

# Programowanie robota FANUC w środowisku symulacyjnym *Roboguide*\*

Mirela Kaczmarek<sup>†</sup>

Laboratorium Robotów Autonomicznych  
Wydział Elektroniki  
Politechnika Wrocławska

## Spis treści

<b>1 Cel ćwiczenia</b>	<b>1</b>
<b>2 Wymagania wstępne</b>	<b>1</b>
<b>3 Praca w środowisku <i>Roboguide</i></b>	<b>2</b>
3.1 Podstawowe informacje . . . . .	2
3.2 Programowanie robota . . . . .	3
3.2.1 <i>Simulation Program Editor</i> . . . . .	3
3.2.2 Wirtualny <i>Teach Pendant</i> . . . . .	5
3.2.3 <i>Simulation Program Editor</i> a wirtualny <i>Teach Pendant</i> . . . . .	6
<b>4 Zadania do wykonania</b>	<b>7</b>
4.1 Przygotowanie do pracy . . . . .	7
4.2 CELL1 . . . . .	7
4.3 CELL2 . . . . .	8
4.4 CELL3 . . . . .	8
<b>5 Sprawozdanie</b>	<b>8</b>

## 1 Cel ćwiczenia

Celem ćwiczenia jest zapoznanie z podstawami programowania robotów manipulacyjnych firmy FANUC przy wykorzystaniu środowiska symulacyjnego *Roboguide*.

## 2 Wymagania wstępne

Przed przystąpieniem do realizacji ćwiczenia należy:

---

\*Ćwiczenie laboratoryjne przeznaczone do realizacji w ramach kursu Robotyka (3) – data ostatniej modyfikacji: 22 lutego 2020.

<sup>†</sup>Katedra Cybernetyki i Robotyki

- zapoznać się z Instrukcją Obsługi i programowanie robota FANUC (R-30iA)
- zapoznać się z niniejszą instrukcją.

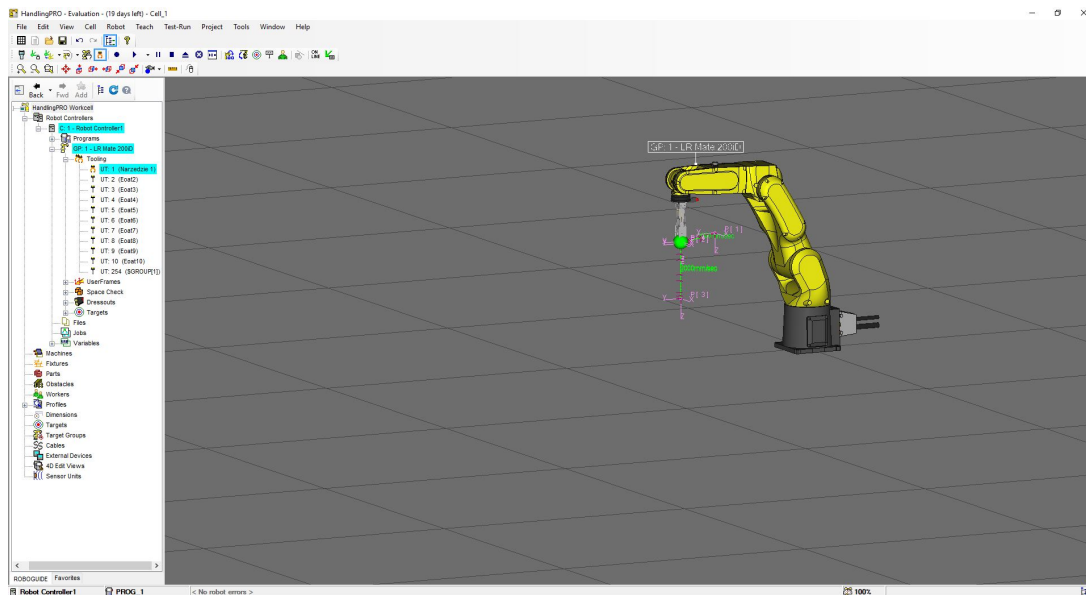
### 3 Praca w środowisku *Roboguide*

*Roboguide* to zintegrowane wirtualne środowisko przeznaczone do modelowania zrobotyzowanych stanowisk produkcyjnych oraz programowania i symulowania pracy robota. Oprogramowanie w pełni odzwierciedla rzeczywiste możliwości robotów firmy FANUC. Więcej informacji dostępnych na stronie

Na potrzeby ćwiczenia zostały stworzone i skonfigurowane komórki (*Cell*) zawierające robota i niezbędne elementy.

#### 3.1 Podstawowe informacje

Po uruchomieniu wybranej komórki zostanie wyświetlone okno programu widoczne na rysunku 1.



Rysunek 1: Widok komórki zawierającej robota

Widok obszaru roboczego można zmieniać przy użyciu różnych kombinacji klawiszy myszki. Możliwe jest wyświetlenie/schowanie okna podpowiedzi komend. W tym celu na belce programu (rysunek 2) należy wybrać ikonkę myszy [6]. Wówczas zostanie wyświetlone/schowane okno widoczne na rysunku 3. Do zmiany widoku obszaru roboczego można wykorzystać także przyciski znajdujące się na belce (rysunek. 2). Po lewej stronie okna znajduje się drzewo projektu zawierające wszystkie elementy należące do projektu (m. in. robota wraz z kontrolerem, programy, elementy komórki takie jak *Parts* czy *Fixture*).



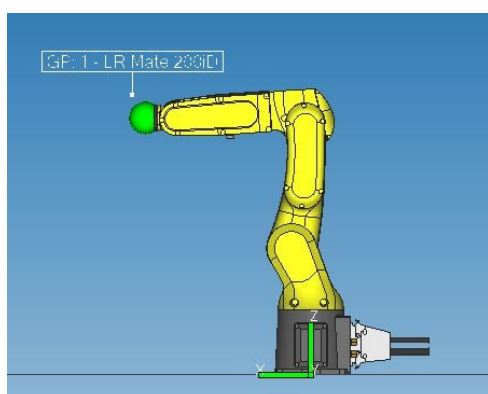
Rysunek 2: Belka programu.

3D World Mouse Commands		
View Functions	Object Functions	MoveTo Functions
Rotate view: RIGHT Drag	Move object, one axis: LEFT Drag triad axis	Move robotto surface: [CTRL] + [SHIFT] + LEFT-Click
Pan view: [CTRL] + RIGHT Drag	Move object, multiple axes: [CTRL] + LEFT Drag triad	Move robotto edge: [CTRL] + [ALT] + LEFT-Click
Zoom in/out: BOTH Drag (mouse Y axis)	Rotate object: [SHIFT] + LEFT Drag triad axis	Move robotto vertex: [CTRL] + [ALT] + [SHIFT] + LEFT-Click
Select object: LEFT-Click	Object property page: DOUBLE-LEFT Click	Move robotto center: [SHIFT] + [ALT] + LEFT-Click

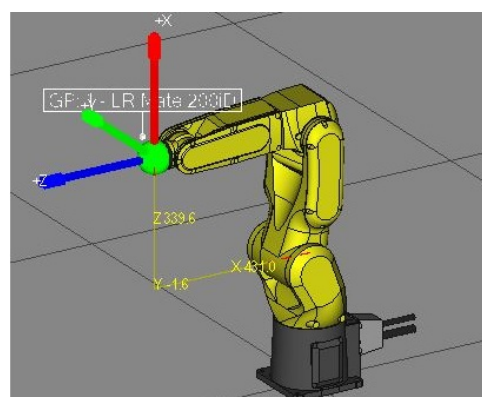
Rysunek 3: Okno dostępnych komend.

Tool center point (TCP) robota jest oznaczony za pomocą zielonej kuli (rysunek 4). Po kliknięciu w zieloną kulę pojawia się układ współrzędnych XYZ (rysunek 4b) umożliwiający przesuwanie końcówki robota wzdłuż osi oraz obrót. Po najechaniu na oś pojawia się symbol ręki wraz z oznaczeniem osi – wówczas możliwe jest przesuwanie końcówki robota wzdłuż osi. Aby obrócić robota wokół zadanej osi należy najechać na koniec osi – wówczas pojawią się strzałki symbolizujące obrót. Jeżeli kolor TCP zmieni się na czerwony, oznacza to, że robot nie jest w stanie osiągnąć zadanego punktu.

Ruch robotem jest możliwy albo poprzez przemieszczanie myszką układu TCP robota, albo z poziomu Teach Pendanta – wówczas sterowanie robotem wirtualnym odbywa się w sposób identyczny jak w przypadku rzeczywistego robota (więcej informacji w rozdziale 3.2.2).



(a) Widoczne TCP.



(b) TCP wraz z układem współrzędnych.

Rysunek 4: Przykładowy widok robota.

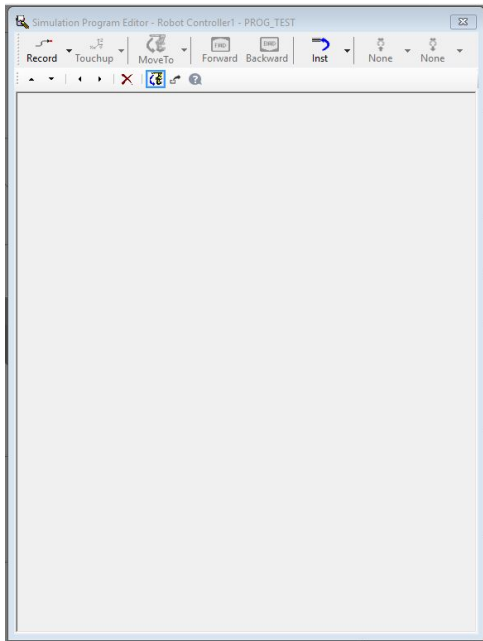
## 3.2 Programowanie robota

W środowisku *Roboguide* istnieje kilka możliwości pisania programów. Program można napisać przy pomocy edytora *Simulation Program Editor*, korzystając z wirtualnego programatora Teach Pendant lub przy wykorzystaniu zintegrowanego edytora języka KARREL. W dalszej części instrukcji omówiono dwa pierwsze sposoby. Program symulacyjny napisany przy użyciu edytora jest określony jako *Simulation Program*, natomiast program napisany przy użyciu Teach Pendanta jako *TP Program*.

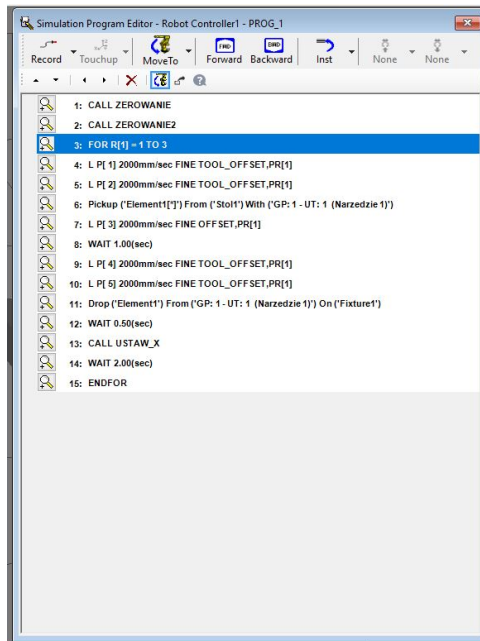
### 3.2.1 *Simulation Program Editor*

Wykorzystanie edytora *Simulation Program Editor* pozwala w prosty sposób zaprogramować ruch robota oraz interakcję pomiędzy elementami stanowiska (pobieranie oraz

odkładanie elementów). Opcja jest dostępna w zakładce *Teach* znajdującej się w na belce głównej okna programu. Po wybraniu *Add Simulation Program* wyświetli się okno, w którym należy wpisać wybraną nazwę programu. Po zawiedzeniu nazwy zostanie otwarty edytor, którego wygląd przedstawiono na rysunku 5. Dostęp do utworzonego programu jest możliwy m. in. z poziomu drzewa projektu (należy rozwinąć zakładki *Robot Controller* oraz zakładkę dostępnego kontrolera i wybrać zakładkę *Programs*).



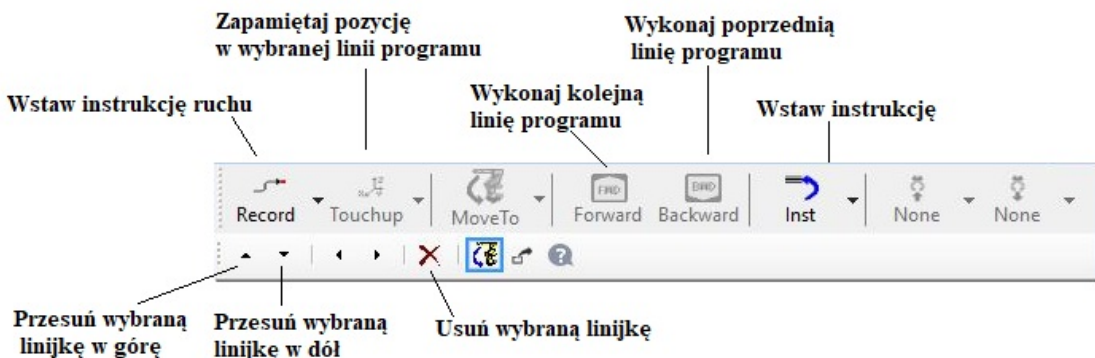
(a) Widok edytora bez programu.



(b) Widok edytora z przykładowym programem symulacyjnym.

Rysunek 5: *Simulation Program Editor* – okno edytora

Na rysunku 6 przedstawiono belkę edytora wraz z opisem niektórych ikon.



Rysunek 6: Opis belki *Simulation Program Editor*

W trakcie programowania można wykorzystać następujące opcje:

- *Record* – wstawianie instrukcji ruchu. Domyślnie wstawiany jest format ruchu *L*. Po rozwinięciu zakładki dostępne są inne formaty ruchu. Instrukcję ruchu można także zmodyfikować już po wstawieniu do programu. Możliwe jest także ustawienie OFFSETu dla punktu.

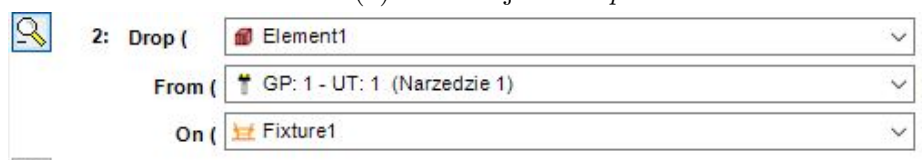
- *Touchup* – zapisanie pozycji w wybranej linii programu.
- *Move To* – przemieszczenie chwytaka do wskazanego elementu typu *Parts* (np. detalu do pobrania),
- *Forward/Backward* – wykonanie kolejnej/poprzedniej linii programu ( umożliwia wykonanie programu krok po kroku),
- *Inst* – wstawianie innych instrukcji (m. in. *Pickup/Drop* (umożliwiających pobieranie/odkładanie elementu), CALL, FOR).

Instrukcje *Pickup* wymaga wskazania jaki element ma zostać pobrany, skąd oraz przy pomocy jakiego narzędzia (rysunek reffig:pickup).

W przypadku instrukcji *Drop* konieczne jest wskazanie jaki element ma być położony, skąd, oraz gdzie ma zostać odłożony (rysunek 7b).



(a) Instrukcja *Pickup*.



(b) Instrukcja *Drop*

Rysunek 7: Przykładowe linie programu zawierające instrukcje *Pickup* i *Drop*


Możliwość pobierania/odkładania elementów jest zdefiniowana poprzez połączenia między elementami wskazane przy tworzeniu komórki.

Do uruchamiania/zatrzymywania symulacji służy grupa przycisków znajdujących się na głównej belce programu (rysunek 8).



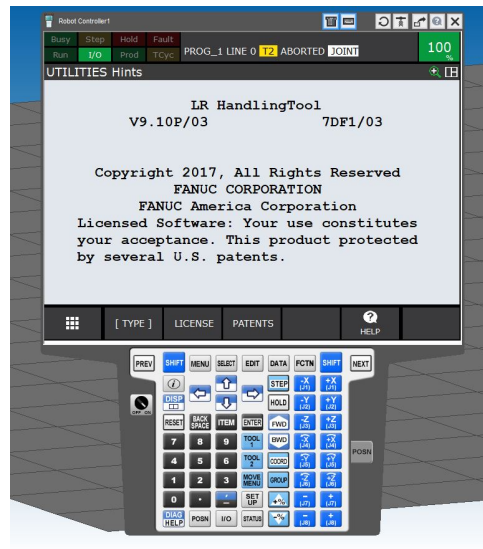
Rysunek 8: Przyciski wykorzystywane do obsługi symulacji.

### 3.2.2 Wirtualny *Teach Pendant*

Drugim sposobem tworzenia programów jest wykorzystanie wirtualnego Teach Pendanta (tzw. iPendant). Aby uruchomić wirtualnego Teach Pendanta należy wybrać ikonę  znajdującą się na belce (rysunek 8).

Po uruchomieniu wyświetli się wirtualny Teach Pendant (rysunek 9).

Praca odbywa się identycznie jak w przypadku rzeczywistego robota z wykorzystaniem rzeczywistego Teach Pendanta. W zależności od wersji wykorzystanego oprogramowania



Rysunek 9: Wirtualny Teach Pendant.

widok Teach Pendanta może być różny (m. in. różny rozkład przycisków), jednak ogólne zasady pracy nie ulegają zmianie.

Niezbędne przy wykonywaniu ćwiczenia informacje można znaleźć w instrukcji [1]. Przemieszczanie robota jest możliwe zarówno poprzez przesuwanie TCP, jak i przy użyciu odpowiednich przycisków na Teach Pendancie.

### 3.2.3 *Simulation Program Editor* a wirtualny Teach Pendant

Korzystanie z *Simulation Program Editor* pozwala na pisanie programów symulacyjnych i znacznie ułatwia pracę. Z poziomu edytora nie jest jednak możliwe modyfikowanie rejestrów robota. Aby zmodyfikować rejestry należy napisać *TP Program* za pomocą Teach Pendanta, a następnie wywołać go w programie symulacyjnym.

Z poziomu programu napisanego przy wykorzystaniu wirtualnego Teach Pendanta nie jest natomiast możliwe przedstawienie interakcji pomiędzy elementami (np. pobieranie i odkładanie detali). Aby operacje mogły być widoczne w symulacji należy stworzyć proste funkcje pobierania i odkładania detali za pomocą *Simulation Program Editor*.

Po otwarciu w Teach Pendancie programu napisanego w *Simulation Program Editor* instrukcje aktywne jedynie w środowisku *Roboguide* (takie jak m.in. podnoszenie/odkładanie przedmiotów) są zapisane w postaci komentarza i podświetlone na żółto. Po przeniesieniu na rzeczywistego robota nie będą aktywne.

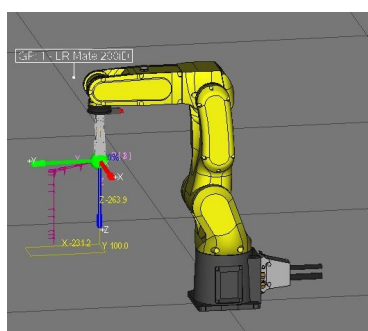


## 4 Zadania do wykonania

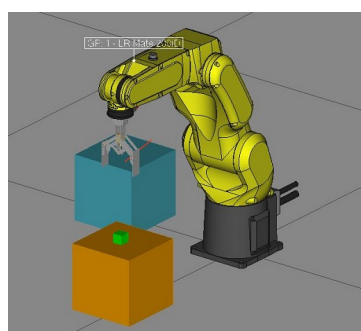
### 4.1 Przygotowanie do pracy

Aby rozpocząć pracę należy uruchomić wskazany przez prowadzącego komputer i zalogować się na konto ROBOGUIDE. Na pulpicie dostępne są pliki: CELL1.rgx, CELL2.rgx oraz CELL3.rgx, zawierające skonfigurowane komórki:

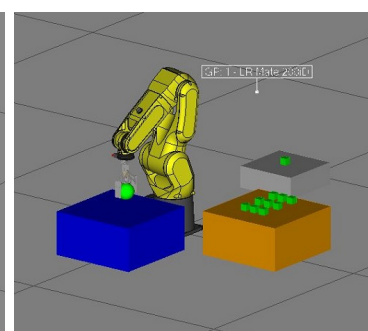
- CELL 1 – w komórce znajduje się robot wraz ze skonfigurowanym chwytakiem (widok komórki przedstawia rysunek 10a),
- CELL 2 – w komórce znajduje się robot, oraz dwa stoły wraz z detalem (widok komórki przedstawia rysunek 10b),
- CELL 3 – w komórce znajduje się robot, dwa stoły, oraz paleta elementów (widok komórki przedstawia rysunek 10c).



(a) Widok komórki CELL1



(b) Widok komórki CELL2



(c) Widok komórki CELL3

Rysunek 10: Dostępne komórki

Zadanie polega na odpowiednim zaprogramowaniu robota, aby wykonywał wskazane czynności. Poniżej przedstawiono propozycje zadań do realizacji w poszczególnych komórkach.

Po otwarciu wybranego pliku w przypadku pytania o nadpisanie należy wybrać opcję TAK.

### 4.2 CELL1

Poniższe zadania należy rozwiązać wykorzystując jedynie Teach Pendanta.

1. Napisać program przemieszczający manipulator między dwoma punktami:
  - w przestrzeni konfiguracyjnej,
  - w przestrzeni zadaniowej.

Zaobserwować różnice.

2. Stworzyć program kreślący okrąg końcówką roboczą manipulatora.
3. Napisać program poruszający efektozem pomiędzy 6 punktami (cztery niech zostaną zdefiniowane bardzo blisko siebie, a dwa względnie daleko). Ustawić dokładność na Fine, a następnie na CNT100. Zaobserwować różnice.

4. Napisać program kreślący kwadrat końcówką chwytaka, następnie zmodyfikować go wykorzystując Offset.
5. Zmodyfikować poprzedni program na podprogram StudKwadrat kreślący kwadrat końcówką manipulatora w płaszczyźnie XY, wywoływany z argumentem wielkości kwadratu.

### 4.3 CELL2


Napisać program w którym robot będzie przynosił element z jednego stołu na drugi. Element ma być pobierany z pomarańczowego stołu i odkładany na niebieski stół. Robot ma wykonać 5 cykli pracy.

Wskazówki:

- Zadanie należy wykonać wykorzystując *Simulation Program Editor* oraz wirtualny Teach Pendant.
- Należy wykorzystać pętlę w celu powtórzenia pięciu cykli pracy.
- Elementy zostały tak skonfigurowane, aby w miejscu pobranego z pomarańczowego stołu detalu po 8 sekundach pojawiał się nowy, a odłożony na niebieski stół element zniknął po 5 sekundach. Detale pojawiają się i znikają automatycznie podczas symulacji.

### 4.4 CELL3

Napisać program w którym będzie przenoszona (element po elemencie) cała paleta detali z pomarańczowego stołu na niebieski

- Zadanie należy wykonać wykorzystując *Simulation Program Editor* oraz wirtualny Teach Pendant.
- W zadaniu warto wykorzystać komendę OFFSET.
- Odległości pomiędzy elementami można zmierzyć wykorzystując narzędzie do mierzenia reprezentowane przez ikonę  znajdującą się na belce obok ikony myszy (rysunek 2).
- Należy zwrócić uwagę na konfigurację opcji w instrukcji *Pickup* i *Drop* w przypadku przenoszenia większej liczby elementów.

## 5 Sprawozdanie

Sprawozdanie z przebiegu ćwiczenia powinno zawierać:

- Imię i nazwisko autora, numer i termin grupy, skład grupy, temat ćwiczenia, datę wykonania ćwiczenia, datę dostarczenia sprawozdania.
- Cel ćwiczenia.
- Opis zrealizowanych zadań:
  - co dane zadanie miało na celu,



- sposób przeprowadzenia zadania,
  - opracowanie otrzymanych wyników,
  - uzyskany rezultat – wnioski.
- Wnioski końcowe.

## Literatura

- [1] Arkadiusz Mielczarek Katarzyna Zadarnowska, Jacek Jagodziński. *Obsługa i programowanie robota FANUC LR Mate 200iC*. Katedra Cybernetyki i Robotyki, Politechnika Wrocławska, Instrukcja Laboratorium Robotów Autonomicznych L1.5, 2018.