Projekt linii magazynowej w *Robot Studio*

Politechnika Warszawska

Wydział Elektryczny

Automatyka i Robotyka Stosowana

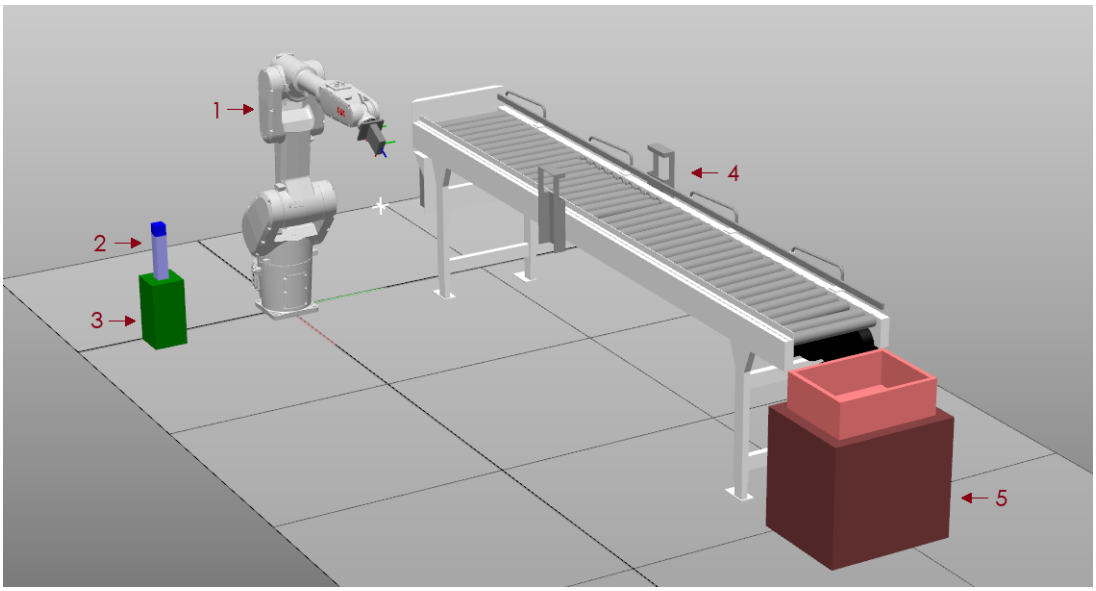
9 stycznia 2024

Spis treści

1 Wstęp

Sprawozdanie jest omówieniem realizacji projektu na przedmiot *Projektowanie zrobotyzowanych linii produkcyjnych*. Celem było utworzenie stanowiska z robotem, który miał przenosić przedmioty z 'magazynu' do miejsca docelowego za pomocą przenośnika taśmowego. Projekt zrealizowano w *Robot Studio* (v.23.3.10550.0).

1.1 Omówienie elementów stanowiska



Rys. 1.1: Całościowy widok stanowiska

Stanowisko składa się z pięciu najistotniejszych elementów widocznych na grafice:

1. Robot (IRB1300) wraz z chwytakiem,
2. Obiekty robocze - pięć sześciennych kostek o wymiarach 50x50x50 mm,
3. Przestrzeń 'magazynu'
4. Przenośnik taśmowy,
5. Przestrzeń docelowa - pudełko, w którym powinny znaleźć się obiekty robocze po zakończeniu pracy robota.

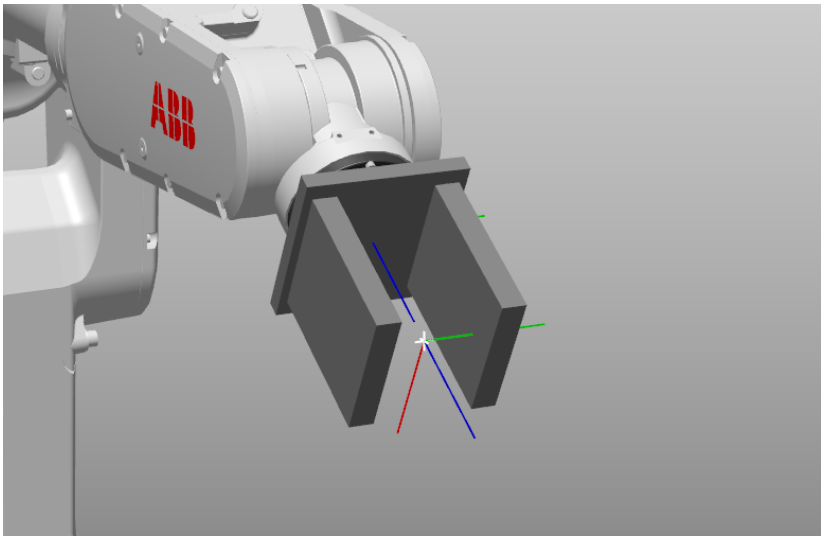
1.2 Robot - informacje wstępne

1.2.1 Model chwytaka

Model chwytaka został wykonany samodzielnie z użyciem podstawowych elementów geometrycznych oferowanych przez *RobotStudio*. Składa się z 3 elementów:

- Podstawy - wymiary: 100x100x10 mm
- Dwóch palców - wymiary: 15x80x100 mm

Maksymalny rozstaw palców wynosi $d_{MAX} = 60$mm. Wymiary chwytaka zostały podyktowane przez przyjętą wielkość obiektu roboczego (50x50x50 mm).



Rys. 1.2: Wygląd chwytka, widoczny również jego układ współrzędnych

1.2.2 Wybór modelu robota

Przy wyborze robota zasugerowano się jego zasięgiem oraz udźwigiem.

Zasięg Zdecydowano, że robot musi mieć zasięg większy od 1m. Wynika to nie tylko z komfortu pracy - obiekty stanowiska można rozłożyć dalej minimalizując ryzyko uzyskania trajektorii kolizyjnej - ale również podyktowane jest dość sporą wysokością zaimportowanego przenośnika taśmowego (wysokość ok. 80cm)

Udźwig Podyktowany jest ciężarem obiektów roboczych oraz chwytaka. Oba elementy domyślnie wykonane są ze stali ($\rho = 7800$ [kg/m^3]), zatem wyliczając ich łączną masę:

$$M = m_g + m_c = (V_g + V_c) \cdot \rho \simeq 3.63\ [kg]$$

Robotem spełniającym oba te wymagania jest **IRB1300** o zasięgu 1.4m oraz udźwigu 7kg.

2 Station Logic

2.1 Chwytak

2.1.1 Tworzenie mechanizmu

By móc oprogramować chwytak oraz wykorzystać go jako narzędzie należało utworzyć mechanizm z wcześniej zamodelowanych komponentów. Wykonano to w następujący sposób:

1. Wybrano opcję *Create mechanism*
2. Nadano nazwę oraz wybrano rodzaj narzędzia *Tool*
3. Dodano 3 elementy mechanizmu - podstawę oraz dwa palce
4. Utworzono dwa złącza pryzmatyczne o rozsunięciu 30mm każde
5. Dodano układ współrzędnych narzędziowych znajdujący się między palcami, oddalony o 20mm od końca ich długości
6. Dodano 3 pozycje mechanizmu:
    - *GripperOpen* - palce maksymalnie rozsunięte
    - *GripperClosed* - palce maksymalnie zsunięte
    - *Home* - domyślna pozycja chwytaka identyczna do *GripperClosed*

Po utworzeniu mechanizmu umieszczono go na robocie poprzez przeciągniecie obiektu chywtaka na obiekt robota.

2.1.2 Utworzenie Smart Componentu

By odseparować funkcje logiczne odpowiadające wyłącznie za chwytak utworzono *Smart Component*, który następnie dodano do *Station Logic*.

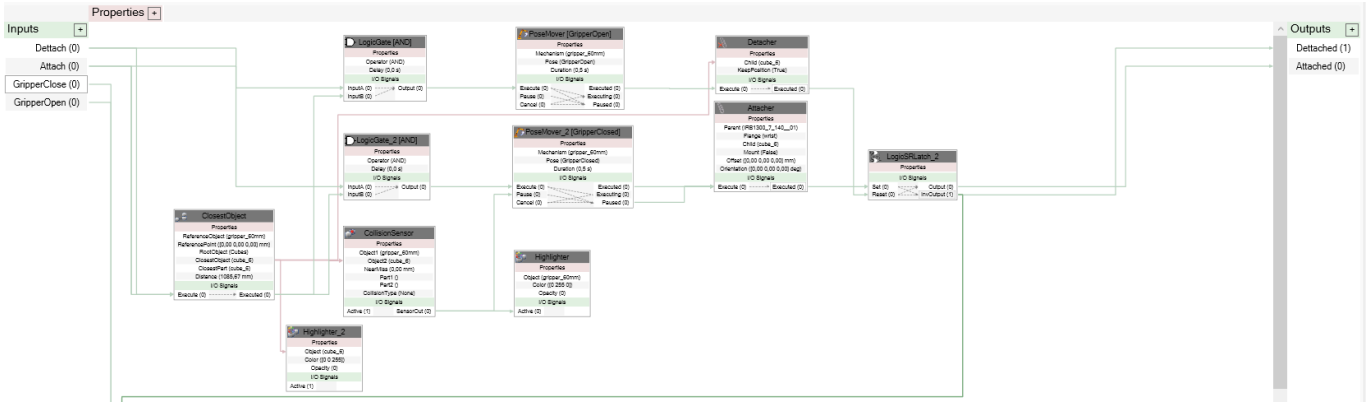
Sygnały logiczne związane z pracą chwytaka:

- *GripperClose* - odpowiada za zamknięcie chwytaka
- *GripperOpen* - odpowiada za otworzenie chwytaka
- *Attach* - odpowiada za 'łapanie' obiektów przez chwytak
- *Detach* - odpowiada za puszczanie obiektu trzymanego przez chwytak
- *Attached* - sygnał wyjściowy sygnalizujący, że chwytak trzyma jakiś obiekt.
- *Dettached* - sygnał wyjściowy sygnalizujący, że chwytak nie trzyma żadnego obiektu.

Chwyt obiektu Funkcja chwytania została zrealizowana przez współpracę czterech bloków funkcyjnych:

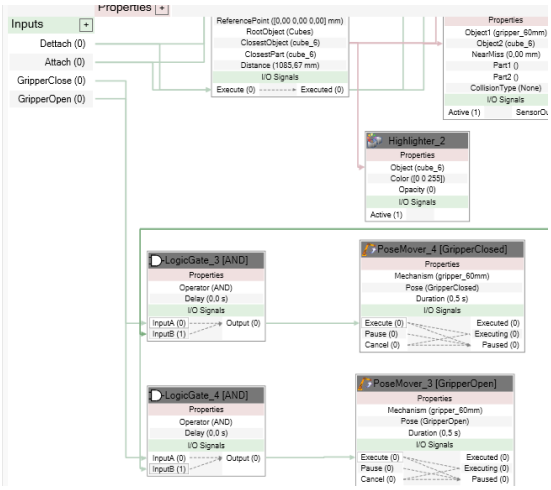
- *Closest Object* - zwraca obiekt znajdujący się najbliżej (w linii prostej) od obiektu referencyjnego (chwytaka). Dodatkowo zastosowano obostrzenie, by funkcja przeszukiwała jedynie dzieci elementu *Cubes* zamiast wszystkich elementów stacji.
- *Collision Sensor* - zwraca logiczny stan wysoki, kiedy obiekty (chwytaka oraz obiektu zwracanego przez *Closest Object*) ulegną kolizji.
- *Attach* - 'przykleja' obiekt dziecka (obiektu roboczego) do rodzica (chwytaka), pozwalając na imitację chwytu.
- *Pose Mover* - zmienia pozycję obiektu (chwytaka) do wcześniej zadeklarowanej pozy.

Wypuszczenie obiektu Zostało zrealizowane niemalże bliźniaczo do funkcji chwytania, jedyną różnicą jest użycie funkcji *Detach*, która 'rozkleja' obiekt rodzica i dziecka oraz brak sensora kolizji.



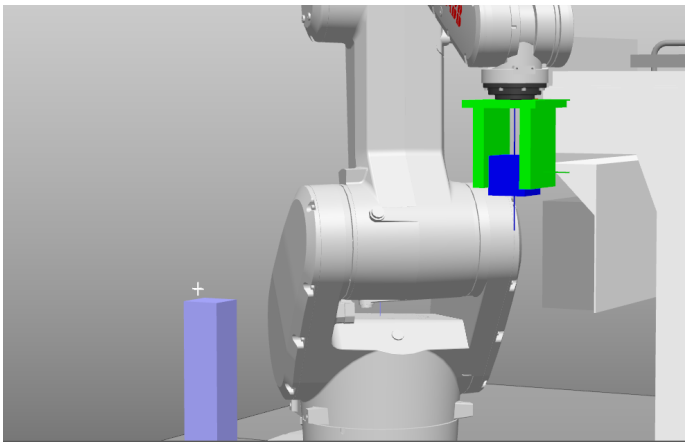
Rys. 2.1: Implementacja opisanej logiki

Zmiany pozycji chwytaka By móc manipulować chwytakiem bez chwytania bądź wypuszczania obiektu dodano prosty fragment kodu:



Rys. 2.2: Otwieranie chwytaka z zabezpieczeniem - funkcja AND sygnału sterującego i *Detached*

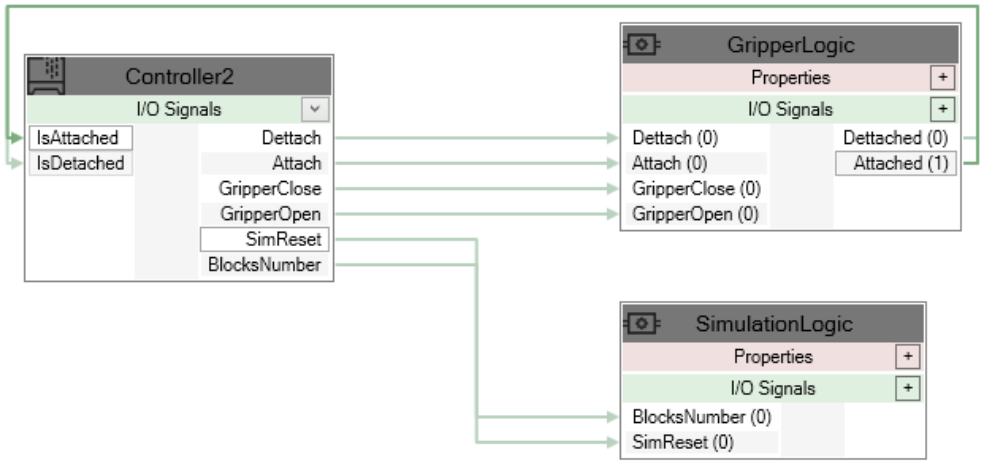
Podświetlanie By zailustrować działanie programu w czasie symulacji dodano bloki *Highlighter*, które zmnieją obiekty na zielono lub niebiesko odpowiednio gdy ulegną kolizji lub są wybrane jako najbliższy obiekt od chwytaka.



Rys. 2.3: Działanie bloków *Highlighter*

2.2 Station Logic

Finalnie logika robota sprowadza się do następnującej formy:



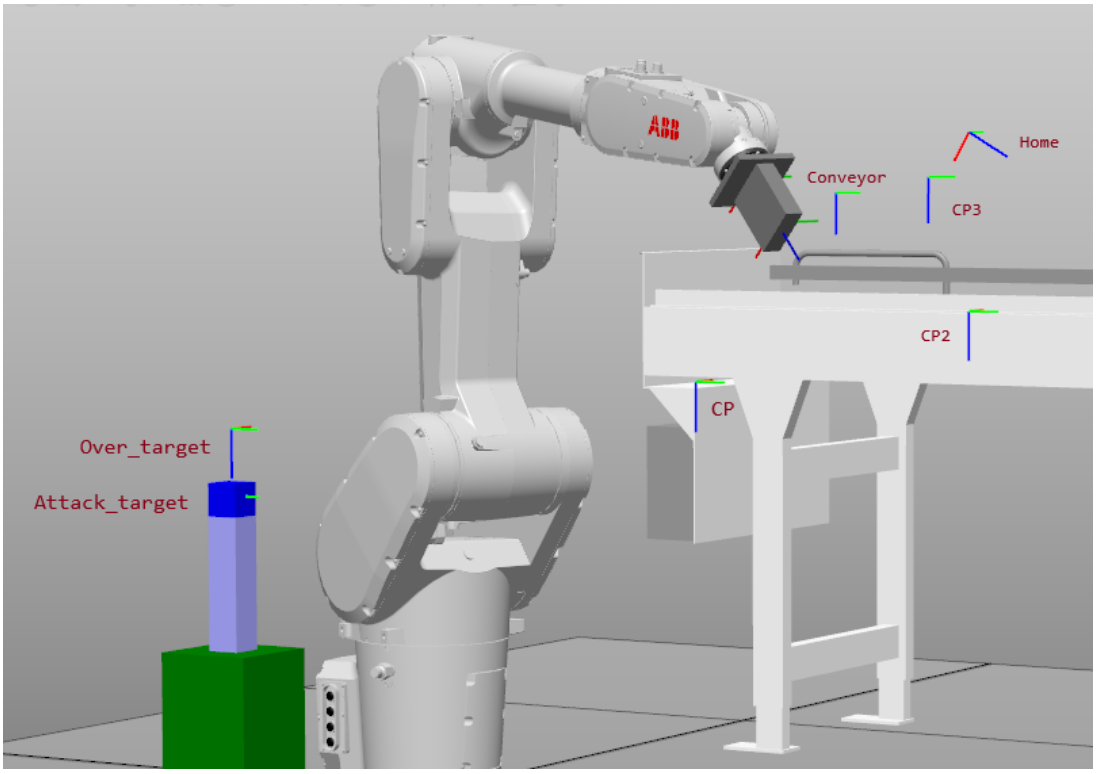
Rys. 2.4: Pełna forma *Station Logic*

Dodatkowo do projektu został dołączony *Smart Component* wraz z dwoma sygnałami. Służył on do debugowania oraz resetowania położenia obiektów roboczych. Został on pozostawiony w razie potrzeby dalszego rozwoju projektu.

3 Program RAPID

3.1 Wyznaczenie podstawowej trajektorii

W pierwszej kolejności wyznaczono *Targety*, na podstawie których później została utworzona ścieżka robota: Następnie



Rys. 3.1: Poglądowe spojrzenie na punkty trajektorii

utworzono ścieżkę obejmującą wszystkie elementy pracy robota - aktywacje sygnałów, rodzaje ruchów etc. Wszystkie ruchy oprócz tych bezpośrednio dotyczących łapania kostek oraz ich opuszczania zostały skonfigurowane jako ruch PNP (*Joint*). Natomiast reszta jako ruchy liniowe o dokładności *zone* = *fine*.

Zaprojektowana ścieżka była poprawna - nie było na niej kolizji ani ostrzeżeń od programu oraz robot zachowywał się zgodnie z oczekiwaniami.

3.2 Program RAPID

Po poprawnej konfiguracji 'podstawowej' trajektorii można było przejść do napisania programu umożliwiającego przeniesie wirtualnie każdej ilości obiektów w powtarzalny sposób bez potrzeby żmudnego planowania podobnych lub tych samych ruchów.

Punktami, które zmieniają swoje położenie w zależności od liczb n przeniesionych obiektów są targety *Over_target* oraz *Attack_target*. Dokładnie zmieniają one swoje położenie o wektor:

$$\vec{off} = -[0,\ 0,\ cube\_dimension] = [0,\ 0,\ -50mm]$$

Pełny kod RAPID:

```
MODULE Module1
    CONST robtarget Home:=[[980.884572681, 0, 1069],  [0.5, 0, 0.866025404, 0], [0,
    ↪  0, 0, 0],  [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget Target_10:=[[682.422, -300, 700], [1, 0, 0, 0], [0, 0, 0, 0],
    ↪  [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget Over_target:=[[75, -625, 630], [0, 0.707106781, 0.707106781, 0],
    ↪  [-1, 0, 0, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget Attack_target:=[[75, -625, 530], [0, 0.707106781, 0.707106781,
    ↪  0], [-1, 0, 0, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget CP:=[[682.422, -300, 700], [0, 0.707106781, 0.707106781, 0], [0,
    ↪  0, 0, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget CP3:=[[980.885, 0, 800], [0, 0.707106781, 0.707106781, 0], [0, 0,
    ↪  0, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget CP2:=[[700, 400, 1000], [0, 0.707106781, 0.707106781, 0], [0, 0,
    ↪  1, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];
    CONST robtarget Conveyor:=[[250, 900, 970], [0, 0.707106781, 0.707106781, 0], [0,
    ↪  0, 1, 0], [9E+09, 9E+09, 9E+09, 9E+09, 9E+09, 9E+09]];

    PROC main()
        !Constants
        CONST num BLOCKS :=5;         !Actual numbers of objects
        CONST num BlockNumber := 5;  !Number of objects used for simulation
        CONST num Offset := 50;

        !Start Sequence
        MoveJ Home,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        Set GripperClose;

        !Block Sequence
        IF BlockNumber > 0 AND BlockNumber <= BLOCKS THEN
            FOR i FROM 1 TO BlockNumber DO
                MoveBlock((i-1)*Offset);
            ENDFOR
        ENDIF

        !Closing Sequence
        MoveJ Home,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        Reset Dettach;
        Reset Attach;
        Reset Dettach;
        Reset SimReset;
        Reset GripperClose;
        Reset GripperOpen;
        WaitTime 3;

    ENDPROC
    PROC MoveBlock(num newOffset)
        !Calculating new positions for pick up
        VAR robtarget newOver := Over_target;
        VAR robtarget newAttack := Attack_target;

        newOver.trans := newOver.trans - [0, 0, newOffset];
        newAttack.trans := newAttack.trans - [0, 0, newOffset];

        !Move sequence
        MoveJ CP,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        MoveL newOver,v1000,fine,gripper_60mm_1\WObj:=Workobject_1;
        WaitTime 1;
        Set GripperOpen;
        Reset GripperOpen;
```

```
        WaitTime 1;
        MoveL newAttack,v1000,fine,gripper_60mm_1\WObj:=Workobject_1;
        WaitTime 1;
        Set Attach;
        Reset Attach;
        WaitTime 1;
        MoveJ newOver,v1000,fine,gripper_60mm_1\WObj:=Workobject_1;
        MoveJ CP,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        MoveJ CP3,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        MoveJ CP2,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        MoveJ Conveyor,v1000,fine,gripper_60mm_1\WObj:=Workobject_1;
        WaitTime 1;
        Set Dettach;
        Reset Dettach;
        WaitTime 1;
        MoveJ CP2,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
        MoveJ CP3,v1000,z100,gripper_60mm_1\WObj:=Workobject_1;
    ENDPROC
ENDMODULE
```

4 Imitacja fizyki

4.1 Fizyka obiektów

W *Robot Studio* można nadawać obiektom różne rodzaje fizyki, dzięki czemu można łatwiej modelować ich zachowanie.

Obiekty robocze skonfigurowano tak by miały fizykę typu *Dynamic*. Dzięki temu ich zachowanie było odwzorowane w sposób rzeczywisty - spadają, odbijają się etc.

Inne elementy stacji zostały skonfigurowane na fizykę typu *fixed*. Dzięki temu obiekty mogły oddziaływać z innymi obiektami, jednak pozostawały 'zastygnięte' w miejscu.

4.2 Taśma przesyłowa

By zaimitować działanie taśmy włączono opcję *Surface Velocity* - dzięki temu obiektom stykającym się z taśmą była nadawana stała prędkość.