DESIGNSPECIFIKATION

Oskar Lundin

Version 1.0

Status

Granskad	Oskar Lundin	2019-11-04
Godkänd		

PROJEKTIDENTITET

Grupp 5, 2019/HT 1, Columbus
Linköpings tekniska högskola, ISY

Namn	Ansvar	Telefon	E-post
Mattias Ljung	Kommunikationsenhetsansvarig (KA)	070-219 03 53	matlj387@student.liu.se
Felix Lindgren	Dokumentansvarig (DA)	013-22 33 44	felli675@student.liu.se
Marcus Nolkrantz	Styrenhetsansvarig (SA)	070-553 48 79	marno874@student.liu.se
Justus Karlsson	Grafiskenhetsansvarig (GA)	072-241 43 77	juska933@student.liu.se
Edwin Johansson	Sensorenhetsansvarig (SEA)	073-673 39 87	edwjo109@student.liu.se
Oskar Lundin	Projektledare (PL)	070-756 80 58	osklu414@student.liu.se

Hemsida: https://gitlab.liu.se/osklu414/tsea29-kartrobot

Kund: Kent Palmkvist, 581 00 LINKÖPING,
013-28 13 47, kent.palmkvist@liu.se

Kursansvarig: Anders Nilsson, 3B:512, 013-28 2635, anders.p.nilsson@liu.se
Handledare: Petter Källström, 013-28-14-92, petter.kallstrom@liu.se

Innehåll

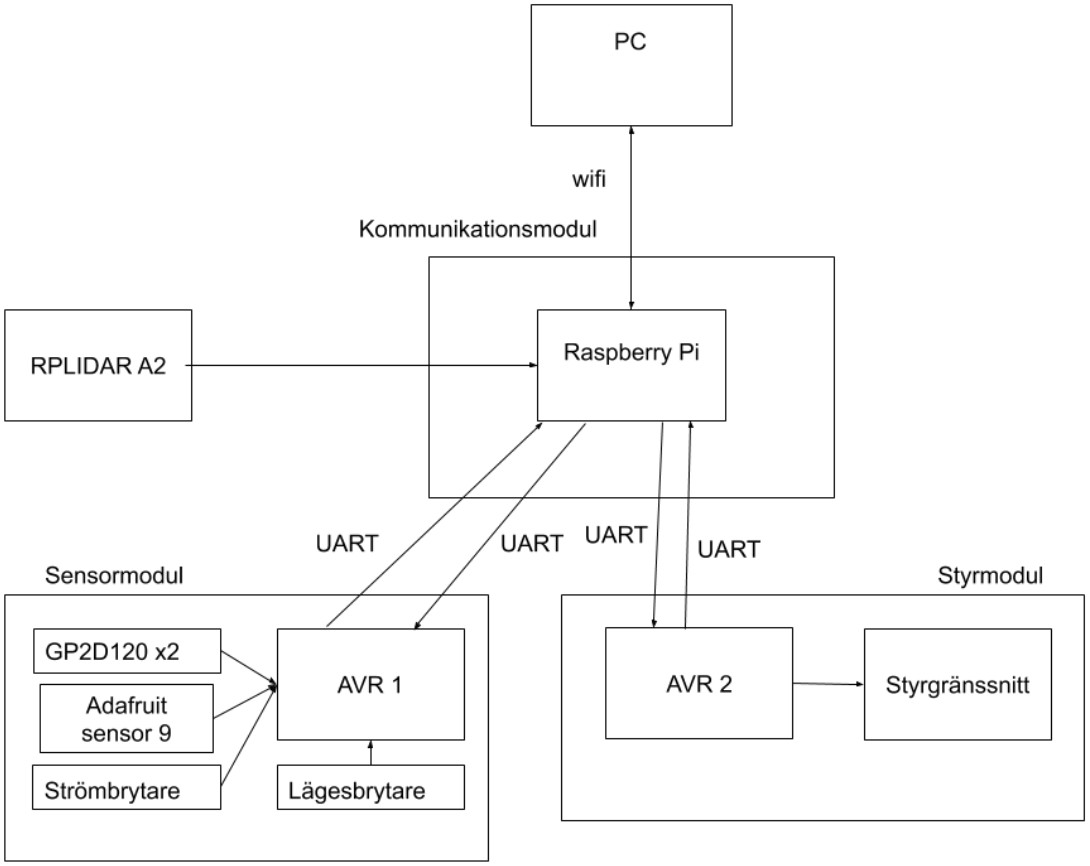
Dokumenthistorik

Version	Datum	Utförda förändringar	Utförda av	Granskad
1.0	2019-11-04	Första versionen	ML, FL, JK, MN, EJ, OL	OL
0.2	2019-10-31	Andra utkastet	ML, FL, JK, MN, EJ, OL	OL
0.1	2019-10-09	Första utkastet	ML, FL, JK, MN, EJ, OL	OL

1 Sammanfattning

Den här designspecifikationen innehåller detaljerad information om hur kartrobotens moduler är implementerade samt hur robotens mjukvara ska konstrueras. Komponenter, kopplingar, flödesschema och gränssnitt beskrivs för varje modul.

Roboten kommer kunna placeras på en okänd bana, autonomt söka sig runt hela området och konstruera en karta som kan visas på en extern dator. Alternativt ska roboten kunna styras manuellt från en extern dator och köras runt fritt. Roboten kommer använda sig av en Rasberry Pi för att kommunicera mellan datorn och de olika mikroprocessorer som styr eller läser från och till motorerna och sensorerna. Chassit kommer med fyra stycken DC-motorer kopplade till hjulen, för att kunna söka ordentligt används en RPLidar för att hitta väggar och hjälpa med navigation, två avståndssensorer åt vänster och höger så att roboten åker rakt samt en sensor med både gyroskop och accelerometer för att bestämma riktning och hastighet m.m.



Figur 1: Blockschema kartrobotens moduler

2 Beskrivning av delsystem

Det färdiga systemet kommer att bestå av fyra stycken delsystem: en styrmodul, en sensormodul, en kommunikationsmodul och en PC programvara. Detta avsnitt beskriver detaljerat egenskaperna för dessa delsystem.

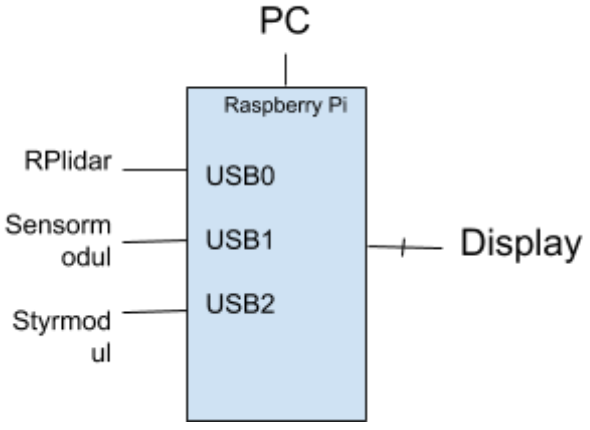
2.1 Kommunikationsmodul

Kommunikationsmodulen ska sammanställa all kartdata och skicka den till PC:n via Wi-Fi. I autonomt läge får den in data från sensormodulen som den använder för att bestämma vilken väg roboten ska ta. Den skickar sedan signaler till styrmodulen för att få roboten att röra på sig.

2.1.1 Datastrukturer

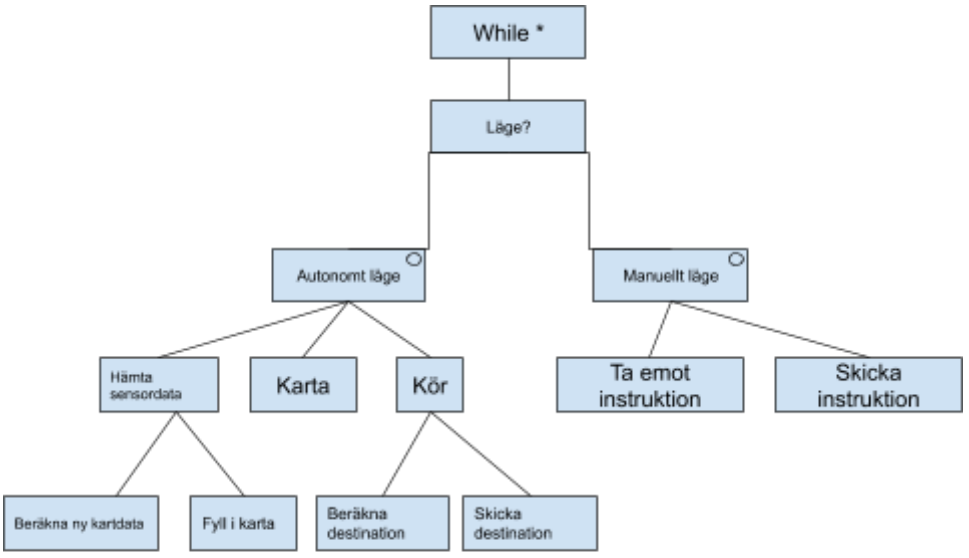
Sensor_communication_transmission - innehåller all sensordata från sensormodulen.
Position - en punkt på en vägg från sensordatan.
Robot - robotens nuvarande position för att ha en referenspunkt till punkterna i sensordatan.
Tile - ett värde på varje $(40cm)^2$ ruta på kartan.
Map - 25x25 array med Tiles för att representera kartan.

2.1.2 Kopplingsschema



Figur 2: Kopplingsschema kommunikationsmodul.

2.1.3 JSP-diagram



Figur 3: JSP-diagram till kommunikationsmodul.

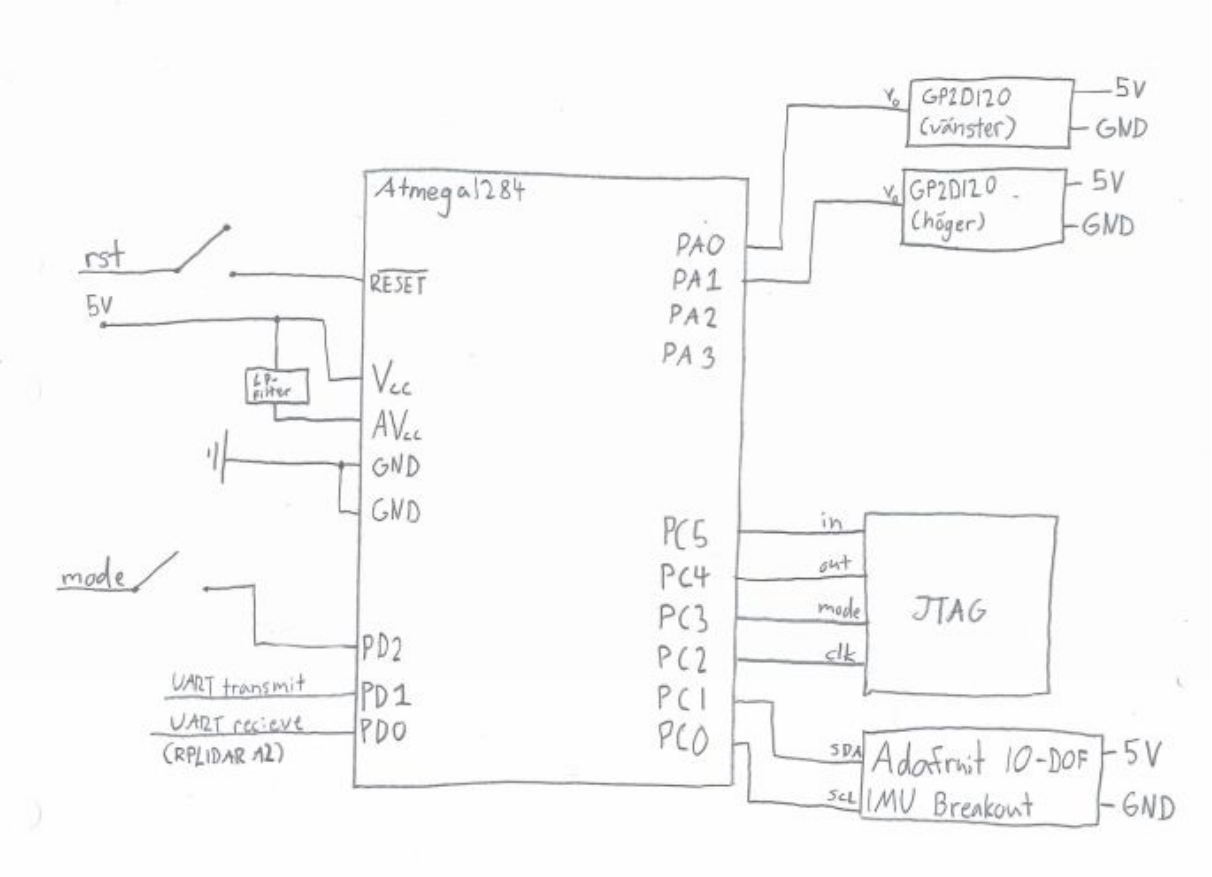
I autonomt läge används till en början "köra höger"-algoritmen. I steget "Beräkna destination" i diagrammet ovan väljs alltså alltid en position så att roboten följer väggen till höger om sig. I mån av tid utbyts denna algoritm till en bättre.

2.1.4 Komponenter

Kommunikationen mellan mikroprocessorerna och Rasberry Pi kommer ske med UART på USB-kablar.

Komponent	**Antal**
Rasberry Pi 3	1
USB kabel	2

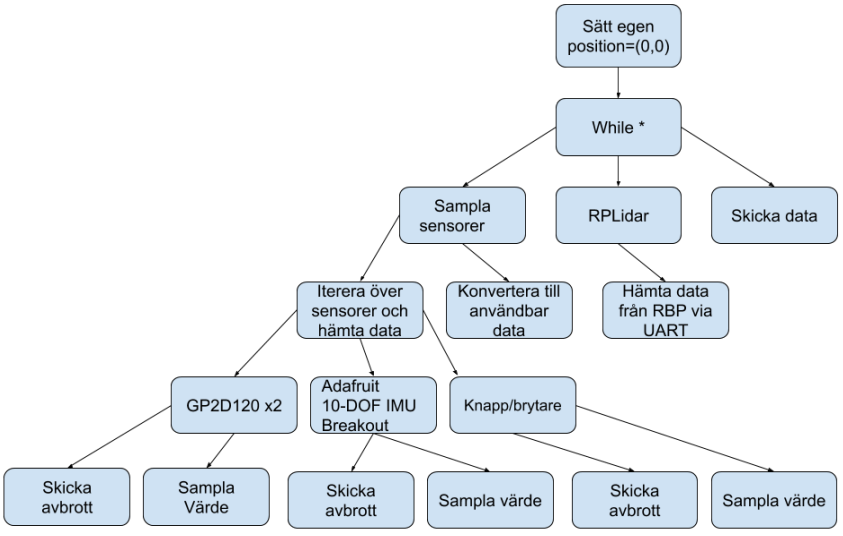
Tabell 1: Komponenter kommunikationsmodul.



2.2 Sensormodul

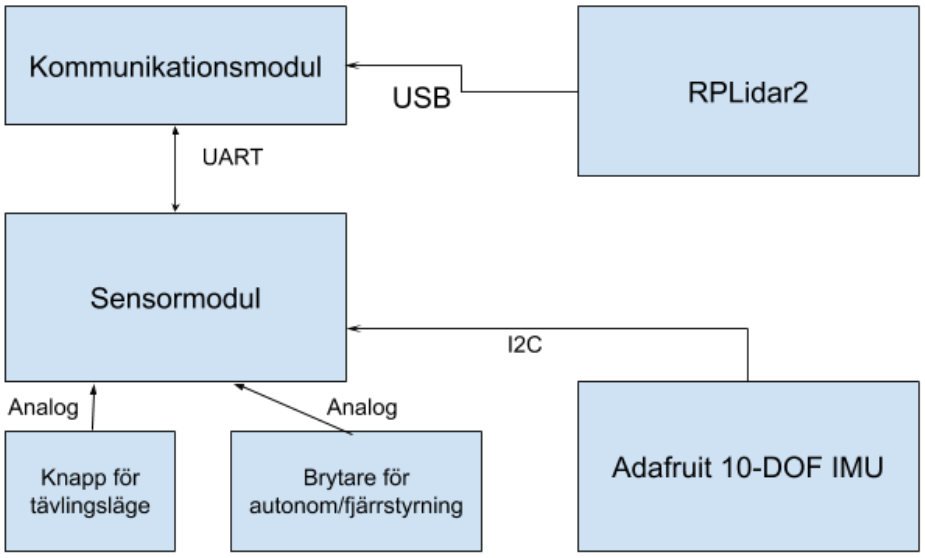
Sensormodulens uppgift är att läsa in sensordata, konvertera den till användbar information och vidarebefordra den till kommunikationsmodulen.

2.2.1 JSP-Diagram



Figur 4: JSP-diagram för sensormodul.

2.2.2 Blockschema



Figur 5: Blockschema sensormodul

2.2.3 Kopplingsschema

Figur 4: Kopplingsschema sensormodul.

2.2.4 Komponenter

Komponent	**Antal**
Atmel AVR (processor)	1
RPLIDAR A2 (avståndssensor)	1
GP2D120 (avståndssensor)	2
Adafruit 10-DOF IMU Breakout (gyroskop + accelerometer)	1
Strömbrytare	2

Tabell 2: Komponenter sensormodul.

2.2.4.1 GP2D120 (avståndssensor)

Två av dessa sensorer finns på robotens vänstra och högra sidan. Syftet med dessa avståndssensorer är att mäta avstånd till väggarna på sidorna av roboten, så att den kan åka i en rak linje. Datan från dessa sensorer används i styrenheten för reglerteknik. Sensormodulen får data från dessa sensorer i form av en analog signal, mellan -0,3V och 5,3V. Utsignalen går till pin PA0 (vänster sensor) och PA1 (höger sensor). Då sambandet mellan avstånd och spänning på utsignalen är olinjär kommer en matematisk funktion tas fram under projektets gång som gör sambandet linjärt.

2.2.4.2 RPLIDAR A2 (avståndssensor, 360 grader)

Denna IR-sensor finns ovanpå roboten. Den har möjlighet att scanna 360 grader, så den används till att mappa banan. På grund av att det finns en risk att sensorn går sönder om den kopplas in slarvigt i en AVR kopplas den istället till Raspberry Pi:n (kommunikationsmodulen). Kommunikationsmodulen vidarebefordrar direkt över den råa sensordatan till sensormodulen via UART, där datan behandlas.

2.2.4.3 Adafruit 10-DOF IMU Breakout (gyroskop + accelerometer)

Denna sensor består bl.a. av ett gyroskop och accelerometer, som kommer användas i roboten för att styra roboten åt vänster och höger och att möta hastighet. Hastigheten mäts genom att integrera över accelerationen. När roboten står still nollställer vi de ackumulerade felen som uppstått vid integrering. Sensorn är placerad på ovansidan av roboten. Kommunikation med sensorn sker via I2C. Insignalen SCL på sensorn kopplas till pin PC0 på AVR:en och utsignalen SDA kopplas till pin PC1.

2.2.4.4 Strömbrytare (man/auto, reset)

Ena strömbrytaren används för att byta mellan autonomt och manuellt läge. I manuellt ska en användare kunna styra roboten externt från en dator medans i autonomt läge ska roboten åka runt och försöka rita upp en karta av sin omgivning. Kopplas till pin PD2.

Den andra strömbrytaren används för att starta om sensormodulens AVR. Kopplas till pin RESET. Den kopplas även till styrmodulens AVR och kommunikationsmodulens Raspberry Pi i deras reset-pin.

2.3 Styrmodul

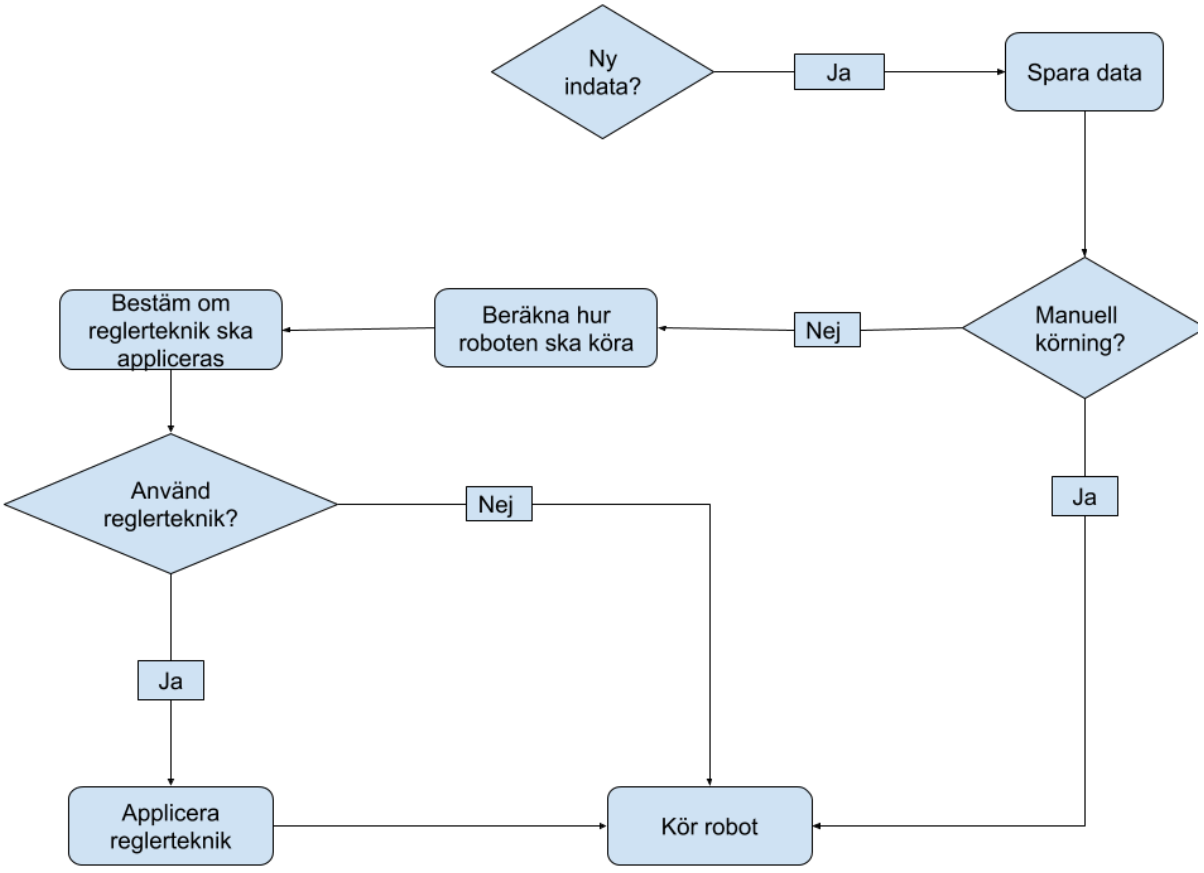
Styrmodulen består av en AVR1284 anluten till två motorer, en för höger- och en för vänster hjulpar. Via en UART skickas seriell data, från kommunikationsenheten till AVRen, som representerar instruktioner för hur roboten ska röra sig. AVRen tolkar datan och konverterar den till signaler som skickas vidare till hjulparen, en signal motsvarande rotationsriktning och en pulsbreddsmodulerad signal motsvarande hastighet. Under konverteringen appliceras även reglerteknik för att robotens svängningar ska bli mer dämpade och den ska kunna köra rakt.

2.3.1 Datastrukturer

Följande datastrukturer kommer användas för att lagra data.

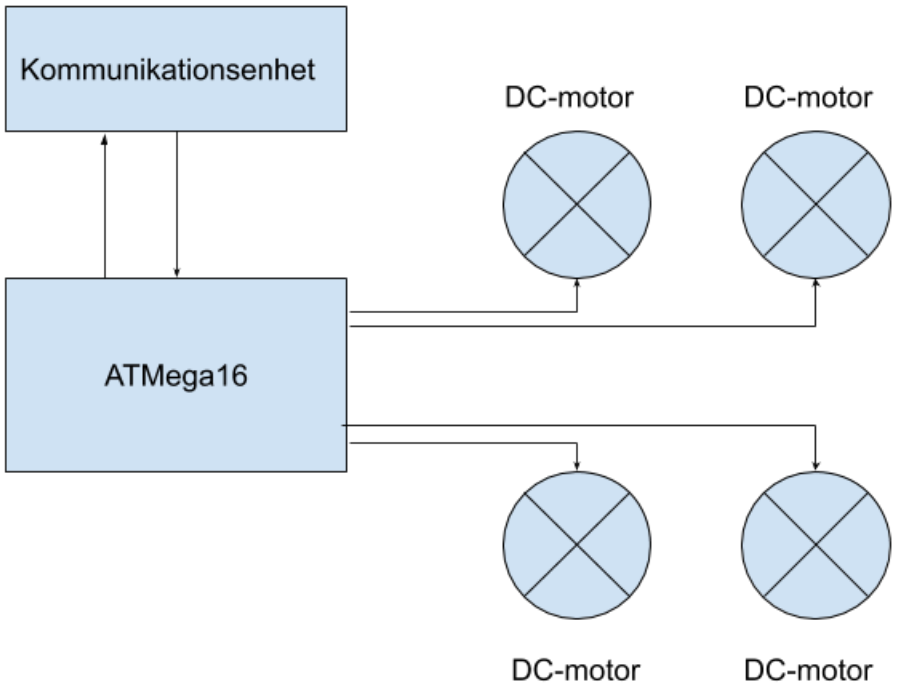
- En boolean motsvarande om roboten styrs manuellt eller autonomt.
- En struct som lagrar robotens nuvarande rotation, nuvarande position i x-led och nuvarande position i y-led samt robotens önskade rotation, önskade position i x-led och önskade position i y-led.
- En struct som lagrar konstanter som ska användas vid reglerteknik beräkningar.
- En enum som motsvarar vilket kommando roboten ska utföra.

2.3.2 Flödesschema



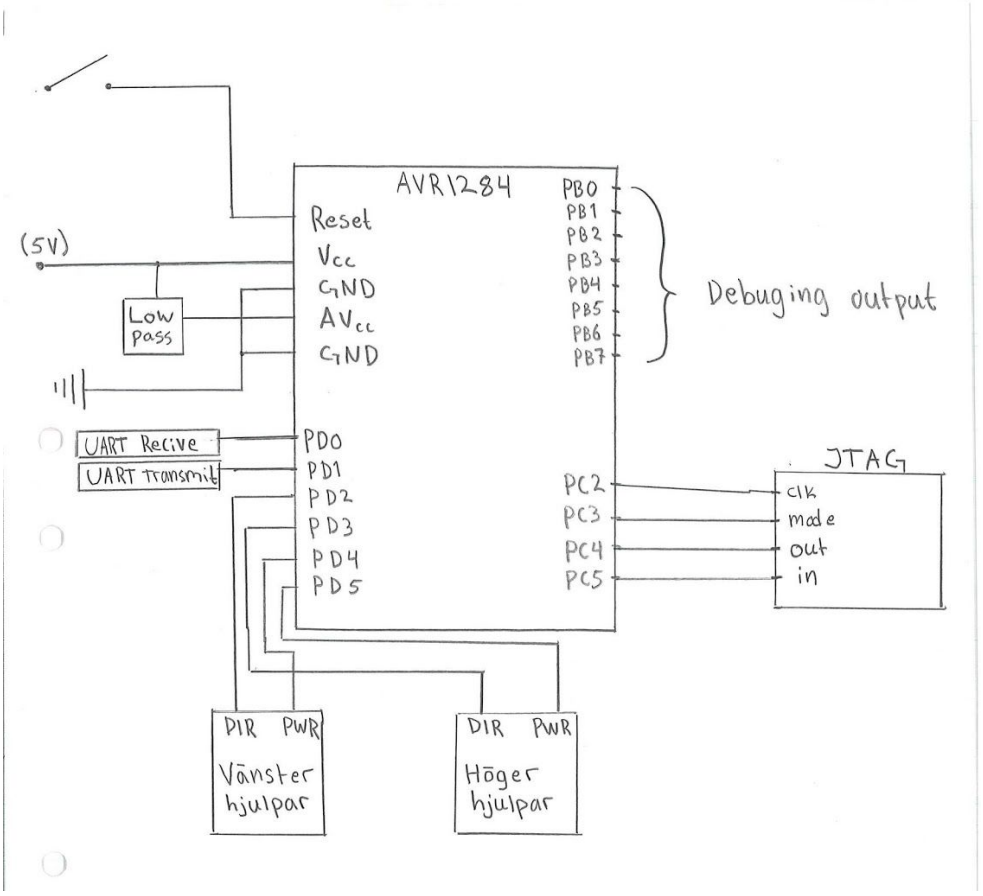
Figur 6: Flödesschema styrmodul.

2.3.3 Blockschema



Figur 7: Blockschema styrmodul

2.3.4 Kopplingsschema



Figur 8: Kopplingsschema styrmodul.

2.3.5 Komponenter

Komponent	**Antal**
Atmel AVR1284 (processor)	1
Hjulpar	2
Lågpassfilter	1

Tabell 3: Komponenter styrmodul.

2.4 PC programvara

Programvaran ligger på den externa PC:n och finns inte på själva roboten. Den är ansvarig för att rita upp en grafisk representation av banan, föra en logg över sensordata samt fjärrstyrning av roboten. Kommunikation mellan roboten och PC:n sker via wifi. Robotens kommunikationsmodul upprättar en Wifi Access Point som PC:n kopplar upp sig till. Kraven på PC:n är följande:

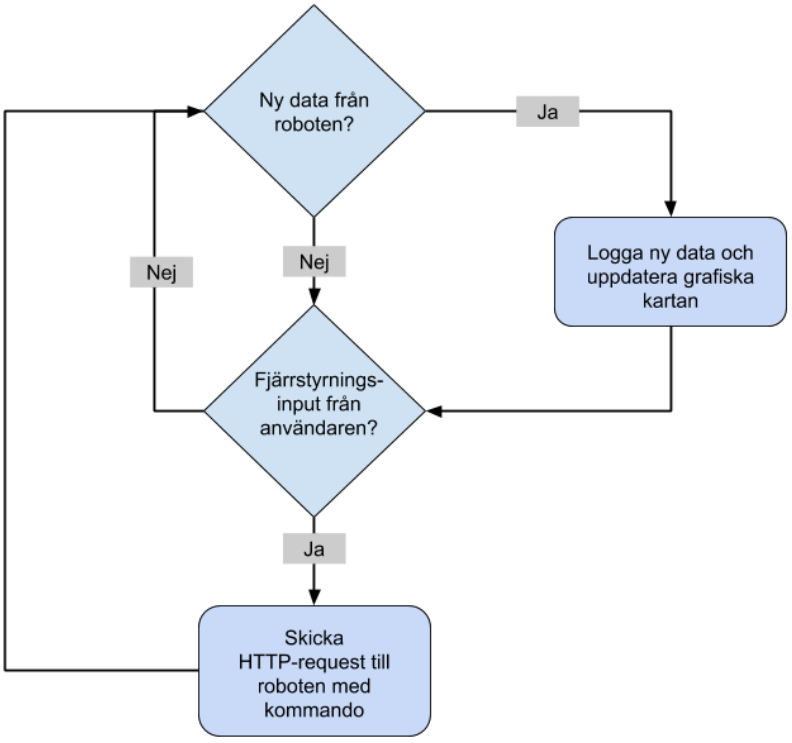
- Nätverkskort för wifi
- Python-version >= 3.4

2.4.1 Design och funktionalitet

Robotens raspberry pi kommer starta upp en HTTP-server på port 8000. Kommunikation sker sedan via HTTP-requests mellan enheterna. Data från PC:n kommer skickas som en JSON-formaterad sträng i requestens body. Roboten kommer skicka tillbaka ett svar med data JSON-formaterad i en sträng.

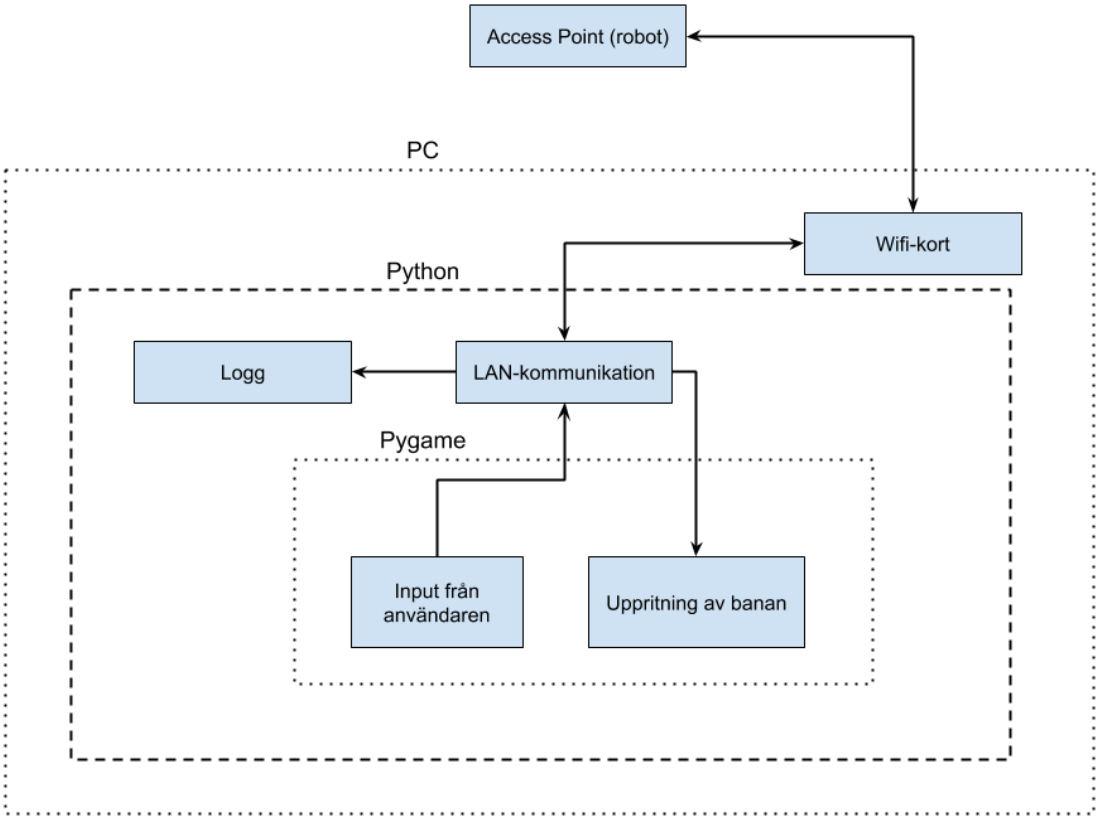
Roboten kommer ha följande routes:

- "/map" - PC:n begär en uppdaterad uppskattning av hur banan ser ut. Datan används för att rita upp en grafisk representation av banan.
- "/robot" - PC:n begär data om robotens nuvarande tillstånd (position och rotation). Roboten ritas sedan ut i den grafiska representation av banan.
- "/data" - PC:n begär styrdata och sensordata för att kunna föra en logg.
- "/remote-control" - Roboten tar emot fjärrstyrningskommandon från PC:n. Om fjärrstyrning ej är aktiverad kommer requesten ignoreras.



Figur 9: Programflöde PC-programvaran.

2.4.2 Struktur



Figur 10: PC-programvarans olika delar.

Programmet på PC:n kommer vara skrivet i python. Python-paketet pygame kommer användas för att rita upp banan och roboten grafiskt. Pygame är en spelmotor och är därför prestandamässigt väl lämpad till att rita upp den dynamiska banan och roboten. För att upptäcka tangentbordstryck vid fjärrstyrning kommer också pygame användas eftersom paketet har smidiga funktioner för detta.

3 Kommunikation mellan moduler

3.1 Gränssnitt

Samtlig kommunikation mellan moduler sker via UART. Mellan kommunikationsmodul och styrmodul samt mellan kommunikationsmodul och sensormodul går all kommunikation i båda riktningarna, så läget *full duplex* används. Datahastigheten är 115200 Bd vilket bör vara tillräckligt för de krav som ställs på systemet.

Samtliga överföringar kontrolleras. Om en överföringstyp inte stämmer överens med dess data kastas den bort. Då *full duplex* används har vi möjlighet att eventuellt verifiera data och begära nya sändningar. Det kan också användas för att felsöka.

När roboten startar skickar sensormodulen och styrmodulen information om sig själva så att kommunikationsmodulen kan avgöra vilken som är vilken.

3.2 Enheter

Som längdenhet används millimeter i samtliga moduler. Då de största avstånden som behandlas är 10 meter (enligt projektdirektiven[1]) kan en endimensionell position representeras som ett 16 bitars heltal. Vinklar representeras i grader mellan 0 och 359. För detta används ett 16 bitars naturligt tal.

3.3 Gemensamma datastrukturer

Följande pseudokod visar de datastrukturer som är gemensamma för robotens olika moduler. Om till exempel en position ska skickas mellan två moduler förväntas följa dessa strukturer. En

En tvådimensionell position (x, y) representeras av två 16 bitars heltal. Samtliga positioner är relativa robotens ursprungliga position.

```
struct position
{
    u16 x
    u16 y
}
```

Robotens tillstånd består av en position och en rotation.

```
struct robot
{
    position pos
    u16 rot
}
```

Kartan representeras som en tvådimensionell matris av så kallade tiles. En tile kan vara en vägg, tom eller okänd. Detta format används när information om kartan skickas till PC:n. Det används också internt i kommunikationsmodulen.

```
enum tile : u8
{
    UNKNOWN
    WALL
    EMPTY
}


struct map
{
    tile tiles[25][25]
}
```

Mellan kommunikationsmodulen och sensormodulen skickas datan som mäts av RP-Lidar:n.

```
struct communication_sensor_transmission
{
    u16 angle
    u16 distance
    u8 quality
    u8 flag
}
```

Mellan sensormodulen och kommunikationsmodulen skickas information från samtliga sensorer i omgångar. Varje meddelande innehåller robotens nuvarande position och rotation, samt alla mätningar som gjorts sedan förra meddelandet. Varje mätning är positionen av en identifierad vägg. Allt mellan roboten och varje mätning kan alltså implicit tolkas som luft, EMPTY.

```
struct sensor_communication_transmission
{
    robot current
    u16 n_measurements
    position measurements[n_measurements]
}
```

Meddelanden mellan kommunikationsmodulen och styrmodulen kan vara av tre olika typer. Den första används i autonomt läge och innehåller robotens nuvarande tillstånd och robotens önskade tillstånd. Den andra typen är för manuell styrning och innehåller ett kommando. Den sista är för kalibrering av reglerteknik,

```
struct communication_steering_transmission
{
    enum : u8
    {
        AUTONOMOUS
        MANUAL
        CALIBRATION
    } type
    union
    {
        struct
        {
            robot current
            robot wanted
        } instruction
        enum : u8
        {
            FORWARD
            FORWARD_LEFT
            FORWARD_RIGHT
            BACKWARD
            HALT
            TURN_LEFT
            TURN_RIGHT
        } command
        struct
        {
            unsigned propotional_constant
            unsigned derivative_constant
        } calibration
    }
}
```

4 Implementeringsstrategi

Konstruktionen skall ske inifrån och ut. Detta innebär att små delar av varje modul konstrueras och sedan sätts ihop steg för steg. Först när en del är färdig och funktionaliteten är testad integreras den med resten av delarna i modulen. Samma tillvägagångssätt tillämpas när det kommer till att sätta ihop de olika modulerna. Varje enskild modul testas innan de kopplas ihop med övriga moduler.

För att analysera de olika modulerna och systemens funktionalitet kan debuggern i Atmel Studio användas. När kommunikation över Wifi har etablerats kan information skickas och sparas den vägen.

5 Referenser

[1] https://www.isy.liu.se/edu/kurs/TSEA29/projekt/Projektdirektiv_kartrobot_19.pdf