

Dokumentacja do projektu z przedmiotu:

**Mechatronika i Szybkie Prototypowanie Układów Sterowania**

Ramię Chwytaka na Lego NXT

LP1 C1, EADI-3

Bełch Jakub

Bazan Arkadiusz

Bizior Mateusz

Rzeszów, I 2017

1. Cel i zakres projektu

Celem projektu było wykonanie standardowego ramienia zakończonego chwytakiem, wzorowanego na typowym rozwiązaniu robota–manipulatora, z tym ze nasz mial 3 stopnie swobody (chwytak nieobrotowy w zadnej osi). Ramie składa sie z 3 przegubów – dwóch w płaszczyznie pionowej oraz jeden pozwalajacy obracac calego robota.

Sterowanie zrealizowano za pomoca interfejsu (API?) wydanego przez LEGO do programu Matlab. Zastosowalismy troche obiektowe podejscie (diagram klas i metod) dzieki czemu kod jest czytelniejszy i latwiejszy w rozbudowie.

1. Rozwiązania konstrukcyjne

Konstrukcja zostala wykonana z klockow LEGO co zapewni szybkie jej wykonanie ale ogranicza swobode wymiarowania, silniki sa duze, trzeba dostosowac sie do rozstawu klockow itp.. Pierwszym problemem bylo usztywnienie przegubu silnika A, ktory trzymal sie tylko na swojej osi a utrzymac musial ciezar calego robota (glownie dwa duze silniki oraz konstrukcje).

1. Algorytm

XXXXXXXXXXXXXXXXXSterowanie zrealizowano za pomoca interfejsu (API?) wydanego przez LEGO do programu Matlab. Zastosowalismy troche obiektowe podejscie (diagram klas i metod) dzieki czemu kod jest czy

1. Napotkane Problemy

sd

1. Podsumowanie i Wnioski

asd

1. Algorytm

Ogolny algorytm wynika z rownania[[1]](#footnote-2) :

(1)

Do obliczen komputerowych wygodny jest zapis macierzowy.

Sprawdzenie czy tranzycja jest gotowa do odpalenia sprowadza sie do sprawdzenia czy kazdy element wektora Mc jest wiekszy lub rowny odpowiadajacemu mu elementowi wiersza macierzy D-, w skrocie czy jest na tyle zetonow zeby bylo z czego zabrac (nie mozna miec ujemnej liczby zetonow)

Przy czym trzeba zauwazyc ze nei mozna od razu przemnozyc calej sieci (wykonac rownania (1) poniewaz wykonanie kazdej tranzycji moze nam zmienic stan wektora Mc i wplynac na kolejne tranzycje.

Powoduje to tez duze uproszczenie oszczedzajac nam mnozenia macierzy (kosztowne obliczeniowo) i sprowadza sie do sprawdzenia czy wektor Mc ≥ Di- (i - sprawdzana tranzycja czyli wiersz macierzy D-) i jesli tak wykonanie dzialania Mc+Di (i - wiersz macierzy incydencji D), i tak po kolei.

Cyfry w macierzach to wagi poszczegolnych strzalek

1. Opis programu

Program posiada glowna klase PetriNet ze wszystkimi potrzebnymi polami i koniecznymi metodami specyficznymi dla klasy.

Metody bardziej uniwersalne sa dostepne w statycznej klasie Helpers (jak import/export z kontrolek, dzialania na wektorach

W programie staralem sie uzyc ulatwien jezyka C# i .NET 4.5 takich jak listy, zdarzenia, oblsuga wyjatkow, zapis/odczyt do pliku, serializacja JSON itp.

1. Oznaczenia i skróty

* – wektor znakowan poczatkowych
* – wektor znakowan aktualnych
* – macierz diagonalna jednostkowa
* – macierz strzalek wchodzacych, dodajacych zetony
* – macierz strzalek wychodzacych, odejmujacych zetony
* – macierz incydencji

1. Kod źródłowy:

<https://github.com/KubaMiszcz/PetriNetCsharp/releases/tag/1.0>

1. Interfejs programu





# Bibliografia

1. **Krzysztof Sacha, prof. dr hab. inż.** http://www.ia.pw.edu.pl/~sacha/petri.html. [Online]

2. www.stackoverflow.com. [Online]

1. przy pierwszym kroku Mc=M0 [↑](#footnote-ref-2)