Sterowanie Robotów

Projekt 1.

Wprowadzenie do RobWorkStudio.

dr inż. Adam Wolniakowski

2018

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z podstawami obsługi środowiska do wizualizacji i symulacji robotów RobWorkStudio (https://www.robwork.dk). W trakcie ćwiczenia zrealizowane zostaną następujące zadania:

- wczytanie komórki roboczej (workcell),
- nauka podstaw obsługi interfejsu RobWorkStudio,
- ustawianie i odczyt pozycji robota,
- analiza struktury komórki roboczej,
- wyznaczenie przestrzeni roboczej manipulatora PA10.

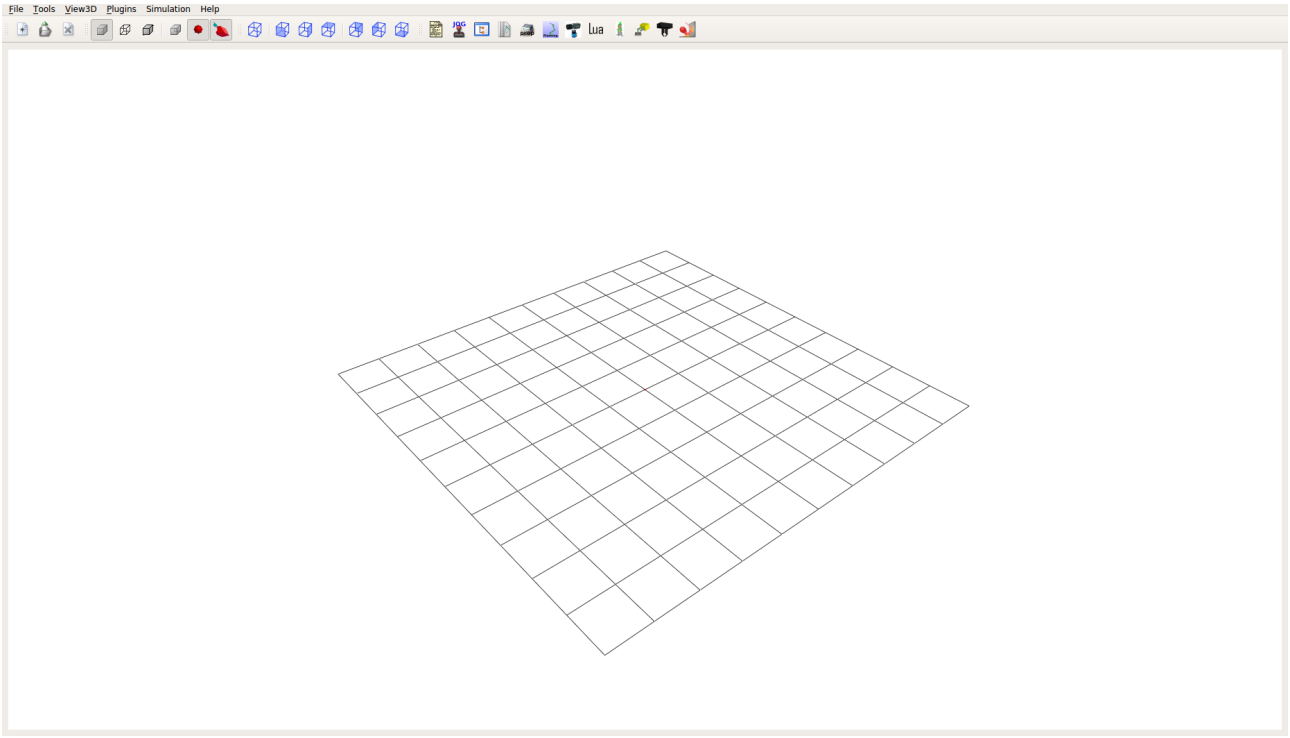
2. Wprowadzenie

2.1. RobWorkStudio

RobWork (https://www.robwork.dk) jest zbiorem bibliotek służących do wizualizacji, symulacji i sterowania robotów, opracowanym przez Maersk McKinney-Moller Instituttet uczelni Syddansk Universitet. Biblioteka zawiera m.in. narzędzia do:

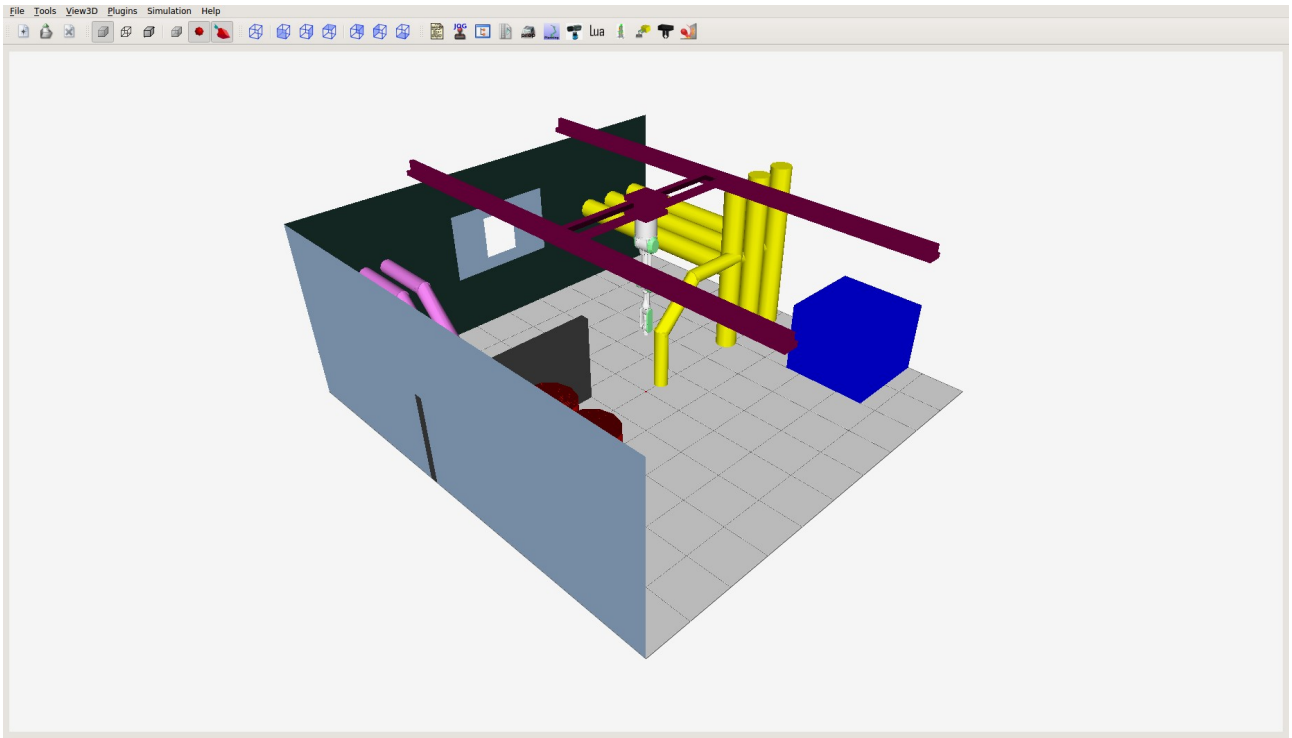
- modelowania kinematyki szeregowych, drzewiastych i równoległych manipulatorów,
- planowania ścieżek, wykrywania kolizji, kinematyki prostej i odwrotnej,
- kinematycznej i dynamicznej symulacji manipulatorów, kontrolerów i sensorów,
- symulacji chwytaków ssawnych i równoległych,
- użytkowania języków skryptowych (Lua, Python).

RobWorkStudio jest elementem biblioteki zapewniającym interfejs graficzny (rys. 1).



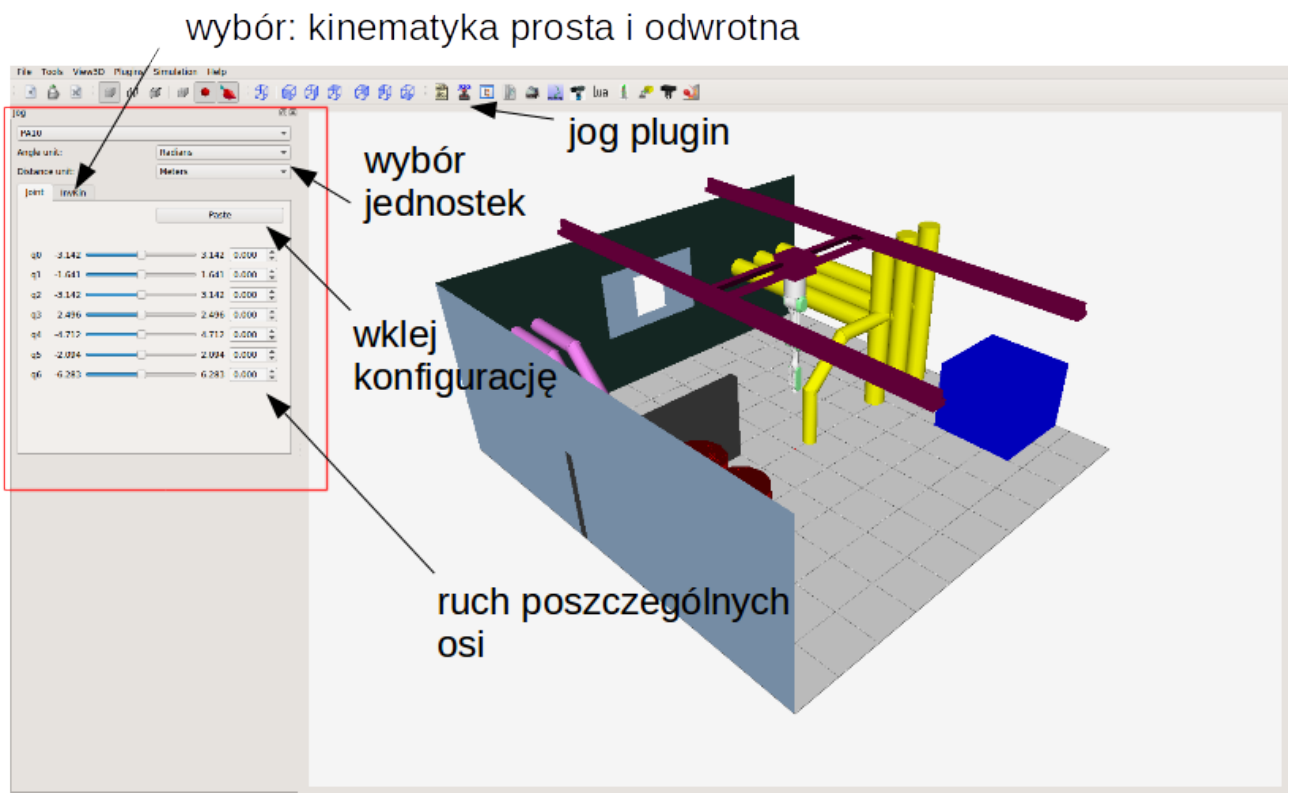
Rys. 1. Widok interfejsu programu RobWorkStudio.

Wczytanie komórki roboczej. Komórkę roboczą otwieramy przy pomocy przycisku *Open* lub wybierając w menu *File* opcję *Open*. Otwieramy pliki z rozszerzeniem **.wc.xml* (na przykład *Scene.wc.xml*) – rys. 2.



Rys. 2. Komórka robocza z robotem PA10 w środowisku RobWork.

Plugin Jog. Plugin ten służy do manipulacji poszczególnymi osiami robota (rys. 3).



Rys. 3. Plugin Jog.

Położenia poszczególnych osi można modyfikować przy pomocy suwaków lub wpisując wartości w odpowiednich polach. Jednostki w jakich wyrażone są położenie można modyfikować.

Plugin *Jog* posiada również zakładkę *InvKin*, gdzie można sterować ruchem robota w przestrzeni kartezjańskiej względem dowolnie wybranego układu współrzędnych (rys. 4).

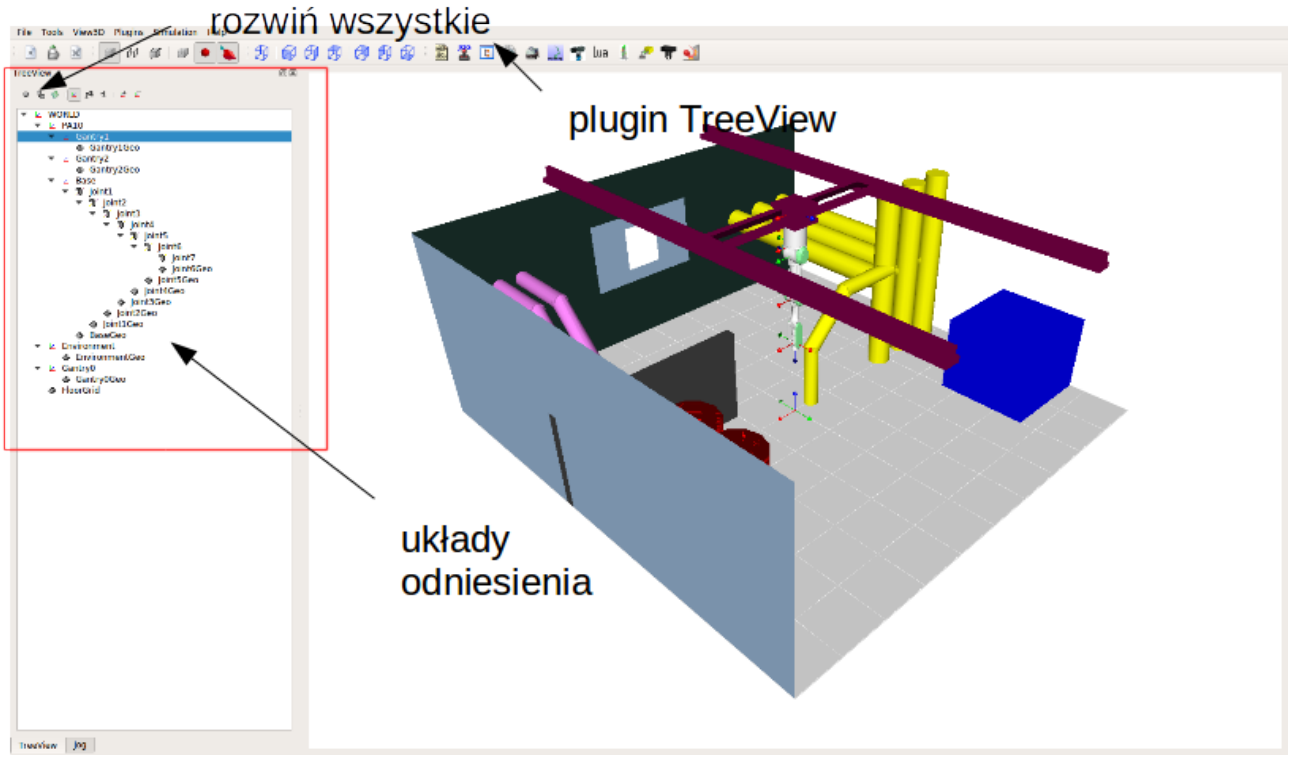
Korzystając z przycisku *Paste* możemy od razu przenieść robota do zadanej konfiguracji. Konfigurację podajemy w formacie np.: Q[7]{1, 1, 1, 1, 1, 1, 1}, gdzie kolejne położenia są podane w **radianach**.

Plugin TreeView. Plugin ten (rys. 5) pozwala na wizualizację struktury kinematycznej komórki roboczej. Można włączyć wyświetlanie dowolnego układu współrzędnych klikając na odpowiedniej pozycji prawym przyciskiem myszy i wybierając opcję *Show/remove frames*.

Można również włączyć/wyłączyć wyświetlanie poszczególnych obiektów klikając opcję *Toggle enabled*. Opcja *Read pose* umożliwia odczyt pozycji danego układu w stosunku do globalnego układu odniesienia WORLD. Odczyt ten będzie widoczny w oknie pluginu *Log* np.:

```
Gantry1:

Transform3D(Vector3D(0, 1.22465e-17, 2.2), Rotation3D(1, 0, 0, 0, -1.83697e-16,
1, 0, -1, -1.83697e-16))
```



Rys. 5. Plugin TreeView.

Plugin Lua. Plugin Lua (rys. 6) służy do podręcznego programowania z wykorzystaniem języka skryptowego Lua.

Aby móc skorzystać z Lua, należy w nim najpierw uzyskać odniesienie do okna RobWorkStudio:

```
studio = rws.getRobWorkStudio()
wc = studio:getWorkCell() -- uzyj aktualnej komorki
state = studio:getState() -- uzyj aktualnego stanu
```

Aby odczytać nazwę komórki roboczej:

```
name = wc:getName()
print(name)
```

Aby odczytać nazwę robota:

```
robot = wc:findDevice("PA10") -- znajdz robota w komorce
name = robot:getName()
print(name)
```

Aby odczytać konfigurację robota w danej chwili:

```
state = studio:getState() -- uaktualnij stan
q = robot:getQ(state) -- odczytaj konfiguracje w aktualnym stanie
print(q)
```

Aby ustawić nową konfigurację robota:

```
q = rw.Q(7, 2, 2, 2, 2, 2, 2, 2) -- nowa konfiguracja
robot:setQ(q, state) -- ustaw robota w nowej konfiguracji
studio:setState(state) -- uaktualnij RobWorkStudio
```

Aby odczytać pozycję dowolnego układu odniesienia:

```
f = wc:findFrame("PA10.Joint1") -- znajdz uklad
t = rw.worldTframe(f, state) -- wzgledem globalnego u.o.
print(t)

g = wc:findFrame("Gantry1")
t = rw.frameTframe(f, g, state) -- wzgledem innego ukladu odniesienia
print(t)
```

Uzyskanie części "translacyjnej":

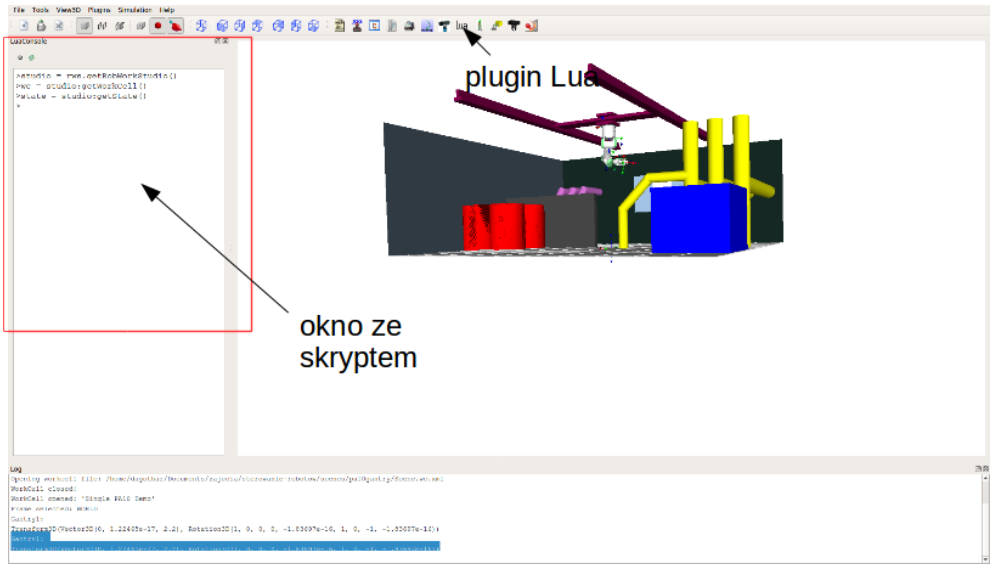
```
p = t:P()
print(p)
```

Uzyskanie części "rotacyjnej":

```
r = t:R()
print(R)
```

Przekształcenie z macierzy rotacji do kątów roll-pitch-yaw i na odwrót:

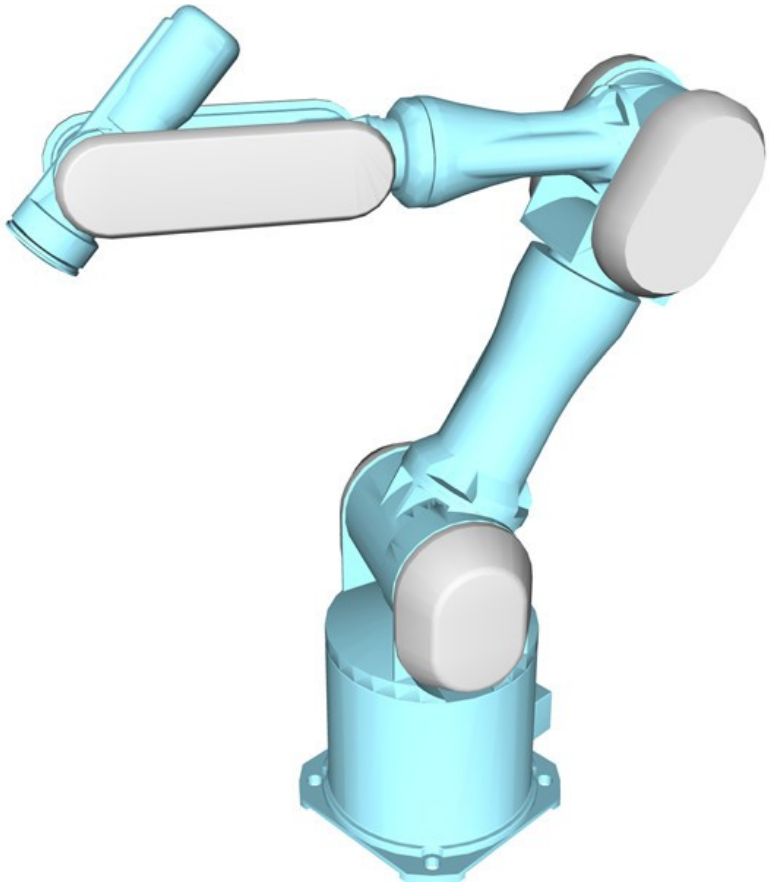
```
rpy = rw.RPYd(t:R())
rot = rpy:toRotation3D()
```



Rys. 6. Okno pluginu Lua.

2.2. Robot Mitsubishi PA10

Robot PA10 (rys. X) jest 7-osiowym manipulatorem wyprodukowanym przez firmę Mitsubishi. Robot wykorzystywany jest w zastosowaniach dydaktycznych i przemysłowych (w tym przy wspólnej pracy z ludźmi).



Rys. X. Model robota PA10 firmy Mitsubishi.

Dane techniczne:

Rozmiar:	Controller: 240(B)x400(L)x200(H) mm
Waga:	Robot: 40 Kg; Controller: 18 Kg
Zasilanie:	Controller: AC 100-240V 50/60 Hz, under 1.5kVA
Prędkość:	max. Sa, S2: 57deg/s; S3, E1: 114deg/s; E2,W1,W2: 360deg/s
Napędy:	Robot: AC servomotors
Sensory:	Position Sensors: Brushless resolver (absolute position)
Producent:	Mitsubishi Heavy Industries, Ltd
Model:	PA10-7CE-ARM
Ilość członów:	7
Typy członów:	R-P-R-P-R-P-R
Udźwig:	10 kg
Powtarzalność:	±0.1 mm

3. Ćwiczenia

3.1. Struktura kinematyczna robota PA10.

Narysuj strukturę kinematyczną robota PA10 uwzględniając typy kolejnych przegubów (obrotowy/przesuwny).

3.2. Struktura kinematyczna komórki roboczej.

Przedstaw rysunek (zrzut ekranu) zawierający wszystkie układy odniesienia występujące w komórce roboczej (pomiń elementy geometrii "Geo"). Podpisz na rysunku ich nazwy i pozycje w stosunku do układu WORLD dla wybranej konfiguracji robota (zapisz konfigurację). Orientację należy podać w postaci kątów RPY.

3.3. Wyznacz przestrzeń roboczą manipulatora PA10.

Utwórz skrypt języka LUA, w którym ustawisz robota PA10 w 1000 losowych pozycji. Dla każdej z osi wyszukaj limity pozycji w pliku *pa10.wc.xml* (przelicz na radiany). Dla każdej konfiguracji należy zmierzyć pozycję końcówki robota względem globalnego układu odniesienia (tylko translację) i zapisać ją w postaci pliku *.csv, np:

0.1, 0.2, 0.3

0.2, 0.4, 1.22

...

Otrzymany plik należy wczytać w Matlabie i wygenerować wykres 3D uzyskanych punktów.

Wskazówki

- pętla w języku LUA:

```
for i = 1,1000,1 do
  print("i=" .. i)
  --...
end
```

- funkcja czekania w języku LUA:

```
local clock = os.clock
function sleep(n)  -- seconds
  local t0 = clock()
  while clock() - t0 <= n do end
end
```

- jak uzyskać położenie końcówki w danym stanie:

```
t = rw.worldTframe(robot:getEnd(), state)
```

- jak wylosować konfigurację robota (zmodyfikuj tę funkcję tak, aby móc losować pomiędzy dwoma dowolnie zadanymi wartościami):

```
function los()
  return math.random() * 6.28 - 3.14;
end

q = rw.Q(7, los(), los(), los(), los(), los(), los(), los())
```

- jak otworzyć, pisać do pliku i go zamknąć:

```
file = io.open("robo.csv", "w")
--...
file:write(t:P()[0] .. ", " .. t:P()[1] .. ", " .. t:P()[2] ..
"\n")
--...
file:close()
```

- jak zapisać widok z poziomu języka LUA:

```
studio:saveViewGL("nazwa_" .. i .. ".png")
```

- jak wygenerować wykres w Matlabie:

```
data = csvread('plik.csv');
figure;
scatter3(data(:, 1), data(:, 2), data(:, 3));
axis equal;
```

4. Sprawozdanie

Sprawozdanie powinno zawierać:

- cel i zakres ćwiczenia,
- treści zadań,
- rysunek ze strukturą kinematyczną robota,
- opis struktury komórki roboczej,
- skrypt napisany dla ćw. 3.3,
- uzyskany obraz przestrzeni roboczej.