# Förstudie regler

Grupp 6

24 februari 2025

Version 0.1

LINKÖPING UNIVERSITY

Status

| | | |
|---|---|---|
| Granskad | Namn | 2025-02-XX |
| 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

## 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 INLEDNING

Detta dokument är en förstudie i kursen TSEA56, Elektronik kandidatprojekt och kommer att handla om den reglerteknik som krävs för att få lagerroboten att fungera samt metoder för att beräkna kortaste vägen genom lagret.

## 1.1 Syfte

Syftet med denna förstudie är att få djupare förståelse för den reglerteknik som krävs för kursen TSEA56, mer specifikt för projektet autonom lagerrobot, och att tränas i akademisk skrivning och informationssamling från källor.

# 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 METOD

Frågeställningarna kommer att besvaras med hjälp av underlag från vetenskapliga texter.

# 4 SPÅRFÖLJNING

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

## 4.1 Val av reglering för spårföljning

I PID-reglering (Proportional-Integral-Derivative) ges insignalen $u(t)$ baserat på avvikelsen $e(t)$ mellan önskad utsignal $r(t)$ och verklig utsignal $y(t)$, alltså $e(t) = r(t) - y(t)$ [1], även kallat återkoppling. Formeln för PID-reglering ser ut som följande:

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

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 men kan resultera i ett fel kallat steady-state error. Den andra termen $K_I \int_0^t e(t)$ kallas för den integrerande delen. Den tar i stället hänsyn till tidigare fel och kan på så sätt eliminera steady-state error. 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.

# 5 BANA OCH RÖRELSEPLANERING

Innan roboten börjar åka kommer dimensioner på lagermiljön samt var varorna finns att anges, därefter ska roboten i autonomt läge beräkna den kortaste vägen genom lagret och sedan eventuellt 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.

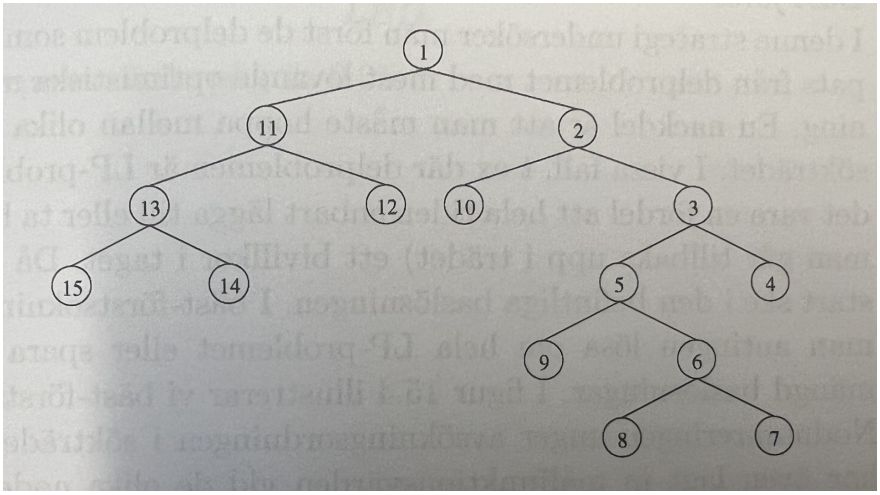
## 5.1 Algoritm för att hämta varorna en och en

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. Avståndet mellan en nod i en fyrvägskorsning och en nod där vara finns är kortare men lika lång från båda hållen vilket innebär att man kan bortse från längdskillnaden. 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.

### 5.1.1 *Djupet först*

Djupet först algoritmen börjar med att först söka en gren så långt som möjligt (exempelvis alltid till höger) tills en lövnod nås. Därefter stegar den tillbaka och fortsätter utforska nästa möjliga gren på samma sätt [2]. Detta säkerställer att alla möjliga vägar genomsökts, se figur1.

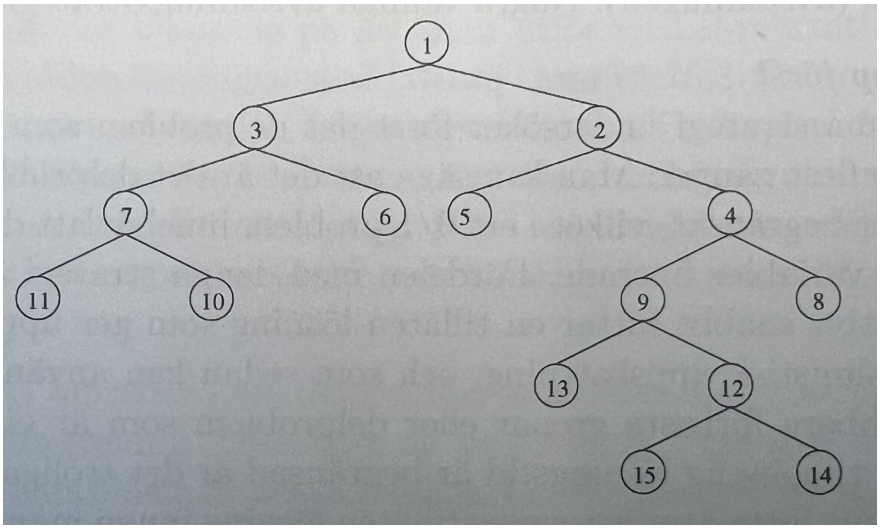



**Figur 1:** Träddiagram djupet först [2].

Denna metod fungerar bra då en väg behöver hittas snabbt men som då inte nödvändigtvis är den kortaste vägen.

### 5.1.2 *Bredden-Först-sökning*

Bredden först 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 2 [2].




**Figur 2:** Träddiagram bredden först [2].

## 5.2 Algoritm för att hämta alla varor i en färd

Om alla varor ska 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 optimala 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 besökts återvänder roboten till startpunkten. Denna algoritm är väldigt billig, men oftast fås inte den mest 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 gås 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) är det troligtvis snabbt nog.

# 6 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.[5].

För att lösa invers kinematik måste armens modelleras och det brukar göras med Denavit-Hartenberg (D-H) parametrar [6]. Parametrarna beskriver varje länks egenskaper, vilket tillsammans ger en bra beskrivning av hela armen. De parametrar som ingår beskrivs nedan:

- $a_i$ : Länklängd, längden av armen efter vridningspunkten
- $\alpha_i$ : Länkvridning, skillnaden i två på varandra följande länkars vridningsaxel.
- $d_i$ : Länkförskjutning
- $\theta_i$ : Länkvinkel, specifika vinkeln servon är inställd på, ändras beroende på armens specifika konfiguration.

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 för mångafrihetsgrader, kan en analytisk lösning också vara svår att tillämpa, då det kan finnas många olika sätt, i vissa fall oändligt många, att orientera armen så att gripklon når en specifik position.

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 [7]. 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.

# 7 DISKUSSION OCH SLUTSATSER

## 7.1 Nödvändiga styrmoder

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

## 7.2 Reglering av styrmoder

I fallet där en bil ska följa en tejp kommer ej kvarstående reglerfel finnas[8]. Den kommer aldrig köra förskjutet längst med tejpen utan kommer reglera så fort den inte är i mitten och sedan dämpa oscillationerna som uppstår efter regleringen. Detta innebär att endast Proportionell (reagera snabbt i rätt riktning) samt Deriverande (dämpa de oscillationer som uppstår från den snabba reaktionen) reglering kommer användas och därmed ingen Integrerande faktor. Funktionen blir därmed:

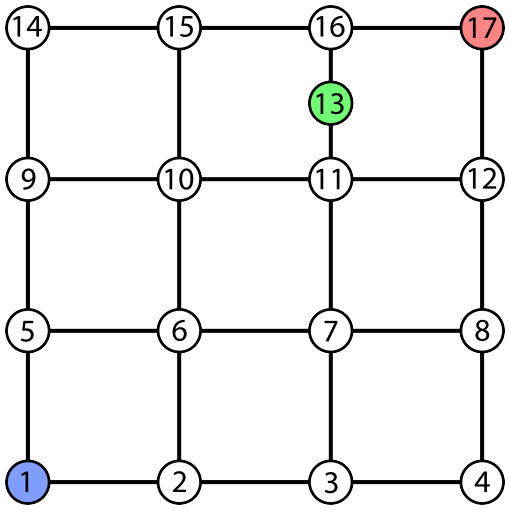
$$u(t) = K_P e(t) + K_D \frac{d}{dt} e(t)$$

Reglering av spårföljning görs alltså med PD-reglering som med hjälp av sensorinput beräknar lateralt fel och justerad styrvinkel, vilket den sedan skickar till kommunikationsmodulen.
Vägnavigeringen kommer regleras med hjälp av input i form av karta och position på varor samt kostnader för olika vägar.

### 7.3 Koordination av servon i robotarm

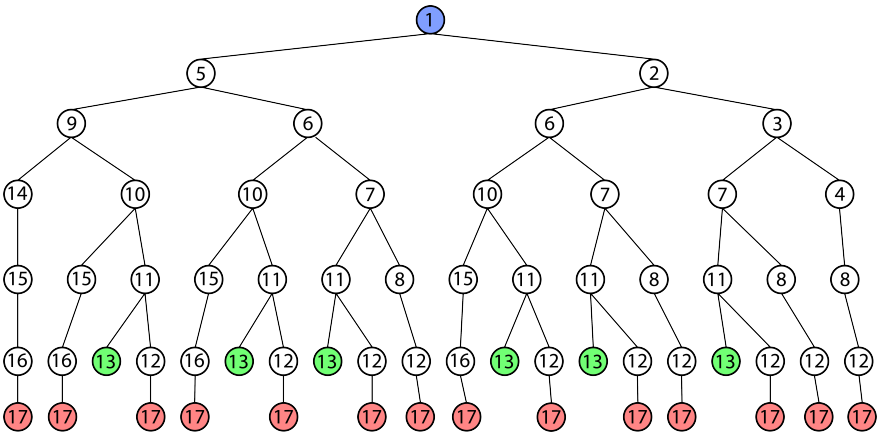
### 7.4 Bana och rörelseplanering

I fallet där varorna hämtas en och en är användningen av bredden först sökning mest effektiv. Då lagret är utformat som ett rutnät skulle inte alla vägar ut ur en nod behöva tillåtas användas för att hitta snabbaste vägen. Om startpunkten antas vara i nedre vänstra hörnet enligt figur 3.



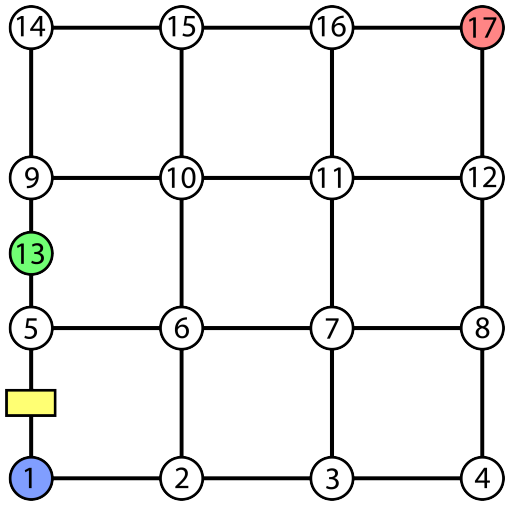

**Figur 3:** Exempel på lagermiljö med numrerade noder.

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 4.



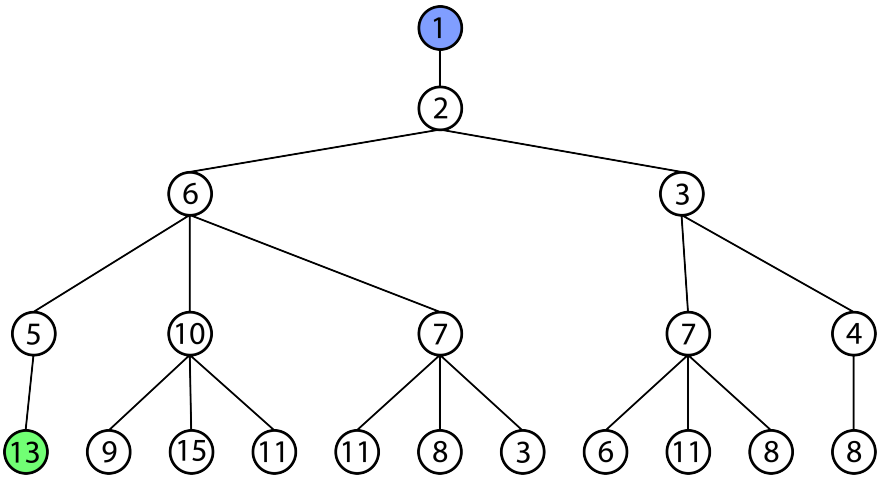

**Figur 4:** Träddiagram över noder i lagermiljö.

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 5.




**Figur 5:** Exempel på lagermiljö med numrerade noder och hinder.

Detta innebär att snabbaste vägen kan kräva steg till vänster eller nedåt för att komma till varan. Därför måste efter påkommet hinder, enligt figur 6 alla vägar avsökas.




**Figur 6:** Träddiagram över lagermiljö med noder och hinder.

Första gången roboten beräknar en väg, utan att veta var hinder befinner sig, fungerar det första sättet men när hinder hittas måste roboten beräkna ny väg och då alltså använda den andra algoritmen. Vid påkommet hinder tar sig roboten tillbaka till närmaste nod, beräknar ny väg samt raderar vägen som hindret låg på.

När roboten väl kommer fram till rätt nod känner den själv av vilken sida om tejpen varan är på.

# 8 REFERENSER

## REFERENSER

[1] T. Glad och L. Ljung, *Reglerteknik: Grundläggande teori*, 4. utg. Lund, Sweden: Studentlitteratur, 2003, ISBN: 978-91-44-02308-1.

[2] J. Lundgren, M. Rönnqvist och P. Värbrand, *Optimeringslära*, 2. utg. Lund, Sweden: Studentlitteratur, 2010, ISBN: 978-91-44-05512-9.

[3] W3Schools, *Traveling Salesman Problem*, https://www.w3schools.com/dsa/dsa_ref_traveling_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] U. of Illinois - Motion Group, *Inverse Kinematics*, Accessed: 2025-02-22, 2024. URL: https://motion.cs.illinois.edu/RoboticSystems/InverseKinematics.html.

[6] 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. URL: https://etj.uotechnology.edu.iq/article_168857.html.

[7] ROBOTIS, *AX-12A e-Manual*, Accessed: 2025-02-23, n.d. URL: https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/.

[8] 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%A4sningar%2FF%C3%B66%2DAVR%2Epdf&parent=%2Fsites%2FLisam%5FTSEA56%5F2025VT%5FM8%2FCourseDocuments%2FProjektmodul%2FF%C3%B6rel%C3%A4sningar.

# 9 APPENDIX