

Laboratorium Podstaw Robotyki

Politechnika Poznańska
Instytut Automatyki i Robotyki

ĆWICZENIE 6

PODSTAWY OBSŁUGI I PROGRAMOWANIA MANIPULATORA STÄUBLI TX60L

Celem ćwiczenia jest zapoznanie się z programowaniem sterowania manipulatorem przemysłowym Stäubli TX60L, który jest umieszczony na dedykowanym stanowisku laboratoryjnym. Ćwiczenie pozwala na poznanie możliwości ruchowego robota, podstawowych komend związanych z realizacją ruchu manipulatora oraz zapoznanie się ze sposobem definiowania zadań robota z wykorzystaniem panelu operatorskiego oraz dedykowanego oprogramowania narzędziowego.

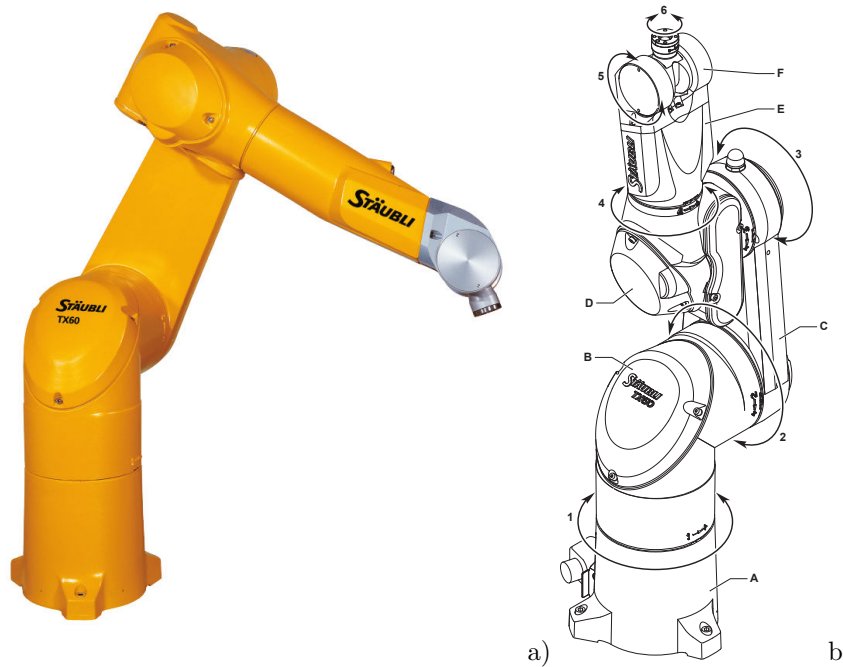
1 System sterowania z manipulatorem Stäubli

System sterowania z manipulatorem Stäubli TX60L składa się z kilku podstawowych elementów:

- ramienia manipulacyjnego TX60L (element wykonawczy systemu) – rys. 1,
- sterownika CS8C (obliczenia kinematyki, generator trajektorii, regulatory pozycji /prędkości /przyspieszenia osi napędów, komunikacja sieciowa, obsługa sygnałów zewnętrznych) – rys. 2,
- ręcznego panelu sterującego – MCP (Manual Control Panel) – rys. 3,
- interfejsu wyboru trybu pracy – WMS (Working Mode Selector) – rys. 4,
- dedykowanego oprogramowania narzędziowego Stäubli Robotics Suite (edycja kodu programu i testowanie programów, wizualizacja działania manipulatora).

Sterownik CS8C odpowiedzialny jest za realizację w czasie rzeczywistym programów sterujących ruchami manipulatora oraz komunikację z urządzeniami zewnętrznymi stanowiącymi wyposażenie stanowiska roboczego. Generowane trajektorie zadane są dyskretyzowane i przesyłane do sterowników osi cyklicznie, z okresem wynoszącym $T_{pt} = 4ms$. Programy sterujące tworzone są w dedykowanym języku VAL 3 [3]. Jest to interpretowany, wielozadaniowy język programowania¹, który pozwala na wytworzenie oprogramowania obsługującego praktycznie dowolny obszar zastosowań robota manipulacyjnego. Język VAL 3 łączy w sobie podstawowe cechy standardowych języków programowania wysokiego poziomu z rozwiązaniami specyficznymi dla sterowania robotów, takimi jak obsługa wielu zadań w czasie rzeczywistym, realizacja ruchów ramienia z różnymi interpolacjami czy komunikacja z urządzeniami wejścia/wyjścia.

¹Język interpretowany to język programowania, który zazwyczaj jest implementowany w formie interpretera, a nie kompilatora. Program w języku interpretowanym nie jest kompilowany, lecz jest przechowywany w postaci kodu źródłowego i dopiero podczas uruchomienia wczytywany, interpretowany i wykonywany przez interpreter języka.



Rysunek 1: a) Manipulator Stäubli TX60L, b) Oznaczenia kolejnych złączy, ich osi obrotów oraz podstawowych elementów składowych mechaniki manipulatora: podstawę (A), bark (B), ramię (C), łokieć (D), przedramię (E) i nadgarstek (F) [1].

1.1 Uruchamianie systemu

W celu załączenia systemu należy przekręcić duże pokrętko znajdujące się w prawym dolnym narożniku szafy sterowniczej o 90 stopni w prawą stronę (przełącznik w położeniu “1” – patrz rys. 2). Po tej czynności następuje załączenie zasilania, ładowanie i inicjalizacja całego systemu, przy czym ramię manipulatora nie zostaje zasilone i pozostaje w pozycji początkowej. Załączenie zasilania napędów ramienia wymaga przytrzymania przycisku zezwolenia w pozycji środkowej i wciśnięciu przycisku załączenia – oba na panelu MCP (patrz opis dalej).

Wyłączenie systemu polega na przekręceniu pokrętła załączającego o 90 stopni w stronę lewą (przełącznik w położeniu “0”). Po tej czynności automatycznie wykonywana jest procedura zapisu stanu i zamykania systemu. Wszystkie dane i stan systemu z chwili poprzedzającej wyłączenie zostaną zachowane i przywrócone po ponownym załączeniu zasilania.

Czerwone przyciski wyłączenia awaryjnego (tzw. E-STOP – *Emergency STOP*), umieszczone na ręcznym panelu sterującym, interfejsie wyboru trybu pracy oraz na osłonach przestrzeni roboczej. Służą do AWARYJNEGO, natychmiastowego zatrzymania manipulatora z jednoczesnym odłączeniem zasilania napędów. W przypadku zagrożenia bezpieczeństwa ludzi i samego manipulatora niezwłocznie nacisnąć przycisk E-STOP!!!

UWAGA! Nie wchodź w przestrzeń roboczą manipulatora przy włączonych napędach! Kontakt robota z człowiekiem grozi poważnym uszkodzeniem ciała!

Ozn. parametru	Nazwa parametru	Wartość
m_w	masa własna	52.5[kg]
m_u	udźwig nominalny	2[kg]
m_{umax}	udźwig maksymalny	3.7[kg]
v_{kmax}	maksymalna prędkość liniowa końcówki roboczej	10.6[m/s]
v_{kmaxTP}	maksymalna prędkość liniowa końcówki przy sterowaniu panelem	0.25[m/s]
Δ_k	powtarzalność w przestrzeni kartezjańskiej	0.03[mm]

Tablica 1: Wybrane parametry i osiągi manipulatora Stäubli TX60L.



Rysunek 2: Widok szafy sterowniczej ze sterownikiem CS8C do manipulatora Stäubli TX60L, 1 – załączanie urządzenia [2].

1.2 Tryby pracy

Wybór trybu pracy jest przeprowadzany z interfejsu operatora WMS (rys. 4). Przełącznik posiadający 3 różne położenia umożliwia wybranie jednego z trzech dostępnych trybów (lokalny, zdalny lub ręczny, patrz rys. 5). Wybrany tryb jest wskazywany przez lampkę kontrolną na WMS i MCP.

1.2.1 Tryb RĘCZNY

Tryb ręczny jest wykorzystywany w następujących przypadkach:

- ręczne przemieszczanie ramienia – operator steruje ruchami ramienia za pomocą panelu sterowania ręcznego MCP – Manual Control Panel,
- test / aktualizacja aplikacji – w takim przypadku ruchy ramienia są sterowane przez program.

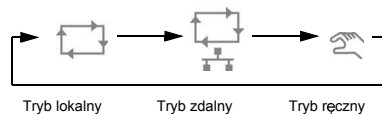
Tryb ręczny umożliwia wykonywanie ruchów robota z niską prędkością (maksymalnie 250 mm/s). Przesunięcia ręczne są wykonywane za pomocą przycisków sterowania ruchem (patrz opis dalej). Ruchy zaprogramowane mogą być wykonywane wyłącznie po wciśnięciu przycisku Move/Hold. Ruch jest przerywany, gdy tylko przycisk zostanie zwolniony (patrz opis dalej). Zasilanie ramienia pozostaje włączone jedynie, kiedy przycisk zezwolenia na ruch (#) jest przytrzymywany w położeniu



Rysunek 3: Panel sterowania ręcznego – MCP (Manual Control Panel), (#) – trópołożeniowy przycisk zezwolenia na pracę ramienia [2].



Rysunek 4: Interfejs wyboru trybu pracy – WMS (Working Mode Selector) [2].



Rysunek 5: Tryby pracy manipulatora Staubli TX60L [2].

środkowym. Przycisk (#) znajduje się na spodniej stronie ręcznego panelu sterowania MCP (patrz rys. 3). Po wybraniu trybu ręcznego, pozostałe tryby robocze nie są aktywne, a przesunięcia nie mogą być inicjowane z jakiegokolwiek urządzenia zewnętrznego.

1.2.2 Tryb LOKALNY

Tryb lokalny umożliwia przemieszczanie robota bez interwencji użytkownika, z maksymalną prędkością określoną dla danej aplikacji. Przesunięcia są wykonywane zgodnie z kolejnymi krokami zapisanymi w programie. Uruchomienie wykonywania programu odbywa się z wykorzystaniem MCP.

NIEBEZPIECZEŃSTWO: Kiedy robot znajduje się w trybie lokalnym, żadne osoby nie mogą znajdować się wewnątrz obszaru roboczego ramienia.

Robot jest gotowy do pracy, jeżeli spełnione zostały następujące warunki:

- zasilanie ramienia jest włączone,

- aplikacja określająca ruchy robota została wprowadzona do pamięci i jest wykonywana.

Polecenie rozpoczęcia ruchu jest wydawane za pośrednictwem MCP, poprzez przycisk `Move/Hold` (patrz opis dalej). Wszystkie ruchy ramienia są sterowane wyłącznie przez aplikację. Operator może jedynie zatrzymać lub ponownie uruchomić wykonywanie przemieszczeń oraz regulować ich prędkość za pomocą przycisku `-/+` (na rys. 6 oznaczony numerem 6).

1.2.3 Tryb ZDALNY

Zasady funkcjonowania trybu zdalnego są takie same, jak w przypadku trybu lokalnego. Uruchomienie wykonywania programu odbywa się za pośrednictwem urządzenia zewnętrznego. Zasady działania w trybie zdalnym są następujące:

- ramię jest zasilane za pośrednictwem systemu zewnętrznego (sterownik, zewnętrzna jednostka MCP) przy wykorzystaniu wejścia sygnału cyfrowego lub przy użyciu polecenia `enablePower` (patrz podręcznik języka VAL 3 [3]),
- polecenie wykonania ruchu `Move/Hold` może być wydawane automatycznie, pod warunkiem, że zasilanie ramienia jest włączone (patrz polecenie `autoConnectMove` w instrukcji języka VAL 3),
- przycisk `Move/Hold` może być nieaktywny, w zależności od uprawnień użytkownika,
- przycisk wyłączania zasilania może być nieaktywny, w zależności od uprawnień użytkownika,
- do sygnału systemowego `enablePower` musi zostać wcześniej przypisane wejście cyfrowe (impuls 200 ms sygnału `enablePower` powoduje włączenie zasilania; kolejny impuls sygnału `enablePower` powoduje wyłączenie zasilania).

NIEBEZPIECZEŃSTWO: Kiedy robot znajduje się w trybie zdalnym, żadne osoby nie mogą znajdować się wewnątrz obszaru roboczego ramienia.

1.3 Panel ręcznego sterowania

Zmianę położenia ramienia robota można realizować za pomocą panelu sterowania ręcznego MCP. Na rys. 6 oznaczono podstawowe grupy przycisków, wykorzystywanych podczas obsługi systemu.

Znaczenie poszczególnych grup przycisków jest następujące:

- (1) Tryb roboczy
Wybrany tryb pracy za pomocą interfejsu operatora WMS jest wyświetlony w pobliżu tego przycisku. Wybrany tryb jest również wyświetlony na panelu przednim WMS.
- (2) Przycisk włączania zasilania ramienia
Przycisk podświetlany umożliwiający włączanie i wyłączanie zasilania manipulatora. Jeżeli lampka kontrolna w kolorze zielonym świeci się, oznacza to, że zasilanie jest doprowadzone. Aby zasilanie ramienia mogło zostać włączone w trybie ręcznym, przycisk zezwolenia (#) (11) musi być przytrzymywany w pozycji wciśniętej środkowej.
- (3) Wyłącznik awaryjny
Przycisk wyłączania awaryjnego może być używany, aby zatrzymać urządzenie w sytuacji zagrożenia bezpieczeństwa operatora.



Rysunek 6: Panel sterowania ręcznego – MCP z zaznaczonymi podstawowymi grupami przycisków [2].

- (4) Przyciski ruchu
Przyciski te są aktywne w trybie ręcznym i umożliwiają sterowanie ruchami ramienia według oznaczeń osi lub współrzędnych, w zależności od wybranego trybu przemieszczania (patrz opis dalej).
- (5) Przyciski wyboru trybu przemieszczania
Kiedy ramię jest zasilane i wybrany jest tryb ręczny, każdy z 4 przycisków umożliwia wybranie trybu przemieszczania (patrz opis dalej). Lampka kontrolna przycisku wskazuje bieżący tryb.
- (6) Przycisk regulacji prędkości
Przycisk umożliwia zmianę prędkości ruchów ramienia manipulatora. Wskaźnik monitora aktualnej prędkości jest wyświetlany na pasku stanu w górnym wierszu na ekranie panelu sterowania ręcznego MCP i wyrażony jest w % ustalonej prędkości maksymalnej w danym profilu operatora: sterowania ręcznego lub programowego. Gdy wyświetlany jest w nawiasie [], to profil operatora nie pozwala na zmianę monitora prędkości.
- (7) Przyciski menu tekstowych
Są wykorzystywane w celu wyboru poleceń wyświetlanych nad przyciskami.
- (8) Przyciski alfanumeryczne
Przyciski służą do wprowadzania danych aplikacji.
- (9) Przyciski interfejsu i nawigacji
Za pomocą tych przycisków można poruszać się lub wybierać pozycje z menu wyświetlanego w głównej części panelu programowania.

- (10) Funkcje sterowania aplikacjami

Przyciski używane w celu uruchamiania / zatrzymywania aplikacji oraz zatwierdzania ruchów ramienia. Przycisk **Stop** służy do zatrzymywania bieżącej aplikacji. Przycisk **Run** umożliwia uruchomienie aplikacji. Przycisk **Move/Hold** umożliwia wykonanie zaprogramowanych ruchów ramienia w trybie ręcznym. Ramię zatrzyma się na zaprogramowanej trajektorii natychmiast, kiedy przycisk zostanie zwolniony. W trybie lokalnym oraz zdalnym, wciśnięcie przycisku **Move/Hold** powoduje zatrzymanie ruchów, a robot przechodzi w tryb pauzy. Kolejne wciśnięcie przycisku powoduje ponowne uruchomienie.

- (11) Włączanie/wyłączenie zezwolenia na ruch ramienia (ZEZWOLENIE NA PRACĘ RAMIENIA)

Przycisk (#) posiada trzy możliwe położenia i steruje stykami, które są: - otwarte, kiedy przycisk nie jest włączony; - zamknięte, kiedy przycisk znajduje się w położeniu środkowym; - otwarte, kiedy przycisk jest wciśnięty do końca i przytrzymywany przez użytkownika. Styki pozostają otwarte aż do zwolnienia przycisku. Przycisk używany jest do umożliwienia włączenia zasilania w trybie ręcznym, ale tylko wtedy, kiedy znajduje się w położeniu pośrednim. Pozostałe 2 położenia uniemożliwiają włączenie zasilania lub powodują jego wyłączenie.

- (12) Przyciski aktywacji wyjść cyfrowych

W trybie ręcznym, przyciski te umożliwiają zmianę stanu wyjść cyfrowych, które zostały do nich przypisane (najczęściej sterowanie chwytakiem). Ręczne sterowanie chwytakiem zostało przypisane do przycisku 1.

- (13) Przyciski Jog

Przyciski są aktywne w trybie ręcznym i umożliwiają alternatywne sterowanie ruchami ramienia według opisu osi lub współrzędnych, w zależności od wybranego trybu przemieszczania.

1.4 Tryby ruchów ręcznych

W trybie ręcznym użytkownik może zmieniać położenie ramienia manipulatora w jednym z czterech trybów (rys. 7):

- **JOINT** – ruch w złączach czyli w przegubach (zawsze dostępny),
- **FRAME** – ruch w wybranym kartezjańskim układzie współrzędnych (domyślny to układ podstawowy *world*), wymagana otwarta aplikacja,
- **TOOL** – ruch w układzie wybranego narzędzia (domyślne narzędzie to *flange*), wymagana otwarta aplikacja,
- **POINT** – ruch do zdefiniowanego punktu w układzie zgodnym z trybem wskazania (podświetlenia nazwy punktu); tryb ten umożliwia poruszanie ramieniem robota w trybie ręcznym pomiędzy zdefiniowanymi punktami przestrzeni kartezjańskiej i/lub w złączach.



Rysunek 7: Tryby ruchów ręcznych.

Po wybraniu trybu przemieszczania spośród dostępnych **JOINT**, **FRAME**, **TOOL** lub **POINT**, zaświeci się odpowiadająca mu lampka kontrolna. Aby poruszyć ramieniem, należy wcisnąć jeden z przycisków przemieszczania z grup (4) lub (13) (rys. 6) lub - w trybie **POINT**, po wskazaniu punktu

z aplikacji - za pomocą przycisku *Move/Hold*. Zamiar wywołania ruchu w trybie ręcznym poprzedzić ustawieniem prędkości ramienia za pomocą przycisku kontroli prędkości, zalecana prędkość to do 50%. Wybór jednego z trybów przemieszczania powoduje automatyczne wyświetlenie strony sterowania ręcznego na ekranie MCP. Aby wyjść z tej strony, należy wcisnąć przycisk *Esc*. Aby powrócić na tę stronę, należy ponownie wybrać tryb przemieszczania (*JOINT*, *FRAME*, *TOOL*, *POINT*).

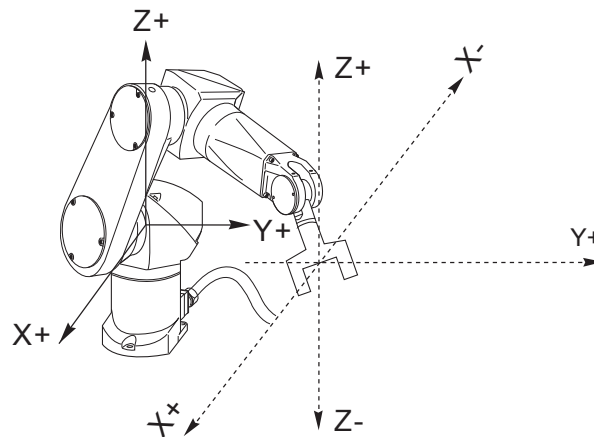
1.4.1 Przemieszczanie w trybie przegubowym (*JOINT*)

Po włączeniu zasilania ramienia, wcisnąć przycisk *Joint* w MCP. Zaświeci się lampka kontrolna. Przyciski z grupy (4) umożliwiają wykonywanie ruchów w trybie przegubowym (*JOINT*), wokół poszczególnych osi (1, 2, 3, itd. - patrz rys. 1). Przemieszczenia wykonywane są w kierunkach do przodu (zestaw przycisków oznaczonych symbolem *+*) lub do tyłu (zestaw przycisków oznaczonych symbolem *-*). Możliwe jest równoczesne przemieszczanie wielu osi. W takim przypadku świeci się jedynie lampka kontrolna ostatniego wciśniętego przycisku osi (*+* lub *-*) oraz lampka *Jog* (13). Po wciśnięciu jednego z przycisków *SEL* grupy *Jog* (13), numer wybranej osi zmienia się, a odpowiadająca jej lampka kontrolna w grupie (4) zaświeca się. Po wciśnięciu jednego z przycisków *-/+* sekcji *Jog*, wybrana oś ulega przemieszczeniu.

1.4.2 Przemieszczanie w trybie kartezjańskim (*FRAME*, *TOOL*)

Po włączeniu zasilania ramienia, wcisnąć przycisk *Frame* lub *Tool* w MCP. Zaświeci się lampka kontrolna. Wciskając przyciski sterowania ruchem (4) lub jeden z przycisków grupy *Jog* (13), możliwe jest wykonywanie przemieszczeń wzdłuż – przyciski *X*, *Y*, *Z* – i wokół – przyciski *RX*, *RY*, *RZ* – trzech osi bieżącego układu odniesienia (domyślnie względem układu o nazwie zmiennej *World* typu *Frame*). Przemieszczenia wykonywane są w kierunku do przodu (zestaw przycisków oznaczonych symbolem *+*) lub do tyłu (zestaw przycisków oznaczonych symbolem *-*). Jeżeli wciśnięty został przycisk *Tool*, to przemieszczenia są wykonywane wzdłuż / wokół osi bieżącego układu powiązanego z narzędziem (domyślnym jest zmienna o nazwie *flange* typu *tool*).

Przywiązanie charakterystycznych kartezjańskich układów współrzędnych (układ podstawowy, układ narzędzia) przedstawiono na rys. 8.



Rysunek 8: Sposób przywiązania osi kartezjańskich układów współrzędnych dla manipulatora TX60L [2].

1.4.3 Przemieszczanie w trybie *POINT*

Tryb *POINT* umożliwia sterowanie przemieszczeniami tylko do punktów zapamiętanych w danej aplikacji. Stąd aby wyświetlić punkty aplikacji, należy najpierw wybrać w niej narzędzie. Jeżeli wy-

brany zostanie tryb przemieszczania POINT, za pomocą przycisku Point, to na ekranie wyświetlane będzie okno zawierające zapamiętane wcześniej punkty z rozróżnieniem ich typów Point/Joint oraz sposób dojazdu do nich czyli MODE (opisany dla aktualnie podświetlonego punktu). Sposób realizacji dojazdu do punktu w trybie POINT może przyjąć jeden z trzech wariantów:

- **Line** – przemieszczenie do punktu docelowego odbywa się po linii prostej (tylko dla punktów typu Point),
- **Joint** – przemieszczenie odbywa się według najkrótszej zmiany współrzędnych osiowych manipulatora (dla punktów typu Point oraz Joint),
- **Align** – przemieszczenie osi Z narzędzia (kierunek natarcia narzędzia) do wyrównania równoległego z najbliższą osią bieżącego układu odniesienia (zwykle **Frame: World**). Tu tylko nadgarstek wykonuje obrót do najbliższej osi bieżącego układu odniesienia, bez przesunięcia.

Kolejne przełączanie tych wariantów odbywa się za pomocą menu kontekstowego trybu POINT, pod przyciskiem F7, **Mode**. Wcześniej zapamiętane punkty widoczne są w zależności od wybranego sposobu realizacji dojazdu.

Robot rozpocznie dojazd do wybranego punktu po jego podświetleniu (użycie strzałek) i po wciśnięciu przycisku Move/Hold. Znalezienie się robota w punkcie wybranym jako docelowym jest sygnalizowane przed nazwą punktu przez sygnaturę @.

1.4.4 Sterowanie ręczne chwytakiem

Funkcję ręcznego sterowania otwarciem/zamknięciem chwytaka podłączono do przycisku 1 w grupie (12) na panelu sterowania ręcznego (MPC) (rys. 6). Jeżeli chwytak jest otwarty, to naciśnięcie przycisku spowoduje zamknięcie szczęk chwytaka. Gdy chwytak jest zamknięty, to naciśnięcie przycisku 1 spowoduje otwarcie szczęk chwytaka.

UWAGA: Podczas zamykania chwytaka zachować szczególną ostrożność, gdy obiekt do uchwycenia podajemy z ręki. Gdy chwytak jest zamknięty na uchwyconym obiekcie (klocek, pisak), nie otwierać go przypadkowo.

- 1.1 Przeprowadzić załączenie systemu.
- 1.2 Przełączyć tryb sterowania manipulatorem na ręczny.
- 1.3 Załączyć napędy robota.
- 1.4 Dla wszystkich przestrzeni ruchu JOINT, FRAME, TOOL zrealizować ruchy poszczególnych *stopni swobody* ramienia i zaobserwować różnice w ich wykonaniu. Uzasadnić różny sposób poruszania się robota w różnych przestrzeniach.
- 1.5 Otworzyć i zamknąć chwytak manipulatora.
- 1.6 Ustawić ramię w położeniu wyjściowym.

2 Aplikacje i menadżer aplikacji w systemie manipulatora Stäubli

W systemie manipulatora Stäubli przyjęto koncepcję, w której realizacja zadania w sposób automatyczny wymaga przygotowania tzw. aplikacji. Aplikacja zawiera między innymi programy w języku VAL3 oraz zmienne związane z lokalizacjami i układami. Po uruchomieniu sterownika robota (stan domyślny na stanowisku laboratoryjnym) w jego pamięci operacyjnej nie ma załadowanej żadnej aplikacji i jest możliwa jedynie ręczna manipulacja ramieniem. Aplikacje są przechowywane w pamięci dyskowej sterownika, z której można je załadować do pamięci operacyjnej (jednocześnie w pamięci może znajdować się więcej niż jedna aplikacja).

Przygotowanie i zarządzanie aplikacjami z poziomu panelu sterowania ręcznego MCP umożliwia menadżer aplikacji, dostępny za pomocą przycisku **Menu**, po użyciu którego pojawia się okno główne menu (z podświetloną opcją **Application Manager**). Do poruszania się po menu menadżera służą przyciski interfejsu i nawigacji (9) oraz przyciski menu tekstowych (7), przedstawione na rys. 6. Poruszanie się po menu odbywa się następująco:

- wybór podświetlenia opcji menu realizowany jest przez przyciski strzałek \uparrow lub \downarrow ,
- wejście w pozycję menu umożliwia przycisk \leftrightarrow , **Return**,
- rozwinięcie lub zwinięcie pola w strukturze menu poprzedzonego odpowiednio znakiem $+$ lub $-$ umożliwiają przyciski strzałek \rightarrow – rozwinięcie oraz \leftarrow – zwinięcie.

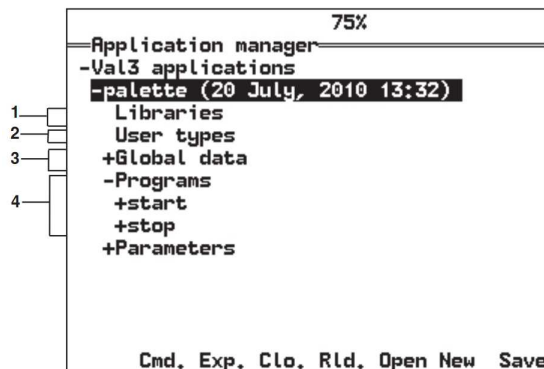
Utworzenie nowej aplikacji użytkownika w menadżerze umożliwia wybór **New** z menu przycisków tekstowych. Domyślna, nowa aplikacja składa się z:

- (1) *libraries* – bibliotek, w których można umieszczać zbiory funkcjonalności (podprogramy, funkcje) definiowane przez użytkownika,
- (2) *user types* – listy typów definiowanych przez użytkownika,
- (3) *global variables* – zmiennych globalnych, w których można wyróżnić między innymi zmienne związane z układami i lokalizacjami typu **Frame**, **Point** (grupa zmiennych **World**) oraz narzędziami typu **tool** (grupa zmiennych **flange**),
- (4) *programs* – dwóch programów domyślnych **start()** oraz **stop()**, które zawsze uruchamiane są odpowiednio przy uruchamianiu i kończeniu pracy aplikacji (programy te nie mogą zawierać parametrów wejściowych, nie można ich usunąć ani zmienić nazwy),
- (5) *parameters* – zestawu domyślnie ustawionych parametrów aplikacji.

Przykładowy wygląd okna menadżera z otwartą aplikacją pokazano na rysunku 9.

W otwartej aplikacji można definiować układy współrzędnych związanych z narzędziem roboczym, układem związanym z realizowanym zadaniem, uczyć lub definiować lokalizacje w przestrzeni kartezjańskiej lub konfiguracyjnej oraz przygotowywać programy sterujące dla robota w języku VAL3. W przypadku układu narzędzia możliwe jest wyłącznie jego zdefiniowanie przez wprowadzenie danych numerycznych. Układy związane z realizowanym zadaniem (bazowe) mogą być definiowane przez dane numeryczne lub nauczane metodą trzypunktową.

Zmienne związane z układami współrzędnych i lokalizacjami, w tym zmienne podstawowych typów danych, w języku VAL3 przechowywane są w tablicach lub kolekcjach. Domyślnie przy definiowaniu zmiennej tworzona jest tablica jednowymiarowa jednoelementowa przy czym wielkość tablicy można dynamicznie zmieniać. Maksymalnie można utworzyć tablice trójwymiarowe.



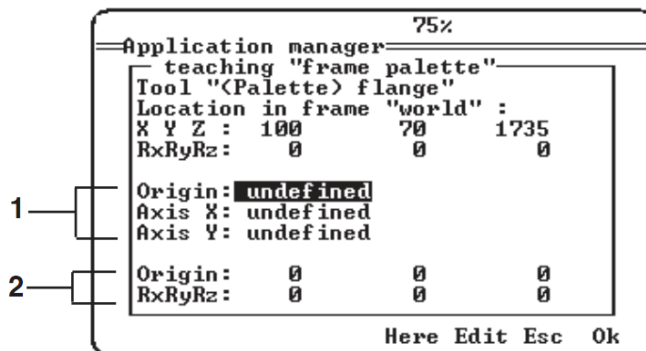
Rysunek 9: Widok ekranu menadżera aplikacji z przykładową otwartą aplikacją i widokiem menu przycisków tekstowych.

2.1 Definiowanie narzędzia

Definicję układu związanego z punktem charakterystycznym narzędzia TCP (ang. *Tool Center Point*) wprowadza się w aplikacji w grupie zmiennych globalnych (`Global data->flange`), wprowadzając nową zmienną typu `tool` (przycisk `New`). Po zadeklarowaniu zmiennej można podać dane numeryczne nowego układu narzędzia oraz skonfigurować wyjście cyfrowe, odpowiedzialne za działanie narzędzia. Opcjonalnie można wprowadzić czasy opóźnienia, związanego z działaniem narzędzia. Dane te wprowadza się poprzez naciśnięcie przycisku `↔`, `Return` (odpowiada `Enter`) na wybranej nazwie zmiennej typu `tool`.

2.2 Definiowanie własnego układu odniesienia

Kartezjański układ odniesienia (użytkownika), z którym związane ma być zadanie dla robota, wprowadza się w grupie zmiennych globalnych `world`, dodając nową zmienną typu `frame`. Układ ten można zdefiniować, podając wartości numeryczne lub nauczyć, po wybraniu przycisku `Teac`, wskazując trzy punkty określające początek układu `Origin` i punkty na osiach: `X – Axis X` oraz `Y – Axis Y` (patrz rys. 10). Uczenie realizuje się przez dojazd TCP wybranego narzędzia do wyżej wymienionych punktów (fragment okna (1) na rys. 10) i zatwierdzanie przyciskiem `Here`. Końcowy wynik nauczonego układu jest widoczny w fragmencie okna (2) rysunku 10, w którym `Origin` reprezentuje pozycję `X Y Z` aktualnego układu współrzędnych względem układu `world`, natomiast `RxRyRz` reprezentują orientację aktualnego układu współrzędnych względem układu `world`.



Rysunek 10: Widok ekranu z oknem do uczenia nowego układu odniesienia metodą trzypunktową.

2.3 Uczenie punktów

Punkty charakteryzujące ruch manipulatora mogą być reprezentowane w przestrzeni kartezjańskiej, wówczas są typu `Point` lub w przestrzeni złącz, wtedy są typu `Joint`. W przypadku definiowania lub uczenia punktów lokalizacji, należy utworzyć nową zmienną, podając jej nazwę oraz typ. Zmienna ta, w zależności od typu `Point/Joint`, widoczna jest w odpowiedniej grupie pod wybraną nazwą własną, a jej opis jest poprzedzony sygnaturą 0, co oznacza, że nowo utworzona zmienna ma podstawione/wprowadzone wartości równe zero. W strukturze aplikacji zmienne typu `Point` są widoczne w grupie `world`, zmienne typu `Joint` w grupie `joint`. Aby zmienna opisująca punkt położenia manipulatora zawierała wartości danej pozycji, należy je wprowadzić ręcznie, lub wskazać aktualną pozycję manipulatora za pomocą opcji menu kontekstowego `F1, Here`.

Przed wskazaniem opcji `Here` można przemieścić ramię manipulatora, w wybranym układzie odniesienia, do dowolnej pozycji. Po wyborze tej opcji, przy nazwie zmiennej pojawi się sygnatura `@`, co oznacza, że ramię manipulatora znajduje się w tej pozycji.

W zmiennej typu `Point`, poza polami opisującymi pozycję i orientację, występują dodatkowo pola odpowiedzialne za wybranie określonej konfiguracji ramienia, która ma być uwzględniona przy dojeździe do tego punktu. Konfiguracja ta zawiera parametry ustawień osi barku, łokcia i nadgarstka, które mogą przyjmować różne warianty (patrz 3.1, typ strukturalny `config`). Typowo pozostawia się konfigurację ustawioną w trakcie uczenia jako domyślną, co zapewnia, że sterownik nie wybierze innej podczas realizacji obliczeń dla tego punktu.

Podobnie można zdefiniować lub nauczyć lokalizacji w przestrzeni złącz, poprzez utworzenie nowej zmiennej typu `Joint` w grupie `joint`. Dla lokalizacji tego typu nie ma żadnych dodatkowych informacji, oprócz sześciu kątów poszczególnych osi manipulatora.

2.4 Dojazd do punktu

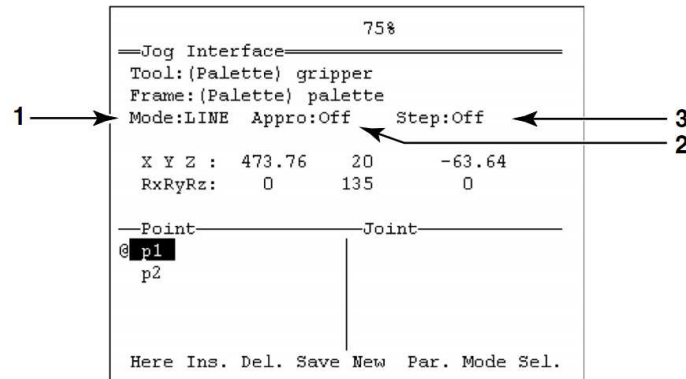
Mając wprowadzone do aplikacji punkty lokalizacji, można skorzystać z czwartego trybu ruchu ręcznego, umożliwiającego dojazd do punktu z możliwością wyboru jednej z trzech dostępnych strategii dojazdu, które zostały omówione w punkcie 1.4.3. Po wybraniu trybu `POINT`, przemieszczania do punktu widoczne będą w oknie pokazane na rys. 11. W oknie tym, w dolnej części widoczne, są dostępne lokalizacje typu `Point` (po lewej) lub `Joint` (po prawej). W górnej części widoczny jest bieżący układ narzędzia oraz układ bazowy, związany z punktami w przestrzeni kartezjańskiej (zmiana poprzez przycisk `Sel.`). Zaznaczone na rysunku elementy oznaczają odpowiednio:

- (1) wybrany sposób dojazdu do punktu (opis w 1.4.3, przycisk `Mode`),
- (2) sposób podejścia do punktu z opcją zatrzymania w zadanej odległości przed punktem wzdłuż osi `Z` narzędzia (przycisk `Par.`),
- (3) wybór trybu dojazdu w sposób ciągły lub z zadany krokiem (przycisk `Par.`).

W centralnej części znajduje się informacja o wybranej lokalizacji w przestrzeni kartezjańskiej bądź konfiguracyjnej w zależności od typu punktu. Z poziomu tego okna można również modyfikować współrzędne istniejących punktów, wstawiać nowe pola do zmiennych tablicowych lokalizacji oraz dodawać nowe zmienne lokalizacji w bieżącej aplikacji.

Uruchamianie przygotowanej aplikacji w trybie sterowania ręcznego, jak i lokalnego, jest bardzo podobne. Zasadniczą różnicą jest konieczność trzymania przycisku zezwolenia `Move/Hold` w trybie ręcznym. Procedura uruchomienia przebiega w następujący sposób:

- wybranie trybu pracy z interfejsu operatora `WMS`,
- uruchomienie napędów przyciskiem włączania zasilania ramienia (dla trybu ręcznego najpierw wcisnąć przycisk zezwolenia na ruch (`#`)) na panelu ręcznego sterowania `MPC`,
- wciśnięcie przycisku uruchomienia aplikacji `Run` (może być konieczny wybór aktywnej aplikacji z listy otwartych),
- realizacja programu aplikacji rozpoczyna się po wciśnięciu przycisku `Move/Hold`.



Rysunek 11: Widok ekranu z oknem w trybie POINT (sterowania do punktu).

3 Elementy języka programowania VAL3

Programowanie zadań dla robota jest realizowane w języku VAL3 [3]. Język ten jest wykorzystywany zarówno w przypadku tworzenia programu za edytor w menadżerze aplikacji w panelu sterowania ręcznego MPC, jak i za pomocą dedykowanego oprogramowania narzędziowego *Stäubli Robotics Suite 2013*.

3.1 Typy, zmienne w VAL3

W języku VAL3 dostępne są następujące typy proste:

- **bool** – typ logiczny (**true/false**),
- **num** – typ wartości numerycznych (całkowitych lub zmiennoprzecinkowych),
- **string** – typ dla łańcuchów znaków (znaki ASCII/Unicode),
- **dio** – dla wejść/ wyjść cyfrowych,
- **aio** – dla wejść/wyjść analogowych,
- **sio** – dla portów wejść/ wyjść, łączy szeregowych lub internetowych,
- **screen** – dla wyświetleń ekranowych MCP i dostępu do klawiatury.

W systemie programowania robota VAL3 zdefiniowano następujące typy strukturalne:

- **trsf** – typ dla transformacji geometrycznych w układzie kartezjańskim XYZ, zawiera następujące pola: **x**, **y**, **z**, **rx**, **ry**, **rz**;
Typ ten definiuje zmianę położenia i/lub orientacji. Jest kompozycją translacji i rotacji. Transformacja sama w sobie nie reprezentuje pozycji w przestrzeni, ale jest interpretowana jako położenie i orientacja punktu lub ramki w układzie kartezjańskim względem innej ramki, np. ramki związanej z układem podstawowym;
- **frame** – typ dla reprezentacji pozycji (położenia i orientacji) w kartezjańskim układzie odniesienia (wykorzystuje pola struktury typu **trsf**), zawiera następujące pola: **x**, **y**, **z**, **rx**, **ry**, **rz**;

- **tool** – typ opisujący narzędzie, zawiera następujące pola: **x**, **y**, **z**, **rx**, **ry**, **rz**, **IOLink**, **otime**, **ctime**;

Typ ten określa geometrię narzędzia oraz sterowanie nim. Pola **x**, **y**, **z** służą do opisu pozycji TCP w układzie współrzędnych robota, **rx**, **ry**, **rz** służą do opisu orientacji narzędzia. Pole **IOLink** – wyjście używane do aktywacji narzędzia, **otime** – czas potrzebny do otwarcia narzędzia (w sec.), **ctime** – czas potrzebny do zamknięcia narzędzia. Narzędzie referencyjne typu **tool** o nazwie **flange** jest zawsze zdefiniowane w aplikacji VAL3. Jest to zmienna, która reprezentuje opis kolnierza robota (koniec łańcucha kinematycznego) służący do podłączenia każdego zdefiniowanego później narzędzia. Każde narzędzie wykorzystywane w programowaniu robota jest dołączone pośrednio lub bezpośrednio do kolnierza.

- **config** – typ dla konfiguracji manipulatora, która określa rodzaje ułożenia ogniwo manipulatora tworzących bark (ogniwo: kolumna – ramię), łokieć (ogniwo: ramię – przedramię) i nadgarstek (ogniwo: przedramię – kiść), zawiera następujące pola: **shoulder**, **elbow**, **wrist**. Każde pole może przyjąć wartość jednego z 4 parametrów, umożliwiając określenie ułożenia ogniwo w wyróżnionych przegubach:

- **shoulder** (bark): **righty** – z prawej, **lefty** – z lewej, **ssame** – jak poprzednio, **free** – dowolne,
- **elbow** (łokieć): **epositive** – do góry, **enegative** – do dołu, **esame** – jak poprzednio, **efree** – dowolnie,
- **wrist** (nadgarstek): **wpositive** – do góry, **wnegative** – do dołu, **wsame** – jak poprzednio, **wfree** – dowolnie.

- **point** – typ reprezentujący pozycję narzędzia czyli jego położenie i orientację w układzie kartezjańskim względem układu odniesienia (pola struktury typu **trsf**) oraz konfigurację ramienia robota w tym punkcie (pola struktury typu **config**), zawiera następujące pola: **x**, **y**, **z**, **rx**, **ry**, **rz**, **shoulder**, **elbow**, **wrist**.
- **joint** – typ dla pozycji robota w układzie współrzędnych wewnętrznych, zawiera następujące pola: **j1**, **j2**, **j3**, **j4**, **j5**, **j6**.
- **mdesc** – typ dla parametrów ruchu robota, zawiera następujące pola: **accel**, **vel**, **decel**, **tvel**, **rvel**, **blend**, **leave**, **reach**. Typ ten przechowuje kolejno wartości: przyspieszenia robota w mm^2/s – **accel**, prędkości robota w mm/s – **vel**, opóźnienia robota w mm^2/s – **decel**, maksymalnej prędkości translacji TCP w mm/s – **tvel**, maksymalnej prędkości kątowej obrotu narzędzia wokół TCP w $\text{st.}/\text{s}$ – **rvel**, **blend** – pole wyboru rodzaju interpolacji w otoczeniu punktów pośrednich ścieżki (**off/joint/Cartesian**), **leave** – odległość w mm przed punktem pośrednim, **reach** – odległość w mm za punktem pośrednim.

3.2 Wybrane instrukcje kontroli programu

Pętle programowe:

- ```
while <bool bCondition>
 blok z kodem programu
endWhile
```
- ```
do
    blok z kodem programu
until <bool bCondition>
```

wykonują bloki z kodem programu tak długo, dopóki warunek **bCondition** jest prawdziwy. W pierwszym przypadku warunek jest sprawdzany przed wykonaniem bloku programu, w drugim po jego wykonaniu.

- `for <num nCounter> = <num nBeginning> to <num nEnd>`
 blok z kodem programu
`endFor`

Pętla pomiędzy instrukcjami `for` i `endFor` wykonuje się dopóki zmienna `nCounter` nie osiągnie wartości `nEnd`; `nBeginning` – wartość początkowa indeksu pętli.

3.3 Wybrane instrukcje przetwarzania danych i ruchu robota

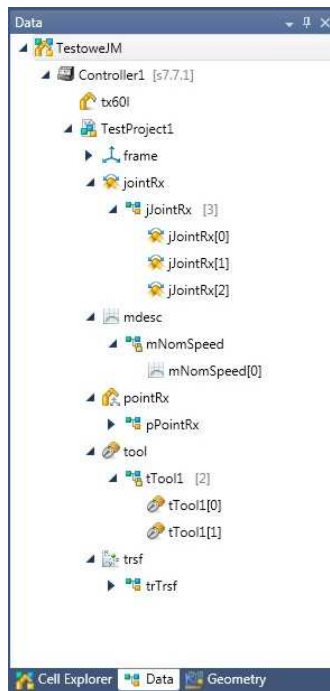
- `joint herej()`
 Instrukcja ta zwraca aktualną pozycję kątową wszystkich osi robota i może być przypisana do zmiennej typu `joint`, np.:
`j_point = (0,0,0,0,0,0)`
`j_point = herej()`
- `point appro(point pPosition, trsf trTransformation)`
 Instrukcja zwraca punkt zmodyfikowany przez transformację `trTransformation`, która jest definiowana względem tego samego układu odniesienia, co argument `pPosition`. Wynik można podstawić do zmiennej typu `point`.
- `point compose(point pPosition, frame fReference, trsf trTransformation)`
 Instrukcja zwraca punkt zmodyfikowany przez transformację `trTransformation`, która jest definiowana względem układu odniesienia `fReference`. Wynik można podstawić do zmiennej typu `point`.
- `num movej(point pPosition, tool tTool, mdesc mDesc)`
 Instrukcja ruchu robota do punktu `pPosition` z wykorzystaniem narzędzia `tTool` oraz z parametrami ruchu `mDesc`. Ruch odbywa się najkrótszą drogą dla współrzędnych osiowych manipulatora.
- `num movel(point pPosition, tool tTool, mdesc mDesc)`
 Instrukcja ruchu liniowego do punktu `pPosition` z wykorzystaniem narzędzia `tTool` oraz z parametrami ruchu zawartymi w zmiennej `mDesc`. Ruch odbywa się najkrótszą drogą dla współrzędnych kartezjańskich.
- `open(tool tTool)`
 Instrukcja otwarcia narzędzia `tTool`.
- `close(tool tTool)`
 Instrukcja zamknięcia narzędzia `tTool`.

4 Dedykowane oprogramowanie narzędziowe dla manipulatora Stäubli

Oprogramowanie narzędziowe Stäubli Robotics Suite 2013 (SRS) umożliwia tworzenie, edytowanie, uruchamianie i testowanie programu, transfer programu pomiędzy sterownikiem robota CS8C i środowiskiem SRS, symulacje pracy modelu robota w środowisku graficznym. Wprowadzenie nowego projektu realizuje się poprzez nadanie mu nazwy własnej i zapis na dysku komputera oraz wybranie typu manipulatora i sterownika. W oknie głównym oprogramowania SRS, po wybraniu opcji **Home** → **Show 3D View**, możemy pokazać graficzny manipulator. Po wybraniu opcji **Home** → **General** → **Cell Explorer** możemy przejść do zasadniczego okna pokazującego strukturę projektu w trzech zakładkach: **Cell Explorer**, **Data** oraz **Geometry** (patrz rys. 12).

Widok, przedstawiony na rys. 12, zawiera projekt w oprogramowaniu SRS o nazwie **TestoweJM**, który ma strukturę hierarchiczną. Do nazwy projektu przypisany jest sterownik o nazwie **Controller1** [s.7.7.1], do sterownika z kolei dołączony jest model robota. W tym przypadku jest to model manipulatora **tx601**. Ostatnim elementem struktury projektu jest aplikacja użytkownika o nazwie własnej **TestProject1**, przeznaczona do sterowania robotem². Aplikacja użytkownika w zakładce **Cell Explorer** jest po rozwinięciu prezentowana także w strukturze hierarchicznej.

Zakładki **Cell Explorer/Data/Geometry** umożliwiają obsługę programów w aplikacji użytkownika, zakładka **Data** umożliwia obsługę zmiennych programu i podprogramów, zakładka **Geometry** umożliwia obsługę zmiennych ulokowanych w różnych układach geometrycznych.



Rysunek 12: Widok pola Data w oknie głównym projektu.

4.1 Aplikacje

Zadania związane z programowaniem robota mogą być realizowane w aplikacji przygotowanej przez użytkownika, niezależnie od ręcznego panelu sterującego MPC. Tworzenie nowej aplikacji polega na wybraniu kontrolera w oknie **Cell Explorer** i zaznaczeniu go. Korzystając z tego kontekstu możliwe jest wybranie opcji **VAL3 → New Application**, a po pojawieniu się właściwego okna, należy wpisać nazwę aplikacji, wskazać lokalizację dyskową oraz wybrać szablon (dostępny default). Istniejącą aplikację możemy odczytać z dysku, wybierając opcję **VAL3 → Open Application**, także w kontekście wybranego kontrolera.

4.2 Programy i podprogramy

Tworzenie programu realizowane jest przez wybranie aplikacji (zaznaczenie w **Cell Explorer**) i wybranie opcji **VAL3 → New Program**. Wybranie programu i jednokrotne kliknięcie umożliwia rozwinięcie listy jego parametrów oraz zmiennych lokalnych, dwukrotne kliknięcie umożliwia

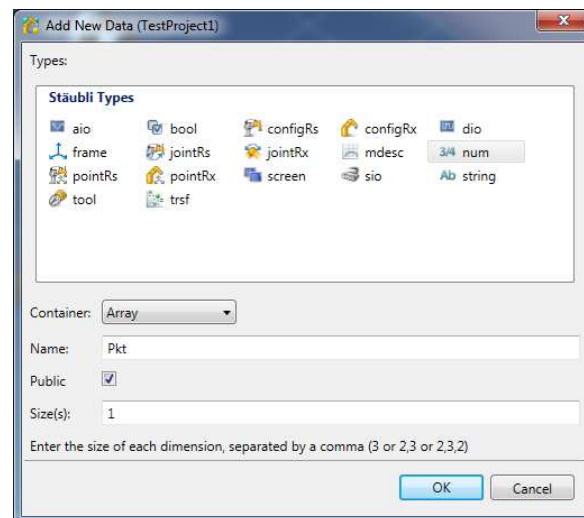
²Aplikację użytkownika w całości można przetransferować na dysk sterownika robota CS8C i uruchomić za pomocą menadżera w MPC.

wyświetlenie okna edycji programu. Programy i podprogramy w VAL3 zapisywane są pomiędzy słowami kluczowymi **begin** oraz **end**. Można utworzyć wiele programów dostępnych z jednej aplikacji. Ich spis dla danej aplikacji znajduje się w oknie **Cell Explorer**. W każdej aplikacji znajdują się domyślne programy **start()** i **stop()**. Wykonanie aplikacji związane jest zawsze z wykonaniem programów **start()** i **stop()**. Dla przykładu, aby uruchomić program użytkownika w aplikacji, należy wywołać go komendą **call** wewnątrz programu **start()**.

4.3 Zmienne

Zmienne w aplikacji robota mogą być deklarowane jako globalne – **Public**, jako lokalne – **Locals** lub jako parametry programów – **Parameters**.

W celu lokowania nowej zmiennej, należy wybrać aplikację (zaznaczając jej nazwę), a następnie opcję **VAL3** → **New Data**, skutkiem czego wyświetlone zostanie okno **Add New Data** (dla zasięgu aplikacji) (patrz rys. 13). Okno zawiera typy proste i złożone, umożliwia edycję nazwy zmiennej i wybór typu, pojemnika (**Array/Collection**), zasięgu stosowania (**Local/Public**) oraz rozmiaru.



Rysunek 13: Widok okna dodawania zmiennych w ramach typów Stäubli w oprogramowaniu SRS.

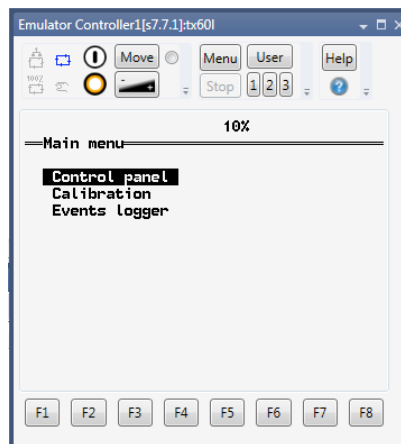
4.4 Transfer aplikacji pomiędzy sterownikiem robota i oprogramowaniem SRS

W celu przetransferowania aplikacji pomiędzy sterownikiem robota CS8C a komputerem PC za pomocą funkcji transferu w środowisku SRS, wybieramy w tym środowisku opcję **Home** → **Transfer Manager**. W oknie menadżera transferu po prawej stronie okna pojawiają się zasoby sterownika robota CS8C (dane na jego dysku), po lewej stronie zasoby oprogramowania SRS (wskazany katalog na dysku komputera). Połączenie umożliwia przepisanie danych z dysku sterownika robota CS8C na dysk komputera z oprogramowaniem SRS. Wyboru danych do transferu dokonuje się przez wskazanie i zaznaczenie danych do przeniesienia. Po dokonaniu transferu, celem uruchomienia i testowania, należy wczytać aplikację VAL3 do środowiska SRS. W przypadku odwrotnym, tj. skopiowania aplikacji VAL3 ze środowiska SRS do sterownika robota CS8C, w celu uruchomienia aplikacji w sterowniku robota CS8C, należy w ręcznym panelu sterującym MPC za pomocą menadżera aplikacji wybrać ją z dysku, a następnie otworzyć.

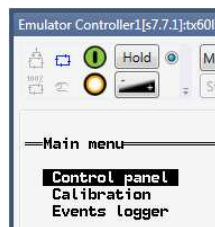
4.5 Sprawdzenie i uruchomienie zadania w symulatorze

Warunkiem uruchomienia programu robota napisanego przez użytkownika jest jego bezbłędna kompilacja. W celu sprawdzenia poprawności działania programu wybieramy w zakładce **Cell Explorer** właściwą aplikację (zaznaczając jej nazwę) oraz z menu głównego opcję **VAL3** → **Check Syntax**. Kompilacji poddane są wszystkie programy i podprogramy danej aplikacji. Wyniki kompilacji możemy zaobserwować w oknie **Output**, wybierając w nim pod-opcję **Syntax Checker**.

Aby przetestować zadanie na symulatorze robota, należy napisać program w języku VAL3 uruchomić na emulatorze kontrolera, oraz dodatkowo można uruchomić symulację ruchu robota w środowisku graficznym. W tym celu należy uruchomić **Emulator Controller1[s.7.7.1]**, co następuje w wyniku wybrania w **Cell Explorer** **Controller1[s.7.7.1]** i z menu kontekstowego prawym przyciskiem myszy wybrania opcji **Show Emulator** (skrót Ctrl R, S). Widok uruchomionego emulatora kontrolera przedstawiono na rys. 14.



Rysunek 14: Widok emulatora kontrolera w SRS.



Rysunek 15: Widok emulatora kontrolera w SRS z włączonym przyciskiem napędów oraz włączonym przyciskiem Move/Hold.

Uruchomienie programu w emulatorze wymaga ustawienia emulatora w tryb lokalny lub zdalny (rys. 5) oraz ustawienia jego przycisków jak na rys. 15.

Obserwacja ruchu robota jest możliwa po wyświetleniu widoku jego modelu, następuje to po wybraniu opcji **Home** → **Show 3D View**. Synchronizacji modelu robota z emulatorem w środowisku graficznym dokonuje się poprzez wybranie opcji **Simulation** → **Start Synchro**.

5 Zadanie paletyzacji

Zadanie paletyzacji polega na przenoszeniu elementów manipulowanych z pozycji źródłowych do pozycji docelowych z wykorzystaniem pętli programowych. Pozycje źródłowe i docelowe są statyczne,

wyróżnia je możliwość ustalenia odległości względem wybranej pozycji początkowej³ dla każdej palety. Pozycje źródłowe jednej palety oraz pozycje docelowe drugiej palety są zwykle skonfigurowane w jednym układzie współrzędnych. Układane przedmioty palety mają leżeć na jednej płaszczyźnie, w ustalonej względem siebie odległości i orientacji. Posługując się klockami testowymi, należy pobrać kolejno 4 klocki z pozycji źródłowej (np. ułożone w linii prostej) i przenieść je do pozycji docelowej (np. ułożenie 2×2), przy wykorzystaniu zmiennych, instrukcji ruchu i pętli programowych. Zadanie paletyzacji podzielić można na 3 etapy:

- Pozycjonowanie manipulatora i podjęcie klocków z miejsc (pozycji) źródłowych – z palety P1,
- Przeniesienie przy pomocy zaprogramowanych ruchów manipulatora,
- Pozycjonowanie manipulatora i odłożenie klocków na pozycje docelowe – na paletę P2.

Przykładowy program, napisany do wybranego zadania paletyzacji, znajduje się na rys. 16.

W programie tym wykorzystano układ współrzędnych użytkownika o nazwie `fLocalP` typu `frame`, w którym określono pozycje jako punkty początkowe obu palet. Robot operuje zdefiniowanym narzędziem o nazwie `chwytak1` typu `tool` z dostosowanymi do wymagań zadania, ustalonymi parametrami ruchu `nomPaleta` typu `mdesc` (przeniesienie klocków) oraz z prędkościami zmniejszonymi w parametrach ruchu w zmiennej `slowPaleta` typu `mdesc` podczas realizacji ruchów podniesienia i opuszczenia klocka. Podczas zamykania chwytaka oraz otwierania chwytaka na klocku ramię manipulatora powinno być w bezruchu.

Punkty początkowe palet zostały zdefiniowane na podstawie pozycji TCP narzędzia w zmiennych typu `PoinRx` dla palety P1: `P1_Rx1`, dla palety P2: `P2_Rx1`. Obie zmienne są zasobnikami rodzaju `array`. W ich polach o indeksie 0 przechowywana jest ustalona nominalna pozycja początkowa nad paletą, w polach o indeksie 1 przechowywane są bieżące wartości pozycji, wyliczane na podstawie indeksowania elementów palet oraz wysokości nad paletami.

³Pozycja początkowa to pozycja, od której rozpoczyna się realizację podnoszenia/odkładania przedmiotów z/na palety/tę.

6 Przebieg zadania

- 6.1 Ułożyć klocki na stanowisku źródłowym w linii i wybrać nad pierwszym klockiem punkt startowy operacji podniesienia klocka.
- 6.2 Przełączyć tryb sterowania manipulatorem na ręczny.
- 6.3 Za pomocą menadżera aplikacji w panelu MPC otworzyć aplikację użytkownika VAL3, nadać jej nazwę i zapisać ją na dysku sterownika robota.
- 6.4 Zdefiniować narzędzie robocze – chwytak. W tym celu utworzyć zmienną typu `tool` o wybranej nazwie, która ma reprezentować narzędzie robocze dołączane do kołnierza (`flange`). Zapoznać się z geometrią chwytaka i w parametrach utworzonej zmiennej zapisać odpowiednie wartości odległości oraz kąty orientacji wokół osi. Dla zamontowanego chwytaka parametry te są następujące: $x = 70\text{mm}$, $y = 0\text{mm}$, $z = 140\text{mm}$, $rx = 0^\circ$, $ry = 45^\circ$, $rz = 0^\circ$.
- 6.5 Po zdefiniowaniu w aplikacji użytkownika narzędzia, wybrać je do pracy, po uruchomieniu sterowania ręcznego w trybie Tool. Za pomocą ręcznie wymuszanych ruchów w układzie narzędzia, sprawdzić poprawność jego konfiguracji. Rotacje powinny odbywać się wokół TCP skonfigurowanego i wybranego narzędzia, translacje wzdłuż osi narzędzia.
- 6.6 Uchwycić pisak za pomocą chwytaka, zachowując ostrożność przy ręcznym podawaniu pisaka.
- 6.7 Zdefiniować narzędzie wskazujące – chwytak z pisakiem. W tym celu utworzyć nową zmienną typu `tool` o wybranej nazwie, której parametry będą zdefiniowane względem układu opisującego chwytak, i w jej parametrach uzupełnić długość pisaka: $z = 150\text{mm}$, pozostałe parametry równe zero.
- 6.8 Zdefiniować własny układ bazowy dla zadania paletyzacji. W tym celu utworzyć zmienną typu `frame` i przypisać jej dane, postępując zgodnie z procedurą definiowania układu bazowego. Punkty charakterystyczne podczas definicji układu bazowego wskazywać pisakiem.
- 6.9 Wyjąć pisak z chwytaka, zachowując ostrożność przy ręcznym odebraniu pisaka.
- 6.10 Najechać manipulatorem z otwartym chwytakiem na pozycję znajdującą się bezpośrednio nad klockiem tak, aby podjęcie klocka było precyzyjne. Podczas osiągnięcia tej pozycji zmniejszyć prędkość ruchu (np. do 1 – 0.2%).
Uwaga! Niedozwolone jest dotknięcie chwytakiem podstawy, na której leży podnoszony klocek.
Zamknąć chwytak, upewniając się czy uchwycenie klocka w chwytaku jest prawidłowe.
- 6.11 Przenieść uchwycony klocek ponad paletę, na wysokość odpowiadającą wysokości przeniesienia, np. 100mm nad paletę. Wysokość ta ma na celu bezkolizyjne przeniesienie klocka. Zapamiętać tę pozycję w zmiennej typu `pointRx`, np. `P1_Rx1[0]` – będzie to punkt opisujący pozycję początkową na palecie P1.
- 6.12 Przenieść ramię robota z uchwyconym klockiem do pozycji nad miejscem odkładania na palecie P2. Wybrać punkt położony na wysokości ok. 0.5-1mm ponad powierzchnią P2 i zapamiętać w zmiennej typu `PointRx`, np. `P2_Rx1[0]` – będzie to punkt opisujący pozycję początkową na palecie P2.
- 6.13 Otworzyć chwytak, pozostawiając klocek na pozycji pierwszego elementu palety P2 i odsunąć chwytak robota ponad klocek.
- 6.14 Zapisać aplikację użytkownika VAL3 na dysku sterownika robota za pomocą menadżera aplikacji w MPC oraz zamknąć aplikację. Aplikacja użytkownika oraz zdefiniowane w niej układy odniesienia, narzędzie i punkty pozycjonowania są elementami referencyjnymi do realizacji dalszej części zadania.

- 6.15 Załączyć zasilanie komputera PC na stanowisku i w systemie operacyjnym uruchomić oprogramowanie narzędziowe Stäubli Robotics Suite 2013 (SRS) (dostępne na pulpicie użytkownika Student).
- 6.16 Przygotowaną aplikację w MPC przetransferować z dysku sterownika robota CS8C na dysk komputera PC za pomocą funkcji transferu środowiska SRS, celem napisania programu realizującego zadanie paletyzacji w cyklu automatycznym.
- 6.17 W edytorze środowiska SRS, w języku VAL 3, napisać program wykorzystujący instrukcje ruchu pomiędzy punktami referencyjnymi pozycji palet, przygotowanych za pomocą ramienia manipulatora. Instrukcje ruchu umieścić wewnątrz pętli indeksujących zmiany pozycji punktów referencyjnych, stosownie do numeru pobieranego klocka oraz numeru miejsca na palecie.
- 6.18 W symulatorze przetestować działanie aplikacji z programem paletyzacji. Przetestowany i działający poprawnie program, zawarty w aplikacji użytkownika, przetransferować teraz z dysku komputera PC na dysk sterownika robota CS8C za pomocą funkcji transferu w oprogramowaniu SRS.
- 6.19 Pod nadzorem prowadzącego, za pomocą menadżera aplikacji z panelu MPC, uruchomić przygotowaną aplikację i krokowo przetestować przebieg paletyzacji w rzeczywistym środowisku.
- 6.20 Po zakończeniu pracy z robotem należy ustawić manipulator w pozycji wyjściowej, tak jak na rys. 1b, zamknąć aplikację użytkownika w menadżeże aplikacji w MPC, a następnie wyłączyć zasilanie robota.

Literatura

- [1] Stäubli Arm – TX series 60 family. Instruction manual, D28082804B, TX60©Stäubli 2013.
- [2] Stäubli CS8 C Controller. Instruction manual, D28081204C, CS8 M©Stäubli 2014.
- [3] Stäubli VAL 3 reference manual, ver. 7, D28077104A, VAL3©Stäubli 2010.

```

begin
P1_xPrzes = 50 // przesunięcie pozycji klocka w osi X dla palety P1
P1_yPrzes = 0 // przesunięcie w osi Y dla palety P1
dyst_zP1 = 150 // odsunięcie w osi Z dla palety P1
P2_xPrzes = 50 // przesunięcie pozycji klocka w osi X dla palety P2
P2_yPrzes = 50 // przesunięcie pozycji klocka w osi Y dla palety P2
dyst_zP2 = 150 // odsunięcie w osi Z dla palety P2

// ruch do pozycji początkowej palety P1 z dowolnej pozycji (w złączach)
movej(P1_Rx1[0],chwyta1,nomPaleta)
// czekaj z realizacją następczej instrukcji do zakończenia ruchu
waitEndMove()

open(chwyta1)

for ip = 1 to 2
  for jp = 1 to 2

    // wyliczenie pozycji kolejnego klocka na palecie P1
    P1_Rx1[1] = P1_Rx1[0]
    act_px = (ip+jp-2)*P1_xPrzes
    act_py = 0
    // zmodyfikowanie pozycji na palecie P1
    P1_Rx1[1] = compose(P1_Rx1[1],fLocalP,{act_px,act_py,0,0,0})
    move1(P1_Rx1[1],chwyta1,nomPaleta)

    // precyzyjne zbliżenie do pozycji palety P1
    P1_Rx1[1] = compose(P1_Rx1[1],fLocalP,{0,0,-dyst_zP1,0,0,0})
    move1(P1_Rx1[1],chwyta1,slowPaleta)
    waitEndMove()

    // uchwycenie klocka
    close(chwyta1)
    // oddalenie od pozycji palety P1
    P1_Rx1[1] = compose(P1_Rx1[1],fLocalP,{0,0,dyst_zP1,0,0,0})
    move1(P1_Rx1[1],chwyta1,nomPaleta)

    // ruch do palety P2
    movej(P2_Rx1[0],chwyta1,nomPaleta)

    // wyliczenie pozycji kolejnego klocka na palecie P2
    P2_Rx1[1] = P2_Rx1[0]
    act_px = (ip-1)*P2_xPrzes
    act_py = (jp-1)*P2_yPrzes

    // zmodyfikowanie pozycji na palecie P2
    P2_Rx1[1] = compose(P2_Rx1[1],fLocalP,{act_px,act_py,0,0,0})
    move1(P2_Rx1[1],chwyta1,nomPaleta)
    waitEndMove()

    // precyzyjne zbliżenie do pozycji na P2
    P2_Rx1[1] = compose(P2_Rx1[1],fLocalP,{0,0,-dyst_zP2,0,0,0})
    move1(P2_Rx1[1],chwyta1,slowPaleta)
    waitEndMove()

    // odłożenie klocka
    open(chwyta1)
    // oddalenie od pozycji palety P2
    P2_Rx1[1] = compose(P2_Rx1[1],fLocalP,{0,0,dyst_zP2,0,0,0})
    move1(P2_Rx1[1],chwyta1,nomPaleta)

    // ruch do palety P1
    movej(P1_Rx1[0],chwyta1,nomPaleta)

  endFor
endFor

end

```