

Bästa Undsättningsrobot

21 maj 2025

# Teknisk dokumentation

G05

21 maj 2025

Version 1.0

Status

Granskad	
Godkänd	2025-xx-xx

### Projektidentitet

Grupp E-post: [tsea56\\_grupp\\_5-users@liuonline.onmicrosoft.com](mailto:tsea56_grupp_5-users@liuonline.onmicrosoft.com)

Hemsida:

Beställare: Kent Palmkvist, Linköpings universitet  
Tfn: 013-28 13 47  
E-post: [kent.palmkvist@liu.se](mailto:kent.palmkvist@liu.se)

Handledare: Petter Källström, Linköpings universitet  
Tfn:  
E-post: [petter.kallstrom@liu.se](mailto:petter.kallstrom@liu.se)

Kursansvarig: Mattias Krysander, Linköpings universitet  
Tfn: 013-28 21 98  
E-post: [mattias.krysander@liu.se](mailto:mattias.krysander@liu.se)

### Projektdeltagare

Namn	Ansvar	E-post
Leo Bärlund	Mjukvaruansvarig	<a href="mailto:leoba684@student.liu.se">leoba684@student.liu.se</a>
Jonatan Bennich	Testansvarig	<a href="mailto:jonbe892@student.liu.se">jonbe892@student.liu.se</a>
Albin Wallerman	GUI-ansvarig	<a href="mailto:albwa916@student.liu.se">albwa916@student.liu.se</a>
Olle Tillberg	Hårdvaruansvarig	<a href="mailto:ollti262@student.liu.se">ollti262@student.liu.se</a>
André Thyberg	Dokumentansvarig	<a href="mailto:andth093@student.liu.se">andth093@student.liu.se</a>
Sebastian André-Jönsson	Projektledare	<a href="mailto:seban171@student.liu.se">seban171@student.liu.se</a>

**DOKUMENTHISTORIK**

Version	Datum	Utförda ändringar	Utförda av	Granskad
1.0	2025-05-21	Slutversion	G05	

## INNEHÅLL

1	Inledning	1
1.1	Syfte och bakgrund . . . . .	1
2	Beskrivning av produkt	1
3	Beskrivning av systemet	2
3.1	Övergripande beskrivning . . . . .	2
3.2	Nivåshiftare . . . . .	2
3.3	Motorer . . . . .	3
3.4	Sensorer . . . . .	3
4	Beskrivning av modulerna	3
4.1	Huvudmodul . . . . .	3
4.2	Styrmodul . . . . .	6
4.3	Sensormodul . . . . .	9
4.4	Användargränssnitt . . . . .	11
5	Slutsatser	12
	Referenser	13
	<b>Appendix</b>	<b>14</b>
A	Sensormodul – kod	14
B	Styrmodul – kod	15
C	Huvudmodul – kod	17
D	Robotens kopplingschema	18

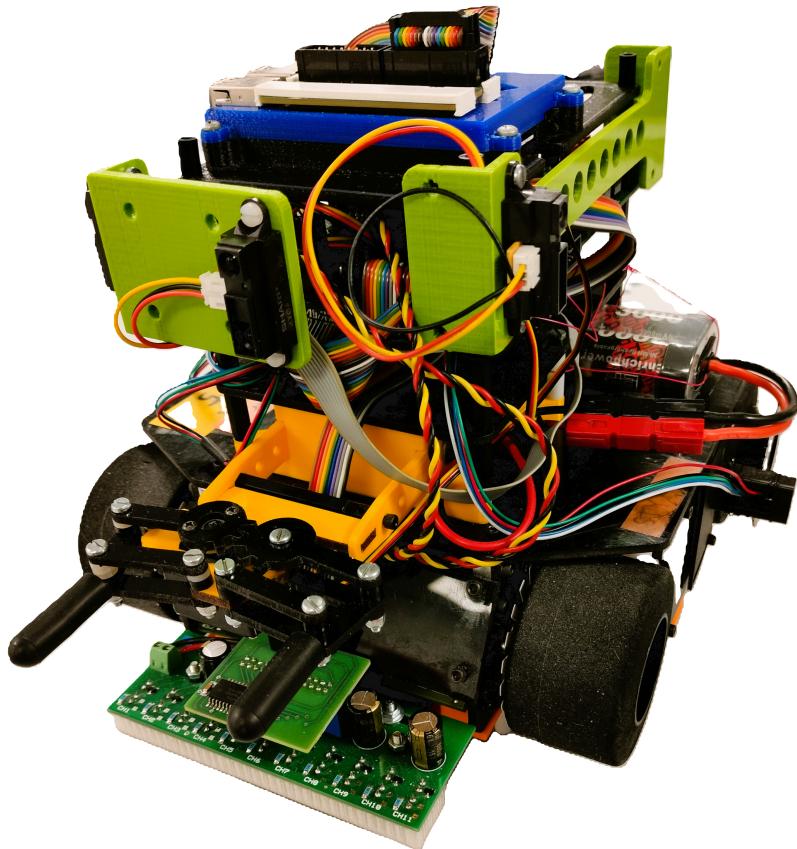
## 1 INLEDNING

### 1.1 Syfte och bakgrund

Syftet med produkten är att i en labyrint lokalisera en nödställd och leverera ett paket. Produkten är en autonom robot som kan kartlägga och navigera en labyrint och lokalisera en nödställd. Sedan återvänder den till startpunkten, hämtar ett paket och levererar det till den nödställda så snabbt som möjligt.

## 2 BESKRIVNING AV PRODUKT

Produkten är en autonom robot som kan kartlägga en labyrint, och identifiera en nödställd i labyrinten. Därefter hämtar den upp proviant vid start, och lämnar detta till den nödställda. Den fungerar också att köra manuellt från en styrdator. [Figur 1](#) visar en bild på robotten.



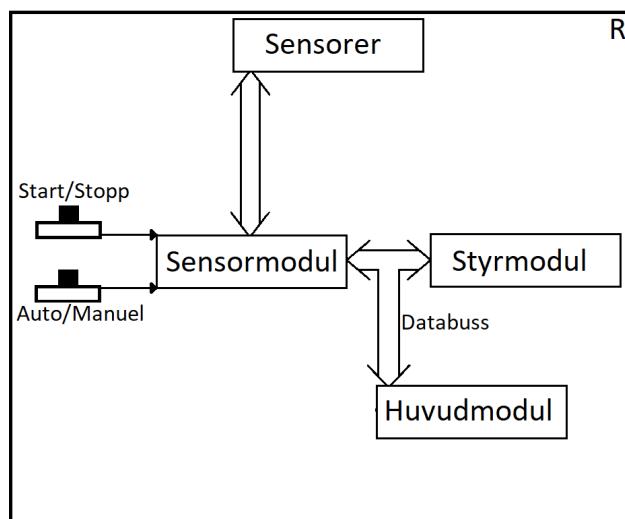
**Figur 1:** Bild på produkten

### 3 BESKRIVNING AV SYSTEMET

Undsättningsroboten bygger på robotplattformen Sumo som är en fyrfjuling med en motor per hjul. Till Sumo monterades ett VIR-kort för koppling av modulerna. Hela systemet drivs med 7.2V NiMH-ackumulator. I Sumo finns en inbyggd spänningsregulator som ger 5V.

### **3.1 Övergripande beskrivning**

Roboten består av tre moduler, som kan kommunicera med varandra. En huvudmodul, en sensormodul och en styrmodul. En bild av hur robotens moduler hänger ihop visas i [Figur 2](#). Huvudmodulen sköter kommunikationen mellan styrdator och andra moduler. T.ex. kan styrmodulen ta emot data från sensorerna via huvudmodulen. Huvudmodulen kommunicerar också med styrdatorn. Exempelvis genom att skicka information från sensorer, karta som uppdateras kontinuerligt och att kunna ta emot kommandon från styrdatorn vid manuell körning. Programmeringsspråket som används är C/C++. Huvudmodulen programmeras i C++ med SFML-biblioteket för användargränssnittet. Styr- och sensormodulen programmeras i C.



**Figur 2:** Beskrivning av moduler

### 3.2 Nivåshiftare

För att möjliggöra kommunikation mellan Raspberry Pi och övriga moduler används en nivåskiftare. Raspberry Pi arbetar med logknivåer på 3,3V, medan AVR använder 5V-logik. Alltså är de inte kompatibla utan nivåshiftaren. I denna länk finns databladet till nivåshiftaren. Där finns pinouten på flatkablarna. [1]

### 3.3 Motorer

Motorerna är färdigmonterade i roboten från början. Varje hjul drivs av en egen motor. Motorerna styrs parvis, vilket möjliggör differentialstyrning, där roboten kan styras genom att hjulen rör sig olika snabbt på de olika sidorna av roboten. Det som används är fyra växlade DC-motorer. De har nominell spänning 7.2 V med varvtal 291 RPM.

### 3.4 Sensorer

Roboten har ett antal sensorer för att få information om dess omgivning. Den har avståndsmätare med kortare räckvidd på höger och vänster sida samt en med längre räckvidd på framsidan. Dessa används för att reglera under körning samt för att rita ut karta. Ett gyro finns monterat för att mäta rotation när roboten svänger. För att mäta robotens fart används läsgafflar som läser av kugghjulsskivor monterade på hjulen. Detta används vid reglering och positionering. Roboten har en reflexsensormodul för att hitta markering av nödställd och positionering vid lastplats. Denna modul finns på robotens framsida och är riktad ner mot golvet.

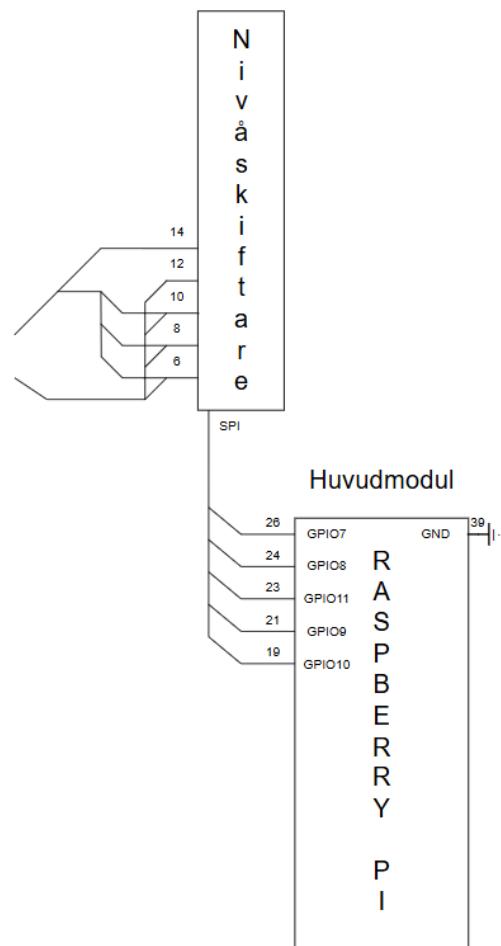
## 4 BESKRIVNING AV MODULERNA

Delsystemen beskrivs för att ge en god inblick i funktion och förhållande till andra delsystem.

### 4.1 Huvudmodul

Huvudmodulen består av en Raspberry Pi. Den sköter kommunikationen mellan modulerna, kommunikationen med styrdator samt större beräkningar. [Figur 3](#) visar huvudmodulens kopplingsschema. Raspberry Pi drivas av 5V, så driftsättning direkt från den inbyggda spänningsregulatorn i Sumon är möjlig.

Kommunikationsmodulen som används är SPI. På grund av detta måste Raspberryn fråga sensormodulen när den vill ha sensordata. Inget avbrott får ske när denna begäran har skickats. Huvudmodulen agerar värd för ett wifi-nätverk som möjliggör fjärrstyrning. VNC används för att spegla det grafiska gränssnittet som körs på huvudmodulen till styrdatorn samt skicka kommandon ifrån styrdatorn till huvudmodulen.

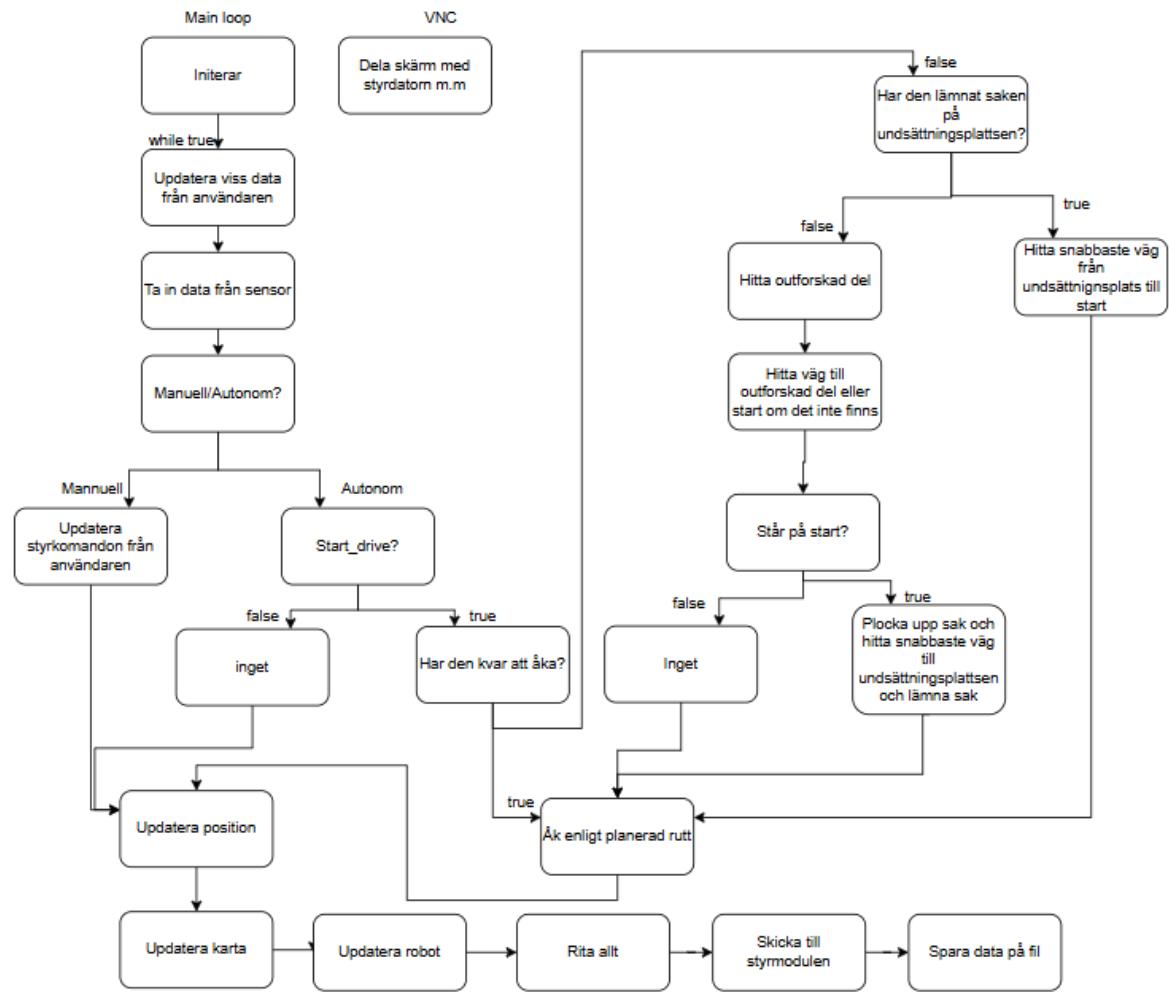


**Figur 3:** Kopplingsschema för huvudmodulen

#### 4.1.1 Mjukvara

Programmet har i uppgift att skicka vidare data från sensormodulen till styrmodulen. Data skickas i intervall om 4 odometersamplingar för att säkerställa stabil reglering. Den har också uppgiften att kartlägga labyrinten och planera en rutt som roboten ska åka, samt kommunicera med en annan dator trådlöst. Huvudmodulen ritar också upp användargränssnittet.

## Huvudmodulen (Språk C++)



**Figur 4:** Flödesschema för huvudmodul

Programmet utgår från att labyrinten är uppdelad i moduler á 40x40 cm. En modul kan antingen vara en väg, vägg eller okänd. Initialt lagras alla moduler i en vektor som innehåller alla möjliga noder. Allteftersom roboten lokaliseras sig i labyrinten tas element ut från den vektorn och läggs till i en ny vektor. Antingen en vektor med väggar eller vektor med vägar. Detta sätt att lagra modulerna gjorde övriga algoritmer lätt att implementera.

### 4.1.2 Algoritmer

Huvudmodulen innehåller de större algoritmerna som roboten använder för att utföra uppgiften. Ett flödesschema över huvudmodulen visas i [Figur 4](#).

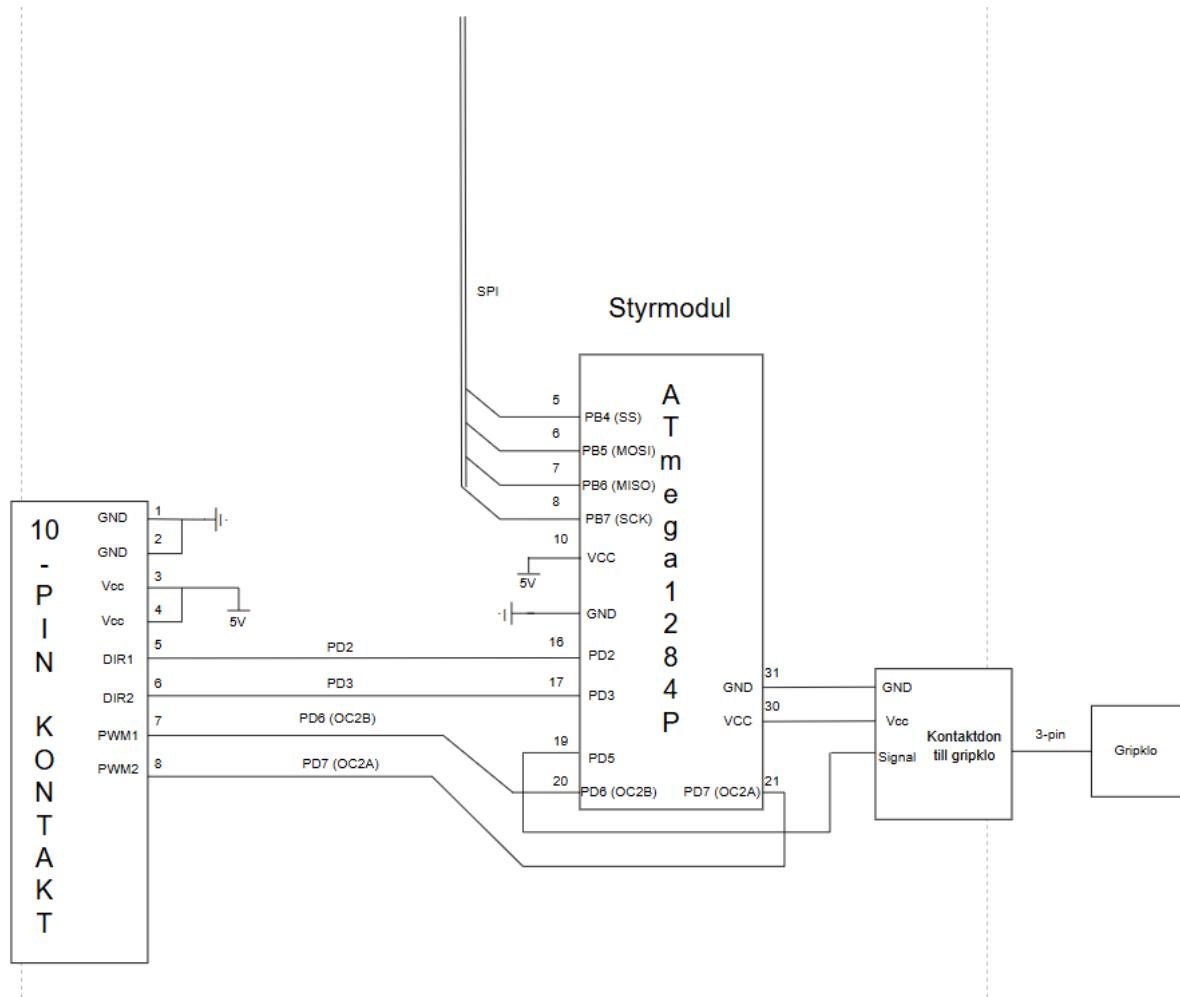
För att optimera rutten mellan startpunkten och den nödställda har Dijkstras algoritm implementerats. Algoritmen tillämpas på en datastruktur där varje identifierad vägposition i labyrинten representeras som en nod i en graf. Dessa noder lagras i en vektor, och en separat funktion används för att identifiera grannoder baserat på deras relativa positioner i rutnätet.

Grafen konstrueras med enhetliga bågkostnader, där varje förflyttning mellan två angränsande noder tilldelas en kostnad av ett. Detta möjliggör en beräkning av den kortaste vägen från startpunkten till målpunkten, baserat på det aktuella kartläggningssläget.

En sökalgoritm implementerades för att möjliggöra systematisk utforskning av labyrинten. Algoritmen bygger på att roboten kontinuerligt uppdaterar en lista över tidigare besökta positioner. Vid varje beslutspunkt utvärderar roboten angränsande rutor för att identifiera möjliga färdvägar, med en prioritet att fortsätta framåt om detta är möjligt. Om ingen ny väg finns tillgänglig från den aktuella positionen, söker roboten igenom listan över besökta platser för att lokalisera närmaste punkt med en obesökt väg. När en sådan position har identifierats, används den optimerade ruttplaneringsalgoritmen beskriven ovan för att minimera avståndet roboten åker till den nya destinationen.

## 4.2 Styrmodul

Styrmodulen tar in information från huvudmodulen om var i labyrинten den ska åka, och ser till att roboten kan åka dit. [Figur 5](#) visar styrmodulens kopplingsschema.

**Figur 5:** Kopplingsschema för styrmodulen

#### 4.2.1 Hårdvara

Styrmodulen består av en AVR-enchipssdator av modell ATmega16 som drivs av 5 V. Den möjliggör drivning direkt från den inbyggda spänningsregulatorn i Sumo. Här listas de komponenter som styrs av styrmodulen.

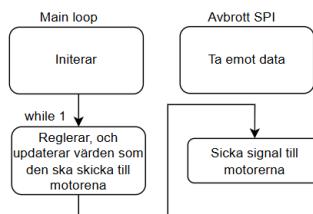
Komponent	Beskrivning	Antal
Drivmotor	För manövrering av robot	4st
Gripklo	Kan plocka upp och släppa föremål	1st

#### 4.2.2 Mjukvara

Programmet får information från huvudmodulen i form av instruktioner vad den ska göra och data från sensormodulen som skickas via huvudmodulen. Instruktionerna används så att den vet var den ska åka och datan från sensormodulen

används för reglering så att den kan lösa instruktionen. Styrmodulen kan ta emot 256 antal instruktioner, i form av heltalet. När den tar emot en instruktion körs koden som står under if-satsen som hör till just den instruktionen. Modulens flödesschema illustreras i [Figur 6](#).

### Styrmodul (Språk C)



**Figur 6:** Flödesschema för styrmodulen

#### 4.2.3 Styrning och reglering

Roboten är differentialstyrd vilket innebär att den styrs genom att hjulen roterar med olika hastighet på de olika sidorna. Varje hjul har en egen motor, och hjul på samma sida av roboten är sammankopplade. Motorerna tar in två signaler. En DIR-signal som bestämmer vilket håll hjulen ska snurra, och en PWM-signal som bestämmer motorernas hastighet.

Huvudmodulen skickar information till styrmodulen om vilken ruta den ska åka till närmast, och vilken väg den ska ta dit. Huvudmodulen förser styrmodulen med sensordata. Styrmodulen omvandlar denna information till DIR- och PWM-signaler som skickas till motorerna.

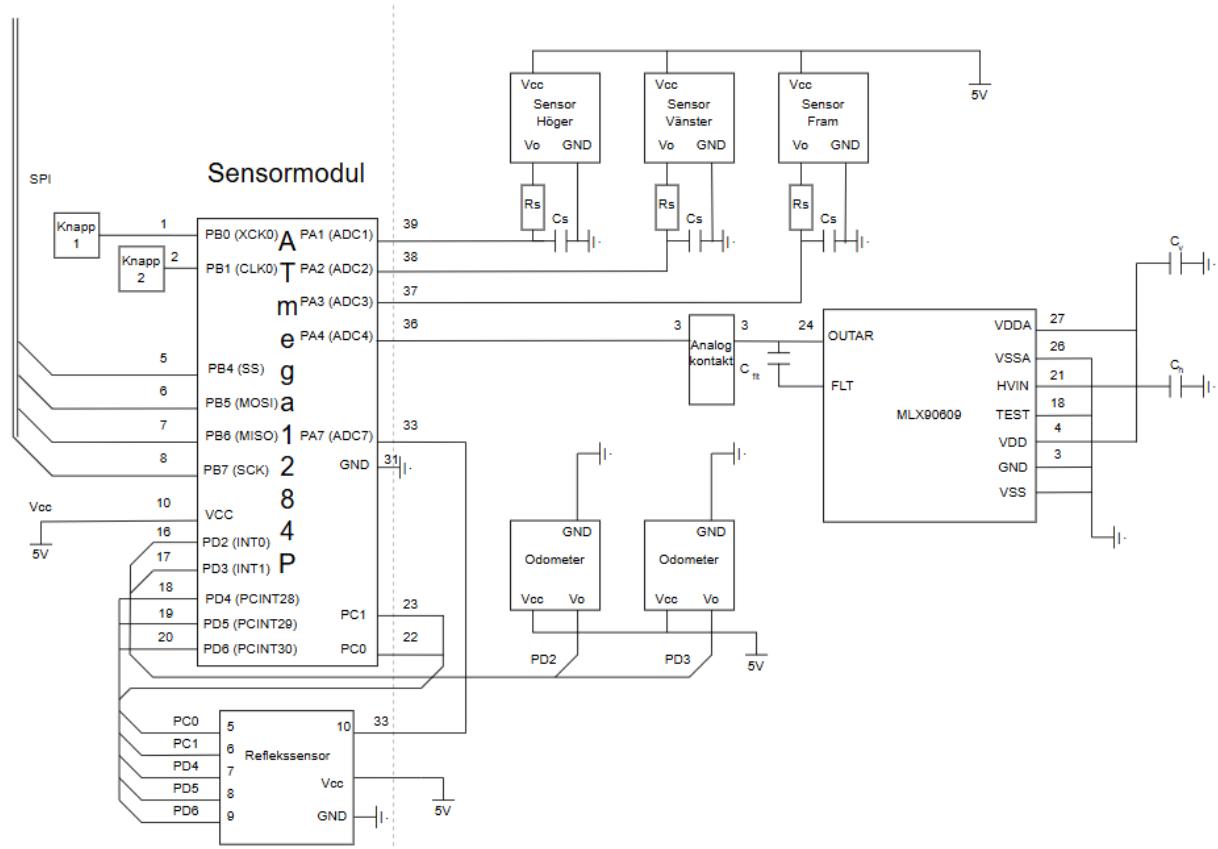
Styrmodulen innehåller en PD-regulator som ser till att roboten kan åka rakt fram på en raksträcka. I regulatornens D-del sker derivering med avseende på sträckan för att undvika fel som uppkommer om man kör olika hastigheter. I regulatorn används även dynamiska parametrar. Värdet på K<sub>P</sub> och K<sub>D</sub> ändras med robotens hastighet. Parametrarna är högre om hastigheten är högre och vice versa. Robotens vinkel på raksträckan bestäms genom att använda sensordata för att se hur avståndet till väggarna förändras på båda sidor. Detta för att vi får räkna med avvikelse på avstånd mellan väggar i labyrinten. I kurvor används data från gyro för att se när roboten vrider sig 90 grader. Gyron används också när roboten ska tvärvända 180 grader på plats.

PD-regulatorn exekveras med mellanrum som beror på indata från odometern. Om odometern hinner skicka en mätning per regleringsalgoritmen så finns det risk för stora fel i regleringen. Därför låter vi PD-algoritmen exekveras en gång per 4 samplingsvärdet från odometern. Detta gör att sträckan hålls mer konstant mellan iterationerna, vilket ger mer pålitlig reglering. Det ligger på huvudmodulen att hålla ordning på när rätt antal samplingar kommer in. Då skickas datan till styrmodulen som använder datan för att reglera.

Styrmodulen är utrustad med en P-regulator som används för att följa en tejp placerad på marken. Denna funktion aktiveras när roboten ska positionera sig inför autonom upplockning. Regleringen baseras på data från reflexsensorer i sensormodulen, vilka detekterar ljusintensitet under roboten och därmed avgör om underlaget är ljus eller mörkt. Syftet med regleringen är att centrera den mörka tejpen under robotens mittpunkt. Detta uppnås genom att justera styrningen proportionellt mot avvikelsen mellan tejpens position och robotens mittlinje.

### 4.3 Sensormodul

På sensormodulen sitter en enchipssdator av modell AVR. [Figur 7](#) visar sensormodulens kopplingsschema. För att bestämma robotens rotation används gyro. För att läsa avstånd till väggarna används IR-sensorer. Lokalisering av de nödställda och lastplats görs med hjälp av en skena med reflexsensorer. För att mäta hjulens rotation används mätgafflar och kugghjulskivor.



**Figur 7:** Kopplingsschema för sensormodulen  
 $C_{flt} = 22nF, C_v = 1uF, C_h = 1uF, C_s = 100nF, R_s = 18k\Omega$

#### 4.3.1 Avståndsmätning

Till sensormodulen finns tre IR-avståndsmätare kopplade. En kommer placeras på robotens högra sida, en på robotens vänstra och en riktad framåt. Sensorerna på höger och vänster sida är av typ GP2Y0A41SK som har en räckvidd på 4 cm till 30 cm. Sensorn rakt fram är av typ GP2Y0A21 och har en räckvidd på 10 cm till 80 cm[1].

#### 4.3.2 Vinkelhastighetsmätning

Ett gyro av typ MLX90609 används för att mäta robotens rotation. För att mäta hjulens rotation används läsgafflar av typen OPB990T51Z som läser av en kugghjulskiva monterad på hjulen.

#### 4.3.3 Lokalisering

Markering av nödställd och lastplats görs med svart eltejp på golvet. Lokalisering av denna tejp och information som behövs för att följa tejpen görs med hjälp av en reflexsensormodul.

#### 4.3.4 Hårdvara

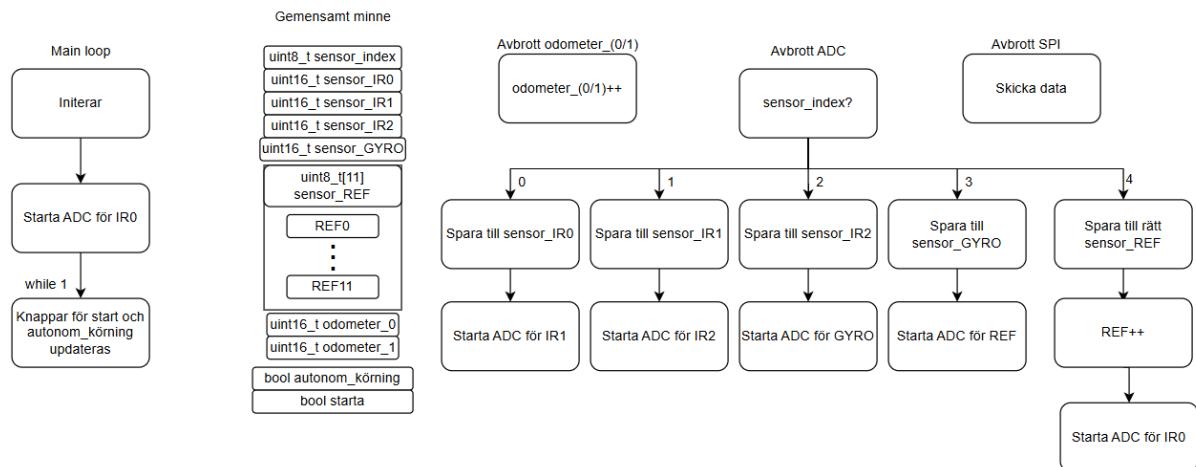
Här listas alla komponenter som utgör sensormodulen.

Komponent	Beskrivning	Antal
Tryckknapp	Knapp för att välja autonom eller manuell körning	1st
Tryckknapp	Knapp för att starta en autonom körning	1st
MLX90609	Gyro för att mäta rotation	1st
GP2Y0A41SK	IR avståndsmätare 4-30 cm	2st
GP2Y0A21	IR avståndsmätare 10-80 cm	1st
Resistans 18kohm	Resistans för konstruktion av lågpassfilter	3st
Kapacitans 100nF	Kapacitans för konstruktion av lågpassfilter	3st
OPB990T51Z	Läsgaffel för mätning av hjulens rotation	2st
Kugghjulsskiva	För mätning av hjulens rotation	2st
Reflexsensormodul	Sensorer för att detektera svart tejp på marken	1st

#### 4.3.5 Mjukvara

Programmet för sensormodulen läser kontinuerligt av data från sensorer och uppdaterar variabler, dess flödesschema visas i [Figur 8](#). Port A hanterar konvertering från analog till digital signal. För att den hela tiden ska jobba så startar en konvertering och sedan när avbrottet från adcn signalerar att den är klar kommer den att starta konvertering för pinnen efter och så vidare. För multiplexern till reflexsensorn så kommer den att läsa en tejpsensor per loop. Det tar alltså 11 loopar att läsa igenom alla tejpsensorvärdet eftersom tejpsensormodulen består av 11 sensorer. I main-loopen skickas data till huvudmodulen när den frågar om det. När huvudmodulen ber om data får inga avbrott ske i sensormodulen.

## Sensormodul (Språk C)



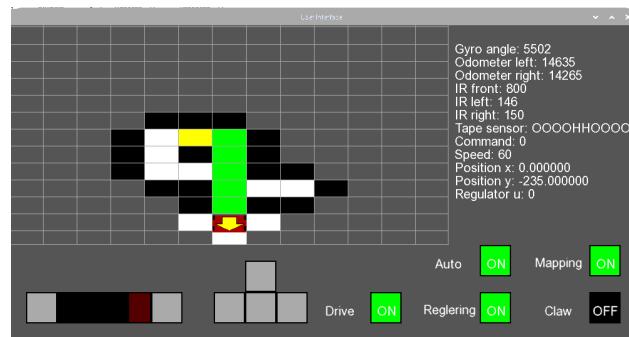
**Figur 8:** Flödesschema för sensormodul

### 4.3.6 Samplingsfrekvens

Samplingsfrekvensen för sensorerna bestäms av hur snabbt alla ADC-avbrott kan göras. Från laborationen i AVR vet vi att en AD-omvandling tar ca 100 mikrosekunder. Utifrån detta uppskattade vi att det tar ca 600 mikrosekunder att AD-omvandla alla mätvärden i loopen. Därför ligger samplingsperioden nästan runt 600 mikrosekunder.

## 4.4 Användargränssnitt

För att kommunicera och styra roboten behöver den kommunicera med styrdatorn. I huvudmodulen skapas ett användargränssnitt där kartan över labyrinten ritas ut under körning. Knappar för att styra roboten under manuell körning finns i gränssnittet. Maskinstatus visas också i gränssnittet, till exempel dess fart. VNC används för att spegla användargränssnittet till styrdator samt skicka kommandon från styrdator till huvudmodul. [Figur 9](#) visar användargränssnittets utseende.



**Figur 9:** Bild av användargränssnittet

## 5 SLUTSATSER

En nuvarande begränsning hos produkten är att den endast fungerar i labyrinter där väggarna har en fast storlek på 40 cm, där det inte förekommer några öppna ytor, och där alla svängar är 90 grader. För att vidareutveckla produkten vore ett naturligt nästa steg att öka robotens flexibilitet så att den kan navigera i mer varierade labyrinter. Detta skulle dock kräva omfattande förändringar, eftersom mycket av programmets nuvarande struktur bygger på antagandet att labyrinten är uppbyggd av moduler i storleken 40x40 cm.

Det finns även förbättringspotential inom regleringen. Genom att justera parametrarna kan man förbättra styrningen, men det kan också vara värt att undersöka alternativa regleringsstrategier för att uppnå bättre prestanda.

## REFERENSER

- [1] L. universitet. "Vanheden." Åtkomstdatum 2025-05-19. (2025), URL: <https://da-proj.gitlab-pages.liu.se/vanheden/>.

# Appendix

## A SENSORMODUL – KOD

```

1 // Programmerare: S.André-Jönsson, J.Bennich, L.Bärlund, A.Thyberg, O.Tillberg, A.Wallerman
2 //Datum: 2025-05-21
3 //Version: 1.0
4 #include <avr/io.h>
5 #include <avr/interrupt.h>
6 #include <math.h>
7
8 volatile uint16_t sensor_data[6]; //{ir_sensor_right, ir_sensor_left, ir_sensor_front, gyro_angle, odometer_left, odometer_right}
9 volatile uint16_t temp_sensor_data[6]; //{ir_sensor_right, ir_sensor_left, ir_sensor_front, gyro_angle, odometer_left, odometer_right}
10 volatile uint16_t ir_temp[3];
11 volatile uint8_t auto_drive;
12 volatile uint8_t start_drive;
13 volatile uint8_t tape_sensor_module[11];
14
15 volatile uint8_t ir_count = 0;
16
17 volatile int8_t byte_count;
18 volatile int8_t sensor_index = 0;
19 volatile int8_t tape_sensor_index = 1;
20
21 volatile int16_t gyro_data;
22 volatile int16_t gyro_calibration_index = 0xff; // ändrat från uint16_t
23 volatile uint32_t gyro_bias;
24
25
26 void registerInit()
27 {
28     for(int i = 0; i<6; i++)
29     {
30         sensor_data[i] = 0; //{ir_sensor_right, ir_sensor_left, ir_sensor_front, gyro_angle, odometer_left, odometer_right}
31         temp_sensor_data[i] = 0; //{ir_sensor_right, ir_sensor_left, ir_sensor_front, gyro_angle, odometer_left, odometer_right}
32         if(i<3)
33         {
34             ir_temp[i] = 0;
35         }
36     }
37     auto_drive = 0;
38     start_drive = 0;
39     for(int i = 0; i<11; i++)
40     {
41         tape_sensor_module[i] = 0;
42     }
43     byte_count = 0;
44     gyro_data = 0;
45     gyro_calibration_index = 0xff;
46     gyro_bias = 0;
47 }

```

**Figur 10:** En del av koden för sensormodulen

## B STYRMODUL – KOD

```

1 #include <avr/io.h>
2 #include <avr/io.h>
3 #include <avr/interrupt.h>
4 #include <stdlib.h>
5
6 // -----
7 // Filnamn: Steermodule.c
8 // Här finns koden för AVR:en som används som styrmodul.
9 // Programmerare: Albin Wallerman, Leo Bärklund, André Thyberg, Olle Tillberg, Sebastian André-Jönsson, Jonathan Bennich
10 // -----
11
12 //Här initierar vi variabler som antingen skickas med SPI från Raspberryn eller som används lokalt i styrmodulen.
13 volatile uint16_t sensor_data[6] = {}; //{{ir_sensor_front, ir_sensor_right, ir_sensor_left, gyro_angle, odometer_left, odometer_right}
14 volatile uint8_t command = 0;
15 volatile uint8_t temp_command = 0;
16 volatile uint8_t tape_sensor_module[11] = {};
17 volatile uint32_t millis_counter = 0;
18 volatile int8_t byte_count = 0;
19 volatile int32_t temp_kp = 0;
20 volatile int32_t temp_kd = 0;
21 volatile int32_t tape_mean = 0;
22 volatile uint8_t autonom = 0;
23 volatile uint16_t regulator_max_distance = 180;
24 uint8_t speed = 127;
25 int32_t u = 0;
26
27 void registerInit(void) // initiera portar mm.
28 {
29
30     DDRD |= (1<<DDD2|1<<DDD3|1<<DDD6|1<<DDD5 | 1<<DDD7); // DIR0, DIR1, PWM1 och PWM2 och P05/gripklo satta som utgångar
31     OCR2B = 0; // initial hastighet 0
32     OCR2A = 0; //
33     OCR1A = 0; //Gripklo start läge
34     PORTD &= ~(1<<PD2); // initial riktning framåt på vänster hjul
35     PORTD &= ~(1<<PD3); // initial riktning framåt på höger hjul
36
37     // Motor set-up
38     TCCR2A |= (1 << COM2A1) | (0 << COM2A0) | (1 << COM2B1) | (0 << COM2B0) | (1 << WGM20) | (1 << WGM21); // Sätter Timer2 till Fast PWM-läge sida
39     //      WGM sätter hur långt klockan/counter räknar, COM2x1:0 Clear OC2x on Compare Match, set OC2x at BOTTOM,
40     // (non-inverting mode). Se datablad
41
42     TCCR2B |= (0 << WGM22) | (1 << CS21); //CS21 sätter Prescaler 8
43
44
45 }
46
47 void SPI_SlaveInit(void) //initiera slave-portar
48 {
49     DDRB |= (1<<DDB6);
50     SPCR = (1<<SPIE) | (1<<SPE);
51 }
```

**Figur 11:** En del av koden för styrmodulen



## C HUVUDMODUL – KOD

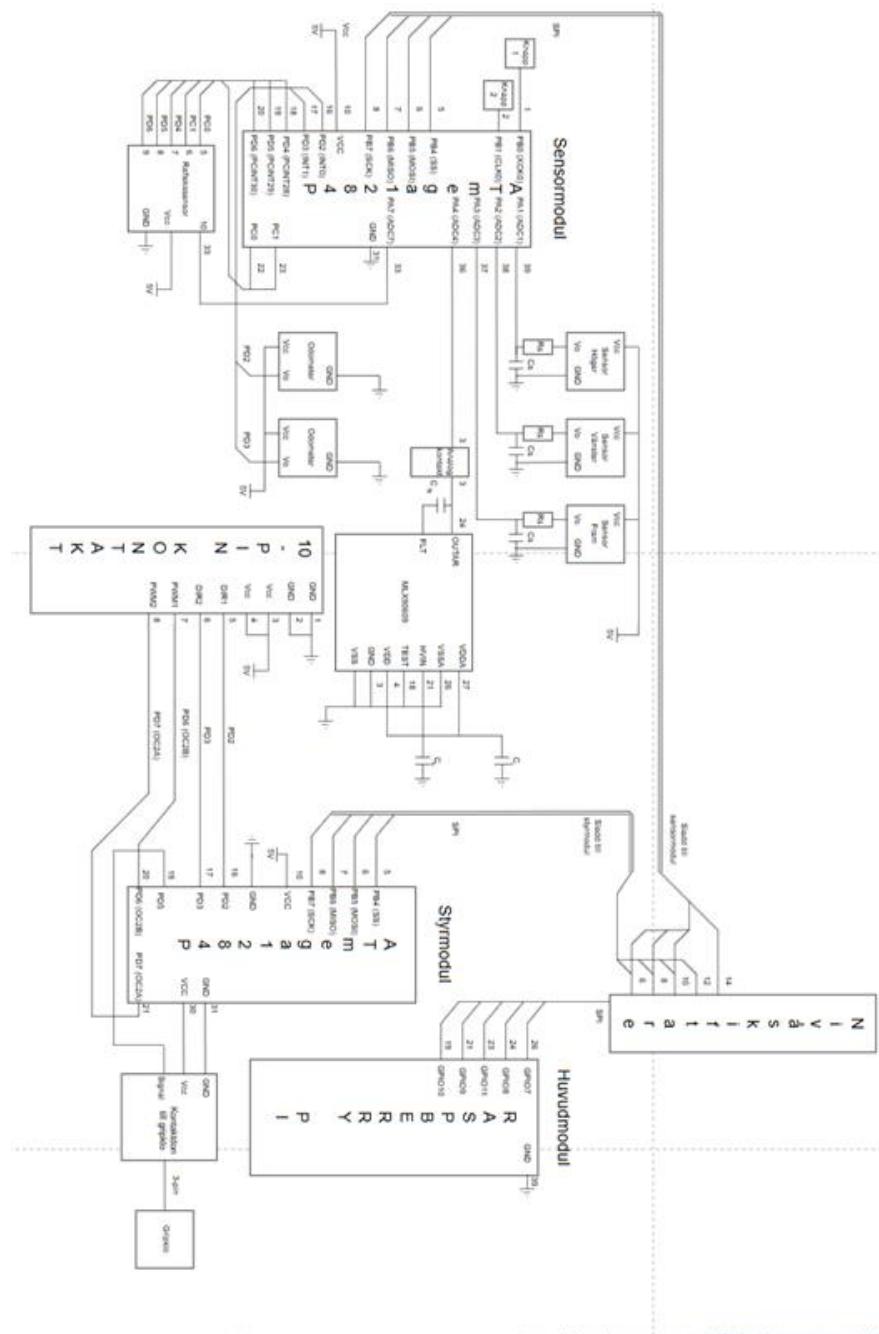
```

1 // Programmerare: S.André_Jönsson, J.Bennich, L.Bärlund, A.Thyberg, O.Tillberg, A.Wallerman
2 //Datum: 2025-05-21
3 //Version: 1.0
4
5 #include <SFML/Graphics.hpp>
6 #include <iostream>
7 #include <string>
8 #include <cmath>
9 #include "Button.hpp"
10 #include "GlobalVariables.hpp"
11 #include "ArrowKeys.hpp"
12 #include "Bar.hpp"
13 #include "InfoBar.hpp"
14 #include "InfoList.hpp"
15 #include "Toggle.hpp"
16 #include "Map.hpp"
17 #include "Grid.hpp"
18 #include "Robot.hpp"
19 #include "PositionCalc.hpp"
20 #include "SearchAlgo.hpp"
21 #include "SPI.hpp"
22 #include "EventHandler.hpp"
23 #include "Shortest_Path.hpp"
24 #include "to_txt.hpp"
25 #include "Line.hpp"
26
27 int tid = 0; // används för att införa en födröjning i PositionCalc
28 int main()
29 {
30     int Width = 1000; // bredd på fönstret
31     int Height = 500; // höjd på fönstret
32     int map_width = 700; // bredd på kartan
33     int map_height = 700; // höjd på kartan
34     int font_size {20}; // storlek på texten i GUI
35     float button_size {50}; // storlek på knapparna
36     float bar_size {150}; // storlek på hastighetsreglaget
37
38     sf::Vector2f position_keys {325, 375};
39     sf::Vector2f position_bar {25, 425};
40     sf::Vector2f position_info_list {710, 25};
41     sf::Vector2f position_map_toggle {825, 350};
42     sf::Vector2f position_auto_toggle {650, 350};
43     sf::Vector2f position_claw_toggle {825, 425};
44     sf::Vector2f position_reg_toggle {650, 425};
45     sf::Vector2f position_start_toggle {475, 425};
46     sf::Vector2f line1_start {0, 350}; // startposition för linje runt kartan
47     sf::Vector2f line1_end {700, 350}; // slutposition för linje runt kartan
48     sf::Vector2f line2_start {700, 0}; // startposition för linje runt kartan
49     sf::Vector2f line2_end = line1_end; // slutposition för linje runt kartan
50
51     GlobalVariables global_variables{}; //skapa en instans av GlobalVariables där globala variabler lagras

```

**Figur 12:** En del av koden för huvudmodulen

## D ROBOTENS KOPPLINGSSCHEMA

**Figur 13:** Hela robotens kopplingsschema