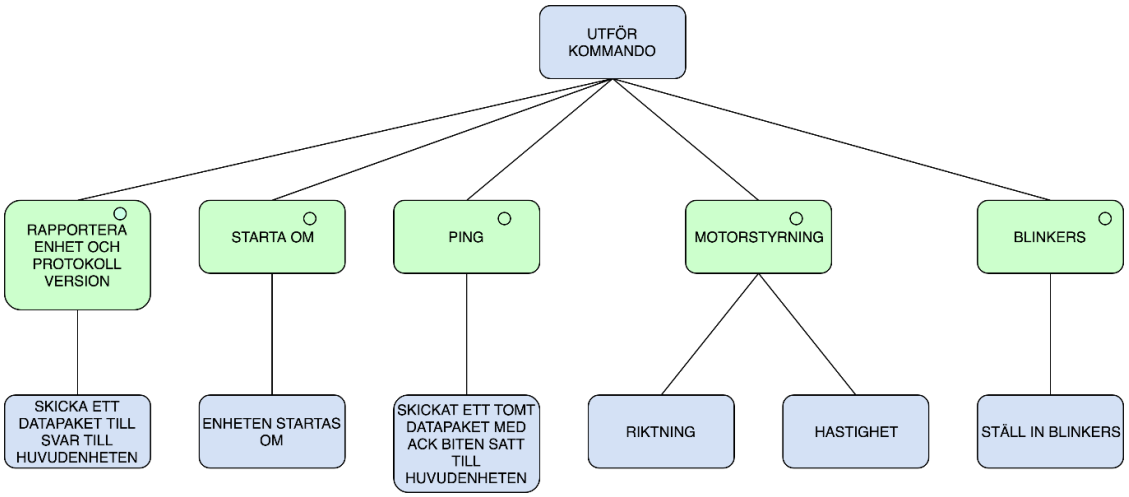
För att kunna ta emot data från huvudenheten kommer styrenheten alltid ta emot ett kontrollbyte först via UART från huvudenheten. Efter det kommer styrenheten hoppa till en avbrottsrutin som beskrivs i JSP-diagrammet i figur 12 nedan.



Figur 12: JSP-diagram över mottagande av data från huvudenhet.

5.6.3 Utföra ett mottaget kommando

En mer detaljerad beskrivning av hur ett kommando utförs av styrenheten illustreras i figur 13 nedan.



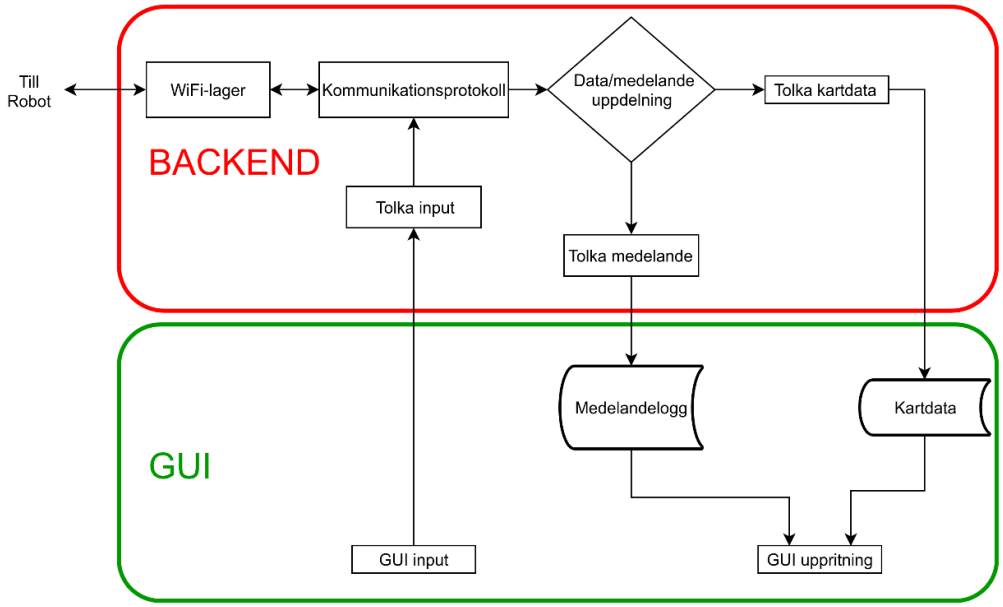
Figur 13: JSP-diagram över kommandon som går att utföra.

6 Delsystem 4 – Styr och uppritnings programmet

Styr och uppritnings programmet, SOUP, ska ta emot datapaket från roboten som beskriver kartan och rita upp denna. Programmet kommer även att kunna tolka användarinput som skickas i form av kommandon till roboten samt ta emot statusrapporter ifrån roboten.

6.1 Mjukvara

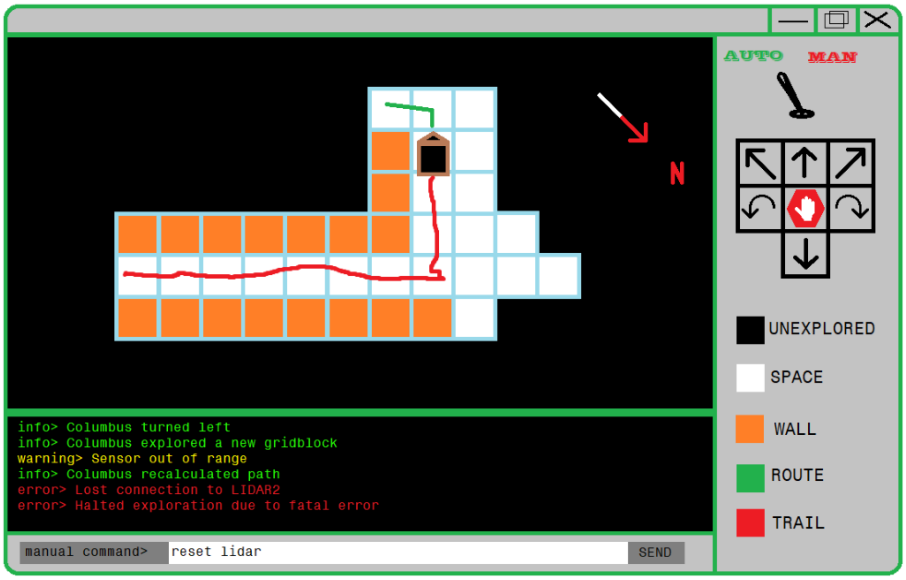
SOUP kommer att bestå av en del som hanterar det grafiska användargränssnittet och ett så kallat backend som hanterar all data och kommunikation, se figur 14. De olika grafiska komponenterna kommer att kopplas mot de bakomliggande funktionerna för att hantera data. Delar delar av programmet kommer att nyttja funktionalitet från ett framework som heter QT [3].



Figur 14: SOUP är uppdelat i ett grafiskt användargränssnitt och ett backend.

6.2 Gränssnitt

Programmet kommer att ha ett grafiskt gränssnitt som ska visa en tolkning av kartan utifrån ett ovanifrånperspektiv. Det grafiska användargränssnittet använder sig av QT för att rita fönster och andra grafiska komponenter. Robotens status skall skrivas ut i ett kommandofält under kartan, se figur 15. Det grafiska gränssnittet skall även indikera om roboten körs i autonomt- eller manuellt läge, samt de kommandon som roboten använder för att köra manuellt.



Figur 15 – Konceptbild för styr- och uppritningsprogrammet där Columbus världsbild och logg visas. Det går även att styra Columbus med den manuella kontrollpanel.

6.3 Kommunikation

SOUP kommer att kommunicera med roboten via ett WiFi-nät som roboten har skapat. Kommunikationen över nätverket kommer ske via en TCP socket med hjälp av QT. Vid begäran av programmet kommer data för robotens lagrade karta och nuvarande mål samt den väg som roboten har tagit från sin startposition att skickas från roboten. Programmet skall även skicka kommandon för hur roboten skall förflytta sig på kartan genom kommandon från ett tangentbord eller via det grafiska interfacet i de fall då roboten körs i manuellt läge.

Referenser

[1] Kartrobbot Grupper, ”Spelregler - TSEA29,” Kartrobbot Grupper, 2018. [Online]. Available: http://www.kartrobot.se/documents/banspecifikation_v.1.0.pdf. [Använd 30 10 2018].

[2] ” raspberrypi.org,” 30 10 2018. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[3] ”QT bibloteket,” [Online]. Available: https://www.qt.io/ . [Använd 30 10 2018].

BILAGA 1

1 Features

Protokollet som används mellan modulerna på Columbus har många angenäma egenskaper.

- Paketbaserat
- Utökningsbart
- Hantering av borttappade paket
- Bättre än postnord
- Kommunikation kan inledas från vilken av de två enheterna som helst
- Upp till 32 bytes kan skickas per paket

2 Användningsområde

Protokollet är framtaget för att användas med UART, men kan fungera även för andra tvåvägs punkt-till-punkt interface.

3 Översikt

All kommunikation sker med så kallade paket. Ett paket inleds alltid av en kontrollbyte. Kontrollbyten bestämmer vilken sorts paket det är. Det kan vara ett statuspaket, ett kommandopaket, ett datapaket eller ett felpaket.

Ett kommandopaket och ett datapaket följs alltid av en byte, men kan följas av ytterligare 32 bytes. Detta bestäms också utav kontrollbyten. Tomma paket och felpaket består endast av en kontrollbyte.

När alla delar av ett kommandopaket eller ett datapaket mottagits korrekt måste det besvaras med en kontrollbyte som har flaggan ACK satt. Även detta paket kan innehålla ett kommando eller data.

Innehåll

4 Dokumenthistorik

Version	Datum	Utförda ändringar	Utförda av	Granskad
v0.1	2018-10-07	Första versionen av specifikationen. Påbörjar beskrivning av protokoll v0.1	JG	
v0.2	2018-10-14	Fyllt i fler kommandon för protokoll v0.1	Alla	
v0.3	2018-10-30	Fyllt på fler kommandon. Färdigställt version 0.1 av protokollet.	Alla	

5 Definitioner

SOUP Styr- och uppritningsprogrammet

6 Kontrollbyte

Kontrollbyten är den första byten som sänds i ett paket. Den bestämmer vilket slags paket som skickas och indikerar även acknoledge när det är relevant. Om det är ett kommandopaket eller ett datapaket som skickas så anger den hur många extra bytes som skickas efter kontrollbyten. Om det är ett felpaket så indikerar den vilket fel som uppstått.

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	ACK	T1	T0	S4 L4 E4	S3 L3 E3	S2 L2 E2	S1 L1 E1	S0 L0 E0

ACK

ACK biten indikerar att det senaste paketet togs emot i sin fulla längd.

ACK	**Beskrivning**
0	Detta är inte ett svar på en lyckad mottagning av ett paket.
1	Detta är en bekräftelse på att det senaste paketet togs emot i sin fulla längd.

T1:T0

T1 och T0 anger tillsammans vilket sorts paket som skickas.

T1:T0	**Beskrivning**
00	Statuspaket.
01	Kommandopaket.
10	Datapaket.
11	Felpaket.

S4:S0 / L4:L0 / E4:E0

Denna del av kontrollbyten har olika betydelse beroende på vilken sorts paket det är. Om det är ett statuspaket så anger bitarna en status. Om det är ett kommando- eller datapaket så anger bitarna hur många extra efterföljande bytes som kommer skickas. Om det är ett felpaket så anger det en felkod.

T1:T0	**Bit 4:0**	**Funktion**
00	S4:S0	Anger en status. Se Kapitel 7 för detaljerad beskrivning.
01	L4:L0	Anger hur många bytes som skickas efter den första kommandobyten.
10	L4:L0	Anger hur många bytes som skickas efter den första databyten.
11	E4:E0	Anger vilket fel som rapporteras. Se Kapitel 9 för detaljerad beskrivning.

6.1 Statuspaket

Statuspaketen är främst avsedda till att kunna förmedla ett acknowledge för ett tidigare mottaget paket. De kan dock användas för att informera mottagaren om olika saker med hjälp av den statuskod som skickas med.

Se Kapitel 7 för en komplett lista över statuskoder.

6.2 Kommandopaket

Ett kommandopaket indikerar att efterföljande byte, eller bytes, utgör ett eller flera kommandon till mottagaren. Det skickas alltid åtminstone en byte efter kontrollbyten i ett kommandopaket.

För att se vilka kommandon som finns, och vilka eventuella argument eller svar de väntar sig, se Kapitel 8.

6.3 Datapaket

Datapaket är bara ren data. Kan endast skickas som svar på tidigare kommando, som kräver data som svar.

6.4 Felpaket

Ett felpaket kan indikera flera olika fel. Se Kapitel 9 för detaljerad beskrivning av felkoder.

7 Statuskoder

S4:S0	**Beskrivning**	**Från version**
0x00	Tomt paket.	0.1
0x01	Redo.	0.1
0x02	Klar.	0.1
0x03	System OK	
0x04 – 0x1F	Reserverade statuskoder.	

7.1 [0x00] — Tomt paket

Ett tomt paket används för att kunna förmedla ett paket med ACK-biten satt, utan att behöva förmedla ytterligare information.

7.2 [0x01] — Redo

Denna kod indikerar att mottagaren är redo för ny data eller för nya kommandon.

7.3 [0x02] — Klar

Denna kod indikerar att mottagaren har fullföljt ett kommando.

7.4 [0x03] — System OK

Denna kod indikerar att det sändande systemet gjort en systemkontroll och att inga fel påträffades.

8 Kommandon

Det här kapitlet beskriver i detalj vilka kommandon som finns, vilka argument de skickar och vilken eventuell data som måste skickas som svar. Då fler bytes skickas i ett paket skickas det första kommandot först, följt av eventuella parametrar, sedan nästa kommando, med eventuella parametrar, och så vidare.

Exemplet visar ordningen för bytes då flera kommandon skickas i samma paket

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Kontrollbyte	Kommando 0	Parameter 0	Parameter 1	Kommando 1	Parameter 0	Kommando 2	Kommando 3

8.1 Sammanfattning av kommandon

Kommando	Antal argument	Beskrivning	Från version
Generella kommandon			
0x00	0	Rapportera enhet och protokollversion.	0.1
0x01	0	Starta om.	0.1
0x02	0	Ping.	0.1
0x03	0	Utför systemtest	0.1
0x04	0	Skickar enhetsspecifika felkoder	0.1
0x05	0	Skickar enhetsspecifika infokoder	0.1
0x06 – 0x3F	-	Reserverade generella kommandon.	
Styrenhetskommandon			
0x40	1	Styrkommando	0.1
0x41	1	Blinkerskommando	0.1
0x42 – 0x7F	-	Reserverade styrenhetskommandon.	
Sensorenhetskommandon			
0x80	0	Lidar fram	0.1
0x81	0	Sensorer fram	0.1
0x82	0	Sensorer vänster	0.1
0x83	0	Sensorer höger	0.1
0x84	0	Sensor bak	0.1
0x85	0	Gyro	0.1
0x86	0	All sensordata	0.1
0x87	0	Lidar scan	0.1
0x88 – 0xBF	-	Reserverade sensorenhetskommandon.	
Huvudenhetskommandon			
0xC0	-	Styr Fram	0.1
0xC1	-	Styr Bak	0.1
0xC2	-	Rotera Höger	0.1
0xC3	-	Rotera Vänster	0.1
0xC4	-	Styr Fram höger	0.1
0xC5	-	Styr Fram vänster	0.1
0xC6	-	Stopp	0.1
0xC7	0	Begär nuvarande kartdata	0.1
0xC8	0	Begär kontinuerlig kartdata vid uppdatering	0.1
0xC9	0	Avbryt kontinuerlig kartdata vid uppdatering	0.1
0xCA	0	Begär färdväg	0.1
0xCB	0	Begär historik	0.1
0xCC	0	Utför huvudenhetstest	0.1
0xCD	0	Utför styrenhetstest	0.1
0xCE	0	Utför sensorenhetstest	0.1
0xCF	0	Skickar uppdaterad kartdata	0.1
0xD0	-	Skickar loggrad	0.1
0xD1	-	Påbörja eller återuppta autonom körning	0.1
0xD2	-	Pausa autonom körning	0.1
0xD3	-	Avbryt autonom körning	0.1
0xD4	-	Rensa kartdata	0.1
0xD5 – 0xFF	-	Reserverade SOUP-kommandon.	

8.2 Generella kommandon

8.2.1 [0x00] — Rapportera enhet och protokollversion

	Antal	Typ	Beskrivning
Argument	0		
Svar	1	Data	[version]

Det här kommandot begär enhetstyp och protokollversion ifrån den mottagande enheten. Svaret skickas i ett datapaket med en byte.

Svarsbyte 0

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
	U2	U1	U0	M1	M0	s2	s1	s0

U1:U0

Enhetstypen anges med U1:U0.

U2:U0	**Typ av enhet**	**Från version**
000	Huvudenhet.	0.1
001	Styrenhet.	0.1
010	Sensorenhet.	0.1
011-111	Reserverade för framtida enheter.	

M1:M0

Anger major version av protokollet. Den här specifikationen gäller protokollet med major version 0.

s1:s0

Anger minor version av protokollet. Den här specifikationen gäller protokollet med major version 1.

8.2.2 [0x01] — Starta om

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Begär den mottagande enheten att starta om.

8.2.3 [0x02] — Ping

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Ett kommando som finns bara för att kunna få ett ACK-meddelande tillbaks.

8.2.4 [0x03] — Utför systemtest

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Utför test på egen och skicka vidare samma kommando till eventuella underliggande enheter

Svarar med statuskod SYSTEM OK ifall alla tester var ok, eller Enhetsspecifik Felkod ifall ett eller fler delsystem var NOK.

Svaret har ACK satt.

8.2.5 [0x04] — Skickar enhetsspecifika felkoder

	Antal	Typ	Beskrivning
Argument	1 – n		[felkod 1] ... [felkod n]
Svar	0	Status	ACK

Argument1: samma som svarsbyte0 i protokollversion

Argument2-x: enhetsspecifika felkoder

Svar: Enbart Tomt ACK-paket

8.2.6 [0x05] — Skickar enhetsspecifika infokoder

	Antal	Typ	Beskrivning
Argument	1 – n		[infokod 1] ... [infokod n]
Svar	0	Status	ACK

Argument1: samma som svarsbyte0 i protokollversion

Argument2-x: enhetsspecifika infokoder

Svar: Enbart Tomt ACK-paket

8.3 Styrenhetskommandon

Det här kapitlet kommer behandla de möjliga kommandon som ska skickas mellan styr- och huvudenheten.

8.3.1 [0x40] — Styrkommando

	Antal	Typ	Beskrivning
Argument	1		[riktning och hastighet]
Svar	0	Status	ACK

Kommandot anger rotationsriktning och hastighet för motorerna.

Bit	7	6	5	4	3	2	1	0
	RRV	HV2	HV1	HV0	RRH	HH2	HH1	HH0

Bit 7:0	Beskrivning
RRV	Rotationsrikning på vänster hjulpar.
RRH	Rotationsrikning på höger hjulpar.
HV2:HV0	Hastighetsbitar för vänster hjulpar.
HH2:HH0	Hastighetsbitar för höger hjulpar

Hastighetsbitarna för vänster och höger hjulpar består av 3 bitar vilken kan ge 8 olika hastigheter för hjulparen.

8.3.2 [0x41] — Blinkerskommando

	Antal	Typ	Beskrivning
Argument	1		[blinkers på/av]
Svar	0	Status	ACK

Kommando för att aktivera blinkers.

Bit	7	6	5	4	3	2	1	0
	X	X	X	X	X	X	VB	HB

Bit 1:0	Beskrivning
VB	Aktivera vänster blinkers.
HB	Aktivera höger blinkers.

8.4 Sensorenhetskommandon

Det här kapitlet kommer behandla de möjliga kommandon som ska skickas mellan sensor- och huvudenheten.

8.4.1 [0x80] — Lidar fram

Begär sensordatan från sensorn Lidar Lite v3 i neutralt läge. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.2 [0x81] — Sensorer fram

Begär sensordatan från de två framåtriktade GP2Y0A21 sensorerna. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.3 [0x82] — Sensorer vänster

Begär sensordatan från de två vänsterriktade GP2D120 sensorerna. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.4 [0x83] — Sensorer höger

Begär sensordatan från de två högerriktade GP2D120 sensorerna. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.5 [0x84] — Sensorer bak

Begär sensordatan från den bakåtriktade GP2Y0A02YK sensorn. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.6 [0x85] — Gyro

Begär sensordatan från gyrot MLX90609. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.7 [0x86] — All sensordata

Begär sensordatan från samtliga sensorer. 0 argument.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.4.8 [0x87] — Lidar scan

Begär att en svepning och mätning utförs av Lidar Lite v3 sensorn.

Svar: Databytes ända till klar, då skickas Statuspaketet KLAR.

8.5 Huvudenhetskommandon

8.5.1 [0xC0] — Styr fram

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.2 [0xC1] — Styr bak

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.3 [0xC2] — Rotera höger

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.4 [0xC3] — Rotera vänster

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.5 [0xC4] — Styr fram höger

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.6 [0xC5] — Styr fram vänster

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.7 [0xC6] — Stopp

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Beordrar huvudenheten att stanna roboten.

8.5.8 [0xC7] — Begär nuvarande kartdata

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Begär hela kartdatan ifrån huvudenheten.

Svar: Databytes ända till klar, då skickas Statuspaket KLAR

8.5.9 [0xC8] — Begär kontinuerlig kartdata

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Begär att huvudenheten skall skicka de delar av kartan som uppdateras, när de uppdateras. Uppdaterad kartdata skickas med kommandot [0xCF] — Leverera uppdaterad kartdata.

8.5.10 [0xC9] — Avbryt kontinuerlig kartdata

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Begär att huvudenheten skall sluta skicka uppdateringar om kartan.

8.5.11 [0xCA] — Begär färdväg

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Inga argument

Svar: Databytes ända till klar, då skickas Statuspaket KLAR

8.5.12 [0xCB] — Begär historik

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Inga argument

Svar: Databytes ända till klar, då skickas Statuspaket KLAR

8.5.13 [0xCC] — Uför huvudenhetstest

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Ber huvudenhet att utföra systemtest.

8.5.14 [0xCD] — Utför styrenhetstest

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Ber styrenheten att utföra systemtest.

8.5.15 [0xCE] — Utför sensorenhetstest

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Ber sensorenheten att utföra systemtest.

8.5.16 [0xCF] — Leverera uppdaterad kartdata

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Huvudenhet → SOUP

Inga argument. Följs av databytes ända till klar, då skickas Statuspaket KLAR

8.5.17 [0xD0] — Skickar loggrad

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

Huvudenhet → SOUP

Inga argument. Följs av databytes (ascii) ända till klar, då skickas Statuspaket KLAR

8.5.18 [0xD1] — Påbörja eller återuppta autonom körning

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.19 [0xD2] — Pausa autonom körning

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.20 [0xD3] — Avbryt autonom körning

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

8.5.21 [0xD4] — Rensa kartdata

	Antal	Typ	Beskrivning
Argument	0		
Svar	0	Status	ACK

9 Felkoder

E4:E0	Beskrivning	Från version
0x00	Fel antal bytes mottagna.	0.1
0x01	Kommandot hanteras ej.	0.1
0x02	Korrupt kontrollbyte.	0.1
0x03	Systemet upptaget.	0.1
0x04 – 0x1F	Reserverade felkoder.	

9.1 [0x00] — Fel antal bytes mottagna

Antalet bytes som togs emot stämmer inte överens med vad kontrollbyten angav.

9.2 [0x01] — Kommandot hanteras ej

Något av kommandona i det senaste paketet som enheten tog emot stöds inte.

9.3 [0x02] — Korrupt kontrollbyte

Den senast mottagna kontrollbyten hade inte ett giltigt värde.

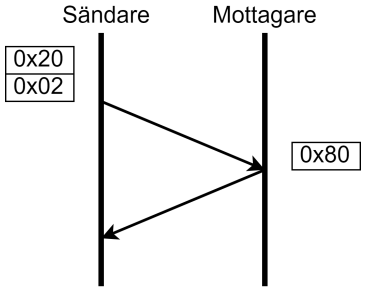
9.4 [0x03] — Systemet upptaget

Den mottagande enheten kunde inte ta emot det senaste paketet på grund av att det är upptaget.

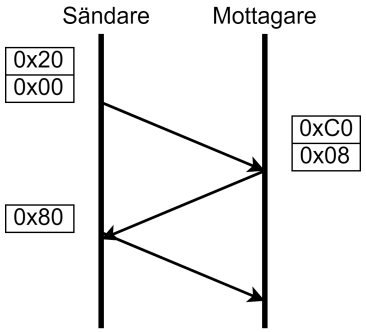
10 Exempel

För att klargöra hur en kommunikation skulle kunna se ut följer här ett antal exempel.

10.1 Ett kommando utan argument och utan svarsbyte



10.2 Ett kommando utan argument, med svarsbyte



10.3 Okänt kommando