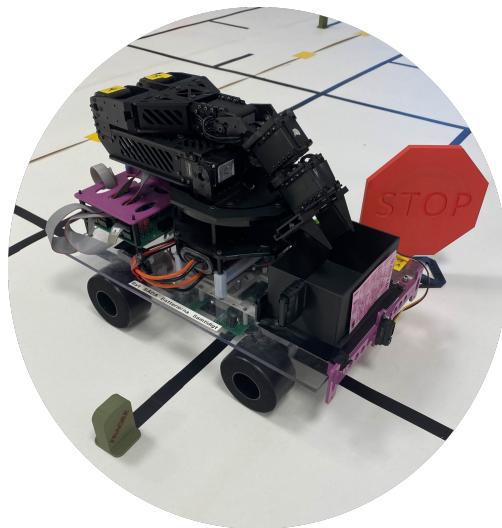


Konstruktion av en autonom lagerrobot Design of an Autonomous Warehouse Robot

Grupp 6

9 juni 2025

Version 1.1



Status

Granskad	Ebba Lundberg	2025-06-09
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno773@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1 Inledning	1
2 Problemformulering	3
3 Kunskapsbas	4
4 Genomförande	4
5 Teknisk beskrivning	5
5.1 Kommunikationenhet	5
5.2 Sensorenhet	6
5.3 Styrenhet	7
5.4 PC	7
6 Resultat	8
7 Slutsats	8

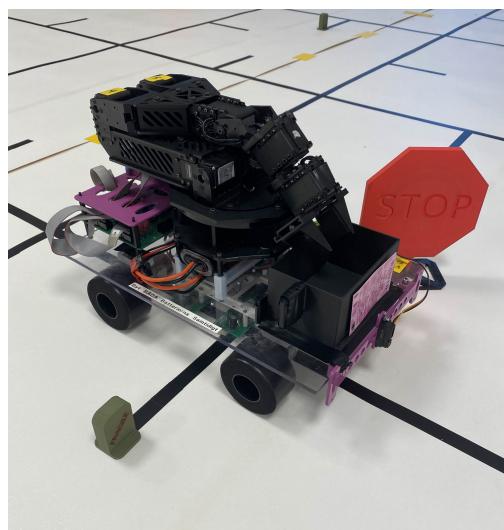
Appendix	11
8 Projektdirektiv	11
9 Kravspecifikation	13
10 Systemskiss	31
11 Projektplan	43
12 Designspecifikation	57
13 Förstudier	74
13.1 Kommunikationenhetens förstudie	74
13.2 Sensorenhetens förstudie	91
13.3 Styrenhetens förstudie	107
14 Användarmanual	125
15 Teknisk dokumentation	138
16 Efterstudie	178

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
1.0	2025-05-21	Första version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS
1.1	2025-06-09	Andra version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS

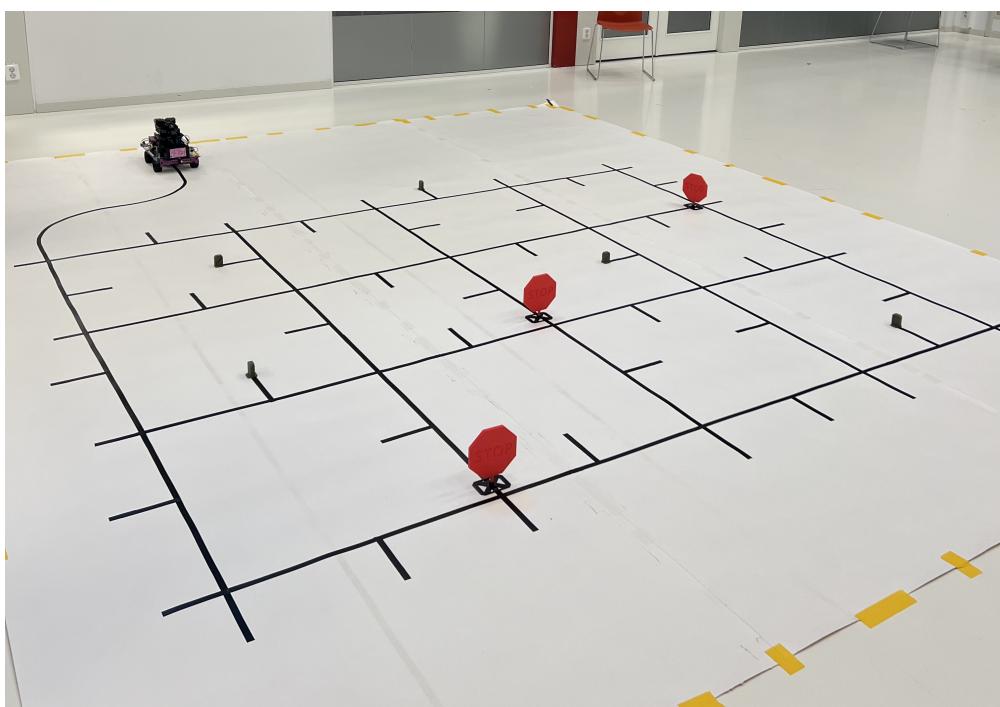
1 INLEDNING

Detta dokument beskriver utvecklingsprocessen och resultatet för konstruktionen av en autonom lagerrobot i kursen TSEA56, elektronik kandidatprojekt. Produkten som tas fram är en autonom lagerrobot bestående av en robotplattform och en robotarm med en gripklo, se Figur 1 för den färdiga konstruktionen. På roboten finns dessutom tre enheter installerade: en sensor-, styr- och kommunikationsenhet. Utöver detta finns en separat PC-enhet där användaren kan kontrollera roboten genom ett gränssnitt.



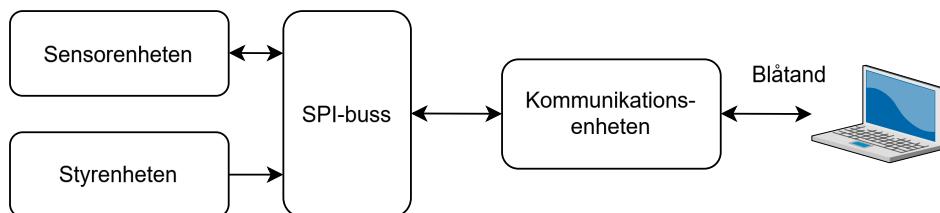
Figur 1: Lagerrobot med robotarm och robotplattform.

Uppdraget som roboten utför består av två huvudsakliga delar. Först följer roboten en krokig linje från återhämtningsstationen till själva lagret. Väl framme navigerar den i en miljö med korsningar, hinder och plockstationer, se Figur 2 för ett exempel på hur denna kan se ut. Lagerroboten vet i förväg hur den ska svänga i korsningarna och var plockstationerna befinner sig, men den identifierar potentiella hinder under körning och anpassar sin väg efter dessa. Vid plockstationerna plockar robotarmen autonomt upp varan och placerar den i en korg på robotplattformen. När alla varorna är hämtade navigerar roboten tillbaka till återhämtningsstationen och lämnar av varorna.



Figur 2: Lagermiljö med varor och hinder.

Översiktligt så fungerar roboten genom ett samspel mellan de fyra modulerna, se Figur 3. Kommunikationsenheten tar emot och skickar kommandon samtidigt som den samlar in och distribuerar data till övriga enheter. Information om robotens omgivning, såsom placeringen av hinder och linjens position, samlas in av sensorenheten. Denna data skickas därefter till kommunikationsenheten där styrbeslut beräknas och kommandon utfärdas till styrenheten. Dessutom skickas all data till PC:n för att visas för användaren. Kommandon till styrenheten kan även skickas direkt från PC:n via kommunikationsenheten om användaren vill styra lagerroboten manuellt. Detta initieras via en knapp i användargränssnittet.



Figur 3: Översikt av hela systemet.

Under projektets gång uppstod flera tekniska utmaningar som behövde lösas. Motorerna i robotarmen visade sig vara svåra att styra på ett tillförlitligt sätt, vilket krävde justeringar i både mjukvara och styrlogik för att få dem att fungera stabilt. Även linjeföljningen behövde förändras från den ursprungliga designen. Det visade sig att mer precision

krävdes för att roboten skulle kunna navigera tillräckligt noggrant i förhållande till uppdragets krav. Kommunikationslösningen stötte också på problem, särskilt när vissa delar behövde köras parallellt. SPI-bussen fungerade inte som förväntat vid multitrådning, vilket ledde till konflikter och svårtolkade fel. Ett återkommande problem var att vissa fel i systemet var svåra att identifiera, eftersom de inte alltid berodde på tydliga buggar i koden. Ofta handlade det om att koden kördes för snabbt i relation till hur hårdvaran svarade, vilket kunde ge upphov till odefinierat beteende. Det blev tydligt att många problem inte enbart hade sin grund i mjukvaran, utan också påverkades av fysiska faktorer, något som skiljer sig från hur man ofta arbetar i mer renodlade programmeringsprojekt.

Kappan är den inledande och sammanhållande delen av kandidatarbetet. Syftet är att ge läsaren en tydlig överblick över projektets olika delar, från den teoretiska bakgrunden till hur arbetet har genomförts och vilka resultat som uppnåddes. Inledningsvis presenteras relevant teori, baserat på de förstudier som genomfördes av gruppmedlemmarna. Dessa förstudier låg till grund för projektets konstruktionsfas och bidrog till att forma arbetets praktiska upplägg. Därefter beskrivs projektets planering, genomförande och resultat. Avslutningsvis följer en reflekterande del där gruppmedlemmarna diskuterar sina erfarenheter, de utmaningar som uppstod samt hur samarbetet fungerade under arbetets gång.

2 PROBLEMFORMULERING

Projektets syfte är att konstruera ett system som ett konstruktionsalternativ till en lagerrobot, som med snabbhet och repeterbarhet utför sitt uppdrag. Lagerroboten är en prototyp till en robot som ska tillverkas. Från en återhämtningsstation kör den genom linjeföljning till en lagermiljö. Där navigerar den kring hinder för att hämta och köra tillbaka alla varor som är placerade i lagret.

Den sluttagna uppdragsbeskrivningen och systemdesignen formuleras dels genom den ursprungliga projektbeskrivningen och dels genom förhandlingar med beställaren. Från det ursprungliga projektdirektivet (se Appendix 8) gäller fortfarande det mesta. Vissa detaljer förtydligas och mindre ändringar beskrivs i kravspecifikationen (se Appendix 9). I kravspecen finns tre olika prioriteter kring vilka krav som ska vara uppfyllda och när de ska vara färdiga. Prioritet 1 ska vara klar till den första leveransen (BP5A), prioritet 2 till slutleveransen (BP5B), och prioritet 3 är tänkt att implementeras i mån av tid. Bland det som detaljeras i kravspecifikationen är att Bluetooth ska användas i kommunikation med PC-enheten och preciseringar på hur lagermiljön ska se ut. Annat som förhandlas bort i kravspecifikationen från projektbeskrivningen är bland annat att gripklons skattade position ska visas i användargränssnittet.

Kraven på roboten presenterar ett antal unika utmaningar för design och konstruktion. I början finns flera konstruktionsalternativ för linjeföljningen och efter en lång process av att testa dessa väljer man till slut att installera två reflexsensorer för att kunna uppskatta robotens färdriktnings. Robotarmens rörelser är också ett stort problem. Till slut bestäms det att den alltid gör samma förutbestämda rörelse vid autonom upplockning.

3 KUNSKAPSBAS

Innan konstruktionsfasen påbörjas genomför gruppen förstudier inom tre centrala områden: kommunikation, styrning och sensorer (se Appendix 13). Syftet är att skapa en teoretisk grund för att kunna fatta beslut kring val av metoder, hårdvara och mjukvara, samt hur dessa delar kan integreras och tillämpas i projektet på ett effektivt sätt.

Utöver förstudierna bidrar även laborationer till att förbereda gruppen inför konstruktionsfasen genom en introduktion i hur komponenter kopplas till mikrodatorer. Under denna period introduceras även mikrodatorns datablad, vilket ger viktig information om hur mikrokontrollerna fungerar.

Förstudierna ger en övergripande förståelse för hur olika kommunikationssätt, servon och sensorer fungerar i teorin. Mer specifik information om den faktiska hårdvaran som finns tillgänglig hittas i datablad som tillhandahålls via ISY:s plattform Vanheden [1].

I Appendix 8 står det tydligt vad roboten förväntas klara av i slutet av projektet. Projektdirektiven och kravspecifikationen (Appendix 9) är viktiga för att definiera projektets omfattning och avgränsningar, vilket gör det möjligt att genomföra projektet.

4 GENOMFÖRANDE

Projektet har i sin helhet utgått från LIPS-modellen. Det är en modell som främst är framtagen för projektarbeten i undervisningssituationer. Modellen har tre faser: före-, under- och efterfasen. Där emellan finns flertalet milstolpar och beslutspunkter som uppfylls i samråd med handledaren eller beställaren. LIPS-modellen tillhandahåller även olika dokumentmallar som till exempel kravspecifikation.

Förefasen började med flertalet föreläsningar där projektuppdraget introducerades samt en förklaring av de olika dokumenten. Laborationer genomfördes för att få grundläggande förståelse för mikrodatorerna och utvecklingsmiljön. En kravspecifikation upprättas där kraven som ska uppfyllas vid olika beslutspunkter specificeras, se Appendix 9. För att planera hur dessa krav ska uppfyllas tas en systemskiss fram (Appendix 10) samt en projektplan (Appendix 11). Dessa tillsammans med tidsplanen utgjorde grunden för planeringen av projektet och tidsplanen var användbar för att få en överblick över vad som behövde göras varje vecka. Slutligen så gjordes en designspecifikation (Appendix 12) där beslut togs om främst hårdvaran som skulle användas i nästa fas. Komponentlistor, blockscheman, kopplingsscheman och pseudokod gjordes här.

Underfasen bestod främst av programmering och testning av kod. Aktiviteterna i tidsplanen följdes så gott det gick, även om många aktiviteter tog kortare eller längre tid än vad som planerats. Till en början delades gruppen in i tre mindre grupper om två som satt med respektive modul. Detta gjordes för att underlätta programmeringen och testningen av respektive delsystem. Designspezifikationen gav en bra grund i början, även om vissa avsteg gjordes vad gäller hård- och mjukvaran under konstruktionen. I början av varje vecka hölls ett veckomöte där gruppen gick igenom vad som hade gjorts föregående vecka, vad som ska göras under veckan och hur tidsplanen följdes. Detta för att ge gruppen och beställaren en löpande översikt om projektets utveckling. Närmare leveranserna så integrerades de olika delsystemen för att skapa en komplett robot, vilket gjorde att gruppen arbetade mer tillsammans än tidigare. Flera

dokument skrevs också inför slutleveransen. Underfasen avslutades med en redovisning för beställaren där gruppen visade att baskraven för projektet uppfylldes.

Efter projektets huvudfas så demonstrerades de slutgiltiga kraven på roboten samt teknisk dokumentation, användarma-nual och kandidatrapport lämnades in. Det följdes av framläggning, opponering och robottävling. Projektet avslutades med en efterstudie.

5 TEKNISK BESKRIVNING

Detta kapitel innehåller en övergripande teknisk beskrivning över de olika delsystemen. Innehållet baseras på förstudien (se Appendix 13) och den tekniska dokumentationen (se Appendix 13). De fullständiga beskrivningarna återfinns i dessa bilagor, medan detta kapitel fokuserar på gruppens egna kreativa lösningar.

5.1 Kommunikationsenhet

Kommunikationsenheten består av en Raspberry Pi och en nivåskiftningsrelä. Den seriella kommunikation som används är SPI-kommunikation, och vid implementationen av denna är informationen i förstudien till stor hjälp. Via SPI-bussen skickas data i byteformat mellan kommunikationsenheten och sensorenheten samt mellan kommunikationsenheten och styrenheten. Mellan PC och kommunikationsenheten skickas data, även här i byteformat, via Bluetooth. All data som skickas samt vad respektive data betyder finns beskrivet i den tekniska dokumentationen 15.

Mycket tid läggs på dataöverföringen mellan de olika delsystemen och i vilket format data skickas. För att data ska kunna skickas i byteformat och även kunna användas i alla delsystem omvandlas formatet mellan bytes, strängar, hexadecimalt och så vidare. Här testas mycket fram för att kunna skicka och representera data på ett läsbart sätt.

Det märks så småningom att flera olika processer i kommunikationsenheten behöver vara igång samtidigt för att roboten ska kunna fungera. För att detta ska lyckas behövs threading, något som inte studeras under förstudien. Med detta sagt läggs mycket tid även på detta.

Mycket tid läggs även på snabbaste-vägen-algoritmen. I början är det svårt att veta hur noder och framför allt plockstationer ska definieras, vilket löses genom att döpa noderna till siffror som visas i användargränssnittet och plockstationer som kombinationer av de noder varorna ligger emellan. Först utvecklas en kod som endast beräknar snabbaste vägen från start till en vara och sedan tillbaka, alltså för att hämta alla varor en och en. När beslutet tas att alla varor ska hämtas i en färd behöver algoritmen bli desto mer komplex. Detta löses med hjälp av en så kallad Held-Karp-algoritm. Ännu svårare blir det när vägen ska kunna beräknas om efter utplacerade hinder, då koden både behöver radera vissa vägbitar samt fungera då start- och slutnod inte är samma. Detta löses genom att radera grannar ur grafen samt modifiering av Held-Karp-algoritmen.

När en lista med ordningen i vilken noderna ska besökas är klar översätts den till styrbeslut som roboten kan tolka. Detta innebär att noderna i lagret översätts till koordinater och sedan beräknas vilken riktning roboten kommer ifrån och vilken den ska åka mot.

Det är en utmaning att implementera snabbaste-vägen-koden tillsammans med övrig kod på Raspberry Pi och att skicka de listor och värden som behövs för att kunna beräkna vägen. Att ta reda på var roboten befinner sig i lagret är viktigt både för att kunna placera ut hinder och för att veta vilka varor som fortfarande behöver hämtas efter att hinder uppstått. Detta lösas genom att efter varje utförd instruktion raderas den från listan med styrbeslut och sedan jämförs längden på listan med listan för nodordningen. På detta sätt kan robotens position och nästa planerade nod bestämmas, vilket är det som behövs för att beräkna ny väg. Varje gång roboten får kommandot 'plocka' tas varan bort vars noder matchar robotens position, och på så sätt finns endast de varor som ännu inte plockats upp kvar i listan med mål.

Det är mycket svårt att testa snabbaste-vägen-koden eftersom det finns nästan oändligt många olika situationer som kan uppstå. Några fel som uppstår är till exempel när det finns två vägar roboten kan välja mellan och båda kräver vändning. Eftersom koden säger att om det finns fler än en väg som är kortast ska den som inte kräver vändning väljas. Ett annat problem som uppstår är att koden inte kan hantera när sista varan ligger precis innan slutnoden, vilket lösas genom att hantera dessa fall specifikt.

5.2 Sensorenhet

De olika sensorerna som används i projektet skiljer sig en del i svårighetsgrad. Det beror dels på hur utförliga respektive datablad är. En annan aspekt är att ingen av sensorerna är särskilt lika i implementeringen, vilket gör att erfarenheterna från en sensor inte nödvändigtvis underlättar vid implementeringen av en annan sensor.

Den svåraste sensorn att implementera är gyroskopet. Den skiljer sig tydligt från de andra sensorerna både i användningsområde och implementation. En stor skillnad är att gyroskopet inte skickar data kontinuerligt, utan endast i vissa situationer. Detta kräver mer funktionalitet i kommunikationsheten jämfört med de andra sensorerna. Att starta och stänga av sensorn under körning är unikt för gyroskopet. Det är även svårt att behandla datan från gyroskopet. Mer detaljerad information om sensorns data finns i Appendix 13.2. Trots detta är den slutgiltiga lösningen för gyroskopet den enklaste av sensorerna. Detta beror framför allt på att situationerna där gyrot används är mycket specifika, till exempel vid rotationer på 90 eller 180 grader. Genom att känna till dessa specifika vinklar går det att mäta upp motsvarande värden i sensorns output genom att rotera roboten 90 respektive 180 grader. Detta möjliggör även att undvika ineffektiva flyttalsberäkningar på mikrodatorn.

Avståndssensorn är den första sensorn som implementeras och den sensor som är enklast att förstå rent teoretiskt. Databladet är tydligt och innehåller en graf som är mycket användbar. Inledningsvis utvecklas en generell funktion som beräknar godtyckliga avstånd inom sensorns intervall. Detta visar sig dock vara ineffektivt ur ett beräkningsperspektiv för mikrodatorn. Den slutgiltiga lösningen utnyttjar istället grafen i databladet där diskreta spänningsvärden motsvarar diskreta avstånd, se Appendix 13.2. Detta möjliggör användning av linjär interpolering, vilket gör implementeringen mycket enkel i slutändan.

Linjesensorn är den mest tidskrävande sensorn att implementera och kräver också mest kod. Databladet för sensorn är inte särskilt utförligt, men det påverkar inte arbetet nämnvärt eftersom teorin bakom linjesensorn är lätt att förstå, se Appendix 13.2. I kombination med en föreläsning i kursen som förklarar hur en linjesensor kan användas för linje-följning blir dess syfte tydligt. Kodmängden för linjesensorn är betydligt större än för de andra sensorerna. Kodbasen växer dessutom ytterligare under projektets gång när beslut tas att använda två linjesensorer för att förenkla regleringen, se Appendix 13. Regleringen, som beskrivs i den tekniska dokumentationen, är något grupperna är särskilt nöjd med eftersom den slutgiltiga lösningen visar sig vara betydligt enklare än tidigare testade metoder.

5.3 Styrenhet

Implementeringen av styrenheten har en varierande svårighetsgrad, men är överlag mindre komplicerad än vad som först förväntades.

Det första som implementeras är körningen av robotplattformen. Den största utmaningen är att generera korrekta PWM-signaler för att styra hjulen i rätt hastighet, men eftersom databladet är tydligt och bra uppstår inga större problem. När hjulen kan rulla visar sig alla nödvändiga funktioner vara enkla att implementera. Det handlar mest om att konfigurera DIR-signalerna, som styr hjulens rotationsriktning, för att skapa funktioner för att köra framåt, rotera och backa.

Det blir något mer komplicerat när en konfigurerbar svängning för linjeföljningen ska implementeras. Som argument till funktionen används resultatet från regleralgoritmen. Efter flera varianter väljs följande metod: först sätts höger och vänster hjulpar till en basfart. Sedan, beroende på åt vilket håll robotplattformen ska svänga, adderas reglervärde till hastigheten på det hjulpar som är på den yttre sidan av svängen, samtidigt som samma värde multiplicerat med 1,25 subtraheras från hjulparet på insidan av svängen. Denna metod ger en svängrörelse som fungerar bra både på de raka linjerna i lagret och i svängarna.

När styrningen av robotplattformens hjul börjar fungera bra läggs mer fokus på att komma igång med robotarmen. Till en början används två testservon. Eftersom vissa leder i robotarmen har två servon behöver grundläggande kontroll över servona uppnås innan armen kan användas. Styrningen av servona sker via UART, men eftersom den endast kan hantera halvduplex krävs en transistorkrets med en styrsignal som växlar riktning.

Att komma igång med UART-kommunikationen går ändå smidigt, då det finns tydlig och bra information om hur datan som skickas till servona ska vara formaterad på tillverkarens hemsida, samt bra instruktioner om UART i mikrodatorns datablad. När kommunikationen väl fungerar går utvecklingen snabbt framåt, och den största utmaningen blir att lösa en kontinuerlig styrning av servona. Målet är att få en så mjuk och jämn rörelse som möjligt, men rörelserna blir då för korta, vilket gör att armen tappar styrka. För att lösa detta ökas den minsta rörelsesträckan, vilket ger en något hackigare rörelse, men armen blir betydligt starkare.

5.4 PC

Implementationen av användargränssnittet tar mycket tid men är relativt enkel. Det främsta paketet som används är tkinter, vilket möjliggör olika val i design och format och är väldigt lätt att arbeta med. Grunderna består i att skapa de olika rutorna som behövs för att uppfylla kraven, såsom datavisning, manuell styrning och kartläggning.

Därefter utvecklas gränssnittet med funktioner där knappar och rutor fryser vid knapptryck för att skilja mellan manuell och autonom styrning. Användarvänligheten förbättras med hjälp av färger och andra designval. I en av rutorna visas styrbeslutet, som består av kommandon som exempelvis "Höger" och "Plocka". När varor ska plockas upp visas texten "Vara" följt av de noder varan ligger mellan, så att användaren i god tid vet i vilken ordning varorna ska hämtas. När hinder detekteras tas nya styrbeslut emot och uppdateras i användargränssnittet (GUI:n).

6 RESULTAT

Roboten används i manuellt eller autonomt läge. I manuellt läge styr användaren roboten via användargränssnittet. Detta kräver en Bluetooth-uppkoppling mellan användarens dator och roboten. Se en mer detaljerad beskrivning i Appendix 14.

I autonomt läge lägger användaren ut varor i användargränssnittet. När användaren startar ett autonomt uppdrag kalibreras linjesensorerna genom att placera den främre linjesensorn över en tejbit. Roboten beräknar sedan en optimal väg, hämtar varorna och lämnar tillbaka dem vid återlämningsstationen. Om hinder uppstår på vägen undviker roboten dessa och beräknar en ny optimal väg. Flera sensor- och styrvärden visas också i användargränssnittet. Se mer detaljerad beskrivning i Appendix 14.

Den sluttgiltiga produkten klarar huvudsakligen de krav som ställs i kravspecifikationen. Till första delleveransen BP5A uppfylls dock inte alla prioritet 1-krav i tid. Dessa krav samt resterande prioritet 2-krav uppnås ändemot till slutleveransen BP5B. Däremot uppfylls inga av prioritet 3-kraven, de som ska arbetas mot i mån av tid.

7 SLUTSATS

Sammanfattningsvis är gruppen nöjd med resultatet. Kommunikationen mellan alla gruppmedlemmar är bra, vilket underlättar genomförandet av projektet. Alla är delaktiga och samlar på sig mycket kunskap, särskilt inom de specifika områden som respektive medlem arbetar med. Uppdelningen av arbetet i separata enheter gör att varje person blir väldigt duktig på just sin del, men det gör det också svårare att sätta sig in i de andra gruppmedlemmarnas kod. Detta löser gruppen effektivt genom att de flesta mot slutet jobbar tillsammans på samma problem, vilket underlättar förståelsen för kod man själv inte har skrivit. Resultaten uppfyller kraven som specificeras i kravspecifikationen. Gruppen är nöjd med genomförandet då upplägget med mindre smågrupper gör arbetet tidseffektivt.

Om gruppen hade gjort om projektet planerar den att lägga större fokus på bättre aktivitetsplanering och utveckling av verktyg för effektivare kodtestning tidigare i processen. Regleringen för linjeföljning är mycket tidskrävande. Att hitta rätt värden för K_P och K_D tar lång tid, dels för att det kräver många tester, dels för att K_P och K_D behöver ändras i Raspberry Pi-koden. Detta innebär att enheten måste kopplas till datorn och startas om varje gång nya värden ska testas, vilket är tidsödande. Om det istället går att ändra K_P och K_D direkt via användargränssnittet skulle mycket tid sparas. En annan förbättring är att lägga mer tid på aktivitetsplaneringen från början. Fler och mer detaljerade aktiviteter skulle göra det tydligare och underlätta planeringen av arbetet.

Om projektet har mer tid finns det två huvudsakliga områden som kan vidareutvecklas. Gruppen utnyttjar mycket under projektet att hårdkoda olika värden, främst för att förenkla lösningarna och för att hinna klart i tid. Med mer tid vill gruppen försöka skapa mer generella lösningar som är mer applicerbara i riktiga tillämpningar. Sensoransvariga vill gärna testa andra sensorer, som till exempel kameran till Raspberry Pi och LIDAR. Detta kan potentiellt möjliggöra en mer generell autonom uppluckning av varor jämfört med en hårdkodad rörelse. Det är dessutom intressant att få roboten att ta den snabbaste vägen istället för den kortaste vägen. Den kortaste vägen kan involvera många svängar, och dessa tar tid. Det är inte alltid den kortaste vägen som är den snabbaste, och det är något som gruppen gärna vill försöka implementera.

För att göra projektet mer intressant i framtiden kan man införa en maxlast, alltså en begränsning på hur många varor roboten kan ta med sig under en körning. Detta, i kombination med att roboten blir längsammare ju fler varor den bär för att simulera vikt, gör projektet mer intressant ur en optimeringssynpunkt. Är det värt att åka färre gånger men längsammare, eller fler gånger och snabbare?

REFERENSER

- [1] ISY, *Vanheden*, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/>, 2025.

Appendix

8 PROJEKTDIREKTIV

TSEA56: Projektdirektiv för en lagerrobot

Beställare: Mattias Krysander
Datum: 2024-01-09

Inledning

Vi har tänkt oss att starta tillverkning av robotar för lagerhantering. Robotens väg genom en fabrik eller ett lager markeras med en svart linje på golvet. De positioner, ”plockstationer”, där roboten kan stanna för att hämta eller lämna en vara markeras på ett speciellt sätt på golvet. Lagerroboten ska ha en arm med vilken den kan plocka upp varor i lagret. Roboten ska kunna hämta beställda varor i lagret och lämna dem på en utlämningsstation. För att utvärdera hur man kan bygga en sådan lagerrobot så önskar vi beställa ett antal prototyper. Dessa ska delta i en tävling där vi kan utvärdera olika konstruktionsalternativ. För att få olika konstruktioner vill vi att ni skriver kravspecifikationen i dialog med oss. Nedan ges ett antal grundkrav som ska vara gemensamma i alla kravspecifikationer. Kraven är inte numrerade vilket de dock ska vara i era kravspecifikationer. Ni förväntas lägga till mer text som beskriver roboten i allmänna termer, figurer samt unika krav för just er robot. Kom ihåg att krav kan prioriteras.

Uppdraget

Roboten ska kunna manövrera autonomt i en bana enligt banspecifikationen nedan och så snabbt som möjligt hämta ett antal varor från olika plockstationer och lämna dem på en utlämningsstation. Det är tillåtet att hämta alla varor i en körning eller varorna en och en. Banan har ett känt utseende och positionerna på de varor som ska hämtas matas in via gränssnittet på en bärbar dator och skickas sedan trådlöst till roboten. Varorna ska plockas upp antingen fjärrstyrts från en dator utan att se roboten eller autonomt. Varorna ska lämnas autonomt vid utlämningsstationen. Det kan finnas tillfälliga hinder i lagret. Om så är fallet så ska roboten använda en alternativ väg. Banan består av svart tejp på vitt underlag. Tejpens bredd ligger i intervallet 14-18 mm. Banan består av ett mönster av rektanglar där hyllor (plockstationer) finns. Det ska gå att definiera antalet rektanglar i X- och Y-led via gränssnittet. Från lagrets ena hörn finns en tejpad linje till utlämningsstationen. Linjen till utlämningsstationen kan innehålla kurvor. Kurvradien kommer inte att understiga 25 cm. Utlämningsstationen markeras med ett tvärstreck som är 30 cm. Banan har indikeringar för plockstationerna. Dessa är alltid vinkelräta mot banan. Indikeringen för en plockstation har en längd på minst 10 cm och finns enbart på den sida där plockstationen befinner sig. Detaljer i banspecifikationen bestäms i samråd med beställaren.

Systemarkitektur

Systemet ska bestå av en robot samt mjukvara till dator för styrning och övervakning. För att senare kunna testa alternativa sensorer, fjärrstyrningar och även styralgoritmer, ska roboten vara moduluppbyggd. Gränssnitten mellan modulerna ska vara noggrant specificerade. Man ska enkelt kunna byta ut en modul mot en annan. Varje modul ska innehålla minst en egen processor. Följande tre moduler ska ingå i konstruktionen:

- Kommunikationsenhet (trådlös kommunikation, wifi eller blåtand)
- Styrenhet (motorer, arm, display, styrlogik)
- Sensorenhet (innehåller alla sensorer och signalbehandling)

Fjärrstyrning och övervakning

Robotplattformen ska kunna styras trådlöst. Följande kommandon ska finnas: Fram, fram vänster, fram höger, rotera höger, rotera vänster, back, stopp och start av autonomt uppdrag. Robotarmens samtliga motorer ska också kunna fjärrstyras genom att styra gripklons läge framåt, bakåt, uppåt, nedåt, vänster, höger, grip och släpp. Om linjesensorerna behöver kalibreras ska detta kunna initieras från laptopen. Under körning i banan ska roboten fortlöpande skicka mätdata från linjesensorerna, lateral position, avstånd till eventuellt hinder, motorernas gaspådrag och styrbeslut som t ex planerad färdväg. När armen är aktiv ska även gripklons skattade position visas. Dessa data ska presenteras på datorns skärm på ett användarvänligt sätt och dessutom gå att spara på ett format som gör det möjligt att plotta valda signaler som funktion av tiden för en given körning i t ex Matlab eller Python. Det vore trevligt med en grafisk visualisering av lagret där robotens position, varorna som ska hämtas, planerad färdväg och detekterade hinder visas.

Diverse övriga funktionskrav

Det ska finnas en brytare på roboten med vilken man väljer fjärrstyrningsläge eller autonomt läge. Det ska finnas någon form av styralgoritm (exempelvis PD-reglering), så att roboten kan följa linjen utan att ”slingra” sig fram (verifieras genom dokumenterade testkörningar). Tolkningen av signalerna från linjesensorerna ska dokumenteras noga i den tekniska dokumentationen. Parametrar till robotens styralgoritm ska kunna initieras trådlöst från datorn. Robotarmen ska ha mjuka rörelser och inte ”hacka” sig fram för att inte överbelasta servona.

Tävlingsregler:

Vinnare är den robot som på kortast tid placerar alla specificerade varor på utlämningsplatsen. Repeterbarhet ska uppvisas. Reglerna bestäms i samråd med beställaren.

Övriga krav

Projektet ska bedrivas enligt LIPS-modellen och samtliga dokument ska utgå från LIPS-mallar. I förefasen ingår att projektgruppen ska ta fram en kravspecifikation, en systemskiss och en projektplan med tidplan. Samtliga dessa dokument ska godkännas av beställaren. Budget för förefasen finns på beställarens hemsida. Efter godkänd projektplan (BP2) ska projektet ta 230 arbets timmar/person att slutföra. Vid verifiering av baskrav (BP5a) ska autonom linjeföljning och manuell armstyrning demonstreras. Detaljer för baskrav utarbetas i samråd med beställare. Vid slutleveransen (BP5b) ska det finnas en fungerande robot samt teknisk dokumentation med användaranvisning. Projektets delleveranser och slutleverans ska senast ske vid de datum som finns specificerade på beställarens hemsida. Även formen för slutleveransen beskrivs på hemsidan.

9 KRAVSPECIFIKATION

Kravspecifikation

Grupp 6

29 april 2025

Version 1.2



Status

Granskad	Ebba Lundberg	2025-04-29
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

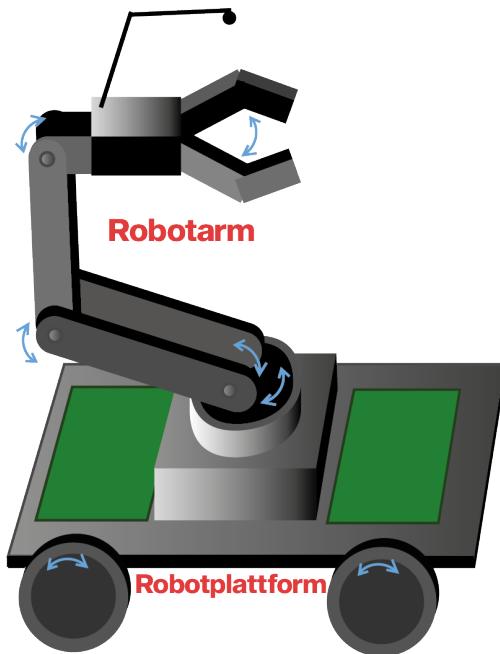
1	Inledning	1
1.1	Parter	1
1.2	Syfte och mål	2
1.3	Användning	2
1.4	Bakgrundsinformation	2
2	Översikt av systemet	2
2.1	Grov beskrivning av produkten	3
2.2	Produktkomponenter	3
2.3	Beroenden till andra system	3
2.4	Ingående delsystem	3
2.5	Avgränsningar	3
2.6	Generella krav på hela systemet	3
3	Banutformning och varor	4
3.1	Krav på banutformning och varor	5
4	Kommunikationsenhet	5
4.1	Krav på kommunikationenheten	5
5	Sensorenhet	6
5.1	Krav på sensorenheten	6
6	Styrenhet	6
6.1	Krav på styrenheten	6
7	Persondator	7
7.1	Krav på persondator	7
8	Prestandakrav	8
9	Tillförlitlighet	8
10	Ekonomi	8
11	Leveranskrav och delleveranser	8
12	Dokumentation	10
	Referenser	11
13	Appendix	12
A	Projektbeskrivning för lagerrobot	12

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-01-29	Första utkast	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS
0.2	2025-01-31	Andra utkast	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS
1.0	2025-02-03	Första version	LF	LF, EL
1.1	2025-03-03	Omförhandling av krav	AN, EL	AN, EL
1.2	2025-04-29	Omförhandling av krav	AN, EL	AN, EL

1 INLEDNING

Detta dokument är en kravspecifikation för en produkt som ska tas fram i kursen TSEA56, Elektronik kandidatprojekt. Produkten som ska tas fram är en lagerrobot, se figur 1, bestående av en robotplattform och en robotarm med en gripklo. Roboten ska kunna navigera i en känd lagermiljö med specifika plockstationer där varor är placerade i förväg. Vid plockstationerna ska varorna plockas upp med hjälp av robotarmen för att sedan lämnas vid en utlämningsstation.



Figur 1: Lagerrobot med robotarm och robotplattform.

Kraven på det färdiga systemet kommer att beskrivas som i tabell 1 där varje krav har ett kravnummer, version, beskrivning och prioritet. De olika prioriteringarna anger när kravet ska vara uppfyllt. Prioritet 1 ska vara uppfyllt vid BP5a, prioritet 2 vid BP5b och prioritet 3 i mån av tid när resterande krav är uppfyllda.

Tabell 1. Exempel på ett krav.

Krav	Version	Beskrivning	Prioritet
1.1	Original	Exempel på ett krav	1

1.1 Parter

Kravspecifikationen är framtagen i samråd med gruppen och beställaren. Gruppen utgörs av Linus Funquist, Ebba Lundberg, Andreas Nordström, Sigge Rystedt, Ida Sonesson och Lisa Ståhl. Beställare är Mattias Krysander.

1.2 Syfte och mål

Projektets syfte är att konstruera ett system som konstruktionsalternativ till en lagerrobot, som med snabbhet och repeterbarhet kan utföra sitt uppdrag. I projektdirektivet, se appendix A, beskrivs lagerroboten som en prototyp till en robot som ska tillverkas. Denna kommer prövas i en tävling för att utvärdera hur väl målen kring snabbhet och repeterbarhet är uppfyllda.

1.3 Användning

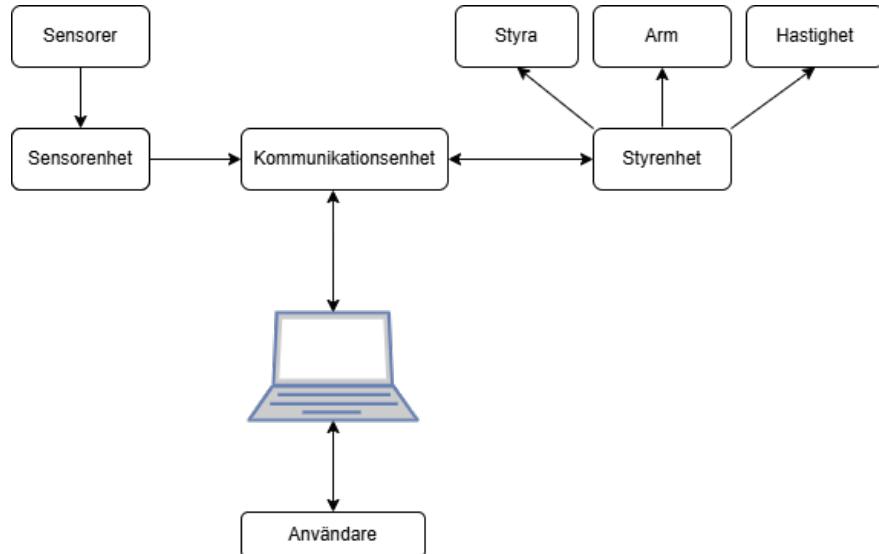
För att använda systemet placeras roboten vid utlämningsstationen, därifrån den autonomt navigerar till en plockstation. Användaren kontrollerar sedan robotarmen manuellt för att plocka upp varan och återaktiverar sedan det autonoma systemet, då bär roboten tillbaka varan till utlämningsstationen innan denna process återupprepas tills alla varor är upplockade. Systemet kommer dessutom att kunna styras fullständigt manuellt.

1.4 Bakgrundsinformation

Kraven är grundade i projektdirektivet, se appendix A. Projektmodellen som beskrivs i *Projektmodellen LIPS* av Svensson och Krysander [1] kommer att följas.

2 ÖVERSIKT AV SYSTEMET

Systemet består av fyra delsystem. Kommunikationsenheten, styrenheten och sensorenheten sitter på roboten. Det fjärde systemet, persondatorn, kommunicerar med kommunikationsenheten trådlöst. En översikt av hela systemet visas i figur 2.



Figur 2: Generell skiss av systemet i sin helhet.

2.1 Grov beskrivning av produkten

Systemet ska designas så att roboten både kan styras manuellt via en dator och autonomt. Lagermiljön, som beskrivs under rubrik 3, är känd för systemet och består av ett rutnät med hinder, plockstationer och en väg till utlämningsstationen. Roboten ska kunna börja vid utlämningsstationen, autonomt följa en linje till lagret med hjälp av sensorer och kunna hitta tillbaka med upplockade varor. I själva lagret ska den kunna undvika hinder och navigera till plockstationer där varorna ligger.

2.2 Produktkomponenter

Roboten kommer att bestå av en robotplattform som har fyra oberoende körbara hjul samt en robotarm som kan plocka upp och släppa objekt. På robotplattformen kommer det även att sitta sensorer och flera mikrodatorer. I systemet kommer det utöver nämnda komponenter att ingå en persondator.

2.3 Beroenden till andra system

Det finns inget beroende till andra system, alla ingående system är beskrivna i denna kravspecifikation.

2.4 Ingående delsystem

Det finns fyra delsystem. Dessa är kommunikationsenheten, sensorenheten, styrenheten och persondatorn. Dessa är beskrivna i detalj i kapitel 3-6.

2.5 Avgränsningar

Persondatorn får ej användas till tyngre beräkningar.

Varje projektmedlem får ej överstiga 230 arbetsimmar efter BP2.

2.6 Generella krav på hela systemet

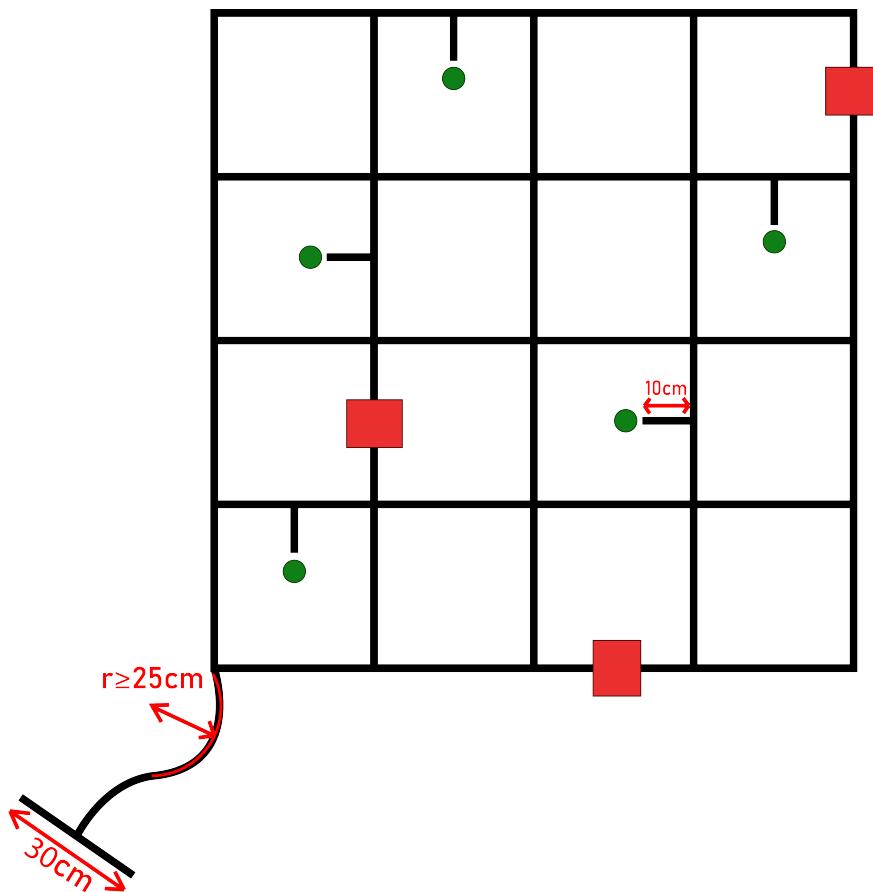
Följande tabell beskriver kraven på hela systemet.

Krav	Version	Beskrivning	Prioritet
2.1	Original	Systemet ska kunna navigera en banan som utformas efter kraven 3.1 till 3.11	2
2.2	Original	Följande kommandon ska finnas för robotplattformen: fram, fram höger, fram vänster, rotera höger, rotera vänster, back, stopp och start av autonomt uppdrag	1
2.3	Original	Följande kommandon ska finnas för robotarmen: framåt, bakåt, uppåt, nedåt, rotera höger, rotera vänster, grip och släpp	1
2.4	Original	Varje modul ska innehålla minst en egen processor	1
<i>forts. på nästa sida</i>			

forts. från föregående sida			
Krav	Version	Beskrivning	Prioritet
2.5	Original	Varorna ska plockas upp fjärrstyrts från en dator utan att se roboten	3
2.6	Original	Robotarmen ska autonomt kunna plocka upp varor	2
2.7	Original	Gränssnitten mellan modulerna ska vara noggrant specificerade i den tekniska dokumentationen	2

3 BANUTFORMNING OCH VAROR

Systemet kommer att navigera en bana, se figur 3, som representerar ett lager eller fabrik med varor.



Figur 3: Lagermiljön med specificerade mått. Röda fyrkanter är hinder och gröna prickar är plockstationer.

3.1 Krav på banutformning och varor

Följande tabell beskriver kraven för banan och varorna.

Krav	Version	Beskrivning	Prioritet
3.1	Original	Banan ska vara markerad med svart tejp på vitt underlag	1
3.2	Original	Tejpens bredd ska vara mellan 14-18 mm	1
3.3	Original	Banan ska utgöras av rektanglar med plockstationer	1
3.4	Original	Antalet rektanglar som utgör banan ska gå att definiera i X- och Y-led via gränsnittet	1
3.5	Original	En tejpad linje ska gå från banans ena hörnet till utlämningsstationen	1
3.6	Original	Linjen till utlämningsstationen kan innehålla kurvor	1
3.7	Original	Kurvradien på linjen kommer inte att understiga 25 cm	1
3.8	Original	Utlämningsstationen ska vara markerad med ett 30 cm långt tvärstreck	1
3.9	Original	Banan ska ha indikeringar för plockstationerna	1
3.10	Original	Indikeringarna för plockstationerna ska alltid vara vinkelräta mot banan och 10 cm långa	1
3.11	Original	Banan ska kunna innehålla tillfälliga hinder	1
3.12	Original	Varorna ska vara ungefär samma storlek som en tändsticksask	1

4 KOMMUNIKATIONSENHET

Kommunikationsenheten interagerar med styr- och sensorenheten samt persondatorn genom att skicka och ta emot data mellan de olika enheterna.

4.1 Krav på kommunikationsenheten

Följande tabell beskriver kraven för kommunikationsenheten.

Krav	Version	Beskrivning	Prioritet
4.1	Original	Kommunikationsenheten ska kunna ta emot data från sensorenheten	1
4.2	Original	Kommunikationsenheten ska kunna skicka data till styrenheten	1
4.3	Original	Användaren ska kunna kommunicera med kommunikationsenheten via trådlös kommunikation	1
4.4	Original	Kommunikationsenheten ska få plats på robotplattformen	1

5 SENSORENHET

Sensorenheten samlar in mätvärden, bearbetar dessa till användbar data och vidarebefordrar dem till kommunikationsenheden. Sensorenheten består av en mikrodator och utbytbara sensorer. Alla sensorer levererar data till mikrodatorn, som omvandlar informationen till SI-enheter.

5.1 Krav på sensorenheten

Följande tabell beskriver kraven för sensorenheten.

Krav	Version	Beskrivning	Prioritet
5.1	Original	Hela sensorenheten ska få plats på robotplattformen	1
5.2	Original	Sensorerna ska kunna bytas ut	1
5.3	Original	Sensorerna ska kopplas till mikrodatorn	1
5.4	Original	Sensorenheten ska kunna samla in data	1
5.5	Original	Sensorenheten ska kunna bearbeta och omvandla data till ett format som är läsbart och hanterbart för kommunikationseenheten	1

6 STYRENHET

Styrenheten har som uppgift att manövrera roboten genom att kontrollera hjulen på robotplattformen samt hantera robotarmen enligt figur 1. Dessa funktioner utförs baserat på data som tas emot från kommunikationseenheten.

6.1 Krav på styrenheten

Följande tabell beskriver kraven för styrenheten.

Krav	Version	Beskrivning	Prioritet
6.1	Original	Hela styrenheten ska få plats på robotplattformen	1
6.2	Original	Styrenheten ska kunna ta emot instruktioner från kommunikationseenheten	1
6.3	Original	Robotplattformen ska kunna utföra kommandon givna av kommunikationseenheten	1
6.4	Original	Robotarmen ska kunna utföra kommandon givna av kommunikationseenheten	1

7 PERSONDATOR

Persondatorn tar emot data trådlöst från kommunikationsenheten. Från datorn ska det vara möjligt att köra roboten och kontrollera robotarmen.

7.1 Krav på persondator

Följande tabell beskriver kraven för persondatorn.

Krav	Version	Beskrivning	Prioritet
7.1	Original	Det ska vara möjligt att styra roboten fjärrstyrts från datorn via användargränssnittet	1
7.2	Original	Det ska vara möjligt att styra robotarmen fjärrstyrts från datorn via användargränssnittet	1
7.3	Original	Kalibrering av linjesensor ska kunna initieras från persondatorn	1
7.4	Original	Data* ska representeras på persondatorns skärm på ett användarvänligt sätt	1
7.5	Original	GUIt ska visa gripklons skattade position relativt till robotens position, när armen är aktiv.	2
7.6	Original	Datan beskriven i krav 7.4 och 7.5 ska gå att spara på ett format som gör det enkelt att plotta valda signaler som funktion av tiden	2
7.7	Original	Persondatorn ska grafiskt visualisera lagret, inklusive robotens position, de varor som ska hämtas, den planerade färdvägen och identifierade hinder	3
7.8	Original	Sensor och styrdata (nämnda i krav 7.4 och 7.5) ska uppdateras minst 1 gång per sekund.	1

*** Följande data ska ingå:**

- Lateral position från linjesensorer
- Avstånd till eventuellt hinder
- Om sensor för rotation finns så ska robotens rotation i grader visas när den roterar i korsningar.
- Motorernas gaspådrag (vänster/höger)
- Styrbeslut:
 - Planerad besöksordning av upphämtningsplatserna
 - Planerad färdväg till nästa upphämtningsplats representerad som en sekvens av vägval i korsningarna (vänster/höger/rakt)

8 PRESTANDAKRAV

För att säkerställa att roboten uppfyller en lägsta standard listas prestandakrav nedan. En fullständig körsim är att roboten autonomt hämtar tre varor i lagret, undviker hinder och lämnar varorna på en specificerad utlämningsplats.

Krav	Version	Beskrivning	Prioritet
8.1	Original	Upplockning får ske autonomt utan visuell kontakt med roboten	2
8.2	Original	Roboten ska klara av att genomföra tre av fyra körsimmar	2
8.3	Original	Roboten ska ej slingra sig fram under körsimning med hjälp av styralgoritmer som PD-reglering	2

9 TILLFÖRLITLIGHET

Följande krav beskriver produktens tillförlitlighet.

Krav	Version	Beskrivning	Prioritet
9.1	Original	Robotarmen ska kunna utföra rörelser på ett hållbart sätt genom att endast använda sig av mjuka rörelser och undvika extremlägen	1
9.2	Original	Roboten ska ej skada gods eller nudda eventuella hinder i lagermiljön	2

10 EKONOMI

Följande tabell beskriver den planerade budgeten över projektets arbete.

Krav	Version	Beskrivning	Prioritet
10.1	Original	Efter BP2 har varje gruppmedlem 230 timmar till sitt förfogande	2

11 LEVERANSKRAV OCH DELLEVERANSER

Respektive leverans ska vara inlämnad senast kl. 16:00 på angivet datum.

Krav	Version	Beskrivning	Prioritet
11.1	Original	Kravspecifikationen v0.1 ska lämnas in till beställare via git senast 30:e januari	1
11.2	Original	Kravspecifikationen v1.0 ska vara klar samt tidsrapportering för respektive gruppmedlem ska meddelas senast 6:e februari	1
11.3	Original	Version 0.1 av projektplan, tidplan och systemskiss ska vara inlämnade till beställare senast 14:e februari	1
11.4	Original	Slutlig version av projektplan, tidplan och systemskiss ska vara inlämnade till beställaren senast 24:e februari	1
11.5	Original	Första version av förstudien på minst fem sidor ska skickas till skrivuppgiftshandledare senast 24:e februari	1
11.6	Original	Version 0.1 av designspecifikationen ska vara inlämnad till handledaren via git senast 4:e mars	1
11.7	Original	Version 1.0 av designspecifikationen ska vara inlämnad till handledaren senast 12:e mars	1
11.8	Original	Första veckorapporten ska lämnas in senast 31:a mars	1
11.9	Original	Version 1.0 av förstudien ska skickas till skrivuppgiftshandledare senast 7:e april	1
11.10	Original	Skrivuppgiften ska vara godkänd och skickas till beställare senast 29:e april	1
11.11	Original	Veckorapport med utökad statusrapport enligt angivna frågor ska lämnas in senast 5:e maj	1
11.12	Original	Version 1.0 av kappan (exklusive appendix) ska levereras senast 21:a maj	1
11.13	Original	Teknisk dokumentation och användarhandledning (båda version 1.0) ska vara inlämnade till beställare senast 21:a maj	1
11.14	Original	Efterstudien ska vara inlämnad senast 9:e juni	1
11.15	Original	Komplett och väldokumenterad källkod ska vara incheckad på git senast 9:e juni	1
11.16	Original	Komplett kandidatrapport inlämnad som en pdf-fil med alla appendix inkluderade ska vara inlämnad senast 9:e juni	1
11.17	Original	Veckorapport, inkluderande tidrapport och statusrapport, ska lämnas in till beställare följande datum: 31/3, 7/4, 14/4, 22/4, 28/4, 5/5, 12/5, 19/5, 26/5, 2/6, 9/6	1
11.18	Original	All utrustning och nycklar ska vara återlämnade till beställare senast 9:e juni	1

12 DOKUMENTATION

Tabell 12 listar de dokument som ska produceras. Projektet ska bedrivas enligt LIPS-modellen och samtliga dokument ska utgå från LIPS-mallar.

Tabell 12: Dokument som skall produceras.

Dokument	Språk	Syfte	Målgrupp	Format
Projektplan	Svenska	Ett dokument för att vägleda och realisera projektgenomförandet	Beställare	Excel
Kravspecifikation	Svenska	Samling av krav för det som ska produceras. Utgör grund för systemskiss och övriga specifikationer och projektdokument	Beställare	PDF
Designspecifikation	Svenska	En detaljbeskrivning över produktens konstruktion	Handledare	PDF
Systemskiss	Svenska	Översiktlig representation av produktens samtliga delar och dess komponenter med funktion. Ger även en överblick över arbetets uppdelning	Beställare	PDF
Förstudier	Svenska	Fördjupning för samtliga projektmedlemmar inom relevanta och nödvändiga områden	Beställare	PDF
Mötesprotokoll	Svenska	Dokument över samtliga överenskommelser under planerade möten	Beställare	PDF
Teknisk dokumentation	Svenska	Tydligt förmedla information över funktion och användning av produkten för framtidens användare	Beställare	PDF
Användarmanual	Svenska	Beskrivning över hur produkten används	Beställare	PDF
Efterstudie	Svenska	Utvärdering samt analys över projektets arbete för att identifiera brister och lärdomar för vidareutveckling	Beställare	PDF

REFERENSER

[1] *Projektmodellen LIPS* (2011), Tomas Svensson och Christian Krysander, uppl. 1:1, Studentlitteratur AB, Lund, ISBN 978-91-44-07525-9

13 APPENDIX

A PROJEKTBESKRIVNING FÖR LAGERROBOT

TSEA56: Projektdirektiv för en lagerrobot

Beställare: Mattias Krysander
Datum: 2024-01-09

Inledning

Vi har tänkt oss att starta tillverkning av robotar för lagerhantering. Robotens väg genom en fabrik eller ett lager markeras med en svart linje på golvet. De positioner, ”plockstationer”, där roboten kan stanna för att hämta eller lämna en vara markeras på ett speciellt sätt på golvet. Lagerroboten ska ha en arm med vilken den kan plocka upp varor i lagret. Roboten ska kunna hämta beställda varor i lagret och lämna dem på en utlämningsstation. För att utvärdera hur man kan bygga en sådan lagerrobot så önskar vi beställa ett antal prototyper. Dessa ska delta i en tävling där vi kan utvärdera olika konstruktionsalternativ. För att få olika konstruktioner vill vi att ni skriver kravspecifikationen i dialog med oss. Nedan ges ett antal grundkrav som ska vara gemensamma i alla kravspecifikationer. Kraven är inte numrerade vilket de dock ska vara i era kravspecifikationer. Ni förväntas lägga till mer text som beskriver roboten i allmänna termer, figurer samt unika krav för just er robot. Kom ihåg att krav kan prioriteras.

Uppdraget

Roboten ska kunna manövrera autonomt i en bana enligt banspecifikationen nedan och så snabbt som möjligt hämta ett antal varor från olika plockstationer och lämna dem på en utlämningsstation. Det är tillåtet att hämta alla varor i en körning eller varorna en och en. Banan har ett känt utseende och positionerna på de varor som ska hämtas matas in via gränssnittet på en bärbar dator och skickas sedan trådlöst till roboten. Varorna ska plockas upp antingen fjärrstyrts från en dator utan att se roboten eller autonomt. Varorna ska lämnas autonomt vid utlämningsstationen. Det kan finnas tillfälliga hinder i lagret. Om så är fallet så ska roboten använda en alternativ väg. Banan består av svart tejp på vitt underlag. Tejpens bredd ligger i intervallet 14-18 mm. Banan består av ett mönster av rektanglar där hyllor (plockstationer) finns. Det ska gå att definiera antalet rektanglar i X- och Y-led via gränssnittet. Från lagrets ena hörn finns en tejpad linje till utlämningsstationen. Linjen till utlämningsstationen kan innehålla kurvor. Kurvradien kommer inte att understiga 25 cm. Utlämningsstationen markeras med ett tvärstreck som är 30 cm. Banan har indikeringar för plockstationerna. Dessa är alltid vinkelräta mot banan. Indikeringen för en plockstation har en längd på minst 10 cm och finns enbart på den sida där plockstationen befinner sig. Detaljer i banspecifikationen bestäms i samråd med beställaren.

Systemarkitektur

Systemet ska bestå av en robot samt mjukvara till dator för styrning och övervakning. För att senare kunna testa alternativa sensorer, fjärrstyrningar och även styralgoritmer, ska roboten vara moduluppbyggd. Gränssnitten mellan modulerna ska vara noggrant specificerade. Man ska enkelt kunna byta ut en modul mot en annan. Varje modul ska innehålla minst en egen processor. Följande tre moduler ska ingå i konstruktionen:

- Kommunikationsenhet (trådlös kommunikation, wifi eller blåtand)
- Styrenhet (motorer, arm, display, styrlogik)
- Sensorenhet (innehåller alla sensorer och signalbehandling)

Fjärrstyrning och övervakning

Robotplattformen ska kunna styras trådlöst. Följande kommandon ska finnas: Fram, fram vänster, fram höger, rotera höger, rotera vänster, back, stopp och start av autonomt uppdrag. Robotarmens samtliga motorer ska också kunna fjärrstyras genom att styra gripklons läge framåt, bakåt, uppåt, nedåt, vänster, höger, grip och släpp. Om linjesensorerna behöver kalibreras ska detta kunna initieras från laptopen. Under körning i banan ska roboten fortlöpande skicka mätdata från linjesensorerna, lateral position, avstånd till eventuellt hinder, motorernas gaspådrag och styrbeslut som t ex planerad färdväg. När armen är aktiv ska även gripklons skattade position visas. Dessa data ska presenteras på datorns skärm på ett användarvänligt sätt och dessutom gå att spara på ett format som gör det möjligt att plotta valda signaler som funktion av tiden för en given körning i t ex Matlab eller Python. Det vore trevligt med en grafisk visualisering av lagret där robotens position, varorna som ska hämtas, planerad färdväg och detekterade hinder visas.

Diverse övriga funktionskrav

Det ska finnas en brytare på roboten med vilken man väljer fjärrstyrningsläge eller autonomt läge. Det ska finnas någon form av styralgoritm (exempelvis PD-reglering), så att roboten kan följa linjen utan att ”slingra” sig fram (verifieras genom dokumenterade testkörningar). Tolkningen av signalerna från linjesensorerna ska dokumenteras noga i den tekniska dokumentationen. Parametrar till robotens styralgoritm ska kunna initieras trådlöst från datorn. Robotarmen ska ha mjuka rörelser och inte ”hacka” sig fram för att inte överbelasta servona.

Tävlingsregler:

Vinnare är den robot som på kortast tid placerar alla specificerade varor på utlämningsplatsen. Repeterbarhet ska uppvisas. Reglerna bestäms i samråd med beställaren.

Övriga krav

Projektet ska bedrivas enligt LIPS-modellen och samtliga dokument ska utgå från LIPS-mallar. I förefasen ingår att projektgruppen ska ta fram en kravspecifikation, en systemskiss och en projektplan med tidplan. Samtliga dessa dokument ska godkännas av beställaren. Budget för förefasen finns på beställarens hemsida. Efter godkänd projektplan (BP2) ska projektet ta 230 arbets timmar/person att slutföra. Vid verifiering av baskrav (BP5a) ska autonom linjeföljning och manuell armstyrning demonstreras. Detaljer för baskrav utarbetas i samråd med beställare. Vid slutleveransen (BP5b) ska det finnas en fungerande robot samt teknisk dokumentation med användaranvisning. Projektets delleveranser och slutleverans ska senast ske vid de datum som finns specificerade på beställarens hemsida. Även formen för slutleveransen beskrivs på hemsidan.

10 SYSTEMSKISS

Systemskiss

Grupp 6

2025-03-03

Version 1.1



Status

Granskad	Ebba Lundberg	2025-03-03
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno773@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Beställare	1
2	Inledning	1
3	Översikt av systemet	1
4	Kommunikationsenhet	2
4.1	Översikt av kommunikationsenheten	2
4.2	Raspberry Pi	2
5	Sensorenhet	3
5.1	Översikt av sensorenheten	3
5.2	AVR-mikrodator	4
5.3	Avståndssensorer	4
5.4	Reflexsensorer	4
5.5	Kamera	4
5.6	Gyro	4
5.7	Odometer	4
6	Styrenhet	5
6.1	Översikt av styrenheten	5
6.2	Hjul	5
6.3	Robotarm	5
6.4	Mikrodatorn	5
7	Persondator	6
	Referenser	7

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-02-12	Första utkast	EL, AN, IS, LS, SR, LF	EL, AN, SR, LS
1.0	2025-02-24	Första version	EL, AN, IS, LS, SR, LF	AN

1 BESTÄLLARE

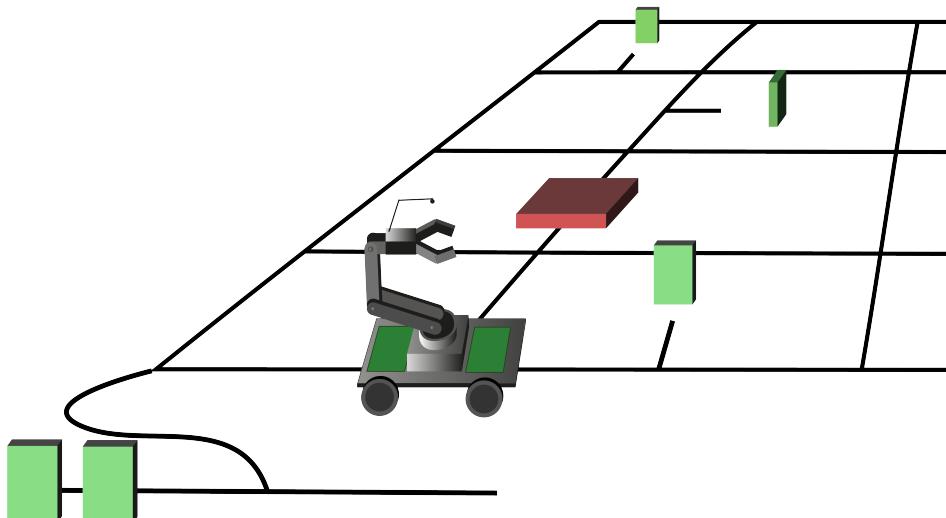
Beställare av projektet är Mattias Krysander på ISY, LiU.

2 INLEDNING

Detta dokument är en systemskiss i kursen TSEA56, Elektronik kandidatprojekt, som ska ge en översikt av den lagerroboten som ska tas fram. Lagerroboten ska fungera som beskrivet i kravspecifikationen. Systemskissen ger en översikt av systemet, dess ingående delsystem och en kort beskrivning av deras funktion.

3 ÖVERSIKT AV SYSTEMET

Systemet består av fyra delsystem: kommunikationsenheten, sensorenheten, styrenheten och en persondator. Se figur 5 för en enkel översikt. Delsystemen är beskrivna närmare i kapitel 4-7.



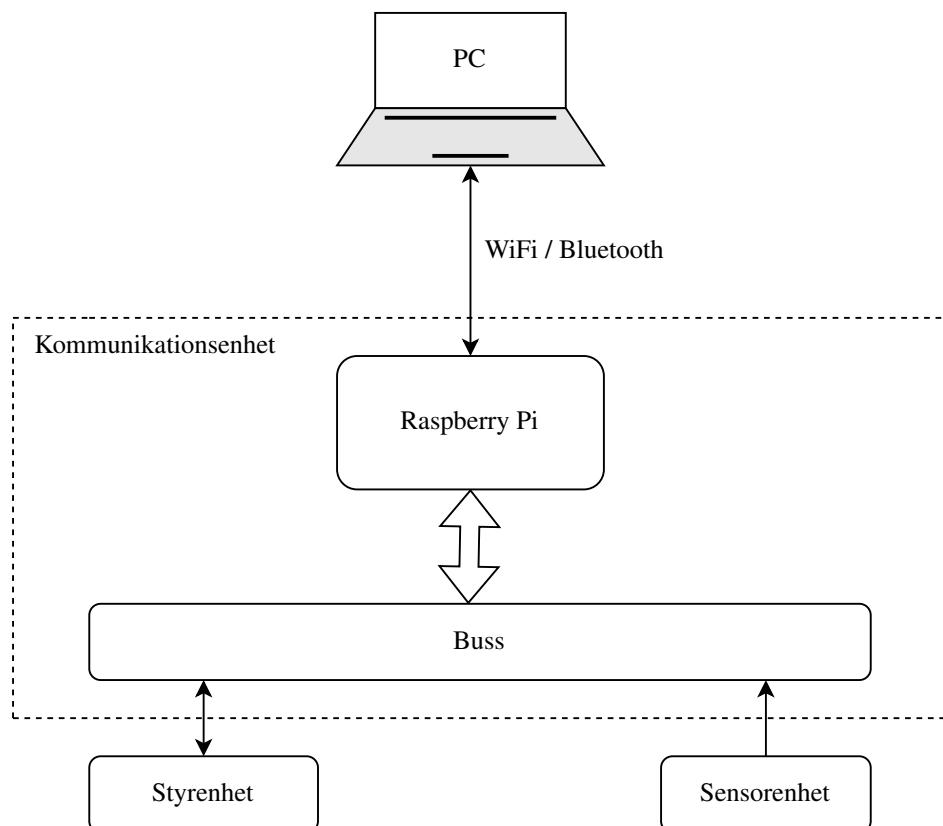
Figur 1: Översikt av system.

4 KOMMUNIKATIONSENHET

I följande kapitel följer översiktlig information om kommunikationsenheten.

4.1 Översikt av kommunikationenheten

Kommunikationenheten, se figur 2, har två huvudsakliga uppgifter beroende på läge. I autonomt läge är dess främsta uppgift att ta emot/ skicka data från sensor- och styrenheterna och skicka till PC:n. I manuellt läge ska enheten ta emot sensordata, men främst ta emot data från PC:n och vidarebefordra den till styrenheten.



Figur 2: Diagram av kommunikationenheten.

4.2 Raspberry Pi

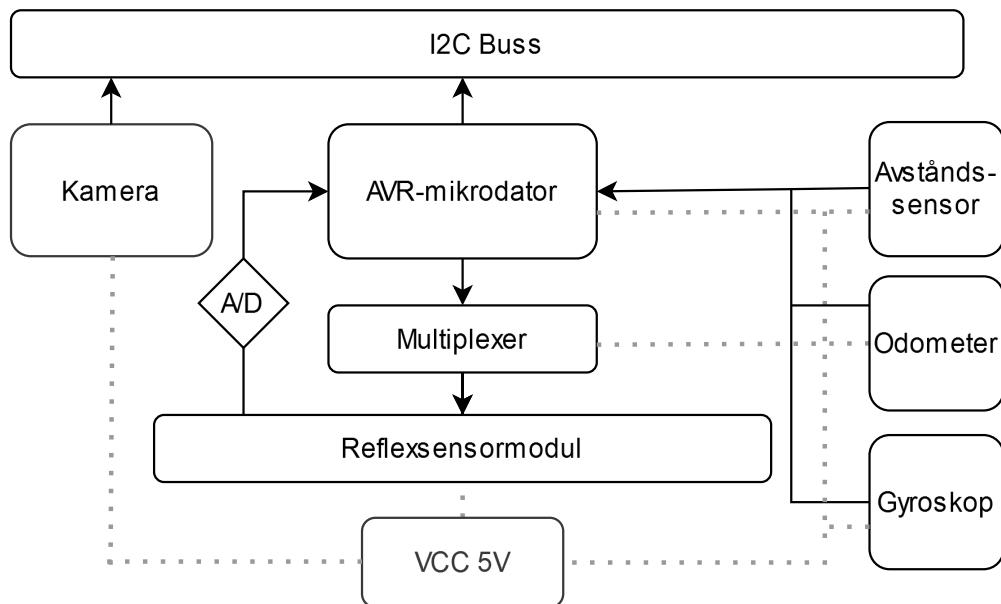
Kommunikationenheten kommer ta form av en så kallad Raspberry Pi. Denna modul är multifunktionell i avseendet att den har flera olika portar vilket gör att den är kompatibel med flera olika enheter. Den kan med hjälp av antingen bluetooth eller WiFi kopplas samman med PC för överföring av nödvändiga data. Den interna kommunikationen på robotplattformen kommer att ske seriellt via till exempel något av de tre följande språken/ kommunikationssätten: I2C, UART eller SPI.

5 SENSORENHET

I följande kapitel beskrivs vilka sensorer som kan användas och hur de kommer att implementeras på lagerroboten.

5.1 Översikt av sensorenheten

Sensorenheten, se översikt i figur 3, består av en kamera, AVR-mikrodator, multiplexer, reflexsensormodul, avståndssensorer och ett gyro. Enheten kommer att användas för att systemet ska kunna navigera lagermiljön och användaren manuellt ska kunna plocka upp varor utan att se lagerroboten. Reflexsensormodulen kommer att användas för att roboten ska kunna följa svarta linjer i lagermiljön och identifiera plockstationer samt hämtningsstationer. Vid korsningar i lagret ska roboten kunna rotera för att välja väg. För detta kommer ett gyro att användas så att roboten roterar korrekt. Olika avståndssensorer kan komma att användas för att hitta hinder. Det kommer att finnas avståndssensorer monterade både fram och bak på robotplattformen samt på dess vänstra och högra sida. Avståndssensorer i främre delen på roboten behövs för att upptäcka hinder, likaså behövs en bak till ifall roboten behöver backa. Sensorerna på sidan behövs för att mäta avståndet till varorna som ska plockas upp. För att en användare ska kunna plocka upp varor utan att se lagerroboten krävs en kamera som kommer vara installerad på roboten eller robotarmen.



Figur 3: Diagram av sensorenheten.

5.2 AVR-mikrodator

En AVR-mikrodator kommer användas för att behandla utsignalerna från alla sensorerna utom kameran. Som illustrerat i figur 3 så skickar reflexsensormodulen en A/D signal till AVR-mikrodator. Denna kommer omvandlas och därefter regleras med en enkel tyngdpunktsberäkning

$$k_T = \frac{\sum_k m_k k}{\sum_k m_k} \quad (1)$$

där k_T och m är tyngdpunkten respektive magnituden en reflektor detekterar. Syftet är att hitta tyngdpunktens avvikelse från ett jämviktsvärde för att beräkna hur mycket roboten ska svänga åt höger eller vänster. Dessutom ska AVR-mikrodatorn beräkna vilka detektorer på reflexsensormodulen som ska vara aktiverade.

5.3 Avståndssensorer

Roboten kommer att ha IR-sensorer fram och bak på roboten. Dessa har ett detekteringsintervall på antingen 4-30 cm eller 10-80 cm. Eventuellt kommer avstånd mätas genom en ultraljudssensor. Dess detekteringsintervall ligger mellan 3-300 cm.

På vänster och högersidan kommer det sitta lasersensorer eller IR för triangulering. Dessa är till för att kunna implementera en autonom upplockning av objekten med robotarmen.

5.4 Reflexsensorer

En reflexsensormodul bestående av 11 reflexsensorer kommer användas för att roboten ska följa tejplinjer på underlaget. Det kommer implementeras i kod i form av linjeföljning. Sensorerna i modulen kommer multiplexas för att spara ström och inte överhetta detektorerna. Modulens utsignal är A/D vilket skickas till och tolkas av AVR-mikrodatorn.

5.5 Kamera

En kamera kommer att monteras på roboten eller robotarmen. Den kommer att användas för möjliggöra manuell upplockning av föremålen med robotarmen. Den kopplas till en Raspberry PI vilket är mikrodatorn i kommunikationsenheten. Kameran är alltså kopplad direkt till kommunikationsenheten.

5.6 Gyro

Ett gyro kommer att användas för att mäta vridningen på roboten. Detta kommer främst utnyttjas när roboten ska svänga såsom vid rotation i korsningar.

5.7 Odometer

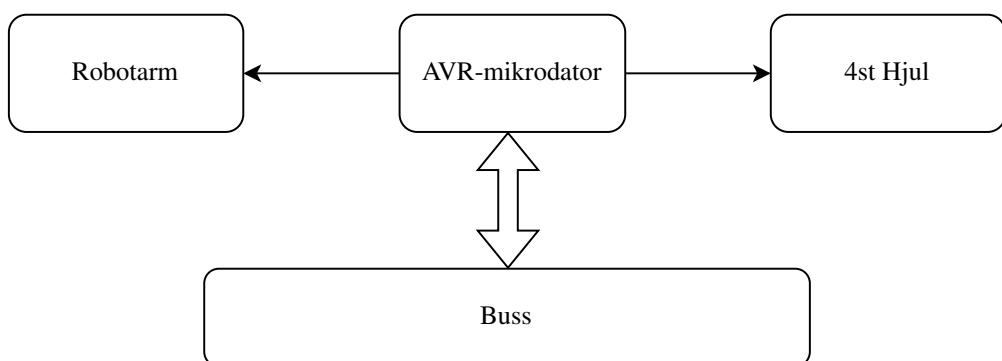
En odometer används för att mäta avståndet ett föremål har åkt. I den här kursen finns olika sätt att implementera en odometer. Det första sättet är att ha svart-vita band på hjulen och sedan mäta dessa med reflexsensorer. Andra sättet är att sätta magneter på hjulen och läsa av dessa med hjälp av en halleffektsensor. Odometern används förmodligen inte i detta projekt.

6 STYRENHET

I följande kapitel beskrivs styrenhetens funktioner, vad den ska styra och hur det ska genomföras.

6.1 Översikt av styrenheten

Styrenheten, se översikt i figur 4, kommer framför allt att användas för att ge robotens respektive motorer och servon korrekta instruktioner för de rörelser som ska utföras. Instruktionerna fås av kommunikationsenheten. I styrenheten ingår en AVR-mikrodator, fyra individuellt kontrollerbara motordrivna hjul och en robotarm med servon för hantering av lagervaror.



Figur 4: Diagram av styrenheten.

6.2 Hjul

På robotplattformen kommer det att sitta fyra hjul. Dessa ska kunna kontrolleras var för sig så att robotten kan åka framåt, bakåt, svänga åt vänster, höger samt snurra.

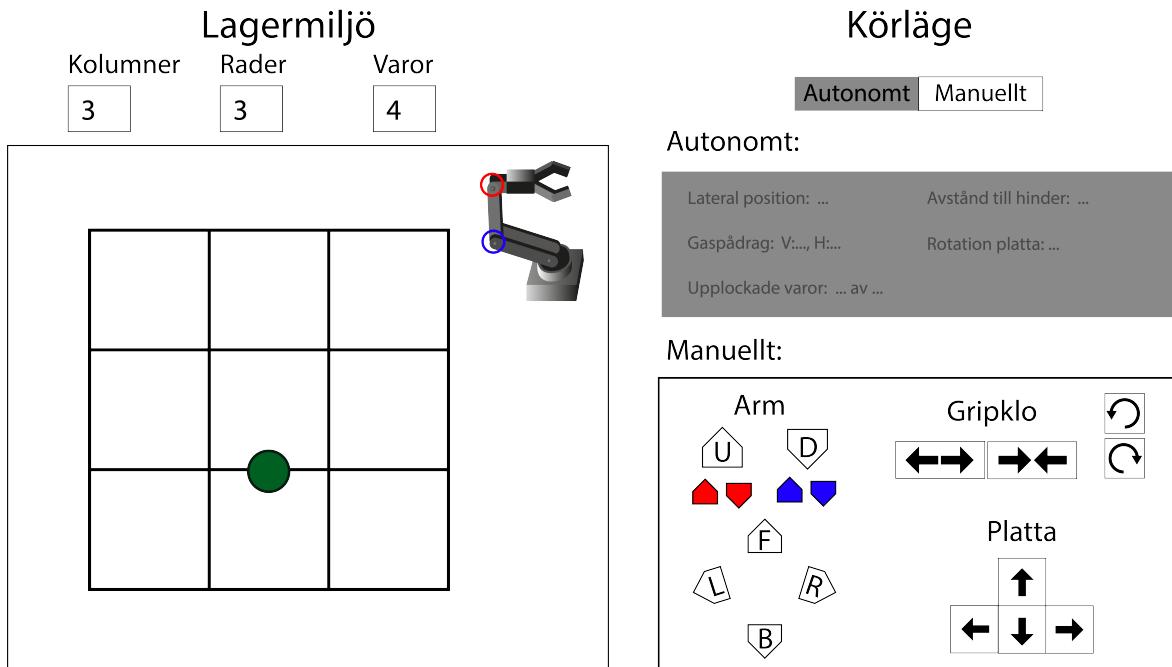
6.3 Robotarm

Robotarmen kommer att vara av typen Trossenrobotics Reactor. Servon av typen AX-12A sitter i armens ledar. Deras position bestäms med hjälp av längden på PWM-signaler. Armen sitter på en roterande platta och har en gripklo som kan rotera runt egen axel. På armen finns även plats för montering av en kamera eller annan sensor, vilket kan komma att användas för manuell eller autonom upplockning. Detta lösas genom inverterad kinematik.

6.4 Mikrodatorn

Mikrodatorn kommer att få information från kommunikationsenheten. Vid manuell köring får instruktioner om vilket håll robotten ska styras, medan vid autonom köring får ett reglerfel som mikrodatorn omsätter till en riktning via PD-reglering.

7 PERSONDATOR



Figur 5: Översikt av GUI.

PC:n fyller olika funktioner beroende på om roboten körs i autonomt eller manuellt läge. I autonomt läge tar PC:n emot data från kommunikationsmodulen som sedan plottas på lämpligt sätt i GUI:n (Graphical User Interface). Är roboten i manuellt läge kan den styras från PC:n. I GUI:n ska dimensionerna på lagermiljön samt mellan vilka korsningar plockstationer finns definieras. Det ska även gå att stänga av och sätta på autonomt läge i GUI:n. Vid körning i autonomt läge ska linjesensorernas laterala position, avstånd till eventuellt hinder, robotens rotation i grader vid rotation i korsning samt motorernas gaspådrag kunna avläsas. GUI:n kan göras i valfritt programmeringsspråk, till exempel Python eller C++. Matlab kan komma att användas för plottnings i GUI:n.

REFERENSER

[1] *Projektmodellen LIPS* (2011), Tomas Svensson och Christian Krysander, uppl. 1:1, Studentlitteratur AB, Lund, ISBN 978-91-44-07525-9

11 PROJEKTPLAN

Projektplan

Grupp 6

2025-03-09

Version 1.2



Status

Granskad	Ebba Lundberg	2025-03-09
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno773@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Beställare	1
2	Översiktlig beskrivning av projektet	1
2.1	Syfte och mål	1
2.2	Leveranser	1
2.3	Begränsningar	1
3	Organisationsplan för hela projektet	1
3.1	Villkor för samarbetet inom projektgruppen	1
3.2	Definition av arbetsinnehåll och ansvar	2
4	Dokumentplan	2
5	Utvecklingsmetodik	2
6	Utbildningsplan	2
6.1	Egen utbildning	2
7	Rapporteringsplan	2
8	Mötesplan	3
9	Resursplan	3
9.1	Personer	3
9.2	Material	3
9.3	Lokaler	3
9.4	Ekonomi	3
10	Milstolpar och beslutspunkter	4
10.1	Milstolpar	4
10.2	Beslutspunkter	4
11	Aktiviteter	5
12	Tidplan	5
13	Prioriteringar	5
14	Projektavslut	6
	Referenser	7
15	Appendix	8
A	Tidplan	8

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-02-12	Första utkast	EL, LS, IS, LF, SR, AN	EL, SR, LS, AN
1.0	2025-02-24	Första version	EL, LS, IS, LF, SR, AN	AN
1.1	2025-03-03	Första version med ändringar	EL, IS	EL
1.2	2025-03-09	Första version med uppdaterad tidsplan	EL	EL

1 BESTÄLLARE

Beställare av projektet är Mattias Krysander på ISY, LiU.

2 ÖVERSIKTlig BESKRIVNING AV PROJEKTET

Systemet ska designas så att en robot både kan styras manuellt via en dator och autonomt. Lagermiljön, som roboten ska kunna navigera, är känd för systemet och består av ett rutnät med hinder, plockstationer och en väg till utlämningsstationen. Roboten ska kunna börja vid utlämningsstationen, autonomt följa en linje till lagret med hjälp av sensorer och kunna hitta tillbaka med upplockade varor. I själva lagret ska den kunna undvika hinder och navigera till plockstationer där varorna ligger.

2.1 Syfte och mål

Projektets syfte är att konstruera ett system som konstruktionsalternativ till en lagerrobot, som med snabbhet och repeterbarhet kan utföra sitt uppdrag. Lagerroboten är en prototyp till en robot som ska tillverkas. Denna kommer prövas i en tävling för att utvärdera hur väl målen kring snabbhet och repeterbarhet är uppfyllda.

2.2 Leveranser

Leveranser och deras datum specificeras i kravspecifikationen sektion 11 [2].

2.3 Begränsningar

När lagermiljön och plockstationer är definierade i persondatorn och autonomt läge start, ska roboten klara sig själv. Undantaget är om robotarmen ska styras manuellt, vilket sker från persondatorn. Material som kan användas är endast det som ISY tillhandahåller. Projektgruppen har ett begränsat antal timmar att tillhandahålla, se avsnitt ekonomi.

3 ORGANISATIONSPLAN FÖR HELA PROJEKTET

Organisationsplanen kommer i huvudsak att följa LIPS-modellen som beskrivs i *Projektmodellen LIPS* [1]. Den följer en v-modell bestående av: krav, planering, design, implementation och test & verifiering med regelbunden återkoppling mellan stegen. Syftet är att hitta problem innan projektet har gått för långt och korrigeringar blir alltför tidskrävande.

3.1 Villkor för samarbetet inom projektgruppen

Se gruppkontrakt [3].

3.2 Definition av arbetsinnehåll och ansvar

Roll	Namn	Beskrivning
Styrehetsansvarig	Linus Funquist	Delansvarig över styrenheten
Dokumentansvarig, kommunikationsansvarig	Ebba Lundberg	Övergripande ansvar för samtliga dokument samt kommunikationsenheten
Projektledare, Sensoransvarig	Andreas Nordström	Övergripande ansvar över projektet och delansvarig över sensorenheten
Sensoransvarig	Sigge Rystedt	Delansvarig över sensorenheten
Dokumentansvarig, kommunikationsansvarig	Ida Sonesson	Övergripande ansvar för samtliga dokument samt kommunikationsenheten
Styrehetsansvarig, designansvarig	Lisa Ståhl	Delansvarig över styrenheten och systemdesign

4 DOKUMENTPLAN

Se kravspecifikationen sektion 12 [2].

5 UTVECKLINGSMETODIK

För att säkerställa ett effektivt arbetsflöde kommer arbetet att delas upp baserat på expertis. Uppdelningen grundar sig bland annat på förstudier. När en uppgift är slutförd kommer den frigjorda tiden att användas för att stödja övriga gruppmedlemmar i deras arbete.

6 UTBILDNINGSPLAN

För att kunna genomföra projektet kommer det under projektets gång att behöva genomföras utbildningar i form av föreläsningar, laborationer, seminarier samt förstudier.

6.1 Egen utbildning

Den egna utbildningen kommer främst riktas mot kunskaper inom områden såsom reglerteknik, sensor och kommunikation för att möjliggöra förstudierna.

7 RAPPORTERINGSPLAN

En veckorapport ska skickas in till beställare varje vecka. Denna inkluderar en tidsrapport och en statusrapport som besvarar följande frågor: - Vilka framsteg har gjorts sedan förra tidrapporten? - Finns det några problem? - Vad ska göras under den kommande veckan?

Veckorapporten ska skrivas av samtliga i gruppen under ett planerat möte varje måndag.

8 MÖTESPLAN

Under projektets gång kommer regelbundna möten genomföras en gång i veckan, med målet att gå igenom förra veckans arbete samt planera nästa vecka.

9 RESURSPLAN

Nedanstående punkter beskriver vilka olika typer av resurser gruppen har att tillgå.

9.1 Personer

Arbete förväntas av samtliga i gruppen utöver röda dagar, tentaperioder och planerade ledigheter. Planerad ledighet bör förvarnas om i god tid och berörd gruppmedlem bör kompensera denna frånvaro enligt överenskommelse inom gruppen.

9.2 Material

Materialet som kommer att användas tillgodoses av ISY och innefattar robotplattform, robotarm, mikrodatorer, sensorer, batterier samt motorer för reglering.

9.3 Lokaler

Projektgruppen kommer att ha tillgång till en laborationssal i Muxen samt Visionen där tester kommer att utföras.

9.4 Ekonomi

Gruppen har 230 timmar per person, totalt 1380 timmar för att lösa uppgiften. Dessa börjar räknas efter BP3.

10 MILSTOLPAR OCH BESLUTSPUNKTER

Följande underrubriker beskriver projektets milstolpar och beslutspunkter samt när dessa infaller.

10.1 Milstolpar

Tabellen nedan visar samtliga milstolpar i projektet samt datumet då de infaller.

Nr	Beskrivning	Datum
0	Designspecifikation 1.0 inlämnad	12/3
1	Förstudie inlämnad	7/4
2	Bussen klar	10/4
3	Manuell styrning	25/4
4	Manuell styrning av robotarm	2/5
5	Roboten kan följa tejpbil	6/5
6	Autonom körning	13/5
7	Fullständig GUI	19/5
8	Färdig presentation	30/5
9	Färdig rapport	6/6

10.2 Beslutspunkter

Tabellen nedan visar samtliga beslutspunkter i projektet samt datumet då de infaller.

Nr	Beskrivning	Datum
0	Projektgruppen formad och projektgrupp tilldelad	24/1
1	Kravspecifikationen v1.0 ska vara klar	6/2
2	Efter godkänd projektplan, tidplan och systemskiss hålls BP2 mötet med beställaren.	24/2
3	Designspecifikation godkänd	12/3
4	Nuvarande design godkänd av handledare	17/4
5a	Färdigställandet av baskrav verifieras	7/5
5b	Färdigställandet av slutkrav verifieras	21/5
6	Projektet avslutas	9/6

11 AKTIVITETER

Tabellen nedan visar samtliga aktiviteter i projektet, uppskattad tid och beroenden till andra aktiviteter.

Nr	Aktivitet	Beroende av aktivitet nr	Beräknad tid
1	Skriva designspecifikation		80
2	Kommunikation mellan PC och kommunikationseenheten		10
3	Konstruera en fungerande buss mellan delsystemen		90
4	Seriell överföring av data mellan PC och styrmodul	2, 3	35
5	Seriell överföring av data från sensormodul till PC	2, 3, 7	30
6	Fungerande kortaste väg algoritm med hinder		35
7	Få alla sensorer att kunna läsa data		40
8	Skriv kod för sensormodulen som gör om sensordata till läsbara storheter		60
9	Spara sensordata på PC	5	20
10	Installera sensorenhet på robotplattform		20
11	Få reflexsensorerna att registrera en tejpbit och lagerroboten kan stanna via en avbrottsrutin	4, 5, 14	80
12	Styrenheten kan skicka kommandon till stymotorerna		40
13	Skapa GUI som möjliggör testning		40
14	Få robotplattformen att röra sig genom manuell styrning från PC	4, 13, 18	60
15	Kunna styra robotarmen manuellt	4, 13	120
16	Kunna plocka upp vara med robotarmen via fjärrstyrning	15	60
17	Skapa fullständig GUI för PC	4, 9, 13	60
18	Installera styr- och kommunikationseenhet på robotplattformen		50
19	Få robotten att röra sig genom autonom styrning	14	80
20	Robotten kan åka till och från hämtningsstationen	11, 19	70
21	Skriva användarhandledning		30
22	Skriva presentation		20
23	Skriva efterstudie		15
24	Skriva kappa		20
25	Skriva teknisk dokumentation		60

12 TIDPLAN

Se Appendix A.

13 PRIORITERINGAR

Prioriteringar på projektet kommer att följa det som beskrivs i kravspecifikationen. Prioritet 1 krav kommer göras först följt av 2 och i mån av tid 3. Om nödvändigt kommer förseningar diskuteras med beställaren för omförhandling av kravspecifikationen.

14 PROJEKTAVSLUT

Projektet avslutas den 9:e juni. Då ska:

- Efterstudien vara inlämnad
- Komplett och väldokumenterad källkod vara incheckad på git
- Komplett kandidatrapport vara inlämnad som en pdf-fil med all appendix inkluderade
- All utrustning och nycklar vara återlämnade

REFERENSER

- [1] *Projektmodellen LIPS* (2011), Tomas Svensson och Christian Krysander, uppl. 1:1, Studentlitteratur AB, Lund, ISBN 978-91-44-07525-9
- [2] *Kravspecifikation* (2025), Grupp6: Kravspec_grupp6_version1.0.pdf
- [3] *Gruppkontrakt* (2025), Grupp6: Gruppkontrakt_grupp6.pdf

15 APPENDIX

A TIDPLAN

12 DESIGNSPECIFIKATION

Designspecifikation

Grupp 6

2025-03-18

Version 1.0



Status

Granskad	Andreas Nordström	2025-03-13
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno773@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Beställare	1
2	Inledning	1
3	Översikt av systemet	1
4	Sensorenhet	2
4.1	Komponenter	2
4.2	Kretsschema	2
5	Kommunikationsenhet	4
5.1	Komponenter	5
5.2	Blockschema	6
5.3	Mjukvara	6
6	Styrenhet	8
6.1	Komponenter	8
6.2	ATmega1284P	8
6.3	Körning av robotplattformen	8
6.4	Robotarmen	9
7	Persondator	11
7.1	Graphical User Interference, GUI	11
	Referenser	12

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-03-04	Första utkast	EL, AN, IS, LS, SR, LF	EL, AN, IS, LS, SR, LF
0.2	2025-03-12	Andra utkast	EL, AN, IS, LS, SR, LF	EL, AN, IS, LS, SR, LF
1.0	2025-03-13	Första version	IS, LS, SR	IS, LS, SR, LF

1 BESTÄLLARE

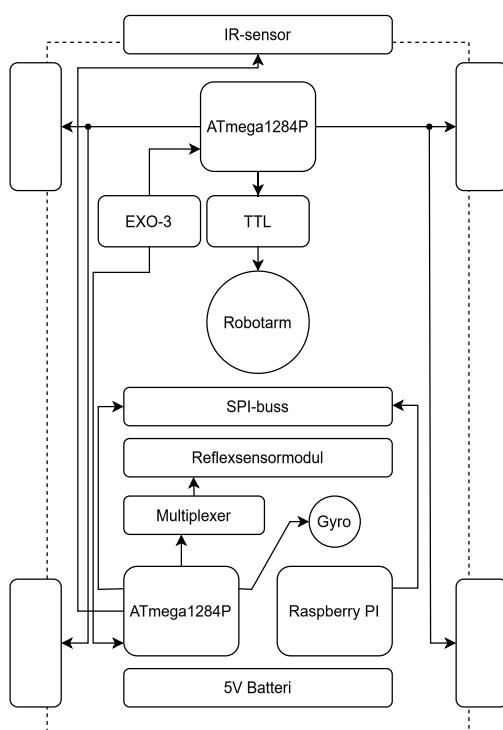
Beställare av projektet är Mattias Krysander på ISY, LiU.

2 INLEDNING

Detta dokument ligger till grund för projektets konstruktions- och implementationsfas. Här anges vilka komponenter som ska användas, kopplingsscheman för de olika enheterna samt grundläggande pseudokod.

3 ÖVERSIKT AV SYSTEMET

Systemet består av en sensor-, kommunikations- och styrenhet. Dessa kommer vara installerade på roboten som illustreras i figur 1. Följande kapitel beskriver respektive delsystem i detalj.



Figur 1: Blockschema över systemet.

4 SENSORENHET

att sensorerna är korrekt anslutna till ATmega1284P-mikrodatorn på hårdvarunivå och att de är korrekt initierade i mjukvaran. Sensorenhetens uppgift är att samla in data som sedan skickas vidare till kommunikationsenheten. Sensor-datan kommer skickas kontinuerligt från samtliga sensorer vilket utgör grunden till de styrbeslut som tas samt kartläggningen av lagermiljön.

4.1 Komponenter

I tabell 1 listas de komponenter som ingår i sensorenheten.

Tabell 1: Lista över komponenter.

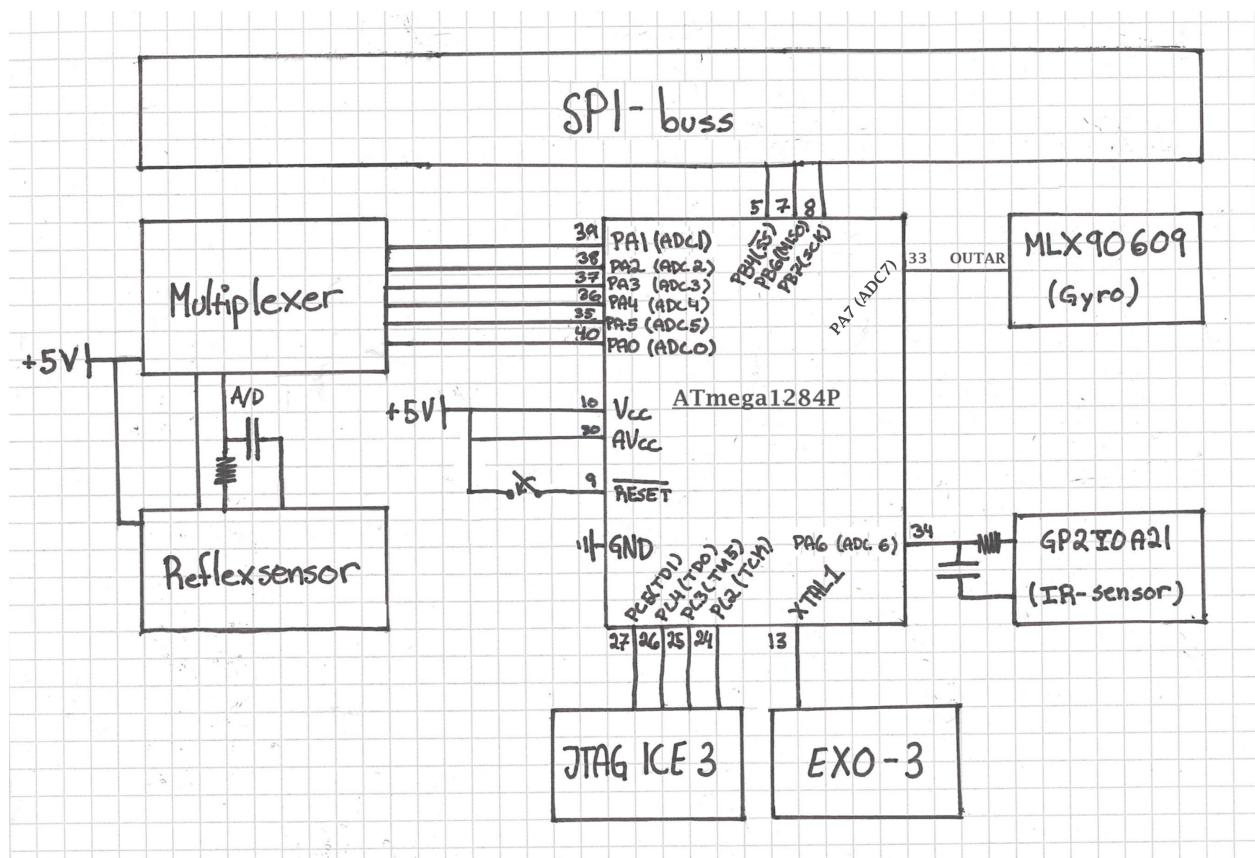
Komponent	Egenskap	Antal
Mikrodator	ATmega1284P	1
Emulator	JTAG ICE 3	1
Oscillator	EXO-3	1
Avståndssensor	GP2Y0A21, 10-80cm	1
Reflexsensormodul		1
Multiplexer	För reflexsensormodulen	1
Gyroskop	MLX90609	1
Resistor	18 kΩ, lågpassfilter	2
Kondensator	100 nF, lågpassfilter	2

4.2 Kretsschema

I detta kapitel presenteras kretsscheman för de olika delarna av sensorenheten. Notera att MLX90609 gyroskopet hade kunnat kopplas direkt till SPI-bussen men för att förenkla programmeringen har den istället kopplats till mikrodatorn via ADC7 porten.

4.2.1 ATmega1284P

Figur 2 visar en detaljerad uppkoppling för mikrodatorn ATmega1284P där övriga delsystem är skissade. Multiplexern styrs via port 34-39 och A/D signalen från reflexsensormodulen lågpassfiltreras innan den tas i emot i port 40 via adaptern. En JTAG ICE 3 är ikopplad via port 24-27 möjliggör programmering och simulering av mikrodatorn. Ytterligare en EXO-3 är ikopplad i port 13 för att styra mikrodatorns interna klocka. IR-sensorn GP2Y0A21:s A/D utsignal är likt reflexsensorn lågpassfiltrerad och kopplad till port 39. I port 9 är en RESET inkopplad. MLX90609 gyrot ikopplad till mikrodatorn via ADC7.



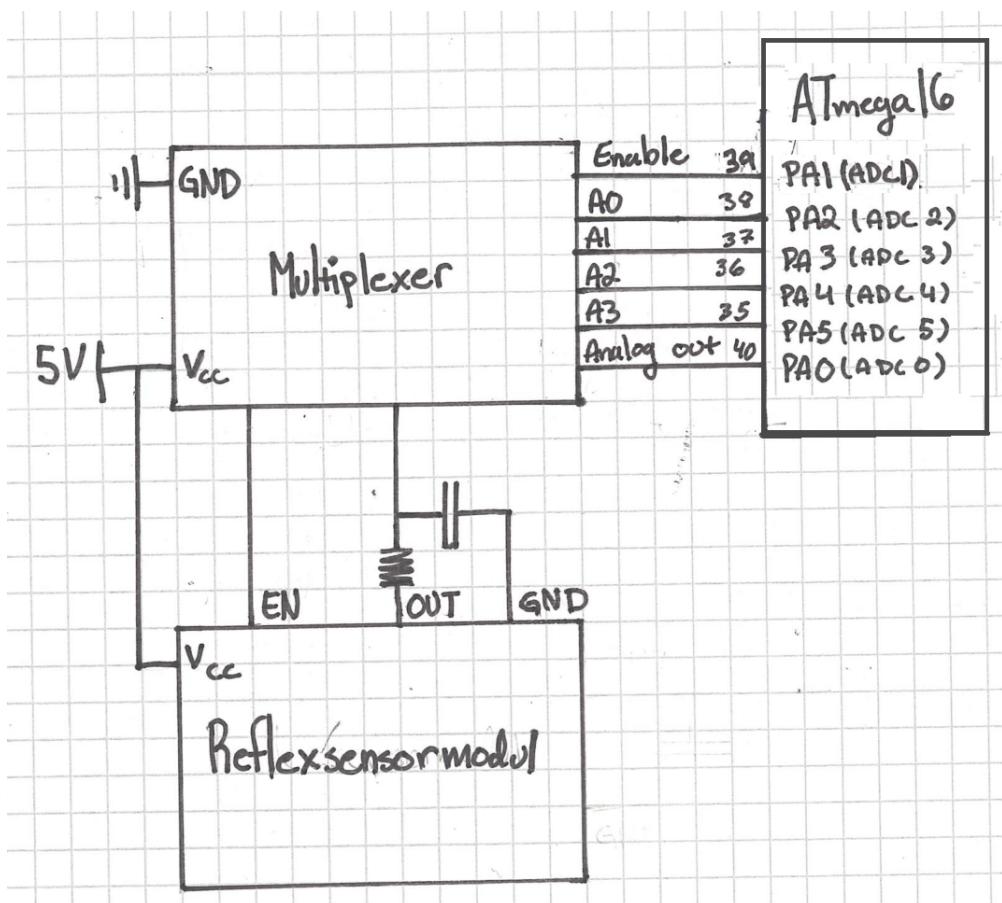
Figur 2: Kretsschema för ATmega1284P.

4.2.2 Reflexsensormodul

Som illustreras i figur 3 styrs och avläses reflexsensormodulen med mikrodatorn genom en adapter som har en inbyggd multiplexer. A0-3 är kopplad till A-porten och väljer vilken reflexsensor som Enable kan aktivera. Analog out ger den lågpassfiltrerade A/D-signalen från reflexsensormodulens OUT. För att kunna avgöra om roboten ska svänga höger eller vänster genomförs här en tyngdpunktsberäkning för att ta fram data på avvikelsen från mitten:

$$k_T = \frac{\sum_k m_k k}{\sum_k m_k} \quad (1)$$

där k_T och m är tyngdpunkten respektive magnituden som en reflektor detekterar. Denna data skickas till kommunikationsenheten där själva regleringen utförs.



Figur 3: Kretsschema för reflexsensormodulen och multiplexern.

4.2.3 Gyro

Gyrots utsignal är en analog vinkelhastighet som är kopplad till AVR-mikrodatorns ADC-omvandlare. Då styrenheten arbetar med vinklar måste vinkelhastigheten integreras över tid för att omvandla denna till en vinkel. Vid integration är tidssteget viktigt, det kan därför vara bra att sampla med en bestämd frekvens med hjälp av hårdvarutimers.

5 KOMMUNIKATIONSENHET

Kommunikationenhetens sköter all intern och extern kommunikation. Den ska ta emot data från styr- och sensorenheten och överför sedan information till PC:n för att kartlägga var roboten befinner sig. Kommunikationenhetens ska även kunna ta emot data via bluetooth från PC:n avsedd för styrenheten för manuell styrning. Vid autonom körning ska kommunikationenheten, med hjälp av data från resterande moduler, kunna ta beslut om kortaste väg.

5.1 Komponenter

Nedan följer en lista på de komponenter som ingår i kommunikationsenheten:

- Raspberry Pi
- Nivåskiftare

5.1.1 SPI-buss

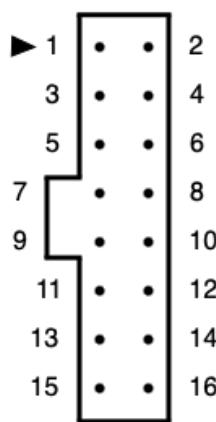
SPI är det bussprotokoll som kommer att användas för kommunikation mellan enheterna. Den kommer att kopplas via nivåskiftaren från Raspberry Pi:n och sedan vidare ut till styr- och sensorenheten och deras respektive AVR:er.

5.1.2 Raspberry Pi

Kommunikationsenhetens mikrodator kommer att vara en Raspberry Pi som kopplas samman med styr- och sensorenheten samt PC:n.

5.1.3 Nivåskiftare

Mellan Raspberry Pi:n och SPI-bussen behövs det kopplas en nivåskiftare, se figur 4, eftersom att dessa komponenter använder sig av olika spänning-logik. Här är det viktigt att sätta nivåskiftaren så att den täcker Raspberry Pi:n helt. Koppling för respektive pin visas i tabell 2.



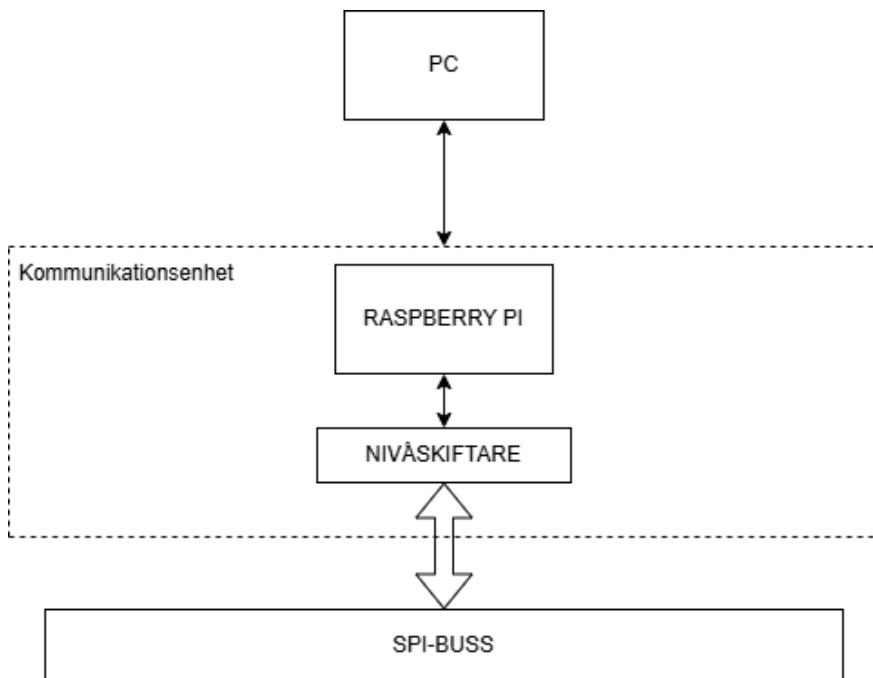
Figur 4: Nivåskiftare med 16 pinnar.

Tabell 2: Pin med respektive koppling för nivåskiftare.

Pin	Koppling	Pin	Koppling	Tecken	Förklaring
1	Gnd	2	Gnd	→	Raspberry → AVR
3	N/C	4	N/C	←	Raspberry ← AVR
5	I2C_SCL ↔	6	SPI_MOSI →	↔	Raspberry ↔ AVR
7	I2C_SDA ↔	8	SPI_MISO ←		
9	N/C	10	SPI_SCLK →		
11	N/C	12	SPI_CE0 →		
13	GPIO27 ←	14	SPI_CE1 →		
15	GPIO17 ←	16	GPIO22 →		

5.2 Blockschema

Nedan följer ett blockschema för kommunikationsenheten, se figur 5, med komponenterna beskrivna i avsnitt 5.1.

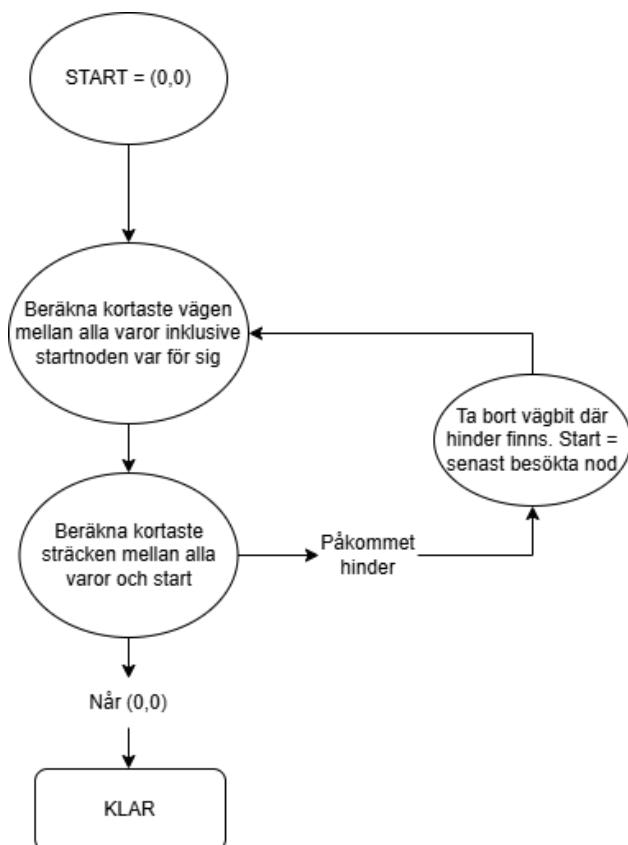
**Figur 5:** Blockschema för kommunikationseinhet.

5.3 Mjukvara

Nedan diskuteras mjukvaran som finns i kommunikationseinheten.

5.3.1 Pseudokod snabbaste vägen

Koden för att hämta alla varor i en körning kommer följa pseudokoden enligt 6. Varje korsning, hörn och vara anges som en nod och mellan varje nodpar finns en kant där alla kanter i detta fall har samma vikt (kostnad). Roboten startar alltid i ett av rutnätets hörn och anger den noden som (0,0), därifrån beräknas snabbaste vägen mellan alla två varor samt mellan alla varor och starten. När detta beräknats jämförs alla vägar där alla varor och start ingår och den kortaste vägen väljs. Om hinder upptäcks när körningen pågår förbjuds kanten där hindret ligger sedan sätts den senast besökta noden till startnod och algoritmen körs från början. När roboten sedan når nod (0,0) med alla varor hämtade är algoritmen klar.



Figur 6: pseudokod snabbaste vägen.

5.3.2 Regleralgoritm för tejpföljning

Kommunikationsenheten kommer även att hantera regleringen av roboten. Sensordata tas emot av sensorenheten och styrbeslut skickas sedan till styrenheten. I autonomt läge kommer regleralgoritmen, med hjälp av data från en tejpsensor under roboten att följa tejpbitten rakt tills det kommer ett nytt styrbeslut.

För autonom tejpföljning kommer PD-reglering att utnyttjas, se ekvation 2. K_P och K_D är konstanter som kommer

bestämmas experimentellt. Vid tejpföljning är målet att tejpen ska vara i mitten av tejpsensorn, och felet ($e(t)$) blir något mått på avvikelsen från mitten.

$$u(t) = K_P e(t) + K_D \frac{d}{dt} e(t) \quad (2)$$

Kommunikationsenheten beräknar de sluttgiltiga signalerna som ska skickas till motorer eller servon, och skickar dem till styrenheten som skickar vidare dem dit de ska.

6 STYRENHET

Styrenhetens uppgift är att styra motorerna och robotarmen utifrån styrdata från kommunikationsenheten.

6.1 Komponenter

I tabell 3 nedan listas komponenterna som ingår i styrenheten.

Tabell 3: Lista över komponenter.

Komponent	Egenskap	Antal
Mikrodator	ATmega1284P	1
Robotarm	PhantomX Reactor	1
Hjul		4
Oscillator	EXO3	1
Resistor	10 kΩ	1

6.2 ATmega1284P

Kärnan av styrenheten är mikrodatorn, som är en Atmel ATmega1284P. Den kommer att vara kopplad till kommunikationsenheten via en SPI-buss, där kommunikationsenheten agerar master.

6.3 Körning av robotplattformen

Robotplattformens fyra hjul går att kontrollera som två hjulpar, vänster respektive höger, och styrs via en DIR och en PWM-signal från mikrodatorn (se kopplingsschema). DIR-signalen bestämmer vilket håll hjulen ska snurra åt, och PWM-signalerna bestämmer hur snabbt de ska snurra. PWM är en periodisk signal som antingen är hög eller låg, den andelen av periodtiden som signalen är hög brukar kallas signalens duty-cykel, och en hög duty-cykel ger snabbare snurr på hjulen. Då robotens olika sidor styrs separat från varandra går det att få den att svänga genom att lägga olika mycket kraft i respektive sida. Styrsignaler skickas till styrenheten från kommunikationsenheten, antingen resultatet av PD-regleringsalgoritmen som tar in data från sensorenheten, eller input från användaren.

6.4 Robotarmen

För att kontrollera robotarmens servon används UART som finns på pins på mikrodatorn. Däremot måste signalerna gå genom en TTL-krets, se figur 8 som ser till att det bara går signaler åt ett håll i taget, då servona kräver att UART-signalerna konverteras till halvduplex. Kommunikation med alla servon sker via samma kontakt, och i datan som skickas adresseras det relevanta servot. I servona finns det mycket data som går att läsa av och många parametrar som går att bestämma. Bland annat går målvinkel och vinkelhastighet att bestämma för respektive servo så att armen kan ställas i rätt position, och så att alla servon är färdiga med sin vridning samtidigt så att armen som helhet rör sig i en jämn rörelse.

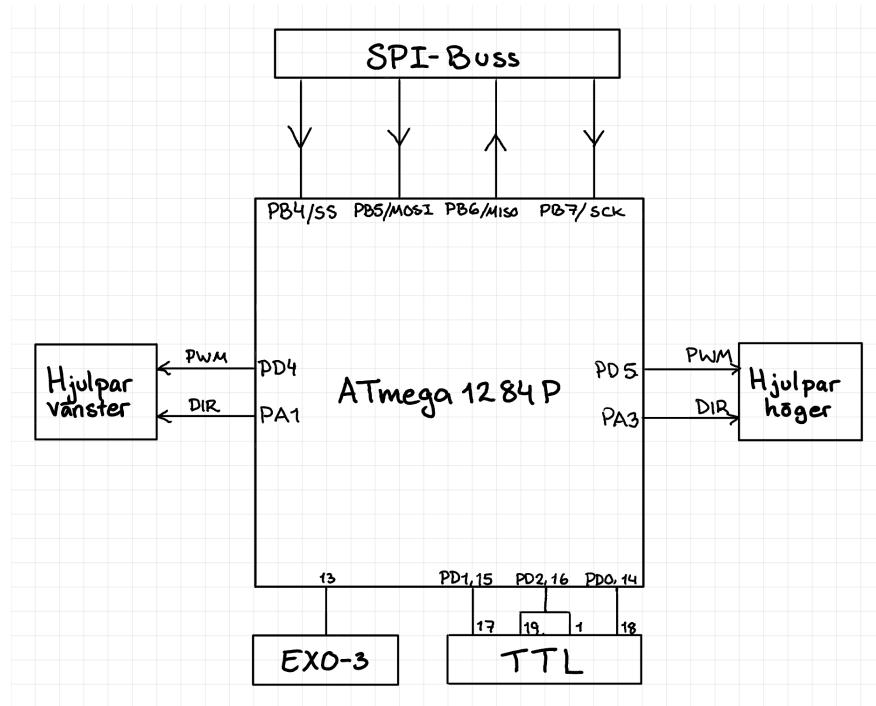
6.4.1 Autonom styrning av robotarmen

För att styra robotarmen autonomt kommer framförallt en sak att behöva tas reda på: var är objektet som ska plockas upp relativt robotarmens bas. Om detta är känt så kan vinklarna som robotarmens servon ska ställas in på beräknas med hjälp av invers kinematik. Detta går antingen att göra analytiskt, eller numeriskt. Dessa beräkningar kommer troligen göras i kommunikationsenheten. Dock, givet att robotten har placerat sig rätt, så kommer objektet alltid att vara placerat på samma plats relativt robotten och det skulle alltså gå att hårdkoda själva plockrörelsen. Om rörelsen hårdkodas så behövs inte invers kinematik eller ytterligare beräkningar.

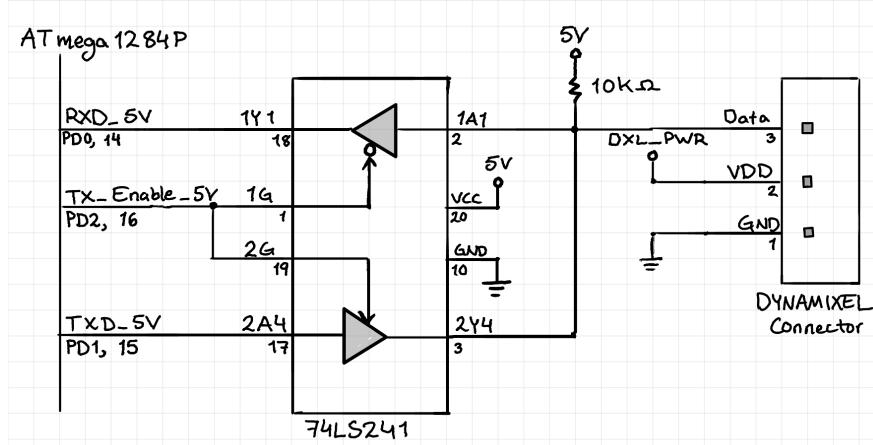
6.4.2 Manuell styrning av robotarmen

Manuell styrning av robotarmen skulle kunna ske på två sätt. Antingen styrs ett servo i taget, eller så matar användaren in armens generella riktning, och så får det beräknas hur servona ska vridas. Att styra ett servo i taget implementeras enkelt, men det andra är inte lika lätt. Ett sätt att lösa det på skulle kunna vara att definiera ett minsta avstånd armen kan röras, och medan användaren exempelvis styr armen uppå flyttas armens målposition upp med bestämd frekvens. Samtidigt beräknas också servonas nya vinklar med inverskinematik och skickas kontinuerligt till robotarmen, även detta med bestämd frekvens. Detta beräknas nog också i kommunikationsenheten.

Nedan följer blockschema för styrenheten, och ett mer detaljerat kopplingsschema för kretsen mellan AVR och robotarm.



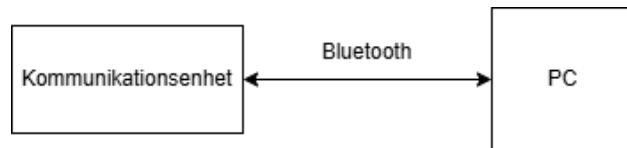
Figur 7: Blockschema över styrenheten



Figur 8: Kopplingsschema för TTL kretsen mellan AVR och robotarm

7 PERSONDATOR

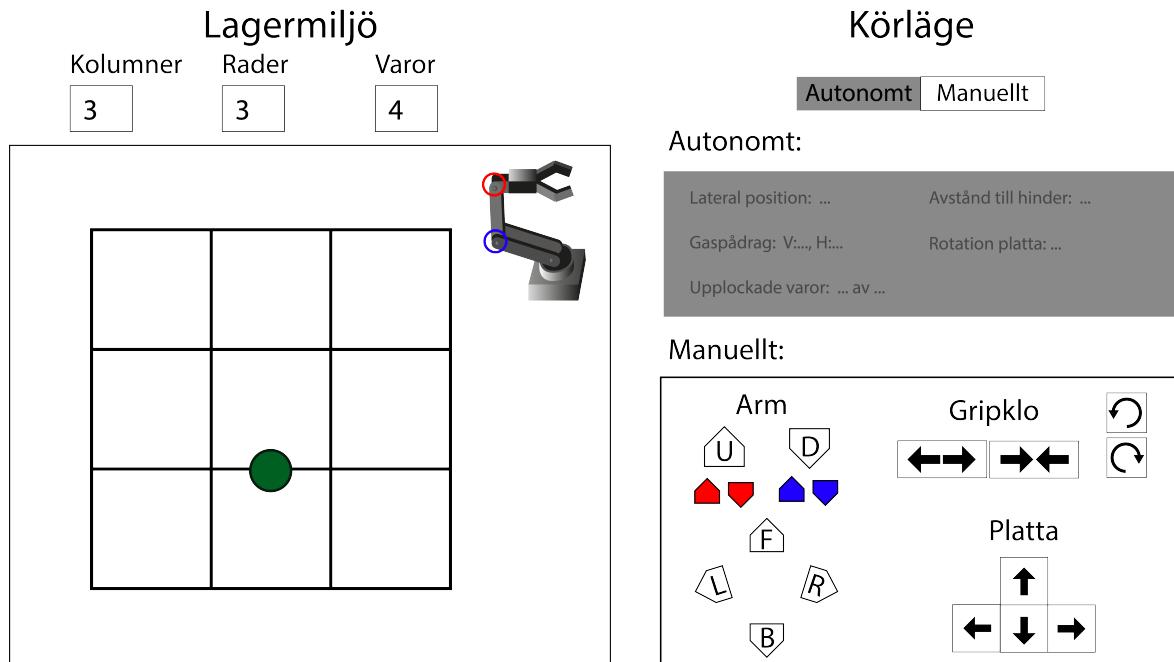
Kommunikationen mellan persondatorn och roboten kommer att ske via bluetooth-länk, se figur 9. Via persondatorn ska det, med hjälp av denna kommunikation, kunna genomföras manuell styrning via GUI:n, se figur 10. Oavsett läge ska även persondatorn kunna ta emot data från roboten för att ge en uppfattning om kartläggning och andra uppgifter såsom exempelvis stymotorernas gaspådrag.



Figur 9: Blockschema över persondatorn.

7.1 Graphical User Interference, GUI

GUI:n kommer att programmeras i ett språk såsom antingen Python, C eller C++. För att förenkla processen kommer ramverket Qt att användas. Qt är kompatibelt med flera plattformar och kommer att göra det enklare att programmera GUI:n då den minskar antalet kodrader som behövs, och är främst framtaget för utveckling av GUI.



Figur 10: Översikt över GUI.

REFERENSER

- [1] *Multiplexer till sensorskenan* (2025), Melexis. Hämtad: 2025-03-04, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/mlx90609.pdf>.

13 FÖRSTUDIER

13.1 Kommunikationsenhetens förstudie

Förstudie om kommunikationsenheten i en lagerrobot

Ebba Lundberg
Ida Sonesson

28 april 2025

Version 1.1



Status

Granskad	Ida Sonesson, Ebba Lundberg	2025-04-28
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Inledning	1
1.1	Syfte	1
1.2	Avgränsningar	1
2	Problemformulering	1
3	Kunskapsbas	2
3.1	Raspberry Pi	2
3.2	Trådlös kommunikation	2
3.3	Seriell kommunikation	3
4	Diskussion	9
4.1	Trådlös kommunikation	9
4.2	Seriell kommunikation	9
5	Slutsatser	10

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-02-24	Första utkast	Ida Sonesson, Ebba Lundberg	Ida Sonesson, Ebba Lundberg
1.0	2025-04-14	Första version	Ida Sonesson, Ebba Lundberg	Ida Sonesson, Ebba Lundberg
1.1	2025-04-28	Första version med ändringar	Ida Sonesson, Ebba Lundberg	Ida Sonesson, Ebba Lundberg

1 INLEDNING

Detta dokument är en förstudie i kursen TSEA56, Elektronik kandidatprojekt, och kommer att handla om kommunikationsenheten i den lagerrobot som ska tas fram. Kommunikationsenheten syftar till den del av roboten som sköter kommunikationen mellan de olika delsystemen som ingår i roboten. De delsystem som roboten består av är kommunikationsenheten, sensorenheten, styrenheten samt en persondator, och det är endast kommunikationsenheten som kommer att behandlas. I denna förstudie kommer olika kommunikationsmetoder mellan delsystemen att studeras för att få en djupare förståelse kring hur dessa fungerar och för att få kunskap om vilka kommunikationsmetoder som är bäst lämpade för roboten.

1.1 Syfte

Syftet med denna förstudie är att få en djupare förståelse för olika kommunikationsmetoder mellan de olika delsystemen som roboten består av, och med denna kunskap senare resonera kring vilka kommunikationsmetoder som passar roboten bäst.

1.2 Avgränsningar

I förstudien diskuteras endast de kommunikationsmetoder som finns tillgängliga i kursen TSEA56, dessa är Wi-Fi och Bluetooth som är inbyggda i en Raspberry Pi samt, I2C, SPI och UART.

2 PROBLEMFORMULERING

Det finns olika sätt att kommunicera mellan olika processorer, och följande frågeställningar kommer att besvaras för att få en större förståelse för vardera:

- Vilka olika principer finns det för kommunikation mellan olika processorer och hur fungerar dessa?
- Vad har de olika principerna för fördelar och/ eller nackdelar och hur fungerar dessa i lagerroboten?

3 KUNSKAPSBAS

Information och fakta hämtas främst från vetenskapliga artiklar. Vanheden kommer att användas som en faktabas för vidare informationssökning. Nedan följer fakta kring olika ämnen relaterade till kommunikation för lagerroboten.

3.1 Raspberry Pi

Raspberry Pi är en multifunktionell interaktiv dator byggd på ett enda kretskort. I denna kurs kommer en serie kallad Raspberry Pi 3B+ att vara tillgänglig. Denna dator har bland annat flera olika I/O-portar som exempelvis HDMI, ethernet och USB. Till lagerroboten kommer denna kommunikationsenhet kopplas till en trådlös enhet, i detta fall PC, med hjälp av antingen Bluetooth eller Wi-Fi. Detta kommer möjliggöra autonom körning och överföring av data [1].

Det finns också GPIO-portar som gör det möjligt för Raspberry Pi:n att kopplas samman med en elektrisk krets, vilket möjliggör flera olika funktioner [2]. Raspberry Pi:n kommer monteras på robotplattformen och ha tråddad kommunikation med sensor- och styrenheterna. Raspberry Pi erbjuder kommunikation via flertalet principer, till exempel via I2C, SPI och UART, och det är dessa som kommer att diskuteras vidare.

3.2 Trådlös kommunikation

Dataöverföringen mellan Raspberry Pi och PC:n kommer att ske trådlöst. Raspberry Pi, som nämnts ovan, kan kommunicera över både Wi-Fi och Bluetooth och nedan följer fakta samt fördelar och nackdelar för vardera.

3.2.1 Wi-Fi

Wi-Fi är en trådlös nätverksteknik som används globalt. Den gör det möjligt för användare att koppla upp olika enheter på internet. Wi-Fi är ett varumärke som använder flera IEEE 802.11-standarder (IEEE Std 802.11) [3].

Wi-Fi har sitt ursprung från sent 90-tal då trådlös uppkoppling blev ett ”revolutionärt koncept” och har sedan dess haft en drastisk utveckling. Teknologin har utvecklats under flera år och genomgått flera ändringar, exempelvis från 802.11b till 802.11ax (Wi-Fi 6). För varje modell har den nyare innehållt en förbättring inom flertalet egenskaper, såsom kapacitet, pålitlighet m.fl. Det enda som förblivit detsamma är namnet Wi-Fi som med sin enkelhet i både uttal och stavning blivit ett effektivt varumärke som idag speglar den fria uppkopplingen [4].

För att ett trådlöst nätverk ska kunna koppla upp sig mot internet används radiovågor, detta gäller även för tekniken bakom Wi-Fi. Syftet med denna trådlösa teknik är att direkt via ett mobilt bredband eller datornätverk, utan fast uppkoppling, kunna ansluta till internet. En så kallad router används för att det trådlösa datornätverket ska kunna kommunicera med internet [5].

Den trådlösa enheten, exempelvis en dator, kommunicerar som nämnt tidigare genom att sända radiovågor. Styrkan på radiovågorna är ungefär densamma som på mobiltelefoner och enligt standard på maxstyrkan så får ett datornätverk som max alstra 0,1 watt för radiovågor med frekvensen 2,45 GHz och 0,2 watt för frekvensen 5 GHz. Desto större avstånd från sändaren desto mindre styrka på radiovågen [5].

Då Wi-Fi uppfyller många fördelar för uppkoppling och delning av data så finns det risker med dess användning.

Beroende på dess säkerhet finns det risk att exempelvis angripare eller hackare kopplar upp sig och utnyttjar detta för att utföra illegala saker på internet eller även orsaka att ägarens personliga information blir offentlig. Data kan därmed bli tillgänglig för offentligheten om nätverket ej hanteras med säkerhet genom att exempelvis införa ett lösenord [6].

3.2.2 Bluetooth

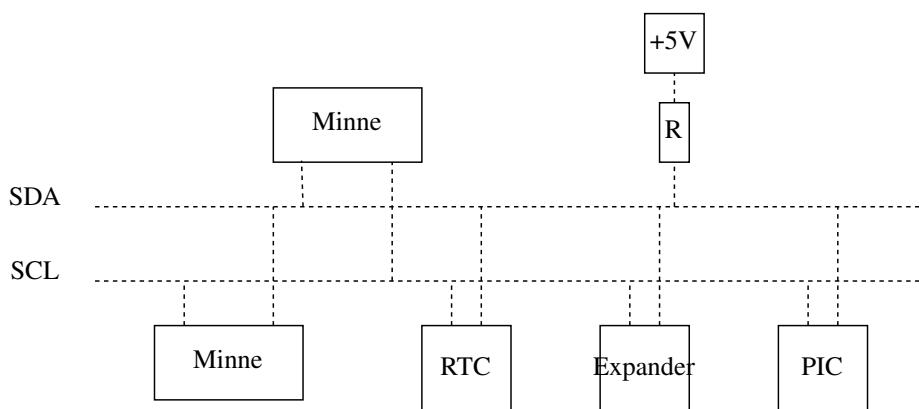
Bluetooth, eller Blåtand, utvecklades i Sverige i slutet av 90-talet och har kommit att bli standard för trådlös kommunikation på kort avstånd. Bluetooth har en relativt kort räckvidd på cirka 10 meter, och eftersom det är radiokommunikation som används så fungerar Bluetooth även om det finns hinder mellan mottagare och sändare av signalen. Här skiljer sig Bluetooth mot kommunikation som inte använder sig av radiovågor, som till exempel infraröda signaler, eftersom signalen där inte kommer fram om den inte har en fri väg mellan sändare och mottagare [7]. Bluetooth använder sig av 2.4 GHz [8].

3.3 Seriell kommunikation

En Raspberry Pi använder sig av 3.3V-logik, medan samtliga virkort i projektet endast har 5V TTL-logik. För att möjliggöra kommunikation mellan Raspberry Pi:n och AVR mikrokontrollerna (Alf och Vegard's RISC) krävs därför något för att konvertera nivån. Ett första alternativ, men också det vanligaste, är användning av UART via USB för att lösa spänningsnivån. Ett annat är att med en inkopplad nivåkonverterande krets använda I2C eller SPI via GPIO. Nedan följer fakta kring respektive alternativ [9].

3.3.1 Inter Integrated Circuit (I2C)

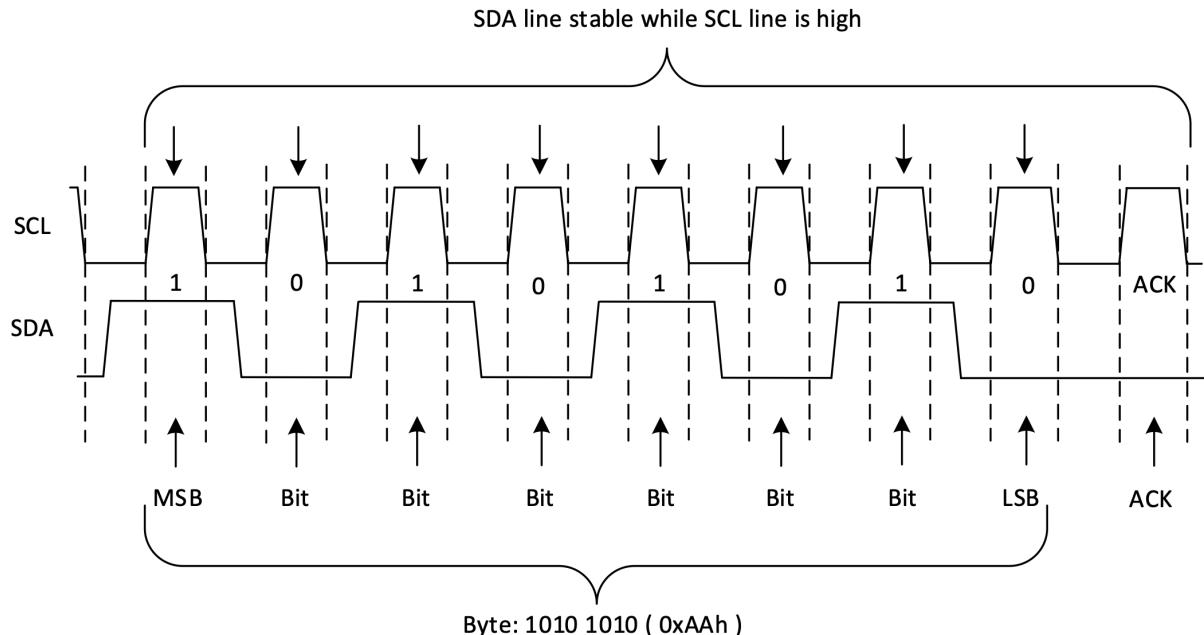
Inter Integrated Circuit, I2C, är halv-duplex och ett sätt för kommunikation via seriell buss mellan IC-kretsar. Halvduplex innebär att kommunikation kan ske i båda riktningarna men bara en riktning i taget. En I2C-buss består enbart av två ledningar, SDA (Serial DAta) och SCL (Serial CLock), till vilka man kan ansluta sina I2C-kretsar vid godtyckligt ställe på bussen, se Figur 1. Till SDA-ledningen krävs även ett så kallat pull-up-motstånd med ett standardvärdet på ungefär $4\text{-}10\text{ k}\Omega$. Det som avgör storleken är antalet I2C-kretsar på bussen samt vilken datatakt som används. Datatakten anges av SCL-signalen och det är den så kallade masterkretsen på bussen som avgör datatakten. Till skillnad från en klocksignal så kan SCL-signalen variera i frekvens och ändrar tillstånd endast då data skickas på ledningen [10].



Figur 1: Översikt av I2C-buss.

På SDA-ledningen skickas endast data mellan två kretsar åt gången. Data skickas från sändaren och skickas till mottagaren. Daten från sändaren skickas alltid med 8 bitar varav riktningen skiftas därefter på SDA-ledningen och mottagaren bekräftar dataåtkomsten med att skicka en ACK-bit (ACKnowledge). Då sändaren tagit emot ACK-biten kan ytterligare 8 bitar data skickas från sändaren. För varje databit som överförs på SDA-ledningen behövs en motsvarande klockpuls på SCL-ledningen. Detta medför att en hel byte kräver totalt 9 klockpulser – åtta för databitarna och en för ACK-signalen [10].

Kommunikation mellan master och slave fungerar på så sätt att mastern först skickar ett START-villkor, och avslutas med att mastern skickar ett STOPP-villkor [11]. Ett START-villkor karakteriseras av en hög till låg förändring på SDA-ledningen samtidigt som SCL-ledningen är hög. Ett STOPP-villkor karakteriseras tvärtemot av en låg till hög förändring på SDA-ledningen samtidigt som SCL-ledningen är hög. Endast en bit skickas per klockpuls av SCL. En byte, alltså 8 bitar, på SDA-ledningen kan antingen vara, till exempel, adressen till en viss enhet, en adress till ett register eller data från en slav som ska läsas. Data skickas med mest signifikant bit (MSB) först, och vilket antal bytes som helst kan skickas från master till slav mellan START- och STOP-villkor [11]. I Figur 2 visas hur dataöverföringen kan se ut.



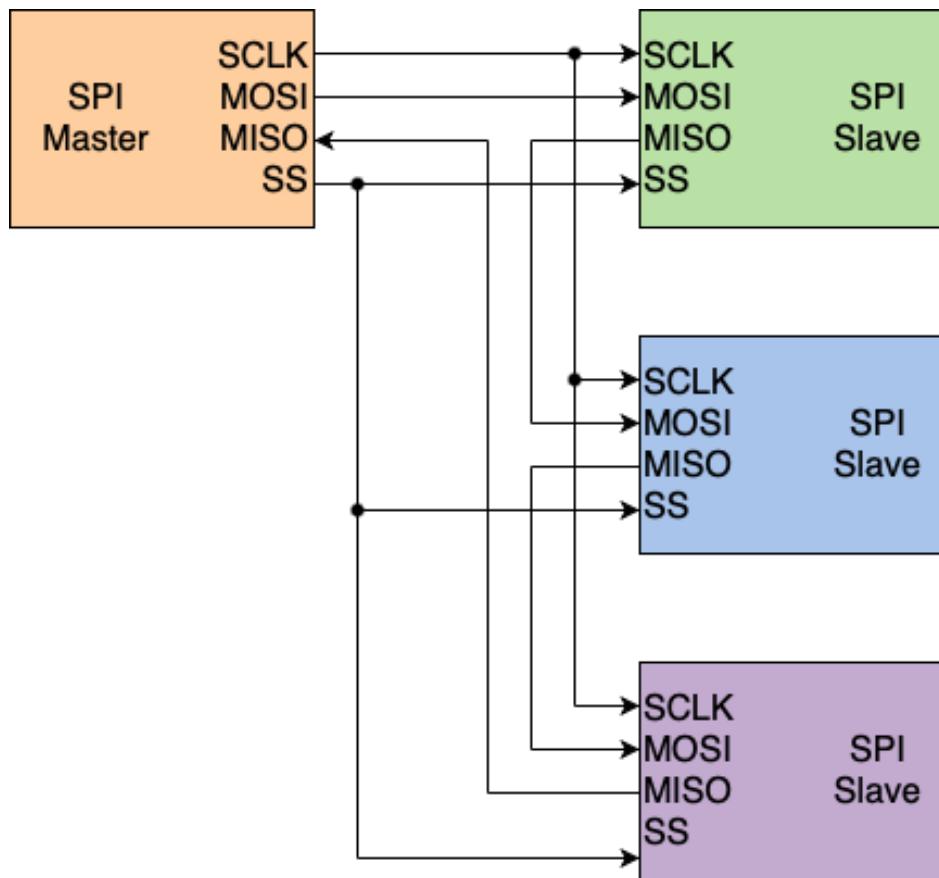
Figur 2: Dataöverföring I2C [11].

I2C-protokollet gör det möjligt att lätt ansluta flera externa enheter som exempelvis sensorer, motorstyrningar och displayer. Flera enheter kan anslutas eftersom I2C är en adresserbar buss, vilket gör det möjligt att skicka data till vald enhet med hjälp av adressfältet. I2C-protokollet erbjuder stor flexibilitet då den endast kräver ett fåtal kablar [12]. Övriga fördelar med en I2C-buss är bland annat dess förmåga att bibehålla ett lågt antal signaler oavsett antalet enheter på bussen. Den inkluderar även en funktionalitet kallad ACK/NACK som förbättrar felhanteringen [13]. Med I2C-protokollet följer dock även nackdelar såsom ökad komplexitet i ”firmware” eller lågnivåhårdvara.

3.3.2 Serial Peripheral Interface (SPI)

Serial Peripheral Interface, SPI, är ett slags kommunikationsprotokoll där full-duplex dataöverföring är möjlig. Full-duplex innebär att kommunikation kan ske i båda riktningar samtidigt. SPI används för kommunikation mellan en enchipssdator eller AVR och sensorer eller liknande enheter, och bygger på ett system av masters och slaves. En master är typiskt en enchipssdator som kontrollerar hela kommunikationsprocessen och datautbytet. Slave-delen av en SPI består av en eller flera enheter som reagerar på masterns kommandon. SPI är ett effektivt kommunikationssätt med hög hastighet som även är relativt lätt att använda eftersom det inte behöver användas någon adressering [14]. Det finns fyra huvudsakliga trådar av masters och slaves, dessa syns i Figur 3 och är:

- MOSI, Master Out Slave In: överför data från master till slave.
- MISO, Master In Slave Out: överför data från slave till master.
- SCLK, Serial Clock: en konfiguerbar klocksignal som synkroniseras med dataöverföringen, styrs av mastern.
- SS, Slave Select: väljer vilken slave enhet som ska vara aktiv.



Figur 3: Seriell koppling av SPI.

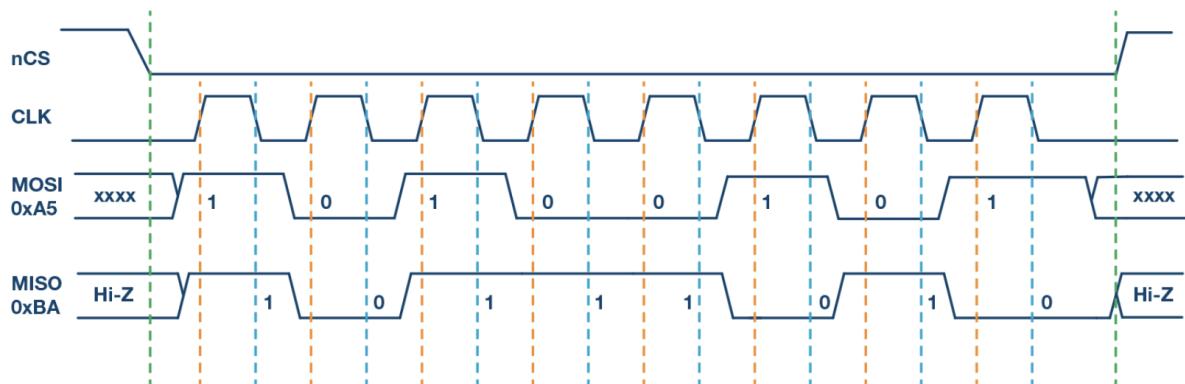
För att skicka data mellan master och slave så måste mastern först skicka en klocksignal och genom Slave select väljs vilken slave som ska vara aktiv. Slave select ska vara låg för att kunna välja slave. Eftersom SPI är full-duplex så kan både master och slave skicka data samtidigt via MOSI och MISO [15].

SPI-datan består av en CPOL-bit och en CPHA-bit som definierar olika lägen eller ”modes”, vilka mastern som sagt väljer [15]. CPOL-biten bestämmer klockpolariteten under det inaktiva läget, som är definierat av att Slave select går från hög till låg i början av en dataöverföring och av att Slave select går från låg till hög i slutet av en dataöverföring. CPHA-biten bestämmer klockfasen. Beroende på värde så bestäms om det är stigande eller fallande flank som ska till exempel sampla eller skifta data. Beroende på värdena på CPOL- och CPHA-bitarna så finns det totalt fyra olika lägen tillgängliga [15], se Tabell 1.

Tabell 1: Tabell över modes.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

I mode 0 är både klockpolariteten och klockfasen 0, vilket innebär att klocksignalen är låg i dess inaktiva läge samt att data sampelas vid stigande flank och att data skickas vid fallande flank. I mode 1 är klockpolariteten 0 och klockfasen 1, vilket innebär att klocksignalen är låg i dess inaktiva läge samt att data sampelas vid fallande flank och att data skickas vid stigande flank. I mode 2 är klockpolariteten 1 och klockfasen 0, vilket innebär att klocksignalen är hög i dess inaktiva läge samt att data sampelas vid fallande flank och att data skickas vid stigande flank. I mode 3 är klockpolariteten 1 och klockfasen 1, vilket innebär att klocksignalen är hög i dess inaktiva läge samt att data sampelas vid stigande flank och att data skickas vid fallande flank [15]. Figur 4 visar ett exempel på hur dataöverföring med SPI kan se ut i mode 0.

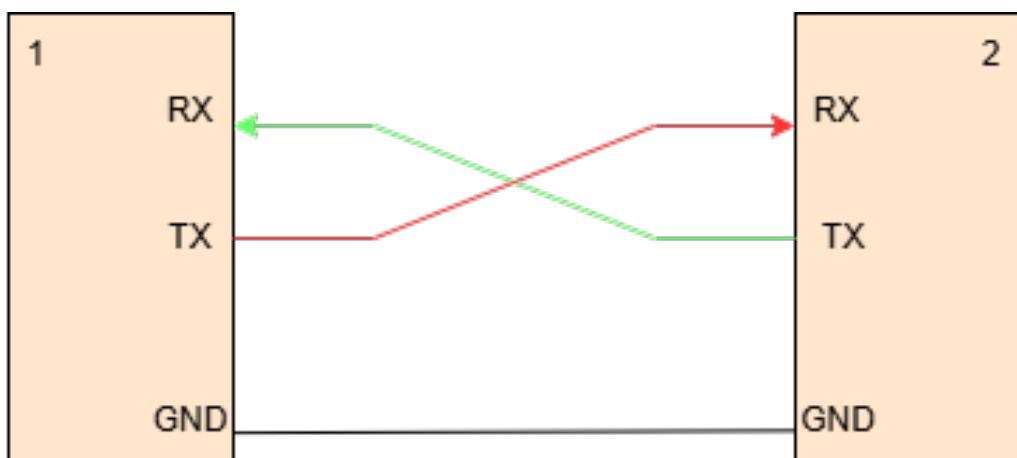


Figur 4: Dataöverföring SPI [15].

Jämfört med andra kommunikationsmetoder så är SPI bättre anpassat för system som kräver lågenergi. SPI kan dock vara relativt känsligt för brus och störningar [16]. SPI har endast möjlighet för ett fåtal slavar, enheter, eftersom respektive slav kräver en separat Slave Select-ledning. I praktiken används oftast 2-4 enheter innan det blir ohanterligt.

3.3.3 Universal Asynchronous Receiver/Transmitter (UART)

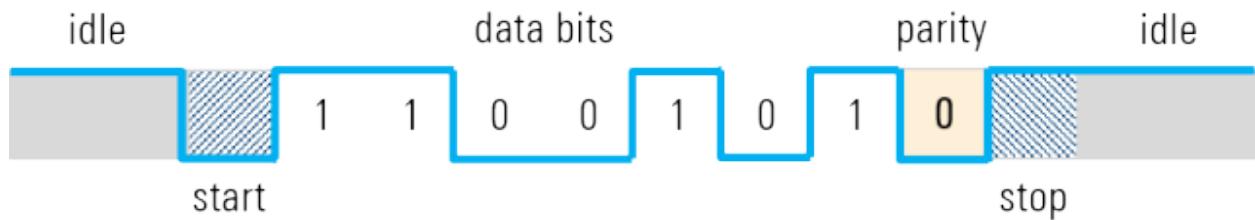
Universal Asynchronous Receiver/Transmitter, UART, är ett protokoll för att skicka data mellan olika enheter som kan agera som en slags förbindelse för asynkron kommunikation, vilket till exempel gör det möjligt för enheter med olika klockfrekvenser att kommunicera utan problem. Data sänds i form av protokoll eller ramar med ett visst antal bitar som symboliseras olika information, se Figur 5. Ramen består av en start-bit, ett specifikt antal data-bitar (vilket oftast är 8 stycken) och en slut-bit, se Figur 6. Sekvensen av bitar kan även innehålla en bit som har med felsökning att göra om detta önskas. Med UART så kommer sändare och mottagare att dela på en och samma klocka, vilket betyder att de måste enas om samma baud-rate, alltså hur många gånger per sekund signalen ändras [17].



Figur 5: Översikt av UART-protokoll.

Eftersom UART är asynkron så behöver inkommende data signaleras via en startbit. Startbiten är en övergång från ett inaktivt högt tillstånd till ett lågt tillstånd som sedan följs av ett visst antal användardatabitar. Slutet på datan indikeras via slutbiten som antingen är en övergång tillbaka till det höga tillståndet, eller fortsättning på det höga tillståndet. En andra slutbit kan adderas med dess funktion att ge mottagaren tid att göra sig redo för nästa ram, detta är dock ovanligt.

Databitarna är användardata och kommer som nämnt innan direkt efter startbiten. Antalet bitar är vanligtvis 7-8 men kan vara allt emellan 5-9. Databitarna är vanligtvis överförda med MSB (minst signifikanta biten) först [17].



Figur 6: Bitformat för UART [17].

UART har flera fördelar där den främsta är dess simplicitet. Det är enkelt att installera både dess mjuk- och programvara. UART kräver väldigt lite strömförbrukning med dess variabla hastighet och kan möjliggöra för full-duplexkommunikation med dess gränssnitt. Full-duplex innebär som nämnts att kommunikation kan ske i båda riktningar samtidigt.

Å andra sidan så besitter UART några släende nackdelar där dess gränssnitt i princip är begränsad till två enheter. I de flesta fall är UART långsammare än I2C och SPI [18].

4 DISKUSSION

I följande avsnitt diskuteras huruvida de olika kommunikationsmetoderna är lämpliga för användning på en lagerrobot samt vårt val av kommunikationsmetoder. Den trådlösa kommunikationen innebär kommunikation mellan PC och robot, mer specifikt Raspberry Pi:n. Den seriella kommunikationen innebär kommunikation på bussen mellan Raspberry Pi:n och styr- och sensorenheten.

4.1 Trådlös kommunikation

Trots att Bluetooth och Wi-Fi skiljer sig åt i funktion, så skiljer det inte mycket i montering på en robot i praktiken. Förutom räckvidden, i vilket Wi-Fi har ett övertag, var det mest frågan om säkerhet och enkelhet som bidrog som argument för det slutgiltiga valet. Som nämnts i avsnitt 4.2.1 finns det risk för att utomstående användare kopplar upp sig mot det Wi-Fi nätverket som används inom projektet. Detta kan inte bara ge störningar utan risken finns även att dessa användare lyckas ändra eller ta bort data som är av vikt för funktionaliteten hos roboten.

Gällande frågan om räckvidd så har Bluetooth endast en räckvidd på cirka 10 meter. I detta projekt kommer en mycket större räckvidd ej vara nödvändig. De olika enheterna som kommunicerar via Bluetooth, i detta fall en PC och roboten, kommer nödvändigtvis inte behöva vara längre ifrån varandra än storleken på lagret. Lagret kommer bestå av ett rutnät med känd storlek efter vilket vi kan anpassa avståndet mellan enheterna.

Utöver ovanstående argument har Bluetooth ytterligare en fördel som till stor del avgör det slutgiltiga valet. Detta rör enkelheten i hur uppkoppling via Bluetooth sker till skillnad från Wi-Fi. För Wi-Fi kommer det alltid krävas tre enheter: PC, Raspberry Pi (roboten) och en router. Bluetooth kräver endast de två förstnämnda enheterna. Detta ger oss som grupp en sak mindre att tänka på, eftersom PC:n och roboten är två enheter vi alltid bär med oss, till skillnad från en router.

4.2 Seriell kommunikation

De olika egenskaperna vi kollade på som låg till grund för valet av kommunikation är komplexitet, energiåtgång, hastighet samt simplicitet vid uppkoppling på virkort.

Med komplexitet hos de olika kommunikationssättet syftar vi mest på hur dataöverföringen fungerar. En fördel med I2C är dess egenskap att möjliggöra för flera masters, men det innefattar också en större komplexitet vid programmering av bussen. SPI däremot har inte denna egenskap av flertalet masters varav det krävs en mindre omfattning av kod. Färre masters ger ett mindre komplext system vilket skulle kunna underlättat implementeringen av buss. Färre masters ger också naturligt färre ledningar att hålla koll på. UART är också enkelt och kräver väldigt lite kod, men nackdelen med denna är dess begränsning i enheter. UART är i princip begränsat till endast två enheter, vilket kan bli problematiskt för en robot med flertalet servon och sensormoduler.

En skillnad mellan I2C och SPI är deras hastighet. I2C har begränsningar gällande maxhastighet vilket SPI inte har. SPI är alltså ett mycket snabbt kommunikationssätt vilket skulle kunna vara en fördel. Även UART är som nämnts relativt långsamt.

Om man tittar på energiförbrukning hos de olika kommunikationssättet så är SPI bra anpassat för lågenergisystem,

I2C ändå drar mer energi och skulle kunna vara bättre anpassat för större system. På roboten kommer vi endast ha tillgång till batterier och av denna anledning är det stor fördel med en kommunikationsenhet som är anpassat för lågenergisystem. Utöver SPI är då även UART en bra kandidat för projektet då dess enkelhet bidrar till låg energiåtgång.

Något vi även fick mer kunskap om vid skrivandet av denna förstudie är hur data skickas vid de olika kommunikationssätten. Vid I2C-kommunikation så skickas data med en avslutande ACK-bit efter varje byte, detta kan användas för att säkerställa att data verkligen tagits emot av respektive slav. Denna uppbyggnad av data finns inte i SPI-protokollet, och det går alltså inte att verifiera om data tagits emot korrekt eller ej. Detta kan vara användbart för oss under arbetsprocessen, förmodligen kommer det att underlätta i felsökningsprocesser eller liknande i fall där vi vill undersöka om data skickats korrekt eller ej.

En sista egenskap som vi ser över hos de olika systemen är hur pass komplex uppkopplingen av sladdar blir på virkortet som sitter på roboten. Då vi kommer ha flertalet enheter bidrar detta till många sladdar och därfor är det en fördel om antalet kan minska för att möjliggöra kommunikationen via bussen. För UART krävs det endast ett fåtal på grund av dess simplicitet och begränsning till enheter. Som följd av komplexiteten av I2C som beskrivits ovan kan detta bidra till en ökad mängd kopplingar och sladdar/kablar. Av denna förklaring kan då SPI, precis som UART, sänka antalet kopplingar och sladdar och bidra till att projektet kan bli något smidigare att genomföra.

5 SLUTSATSER

I följande avsnitt kommer det att dras en slutsats gällande vilka kommunikationsmetoder som passar bäst för roboten och därmed kommer att användas inom projektet.

Baserat på ovan nämnda egenskaper kommer Bluetooth att användas som trådlös kommunikation mellan PC och Raspberry Pi:n samt SPI för kommunikation mellan Raspberry Pi:n och de olika styr- och sensorenheterna. De mest avgörande egenskaperna som är av vikt för detta projekt är låg energiförbrukning och simplicitet. Trots att även UART har en låg energiförbrukning är den inte bäst lämpad för detta projekt då UART endast är anpassat för två enheter. I2C valdes främst bort på grund av dess komplexitet och höga förbrukning av energi.

REFERENSER

- [1] R. P. Ltd, *Getting started with your Raspberry Pi*, Hämtad: 2025-02-11, [Online]. Tillgänglig: <https://www.raspberrypi.com/documentation/computers/getting-started.html>, 2012-2025.
- [2] R. P. Foundation, *GPIO pins*, Hämtad: 2025-03-31, [Online]. Tillgänglig: <https://projects.raspberrypi.org/en/projects/physical-computing/1>, no date.
- [3] ProXPN, *Vad är WiFi och vad står det för?* Hämtad: 2025-02-11, [Online]. Tillgänglig: <https://proxpn.com/sv/wifi/vad-ar-wifi/>, last updated 2024.
- [4] C. B. Portnox, *The Origin of the Word “Wi-Fi”: A Dive into Tech Etymology*, Accessed: 2025-02-23, 2024-06-25. URL: <https://www.portnox.com/blog/security-trends/origin-of-wifi/>.
- [5] Strålsäkerhetsmyndigheten, *Trådlösa datornätverk och wifi*, Accessed: 2025-02-23, 2024-12-09. URL: <https://www.stralsakerhetsmyndigheten.se/omraden/magnetfält-och-tradlos-teknik/tradlos-teknik/tradlosa-datornatverk-och-wifi/>.
- [6] I. Internetkunskap, *Wifi*, Accessed: 2025-02-23, n.d. URL: <https://internetkunskap.se/artiklar/ordlista/wifi/>.
- [7] E. Britannica, *Bluetooth*, Hämtad: 2025-02-18, [Online]. Tillgänglig: <https://www.britannica.com/technology/wireless-communications>, last updated 2025.
- [8] S. C. Jan Storhaug Bill Christman, *Bluetooth Lingo: What is the Difference between Bluetooth and 2.4 GHz?* Hämtad: 2025-04-23, [Online]. Tillgänglig: <https://www.audiologyonline.com/ask-the-experts/bluetooth-lingo-what-difference-between-23092>, 2018-06-11.
- [9] ISY, *Kommunikation Raspberry Pi - ATmega*, Hämtad: 2025-02-23, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/page/kommunikation/>, 2003.
- [10] ISY, *I2C*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/page/kommunikation/i2c/>, 2003.
- [11] J. B. Jonathan Valdez, *Understanding the I2C Bus*, Hämtad: 2025-04-14, [Online]. Tillgänglig: <https://www.ti.com/lit/an/slva704/slva704.pdf?ts=1744594000731>, 2015.
- [12] J. H. Nicholas Zambetti Karl Söderby, *Inter-Integrated Circuit (I2C) Protocol*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://docs.arduino.cc/learn/communication/wire/>, 2024.
- [13] R. Keim, *Introduction to the I2C Bus*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://www.allaboutcircuits.com/technical-articles/introduction-to-the-i2c-bus/>, 2015.
- [14] M. Sufyan, *SPI Protocol: Revolutionizing Data Communication in Embedded Systems*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://www.wewolver.com/article/spi-protocol>, 2024.
- [15] P. Dhaker, *Introduction to SPI Interface*, Hämtad: 2025-04-14, [Online]. Tillgänglig: <https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html>, No date.
- [16] S. Writer, *The Differences Between I2C and SPI (I2C vs SPI)*, Hämtad: 2025-04-14, [Online]. Tillgänglig: https://www.totalphase.com/blog/2021/07/i2c-vs-spi-protocol-analyzers-differences-and-similarities/?srsltid=AfmBOoqlppRKT8i2vnd8i9PB8iT1LYWNZ2ocovJFF_FDrDGVujt0N5O9, 2021.

- [17] V. Kanade, *What Is Universal Asynchronous Receiver-Transmitter (UART)? Meaning, Working, Models, and Uses*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://www.spiceworks.com/tech/networking/articles/what-is-uart/>, 2024.
- [18] N. Busler, *UART - Universal Asynchronous Receiver-Transmitter (universell asynkron mottagare-sändare)*, Hämtad: 2025-04-14, [Online]. Tillgänglig: <https://picockpit.com/raspberry-pi/sv/uart-the-universal-asynchronous-receiver-transmitter/#Advantages>, 2022.

13.2 Sensorenhetens förstudie

Hur kan sensorer integreras i en lagerrobot?

Andreas Nordström
Sigge Rystedt

16 april 2025

Version 1.0



Status

Granskad	Sigge Rystedt	2025-04-16
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Inledning	1
2	Problemformulering	1
3	Teoretisk bakgrund	1
3.1	Avståndssensor	1
3.2	Reflexsensor	6
3.3	Vinkelhastighetssensor	7
3.4	Odometer	8
3.5	Kamera	8
4	Diskussion	8

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-02-24	Första utkast	AN, SR	AN, SR
0.2	2025-04-01	Andra utkast	AN, SR	AN, SR
1.0	2025-04-16	Första version	AN, SR	AN, SR

1 INLEDNING

Den här förstudien görs inom kursen TSEA56 och handlar om att undersöka vilka sensor typer som passar bäst för att bygga en autonom lagerrobot. Förstudien görs innan konstruktionen av roboten för att få en bra överblick över vilka sensorer som finns att välja mellan, vilka som passar för uppgiften och hur de fungerar i praktiken.

Sensorer som har liknande funktioner jämförs utifrån olika kriterier, till exempel noggrannhet, räckvidd och hur lätt de kan användas tillsammans med resten av systemet. Målet är att hitta de sensorer som fungerar bäst för att roboten ska kunna navigera och arbeta på ett bra sätt i ett lager.

Syftet med förstudien är att få en bättre förståelse för de sensorer som finns tillgängliga i kursen TSEA56, så att det blir lättare att välja rätt sensorer till projektet. Endast de sensorer som finns i Vanheden-labbet används i studien, eftersom det är dessa som går att använda i kursens praktiska delar [1].

2 PROBLEMFORMULERING

I kursen finns flera olika sensorer tillgängliga där många kan uppfylla samma funktion inom lagerroboten med varierande resultat. Dessutom kan samma sensor ofta implementeras på många olika sätt. Därför är det viktigt att undersöka de olika sensorerna utifrån frågorna:

- Hur fungerar sensorn?
- Hur kan sensorn användas i lagerroboten?
- Hur ska sensorn integreras i systemet?

3 TEORETISK BAKGRUND

I det här kapitlet presenteras den tekniska och teoretiska grunden som behövs för att förstå och jämföra olika sensorer inför konstruktionen av en autonom lagerrobot. Kapitlet är uppdelat efter sensor typer där varje avsnitt beskriver hur sensorn fungerar, dess tillämpning inom projektet och dess praktiska begränsningar. Informationen bygger främst på datablad, teknisk dokumentation och kursmaterial från TSEA56. Endast sensorer som finns tillgängliga i Vanheden har undersökts.

Kunskapsbasen som ligger till grund för förstudien baseras främst på datablad och teknisk dokumentation. Andra internetkällor kan komma att användas. De sensorer som listas i Vanheden används som utgångspunkt för den vidare informationssökningen.

3.1 Avståndssensor

I det följande kapitlet analyseras olika avståndsmätare, deras teoretiska grund, tillämpningar inom projektet och hur de kan integreras i systemet.

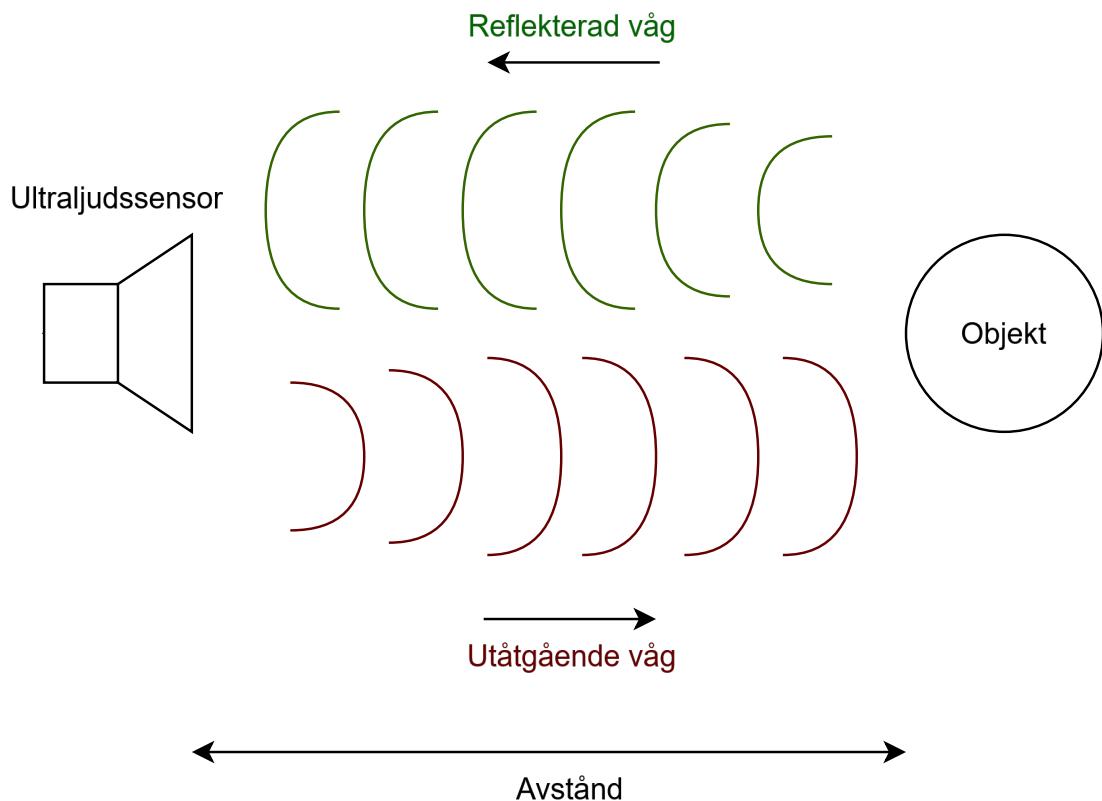
3.1.1 *Ultraljud*

Ett konstruktionsalternativ för avståndssensorerna är ultraljudssensorer. Det är en typ av akustisk sensor, vilket innebär att den använder ljud för att mäta avstånd. De är ofta mindre noggranna än andra alternativ, men har potential att användas för hinderdetektering [2] [3].

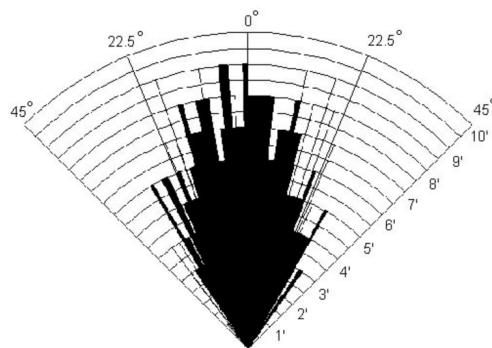
En ultraljudssensor mäter avstånd genom att använda ekotid, vilket visas i figur 1. Sensorn skickar ut en högfrekvent ljudvåg, som reflekteras från ett objekt och där efter tas emot av sensorn. Tiden det tar för en ljudvåg att färdas fram och tillbaka till sensorn är ekotiden. Med denna kan avståndet räknas ut enligt

$$\text{Avstånd} = \frac{\text{Ekotid} \cdot \text{Ljudets hastighet i luft}}{2}. \quad (1)$$

Ultraljudssensorer är vanligtvis mindre noggranna än optiska sensorer, men de är ofta mer kostnadseffektiva och mindre känsliga för störningar. Den ultraljudssensor SRF04 som används i kursen har ett mätintervall på 3-300 cm och sänder ut en ljudvåg med frekvensen 40 kHz. SRF04 har ett detektionsområde som är ungefär 30 grader brett vilket kan ses i figur 2 [2] [3].

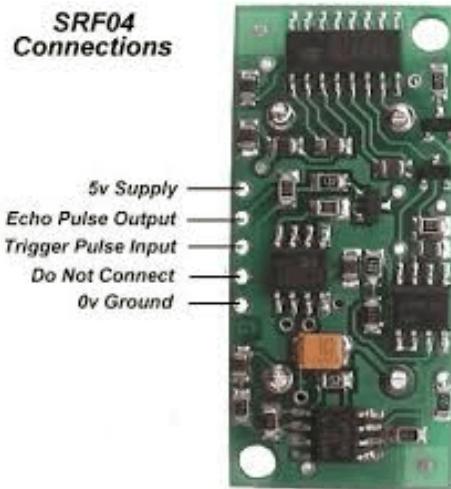


Figur 1: Översikt av ultraljudssensor.



Figur 2: Detektionsområde för SRF04. Källa: [3].

Ultraljudssensorn kan användas i lagerroboten för att detektera hinder, vilket innebär att de skulle behöva vara installerade på robotens framsida. Vidare kräver sensorn fyra anslutningar för att fungera som illustreras i figur 3. Dessa är ström, jord och signalanslutningarna vars data behandlas i mikrodatorn. De två signalanslutningarna är utsignalen av ekot respektive insignalen för ljudpulsen som skickas iväg. Pulsen behöver vara aktiv i minst $10 \mu\text{s}$ för att generera en ljudpuls. Mottagaren är redo att ta emot ljud $100 \mu\text{s}$ efter att ljudpulsen har skickats iväg. Det är för att den ej ska uppfatta den utskickade ljudpulsen direkt som en mottagen puls. När mottagaren aktiveras aktiveras även utsignalen från ekot [3].



Figur 3: Anslutningar för SRF04. Källa: [4].

3.1.2 IR och triangulering

IR-sensorer är ett konstruktionsalternativ för att lagerroboten ska kunna detektera hinder. De använder sig av infrarött ljus och vinkelberäkningar för att bestämma avstånd.

Kursen TSEA56 har två tillgängliga IR-sensorer med passande detekteringsintervall. Den ena modellen, GP2D120, har ett intervall på 4-30 cm och den andra modellen, GP2Y0A21, 10-80 cm. Båda sensorerna använder sig av triangulering, vilket är en process som illustreras i figur 4. En sändare i sensorn skickar ut infrarött ljus som sedan reflekteras av ett objekt. Därefter tar en mottagare på sensorn emot det reflekterade ljuset med en viss vinkel som sedan används för att avgöra avståndet till objektet. En nackdel med IR-triangulering är att det infraröda ljuset som skickas iväg är väldigt smalt, vilket kan resultera i att objekt som befinner sig i lagerrobotens periferi missas. Detta kommer dock inte vara ett problem då avståndssensorn är till för att upptäcka hinder rakt framför roboten. Ytterligare en nackdel med IR-triangulering är att om flera sensorer är riktade åt samma håll kan interferens uppstå, vilket kan göra mätningarna opålitliga [5].

Sensors utsignal är en analog spänning. Det är viktigt att notera att utsignalen ej är linjär. Olinjäriteten är särskilt påtaglig vid korta avstånd. I databladet för sensorn finns det exempel på spänningsvärdet för många olika avstånd. För ett godtyckligt avstånd inom detekteringsintervallet kan man ha en skalfaktor dividerat med spänningen från utsignalen. Detta görs efter att det digitala värdet efter ADC-konverteringen görs om till volt. Datat blir då i cm. Beräkningar med flyttal är väldigt långsamma i mikrodatorn och ska gärna undvikas så mycket det bara går. Ett alternativ till att använda flyttal (float) på mikrodatorer är att representera icke-heltal med så kallad fixed point-aritmetik. I fixed point används heltal för att approximera decimaltal genom att man förbestämmer hur många bitar (eller decimaler) som ska representera den fraktionella delen. Det är alltså ett sätt att simulera decimaler med heltal, vilket är mycket snabbare än att använda inbyggt stöd för flyttal – särskilt på enheter som AVR-mikrodatorer, där flyttalsberäkningar är långsamma och resurskrävande. I databladet finns en graf med relationen mellan spänning och avstånd. Det kan användas för att skapa en tabell och då slipper man beräkningarna [6] [7].



Figur 4: Översikt av triangulering med IR.

3.1.3 Laser

Detta kapitel beskriver lasersensorns funktion och varför den kan behövs på roboten. Lasersensorn fungerar principiellt på samma sätt som de andra två avståndssensorerna med skillnad att den har högre precision. Sensorn som används i kursen är en VL53L0X från Adafruit, även kallad mikrolidar. Till skillnad från en ultraljudssensor har den en mycket smal detekteringskon, och jämfört med en IR-sensor har den högre mätnoggrannhet utan linjäreritsproblem. Sensorns detekteringsintervall sträcker sig från 50 till 1200 mm och fungerar annars på samma sätt som de andra två avståndssensorerna: ljus skickas, reflekteras och tas emot av en mottagare i sensorn som beräknar avståndet [8].

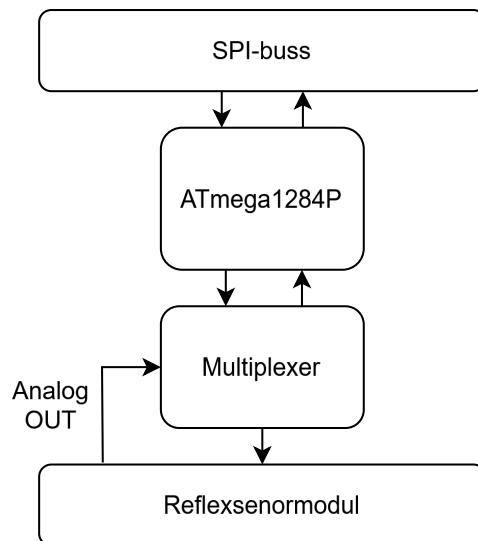
Sensorn kan användas för avståndsmätning till objekten som robotarmen ska kunna plocka upp. Om detta ska ske autonomt behövs hög precision på avståndet, varav denna typ av sensor passar bra. Således behövs två sensorer, en på höger och en på vänstersidan av roboten. Lasersensorn behöver då anslutningar för spänning, jord och två pinnar för I2C. De två sensorerna kan vara kopplade till samma buss så länge man ser till att båda inte har samma adress [8].

3.2 Reflexsensor

I följande kapitel undersöks hur reflexsensorer och reflexsensormodulen kan användas och implementeras på roboten. För att roboten ska kunna identifiera och tolka tejplinjer på marken kan reflexsensorer användas. Sensorerna består av detektorer vars utdata måste omvandlas och tolkas av en AVR-mikrodator. Dessutom måste sensorerna multiplexas för att undvika överhettning [9].

En reflexsensor innehåller en detektor som avger låg respektive hög utsignal när underlaget reflekterar mycket eller lite ljus [10]. I reflexsensormodulen finns 11 reflexsensorer monterade. Modulen kan kopplas till en multiplexer för att individuellt kontrollera vilka detektorer som är av eller på.

Reflexsensormodulen kan användas till att läsa av tejplinjerna under körning och behövs för att roboten ska kunna följa tejplinjer och identifiera plockstationerna samt hämtstationen.



Figur 5: Diagram av sensorenheten.

Reflexsensormodulen ska vara monterad under roboten några millimeter från underlaget. Figur 5 visar det översiktliga flödesschemat för reflexsensormodulen där sensorerna är kopplade till en ATmega1284P. Mikrodatorn kommer att omskriva sensorns analoga utsignal till 10-bitars genom en inbyggd A/D-omvandlare [11]. Därefter kommer den att tolka indata och skicka informationen till styrenheten. Ifall roboten befinner sig på sträckan till lagret från hämtningstationen kommer en enkel tyngdpunktsberäkning användas enligt

$$k_T = \frac{\sum_k m_k k}{\sum_k m_k} \quad (2)$$

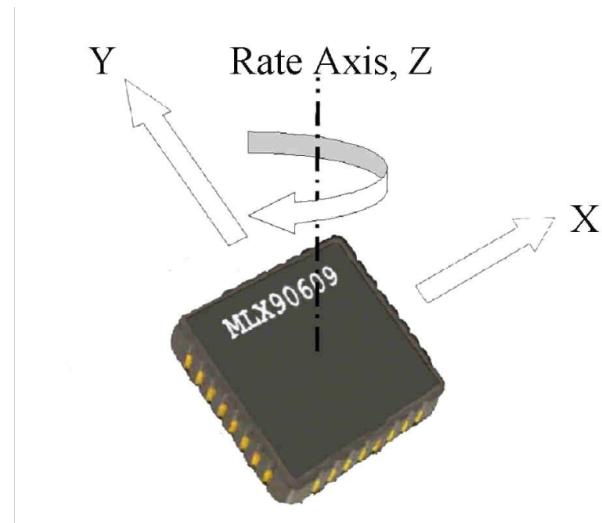
där k_T och m är tyngdpunkten respektive magnituden en reflektor detekterar. Syftet är att hitta tyngdpunktenens avvikelse från ett jämviktsvärde för att beräkna hur mycket roboten ska svänga åt höger eller vänster. För att identifiera hämtningstationen används dock de yttersta detektorerna. Roboten är framme vid hämtningstationen när båda är höga

och lagret när bara en av dem är hög. De kommer fortsätta att användas på samma sätt för att identifiera korsningar och plockstationer i lagret. AVR-mikrodatorn kommer dessutom att kontrollera multiplexern som är kopplad till reflektionsmodulen som periodvis kommer att stänga av detektorerna för att undvika överhettning.

3.3 Vinkelhastighetssensor

I detta kapitel presenteras information om hur en vinkelhastighetssensor fungerar och hur den kan implementeras på roboten. Vinkelhastighetssensorn används för att mäta vinkelhastighet och hjälpa till med orientering och stabilisering. Den sensorn som finns tillgänglig är MLX90609, vilken mäter vinkelhastigheten på z-axeln enligt figur 6. Sensorn är en *Vibrating Structure Gyroscope* (VSG) som använder corioliseffekten för att beräkna rotation. Det innebär att sensorn innehåller en komponent som mekaniskt oscillerar. När vinkelhastighetssensorn sedan roterar, alltså roboten svänger, skapas ytterligare en oscillation av corioliseffekten. När detta sker mäts en skillnad i kapacitans ut som därefter görs om till en spänning som representerar svängning [12].

Vinkelhastighetssensors utsignal är en analog vinkelhastighet som är kopplad till AVR-mikrodatorns ADC-omvandlare. Då styrenheten arbetar med vinklar måste vinkelhastigheten integreras över tid för att omvandla denna till en vinkel. Vid integration är tidssteget viktigt, det kan därför vara bra att sampla med en bestämd frekvens med hjälp av hårdvarutimers [12].



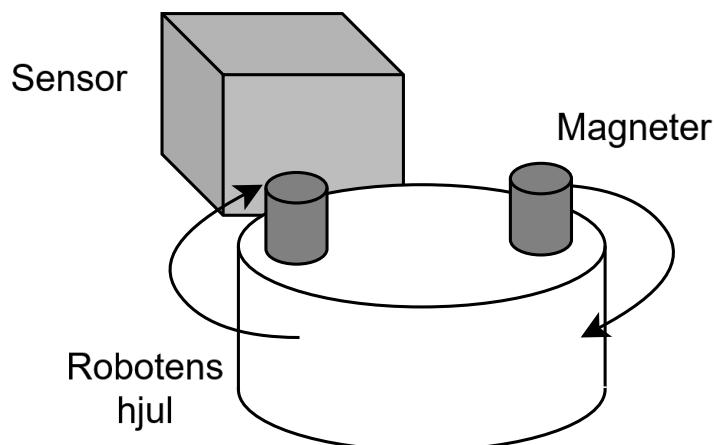
Figur 6: Rotationsaxeln för Vinkelhastighetssensor. Källa: [12].

Sensorerna kommer att användas för att komplettera befintlig data, framför allt då roboten behöver svänga. Vinkelhastighetssensor behövs eftersom roboten kan glida eller slira på underlaget, vilket gör mätningarna opålitliga utan en sådan sensor. Vinkelhastighetssensor behöver placeras vinkelrätt mot den axel som den mäter på, förslagsvis i mitten av roboten.

3.4 Odometer

Kapitlet beskriver de olika metoderna för att implementera en odometer i lagerroboten men eftersom en odometer inte kommer att användas i projektet, så ges endast en övergripande beskrivning.

En odometer används för att mäta färdat avstånd och sensorn kan vara elektrisk eller mekanisk. En av metoderna använder halleffekten. Hjulen är då utrustade med magneter, illustrerat i figur 7. Sensorn mäter spänningen och hur magnetfältet förändras. Med den informationen kan antalet hjulrotationer mäts sedan används för att beräkna avståndet som roboten har färdats [13]. En annan metod som går att använda är att ha en reflexsensor som mäter svart-vita band på hjulen. Därmed går det att avgöra rotationer. Man kan även använda en kugghjulsskiva som är 3D-utskriven. Den i sin tur går att läsa av med två olika läsgafflar som finns att tillgå i kursen.



Figur 7: Översikt över hallsensorn på ett hjul.

3.5 Kamera

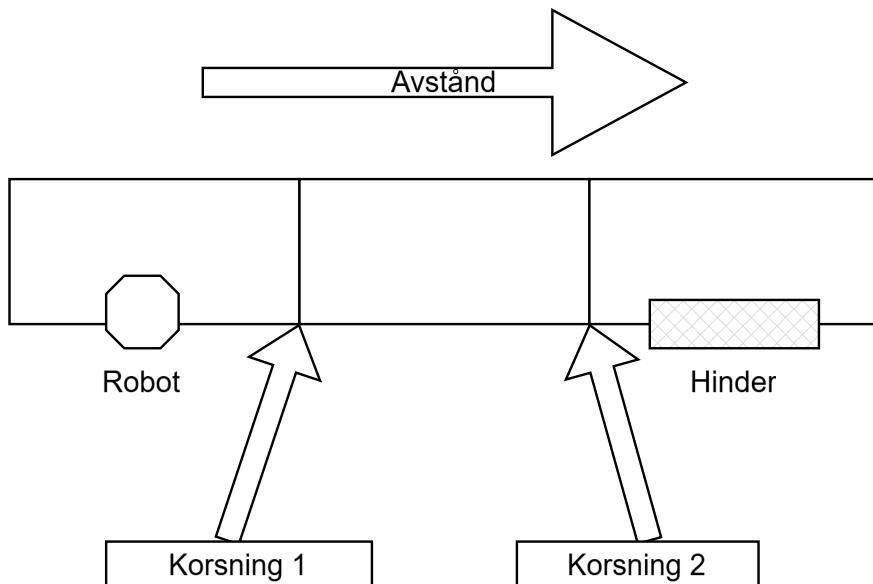
Följande kapitel beskriver kameran som finns att tillgå i kursen. Då kameran förmodligen inte kommer att användas så görs endast en kort beskrivning av den. Det är nödvändigt med en kamera på lagerroboten för att det ska vara möjligt att manuellt plocka upp varor utan att användaren ser roboten. Detta är dock inget krav i projektet utan något som kan implementeras i mån av tid. Modellen som är tillgänglig är en Raspberry Pi Camera Module 2 som kopplas till mikrodatorns CSI-port via en plattkabel. Kameraströmmen måste sedan skickas via blåtand till PC:n. Blåtandsmodulen som finns tillgänglig har en överföringshastighet på ungefär 0,12 MBPS och strömning av en 360p-video med 30 bilder per sekund kräver minst 1 MBPS [14] [15]. Detta innebär att videon måste kraftigt komprimeras för att kunna strömmas till PC:n.

4 DISKUSSION

För att avgöra vilka sensorer som är bäst lämpade till konstruktionen av en lagerrobot har olika sensorer med olika funktioner undersökts i resultatet. För avståndsbördningen kan ultraljud-, IR- eller lasersensorer användas. Sen-

sorer som vinkelhastighetssensorn, odometern och kameran har diskuterats för att undersöka om den behöver vara installerad på robotten. En reflexsensormodul måste vara installerad och i kapitlet 3.2 beskrivs hur denna potentiellt kan implementeras.

Då det finns flera avståndssensorer med samma funktion, kommer en diskussion om deras för- och nackdelar att föras. De största skillnaderna mellan den akustiska och IR-sensorn är tekniken de använder för avståndsbedömning samt detekteringsavståndet. Ultraljudssensorn har ett längre detekteringsintervall än båda IR-sensorerna. Även om ultraljudssensorn kan mäta avstånd upp till 3 m, så kommer det aldrig behövas i det här projektet. Avståndssensorernas primära uppgift är att upptäcka hinder i lagermiljön. Sensorer med lång detekteringsförmåga kan upptäcka hindren väldigt tidigt vilket kan resultera i att den uträknade vägen kan göras mer optimal. Detta illustreras i figur 8 där sensorn svänger vänster redan i korsning 1, istället för korsning 2. Vidare så behöver avståndssensorn bara upptäcka hinder som är rakt framför robotten, alltså är konen på 30 grader som ultraljudssensorn har som detekteringsområde överflödig. IR-sensorerna är mer noggranna än ultraljudssensorn, men de är också mer känsliga för störningar. Detta bör dock inte vara ett problem så länge inte flera IR-sensorer sitter nära varandra och mäter i samma riktning. På bilderna i Vanheden ser det även ut som att ultraljudssensorn tar mer plats än IR-sensorerna.



Figur 8: Ultraljudssensorn problem med för tidig hinderupptäckt.

Lasersensorn är mer exakt än både ultraljuds- och IR-sensorerna, men enligt ISY är den svårare att använda. Den är tänkt att mäta avståndet mellan robotten och de varor som ska plockas upp med robotarmen. För autonom hantering är noggranna mätningar avgörande för att säkerställa precision. Dessutom är det viktigt att robotten är i rätt höjd i förhållande till varan, eftersom lasern annars riskerar att missa den [1].

Vinkelhastighetssensorn behövs på robotten då enbart avståndsbödning inte är tillräckligt för att säkerställa att robotten rör sig på ett korrekt sätt. Om hjulen glider på underlaget så räcker det inte med data från de andra sensorerna för att göra en korrekt rotation, utan då behöver vinkelhastighetssensorn kompensera för detta.

Med hänsyn till projektets restriktioner samt hur lagermiljön ser ut, så kommer en odometer ej att vara nödvändig att ha på roboten.

I mån av tid kan manuell upplockning av varor utan att se roboten implementeras och då krävs en kamera. Raspberry Pi Camera Module 2 är den enda som finns tillgänglig i Vanheden. På grund av de tidigare beskrivna problemen med att strömma videon till PC:n är det dock låg sannolikhet att det kommer finnas tid att implementera kameran.

Sammanfattningsvis kommer följande sensorer mest lämpade att använda i projektet: GP2Y0A21 (avståndssensor), MLX90609(Vinkelhastighetssensor) och reflexsensormodulen. Lasersensorn GP2Y0A21 är det bästa valet för navigering i lagerutrymmet då roboten kommer att följa en fixerad väg och att det ej är nödvändigt att använda ultraljudssensorn. Reflexsensormodulen kommer installeras med en multiplexer och använda en enkel tyngdpunktsberäkning för att beräkna vilken riktning lagerroboten ska åka. Reflexsensorn är helt nödvändig då roboten ska följa tejp på marken. För att undvika problem med att hjulen glider på underlaget vid svängning kommer en vinkelhastighetssensor att användas. Varken en odometer eller kamera kommer behövas på roboten.

REFERENSER

- [1] ISY, *Sensorer*, Hämtad: 2025-03-31, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/page/sensorer/>, 2003.
- [2] Vayuyaan, *Ultrasonic Sensor in Robotics*, Hämtad: 2025-02-06, [Online]. Tillgänglig: <https://vayuyaan.com/blog/ultrasonic-sensor-in-robotics/>, 2023.
- [3] ISY, *SRF04 Ultrasonic Range Finder*, Hämtad: 2025-02-06, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/srf04.pdf>, 2003.
- [4] Robot-Electronics, *SRF04 - Ultra-Sonic Ranger Technical Specification*, Hämtad: 2025-02-24, [Online]. Tillgänglig: <https://www.robot-electronics.co.uk/htm/srf04tech.htm>, 2003.
- [5] S. O. Robots, *SENSORS - SHARP IR RANGE FINDER*, Hämtad: 2025-02-11, [Online]. Tillgänglig: https://www.societyofrobots.com/sensors_sharpirrange.shtml, 2025.
- [6] ISY, *GP2Y0A21YK Optoelectronic Device*, Hämtad: 2025-04-01, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/gp2y0a21.pdf>, no date.
- [7] Pololu, *Sharp/Socle GP2Y0A21YK0F Analog Distance Sensor 10-80cm*, Hämtad: 2025-04-01, [Online]. Tillgänglig: <https://www.pololu.com/product/136>, no date.
- [8] ISY, *Adafruit VL53L0X Time of Flight Micro-LIDAR Distance Sensor Breakout*, Hämtad: 2025-02-11, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/VL53L0X.pdf>, 2017.
- [9] ISY, *Reflektion*, Hämtad: 2025-04-15, [Online]. Tillgänglig: https://da-proj.gitlab-pages.liu.se/vanheden/page/avr_raspberry/.
- [10] ISY, *Reflexsensor*, Hämtad: 2025-02-18, [Online]. Tillgänglig: https://da-proj.gitlab-pages.liu.se/vanheden/pdf/reflex_sensor.pdf, 2007.
- [11] A. Corporation, *ATmega16(L) Datasheet*, Tillgänglig på: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/atmega16.pdf>, 2007.
- [12] ISY, *MLX90609 Angular Rate Sensor (Standard version)*, Hämtad: 2025-02-17, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/mlx90609.pdf>, 2008.
- [13] R. C. AB, *What is a Hall effect sensor?* Hämtad: 2025-02-18, [Online]. Tillgänglig: <https://se.rs-online.com/web/generalDisplay.html?id=ideas-and-advice/hall-effect-sensors-guide>, 2025.
- [14] golightstream, *What is Video Bitrate, and How Does it Affect Video Quality?* Hämtad: 2025-04-01, [Online]. Tillgänglig: <https://golightstream.com/what-is-video-bitrate/>, 2022.
- [15] ISY, *FireFly Bluetooth Modem - BlueSMiRF Gold*, Hämtad: 2025-04-01, [Online]. Tillgänglig: <https://da-proj.gitlab-pages.liu.se/vanheden/pdf/firefly.pdf>, no date.

13.3 Styrenhetens förstudie

Förstudie om reglerteknik för lagerrobot

Linus Funquist, Lisa Ståhl

2025-04-17

Version 1.1



Status

Granskad	Linus Funquist, Lisa Ståhl	2025-04-17
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Inledning	1
1.1	Syfte	1
2	Problemformulering	1
3	Bana och rörelseplanering	1
3.1	Algoritm för att hämta varorna en och en	1
3.2	Algoritm för att hämta alla varor i en färd	5
4	Spårföljning	6
5	Styrning av robotarmen	7
6	Diskussion och slutsatser	8
6.1	Nödvändiga styrmoder	9
6.2	Följning av tejp	9
6.3	Styrning av robotarm	9
6.4	Bana och rörelseplanering	10
7	Referenser	13
8	Appendix	13

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
0.1	2025-02-24	Första utkast	LF, LS	LF, LS
1.0	2025-04-07	Första version	LF, LS	LF, LS
1.1	2025-04-17	Andra version	LF, LS	LF, LS

1 INLEDNING

Detta dokument är en förstudie i kursen TSEA56, Elektronik kandidatprojekt och kommer att handla om att bygga och programmera en lagerrobot som autonomt ska kunna ta sig genom ett lager och hämta upp lagervaror för att sedan lämna dem på avsedd plats. För att roboten ska kunna hitta vägen genom lagret kommer den behöva följa en tejp. I denna förstudie kommer den reglerteknik som krävs för detta projekt att undersökas samt de sökalgoritmer som krävs för att beräkna vägen genom lagret. Frågeställningarna besvaras med hjälp av underlag från vetenskapliga texter.

1.1 Syfte

Syftet med denna förstudie är att undersöka vilken reglerteknik som krävs för att lagerroboten ska kunna utföra sina uppdrag och hur de ska implementeras samt hitta metoder för att beräkna kortaste vägen genom lagret.

2 PROBLEMFORMULERING

Frågeställningarna nedan är det som ämnas besvaras i förstudien:

- Vilka övergripande styrmoder är nödvändiga för att roboten ska kunna utföra sitt uppdrag?
- Hur kan man reglera roboten i de olika styrmoderna?
- Hur koordineras servona i en arm för att kunna styra gripklon till en bestämd position med en mjuk rörelse?

3 BANA OCH RÖRELSEPLANERING

Innan roboten börjar åka kommer dimensioner på lagermiljön samt var varorna finns att anges och därefter ska roboten i autonomt läge beräkna den kortaste vägen genom lagret och sedan beräkna ny väg om hinder påträffas. I fallet där det finns fler än en vara att hämta upp kan roboten antingen åka och hämta varorna för sig eller hämta alla i samma körning och placera dem i någon form av korg som sitter på robotplattformen. Detta kommer beslutas om i ett senare skede och därför utreds här den kortaste vägen i båda fallen.

3.1 Algoritm för att hämta varorna en och en

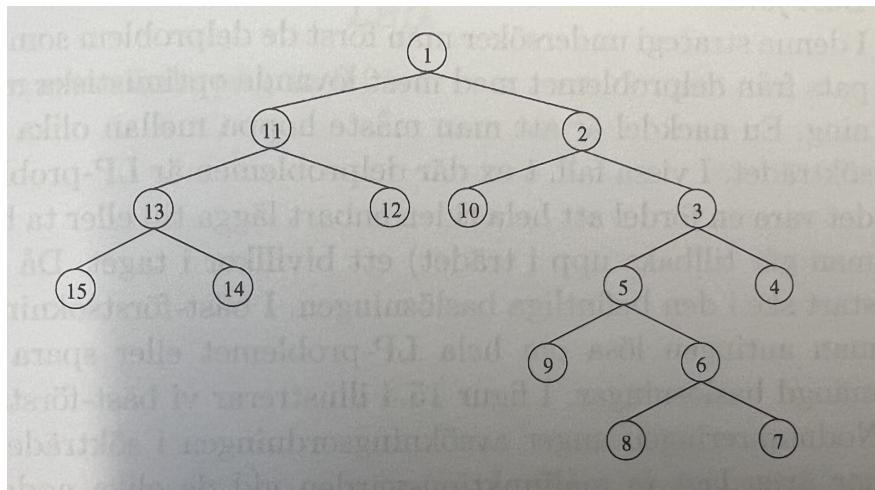
I fallet att varorna ska hämtas var för sig behöver snabbaste vägen till nästa vara beräknas från robotens startposition, vid varuavlämningen i ett av lagrens hörn, alternativt vid påträffat hinder.

Ett sätt att gå tillväga är att ange alla korsningar som noder och vägen mellan dem som kanter där alla vägar mellan två närliggande noder kostar lika mycket.

Noder med varor kommer placeras mellan två korsningsnoder, med samma avstånd till båda. Då blir avståndet mellan varunod och korsningsnod kortare än mellan två korsningar, men eftersom att den placeras i mitten spelar den skillnaden ingen roll. Dessa noder kan sedan ritas upp som ett träd och därmed kan trädsökning användas. Det finns många olika sätt att genomföra en trädsökning som är olika effektiva beroende på hur trädet ser ut och vad syftet med sökningen är [1].

3.1.1 Djupet-Först-Sökning

Djupet-Först-Sökning börjar med att först söka i en gren så långt som möjligt (exempelvis alltid till höger) tills en lövnod nås. Om målet inte hittats stegar den tillbaka och fortsätter utforska nästa möjliga gren på samma sätt [1]. Detta säkerställer att alla möjliga vägar genomsöks tills målet hittats, se figur 1.



Figur 1: Exempel på träddiagram för Djupet-Först-Sökning [1].

Denna metod fungerar bra då en väg behöver hittas snabbt men som då inte nödvändigtvis är den kortaste vägen. I figur 2 visas pseudokoden för Djupet-Först-Sökning.

Algoritm 2: En DFS algoritm.

Input: En GrannListad graf $U = \{n_1, n_2, \dots, n_n\}$ av noder,
en sökt nod *soughtNode*

Output: Den kortaste vägen A till mål noden. (Om det inte
finns någon väg till mål noden så, $A = \emptyset$).

```

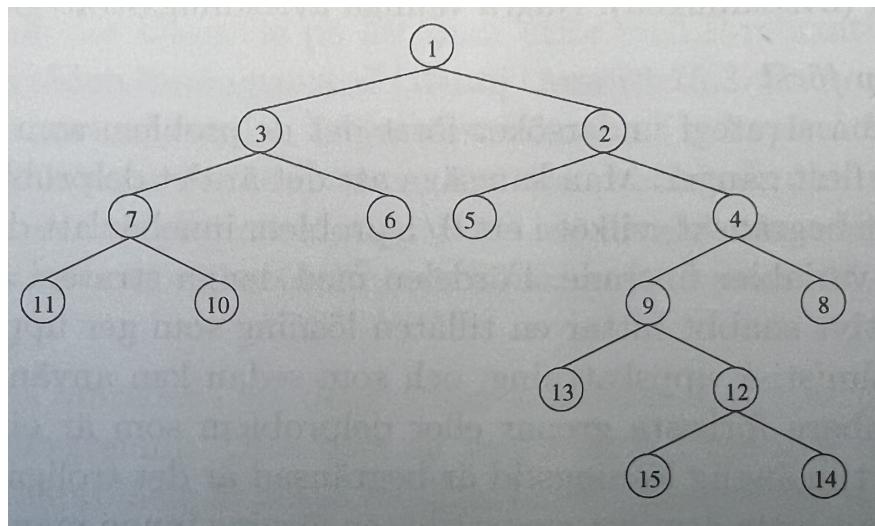
DFS( $U, soughtNode$ )
(1)   Stack pathTo
(2)   node =  $U.root$ 
(3)   while endNotReached
(4)     if node == soughtNode
(5)       return pathTo
(6)     else
(7)       foreach node.neighbours
(8)         if !neighbour.visited
(9)           neighbour.visited = true
(10)          path.add(neighbour)
(11)          neighbourFound = true
(12)          break
(13)        if neighbourFound != true
(14)          node = path.pop()
(15)        if node == root
(16)          endNotReached = false
(17)    return  $\emptyset$ 

```

Figur 2: Pseudokod Djupet-Först-Sökning [2].

3.1.2 Bredden-Först-Sökning

Bredden-Först-Sökning är en algoritm som används för att hitta kortaste vägen från en nod till en annan genom att först undersöka alla delproblem på samma nivå i trädet, för att sedan ta ett steg ner enligt figur 3 [1].



Figur 3: Exempel på träddiagram för Bredden-Först-Sökning [1].

I figur 4 visas pseudokoden för Bredden-Först-Sökning.

Algoritm 1: En BFS algoritm.

Input: En Grannlistad graf $U = \{n_1, n_2, \dots, n_n\}$ av noder,
 en sökt nod *soughtNode*

Output: Den kortaste vägen A till målnoden. (Om det inte
 finns någon väg till målnoden så, $A = \emptyset$).

$\text{BFS}(U, \text{soughtNode})$

```

(1)   Queue queue
(2)   queue.add(U.root)
(3)   while endNotReached
(4)     node = queue.pop() if node == soughtNode
(5)       return (pathTo(node))
(6)     else
(7)       foreach node.neighbours
(8)         if !neighbour.visited
(9)           neighbour.visited = true
(10)          queue.add(neighbour)
(11)        if queue.size() == 0
(12)          endNotReached = false
(13)    return  $\emptyset$ 

```

Figur 4: Pseudokod Bredden-Först-Sökning [2].

3.2 Algoritm för att hämta alla varor i en färd

Ska alla varor hämtas i en tur går problemet istället att formulera som ett handelsresandeproblem. Då lagret beskrivs som ett rutnät där alla avstånd är lika långa blir det en väldigt förenklad variant av den klassiska modellen.

Om målet är att alltid hitta den kortaste vägen mellan alla noder så är det säkraste sättet att studera alla möjliga rutter och hitta den med kortast längd [3]. Detta är väldigt tidskrävande och för större system blir det snabbt väldigt beräkningstungt. Därför kan det vara relevant att istället studera en girig algoritm, eller annan heuristik.

Den lättaste giriga heuristiken att tillämpa kallas nearest neighbor [3]. Den går ut på att roboten alltid väljer att ta vägen till närmaste noden som inte redan besökts. När alla noder har besöks återvänder roboten till startpunkten. Denna algoritm är väldigt billig, men oftast får inte den optimala rutten. Däremot är rutten oftast mycket bättre än godtycklig slumpad rutt.

En annan relevant algoritm är Held-Karp-algoritmen [4]. Som när alla möjliga rutter jämförs fås här den optimala lösningen, men den är betydligt mer effektiv för medelstora system. Först beräknas avstånden mellan alla punkter som vill besökas. Sedan går systematiskt alla delmängder av noder igenom. Först beräknas totala längden mellan alla

kombinationer av tre noder, sedan läggs en fjärde nod till, och så fortsätter det tills alla noder är tillagda. Då är bästa rutten hittad och problemet är löst. Som märks av processen blir det snabbt väldigt många beräkningar om det finns många noder att besöka, men i fallen med medelstora system (som detta projekt troligen är) bör metoden vara snabb nog.

4 SPÅRFÖLJNING

För att roboten ska kunna följa tejpen på ett stabilt sätt, utan att till exempel oscillera fram och tillbaka, behöver reglering med återkoppling användas. Det finns olika typer av reglering med återkoppling som används i olika fall [5].

I PID-reglering (Proportional-Integral-Derivative) ges insignalen $u(t)$ baserat på avvikelsen $e(t)$ mellan önskad utsignal $r(t)$ och veriktig utsignal $y(t)$, alltså $e(t) = r(t) - y(t)$ [6], även kallat återkoppling. Formeln för PID-reglering ser ut som följande:

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t)$$

De olika termerna i ekvationen påverkar regleringen på olika sätt, vilket därmed kan styras genom att ange olika värden på K-variablerna. Den första termen $K_p e(t)$ kallas för den proportionella delen och är, precis som det låter, proportionell mot felet $e(t)$. Därmed ger ett större fel en större justering vilket gör att den kan reagera snabbt. Det går dock inte att enbart använda sig av proportionell reglering, då det kan resultera i ett stationärt fel, det vill säga att algoritmen låser in sig på ett värde som skiljer sig från det önskade värdet. Den andra termen $K_I \int_0^t e(\tau) d\tau$ kallas för den integrerande delen. Den tar i stället hänsyn till tidigare fel och kan på så sätt eliminera det stationära felet. Den tredje termen $K_D \frac{d}{dt} e(t)$ tar, med hjälp av en deriverande faktor, hänsyn till hur snabbt avvikelsen förändras och kan på så sätt dämpa snabba variationer. Nedan följer ett exempel på pseudokod för PID-reglering.

Algorithm 1 Pseudokod Regleralgoritm

Input: $e(t)$

▷ Felsignal

Output: $u(t)$

▷ Utsignal

Initiera variabler: $förra_felet = 0$ K_P = proportionell konstant

▷ Ställ in konstanter (bestäms via testning)

 K_D = deriverande konstant K_I = integrerande konstant Δt = tiden sedan förra regleringen**procedure** PID_REGULATOR($e(t)$) $proportionell_term = K_P * e(t)$ $deriverande_term = K_D * \left(\frac{e(t) - förra_felet}{\Delta t} \right)$

▷ Enkel approximation av derivatan

 $integrerande_term = K_I * Integralapprox.$

▷ Ingen specifik approximation implementeras här

 $u(t) = proportionell_term + deriverande_term + integrerande_term$ $förra_felet = e(t)$ **return** $u(t)$

▷ Returnerar kontrollsignalen

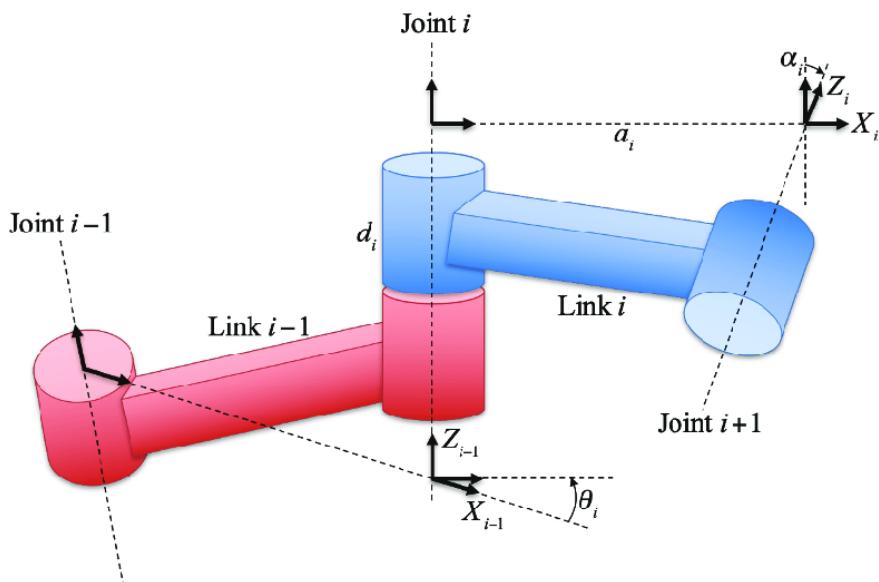
end procedure

5 STYRNING AV ROBOTARMEN

I projektet kommer en robotarm av typen PhantomX reactor användas. Med rotation av basen och "handleden" inräknat har den 5 frihetsgrader, själva armen har då tre vridningspunkter. Servona som sitter i armen är av typen Dynamixel AX-12A. För att beräkna vilka vinklar servona ska ställas in på används invers kinematik. Den typen av problem kan i vissa fall lösas analytiskt, men ibland måste numeriska metoder användas [7].

För att lösa invers kinematik måste armens modelleras och det brukar göras med Denavit-Hartenberg (D-H) parametrar [8]. Parametrarna beskriver varje länks egenskaper, vilket tillsammans ger en bra beskrivning av hela armen. De parametrar som ingår beskrivs nedan, och visualiseras i figur 5:

- a_i : Länklängd, längden av armen efter vridningspunkten
- α_i : Länkvridning, skillnaden i två på varandra följande länkars vridningsaxel.
- d_i : Länkförskjutning
- θ_i : Länkvinkel, specifika vinkeln servon är inställt på, ändras beroende på armens specifika konfiguration.



Figur 5: Visualisering av Denavit-Hartenberg parametrarna [9]

I de flesta fall är en analytisk lösning bäst, om det går att hitta en. Beroende på hur komplicerad robotarmen är och framförallt hur många frihetsgrader den har, kan en analytisk lösning vara svår att härleda. Om armen skulle vara redundant, vilket betyder att den har extra frihetsgrader frihetsgrader, kan en analytisk lösning också vara extra svår att tillämpa. Detta då det kan finnas många olika sätt, i vissa fall oändligt många, att orientera armen så att gripklon når en specifik position.

En annan lösning som bör vara mycket lättare att implementera, men som inte är lika allmän, hade varit att hårdkoda in servonas positioner. Om avståndet från mittlinjen till objektet som ska plockas upp är känt och konstant, så borde armen kunna göra exakt samma rörelse för att plocka upp föremålet varje gång. Den enda riktiga nackdelen med detta är om det finns osäkerheter i robotplattformens position när den ställer sig för upplockning, men om den styrningen är bra nog är det här ett alternativ.

En annan utmaning är att få armen att styras med en mjuk rörelse. Servona som används i projektet, Dynamixel AX-12A, har mycket funktionalitet både för att styra och läsa av relevant mätdata [10]. Just för att få en mjuk och enhetlig kontroll av armen går det att ställa in servons hastighet. För en mjuk och koordinerad rörelse går det att bestämma servonas hastigheter utifrån hur långt den ska förflyttas. På så sätt går det att få så att alla servon börjar röra sig samtidigt, och når sin önskade position samtidigt. Till exempel går det, om en maximal vinkelhastighet bestäms, att sätta hastigheten för den servon som ska förflyttas längst till den hastigheten, och sedan sätta hastigheterna på resterande servon till den hastigheten multiplicerat med en kvot av servonas respektive kommande förflyttning.

6 DISKUSSION OCH SLUTSATSER

I detta avsnitt diskuteras de metoder som presenterats och frågeställningarna besvaras.

6.1 Nödvändiga styrmoder

Den styrmoder som behövs för att roboten ska kunna utföra sina uppdrag är spårföljning, så att roboten följer tejen och vägnavigering så att roboten hittar i lagret.

6.2 Följning av tejp

I regleringen för tejpföljning kommer enbart PD (Proportionell respektive Deriverande reglering) att behöva användas. Felet för tejpföljning fås från tejsensorn på bilen, och är ett mått på hur långt från mitten av bilen tejen är. Ett fel $e(t) = 0$ innebär att tejen ligger mitt under bilen, och rimligtvis önskas då att bilen fortsätter köra rakt. $u(t)$ kommer att representera hur mycket åt något håll bilen måste svänga, och om insignalen (felet) $e(t) = 0$ kommer självklart utsignalen $u(t) = 0$. I detta fall finns då inget kvarstående reglerfel, och en I del behövs därför inte [5].

Däremot behövs en D del i algoritmen, då den används för att minska oscillationer i regleringen. Oscillationer är ofta ett stort problem om enbart en P del används (framförallt vid höga värden på K_P , dvs snabb reglering) så en deriverande del är här väldigt relevant, hur den hjälper i detta fallet är väldigt tydligt. Derivatan av felet är hur fort felet ändras, och det den termen gör här är att göra svängningen mindre aggressiv om den redan är på väg åt rätt håll, och börja svänga ännu mer om bilen är på väg åt fel håll. Ekvationen som kommer användas blir då den nedan, konstanterna tas lättast fram genom testning.

$$u(t) = K_P e(t) + K_D \frac{d}{dt} e(t)$$

Vägnavigeringen regleras med hjälp av input i form av karta och position på varor samt kostnader för olika vägar.

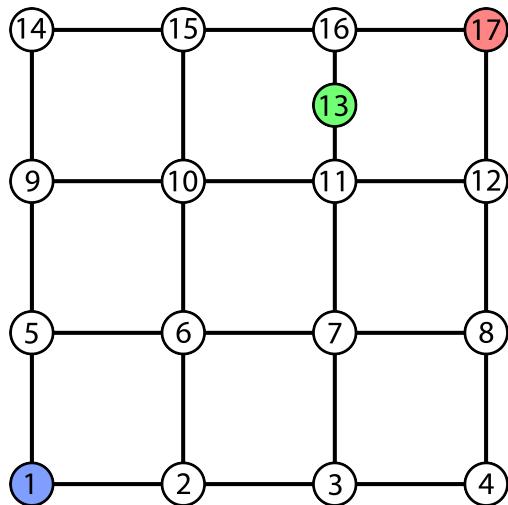
6.3 Styrning av robotarm

För styrning av robotarmen är det lättaste, och helt klart det mest realistiska, att hårdkoda in de servovärden som krävs för att plocka upp föremålet. För det som robotarmen skall användas till i projektet behövs inte mycket mer än så, iallfall inte för det autonoma fallet. Om resten av robotens reglering görs tillräckligt bra kommer avståndet mellan armen och föremålet den ska plocka upp att vara samma varje gång, så att hårdkoda in armens rörelse bör vara en bra lösning. Detta görs lättast genom att ställa upp roboten i positionen där objektet plockas, ställa armen i en position där föremålet nås, och sedan läsa av all relevant information om servonas inställning. Detta går enkelt att göra då servona som används i armen har många inbyggda sensorer som kan läsas av [10].

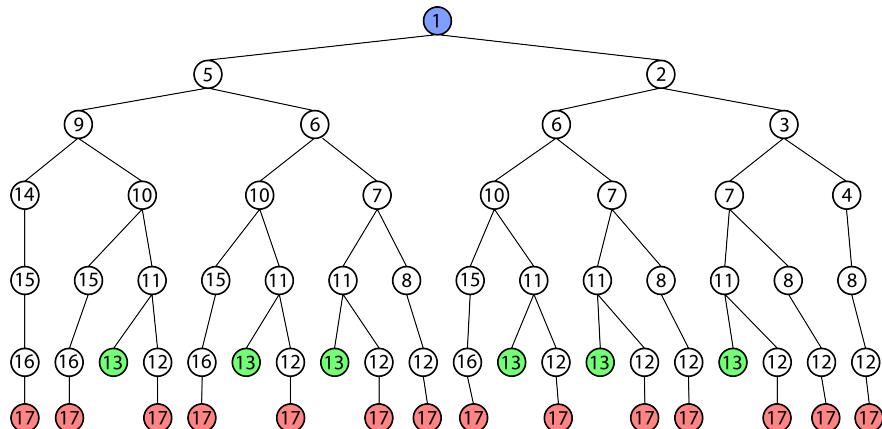
Beroende på hur det implementeras kan manuell styrning av robotarmen bli lite knepigare. En implementering, som nog är lättast för användaren, är om robotarmen tar emot kommandon såsom framåt, bakåt, vänster och liknande. Detta lär dock vara väldigt svårt att implementera, då beräkningarna för hur servona ska röras lär behöva göras med invers kinematik. Detta kan, vilket konstaterades tidigare, vara väldigt komplicerat, framför allt för en robotarm med många ledar. Den mer realistiska, men kanske inte riktigt lika användarvänliga lösningen är att i manuellt läge styra en servo i taget. Detta ger även en större frihet för användaren, men det tar längre tid att ställa in armen i önskad position och är allmänt inte lika intuitivt.

6.4 Bana och rörelseplanering

När position för varan matas in av användaren placeras en nod där avsticket till varan finns. Då robotten väl kommer fram till rätt kant känner den själv av vilken sida om tejpen varan är på. I fallet där varorna hämtas en och en är användningen av Bredden-Först-Sökning mest effektiv för att hitta denna nod då den hittar kortaste vägen och inte bara första bästa som Djupet-Först-Sökning gör. Eftersom lagret är utformat som ett rutnät behöver inte alla vägar ut ur en nod tillåtas för att hitta den snabbaste vägen. Om startpunkten antas vara i nedre vänstra hörnet enligt figur 6 behöver noderna exempelvis endast avsökas uppåt och till höger då ingen snabbare väg kommer kräva steg till vänster eller nedåt. Då fås träd enligt figur 7.

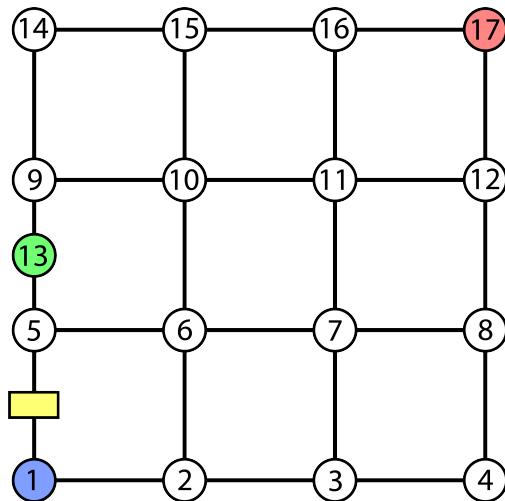


Figur 6: Exempel på lagermiljö med numrerade noder.



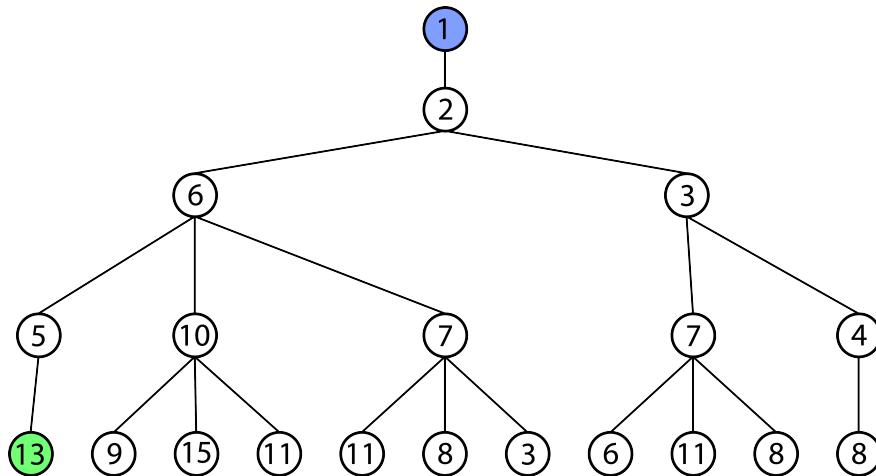
Figur 7: Träddiagram över noder i lagermiljö.

Detta tillsammans med Bredden-Först-Sökning fungerar utmärkt i detta fall, men i verkligheten kommer det även finnas hinder, vilket visas som gul rektangel i figur 8.



Figur 8: Exempel på lagermiljö med numrerade noder och hinder.

Vid påkommot hinder tar sig roboten tillbaka till närmaste nod, beräknar ny väg samt raderar vägen som hindret låg på. Detta innebär att snabbaste vägen kan kräva steg till vänster eller nedåt för att komma till varan. Därför måste efter påkommot hinder, enligt figur 9, alla vägar avsökas.



Figur 9: Träddiagram över lagermiljö med noder och hinder.

I fallet där alla varor hämtas i en körsling finns det några alternativ som skulle vara möjliga att implementera bra. Av de idéer som diskuterades tidigare är den minst användbara nearest neighbor. Den är snabb att beräkna, men eftersom att optimala rutten inte är garanterad så är den tiden som sparas i beräkningen inte mer än tiden som tappas av att roboten åker en suboptimal väg. Däremot hade det nog inte, i de situationer som kommer dyka upp i projektet, varit

en helt orimlig strategi att beräkna alla möjliga kombinationer av upplockningsordningar. Om noderna blir många blir det snabbt väldigt många beräkningar, men även om det är 6 objekt som ska plockas upp så blir det endast 720 kombinationer att beräkna. Detta låter mycket, men till projektet används en Raspberry PI med 1.4 GHz klockfrekvens [11] (vilket betyder att den kan utföra 1.4 miljarder beräkningar per sekund), så även om det skulle bli många beräkningar att göra skulle den extra tiden det tar inte vara jämförbar med den extra tiden en längre väg skulle ta.

Den strategin som känns bäst för att lösa detta problem blir då Help-Karp algoritmen. Den beräknar alla möjliga kombinationer och ger därför alltid den optimala lösningen, men den gör det mer effektivt, framförallt eftersom att den slipper göra samma beräkning flera gånger. Även med denna metod blir det då, som metoden ovan, många beräkningar när antalet noder ökar. Help-Karp metoden är som sagt mer effektiv än den ovan, och blir därför en bra lösning.

7 REFERENSER

REFERENSER

- [1] J. Lundgren, M. Rönnqvist och P. Värbrand, *Optimizeringslära*, 2. utg. Lund, Sweden: Studentlitteratur, 2010, ISBN: 978-91-44-05512-9.
- [2] J. Nordström. "Combinatorial Search." Accessed: 2025-04-07. (2009), URL: <https://www.csc.kth.se/utbildning/kth/kurser/DD2458/popup15/material/oldnotes/combsearch.F4.09.pdf>.
- [3] W3Schools, *Traveling Salesman Problem*, https://www.w3schools.com/dsa/dsa_ref_travelling_salesman.php, Accessed: 2025-02-24, n.d.
- [4] CompGeek, *Held-Karp Algorithm for TSP*, <https://compgeek.co.in/held-karp-algorithm-for-tsp/>, Accessed: 2025-02-24, n.d.
- [5] L. University. "Föreläsning 6 - AVR." Tillgänglig: 24 februari 2025. (2025), URL: https://liuonline.sharepoint.com/sites/Lisam_TSEA56_2025VT_M8/CourseDocuments/Forms/AllItems.aspx?id=%2Fsites%2FLisam%5FTSEA56%5F2025VT%5FM8%2FCourseDocuments%2FProjektmodul%2FF%C3%B6rel%C3%A4snigar%2FF%C3%B66%2DAVR%2Epdf&parent=%2Fsites%2FLisam%5FTSEA56%5F2025VT%5FM8%2FCourseDocuments%2FProjektmodul%2FF%C3%B6rel%C3%A4snigar.
- [6] T. Glad och L. Ljung, *Reglerteknik: Grundläggande teori*, 4. utg. Lund, Sweden: Studentlitteratur, 2003, ISBN: 978-91-44-02308-1.
- [7] U. of Illinois - Motion Group, *Inverse Kinematics*, Accessed: 2025-02-22, 2024. URL: <https://motion.cs.illinois.edu/RoboticSystems/InverseKinematics.html>.
- [8] T. Abaas, A. Khleif och M. Abbood, "Kinematics Analysis of 5 DOF Robotic Arm," *Engineering and Technology Journal*, årg. 38, nr 3A, s. 412–422, 2020. DOI: [10.30684/etj.v38i3A.475](https://doi.org/10.30684/etj.v38i3A.475). URL: https://etj.uotechnology.edu.iq/article_168857.html.
- [9] E. Shahabi, P. T. Lin, K.-A. Yang och C.-H. Kuo, *The four classic DH parameters for rigid linkage kinematics*, https://www.researchgate.net/figure/The-four-classic-DH-parameters-for-rigid-linkage-kinematics_fig1_333761139, Figure from the publication Parametrically Modeled DH Table for Soft Robot Kinematics: Case Study for A Soft Gripper", 2019.
- [10] ROBOTIS, *AX-12A e-Manual*, Accessed: 2025-02-23, n.d. URL: <https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/>.
- [11] R. P. Foundation, *Raspberry Pi 3 B+ Product Brief*, Accessed: 2025-04-07, 2018. URL: <https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf>.

8 APPENDIX

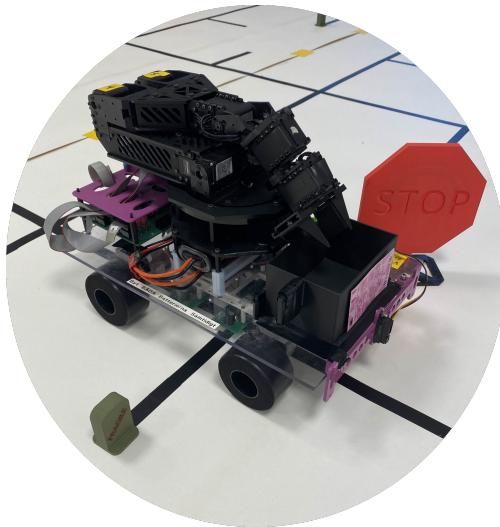
14 ANVÄNDARMANUAL

Användarmanual för en autonom lagerrobot

Grupp 6

6 juni 2025

Version 1.1



Status

Granskad	Ebba Lundberg	2025-06-06
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Inledning	1
2	Hårdvaruinspektion	1
2.1	Strömförsörjning	1
2.2	Kommunikationsinkoppling	2
2.3	Sensorinkoppling	3
2.4	Styrinkoppling	4
3	Gränssnitt	6
4	Manuell styrning	6
5	Autonom styrning	7

DOKUMENTHISTORIK

Version	Datum	Utförda ändringar	Utförda av	Granskad
1.0	2025-05-21	Första version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS
1.1	2025-06-06	Andra version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS

1 INLEDNING

Det här är användarmanualen för grupp 6:s lagerrobot, utvecklad inom kursen TSEA56 våren 2025 på Linköpings universitet. Roboten är designad för att fungera i en känd lagermiljö där den kan navigera mellan olika plockstationer och samla upp varor samtidigt som den undviker eventuella hinder som kan uppstå. Syftet med roboten är att automatisera lagerhantering genom att effektivt plocka och transportera varor, vilket kan bidra till ökad produktivitet och minskad arbetsbelastning för personalen.

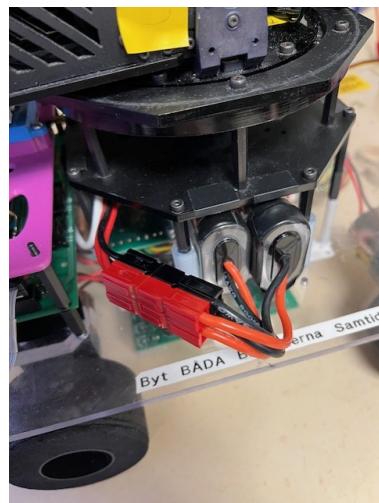
I denna manual beskrivs hur du installerar och startar roboten, hur du använder dess manuella och autonoma funktioner, samt hur du tolkar informationen i användargränssnittet. För att använda roboten krävs en dator med Python installerat samt en aktiv Bluetooth-uppkoppling till roboten.

2 HÅRDVARUINSPEKTION

En okulär inspektion av roboten behövs innan den används. Detta är för att säkerställa att IR-sensorn och linjesensorerna ej är blockerade eller smutsiga samt att inga tydliga skador, som till exempel sprickor, finns på roboten. Utöver detta behöver flera kablar anslutas, vilket gås igenom i kommande delkapitel.

2.1 Strömförsörjning

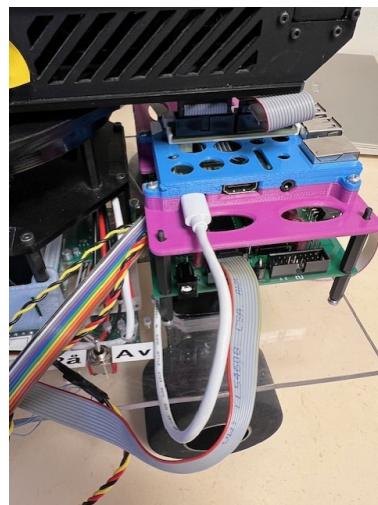
Roboten får ström via två stycken laddbara batterier som kopplas in via en svart-röd sladd i roboten, se figur 1.



Figur 1: Inkoppling av batteri.

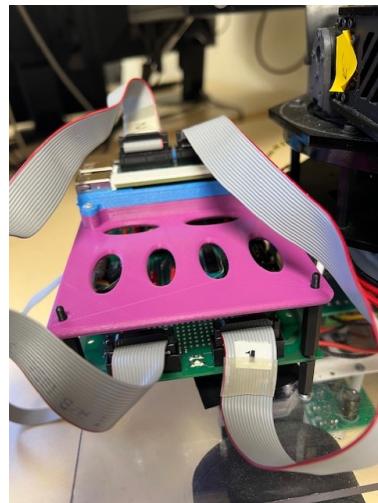
2.2 Kommunikationsinkoppling

Kommunikationssystemet består av en Raspberry Pi. Den får ström via en mikro-USB som är inkopplad i roboten, se figur 2



Figur 2: Ström för Raspberry Pi.

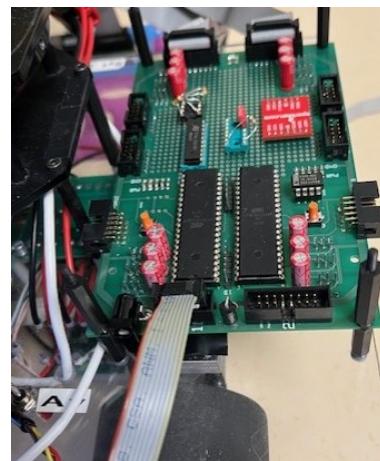
Till Raspberryn behövs två stycken 16-pinnars flatkablar. De kopplas mellan Raspberryn och mikrodatorerna för senorenheten och styrenheten och utgör SPI-bussen, se figur 3.



Figur 3: SPI-koppling mellan Raspberry Pi och ATmega.

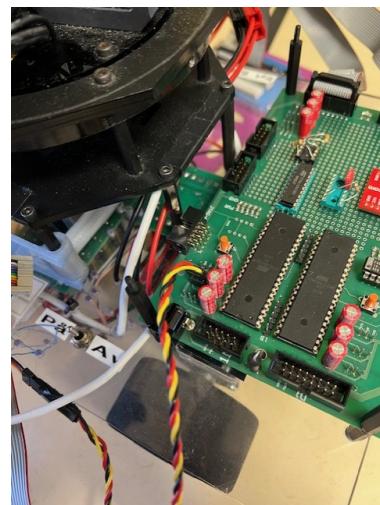
2.3 Sensorinkoppling

Sensorenheten består av en ATmega1284P samt en avståndssensor, två linjesensorer och ett gyroskop som är färdiginkopplat. Virkortet får ström via en 10-polig flatkabel som kopplas in i IDC 1, se figur 4. Denna koppling ger ström till mikrodatorerna för sensorenheten och styrenheten samt avståndssensorn och gyroskopet.



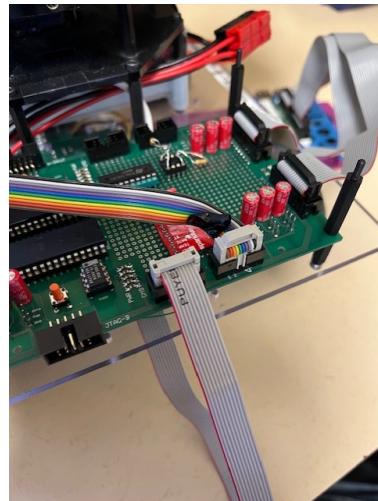
Figur 4: Strömanslutning till virkortet.

Avståndssensorn kopplas in i mikrodatorn med en 3-pinkabel på en specifik port på virkortet, se figur 5



Figur 5: Inkoppling av IR-sensor.

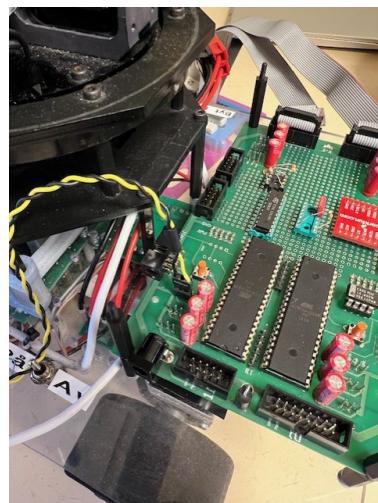
Jord och spänning är föranslutna till linjesensorerna. Den främre linjesensorn ansluts med en 10-polig regnbågsfärgad flatkabel till IDC 4 och den grå bakre ansluts likadant till IDC 3, se figur 6.



Figur 6: Inkoppling av linjesensorerna.

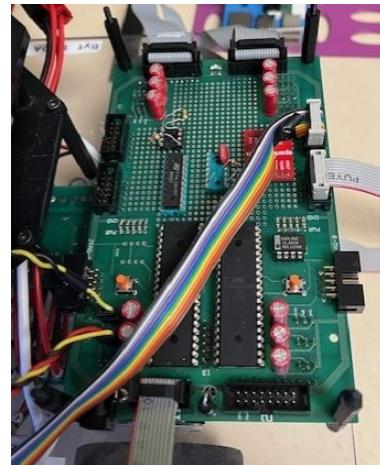
2.4 Styrinkoppling

Robotarmen kopplas in på liknande sätt som avståndssensorn med en 3-pinkabel från armen in i specifik port på virkortet, se figur 7.



Figur 7: Inkoppling av robotarmen.

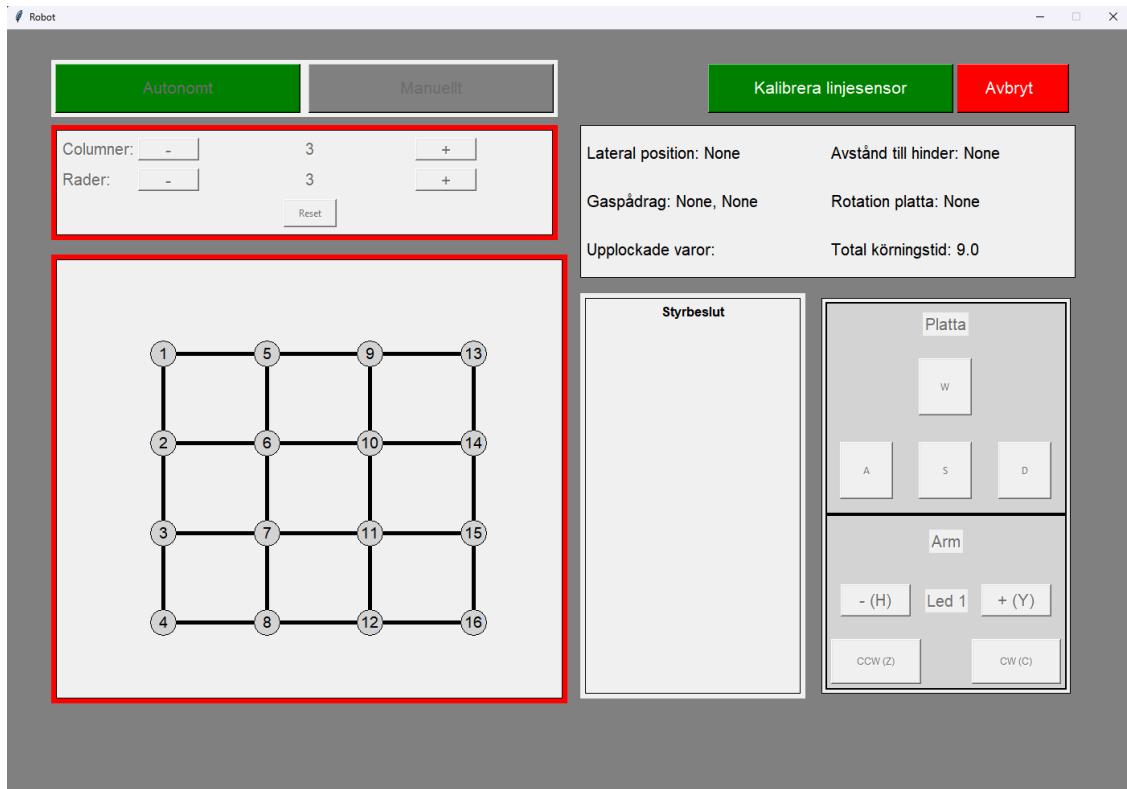
I figur 8 visas alla kopplingar in i virkortet.



Figur 8: Allt inkopplat i virkortet.

3 GRÄNSSNITT

För att styra lagerroboten manuellt eller starta det autonoma uppdraget så måste det göras genom gränssnittet. Börja med att starta programmet och roboten. Detta kräver att Python är installerat då huvudfilen är en Pythonfil. Därefter kan man koppla PC:n till roboten via Bluetooth, anslut då till enheten som heter "raspberrypi". I gränssnittet, se figur 9, kan användaren läsa av: planerad väg, lagermiljön och sensordata.



Figur 9: Användargränssnittet i autonomt läge efter att körning startats.

4 MANUELL STYRNING

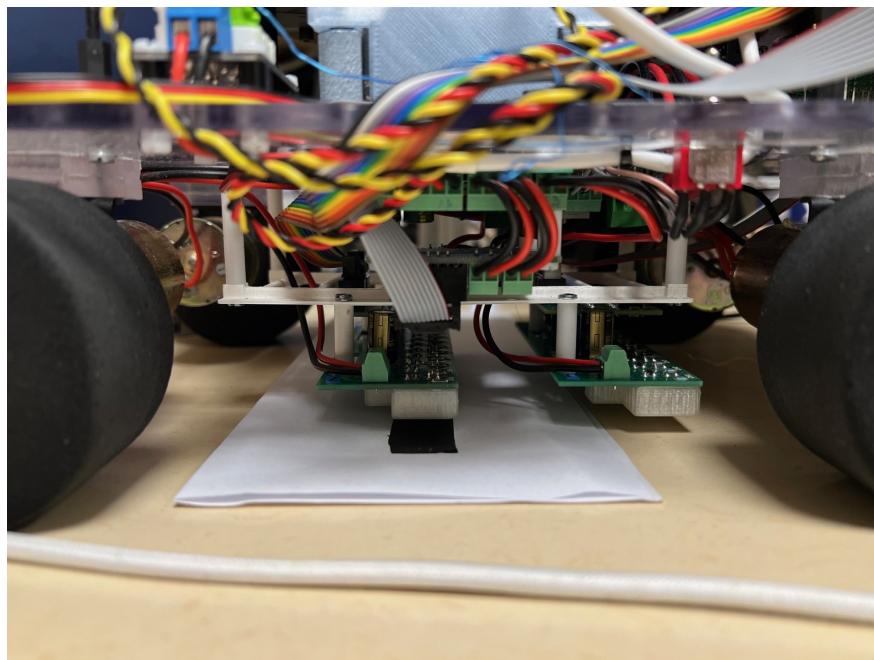
Vid manuell styrning kontrolleras robotplattformen och robotarmen direkt av användaren. När gränssnittet är öppet finns ett antal kommandon tillgängliga som skickas genom att trycka på olika knappar, se tabell 1 för alla tillgängliga kommandon.

Knapp	Kommando
W	Kör rakt
A	Rotera vänster
S	Kör bakåt
D	Rotera höger
W + A	Sväng vänster
W + D	Sväng höger
Y / H	Välj armservo
Z / C	Ändrar ledens vinkel
Starta data	Startar datainsamling
Kalibrera	Kalibrera linjesensorerna
Autonom	Starta autonomt uppdrag

Tabell 1: Alla knappar och deras korresponderade kommandon.

5 AUTONOM STYRNING

Autonom styrning startas via knappen "Autonom" i GUI:n, se figur 9. Robotens linjesensorer behöver kalibreras. Det görs via knappen "Kalibrera" i GUI:n då hela den främre linjesensorn ligger över tejp, se figur 10. Markera därefter i användargränssnittet mellan vilka korsningar där varor som ska hämtas ligger. Placera sist roboten i återhämtningstationen med riktning mot lagret och starta det autonoma uppdraget. När roboten är klar kommer den att lämna av alla varor i återhämtningstationen och stanna.



Figur 10: Illustration över hur linjesensorn ska vara placerad vid kalibrering.

15 TEKNISK DOKUMENTATION

Teknisk dokumentation

Grupp 6

6 juni 2025

Version 1.1



Status

Granskad	Ebba Lundberg	2025-06-06
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno773@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Inledning	1
2	Produkten	1
3	Bakgrund	2
3.1	Sensorenheten	2
3.2	Styrenheten	2
3.3	Kommunikationssystemen	2
3.4	PC	2
4	Systemet	3
5	Modulerna	3
5.1	Sensorenheten	4
5.2	Styrenheten	7
5.3	Kommunikationssystemen	9
5.4	PC	14
6	Slutsatser	15
7	Appendix	17
7.1	Grafer	17
7.2	Kopplingsscheman	18
7.3	Exempelkod	19
7.4	Tabeller	31
7.5	Övrigt	33

DOKUMENTHISTORIK

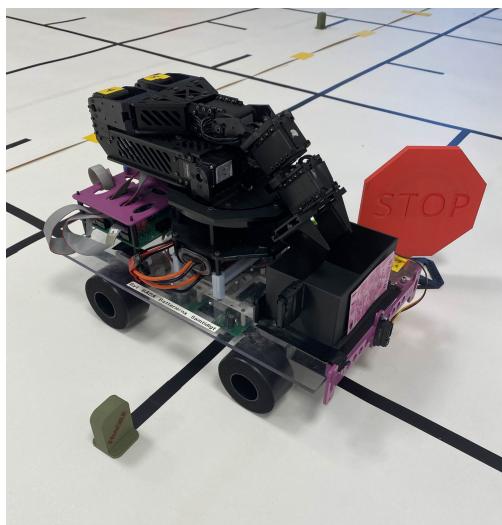
Version	Datum	Utförda ändringar	Utförda av	Granskad
1.0	2025-05-21	Första version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS
1.1	2025-06-06	Andra version	LF, EL, AN, SR, IS, LS	LF, EL, AN, SR, IS, LS

1 INLEDNING

Detta dokument är skrivet för att kunna användas som konstruktionsunderlag vid replikering av lagerroboten samt för att underlätta underhåll och felsökning av både hård- och mjukvara. Allt material som projektet lånat kommer att vara listat tillsammans med beskrivningar av både det översiktliga systemet och alla delsystem i detalj.

2 PRODUKTEN

Produkten som har tagits fram är en lagerrobot, se Figur 1, bestående av en robotplattform och en robotarm med en gripklo. Robotten kan navigera i en känd lagermiljö med hinder och specifika plockstationer där varor är placerade i förväg. Vid plockstationerna ska varorna plockas upp med hjälp av robotarmen för att sedan lämnas vid en utlämningsstation. Lagerroboten använder ett flertal sensorer för att utföra linjeföljning, korrekta svängar och kunna detektera hinder. Daten skickas via en SPI-buss till kommunikationsenheten som också skickar kommandon till styrenheten. På en separat PC kan användaren kontrollera robotten genom ett gränssnitt.



Figur 1: Lagerrobot med robotarm och robotplattform.

3 BAKGRUND

I följande kapitel ges en kort beskrivning av de komponenter som har använts i respektive delsystem.

3.1 Sensorenheten

Sensorenheten är försedd med tre olika typer av sensorer, två specifika algoritmer som används i mjukvaran för särskilda beräkningar och en ATmega1284P mikrokontroller. En IR-sensor används för att detektera hinder, två reflexsensormoduler används för linjeföljning och ett gyroskop är installerat så att roboten ska kunna utföra mer precisa svängningar. Eftersom flyttalsberäkning är långsamma på en ATmega1284P används istället linjärinterpolation för att snabbare beräkna utdata från avståndssensorn. För att kunna avgöra om roboten är till höger eller vänster om linjen den följer, så genomförs en enkel tyngdpunktsberäkning för att ta fram data på avvikelsen mitten och mellan reflexsensormodulerna.

3.2 Styrenheten

Styrenheten ansvarar för alla förflyttningar och rörelser som roboten kan utföra, och består av en ATmega1284P mikrodator. Det styrenheten styr är fyra hjul konfigurerade som två hjulpar och en robotarm med totalt 8 servon. Styrenheten gör inte så många beräkningar själv, utan nästan alla beräkningar görs på andra enheter. Styrenheten tar bara emot kommandon och utför dem. De kommandona kan antingen vara manuell styrning som kommer direkt från användaren via användargränssnittet, eller styrbeslut tagna under autonom körning baserat på sensorvärden och snabbaste-vägen-algoritmen.

3.3 Kommunikationsenheten

Kommunikationsenheten sköter all intern och extern kommunikation. Den består till största del av en enkortsdator av typen Raspberry Pi. Den ska ta emot data från styr- och sensorenheten via virkort och överför sedan information till PC:n via bluetooth-anslutning för att kartlägga var roboten befinner sig. Kommunikationsenheten ska även kunna ta emot data via bluetooth från PC:n avsedd för styrenheten för manuell styrning. Vid autonom körning ska kommunikationsenheten, med hjälp av data från resterande enheter, kunna ta beslut om kortaste väg. Kommunikationen mellan Raspberry Pi:n och sensor- samt styrenheten kommer genomföras via en SPI-buss.

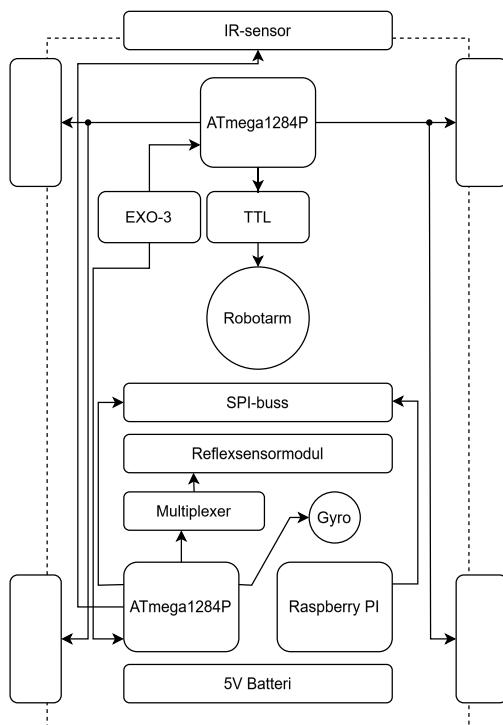
3.4 PC

Denna enhet kommer endast vara en av gruppens bärbara persondator och har som huvuduppgift att kunna föra kommunikation med Raspberry Pi:n. På PC:n används ett användargränssnitt, GUI, för att föra all kommunikation. Via

denna kontrolleras manuell styrning och eventuell data från sensor- och styrenheten kommer visas. Det är även här som användaren kan välja banans storlek och placering av varor som i sin tur skickas över Bluetooth-anslutning till kommunikationsenheten för beräkning av kortaste väg.

4 SYSTEMET

Systemet består av en sensor-, kommunikations- och styrenhet. Dessa kommer att vara installerade på roboten som illustreras i Figur 2. Följande kapitel beskriver respektive delsystem i detalj.



Figur 2: Blockschema över systemet.

5 MODULERNA

Följande kapitel beskriver detaljerat hur alla delsystem är designade och implementerade i det större systemet.

5.1 Sensorenheten

Sensorenhetens uppgift är att samla in data som sedan skickas vidare till kommunikationsenheten. Sensordatana kommer att skickas kontinuerligt från samtliga sensorer vilket utgör grunden till de styrbeslut som tas samt kartläggningen av lagermiljön.

5.1.1 Komponenter och programpaket

I Tabell 1 listas de komponenter som ingår i sensorenheten.

Tabell 1: Lista över komponenter för sensorenheten.

Komponent	Egenskap	Antal
Mikrodator	ATmega1284P	1
Emulator	JTAG ICE 3	1
Oscillator	EXO-3	1
Avståndssensor	GP2Y0A21, 10-80cm	1
Reflexsensormodul		2
Multiplexer	För reflexsensormodulen	2
Gyroskop	MLX90609	1
Resistor	18 kΩ, lågpassfilter	1
Kondensator	100 nF, lågpassfilter	1

I Tabell 2 listas de programpaket som användes i sensorenheten.

Tabell 2: Lista över programpaket för sensorenheten.

Programpaket	Egenskap
avr/io.h	Konfigurerar I/O-portarna
avr/interrupt.h	Avbrott
math.h	Matematiska operationer
util/delay.h	Funktioner för tidsfördröjningar i koden

5.1.2 ATmega1284P

Funktionaliteten hos sensorenheten förutsätter att ATmega1284P mikrodatorn är korrekt ansluten till sensorerna och att dessa initieras korrekt i mjukvaran.

Figur 6 (se Appendix A.2) visar hur alla sensorer är uppkopplade till ATmega1284P. Mikrodatorn har två multiplexar inkopplade som styr vardera reflexsensormodul. Genom dessa kan ATmega1284P kontrollera vilken reflexsensor som är aktiv i modulen och läsa av dess analoga utsignal. Ytterligare en EXO-3 är inkopplad för att styra mikrodatorns klocka, då dess egna interna klocka inte är tillräckligt stabil [1]. IR-sensorn GP2Y0A21:s analoga utsignal lågpass-

filtreras innan den kopplas till ATmega1284P:s A/D-omvandlare. Sist är mikrodatorn kopplad till SPI-bussen genom dess SS, MISO och SCK-portar.

Sensorerna och delsystem inom modulen initieras som i Exempelkod 6 (se Appendix A.3). Alla sensorerna använder extern referensspänning och är inställda att vänsterjustera utdata vilket möjliggör att enklare spara endast de 8 mest signifika bitarna. Den enda skillnaden i deras ADMUX-inställningar är vilken port A/D-omvandlaren ska läsa av. ADCSRA är också identiska då förstoraren (*prescaler* på engelska) sätts till 128 och möjliggör aktivering av A/D-omvandlaren. Prescalern valdes till 128 för att mikrodatorns frekvens är 16 MHz, medan A/D-omvandlaren får bäst upplösning mellan 50-200 kHz. Med en prescaler på 128 fås frekvensen på A/D-omvandlaren till 125 kHz. Alla sensorer initieras separat trots att de har många gemensamma inställningar för att underlätta framtida implementeringar av andra sensorer som kan kräva andra inställningar. Andra viktiga delsystem att initiera är SPI-bussen och avbrott som båda måste aktiveras innan de kan användas.

5.1.3 *IR-sensor*

IR-sensorn GP2Y0A21 används för att detektera hinder i lagermiljön. Efter att ADMUX är inställt till rätt port görs en A/D-omvandling. För att undvika tunga beräkningar med flyttal används istället linjär interpolation. I sensorns datablad finns en graf som visar vilka analoga spänningsvärdet som motsvarar vilket avstånd, se Figur 5 (Appendix A.1). Det går alltså att uppskatta ett värde y för ett givet x mellan två kända punkter (x_0, y_0) och (x_1, y_1) , genom formeln för linjär interpolation

$$y = y_0 + (x - x_0) \cdot \frac{y_1 - y_0}{x_1 - x_0}, \quad (1)$$

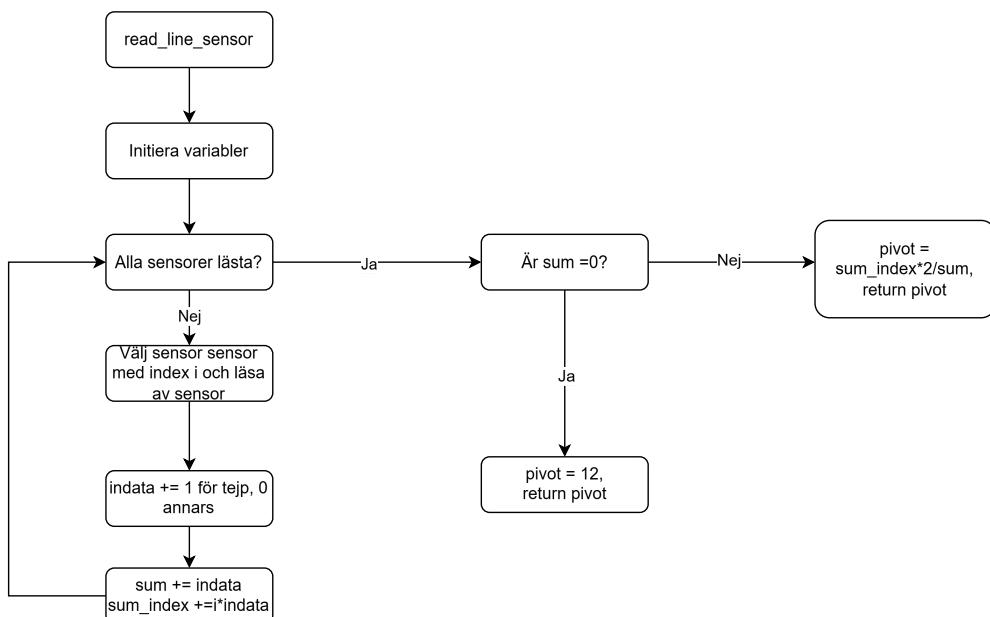
där y är avståndet från avståndssensorn och x är det uppmätta spänningsvärdet. Avläsning av databladet ger värdena (x_0, y_0) och (x_1, y_1) . Avståndet y blir en approximation som ligger mellan y_0 och y_1 . Detta värde sparar sedan i en global variabel som kan skickas till Raspberry Pi:n genom SPI-bussen.

5.1.4 *Reflexsensormoduler*

Reflexsensormoduler behövs för att möjliggöra linjeföljning och detektera korsningar samt plockstationer. En enkel tyngdpunktsberäkning används för att beräkna hur långt ifrån sensorns mittpunkt som linjen är. Tyngdpunkten räknas ut genom

$$k_T = \frac{\sum_k m_k k}{\sum_k m_k}, \quad (2)$$

där k_T och m är tyngdpunkten respektive magnituden som en reflektor detekterar. I Figur 3 illustreras ett flödesschema över tyngdpunktsberäkningen för reflexsensorn.



Figur 3: Flöde för reflexsensor.

Flödesschemat implementerades enligt Exempelkod 1 (se Appendix A.3) där funktionen `read_line_sensor` returnerar ett pivotelement. Om summan av alla reflexsensorerna är noll, alltså att ingen sensor har sett tejp, så blir pivot 12, så att roboten ska åka rakt framåt. För att möjliggöra beräkningar med decimalprecision utan att pivot sparas som ett flyttal i mikrodatorn multipliceras den med 2. Då kan kommunikationsenheten dividera värdet den får med 2 för att få tillbaka det egentliga värdet. Två stycken reflexsensorer används för att underlätta regleringen av roboten. Mer information om regleringen finns i avsnitt 5.3.7.

5.1.5 Gyroskop

Gyrot har som uppgift att mäta 90 eller 180 graders svängar och kan aktiveras av kommunikationsenheten vid två tillfällen. Antingen när den främre reflexsensorn detekterar en korsning eller plockstation och när IR-sensorn detekterar ett hinder. I båda situationerna startar kommunikationenhetens timer via SPI-bussen som kommer att göra ett avbrott var 10 ms. Då utför gyrot en avläsning och uppdaterar datan som kommer att skickas till bussen. Kommunikationenhetens stänger av gyrot när gyrodatan har kommit upp i ett visst hårdkodat värde, i detta fall 60 och 120, som motsvarar 90 eller 180 grader.

5.2 Styrenheten

Styrenhetens uppgift är att på mikrodatorn ta emot instruktioner och data från kommunikationsenheten via SPI-bussen, och utifrån det kontrollera robotens olika motorer och servon. Det som styrs av styrenheten är två hjulpar och en robotarm bestående av 8 servon.

5.2.1 Komponenter och programpaket

I Tabell 3 listas de komponenter som ingår i styrenheten.

Tabell 3: Lista över komponenter för styrenheten.

Komponent	Egenskap	Antal
Mikrodator	ATmega1284P	1
Robotarm	PhantomX Reactor	1
Hjul		4
Oscillator	EXO3	1
Resistor	10 kΩ	1

I Tabell 4 listas de programpaket som användes i styrenheten.

Tabell 4: Lista över programpaket för sensorenheten.

Programpaket	Egenskap
avr/io.h	Konfiguerar I/O-portarna
avr/interrupt.h	Avbrott
math.h	Matematiska operationer
util/delay.h	Funktioner för tidsfördröjningar i koden

5.2.2 Styrning av robotplattformen

I styrenhetens mikrodator finns kod implementerad för styrning av robotplattformen, både manuellt och autonomt. Hur de olika styrmoderna fungerar på just styrenhetens mikrodator är enligt samma princip, skillnaden är till stor del bara vilka instruktioner som skickas från kommunikationenheten, och om det är en människa eller en regleralgoritm som tar beslutet.

På robotplattformen styrs dess respektive hjulpar separat. För att styra hjulparen används en kombination av PWM och DIR-signaler, där PWM-signalerna styr hjulparets fart och DIR styr hjulens riktning. PWM-signaler är periodiska signaler som varierar mellan högt och lågt, och ju större andel av tiden som signalen är hög desto snabbare rullar hjulen. DIR-signalen är binär, så 0 respektive 1 ger olika riktningar.

All kommunikation till styrenheten sker via SPI. När data kommer på bussen triggas ett avbrott, och där behandlas den data som skickats. Ibland skickas bara ett kommando som representerar en handling, exempel på det är fram, bak,

rotera höger och liknande. För mer avancerade kommandon, som det som används vid autonom linjeföljning, kan även extra data skickas med som ska användas som argument till funktionen. För detta används en state machine. Genom att byta till ett specifikt state när ett bestämt kommando skickas kan mikrodatorn sen vänta in så mycket data den behöver, till exempel tar mikrodatorn emot tre bytes som tillsammans representerar utsignalen från regleralgoritmen när linjeföljning är aktivt.

I styrenhetens mikrodators mainloop finns det en lång rad med if-satser, och beroende på vad senaste instruktionen från kommunikationsenheten var körs olika funktioner, som styr roboten på olika sätt. Både autonom styrning och manuell styrning fungerar på detta sätt, och skillnaderna beskrivs nedan.

MANUELL STYRNING AV ROBOTPLATTFORMEN

Under manuell styrning kommer alla instruktioner via kommunikationsenheten från användargränssnittet. Beroende på användarens knapptryck skickas olika kommandon från PC till kommunikationsenheten och sen vidare till styrenheten. Mer om vilket kommando som gör vad beskrivs i kapitlet om kommunikationsenheten.

AUTONOM STYRNING AV ROBOTPLATTFORMEN

Under autonom styrning kommer alla instruktioner från kommunikationsenheten att bero på sensorvärdet och styrbeslutet från kortaste-vägen-algoritmen. På robotplattformen går, som tidigare nämnt, att styra hjulpare separat. Genom att köra de två hjulparena olika snabbt går det då att få roboten att svänga, detta kallas differentialstyrning. För linjeföljningen sätts först en hastighet, och sedan ökas och minskas höger respektive vänster hjulparens hastighet med regleralgoritmens utsignal, se Exempelkod 4. Detta ger att plattformen svänger olika mycket beroende på hur stort reglerfelet är, vilket gör att roboten kan följa linjer. Utöver detta kan robotplattformen även rotera, 90 grader i korsningar och 180 grader om hinder stöts på. Dessa rotationer använder samma funktioner som rotationer i manuellt läge, men hur länge de körs styrs med hjälp av gyrodata från sensorenheten.

5.2.3 Styrning av robotarmen

Robotarmen består av åtta stycken servon, sex dynamixel AX-12A och två dynamixel AX-18A, men båda servotyperna styrs på samma sätt. Kommunikationen sker seriellt via UART, och följer dynamixels kommunikationsprotokoll [2]. Utifrån det protokollet har funktioner skrivits, bland annat `move_servo` och `get_angle`, för exempel på hur styrenheten styr servon se Exempelkod 3. Likt styrning av robotplattformen används här samma principer i manuell och autonom, men skillnader redogörs för nedan. När mikrodatorn startar sätts hastigheten på alla servon till ett värde som är längsammare än servonas originalvärde för att rörelserna inte ska bli lika ryckiga.

MANUELL STYRNING AV ROBOTARMEN

Vid manuell styrning av robotarmen finns det fyra kommandon som används: byt till högre led, byt till lägre led, rotera led motsols och rotera led medsols. Båda byt-led funktionerna byter aktiv led. Det finns sex numrerade ledar, där den första leden roterar basplattan, sjätte leden öppnar och stänger gripklon, och resterande styr armens fysiska ledar. Armens servon har begränsningar i vilka vinklar de kan nå så att de inte jobbar mot armens fysiska konstruktion, och i ledar med två servon används servonas inbyggda loadangle och action kommandon så att servona rörs exakt samtidigt. Det gör att servona laddas med önskade målvinklar, men väntar på ett till kommando, action, innan de faktiskt börjar

rotera. Det finns däremot inga begränsningar satta så att armen inte slår emot robotplattformen eller andra objekt under manuell styrning, det får användaren vara försiktig med själv.

Funktionen som får ett servo att rotera ändrar vinkeln kontinuerligt medan det kommandot skickas, som då skickas medan en knapp i användargränssnittet är nedtryckt. Till att få detta att ske kontinuerligt används en simpel räknare som räknar upp medan den aktiva instruktionen är rotera motsols eller medsols. När räknaren nått upp till ett specifikt värde nollställs den, och servot flyttas en kortare sträcka. Till detta används `get_angle` för att ta reda på den nuvarande positionen, sedan adderas eller subtraheras en konstant och den nya vinkeln ställs in på servot.

AUTONOM STYRNING AV ROBOTARMEN

Alla instruktioner som skickas till robotarmen under autonom styrning är förutbestämda i kommunikationsenheten. Kommandon som kan skickas till robotarmen under autonom styrning är enbart `moveservo`. Styrenheten får via SPI vilken led som ska styras och vilken vinkel den ska ställas in i. Hur dessa vinklar bestämdes beskrivs i [5.3.5](#).

5.3 Kommunikationsenheten

Kommunikationsenheten består av en Raspberry Pi. Dess uppgift är att sköta all kommunikation och de flesta beräkningarna. Kommunikationsenheten tar emot data från sensorenheten, bearbetar den och skickar informationen vidare både till PC:n för visning i användargränssnittet och till styrenheten för vidare styrning. Kommunikationsenheten tar även emot data från styrenheten som sedan skickas till PC:n för visning i användargränssnittet. På Raspberry Pi:n finns kod för SPI- och Bluetooth-kommunikation, reglering, kortaste vägen algoritm samt kod för körning i både manuellt och autonomt läge. På Raspberry Pi:n finns en separat fil med olika funktioner för manuell och autonom styrning samt en funktion som hanterar data.

5.3.1 Komponenter och programpaket

I Tabell 5 listas de komponenter som ingår i kommunikationsenheten.

Tabell 5: Lista över komponenter för kommunikationsenheten.

Komponent	Egenskap	Antal
Mikrodator	Raspberry Pi	1
Nivåskiftare		1

I Tabell 6 listas de programpaket som ingår i kommunikationsenheten.

Tabell 6: Lista över programpaket för kommunikationsenheten.

Programpaket	Egenskap
spidev	SPI-kommunikation
time	Funktioner för tidsfördröjningar och tidsmätningar i koden
socket	Bluetoothkommunikation

5.3.2 SPI-kommunikation

Via SPI-bussen sker det kommunikation mellan kommunikationsenheten och sensorenheten samt mellan kommunikationsenheten och styrenheten. Mellan de olika enheterna skickas data i form av bytes som representerar olika information eller kommandon. Se kopplingsschema i Figur 7 (Appendix A.2).

KOMMUNIKATION MELLAN RASPBERRY PI OCH SENSORENHETEN

Se Tabell 8 (Appendix A.4) för beskrivning av den data som skickas mellan Raspberry Pi:n och sensorenheten.

I en SPI-transaktion skickar Raspberry Pi:n vilken sensor som ska skicka data och sensorenheten skickar datan från den valda sensorn. Undantaget är gyrot som först kan skicka data när Raspberry Pi:n skickar att sensorenheten ska starta gyro-timern. Det finns även ett värde som anger att gyro-timern ska stoppas. För att få reda på roadmark (korsning) status skickas ett specifikt värde och sedan får antingen 0, 1, 2 eller 3 tillbaks beroende på om roadmark är ”en rak väg”, en plockstation åt höger, en plockstation åt vänster eller en korsning.

KOMMUNIKATION MELLAN RASPBERRY PI OCH STYRENHETEN

Se Tabell 9 (Appendix A.4) för beskrivning av den data som skickas mellan Raspberry Pi:n och styrenheten.

Från Raspberry Pi:n skickas data till styrenheten för att styra robotplattformen samt robotarmen. Värdena 0-6 samt 31 och 32 skickas från PC:n till Raspberry Pi:n och sedan direkt vidare till styrenheten för att där leda till olika styrkommandon, detta går att se i Tabell 10 och Tabell 9. Värdet 30 indikerar att kommande tre data beskriver reglerinfo; först en ”status” som berättar om reglerfelet är positivt eller negativt, sedan ”high” och ”low” som är reglerfelet uppdelat i två delar. Värdena 40 och 41 skickar gaspådrag från vänster respektive höger sida tillbaks till Raspberry Pi:n. Värdet 50 berättar att de följande tre värdena beskriver armens aktuella läge för en viss led; först vilken led, sedan vinkeln även här uppdelad i en ”high” och en ”low”. Värdet 60 fungerar på samma sätt, där de tre följande värdena ser ut på samma sätt, men nu istället används för att ta reda på vilket läge den specifika leden ska flyttas till.

5.3.3 Bluetooth-kommunikation

Bluetooth-kommunikationen gäller mellan PC:n och Raspberry Pi:n. Den främsta anledningen till denna anslutning är samtliga dataöverföringar mellan dessa enheter samt för att genomföra manuell styrning av robotplattformen och robotarmen med gripklo.

KOMMUNIKATION MELLAN RASPBERRY PI OCH PC

Bluetooth-anslutningen görs via en socket, se Exempelkod 7 och 8 (Appendix A.3). All dataöverföring börjar med en begäran från PC:n i form av ett tal som skickas i bytes. Vidare svarar Raspberry Pi:n med korrekt data beroende på vilken data den tog emot från PC:n. Data som PC:n vill få tillbaka begärs i en loop som genomförs varje tiohundrasedssekund, se Exempelkod 9 och 10 (Appendix A.3). Data som ska skickas vidare till styrenheten för manuell styrning skickas vid specifika knapptryck. Oavsett om det är data från sensorenheten, styrenheten eller annan data som PC:n begär kommer koden i grund vara enligt exemplen.

Se Tabell 10 (Appendix A.4) för att se vad värdena 1-6 och 31-67 innebär. Värdet 20 betyder att nästa data är aktuell led för armen. Värdet 70 innebär att de tre kommande värdena som skickas kommer att vara lagerbredd, lagerhöjd sedan antal noder det finns varor mellan. Värdena 80, 81 och 82 innebär alla att antingen 0 eller 1 skickas tillbaka till PC:n beroende på status på om det finns nytt styrbeslut, om en vara är upplockad samt om autonom körning är klar. Alla dessa ”statusar” representeras av flaggor i Raspberry Pi-koden. Värdet 99 betyder att körläge ska växlas mellan autonomt och manuellt.

5.3.4 *Main-loop*

I Raspberry Pi:n finns en main-funktion som först initierar bluetooth- och spi-kommunikation, se Exempelkod 14. Här skapas även tråden som Bluetooth körs på. Loopen som hela tiden körs kontrollerar om det är autonomt eller manuellt körläge.

5.3.5 *Autonom körning*

Då autonom körning är aktivt startar en loop i Raspberry Pi-koden som heter autonomous-loop där styrbeslut hanteras samt eventuella hinder. Styrbesluten ligger i en lista som skapas med hjälp av data från PC:n och kortaste väg-algoritmen. Denna lista avgör i vilken ordning beslut ska göras och med hjälp av linjesensorerna sker dessa olika beslut. Hinder avgör med hjälp av avståndssensorn och då den ger ett specifikt värde kommer loopen hamna i en if-sats för nya beslut om kortaste väg och styrbeslut, se Exempelkod 15. I exempelkoden visas beslutet ”Plocka”, men det finns även beslut enligt följande: ”Höger”, ”Vänster”, ”Vänd” och ”Lämna”.

Under autonom körning så ska robotarmens dels plocka upp vara vid styrbeslutet ”Plocka” och även lämna av korgen vid styrbeslutet ”Lämna”, dessa rörelser är hårdkodade och togs fram med hjälp av avläsning av aktuell vinkel för alla ledar för robotarmen. Robotarmen ställdes i ett önskat läge, sedan skickades alla värden från styrenheten till kommunikationsenheten där de sparades i en fil.

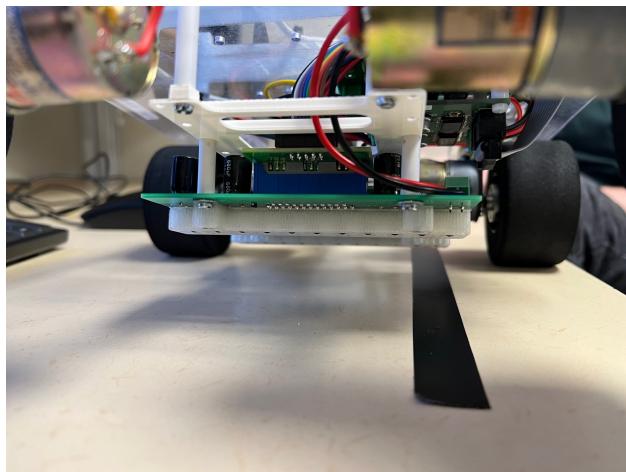
5.3.6 *Manuell styrning*

Manuella styrningen hanteras till största del i en funktion som heter bluetooth-control-loop. Där hanteras samtliga data som skickas från PC:n för olika hanteringar. Beroende på mottagen data kommer man hamna i olika if-satser för olika hanteringar. Detta kan vara att PC:n önskar data från någon sensor- eller styrenhet, eller för styrning av robotplattformen eller armen. Se Exempelkod 16.

5.3.7 *Reglering*

Regleringen för linjeföljningen sker genom en modifierad P-reglering. Algoritmen använder sig av fyra parametrar: front_error, back_error, Kp och Kd. Kp och Kd är konstanter som väljs genom testning. Variablerna front_error och back_error representerar avvikelsen från mitten för den främre respektive bakre reflexsen-

sorn. Se Exempelkod 5(Appendix A.3). Ett scenario som kan uppstå om man inte har med parametern K_p visas i Figur 4.



Figur 4: Illustration av parallellt fel.

Om regleringen endast består av:

$$\text{derivative} = K_D \cdot (\text{front_error} - \text{back_error}) \quad (3)$$

så kommer felet att bli noll trots att robotten inte är i mitten av tejpen och den kommer ej att regleras. Detta lösas genom:

$$\text{proportional} = K_p \cdot (\text{back_error}) \quad (4)$$

som gör att robotten rör sig in mot mitten igen.

5.3.8 **Kortaste-Vägen-Algoritm**

För att robotten ska navigera i lagret krävs algoritmer som beräknar den kortaste vägen mellan olika mål samt översätter användarens input till styrbeslut som robotten kan tolka. Kortaste-Vägen-Koden är uppdelad i två filer, en som endast beräknar vägen till alla varor och en som tar hänsyn till hinder också.

Vid start används den första filen som beräknar den kortaste vägen från startpunkten till samtliga målnoder och därefter tillbaka till start. När robotten stöter på ett hinder aktiveras den andra filen. Denna tar bort den aktuella vägen mellan de två noder som hindret ligger mellan. Startpunkten uppdateras till den senast besökta korsningen, och en ny väg

beräknas från denna punkt till alla mål och avslutas i nod 1 (alltid placerad längst upp i vänstra hörnet).

För att beräkna snabbaste vägen mellan alla målnoder använder sig koden av en kombination av Bredden-Först-Sökning (BFS) och Held-Karp-Algoritmen. Vid första körningen av programmet matas följande in via användargräns-snittet:

- Lagerbredd och lagerhöjd,
- Ett dictionary där varje nyckel är ett heltal från 1 till antalet målnoder,
- Värdet bakom varje nyckel är en lista som innehåller:
 - En enda nod, om det gäller startnoden,
 - Två noder, om det gäller en vara som befinner sig mellan två punkter.

När roboten stöter på ett hinder skickas ytterligare information till systemet:

- En lista med de två noder som hindret ligger mellan,
- Denna lista läggs till i en samlad lista över alla upptäckta hinder.

Vid detta tillfälle uppdateras startnoden till den senast besökta korsningen, och en ny vägberäkning initieras med hänsyn till de aktuella hindren.

Inledningsvis i båda fallen konstrueras en graf som representerar lagrets struktur se Exempelkod 11. Grafen kan betraktas som ett träd där varje nod motsvarar en korsning eller ett hörn i lagret. Denna graf implementeras som ett dictionary, där varje nod fungerar som en nyckel och dess värde är en lista över angränsande noder (grannar).

När roboten stöter på ett hinder aktiveras funktionen `remove_path`, vilken modifierar grafen genom att ta bort grann-relationen mellan de två noder som hindret ligger mellan. Detta innebär att den aktuella vägen inte längre betraktas som möjlig, vilket tvingar algoritmen att söka alternativa rutter.

När grafen skapats och eventuellt modifierats på grund av hinder, skapas en avståndsmatris, se Exempelkod 12. Den-na matris använder Bredden-Först-Sökning, se Exempelkod 13 för att beräkna avståndet mellan samtliga målpär i lagret. Den resulterande avståndsmatrissen skickas därefter till Held-Karp-Algoritmen, som används för att lösa det symmetriska Traveling Salesman Problem (TSP). Algoritmen beräknar den kortaste möjliga vägen som:

- Besöker varje målnod exakt en gång,
- Startar i en angiven startnod och slutar i en angiven slutnod (enligt filen som hanterar hinder),
- Alternativt både startar och slutar i nod 1 (enligt filen utan hinder).

Held-Karp-Algoritmen fungerar genom att iterativt undersöka alla möjliga kombinationer av besökta noder och suc-cessivt lägga till en obesökt nod i taget. Resultatet är:

- En lista som anger i vilken ordning målnoderna ska besökas,
- Den totala kostnaden för den optimala vägen.

Slutligen körs Bredden-Först-Sökning på nytt för att översätta denna besöksordning till en konkret sekvens av noder som roboten ska följa för att nå målnoderna i rätt ordning. När målnod nås anges detta i listan som 'goal'. Här har även specialfall och optimeringar tagits hänsyn till. Om roboten hittar två vägar som är lika långa men där en kräver en vändning och en inte gör det kommer den alltid välja vägen den inte behöver vända då det går fortare, se längre ner i Exempelkod 13.

För att översätta den genererade besökslistan med noder till styrbeslut som roboten kan tolka, används funktionen `path_to_instructions` som börjar med att översätta alla noder till koordinater, där till exempel 1 anges som (0,0), 2 som (0,1) och så vidare. Koordinaterna används sedan för att beräkna från vilket håll roboten kommer och vilket håll den ska åka för att komma till nästa nod. Dessa riktningar översätts till styrbesluten "höger", "vänster", "rakt", "vänd". När besökslistan anger 'goal' istället för en nod anges styrbeslutet `plocka`"i alla fall förutom det sista då lämnaläggs till allra sist. Styrbesluten avgör alltså vad roboten ska utföra i varje korsning alternativt `plockstation` längs vägen.

5.4 PC

All kod på PC:n är kopplat till användargränssnittet (GUI:n). GUI:n är uppdelad i olika fönster som täcker separata krav, dessa är rutnät (bana) med knappar för att ändra storlek och antalet varor, datavisning, styrbeslut och knappar för manuell styrning. Utöver detta finns det även knappar för att ändra läge mellan autonomt och manuellt. Det finns även en knapp för att kalibrera linjesensor och en knapp i autonomt läge för att starta körning. Se Figur 8, 9 och 10 (Appendix A.4) för bilder på GUI:n i de olika lägena. Som visat på bilderna har olika designval lagts till såsom färgändring och textändring vid knapptryck. Datainhämtningen från de olika modulerna och styrbesluten startar då användaren trycker på knappen "Start". Se kapitel 5.3.3 för mer detaljerad beskrivning av hur kod för datainhämtningen ser ut. På samma sätt avbryts denna inhämtning då användaren trycker på samma knapp igen som nu istället säger "Avbryt". I Tabell 7 listas de programpaket som ingår i PC:n.

Tabell 7: Lista över programpaket för PC.

Programpaket	Egenskap	Antal
tkinter	Funktioner för GUI	1
time	Funktioner för tidsfördröjningar och tidsmätningar i koden	1
socket	Blåtandskommunikation	1
itertools	Funktioner för iteration	1
collections deque	Dubbelsidig lista	1

6 SLUTSATSER

HUVUDSAKLIGA BIDRAG

Ett av de främsta bidragen för en fungerande lagerrobot är regleringen. En bra reglering avgör om resten av funktionerna kommer funka. Exempelvis måste roboten komma fram till en plockstation rakt för att den hårdkodade armen ska hamna rätt och ha möjlighet att plocka upp varan. Samma sak gäller vid svängar, med dålig reglering finns risken för att roboten inte hinner identifiera korsningarna.

Något som bidrog till bättre reglering och som implementerades ett tag in i projektet var att använda två reflexsensorer. I början användes endast en reflexsensor vilket gav en otillförlitlig reglering. Två reflexsensorer gjorde regleringen mer stabil.

Ett bidrag som blev en spontan lösning för sämre rotationer var eltejp på hjulen. På grund av friktion och annorlunda yta på de olika hjulen skapade detta en ostabil svängning. Eltejpen skapade därmed en jämnare yta och bättre och jämnt fördelad friktion.

FÖRBÄTTRINGAR

Om mer tid hade funnits, hade gruppen försökt implementera mer generella lösningar. Det är en hel del hårdkodning i mjukvaran som inte riktigt är tillämpbar i industrin, men tillräckligt för projektet. Det uppstår en bugg ibland i reglering som är svår att återskapa och fixa. Gruppen tror att det hade gått att lösa genom att ha tre stycken linjesensorer.

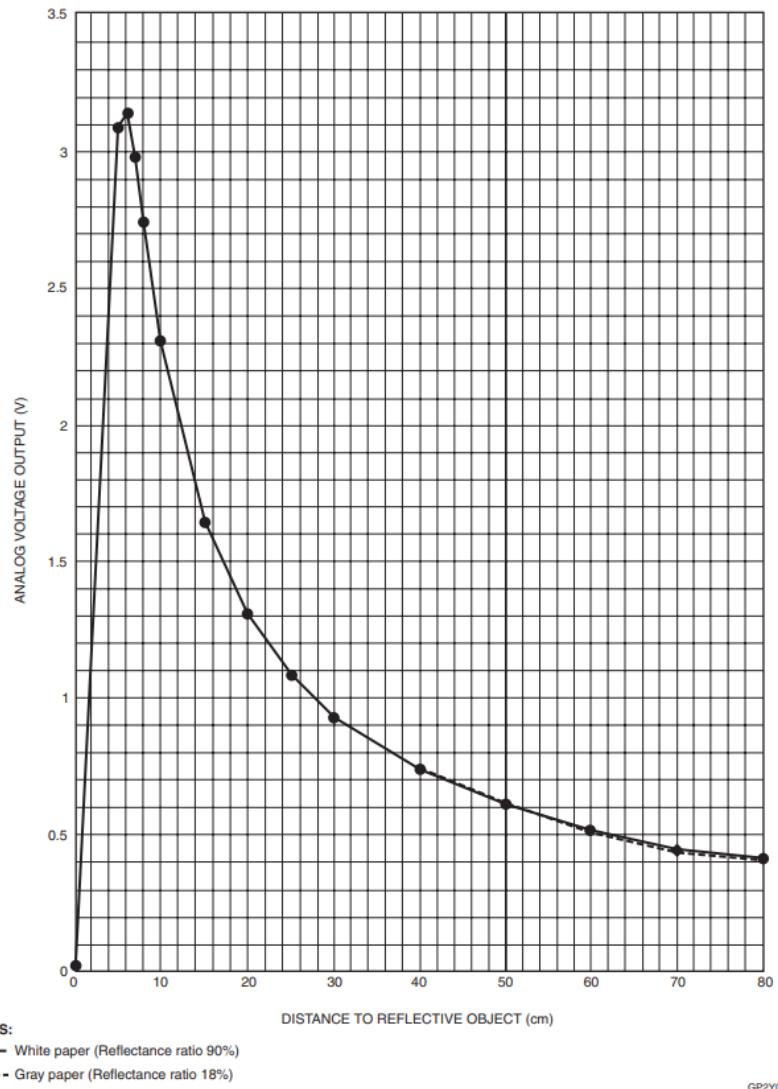
Det finns många värden som ska representera ett specifikt kommando eller som har en specifik betydelse, dessa värden har aldrig riktigt skrivits ner på ett samlat ställe under projektets gång vilket ledde till att dessa tog lång tid att sammanställa vid skrivandet av tekniska dokumentationen. Om projektet skulle göras om skulle detta kunna vara något att tänka på, så att alla värden finns samlade på samma ställe vilket också leder till en smidigare arbetsgång med mindre letande i koden.

REFERENSER

- [1] ISY. *Reflektion*. Hämtad: 2025-04-15. [Online]. Tillgänglig: https://da-proj.gitlab-pages.liu.se/vanheden/page/avr_raspberry/.
- [2] ROBOTIS. *AX-12A e-Manual*. Accessed: 2025-02-23. n.d. URL: <https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/>.

7 APPENDIX

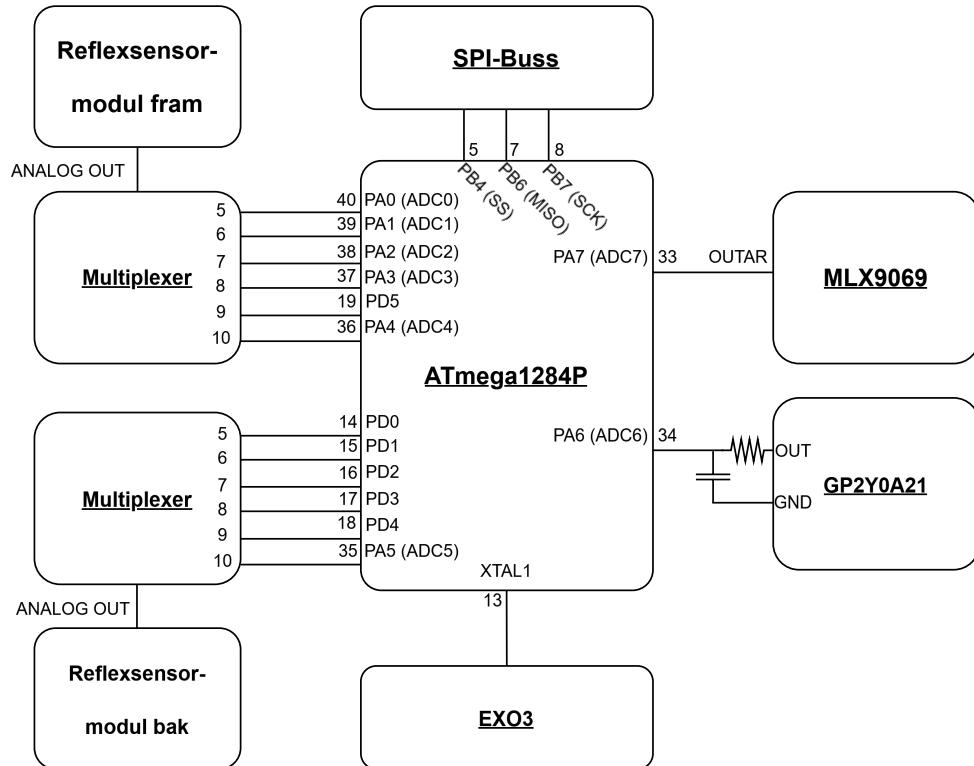
7.1 Grafer



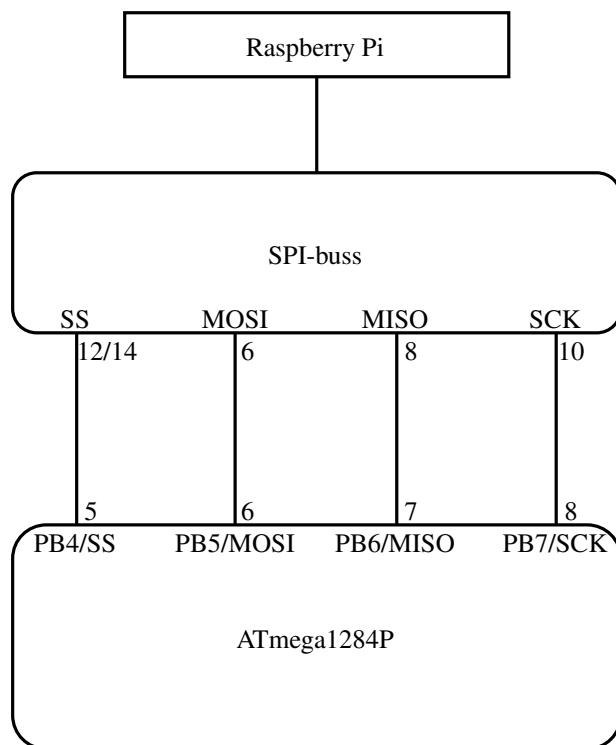
Figur 5: Spänningsvärden för avstånd.

7.2 Kopplingsscheman

Figur 6 visar kopplingsschemat för sensorenheten.



Figur 6: Kopplingsschema för sensorenheten.



Figur 7: Kopplingschema för SPI-bussen.

7.3 Exempelkod

Listing 1: Exempelkod för reflexsensormodulen.

```

1 int8_t read_line_back(int reflex_high)
2 {
3   int i;
4   volatile int8_t data;
5   volatile uint8_t indata = 0;
6   volatile int sum = 0;
7   volatile int sum_index = 0;
8   volatile int pivot = 0;
9
10  for(i = 1; i < 12; i++)
11  {
12    PORTD &= 0xF0;                                // Resets the 4 LSB bits in PORT A.
13    PORTD |= i;                                  // Set multiplexer to index i.
14    PORTD |= 0x10;                               // Start sensor.
15
16
17    indata = is_active_reflex(reflex_high);        // 1 if current sensor sees tape.
18    PORTD &= 0xEF;                                // Turn off sensor.
19
  
```

```

20     sum += indata;
21     sum_index += (i)*indata;
22 }
23
24 if (sum == 0)                                // Sees no tape.
25 {
26     pivot = 12;
27 } else {
28     pivot = (sum_index*2)/sum;                // Center of mass calculation
29 }
30
31 return data = (int8_t)(pivot);              // Center of mass calculation
32 }
```

Listing 2: Exempelkod för styrenhetens mainloop.

```

1
2
3 int main(void)
4 {
5
6     DDRA = 0b1010;
7     PORTA = 0b0010;
8     DDRD = 0b11000110; // Sät TXD och TX enable till output och RXD till input
9     USART_Init(MYUBRR);
10    SPI_init();
11    init_pwm();
12
13    _delay_ms(200);
14    for(unsigned int i=1; i <= 7; i++)
15    {
16        set_speed(i, 80); // Sänk alla servons hastighet frutom gripklon
17    }
18
19    set_speed(8, 500); // Sätt gripklons hastighet snabbare än resterande servon
20
21    while (1)
22    {
23
24        if(current_action == 0x1)
25        {
26            drive_fwd();
27        }
28        else if(current_action == 0x2)
29        {
30            rotate_left();
31        }
32        else if(current_action == 0x3)
33        {
34            reverse();
35        }
36        else if(current_action == 0x4 )
37        {
38            rotate_right();
39        }
40        else if(current_action == FWD_LEFT)
41        {
```

```

42         fwd_left();
43     }
44     else if(current_action == FWD_RIGHT)
45     {
46         fwd_right();
47     }
48     else if(current_action == 0)
49     {
50         stop();
51     }
52     else if(current_action == 0x14 && current_action != last_action)
53     {
54         decrease_speed();
55     }
56     else if(current_action == 0x15 && current_action != last_action)
57     {
58         increase_speed();
59     }
60     // Reglering
61     else if(current_action == 0x30)
62     {
63         float reglercopy;
64         cli();
65         reglercopy = reglerstyr;
66         sei();
67
68         drive_and_turn(reglercopy);
69     }
70     // Servon
71     else if(current_action == 0x31)
72     {
73         counter += 1;
74         if(counter >= timertime)
75         {
76             add1degree_joint(current_joint);
77             counter = 0;
78             _delay_ms(50);
79         }
80     }
81     else if(current_action == 0x32)
82     {
83         counter +=1;
84         if(counter >= timertime)
85         {
86             sub1degree_joint(current_joint);
87             counter = 0;
88             _delay_ms(50);
89         }
90     }
91 }
92 }
```

Listing 3: Exempelkod hur styrenheten styr servon.

```

1 void USART_Transmit( unsigned char* data, unsigned int size )
2 {
3     PORTD |= 0b100;
```

```

4
5 ;
6 // Lgger data i en buffer och skickas
7 for(unsigned int i = 0; i < size; i++)
8 {
9   while ( !( UCSR0A & (1<<UDRE0)) )
10  ;
11  UDR0 = (unsigned char) data[i];
12 }
13
14 while ( !( UCSR0A & (1<<TXC0)) )
15 ;
16 UCSR0A |= 1<<TXC0;
17 }
18
19 void move_servo(unsigned int ID, unsigned int Angle)
20 {
21
22   unsigned int header = 0xFF; //B rja varje med 2st 0xFF
23   unsigned int Length = 0x5; // 2 + antal P
24   unsigned int Instruction = 0x03; // skriv
25   unsigned int P1 = 30; // Address att skriva till
26   unsigned char P2 = (unsigned char)Angle;
27   unsigned char P3 = (unsigned char)(Angle>>8);
28   unsigned int Checksum = ~ (ID + Length + Instruction + P1 + P2 + P3);
29
30   unsigned char data[] = {header, header, ID, Length, Instruction, P1, P2, P3, Checksum};
31   unsigned int data_size = sizeof(data) / sizeof(data[0]);
32   USART_Transmit(data, data_size);
33 }

```

Listing 4: Exempelkod för styrenhetens linjeföljning.

```

1 void drive_and_turn(float turnvalue)
2 {
3   PORTA = 0b1000;
4   speed = reglerspeed;
5   if(speed + turnvalue < 0xFF && speed + turnvalue > 0)
6   {
7     if(turnvalue > 0)
8     {
9       OCR2A = speed + turnvalue;
10    }
11   else if(turnvalue < 0)
12   {
13     OCR2A = speed + turnvalue * 1.25;
14   }
15   else
16   {
17     OCR2A = speed;
18   }
19 }
20 else if(speed + turnvalue > 0xFE)
21 {
22   OCR2A = 0xFE;
23 }
24 else if(speed + turnvalue < 0)

```

```

25  {
26      OCR2A = 0;
27  }
28
29
30
31 if(speed - turnvalue < 0xFF && speed - turnvalue > 0)
32 {
33     if(turnvalue < 0)
34     {
35         OCR2B = speed - turnvalue;
36     }
37     else if(turnvalue > 0)
38     {
39         OCR2B = speed - turnvalue * 1.25;
40     }
41     else
42     {
43         OCR2B = speed;
44     }
45
46 }
47 else if(speed - turnvalue > 0xFE)
48 {
49     OCR2B = 0xFE;
50 }
51 else if(speed - turnvalue < 0)
52 {
53     OCR2B = 0;
54 }
55
56 }

```

Listing 5: Exempelkod för regleringen.

```

1 def PDController(front_error, back_error, KP, KD):
2     derivative = KD * (front_error - back_error)
3     proportional = KP * back_error
4     output = proportional + derivative
5     return output

```

Listing 6: Exempelkod hur delsystem i sensormodulen initieras.

```

1 void init_IR()
2 {
3     ADMUX = (0<<REFS1) | (0<<REFS0) | (1<<ADLAR) | (1<<MUX2) | (1<<MUX1) | (0<<MUX0);           // AREF,
4     left-shift, ADC6.
5     ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
6     // Activate ADC, Prescaler 128.
7 }
8 void init_gyro()
9 {
10    ADMUX = (0 << REFS1) | (0<<REFS0) | (1<<ADLAR) | (1<<MUX2) | (1<<MUX1) | (1<<MUX0);           // AREF,
11    left-shift, ADC7.

```

```

11    ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
12    // Activate ADC, Prescaler 128.
13 }
14
15 void init_line_front()
16 {
17    DDRA |= 0x0F;                                // Ports choice of sensor. Might change
18    DDRD |= 0x20;                                // Port for enable
19
20    ADMUX = (0<<REFS1) | (0<<REFS0) | (1<<ADLAR) | (1<<MUX2) | (0<<MUX1) | (1<<MUX0);           // AREF,
21    left-shift, ADC5.
22    ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
23    // Activate ADC, Prescaler 128.
24 }
25
26 void init_line_back()
27 {
28    DDRD |= 0x1F;                                // Ports for enable and choice of sensor.
29    ADMUX = (0<<REFS1) | (0<<REFS0) | (1<<ADLAR) | (1<<MUX2) | (0<<MUX1) | (0<<MUX0);           // AREF,
30    left-shift, ADC4. VIRA TILL 36
31    ADCSRA = (1<<ADEN) | (0<<ADSC) | (0<<ADATE) | (0<<ADIF) | (0<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
32    // Activate ADC, Prescaler 128.
33 }
34
35 void init_interrupt()
36 {
37    EICRA |= (1<<ISC01) | (1<<ISC00);          // Interrupt on rising edge.
38 }
39
40 void init_SPI()
41 {
42    DDRB = (1 << DDB6);                          // Pin 7 (MISO) to output.
43    SPCR = (1 << SPIE) | (1 << SPE) | (0 << DORD) | (0 << CPOL) | (0 << CPHA);           // Activate SPI-buss.
44 }

```

Listing 7: Exempelkod för hur en Bluetooth-anslutning görs skriven på klienten (PC).

```

1 import socket
2
3 hostMACaddress = 'B8:27:EB:E9:12:27'
4 port = 4
5 global s
6 s = socket.socket(socket.AF_BLUETOOTH, socket.SOCK_STREAM, socket.BTPROTO_RFCOMM) //create socket
7
8 s.connect((hostMACaddress, port))           // trying to connect

```

Listing 8: Exempelkod för hur en Bluetooth-anslutning görs skriven på Raspberry Pi.

```

1 import socket
2
3 hostMACaddress = 'B8:27:EB:E9:12:27'

```

```
4 port = 4
5 s = socket.socket(socket.AF_BLUETOOTH, socket.SOCK_STREAM, socket.BTPROTO_RFCOMM) //create socket
6 s.bind((hostMACaddress, port))    //Binding socket to address and port of the Raspberry Pi
7 s.listen(1)                      //Put socket in listening mode
```

Listing 9: Exempelkod för hur data skickas över Bluetooth från klienten (PC).

```

1 def sendbyte(byte):
2     try:
3         s.send(byte.to_bytes(1, 'big')) //Skicka 1 byte via socketen s
4     except Exception as e:
5         print(f"Fel vid sändning av byte: {e}")
6
7 def receive_data():
8     try:
9         data = s.recv(1) // Vänta på att ta emot 1 byte via socketen s
10        if data:
11            if len(data) == 1:
12                return data[0]
13            else:
14                return data
15        except Exception as e:
16            print(f"Fel vid mottagning av data: {e}")
17    return None
18
19
20 def send_and_receive(command):
21     try:
22         sendbyte(command)
23         received = receive_data()
24         return received
25     except Exception as e:
26         return
27
28 def data_loop(window):
29     data = bt.send_and_receive(command) // command = heltalet
30     data_list.append(data)
31     window.after(100, lambda: data_loop(window)) //lopa funktionen varje tiondelssekund

```

Listing 10: Exempelkod för hur data tas emot och sedan skickas tillbaka över Bluetooth för Raspberry Pi:n.

```

1 data = client.recv(1)
2 if data == "":
3     client.send(response.to_bytes(1, 'big')) //response = vald data att skicka tillbaka

```

Listing 11: Skapa graf - funktionen i snabbaste-vägen-koden.

```

1 def create_graph(width, height):
2     #creates a dictionary with all neighbours downward, right, left and over to all nodes
3     graph = {}
4     nx = width + 1
5     ny = height + 1
6     for y in range(ny):
7         for x in range(nx):
8             node = y * nx + x + 1
9             graph[node] = []
10            # right neighbour
11            if x < nx - 1:
12                graph[node].append(node + 1)
13            # down neighbour
14            if y < ny - 1:
15                graph[node].append(node + nx)

```

```

16     # left neighbour
17     if x > 0:
18         graph[node].append(node - 1)
19     # up neighbour
20     if y > 0:
21         graph[node].append(node-nx)
22
22 return graph

```

Listing 12: Create distance matrix funktionen i snabbaste-vägen-koden

```

1 def create_distancematrix(graph, nodes):
2     nr_nodes = len(nodes)
3     matrix = [[0]*nr_nodes for _ in range(nr_nodes)]
4     for i in range(nr_nodes):
5         for j in range(nr_nodes):
6             if i != j:
7                 path = bfs(graph, nodes[i+1], nodes[j+1], 0, 0)
8                 if i == 0:
9                     matrix[i][j] = len(path) - 2 if path else float('infinity')
10                else:
11                    matrix[i][j] = len(path) - 1 if path else float('infinity')
12
13 return matrix

```

Listing 13: Bredden-Först-Sökning i snabbaste-vägen koden

```

1 def bfs(graph, startnode, goalnode, next_node, last_node):
2
3     paths = []
4     for i in range(len(startnode)):
5         queue = deque([[startnode[i]]])
6         visited = set()
7         while queue:
8             path = queue.popleft()
9             node = path[-1]
10            if node in visited:
11                continue
12            visited.add(node)
13            for neighbor in graph[node]:
14                new_path = path + [neighbor]
15                if [node, neighbor] == mlnod or [neighbor, node] == goalnode or [neighbor] == goalnode:
16                    paths.append(new_path + ['goal'])
17                    queue.append(new_path)
18
19
20            if last_node == 0:
21                return min(paths, key=len)
22
23            if len(goalnode) == 1 and goalnode[0] in startnode and last_node != 1: #om varan ligger
24                j mte nod 1 och inte kommer fr n nod 1 ...
25                for i in startnod:
26                    if [i] != goalnode:
27                        return [goalnode[0], 'goal'] #... ska den ka ut
28
29            shortest_dist = min(len(l) for l in paths)
30            shortest_paths = [l for l in paths if len(l) == shortest_dist]

```

```

30
31     if len(shortest_paths) > 1:
32         for i in shortest_paths[:]:
33             if i[0] != next_node:
34                 shortest_paths.remove(i)
35         shortest = min(shortest_paths, key=len)
36     if shortest[0] != next_node:
37         shortest = [next_node] + shortest
38
39     return shortest

```

Listing 14: Main-loopen i Raspberry Pi:n.

```

1 def main():
2     global nav_plan
3     nav_plan = 0
4     s = bt.init_bluetooth()
5     spi_styr, spi_sensor = spi.initspi()
6
7     bt_thread = threading.Thread(target=bluetooth_listener, args=(s, spi_styr, spi_sensor),
8                                   daemon=True)
9     bt_thread.start()
10
11    KP, KD = 100, 100
12
13    while True:
14        old_Automatic = auto.Automatic
15        if auto.Automatic:
16            autonomous_loop(spi_styr, spi_sensor, KP, KD)
17            time.sleep(0.005)
18        if(old_Automatic == True and auto.Automatic == False):
19            spi.send_spi(spi_styr, 0)
20
21 if __name__ == "__main__":
22     main()

```

Listing 15: Exempelkod för Autonomous-loop i Raspberry Pi:n.

```

1 def autonomous_loop(spi_styr, spi_sensor, KP, KD):
2     global nav_plan
3     global new_plan
4     global goal_collected
5     global has_seen_roadmark
6     global nodeorder
7     global obstacles
8     global goods_deposited
9     try:
10         next_move = nav_plan[0]
11     except:
12         return
13     roadmark_status = auto.check_roadmark(spi_sensor)
14     m_lnoder = hm.get_m_lnoder()
15
16
17     if(auto.check_obstacle(spi_sensor)): //hinder
18         spi.send_spi(spi_styr, 0) //stanna
19

```

```

20     current_node, next_node = fw.find_location(nav_plan, nodeorder) //h mta aktiva noder
21     obstacles.append([current_node, next_node]) //l gg till hinder i lista
22     m_lnoder[1] = [current_node]
23     nav_plan, nodeorder = hm.update_path_obstacles(obstacles, m_lnoder) //ber kna ny v g
24     new_plan = True //skicka styrbeslut till PC
25     auto.rotate_right_180(spi_styr, spi_sensor) //rotera f r att undvika hinder
26
27 if(roadmark_status == 0):
28     has_seen_roadmark = False
29     auto.control_loop(spi_styr, spi_sensor, KP, KD)
30
31 elif(roadmark_status == 1): // dags att plocka vara t h ger
32     if(next_move == "plocka"):
33         goal_collected = True
34         spi.send_spi(spi_styr, 0) //stanna och plocka vara
35         auto.pick_right(spi_styr)
36         current_node, next_node = fw.find_location(nav_plan, nodeorder)
37         hm.remove_goal_node(current_node, next_node)
38         next_move = nav_plan.popleft()
39         if(nav_plan[0] == "v nd"):
40             auto.rotate_right_180(spi_styr, spi_sensor)
41             nav_plan.popleft()
42     else:
43         auto.drive_fwd(spi_styr) // K r f rbi plockstation

```

Listing 16: Exempelkod för bluetooth-control-loop i Raspberry Pi:n.

```

1 def bluetooth_control_loop(data, client, spi_styr, spi_sensor):
2     global new_plan
3     global nav_plan
4     global m_lnoder
5     global goal_collected
6     global goods_deposited
7
8     try:
9         if(data == "20"): // ndrar aktuell armled
10            data = client.recv(size)
11            try:
12                spi.send_spi(spi_styr, [0x20] + list(data))
13            except:
14                print("Invalid data 1")
15
16        elif data == "60": //IR-data
17            response = spi.send_spi(spi_sensor, 0)
18            client.send(response.to_bytes(1, 'big'))
19        elif data == "61": //Reflex-data
20            response = spi.send_spi(spi_sensor, 1)
21            client.send(response.to_bytes(1, 'big'))
22        elif data == "67": //kalibrera linjesensor
23            spi.send_spi(spi_sensor, 7)
24            time.sleep(0.01)
25
26        elif data == "81": //upplockade varor
27        if goal_collected:
28            client.send(b'\x01')
29            goal_collected = False
30        else:

```

```
31     client.send(b'\x00')
32
33 elif data == "82": //avbryta autonomt l ge i gui
34     if goods_deposited:
35         client.send(b'\x01')
36         goods_deposited = False
37     else:
38         client.send(b'\x00')
39
40 else:
41     try:
42         response = spi.send_spi(spi_styr, int(data, 16)) //manuell styrning
43     except:
44         print("Invalid data 2")
45     except:
46         print("Disconnected, looking for new socket")
```

7.4 Tabeller

Tabell 8: Lista över dataöverföring mellan Raspberry Pi och sensorenhet.

Data som Raspberry Pi skickar till sensorenheten	Betydelse
0	Skicka IR-data från sensorenheten till Raspberry Pi
1	Skicka främre reflex-data till Raspberry Pi
2	Skicka bakre reflex-data till Raspberry Pi
3	Starta gyro
4	Stoppa gyro
5	Skicka vinkel från nolläge för gyrot till Raspberry Pi
6	Skicka 0, 1, 2 eller 3 beroende på roadmark status till Raspberry Pi

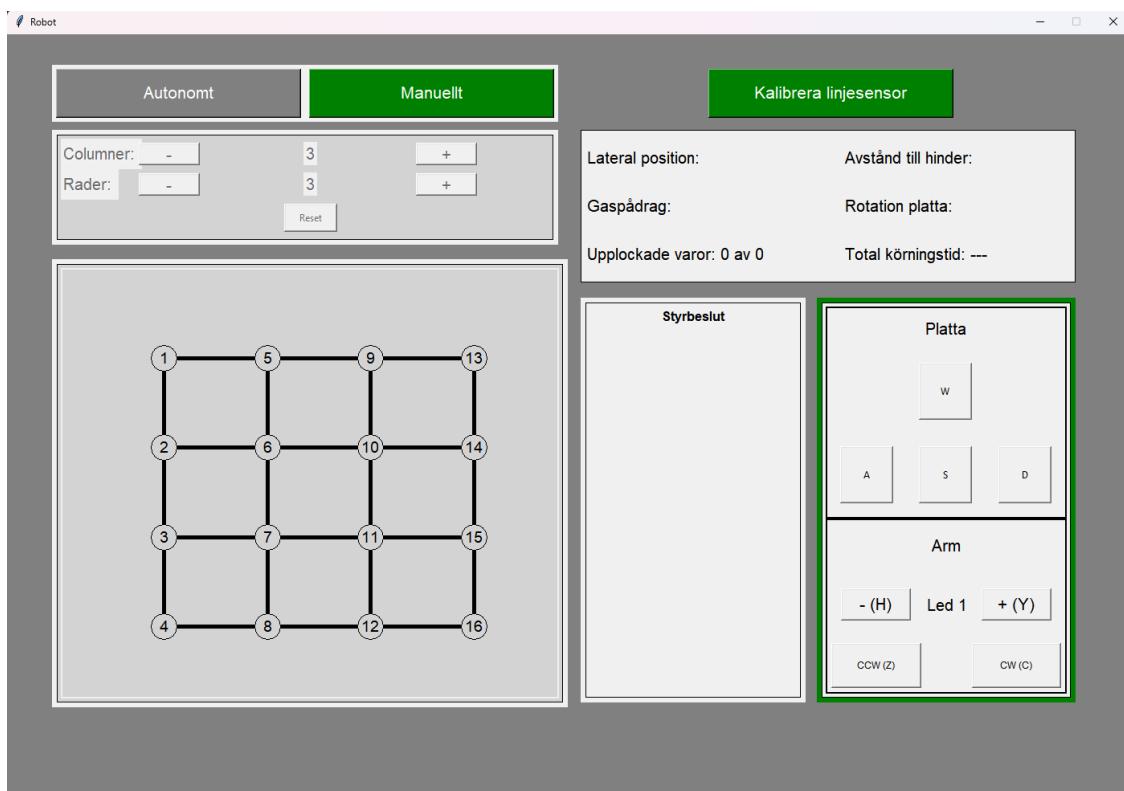
Tabell 9: Lista över dataöverföring mellan Raspberry Pi och styrenhet.

Data som Raspberry Pi skickar till styrenheten	Betydelse
0	Stanna
1	Kör framåt
2	Rotera vänster
3	Kör bakåt
4	Rotera höger
5	Kör vänster framåt
6	Kör höger framåt
20	Ändra aktuell armled
30	Meddela att efter detta värde kommer reglerinfo
31	Rotera gripklo moturs
32	Rotera gripklo medurs
40	Skicka gaspådrag vänster från styrenheten till Raspberry Pi:n
41	Skicka gaspådrag höger från styrenheten till Raspberry Pi:n
50	Meddela att efter detta värde kommer info om armens aktuella läge
60	Meddela att efter detta värde kommer info om hur armen ska flyttas

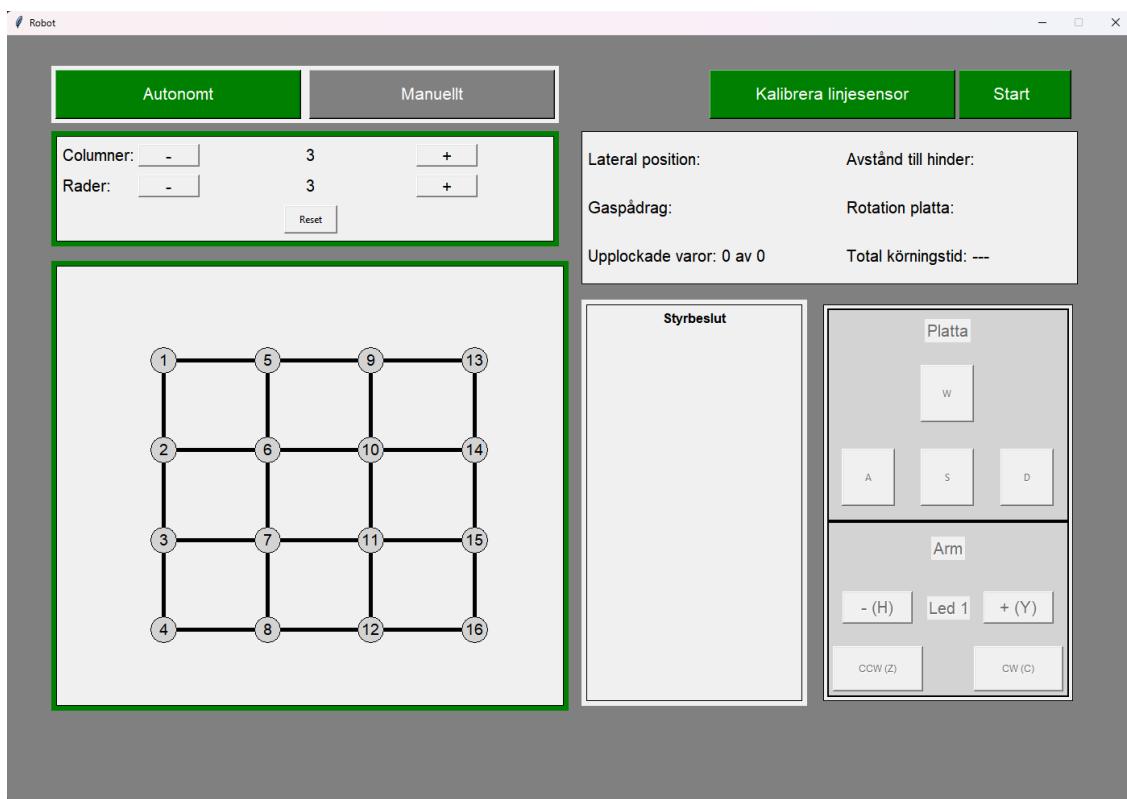
Tabell 10: Lista över dataöverföring mellan PC och Raspberry Pi.

Data som PC skickar till Raspberry Pi	Betydelse
0	Skicka vidare värdet 0 till styrenheten
1	Skicka vidare värdet 1 till styrenheten
2	Skicka vidare värdet 2 till styrenheten
3	Skicka vidare värdet 3 till styrenheten
4	Skicka vidare värdet 4 till styrenheten
5	Skicka vidare värdet 5 till styrenheten
6	Skicka vidare värdet 6 till styrenheten
20	Ändra aktuell armled
31	Skicka vidare värdet 31 till styrenheten
32	Skicka vidare värdet 32 till styrenheten
60	Skicka IR-data från Raspberry Pi till PC
61	Skicka reflex-data från Raspberry Pi till PC
62	Skicka gyro-data från Raspberry Pi till PC
63	Starta gyro
64	Stoppa gyro
65	Skicka gaspådrag höger från Raspberry Pi till PC
66	Skicka gaspådrag vänster från Raspberry Pi till PC
67	Kalibrera reflexsensor
70	Meddela att efter detta värde kommer lagerinfo
80	0 eller 1 skickas från Raspberry Pi till PC beroende på status på styrbeslut
81	0 eller 1 skickas från Raspberry Pi till PC beroende på status på upplockade varor
82	0 eller 1 skickas från Raspberry Pi till PC beroende på status autonom körsning
99	Växla läge mellan autonomt och manuellt

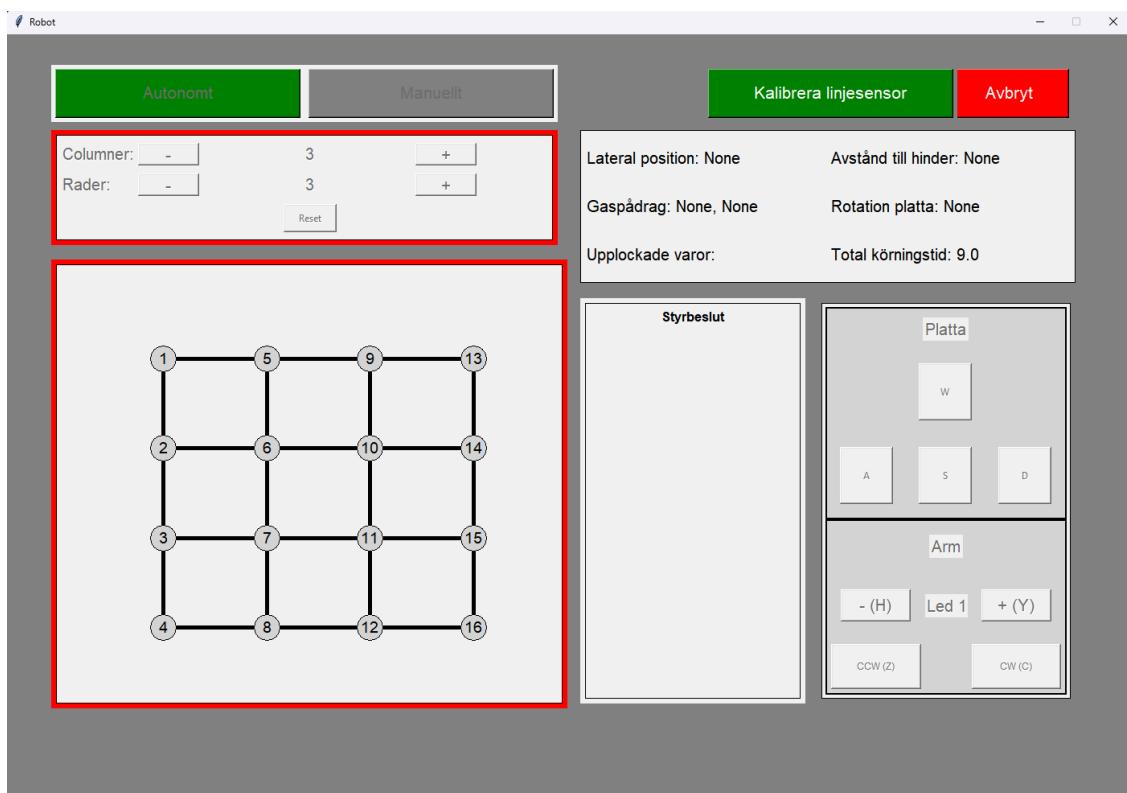
7.5 Övrigt



Figur 8: Användargränssnittet vid uppstart, även manuellt läge.



Figur 9: Användargränssnittet i autonomt läge.



Figur 10: Användargränssnittet vid start av köring i autonomt läge.

16 EFTERSTUDIE

Efterstudie

Grupp 6

9 Juni 2025

Version 1.0



Status

Granskad	Andreas Nordström	2025-06-09
Godkänd	Namn	2025-xx-xx

Beställare:

Mattias Krysander, Linköpings universitet
Telefon: +46 13282198
E-post: mattias.krysander@liu.se

Handledare:

Theodor Lindberg, Linköpings universitet
E-post: theodor.lindberg@liu.se

Projektdeltagare

Namn	Ansvar	E-post
Linus Funquist		linfu930@student.liu.se
Ebba Lundberg	Dokumentansvarig	ebblu474@student.liu.se
Andreas Nordström	Projektledare	andno7733@student.liu.se
Sigge Rystedt		sigry751@student.liu.se
Ida Sonesson	Dokumentansvarig	idaso956@student.liu.se
Lisa Ståhl	Designansvarig	lisst342@student.liu.se

INNEHÅLL

1	Tidsåtgång	1
1.1	Arbetsfördelning	1
1.2	Tidsåtgång jämfört med planerad tid	1
2	PROJEKTFÖRBEREDENDAMOMENT	1
2.1	Reflektioner av de nya laborationerna	1
2.2	Förstudier	2
2.3	Yrkesetik	2
3	ANALYS AV PROJEKTARBETET	2
3.1	Vad hände under de olika faserna	2
3.2	Hur vi arbetade tillsammans (ansvar, beslut, kommunikation etc.)?	3
3.3	Hur använde vi projektmodellen?	3
3.4	Hur fungerade relationen med beställaren?	3
3.5	Hur fungerade relationen med handledaren?	3
3.6	Tekniska framgångar/problem	3
4	Måluppfyllelse	4
4.1	Har studiesituationen påverkat projektet?	4
5	SAMMANFATTNING	5
5.1	De tre viktigaste erfarenheterna	5
5.2	Godaråd till de som ska utföra ett liknande projekt	5

DOKUMENTHISTORIK

Version	Datum	Utförda förändringar	Utförd av	Granskad
1.0	2025-06-06	Version 1.0	LF, EL, AN, SR, IS, LS	AN

1 TIDSÅTGÅNG

Enligt tidsplanen skulle vi ha tillsammans ha lagt 1380 timmar men på grund av bland annat sjukdom lades i slutändan 1313 timmar.

1.1 Arbetsfördelning

Arbetsfördelningen var jämn. Även om det skilje i timmar i slutet har ändå samtliga medlemmar bidragit till att slutresultatet blev så bra som det blev.

1.2 Tidsåtgång jämfört med planerad tid

Fas	Planerad tid i timmar	Använt tid i timmar
Före BP2 (Krav och Planering)	Krav ca 60 tim, Plan ca 100 tim	60 tim
Efter BP2	Ca 1320 timmar	1313 tim

2 PROJEKTFÖRBEREDET MOMENT

Nedan diskuteras de inledande momenten i kursen: laborationer, etik och förstudier.

2.1 Reflektioner av de nya laborationerna

Nedan diskuteras de inledande laborationerna i kursen.

2.1.1 *Mätlab*

Mätlaben var väldigt viktig då logikanalysatorn användes en hel del under projektets gång, framförallt för bussen och robotarmen.

2.1.2 *AVR-laboration*

AVR-laborationen var viktig för att bekanta sig ännu mer med databladet och avbrotsrutiner.

2.1.3 *I2C-laboration*

Även om gruppen inte använde I2C så kändes ändå laborationen relevant och man fick mer övning på att använda logikanalysatorn.

2.2 Förstudier

Tre förstudier skrevs i projektet. De handlade om kommunikationssätt, sensorer och styrkontroll.

2.3 Yrkesetik

Etikmomentet i kursen bestod av både föreläsningar och seminarier. Gruppen uppskattade att detta område integrerades i kursen, då en separat kurs i ämnet hade upplevts som för omfattande sett till det som lärdes ut.

Det var otydligt att ett etikseminarium skulle ta upp gruppkontrakt och hur man skriver dessa. Detta framgick inte i början och gruppen hade redan upprättat ett gruppkontrakt innan seminariet. Detta gjorde att seminariet kändes överflödigt.

Gruppmedlemmarna tyckte att etikseminarierna var bättre än föreläsningarna. Det är ett ämne som mer är lämpat för diskussion jämfört med att sitta och lyssna på någon i två timmar. Gruppen tyckte inte att vi fick användning av etikdelen i projektet.

3 ANALYS AV PROJEKTARBETET

I kommande kapitel görs en analys av projektarbetet utifrån ett par frågeställningar.

3.1 Vad hände under de olika faserna

Hela första perioden fram till mars skrevs en hel del dokument och det var många deadlines att hålla koll på. Den här fasen genomfördes utan några problem.

Under andra perioden så påbörjades konstruktionen av roboten, främst mjukvarukonstruktionen. Den här fasen gick också bra. Problem förekom, men inom ramarna för vad som är normalt i ett sådant projekt.

3.2 Hur vi arbetade tillsammans (ansvar, beslut, kommunikation etc.)?

Samarbetet och kommunikationen inom gruppen fungerade bra. Det har inte uppstått några större problem inom gruppen under projektets gång. Gruppkontraktet som upprättades följdes bra och gruppen hade inte gjort om det.

3.3 Hur använde vi projektmodellen?

LIPS-modellen användes främst fram till BP3 då de flesta dokumenten skrevs. Under konstruktionsfasen användes modellen mot slutet då kandidatrapport och teknisk dokumentation skulle skrivas. Detta dokument baseras också på modellen.

3.4 Hur fungerade relationen med beställaren?

Kommunikationen fungerade bra mellan gruppen och beställaren, och omförhandling av krav gick smidigt. Däremot hade gruppen önskat snabbare svar via Teams.

3.5 Hur fungerade relationen med handledaren?

Gruppen är mycket nöjd med handledaren. Han svarade snabbt på meddelanden via Teams och tog sig nästan alltid tid att komma förbi labbet när frågor uppstod. Jämfört med handledarna för övriga grupper upplevde gruppen att vår handledare var särskilt hjälpsam – ett stort plus!

3.6 Tekniska framgångar/problem

Nedan listas de tre mest besvärliga problemen som gruppen har stött på under projektet. Problem 1 är det mest besvärliga problemet.

1. Problem 1

- felets typ - mjukvarufel/systemintegrationsfel
- Användargränssnittet kraschar efter att för mycket data tagits emot från Raspberryn.
- Förmodad orsak var förmodligen timing-problem över bluetooth-kommunikationen.
- Problemet fixades aldrig helt. En lösning var att minska antalet datahämtningar vilket gjorde att användargränssnittet slutade krasha, på bekostnad av att datan uppdaterades längsammare.

e) Väldigt mycket tid lades ner på detta. Säkert över hundra timmar om man ser till allas timmar.

1. Problem 2

- a) felets typ - mjukvarufel/systemintegrationsfel
- b) Linjesensorn hinner reglera på en korsning ibland, vilket den inte ska.
- c) Förmadad orsak var förmögligen att det tog för lång tid att upptäcka att det var en korsning, och hann då reglera.
- d) Problemet var svårt att återskapa och vi är inte helt säkra på att det var på grund av detta som det skedde. Lösningen som vi tror fungerade var att begränsa gaspådraget oavsett storlek i felet från reglering.
- e) Mycket tid lades ner här också, särskilt när det påverkade robustheten så pass mycket. Förmögligen lades över 100 timmar här också.

1. Problem 3

- a) felets typ - hårdvarufel
- b) Robotarmen gick sönder dagen innan leverans.
- c) Vi vet inte alls varför den gick sönder. Den hade använts på samma sätt hela tiden.
- d) Det fixades genom att vi begärde ut en ny arm.
- e) Då vissa servon skiljde på den nya armen jämfört med den gamla så behövdes onödig tid dagen innan leverans läggas på detta. Ca 30 timmar utspritt på gruppen.

4 MÅLUPPFYLLElse

I följande kapitel diskuteras huruvida målen uppfylldes.

Alla planerade mål har uppnåtts av gruppen och leveransen fungerade bra. Kraven på robusthet uppfylldes och roboten presterade väldigt bra.

4.1 Har studiesituationen påverkat projektet?

Den första perioden på vårterminen var det mer fokus på andra kurser, då man läser två kurser parallellt med projektet. Detta var inget problem, då den här kursen inte kräver så mycket timmar den första perioden. Den andra perioden krävde mycket tid till projektet, men det vägdes också upp av att man endast läser en mindre kurs parallellt. Det var bra att examinatoren tidigt i kursen rekommenderade att man inte läser mer än en kurs parallellt under andra perioden.

5 SAMMANFATTNING

Sammanfattningsvis är gruppen nöjd med hur projektet har genomförts och det slutliga resultatet. Roboten har inte bara löst uppgiften, utan den har också gjort det på ett bra sätt om man ser till robusthet och prestanda.

5.1 De tre viktigaste erfarenheterna

En viktig erfarenhet gruppen tar med sig från projektet är att det ibland kan vara lämpligt att göra om implementationen för till exempel en sensor, även om det är långt in i projektet.

En viktig erfarenhet att ta med sig är vikten av planering. Första fasen av projektet bestående av mycket planering kändes till en början seg men senare under projektet märkte vi hur viktigt det verkligen var med planering. Det underlättade att ha en plan att luta sig mot, trots att det kanske inte stämde helt överens med hur projektet verkligen gick till. Detta innefattar speciellt de olika aktiviteterna och deras respektive tidsåtgång. Under abretets gång uppmärksammade vi att de aktiviteter vi planerat inte var tillräckligt specifika och det var oklart var jobbade timmar skulle läggas.

Det hade hjälpt mycket att ha en tydlig testplan. Att sitta ner tillsammans och försöka komma på många olika fall som kan ske och sedan testa dem.

5.2 Goda råd till de som ska utföra ett liknande projekt

Utnyttja avgränsningarna i projektdirektivet och kravspecifikation och hårdkoda det som kan hårdkodas. Spendera inte orimligt mycket tid på att felsöka mjukvaran, ibland är det fel på hårdvaran och ibland slutar koden fungera från ena dagen till den andra utan någon rimlig anledning. Finns det parametrar som ska ändras och testas ofta, som till exempel konstanter i reglering, så rekommenderar gruppen att man gör det smidigt att ändra dessa värden.