Smart Warehouse Inventory Management

PROIECT DE SEMESTRUL 1, ANUL 3

Student: **Dan Ovidiu**

**Paşca Adrian Alexandru**

Disciplina: **Sisteme bazate pe Cunoaștere**

Cuprins

[1 Introducere 2](#_Toc189414559)

[1.1 Context general 2](#_Toc189414560)

[1.2 Obiective 2](#_Toc189414561)

[1.3 Specificații 2](#_Toc189414562)

[2 Cunoașterea și analiza setului de date 3](#_Toc189414563)

[3 Pre-procesarea setului de date 5](#_Toc189414564)

[4 Modelarea sistemului 7](#_Toc189414565)

[4.1 Identificarea funcțiilor de transfer 7](#_Toc189414566)

[4.2 Proiectarea regulatorului 11](#_Toc189414567)

[5 Concluzii 12](#_Toc189414568)

[5.1 Rezultate obținute 12](#_Toc189414569)

[5.2 Alte funcționalități 13](#_Toc189414570)

[5.3 Direcții de dezvoltare 13](#_Toc189414571)

[6 Bibliografie 14](#_Toc189414572)

# Introducere

## Context general

Automatizarea proceselor de gestionare a stocurilor este esențială pentru eficientizarea fluxurilor de lucru în depozitele moderne. Proiectul "Smart Warehouse Inventory Management" are ca scop dezvoltarea a 2 roboţi autonomi capabili să realizeze sarcini specifice, cum ar fi deplasarea controlată și detectarea traseelor, contribuind astfel la optimizarea operațiunilor de gestionare a depozitelor.

## Obiective

* Colectarea datelor de performanță a motoarelor pentru analiza comportamentului.
* Pre-procesarea datelor pentru eliminarea zgomotului.
* Identificarea funcțiilor de transfer pentru ambele motoare.
* Proiectarea și implementarea unui regulator PI folosind metoda Ziegler-Nichols.
* Implementarea unui algoritm de lane detection pentru navigația autonomă a robotului

## Specificații

* Controlul a două motoare DC utilizând 2 regulatoare PI.
* Utilizarea senzorilor optici pentru detectarea traseului marcat.
* Medii de dezvoltare: Python (pentru pre-procesare), MATLAB (pentru modelare și control), Arduino (testarea funcționalităților).

# Cunoașterea și analiza setului de date

Primul pas a fost să aplicăm diferiți factori de umplere pe fiecare motoraș al MainBot-ului. Cele trei valori sunt 85, 128 și 255, care reprezintă tensiunea aplicată pe fiecare DC motor în parte, care sunt alimentate de 4 baterii de 3.7V, dispuse câte două în serie și apoi conectate în paralel, care oferă o tensiune totală de 7.4V.

A black background with white text

Description automatically generated

Figură .Tensiunea pentru PWM

Fiecare set de date a fost salvat într-un fișier excel separat, care ulterior a fost importat în Matlab pentru vizualizare. Datele au fost preluate cu o perioadă de eşantionare de 0.01s.A screenshot of a computer

Description automatically generated

Figură . RPM pentru PWM =85

A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generated

Figură 4.RPM pentru PWM=255

Figură 3.RPM pentru PWM=128

Am început prin a înțelege setul de date. Având doar trei caracteristici, nu a fost necesar să reducem dimensionalitatea setului de date.

A screenshot of a computer

Description automatically generated

Figură 5.Analiza datelor

# Pre-procesarea setului de date

Pre-procesoarea datelor a fost realizată în PhyCharm/Jupyter Notebook.

Următorul pas a fost aplicarea tehnicilor de eliminare a zgomotelor. Pentru fiecare PWM ales și pentru fiecare motoraș am folosit tehnica care s-a potrivit cel mai bine. Am separat datele în 2 categorii: date pentru identificare și date pentru antrenare/validare. Această caracteristică ne-a ajutat ulterior pentru realizarea modelului matematic.

Identificare

* Motorul DC drept la PWM = 85 – lowpass filterA screen shot of a graph

  Description automatically generated

Figură 6

* Motorul DC stâng la PWM = 85 – median filterA screen shot of a computer

  Description automatically generated

Figură 7

Validare

* Motorul DC drept la PWM = 128 – lowpass filter

A screen shot of a graph

Description automatically generated

Figură 8

* A screen shot of a graph

  Description automatically generatedMotorul DC stâng la PWM = 255 – lowpass filter

Figură 9

După filtrare, datele au fost importate înapoi în Matlab pentru identificare și validarea modelului matematic.

# Modelarea sistemului

Pentru modelarea sistemului ne-am propus să aplicăm răspunsul la treaptă, din care va rezulta un model parametric, de tip cutie gri (parțial cunoscut în avans). Funcția de transfer are forma:

A black background with white text

Description automatically generated

Figură 10..Funcția de transfer de ordinul 1

## Identificarea funcțiilor de transfer

Am aplicat următorul algoritm pentru identificarea parametrilor:

A screenshot of a computer

Description automatically generated

Figură 11..Etapele analizei răspunsului la treaptă

În Matlab, am utilizat următoarea secvență de cod:

* pentru motorul stâng:

A screenshot of a computer

Description automatically generated

S-a obținut modelul cu parametrii K = 24.2915 și T = 0.01. Comparând datele cu modelul obținem:

A screenshot of a computer

Description automatically generated

Figură 11.Modelul mathematic pentru motorul stâng

* A screenshot of a computer

  Description automatically generatedpentru motorul drept:

S-a obținut modelul cu parametrii K = 30.1339 și T = 0.06. Comparând datele cu modelul obținem:

A screen shot of a graph

Description automatically generated

Figură 12.Modelul matematic pentru motorul drept

Mai apoi, am validat modelele pe celelalte două grafice:

* pentru motorul drept

A screen shot of a graph

Description automatically generated

Figură 13

* A screen shot of a graph

  Description automatically generatedpentru motorul stâng

Figură 14

## Proiectarea regulatorului

Am proiectat un regulator de tip PI. Am observat că timpul de răspuns este unul foarte bun pentru ambele modele, deci efectul derivativ poate fi eliminat.

Metoda folosită pentru determinarea regulatoarelor a fost Ziegler-Nichols, care este o tehnică rapidă și eficientă de proiectare. Am dus la limita de stabilitate sistemul, unde am obținut un Kcr și un Tcr, iar mai departe am folosit tabelul:

A white sheet with black text

Description automatically generated

Parametrii regulatoarelor de tip PI sunt:

* pentru motorul drept:

A math equations and numbers

Description automatically generated with medium confidence

* pentru motorul stâng:

A white background with black text

Description automatically generated

# Concluzii

## Rezultate obținute

Am evaluat răspunsul fiecărui motor DC controlat de PI la semnalul de tip treaptă și am obținut următoarele rezultate:

A screen shot of a graph

Description automatically generatedA screen shot of a graph

Description automatically generated

## Alte funcționalități

Am utilizat regulatoarele PI pentru a dezvolta un sistem de lane detection bazat pe doi senzori digitali optici. Acesta funcționează pe un traseu închis la culoare, marcat cu linii albe.

* **Ambii senzori detectează linii:** robotul se deplasează înainte cu viteză constantă.
* **Doar senzorul drept detectează linia:** motorul stâng se oprește, iar cel drept rămâne activ, robotul virând spre stânga.
* **Doar senzorul stâng detectează linia:** robotul virează spre dreapta, prin oprirea motorului drept și menținerea activă a celui stâng.

## Direcții de dezvoltare

* Dezvoltarea regulatoarelor pentru lane detection pentru a păstra o traiectorie optima
* Adăugarea de noi senzori pentru o navigație mai precisă
* Implementarea brațului robotic pentru a muta obiecte
* Conectarea MainBot-ului cu CarryBot-ul

# Bibliografie

https://www.mathworks.com/help/control/ug/proportional-integral-derivative-pid-controllers.html

[**https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols\_method**](https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method)

<https://www.mathworks.com/help/control/ref/dynamicsystem.step.html>

<https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/>

https://busoniu.net/teaching/sysid2024/sysid24ro\_transient\_handout.pdf