easy robots

easyrobots.eu

ESControl

Instrukcja obsługi i programowania użytkownika końcowego

Wersja: 2.1.0

Easy Robots Sp. z o.o.
ul. Gdyńska 32
26-600 Radom
Polska

Robot może posiadać nieopisane w niniejszej dokumentacji funkcjonalności wykorzystywane do celów serwisowych i gwarancyjnych.

Niniejsza dokumentacja została sprawdzona pod kątem zgodności ze sprzętem i oprogramowaniem. Niemniej jednak nie można wykluczyć istnienia rozbieżności, dlatego też firma Easy Robots nie gwarantuje całkowitej zgodności niniejszej instrukcji z dostarczoną wraz ze sprzętem wersją oprogramowania.

Zgodność z wersją oprogramowania: 1.2.5

Wersja dokumentacji: 2.1.0

Data publikacji: 26 lipca 2022

Spis treści

1 Wprowadzenie

1.1 Grupa docelowa

Niniejsza dokumentacja przeznaczona jest dla osób chcących zapoznać się z obsługą i programowaniem robotów Easy Robots (seria ES). Dokument ten w szczególności skierowany jest do automatyków oraz programistów wdrażających robota do pracy na docelowym stanowisku oraz serwisujących go.

1.2 Dokumentacja

W skład dokumentacji robota przemysłowego wchodzi:

- Dokumentacja zbiorcza dotycząca układu mechanicznego, elektrycznego oraz serwisowania produktu.
- Instrukcja programowania i obsługi oprogramowania EScontrol (ten dokument).
- Instrukcje modułów oprogramowania.

Każda instrukcja stanowi oddzielny dokument.

1.3 Symbole

Symbol	Opis
	Wskazówka opatrzona tym znakiem jest krytyczna pod względem bezpieczeństwa użytkownika urządzenia oraz samego urządzenia.
	Wskazówka opatrzona tym znakiem dotyczy bezpieczeństwa użytkownika urządzenia oraz samego urządzenia.
	Wskazówka opatrzona tym znakiem jest uzupełninem informacji zawartych w tekście.

1.4 Słownik pojęć

Manipulator — Zwany również ramieniem robotycznym komponent robota zbudowany z szeregu napędów i łączników tworzący konkretną strukturę kinematyczną na wzór ludzkiej kończyny górnej. Umożliwia manipulowanie w przestrzeni np. przenoszenie przedmiotów itd.

Panel sterujący — Połączony z szafą sterowniczą element wyposażenia robota zawierający ekran dotykowy. Panel sterujący (panel uczenia) pozwala na sterownie ruchem robota, programowanie oraz kontrolę działania urządzenia.

Przycisk bezpieczeństwa — Czerwony przycisk grzybkowy znajdujący się na panelu sterującym oraz szafie sterowniczej. Jego wciśnięcie powoduje zatrzymanie pracy manipulatora poprzez załączenie hamulców napędów oraz odłączenie zasilania napędów. Nazywany również przyciskiem awaryjnego zatrzymania.

Robot — Urządzenie programowalne, składające się z manipulatora, szafy sterowniczej, panelu sterującego oraz niezbędnego okablowania.

Skrypt sterujący	Tekst programu tworzonego przez programistę/automatyka w celu zaplanowania działania robota w czasie pracy na stanowisku. Skrypt sterujący jest zapisany w pamięci robota oraz jest wykonywany podczas jego pracy.
Stanowisko	Zespół urządzeń zestawionych w komórce produkcyjnej, przeznaczonych do wykonywania określonego zadania w procesie produkcyjnym. Obejmuje robota, urządzenia współpracujące oraz infrastrukturę komórki produkcyjnej.
Szafa sterownicza	Element wyposażenia robota w postaci skrzyni, realizujący zadania sterowania pozostałymi elementami urządzenia, komunikacji, przetwarzania sygnałów zewnętrznych oraz zasilania.
TCP	Ang.: *Tool Central Point*, czyli punkt charakterystyczny narzędzia, który uważa się za zakończenie łańcucha kinematycznego.
joint / staw	Element ruchomy manipulatora, zawierający przekładnię, silnik oraz sterownik napędu.

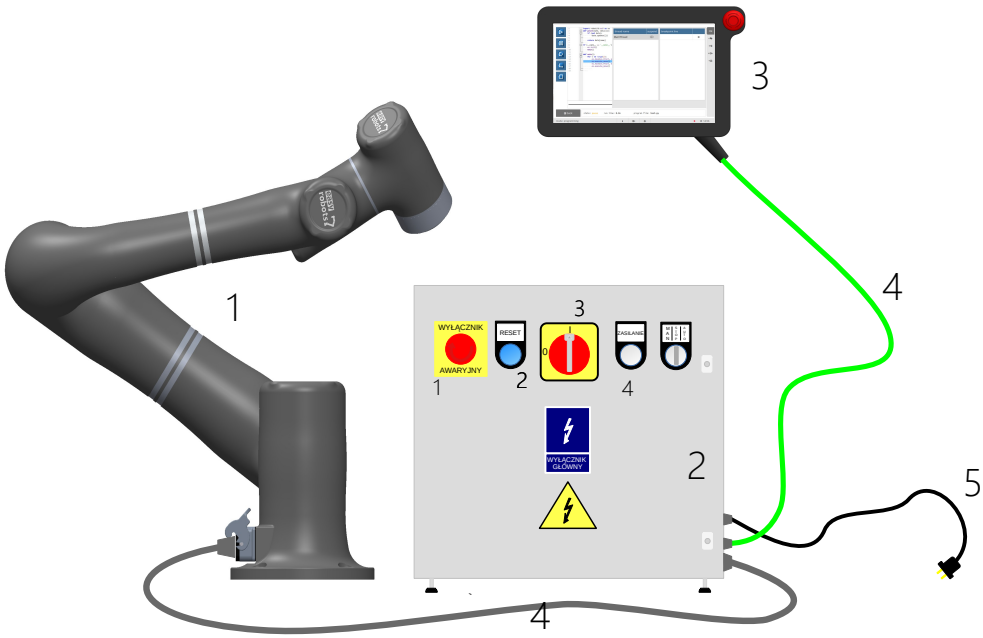
2 Opis produktu

2.1 Robot przemysłowy

Robot w wersji podstawowej składa się z następujących komponentów:

- robot 6-osiowy,
- szafa sterownicza,
- panel sterujący,
- przewody łączące,
- oprogramowanie,
- akcesoria,
- dokumentacja.

Elementy składowe

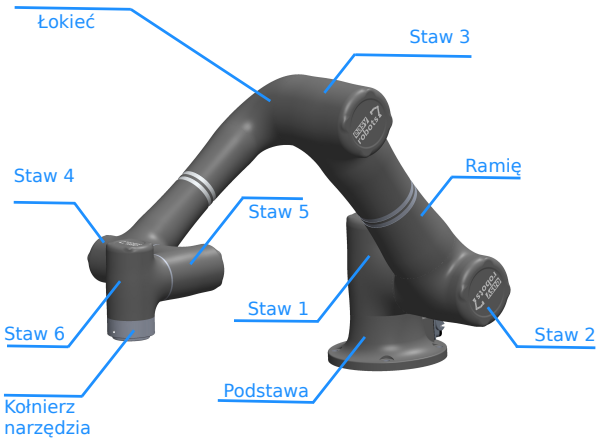


Rysunek 2.1: Robot ES5

Nr	Opis
1	Ramię robota
2	Szafa sterownicza
3	Panel sterujący
4	Przewody łączące
5	Przewód zasilający

Ramię robota

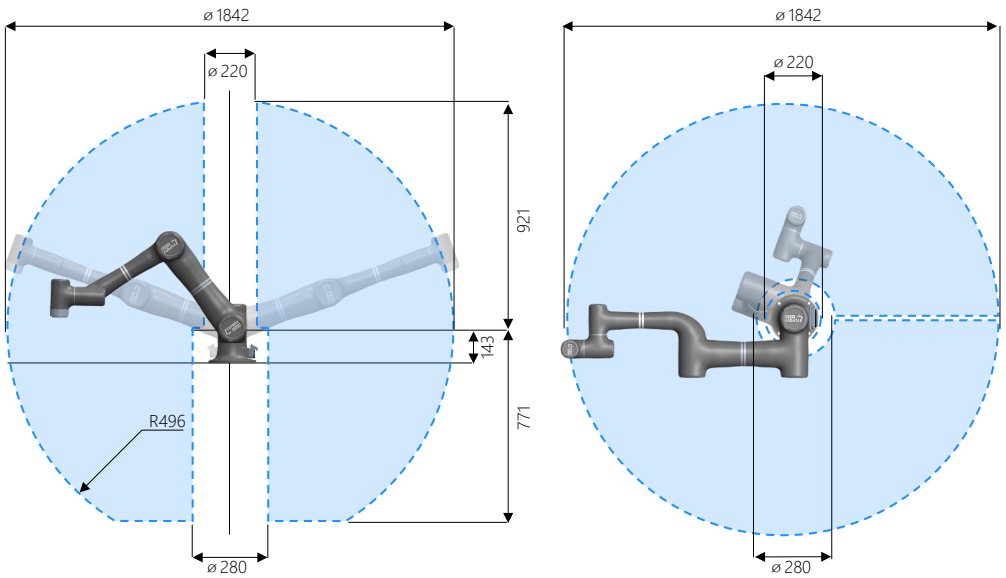
Robot ES5 to najmniejszy i najszybszy z robotów przemysłowych firmy Easy Robots. 6-osiowy robot ES5, waży zaledwie 27 kg i może manipulować ładunkami o masie do 5 kg. Ramię robota ES5 składa się z kilku podstawowych elementów, których nazwy mogą pojawić się w dalszej części instrukcji. Poszczególne elementy przedstawione zostały na rysunku 2.2.



Rysunek 2.2: Robot ES5

Przestrzeń robocza manipulatora

Przestrzeń robocza robota określona jest obszarem przedstawionym na rysunku 2.3. Maksymalny zasięg określony jest przez promień R mierzony od osi stawu 2. Praca w maksymalnym obszarze roboczym powinna odbywać się ze zmniejszonym udźwigiem i ze zmniejszoną dynamiką. Praca poniżej linii podstawy jest możliwa, o ile sposób zamocowania robota to umożliwia.



Rysunek 2.3: Zasięg robota ES5

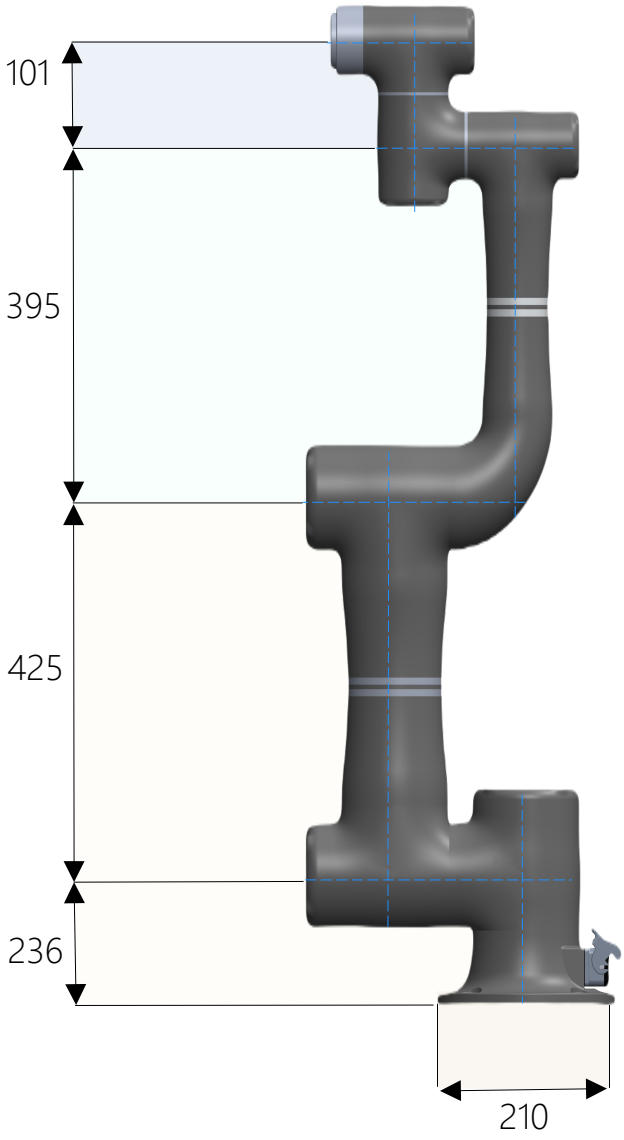
Należy unikać pracy robota w pobliżu podstawy oraz przegubu 1 ze względu na możliwość wystąpienia kolizji.

Parametry techniczne

Podstawowe parametry techniczne robota przedstawiono w poniżej.

Parametr	**Wartość**
Typ robota	ES5
Masa robota	27 kg
Obciążenie	do 5 kg
Zasięg	do 921 mm
Ilość stopni swobody	6
Zakres przegubu 1	± 180 °
Zakres przegubu 2	± 150 °
Zakres przegubu 3	± 156 °
Zakres przegubu 4	± 180 °
Zakres przegubu 5	± 180 °
Zakres przegubu 6	± 360 °
Prędkość przegubów	od 160 do 180 $°/s$
Prędkość narzędzia	do do 2 m/s
Powtarzalność	± 0,1 mm
Hałas	86 dB
Zasilanie	100-240 VAC, 50-60Hz
Temperatura pracy	7-40°C
Pobór mocy	500W, max 1000W
Wilgotność	max 80%

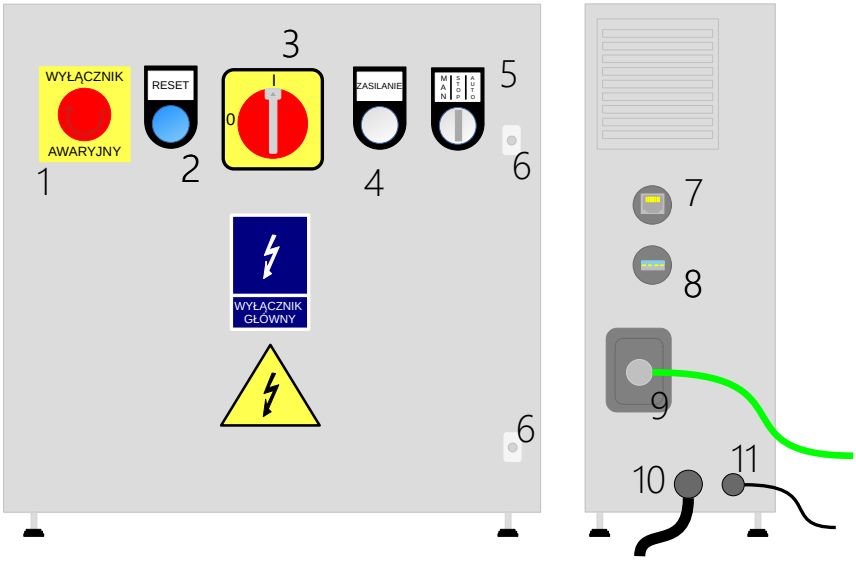
Wymiary ramienia przedstawiono na rysunku 2.1



Rysunek 2.1: Wymiary robota ES5

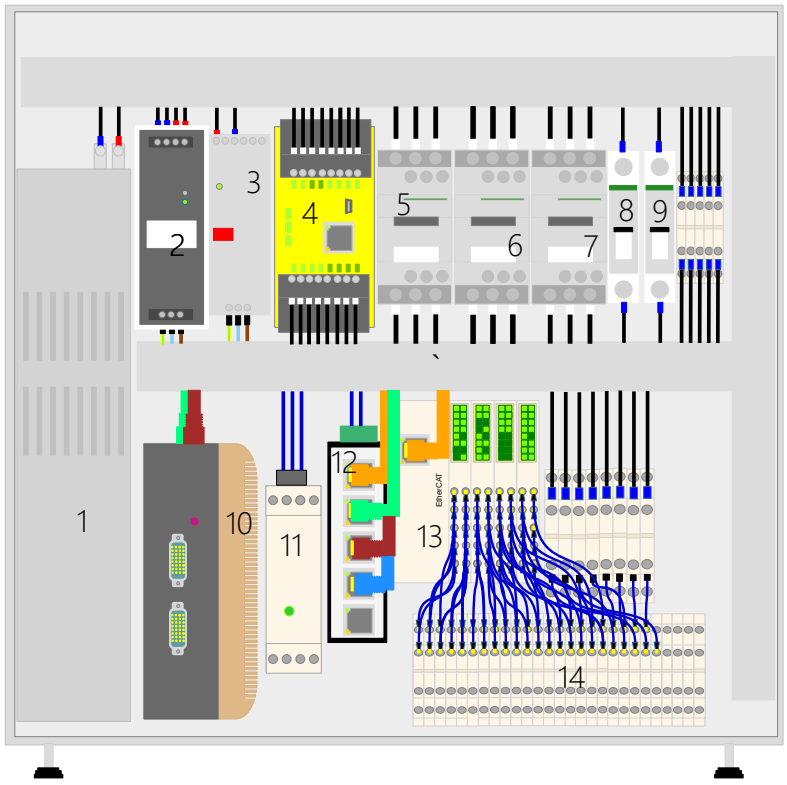
Szafa sterownicza

Na rysunku 2.2 przedstawiono widok frontu oraz boku szafy sterowniczej robota. Na bocznej ścianie szafy sterowania wyprowadzone zostały złącza Ethernet oraz USB komputera sterującego robota. Złącze USB może być wykorzystywane do magazynowania programów robota na nośnikach zewnętrznych. Złącze Ethernet służy do celów serwisowych oraz integracji robota z systemami zewnętrznymi. Pod złączami umieszczono wyprowadzenia przewodów: zasilania oraz łączących szafę sterowniczą z panelem oraz robotem. Na przedniej ścianie szafy sterowniczej umieszczone zostały: Włącznik zasilania głównego, przycisk zatrzymania awaryjnego, przełącznik kluczykowy oraz przycisk resetu. Stan zasilania oraz resetu jest sygnalizowany przez lampki kontrolne.



Rysunek 2.2: Szafa sterownicza: robota 1-wyłącznik awaryjny, 2- przycisk resetu, 3- włącznik główny, 4- lampka kontrolna zasilania, 5-przełącznik kluczykowy, 6-zamki drzwi szafy, 7-złacze Ethernet, 8-złącze USB, 9-przewód panelu, 10-przewód manipulatora, 11- przewód zasilający

.

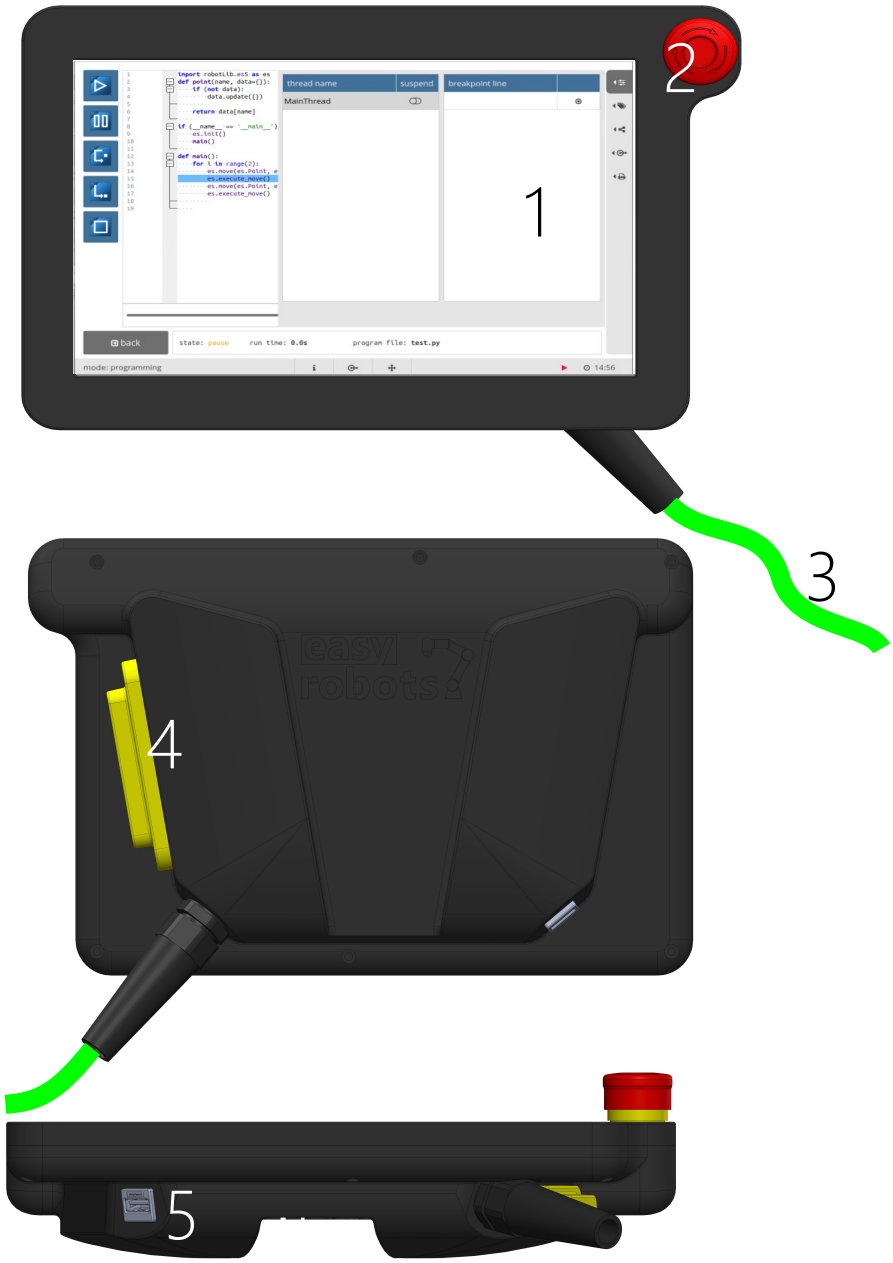


Rysunek 2.3: Zawartość szafy sterowniczej (rysunek poglądowy): 1-zasilacz napędów, 2-zasilacz komputera głównego, 3-zasilacz panelu sterowniczego, 4-sterownik bezpieczeństwa, 5,6,7-styczniki układu zasilania silników, 8,9-wyłączniki nadmiarowo-prądowe, 10-komputer sterujący, 11-moduł hamowania 12-przełącznik sieciowy, 13-moduł IO, 14-listwa zaciskowa

> Wewnątrz szafy sterowniczej panują napięcia sieciowe 100-230V niebezpieczne dla życia i zdrowia człowieka. W trakcie wykonywania połączeń w szafie sterowniczej robota zasilanie powinno być wyłączone

Panel sterujący

Panel sterujący robota służy do komunikacji z robotem. Służy m.in. do programowania robota oraz do uruchamiania programów. Widok panelu sterującego przedstawiono na rysunku poniżej.



Rysunek 2.4: Panel sterujący robota: 1-Ekran wyświetlacza, 2-Przycisk awaryjnego zatrzymania, 3-Przewód zasilający i komunikacyjny, 4-Przycisk zezwolenia na pracę, 5-złącze USB

2.2 Oprogramowanie

Opis

Oprogramowanie EScontrol znajdujące się w pamięci komuptera głównego oraz w pamięci panelu sterowania jest uruchamiane automatycznie po włączeniu robota. Odpowiada ono za kontrolę sprzętu, zapewnia komunikację z urządzeniami zewnętrznymi oraz interakcję z użytkownikiem za pomocą ekranu dotykowego.

Funkcjonalność

Podstawowa wersja oprogramowania dostarczona wraz z urządzeniem zapewnia użytkownikowi następujące możliwości:

- zmiana położenia robota w czasie rzeczywistym,
- pomiary właściwości fizycznych takich jak natężenie prądu, temperatura, napięcie itd. dla każdego z napędów robota,
- odczyt i zapis stanów logicznych kart Wejść/Wyjść dostępnych w szafie sterowniczej,
- zestawienie komunikacji z urządzeniami zewnętrznymi,
- zmiana parametrów sterowania napędami,
- projektowanie trajektorii ruchu robota,
- tworzenie i edycja skryptów sterujacych działaniem robota,
- uruchamianie zapisanych skryptów sterujących,
- tworzenie i realizacja wizualizacji procesu wykonywanego przez robota w czasie pracy na stanowisku,
- zarządzanie danymi i stworzonymi plikami,
- ograniczenie możliwości kontroli dla osób do tego niepowołanych,
- itd.

> *i* Oprogramowanie panelu sterującego posiada rozszerzenia pozwalające znacznie uprościć tworzenie skryptów sterujących w zależności od typu aplikacji realizowanej przez robota. W celu uzyskania informacji na ten temat zaleca się kontakt z firmą Easy Robots (10.2)

Użytkowanie

Dostarczone oprogramowanie jest przeznaczone wyłącznie do użytkowania robotów Easy Robots w sposób pozwalający na przygotowanie go do pracy na docelowym stanowisku oraz jego serwisowaniu.

> ! Za niedozwolone uznaje się każde użytkowanie mające na celu ingerencję w dostarczone oprogramowanie polegające na kopiowaniu oraz modyfikowaniu. Producent nie odpowiada za wynikające z tego tytułu szkody. Ryzyko ponosi wyłącznie użytkownik urządzenia.

3 Obsługa

3.1 Interfejs użytkownika

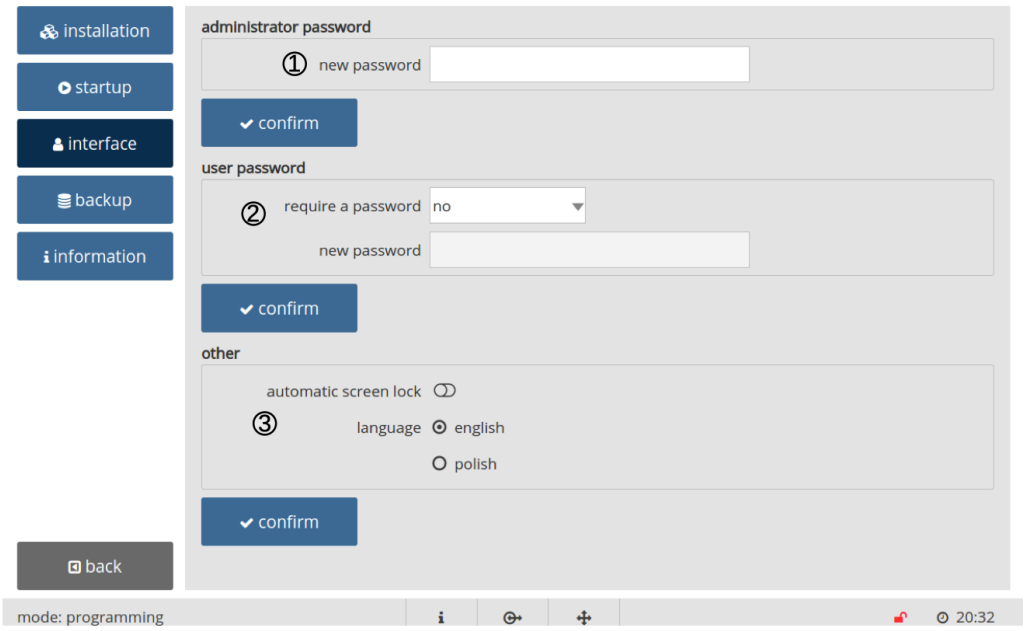
Oprogramowanie EScontrol

Interfejs użytkownika dla robotów serii ES firmy EasyRobots stanowi oprogramowanie EScontrol dostępne z poziomu panelu uczenia. Oprogramowanie umożliwia przede wszystkim na sterowanie manipulatorem, obsługę wejść/wyjść robota, jego konfiguracje oraz tworzenie i odtwarzanie programów. Poszczególne jego funkcjonalności zostały opisane w niniejszej instrukcji. Uzupełnieniem oprogramowania są specjalizowane moduły do konkretnych aplikacji m.in spawanie, paletyzowanie itp. Dokumentacje poszczególnych modułów stanowią uzupełnienie niniejszej instrukcji w zależności od przeznaczenia robota.

Ustawienia interfejsu

Podstawowe ustawienia interfejsu robota możliwe są z poziomu okna ustawień. Aby przejść do odpowiedniego widoku należy z ekrany głównego wybrać przycisk settings a następnie zakładkę interface . Okno ustawień interfejsu przedstawiono na rysunku poniżej. Z poziomu tego ekranu możliwe jest:

1. ustawienie hasła administratora.
2. ustawienie hasła użytkownika,
3. ustawienie automatycznej blokady ekranu i wybór języka interfejsu.



Rysunek 3.1: Ekran ustawień interfejsu użytkownika

i Dostęp do zakładki interface możliwy jest jedynie po zalogowaniu użytkownika o odpowiednim poziomie uprawnień - administrator.

3.2 Język interfejsu

Wybór języka interfejsu użytkownika

Wybór języka interfejsu użytkownika możliwy jest z poziomu zakładki interface (rys.3.1) Języki dostępne do wyboru to:

- angielski,
- polski.

W celu wyboru konkretnego języka konieczne jest wciśniecie przycisku confirm oraz ponowne uruchomienie robota.

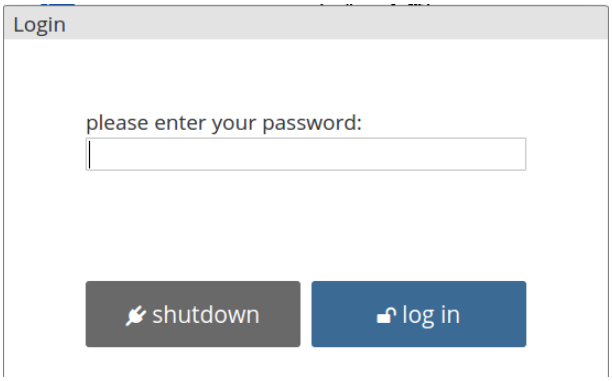
3.3 Zabezpieczenia dostępu

Poziom dostępu

Oprogramowanie EScontrol pozwala na ograniczenie dostępu do wszystkich lub wybranych funkcjonalności. Istnieją dwa poziomy dostępu:

- administratora - pozwala na pełny dostęp do funkcjonalności robota,
- użytkownika - pozwala na pełny dostęp do funkcjonalności robota z wyjątkiem modyfikacji ustawień robota oraz możliwości usuwania programów i konfiguracji zapisanych w pamięci.

Dostęp do poszczególnych poziomów uprawnień możliwy jest z poziomu okna logowania. Pojawia się ono po automatycznym wylogowaniu. Dostęp do okna logowania możliwy jest również z poziomu ekranu głównego - przycisk access level .



Rysunek 3.2: Okno ekranu logowania

Konfiguracja haseł dostępu

Konfiguracja haseł dostępu administratora oraz użytkownika możliwa jest z poziomu okna przedstawionego na rys. 3.1. Wprowadzone hasło należy potwierdzić przyciskiem odpowiednim przyciskiem confirm .

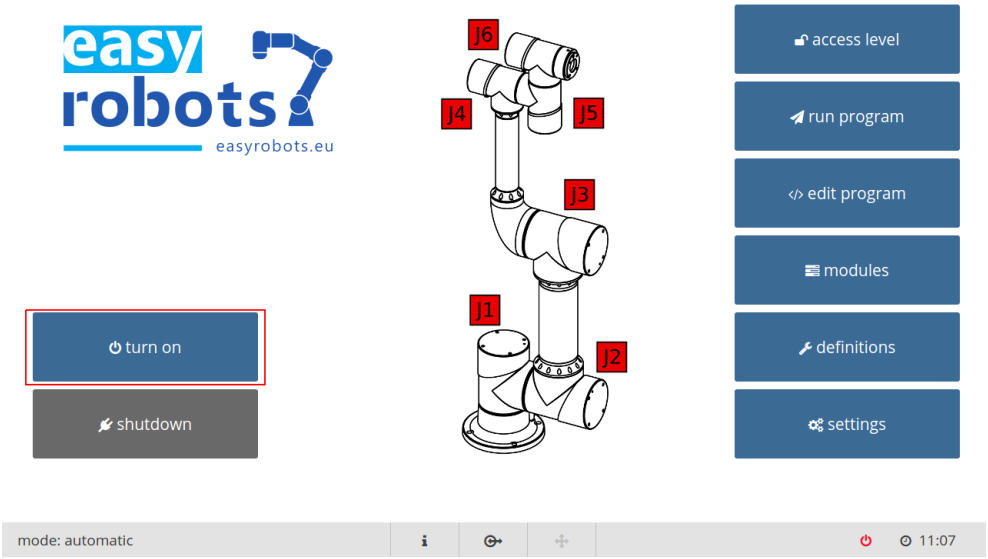
> *i* Zmiana haseł możliwa jest jedynie z poziomu uprawnień administratora

> *i* W ustawieniach domyślnych hasło użytkownika jest wyłączone. Oznacza to, że przy włączeniu robota nie ma potrzeby podawać żadnego hasła. Domyślne hasło dostępowe administratora to: 1234

3.4 Włączanie robota i uruchamianie manipulatora

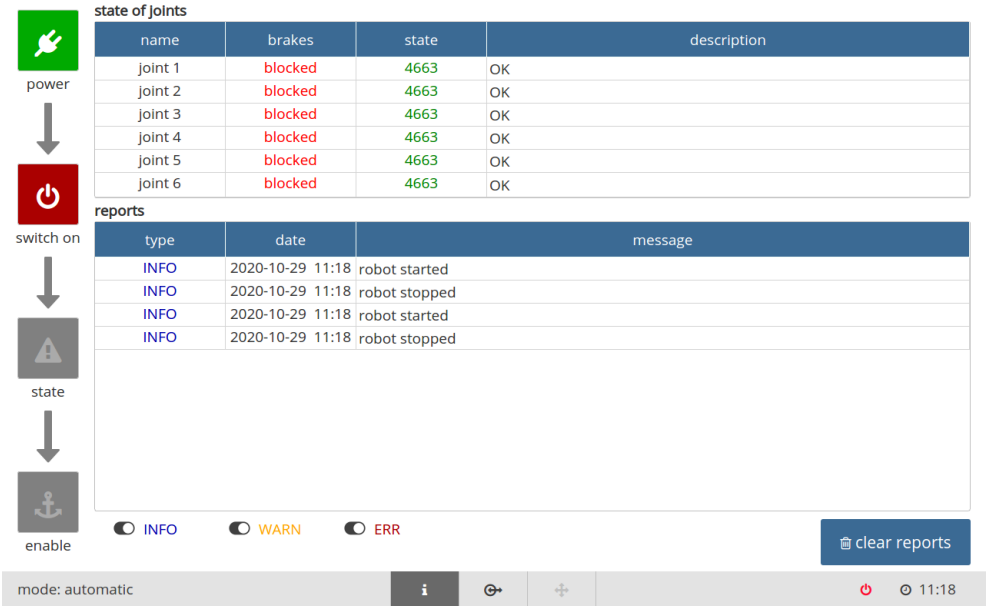
Włączanie robota

W celu włączenia robota należy przekręcić główny włącznik zasilania znajdujący się na szafie sterowniczej robota na pozycję "1". Robot zostanie wtedy zasilony, panel sterowania uruchomi się oraz włączy się aplikacja EScontrol. W celu włączenia robota należy nacisnąć przycisk turn on - rysunek 3.3.



Rysunek 3.3: Okno startowe aplikacji EScontrol z zaznaczonym przyciskiem uruchomienia robota.

Po naciśnięciu przycisku zostaniemy przeniesieni do zawierającego informacje o aktualnym stanie robota - rysunek 3.4.



Rysunek 3.4: Okno zawierające informacje o aktualnym stanie robota.

> *i* Gdy robot jest włączony przycisk turn on zmienia się w przycisk turn off

Kontrolki stanu robota

Aktualny stan robota odwzorowany jest za pomocą kontrolek znajdujących się po lewej stronie okna. W celu włączenia robota należy nacisnąć kontrolkę "switch on".

Kontrolka pokazuje czy robot jest zasilony, czy zresetowany jest obwód bezpieczeństwa.

Kontrolka ta przyjmuje kolor zielony jeśli robot jest włączony, w przeciwnym razie jest ona czerwona. Poprzez naciśnięcie kontrolki można włączyć lub wyłączyć robota

Kontrolka ta sygnalizuje stan robota. Jeśli jest ona zielona nie występują żadne błędy, jeśli jest ona czerwona- występują błędy robota, jeśli jest ona szara robot jest wyłączony. Po naciśnięciu kontrolki następuje próba zresetowania błędów.

Kontrolka ta sygnalizuje, czy robot może się poruszać tj. czy zwolnione są hamulce, czy wciśnięty jest przycisk zezwolenia jeśli robot jest w trybie programowania.

Status napędów i raporty

W oknie znajduje się również tabela zawierająca status każdego stawu, wyświetlany w niej jest status hamulca, aktualny stan stawu oraz krótki opis tego stanu. W drugiej tabeli poniżej znajdują się raporty dotyczące działania robota. Są one podzielone na 3 kategorie: informacja, ostrzeżenie oraz błąd. Aktualnie wyświetlane kategorie można zmieniać za pomocą przełączników znajdujących się pod tabelą. Przycisk clear reports służy do wyczyszczenia tabeli raportów.

3.5 Wyłączanie manipulatora i zamykanie systemu

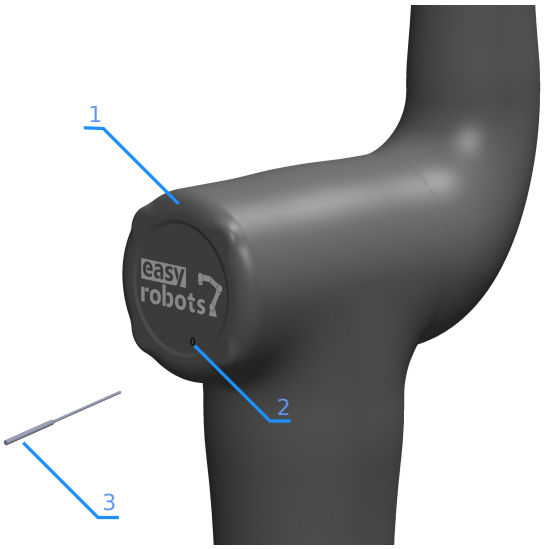
Wyłączanie robota

W celu wyłączenia robota należy zatrzymać program, jeśli jest aktualnie uruchomiony a następnie nacisnąć przycisk turn off w oknie startowym lub kontrolkę w oknie informującym o stanie robota - patrz rozdział 3.4.

3.6 Ręczne przesuwanie osi robota

Ręczne zwalnianie hamulców robota

Robot serii ES umożliwia w stanie bezprądowym ręczne zwolnienie hamulców i obrót każdej z osi. W elementach plastikowych (1) zamykających poszczególne stawy znajduje się otwór (2), w którym należy umieścić element zwalniający (3), docisnąć i przytrzymać. Nastąpi zwolnienie hamulca, co umożliwi obrót. W przypadku ręcznego obrotu osi „1" jedna dłoń jest na korpusie stawu „1", zaś obrót wymuszany jest przez dłoń umieszczoną na korpusie stawu „2". Dla osi od „2" do „6" obrót należy wykonać trzymając jedną dłoń na korpusie danego stawu, zaś drugą dłonią należy wymusić obrót elementu po stronie wyjściowej stawu (4). Po ustawieniu wymaganej pozycji robota należy usunąć element (3) – hamulec zablokuje ruch obrotowy osi.



Rysunek 3.5: Ręczne zwalnianie hamulca. 1-Osłona napędu, 2-Otwór do zwalniania hamulca, 3-Klucz do zwalniania hamulca (dołączony do zestawu)

Ręczne zwalnianie hamulców powinno być wykonywane przez dwie osoby.

Zwalniając ręcznie hamulce może nastąpić samoistne złożenie robota – zaleca się zachowanie szczególnej ostrożności.

Przy ręcznym zwalnianiu hamulców nie należy szarpać ze elementy robota – może to doprowadzić do jego uszkodzenia i utraty gwarancji.

Maksymalny obrót pojedynczej osi do 180º.

4 Pierwsze uruchomienie

4.1 Procedura pierwszego uruchomienia

Pierwsze uruchomienie

Robot dostarczany przez producenta jest wstępnie skonfigurowany i skalibrowany do poprawnej pracy. Procedura pierwszego uruchomienia zasadniczo nie odbiega od procesu uruchamiania robota przedstawionego w poprzednim rozdziale. Dodatkowo podczas pierwszego uruchomienia użytkownik powinien zweryfikować:

- poprawność odblokowywania hamulców,
- konfigurację domyślnych pozycji przegubów.

Działanie hamulców

Aby zweryfikować poprawność odblokowywania hamulców należy:

1. Uruchomić robota zgodnie z procedurą przedstawioną w poprzednim rozdziale
2. Po wciśnięciu przycisku switch on zweryfikować poprawność odblokowania hamulców dla każdego z przegubów. Jeżeli któryś z napędów wskazuje status hamulca **blocked** oznacza to że hamulce wymagają ponownej kalibracji. Jeżeli hamulce odblokują się poprawnie (status **relesed**, należy przejść do następnego kroku
3. Zmienić pozycje stawów robota, a następnie wcisnąć jeden z przycisków bezpieczeństwa. Hamulce robota powinny ulec zablokowaniu - status . **blocked**
4. Zwolnić przycisk bezpieczeństwa oraz zresetować obwód bezpieczeństwa wciskając przycisk RESET znajdujący się na szafie sterowniczej
5. Zweryfikować poprawność zwolnienie hamulców - (status **relesed**)

Podczas procedury odblokowywania hamulców robot może wykonywać drobne ruchy w celu zluzowania rygli hamulców

Jeżeli którykolwiek z hamulców nie odblokowuje się poprawnie, należy dokonać konfiguracji hamulców zgodnie z procedurą przedstawiona w dalszej części tego rozdziału

Pozycje domyślne przegubów

W celu weryfikacji poprawnej konfiguracji domyślnych pozycji przegubów (pozycji synchronicznej), należy:

1. Uruchomić robota zgodnie z procedurą przedstawioną w poprzednim rozdziale.
2. Przejść do okna ruchu robota.
3. Wcisnąć przycisk synchro position .
4. Przemieścić robota do pozycji synchronicznej klikając przycisk move simple

5. Zweryfikować pozycje synchroniczną - suwaki pozycji w pozycji środkowej oraz porywające się strzałki na poszczególnych korpusach napędów robota (rysunek 4.7).

Rysunek 4.6: Weryfikacja poprawnej domyślnej pozycji przegubów

Rysunek 4.7: Poprawna pozycja synchroniczna przegubu

STOP

Jeżeli pozycja któregokolwiek ze stawów robota odbiega od pozycji synchronicznej należy ustawić robota w pozycji synchronicznej (zgodnie ze strzałkami na korpusach napędów), a następnie ustawić aktualne pozycje napędów jako domyślne według procedury przedstawionej w dalszej części tego rozdziału

4.2 Ustawienia hamulców

Hamulec ryglowy

Robot serii ES firmy EasyRobots wyposażony jest w hamulec ryglowy na każdej z 6-ciu osi manipulatora. Ze względu na specyfikę pracy hamulca ryglowego który może naciskać na tarczę hamulca konieczne jest luzowanie rygla hamulca przed próbą jego zwolnienia. Aby hamulce robota były poprawnie odblokowywane, konieczna jest ich kalibracja.

Kalibracja

Kalibracja hamulców robota możliwa jest z poziomu ekranu ustawień serwisowych. Aby do niego przejść należy w menu głównym wybrać settings , a następnie kolejno installation oraz brakes . Na rysunku poniżej przedstawiono okno konfiguracji hamulców. Poszczególne opcje konfiguracji to:

1. Brakes test - pozwala na ręczne zablokowanie/odblokowanie hamulca.
2. Smart brake enabe - włączanie funkcji automatycznego luzowania stawu podczas odblokowywania hamulca.
3. Brake off measure [ADC] - wartość czujnika gdy hamulec jest zwolniony.
4. Brake on measure [ADC] - wartość czujnika gdy hamulec jest aktywowany.
5. Threshold value [ADC] - wartość progu zmiany stanu hamulca.
6. Auto-setting - aktywowanie automatycznej konfiguracja hamulca.
7. Confirm - zapisanie konfiguracji hamulców
8. Start auto-setting - uruchomienie procedury automatycznej konfiguracji hamulców.

Rysunek 4.8: Okno konfiguracji hamulców

4.3 Kalibracja pozycji

Układ napędowy

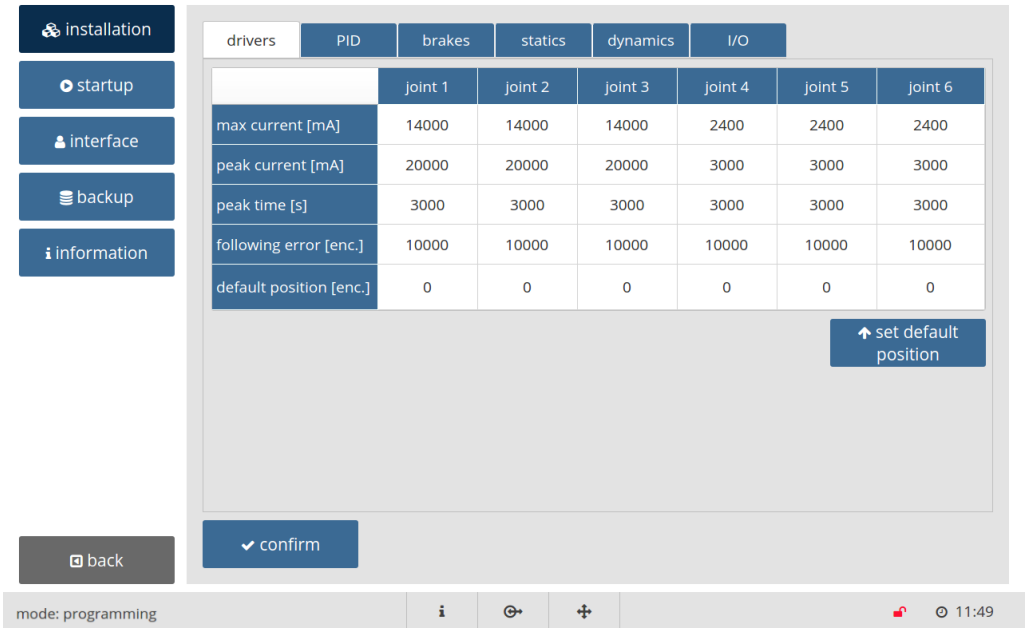
Podstawową wielkością, wymagającą sprawdzenia podczas pierwszego uruchomienia, jest pozycja domyślna (synchroniczna). Jest to takie położenie wszystkich przegubów manipulatora, które determinuje jego postawę pionową. W przypadku stwierdzenia niepoprawnej pozycji synchronicznej manipulatora, należy dokonać jej korekty. W tym celu należy przejść do ustawień oraz wybrać kartę o nazwie installation oraz zakładkę drivers . Ustawienia dotyczące pozycji synchronicznej znajdują się w tabeli pod nazwą "default position".

> *i* Karta installation dostępna jest jedynie po uzyskaniu odpowiednich uprawnień dostępu, poprzez wpisanie hasła administratora urządzenia.

Kalibracja

W celu wprowadzenia korekty pozycji domyślnych należy postępować według niżej przedstawionej procedury.

1. Ustawić manipulator w odpowiedniej pozycji za pomocą okna ruchu lub ręcznie (po manualnym zwolnieniu hamulców),
2. Wcisnąć przycisk set default position ,
3. Wcisnąć przycisk confirm .
4. Pojawi się okno potwierdzające zapis oraz informujące o konieczności ponownego uruchomienia robota.
5. Uruchomić ponownie robota.



Rysunek 4.9: Okno ustawień serwonapędów

5 Bezpieczeństwo

5.1 Ogólne zasady bezpieczeństwa

Należy pamiętać, że robot może generować duży moment obrotowy na przegubach oraz wykonywać bardzo szybkie ruchy, które w skutek kolizji mogą przyczynić się do uszkodzenia ciała lub zagrozić życiu operatora. Ponadto robot w trakcie nieprawidłowej pracy lub niewłaściwego użytkowania może uszkodzić część stanowiska pracy lub swoją konstrukcję.

Osoba pracująca na stanowisku zrobotyzowanym w każdym momencie powinna mieć świadomość w jakiej pozycji znajduje się robot oraz jaki jest stan jego pracy: czy program jest w trakcie wykonywania lub czy robot zatrzymany awaryjnie lub w trakcie postoju.

Instalacja i obsługa robota zawsze musi być wykonywana przez przeszkolony personel.

Robot przemysłowy może być użytkowany tylko w technicznie sprawnym stanie oraz zgodnie z jego przeznaczeniem i z uwzględnieniem zasad bezpieczeństwa oraz grożących niebezpieczeństw.

Wszelkie usterki mogące mieć wpływ na bezpieczeństwo pracy na stanowisku zrobotyzowanym powinny być niezwłocznie usuwane.

Naprawy, konserwacje i czyszczenie urządzenia należy przeprowadzać zgodnie z wytycznymi niniejszej instrukcji oraz obowiązującymi przepisami BHP. W innym przypadku grozi to obrażeniami ciała lub uszkodzeniem urządzenia.

Należy stosować tylko i wyłącznie oryginalne części zamienne, zgodne ze specyfikacją.

Odpowiedzialność

Informacje dotyczące bezpieczeństwa zawarte w niniejszym dokumencie nie mogą być podstawą do wystąpienia przeciwko firmie EasyRobots Sp. z o.o. ponieważ przestrzeganie opisanych tu zasad bezpieczeństwa nie daje gwarancji, że robot przemysłowy nie wyrządzi szkód materialnych lub obrażeń ciała.

Zabrania się wykonywać modyfikacji robota bez zezwolenia firmy EasyRobots Sp. z o.o. Za uszkodzenie robota lub szkody wyrządzone w skutek nieautoryzowanej ingerencji odpowiedzialność ponosi wyłącznie użytkownik.

Za instalację i użytkowanie robota, zgodnie z wymogami bezpieczeństwa obowiązującymi w kraju, gdzie instalacja ma miejsce, odpowiada integrator.

Producent nie ponosi odpowiedzialności za wszelkie straty i szkody powstałe w wyniku nieprawidłowego montażu urządzenia, bądź korzystania z niego niezgodnie z przeznaczeniem, a także postępowania niezgodnego z zapisami niniejszej instrukcji.

Instalacja elektryczna

Instalacja elektryczna zasilająca robota powinna być wykonana zgodnie z obowiązującymi normami oraz posiadać odpowiednie parametry elektryczne, potwierdzone pomiarami przeprowadzonymi przez uprawnionego elektryka.

Urządzenie jest pod napięciem. Dostęp do urządzenia jest ograniczony tylko do autoryzowanych firm oraz wykwalifikowanych i przeszkolonych pracowników.

5.2 Personel

Każda osoba wykonująca prace związane z robotami firmy EasyRobots musi uprzednio przejść stosowne szkolenie. Przed rozpoczęciem prac personel musi być poinformowany o potencjalnych zagrożeniach oraz zakresie wykonywanych prac.

Integrator

Integrator ponosi odpowiedzialność za montaż, wdrożenie robota zgodnie z obowiązującymi przepisami. Do obowiązków integratora należą:

- montaż robota,
- wdrożenie,
- realizacja kooperacji z maszynami współpracującymi,
- zastosowanie niezbędnych technik bezpieczeństwa np. wygrodzeń, przycisków awaryjnego zatrzymania, itd.,
- wykonanie analizy ryzyka dla stanowiska,
- wystawienie deklaracji zgodności,
- nadanie oznaczenia CE,
- przygotowanie instrukcji obsługi stanowiskowej zgodnie z obowiązującymi przepisami,
- przeprowadzenie niezbędnych szkoleń dla użytkownika.

Operator

Osoba sklasyfikowana jako operator stanowiska powinna być przeszkolona przez integratora w zakresie przepisów BHP oraz zasad obsługi stanowiska, na którym zainstalowany jest robot. Do zadań operatora najczęściej należą następujące czynności:

- włączanie i wyłączanie robota,
- wybór programu z panelu sterowania robota,

- awaryjne wyłączanie robota poprzez przycisk awaryjnego zatrzymania w sytuacji zagrożenia osób znajdujących się w otoczeniu stanowiska lub wystąpienia sytuacji awaryjnej, w której istnieje zagrożenie uszkodzenia robota.

Programista

Programista jest osobą, która jest przeszkolona przez producenta lub integratora. Podstawowym zadaniem programisty jest tworzenie programów pracy robota zgodnie z jego przeznaczeniem po integracji.

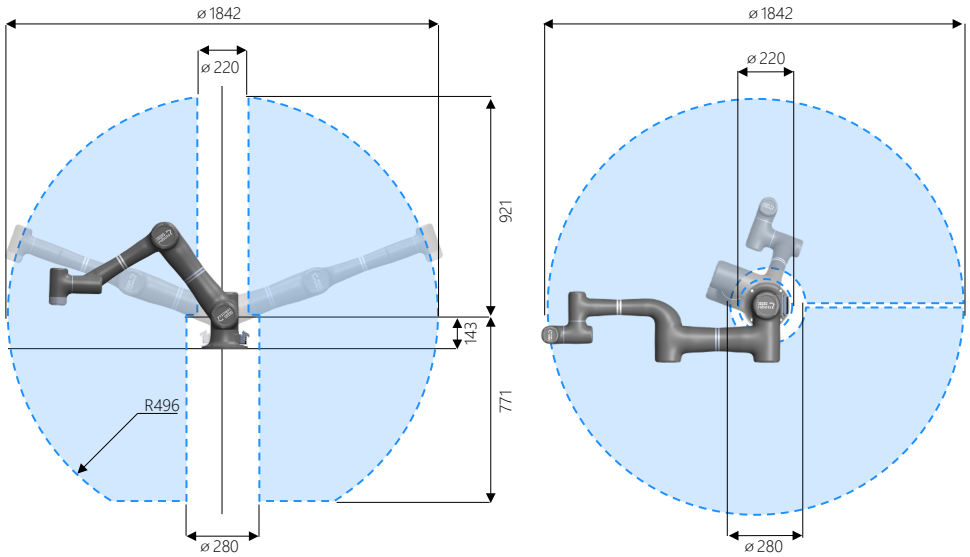
Serwisant

Serwisant jest osobą, która jest przeszkolona przez producenta lub integratora do realizacji prac naprawczych lub konserwacyjnych, wymagających demontażu lub regulacji poszczególnych elementów robota, a także stanowiska. Do celów kontrolnych serwisant może wykonywać zakres zadań przewidziany dla programisty lub operatora.

5.3 Strefa bezpieczeństwa

Obszar roboczy

Obszar roboczy musi być odseparowany za pomocą urządzeń ochronnych w sposób gwarantujący bezpieczeństwo osobie znajdującej się w pobliżu stanowiska zrobotyzowanego (rys. 5.10).



Rysunek 5.10: Zasięg robota ES5

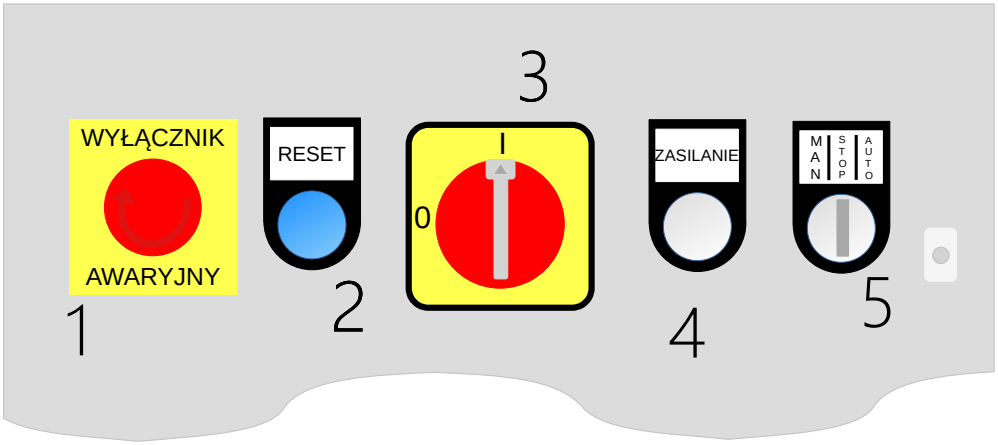
Urządzenia ochronne

Zewnętrzne urządzenia ochronne takie jak drzwi z wyłącznikiem bezpieczeństwa, kurtyny bezpieczeństwa, maty bezpieczeństwa, itp. powinny znajdować się poza obszarem roboczym. Umiejscowienie tych urządzeń powinno uwzględniać czas zatrzymania awaryjnego.

5.4 Środki bezpieczeństwa

Szafa sterownicza

Panel szafy sterowniczej wyposażony został w szereg urządzeń do kontroli zasilania i obwodów bezpieczeństwa robota.



Rysunek 5.11: Szafa sterownicza - kontrola i bezpieczeństwo. 1-Przycisk zatrzymania awaryjnego, 2-Przycisk/lampka kontrolna resetu, 3-Włącznik główny zasilania, 4-Lampka kontrolna zasilania, 5-Przełącznik kluczykowy trybu pracy

Obwód bezpieczeństwa

System sterowania Robota wyposażony jest w niezależny obwód bezpieczeństwa, pozwalający na natychmiastowe wyłączenie zasilania elementów napędowych w momencie wystąpienia zagrożenia lub sytuacji awaryjnej. System bezpieczeństwa robota zaprojektowany jest zgodnie z wytycznymi normy PN-EN ISO 13849-1:2016 i zapewnia poziom niezawodności d kategorii 3.

Przycisk zatrzymania awaryjnego

Na panelu operatora oraz na obudowie szafy sterowniczej znajduje się przycisk zatrzymania awaryjnego. Przycisk taki przerywa obwód zasilania stopnia mocy robota. Robot zatrzyma się wówczas zatrzymaniem z kategorią zatrzymania 0. Aby przywrócić zasilanie napędów robota, należy wyciągnąć przycisk do pozycji domyślnej i wcisnąć przycisk resetu obwodu bezpieczeństwa.

W normalnych warunkach, gdy nie występują sytuacje niebezpieczne, obwód bezpieczeństwa nie powinien być używany do zatrzymania urządzenia. W przypadku normalnych warunków pracy należy w tym celu używać odpowiednich elementów sterujących panelu operatorskiego, jak opisano w kolejnych rozdziałach.

Przycisk zezwolenia

Przycisk zezwolenia jest to przełącznik trójpozycyjny, zlokalizowany na spodniej stronie panelu operatora (rys. 5.12). W trakcie pracy w trybie programowania przycisk ten musi być wciśnięty do połowy. W innym wypadku ruch robotem jest niemożliwy ze względu na przerwany obwód zasilania stopnia mocy. Specyfika działania przycisku zezwolenia ułatwia bezpieczną pracę operatora w ob-

szarze przestrzeni roboczej robota. W sytuacji zagrożenia odpuszczenie pozycji środkowej przycisku powoduje gwałtowne zatrzymanie robota. Podobny efekt nastąpi po wciśnięciu przycisku do pozycji końcowej. Funkcjonalność przycisku zezwolenia jest określona w normie ISO 10218:1.

Rysunek 5.12: Panel sterujący robota z przyciskiem zezwolenia.

5.5 Dodatkowe wyposażenie ochronne

Wygrodzenia

W celu zapewnienia bezpieczeństwa integrator zobowiązany jest zastosować wygrodzenia. Wygrodzeń tych nie wolno usuwać. Wyjątek stanowią operacje przeprowadzenia konserwacji/napraw, po których wygrodzenia muszą być ponownie zainstalowane ze wszystkimi mocowaniami i wszelkimi urządzeniami zabezpieczającymi.

5.6 Tryby pracy

Tryb programowania

Sterowanie robotem przez operatora, w trybie programowania:

- wyklucza pracę automatyczną,
- ruch robota jest możliwy przy włączonym przycisku zezwolenia,
- prędkość robota jest ograniczona do 250 mm/s dla każdego poruszającego się członu.

Tryb pracy automatycznej

Tryb pracy automatycznej jest wykorzystywany podczas wykonywania automatycznej, cyklicznej pracy robota, z maksymalną zaplanowaną dla opracowanych trajektorii prędkością. W tym trybie powinny działać wszystkie dodatkowe urządzenia bezpieczeństwa. Podczas pracy w trybie automatycznym:

- robot może poruszać się z maksymalnymi dopuszczalnymi prędkościami,

- przycisk zezwolenia nie jest obsługiwany,
- zabrania się sterowania w trybie ręcznym.

W trakcie pracy w trybie automatycznym zabronione jest przebywanie człowieka w obrębie obszaru roboczego

Przełączanie trybów pracy

Pomiędzy poszczególnymi trybami pracy robota można przełączać za pomocą klucza umieszczonego na obudowie szafy sterowniczej (rys. 5.11).

5.7 Zatrzymanie awaryjne

Kategorie bezpieczeństwa

Typ zatrzymania awaryjnego definiuje się poprzez pojęcie "Stop Category". Jest to klasyfikacja określająca w jaki sposób ruch robota powinien być zatrzymany aby zachować znamiona zatrzymania bezpiecznego. Wyróżnia się trzy różne typy (na podstawie EN 60204-1):

- Kategoria 0 - niekontrolowane zatrzymanie silnika - w sytuacji zagrożenia zasilanie zostaje natychmiast odłączone od silnika robota, a hamulce załączone. Silnik przestaje generować moment obrotowy, lecz nadal obraca się pod wpływem działania sił bezwładności, aż do momentu całkowitego zatrzymania.
- Kategoria 1 - energia jest wciąż dostarczana do napędu, którego zatrzymanie ma nastąpić w skutek procesu hamowania. Po całkowitym zatrzymaniu następuje odcięcie zasilania.
- Kategoria 2 - energia jest wciąż dostarczana do napędu, którego zatrzymanie ma nastąpić wskutek procesu hamowania. Po całkowitym zatrzymaniu NIE następuje odcięcie zasilania.

Parametry zatrzymania

W tabeli poniżej przedstawiono wyniki pomiarów czasu i dystansu pokonanego przez punkt TCP przy zatrzymaniu wskutek wciśnięcia przycisku zatrzymania awaryjnego (kategoria zatrzymania 0). Pomiar został dokonany w skrajnie niekorzystnym położeniu. Pomiar uwzględnia okres od momentu wystawienia sygnału zatrzymania, do czasu osiągnięcia zerowej prędkości.

	Odległość zatrzymania	**Czas zatrzymania**
Przegub 1-2	80°	0,5[s]
Przegub 3	64°	0,4[s]
TCP	0,85[m]	0,5[s]

Tabela 5.1: Parametry zatrzymania awaryjnego

6 Sterowanie robotem

6.1 Obsługa modułu Wejść/Wyjść

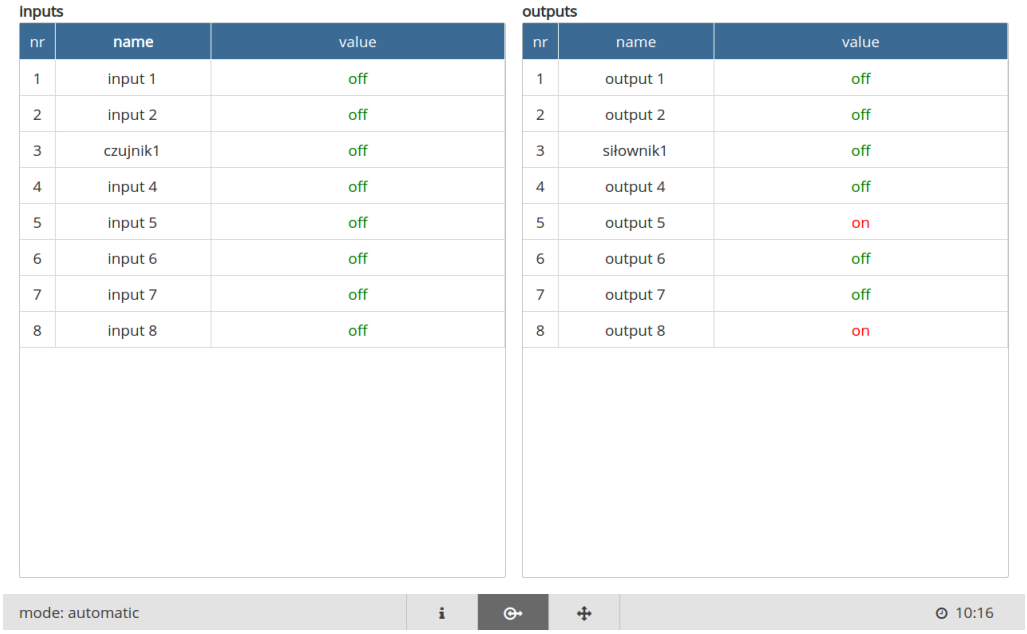
Okno kontroli I/O

Do obsługi modułu Wejść/Wyjść przeznaczone jest specjalne okno. Można je otworzyć za pomocą przycisku znajdującego się na pasku stanu robota - przycisk zaznaczony został na rysunku 6.13

Rysunek 6.13: Przycisk do włączenia okna obsługi wejść/wyjść.

Obsługa wejść oraz wyjść

Okno obsługi wejsc wyjść przedstawione zostało na rysunku 6.14. W tabelach wyświetlany jest aktualny stan każdego z wejść oraz wyjść cyfrowych. Możliwa jest zmiana stanu wyjścia cyfrowego poprzez naciśnięcie przycisku w kolumnie "value" dla wybranego wyjścia. Ponadto możliwa jest również zmiana nazwy dla każdego wejścia oraz wyjścia, w tym celu należy zedytować tekst w kolumnie "name" dla odpowiedniego wejścia lub wyjścia. Na rysunku 6.14 zmienione zostały nazwy dla trzeciego wejścia oraz wyjścia.

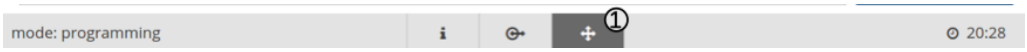


Rysunek 6.14: Okno obsługi wejść oraz wyjść cyfrowych.

6.2 Sterowanie manipulatorem

Okno ruchu

Do poruszania manipulatorem służy specjalny ekran, który dostępny jest z poziomu paska stanu robota - rysunek 6.15



Rysunek 6.15: Przycisk włączenia ekranu ruchu manipulatora

Ruch manipulatorem

Na rysunku 6.16 przedstawiono główne okno służące do poruszania ramieniem robotycznym. Okno to składa się z następujących sekcji:

1. parametrów ruchu,
2. zdefiniowanych pozycji manipulatora,
3. poruszania ramieniem robota,
4. poleceń ruchu.

Rysunek 6.16: Główne okno do poruszania ramieniem robota - współrzędne przegubów.

Parametry ruchu

Korzystając z suwaków można zmieniać prędkość (*speed*) oraz przyspieszenie (*acceleration*). Przyciski po prawej stronie służą do ustawiania zapisanych prędkości i przyspieszeń - domyślnie tryb wolny 1% prędkości i 5% przyspieszenia oraz analogicznie tryb normalny 20% i 20% i tryb szybki 50% i 35%. Aby zmienić domyślną wartość należy ustawić prędkość oraz przyspieszenie na żądaną wartość oraz przytrzymać klawisz wybranego trybu przez dwie sekundy.

Poruszanie ramieniem robota

Poruszanie ramieniem robota możliwe jest poprzez wybranie jednej z czterech dostępnych zakładek:

base poruszanie robotem w przestrzeni kartezjańskiej względem pozycji ope-

ratora,

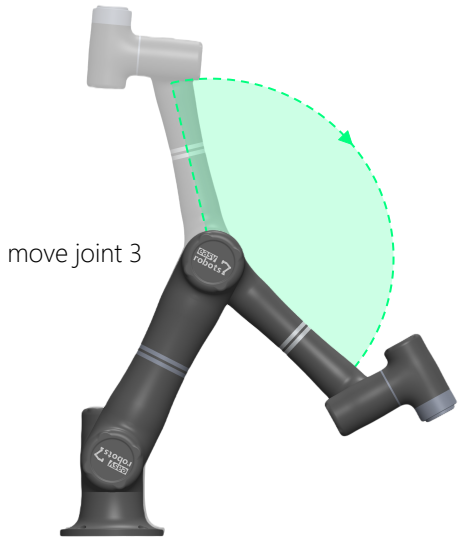
coords poruszanie robotem w przestrzeni kartezjańskiej względem wybranego układu odniesienia,

tool poruszanie robotem w przestrzeni kartezjańskiej względem wybranego układu współrzędnych narzędzia,

joint poruszanie robotem w przestrzeni przegubów.

Przestrzeń przegubów

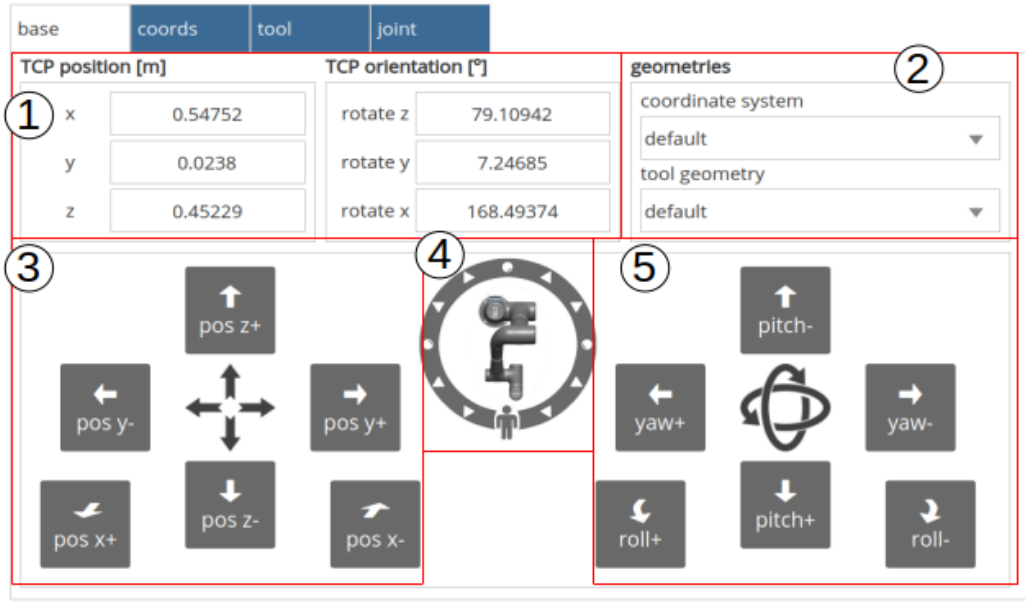
Klikając na zakładkę joint możliwe jest niezależne poruszanie każdym z przegubów robota w przestrzeni przegubów. Za pomocą strzałek użytkownik może poruszać wybranym stawem manipulatora. Po nastawieniu docelowej pozycji przegubów robota (suwaki lub pola edycyjne) i wybraniu przycisku move simple z sekcji poleceń, ramię robotyczne przemieści się do wybranej pozycji. Przykładowy ruchu w przestrzeni przegubów przedstawiono na rysunku poniżej.



Rysunek 6.17: Ruch jednym z przegubów manipulatora

Przestrzeń kartezjańska

Zakładki base , coords , tool umożliwiają poruszanie robotem w przestrzeni kartezjańskiej tj. przemieszczanie efektora robota w przestrzeni trójwymiarowej (x,y,z). Zakładka base pozwala na poruszanie robotem względem pozycji operatora.



Rysunek 6.18: Sekcja ruchu w przestrzeni kartezjańskiej

Sekcja poruszania robotem w przestrzeni kartezjańskiej składa się z:

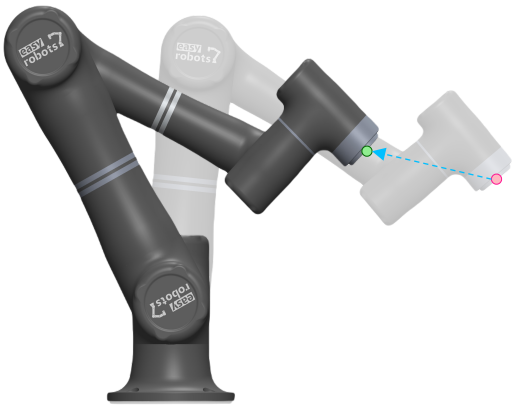
1 Sekcja pozycji oraz orientacji punktu centralnego narzędzia (TCP - ang. tool center point) względem środka układu współrzędnych. Wyświetla dane o położeniu i rotacji oraz pozwala na wprowadzenie nowej pozycji efektora. Wartości wyświetlane w tej sekcji są zależne od wybranego układu współrzędnych oraz geometrii narzędzia zdefiniowanych w sekcji nr 2.

2 Sekcja geometrii pozwala na odniesienie ruchu względem zdefiniowanych wcześniej układów współrzędnych oraz rotacji względem zdefiniowanych geometrii narzędzia.

3 Sekcja przycisków do przemieszczania efektora robota w przestrzeni (x,y,z).

4 Selektor orientacji robota. Pozwala na ustawienie orientacji operatora względem robota. Uzyskuje się to poprzez przeciągnięcie po ekranie obwodu okręgu wokół grafiki robota, do pozycji, która odzwierciedla położenie operatora względem robota. Powoduje to synchronizację kierunków oznaczonych na przyciskach z pola 3 i 5 zgodnie z kierunkami intuicyjnymi dla operatora.

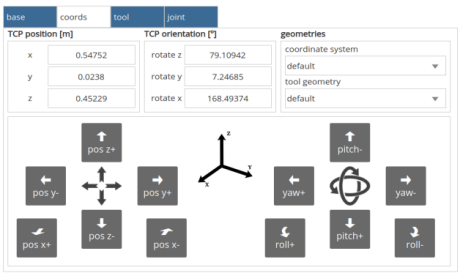
5 Sekcja przycisków do rotowania efektorem robota względem punktu centralnego narzędzia.



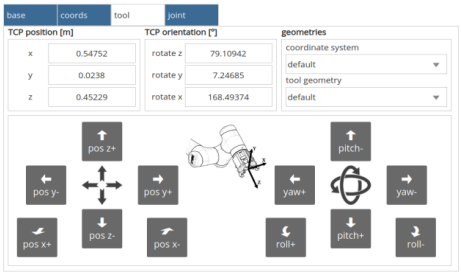
Rysunek 6.19: Przykładowy ruch robota w przestrzeni kartezjańskiej

Po wybraniu zakładki coords wyświetlony zostanie widżet przedstawiony na rysunku 6.20a. W tym trybie osie ruchu robota pokrywają się z osiami wybranego układu współrzędnych.

Po wybraniu zakładki tools wyświetlony zostanie widżet przedstawiony na rysunku 6.20b. W tym trybie osie ruchu robota pokrywają się z osiami wybranego układu współrzędnych narzędzia. Osie domyślnego układu współrzędnych narzędzia przedstawione są na rysunku, zastępującym selektor orientacji robota.



(a) Okno ruchu względem układu odniesienia.



(b) Okno ruchu względem układu współrzędnych narzędzia.

Rysunek 6.20

Przed wykonaniem ruchu w przestrzeni kartezjańskiej należy upewnić się, że ustawiono prawidłowy układ odniesienia oraz geometrię narzędzia. Więcej informacji o układach odniesienia oraz geometrii narzędzia znaleźć można w rozdziale 9 Konfiguracja

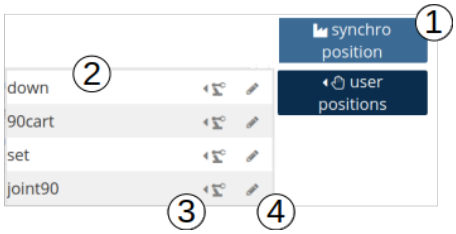
Pozycje predefiniowane

W celu ułatwienia pracy użytkownika można zdefiniować maksymalnie 4 pozycje manipulatora (np. przed gniazdem produkcyjnym, itd.) i przemieszać manipulator do nich kliknięciem odpowiedniego przycisku i wydaniem polecenia ruchu. Na rysunku poniżej przedstawiono sekcje zdefiniowanych pozycji manipulatora. Sekcja ta składa się z następujących przycisków:

1. wybrania pozycji synchronicznej,
2. wybrania predefiniowanej pozycji robota,
3. przypisywania pozycji,

4. zmiany nazwy zapisanej pozycji.

Wybranie synchronicznej lub predefiniowanej pozycji robota spowoduje wpisanie pozycji w odpowiednie pole edycyjne. Przemieszczenie robota do wybranej pozycji możliwe jest za pomocą przycisków poleceń ruchu opisanych poniżej. Zdefiniowana pozycja użytkownika może zostać zmodyfikowana za pomocą przycisku przypisania (3).

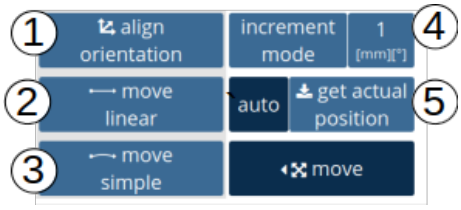


Rysunek 6.21: Zdefiniowane pozycje robota

Przyciski poleceń ruchu

Sekcja poleceń ruchu składa się z następujących przycisków:

1. wyrównania orientacji do najbliższej pełnej wartości kąta (co 45 stopni),
2. ruchu liniowego do zadanego punktu,
3. ruchu swobodnego do zadanego punktu,
4. ustawienie trybu inkrementalnego,
5. pobrania aktualnej pozycji i rotacji robota, przycisk auto odpowiada za automatyczne aktualizowanie pozycji i rotacji.



Rysunek 6.22: Przyciski poleceń ruchu

Poruszanie robotem może odbywać się w trybie ciągłym lub inkrementalnym. W trybie ciągłym robot będzie poruszał się z zadaną prędkością do momentu puszczenia przycisku. W trybie inkrementalnym po naciśnięciu przycisku robot przemieści się z wybraną prędkością o zadany inkrement. Inkrement można zmienić naciskając na przycisk wyświetlający jego wartość. Wartość inkrementu podawana jest w milimetrach dla zmiany pozycji kartezjańskiej oraz w stopniach dla orientacji narzędzia robota i pozycji podawanej we współrzędnych przegubowych. Szczegółowe informacje na temat rodzajów ruchów manipulatora zostały przedstawione w rozdziale 8.3.

i Ruch liniowy oraz wyrównanie orientacji odnoszą się jedynie do ruchów w przestrzeni kartezjańskiej. Podczas ruchu w przestrzeni przegubów przyciski te są wyszarzane.

7 Zarządzanie programem

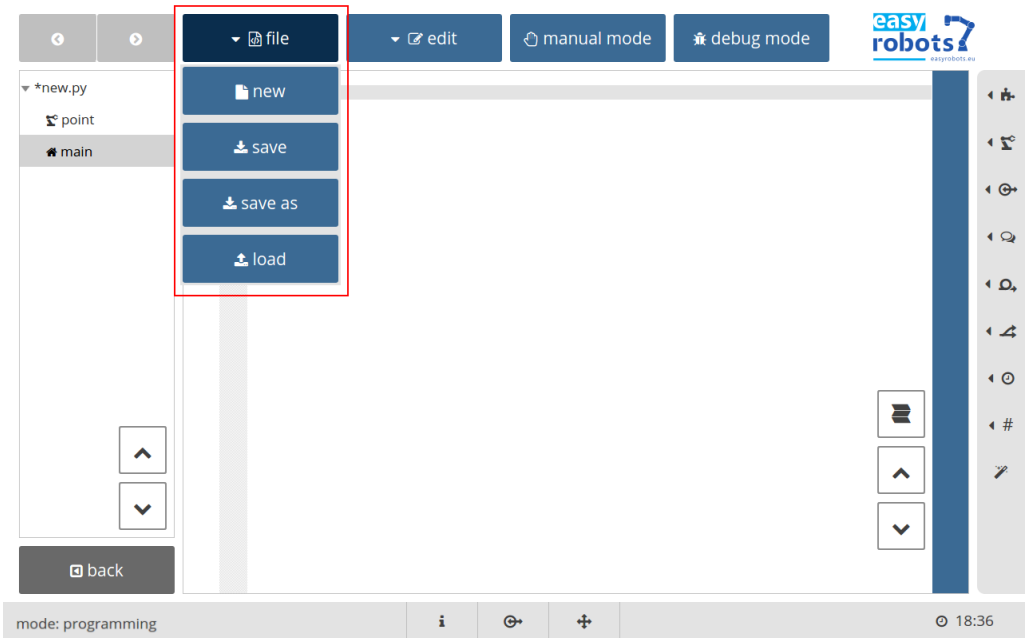
7.1 Tworzenie i usuwanie skryptów

Programy robota

Robot, jako urządzenie wielofunkcyjne, może posiadać wiele różnych programów (skryptów) podzielonych na pliki. Ich ilość nie jest ograniczona przez producenta urządzenia i jest zależna tylko od pojemności jego pamięci.

Zapis i odczyt

Po przejściu do edytora skryptu robota tworzony jest nowy plik skryptu, dzięki czemu można od razu przejść do programowania. W celu odtworzenia programu ze skryptu, należy go zapisać. Służy do tego menu file w górnym pasku edytora. W celu odczytania wcześniej stworzonego i zapisanego programu, również należy posłużyć się menu file edytora.

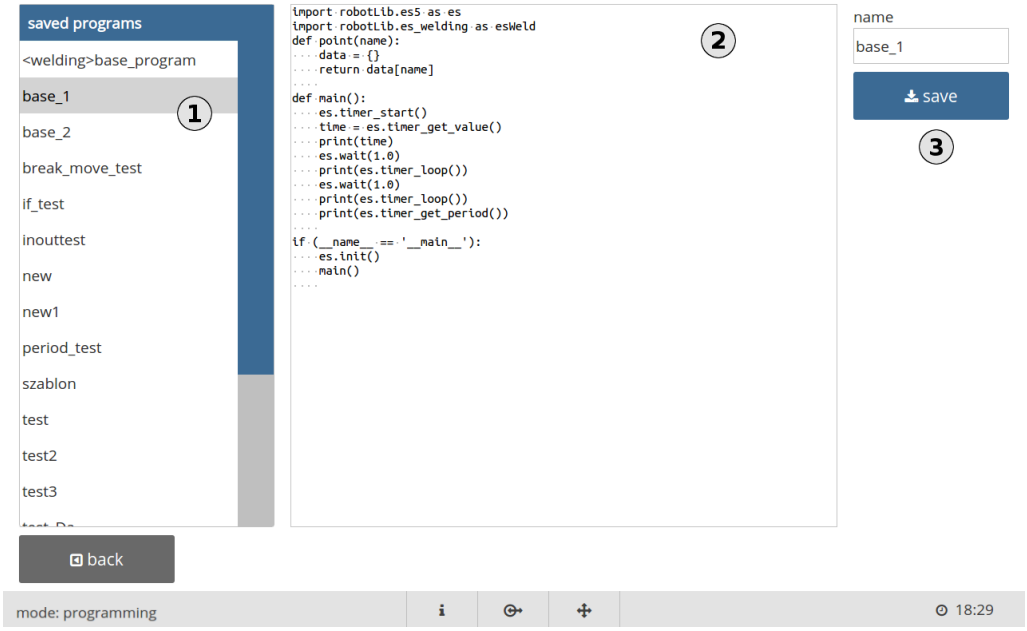


Rysunek 7.23: Zapis i odczyt plików programu

Okno zarządzania

Zarządzanie plikami programów odbywa się za pośrednictwem okna zarządzania. Umożliwia ono:

1. wybór z listy zapisanych programów,
2. podgląd zaznaczonego programu,
3. wczytanie, usunięcie bądź zapisanie programu pod wybraną nazwą.

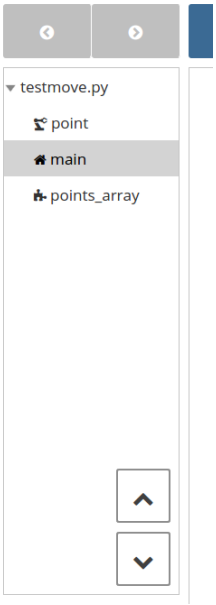


Rysunek 7.24: Okno zarządzania programami

7.2 Podział widoku skryptu

Podstawy

Widok programu robota został podzielony na funkcje w nim zawarte. Każda funkcja jest reprezentowana przez element w tabeli. Po włączeniu podglądu danej funkcji poprzez kliknięcie jej nazwy, jej zawartość jest wyświetlana w edytorze. Przyciski ze strzałkami ułatwiają poruszanie się pomiędzy funkcjami.



Rysunek 7.25: Tabela zawierająca funkcje skryptu

Funkcje podstawowe

Po utworzeniu nowego programu, tabela zawiera funkcje: "main" oraz "point". Pierwsza z nich jest funkcją główną w programie i powinna być uzupełniona przez

operatora elementami programu robota. Druga z nich jest przedstawiona w edytorze w formie tabeli punktów orientacyjnych robota, które podczas tworzenia programu są tworzone z przypisanymi im nazwami.

Konfiguracja

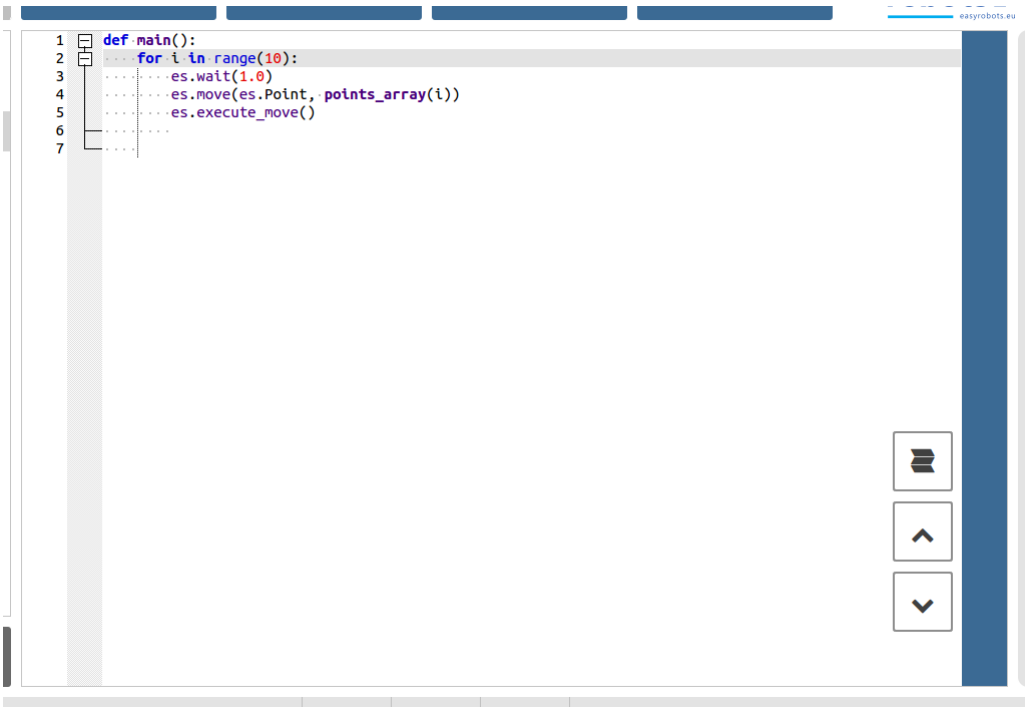
Zanim wykonany zostanie program zapisany w funkcji "main", wykonywane są operacje ustalające warunki wykonywania programu. Należą do nich: ustawianie domyślnej geometrii narzędzia czy układu odniesienia, ustalanie globalnego współczynnika prędkości i przyspieszenia, itd. Okno konfiguracji pozwala wpływać na proces ustalania warunków początkowych pracy programu. Aby przejść do okna konfiguracji należy aktywować nazwę programu w tabeli zawierającej funkcje programu.

Rysunek 7.26: Konfiguracja warunków początkowych działania programu robota

7.3 Dodawanie, edycja i usuwanie komend

Edytor

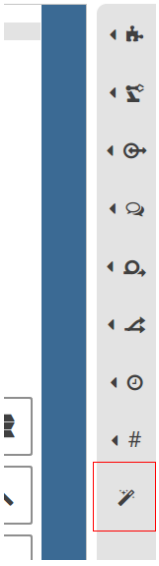
Okno edycji programu wyświetla skrypt w postaci tekstowej, podzielonej na linie, zgodnie ze sposobem zapisu programu w języku Python. W trybie podstawowym każda linia programu traktowana jest jak osobny element programu, który można zmieniać, usuwać, czy dodawać kolejne. Aby dodać nowy element programu, należy zaznaczyć linię, przed którą zostanie wstawiony nowy element, oraz dokonać jego wstawienia. Przyciski ze strzałkami ułatwiają poruszanie się pomiędzy liniami programu, natomiast przycisk zwijania pozwala na ukrywanie bloków programu, aby zwiększyć jego czytelność.



Rysunek 7.27: Edytor programu robota

Formularze

Panel boczny edytora zawiera pogrupowane tematycznie formularze, pozwalające na szybkie dodawanie najczęściej używanych komend. Za pomocą wymienionego panelu można także uruchomić kreator tworzenia ścieżki, okno dodawania komend, czy kreator tworzenia gotowych funkcji.

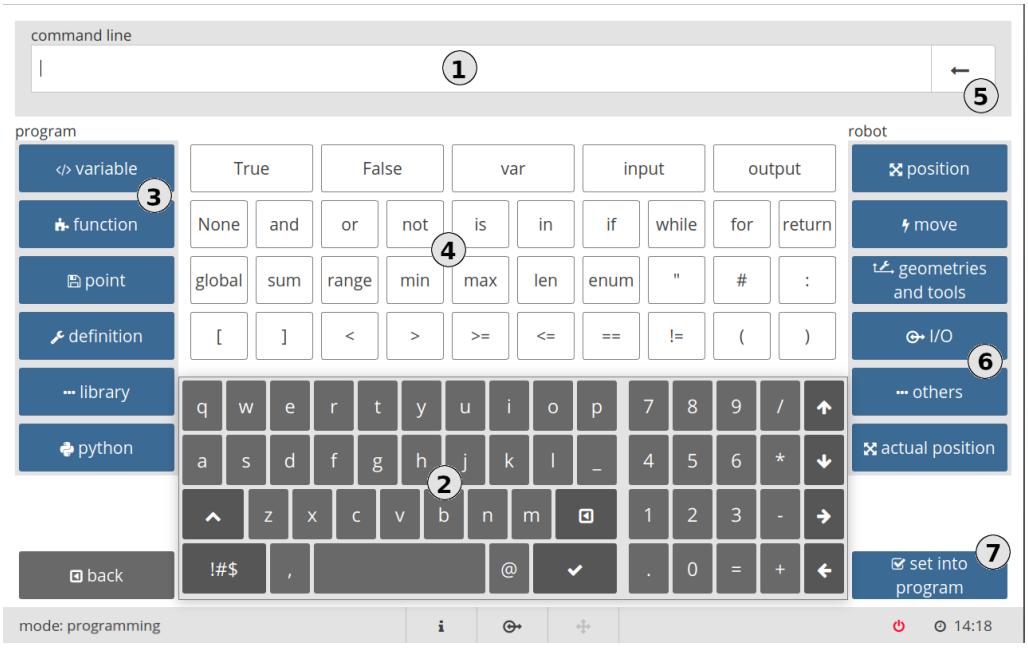


Rysunek 7.28: Panel z przyciskami dodawania komend (na czerwono: przycisk do okna tworzenia komend niestandardowych).

Okno tworzenia komend

W celu stworzenia komendy, dla której nie przewidziano specjalnego formularza, można wykorzystać okono tworzenia komend. Służy do tego przycisk znajdujący się na panelu bocznym edytora. Elementami wspomnianego okna są:

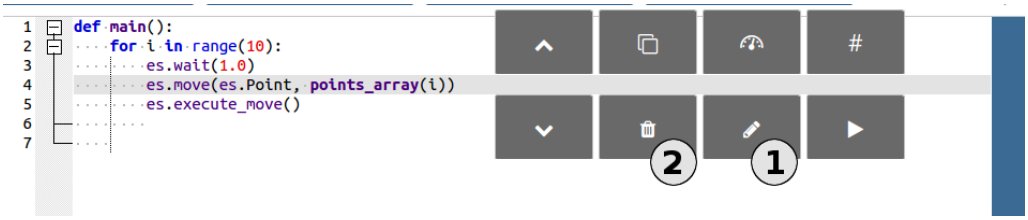
1. pole edycji linii
2. klawiatura
3. przyciski uruchamiające panele z zawartością programu i systemu gotowe do wstawienia do tworzonej komendy
4. panel z najczęściej wybieranymi elementami poleceń
5. przycisk usuwania ostatniego elementu z pola edycji
6. przyciski uruchamiające panele zawierające funkcje biblioteki robota
7. przycisk zatwierdzenia dodawania do programu



Rysunek 7.29: Okno edycji komend niestandardowych

Edycja

Aby edytować stworzoną linię skryptu należy wybrać ją w edytorze oraz wcisnąć przycisk edycji w wyświetlonym menu(1). Oprogramowanie spróbuje dopasować ją do jednego z formularzy, aby ułatwić jej edycję. Jeśli się to uda, zostanie wyświetlony odpowiedni formularz z wprowadzonymi danymi w celu ich edycji. Jeśli dopasowanie nie powiedzie się, zostanie wyświetlone standardowe okno tworzenia komend wraz z edytowaną linią. Po zatwierdzeniu, linia zostanie zamieniona.



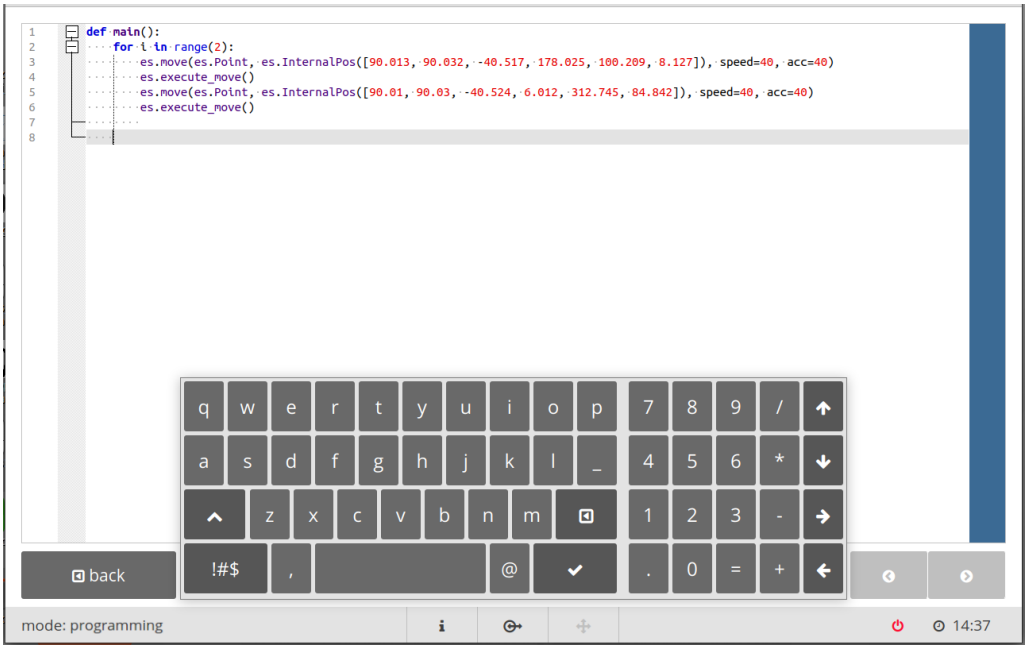
Rysunek 7.30: Pasek menu edytora

Usuwanie

Aby usunąć linię ze skryptu należy wybrać ją w edytorze oraz wcisnąć przycisk usuwanie w wyświetlonym menu (2). Linia zostanie natychmiast usunięta. Jeśli usuwana linia zawierała blok kodu (komendy: *if*, *for*, *while*, itd.), blok ten zostanie usunięty razem z nią.

Edycja ręczna

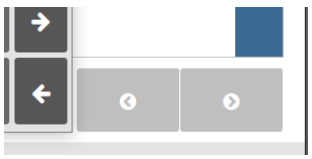
Naciśnięcie przycisku trybu ręcznego manual uruchamia prosty edytor tekstowy w celu edycji wybranej funkcji programu. Po aktywacji tego trybu, zostanie pokazany okno edytora, pojawi się kursor oraz klawiatura. W celu zakończenia edycji ręcznej należy wcisnąć przycisk "back". Napisany skrypt zostanie wtedy sprawdzony pod kątem zgodności składni i modyfikacje zostaną wprowadzone do pamięci.



Rysunek 7.31: Tryb edycji ręcznej

Cofanie operacji

Modyfikując skrypt edytor tworzona jest jego historia. W celu poruszania się po historii skryptu należy wykorzystać przyciski historii edycji.



Rysunek 7.32: Przyciski historii edycji

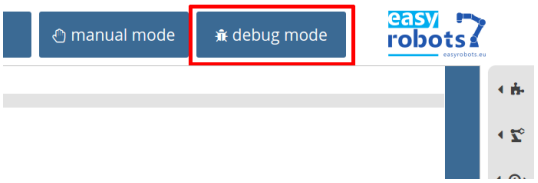
7.4 Odnajdywanie błędów w skryptach

Błędy składni

Próba dodania błędnie sformułowanej linii skryptu zakończy się wyświetleniem odpowiedniego komunikatu już na etapie dodawania. Dotyczy to również trybu manualnego. Z założenia zatem skrypty tworzone za pomocą edytora nie zawierają błędów składniowych.

Tryb debuggowania

Do przetestowania programu w poszukiwaniu błędów semantycznych oraz logicznych służy tryb edytora nazwany "trybem debuggowania". Przy jego pomocy zapisany program można uruchomić w kontrolowanych warunkach, testując jego przebieg oraz sprawdzając wartości zmiennych, a także pozostałych zasobów.



Rysunek 7.33: Przycisk trybu odnajdywania błędów znajdujący się w edytorze

Rysunek 7.34: Okno debuggowania programu

Widok programu

Widok programu w trybie debuggowania zawiera cały skrypt, taki jak jest przechowywany w pliku; bez podziału na poszczególne funkcje. Zielony znacznik programu w postaci strzałki wskazuje aktualnie wykonywaną linię.

Sterowanie przepływem

Kontrola nad programem jest sprawowana za pośrednictwem przycisków sterowania. Ich funkcje to:

1. wznowienie wykonywania programu w trybie ciągłym,
2. wstrzymanie wykonywania programu,
3. wykonanie pojedynczej linii w programie (jeśli linia jest odwołaniem do

funkcji, to wykonaj całą funkcję),

4. wykonanie pojedynczej linii w programie (jeśli linie jest odwołaniem do funkcji, to przejdź do tej funkcji),
5. zatrzymaj wykonywanie programu i zacznij od początku,

Rysunek 7.35: Przyciski kontroli przepływu programu

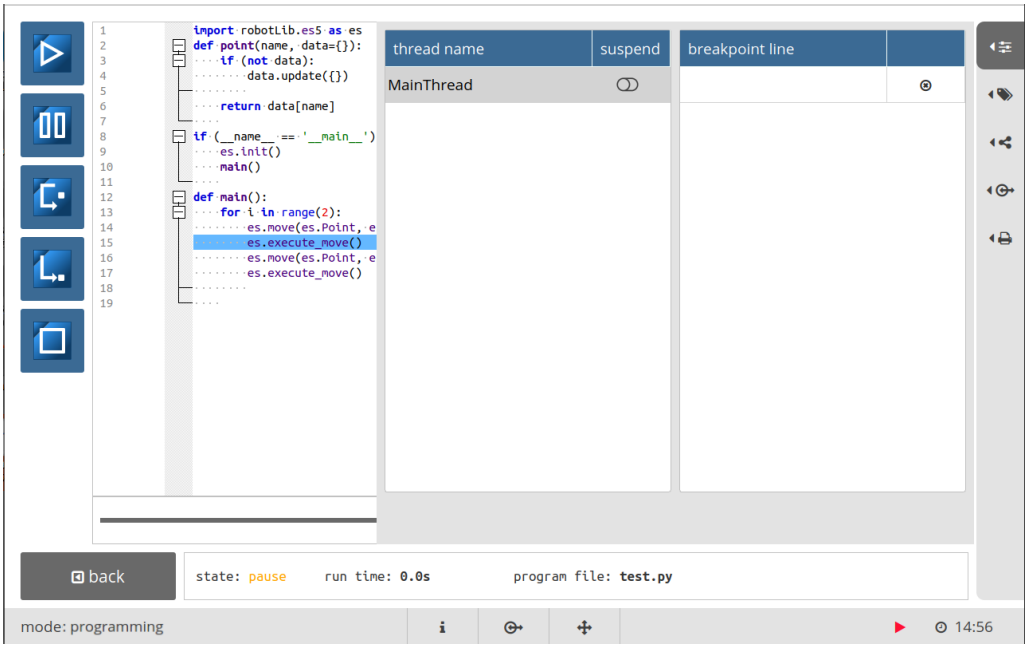
Sterowanie prędkością

Pasek sterowania prędkością umożliwia zmianę globalnej prędkości manipulatora. Zmiany można dokonać również podczas działania programu, lecz ustawiona prędkość zostanie zastosowana począwszy od następującego po zmianie rozkazu ruchu.

Punkty przerwania

Zatrzymanie wykonywania programu przed wykonaniem linii w skrypcie o podanym numerze jest realizowane przez mechanizm tzw. "breakpoint'ów". Dodanie oraz usunięcie takiego punktu w programie może zostać wykonane:

- przy pomocy tabeli w menu bocznym,
- poprzez wciśnięcie przycisku na marginesie widoku programu.



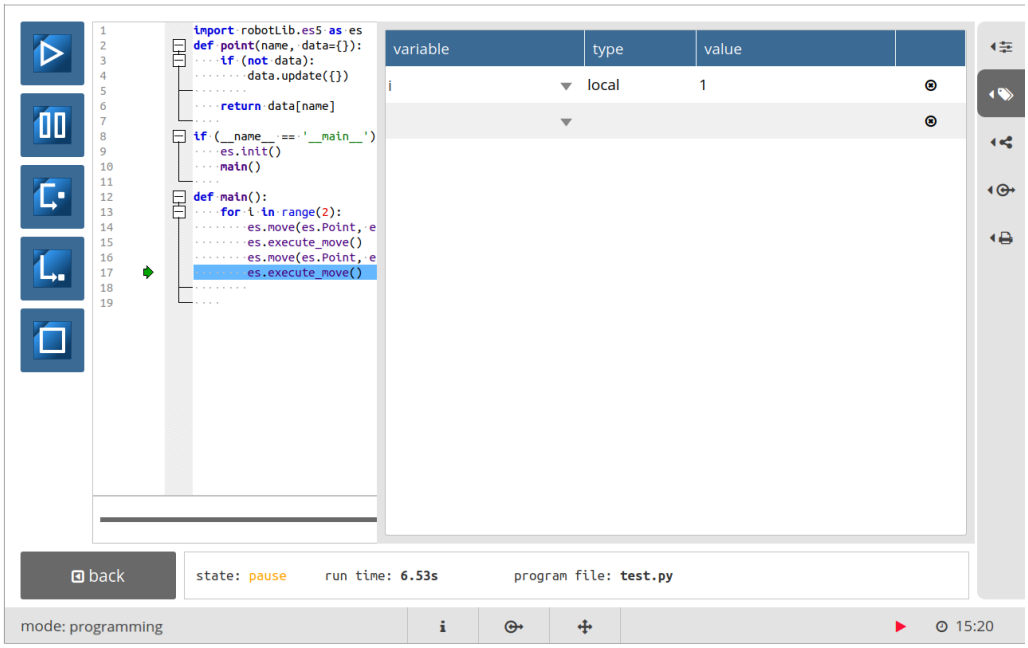
Rysunek 7.36: Dodawanie punktów zatrzymania programu

Sterowanie pracą wątków

Przy pomocy tabeli bocznej programu można sterować podglądem wykonywania programu w poszczególnych wątkach poprzez zaznaczenie odpowiedniej linii w tabeli w panelu bocznym. Oznaczenie wątku jako "suspended" zawiesza wkonywanie wątku do czasu jego odwieszenia.

Podgląd zmiennych

Podgląd zmiennych, globalnych oraz lokalnych, jest realizowany przy pomocy tabeli z menu bocznego. Wybranie nazwy zmiennej w tabeli powoduje cykliczny jej odczyt.



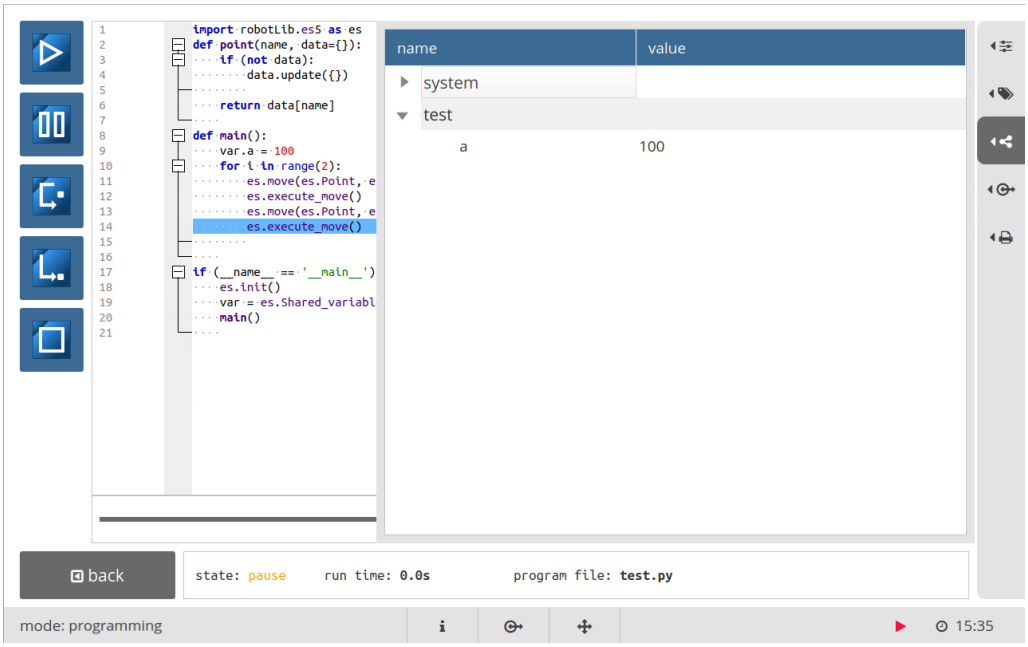
Rysunek 7.37: Podgląd wartości zmiennych

Jeśli typ zmiennej nie należy do zbioru typów prostych, takich jak: 'int', 'float', 'string', to wartość zmiennej odczytywanej w ten sposób będzie sygnalizowana

jako 'complex'. Jeśli badana zmienna nie jest widoczna w aktualnie przetwarzanym bloku programu, lub nie została jeszcze zainicjowana, jej wartość będzie wynosić 'fault'.

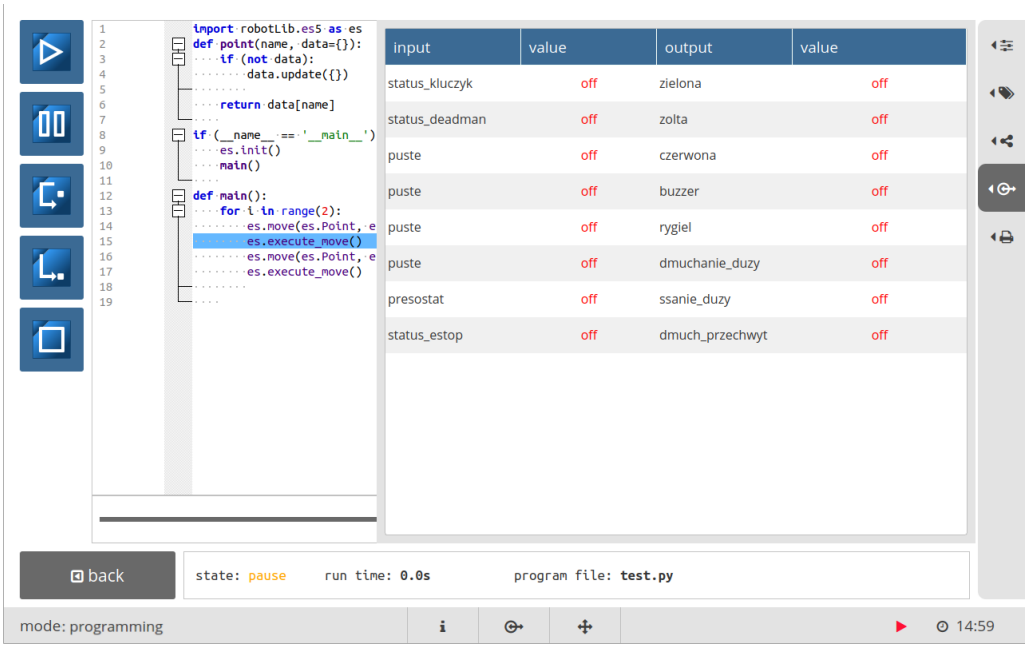
Podgląd zmiennych współdzielonych

Panel w menu bocznym umożliwia podgląd wartości zmiennych które docierają do wizualizacji panelu oraz są zapisywane do pliku. Zmienne są pogrupowane względem zadeklarowanej przestrzeni nazw. Najczęściej przestrzenią nazw zmiennnych w programie jest nazwa tego pliku programu.



Rysunek 7.38: Podgląd wartości zmiennych współdzielonych

Podgląd I/O



Rysunek 7.39: Podgląd statusu modułu I/O

Ważnym elementem testowania działania programu jest podgląd stanów modułu I/O. W trybie debuggowania jest to również możliwe za pomocą panelu z menu bocznego.

Podgląd printoutów programu

Użycie funkcji "es.log" w skrypcie programu robota powoduje przesłanie zawartej w niej wiadomości do konsoli, której podgląd znajduje się m.in. panelu bocznym okna debuggowania skryptu robots.

Rysunek 7.40: Podgląd statusu modułu I/O

7.5 Uruchamianie gotowych skryptów

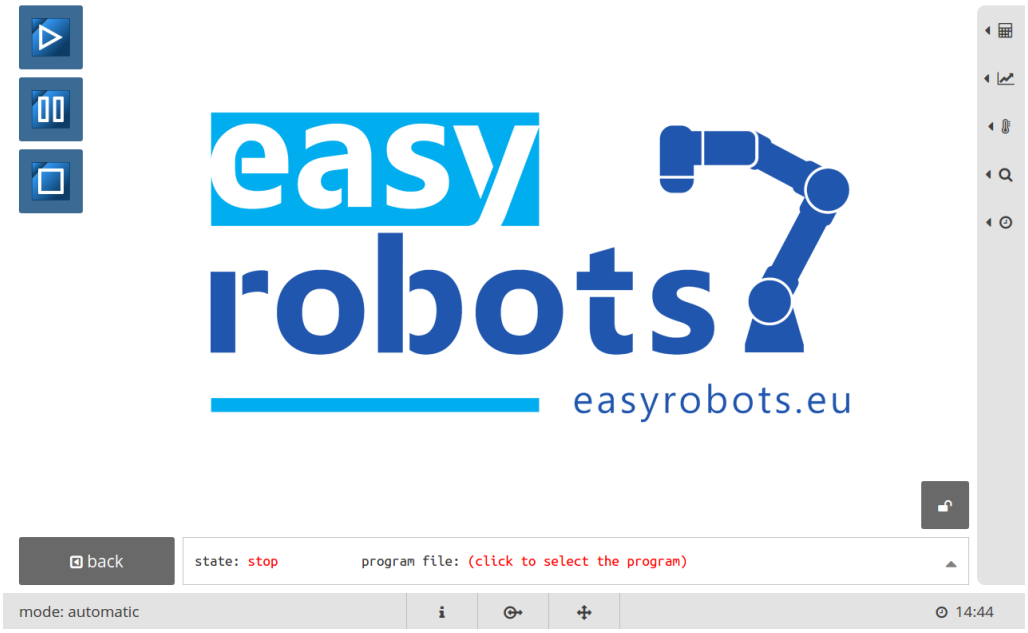
Okno uruchamiania programu

Uruchomienie wcześniej utworzonego skryptu robota wymaga przejścia do okna uruchamiania programów. W tym celu należy wcisnąć przycisk run program w oknie startowym.

> Wejście do okna uruchamiania programu możliwe jest tylko, gdy robot jest włączony. W przeciwnym wypadku wyświetlona zostanie informacja o tym, że robot jest wyłączony.

Sterowanie

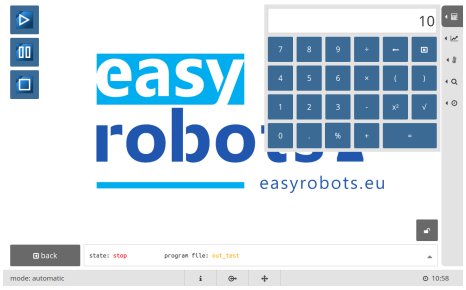
W celu uruchomienia zapisanego w pamięci programu robota należy wybrać go z rozwijanej listy oraz wcisnąć przycisk uruchamiania. Przycisk służy do zatrzymywania wykonywania programu, bez możliwości jego kontynuowania od miejsca zatrzymania. Wstrzymanie programu, po którym możliwe jest jego wznowienie, realizowane jest za pomocą przycisku. Kontynuacja programu jest możliwa po naciśnięciu przycisku uruchamiania programu.



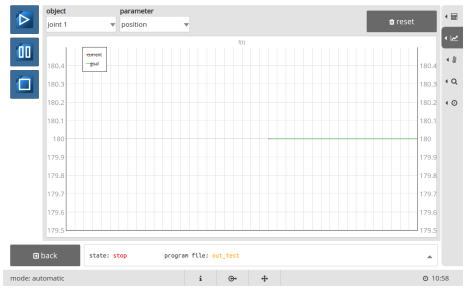
Rysunek 7.41: Okno uruchamiania programu.

Moduły dodatkowe

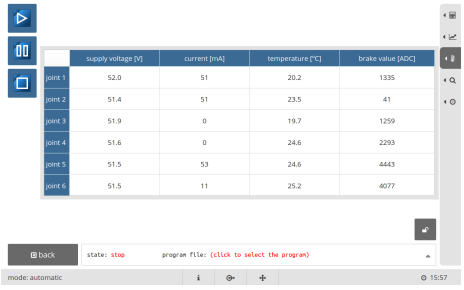
Po prawej stronie okna uruchamiania znajdują się przyciski służące do rozwijania aplikacji pomocniczych. Są to: kalkulator, wykresy parametrów dynamicznych robota, tabela pomiarowa, podgląd programu, stoper. Kalkulator jest prostym oknem umożliwiającym wykonanie podstawowych działań matematycznych - rysunek 7.42a. Przy pomocy wykresów możemy śledzić zmiany pozycji, prądu, oraz napięcia na każdym z przegubów robota- rysunek 7.42b. Okno z pomiarami udostępnia aktualne wartości prądu, napięcia, temperatury oraz czujnika odległości hamulca dla każdego z przegubów - rysunek 7.42c.



(a) Okno kalkulatora.



(b) Okno wykresów.



(c) Okno pomiarów.

Rysunek 7.42: Okna dodatkowych aplikacji.

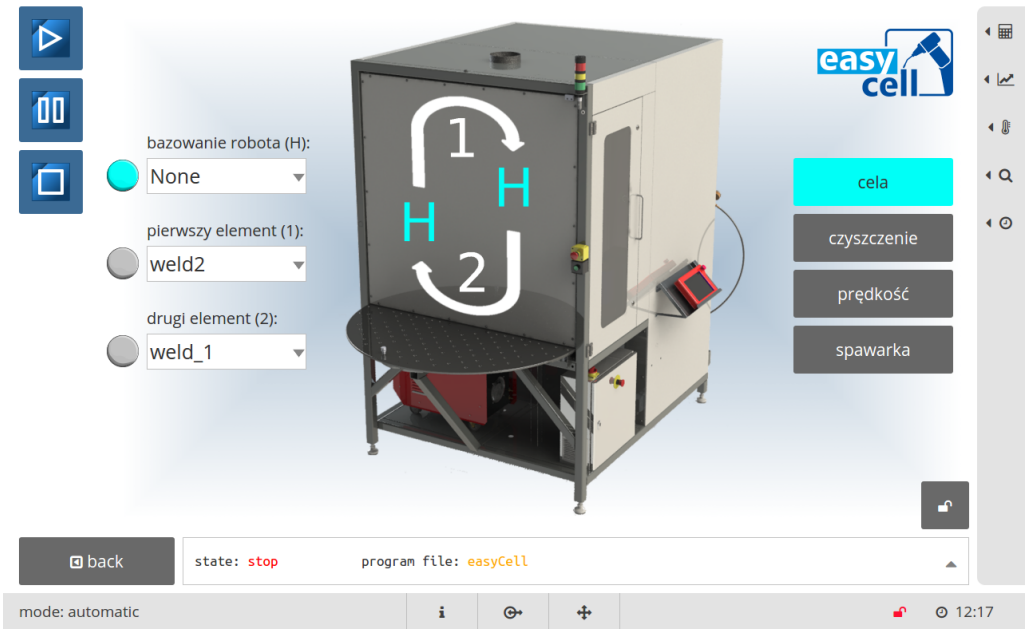
Okno podglądu programu wyświetla kod aktualnie wybranego programu - rysunek 7.43a. Okno ze stoperem pozwala na wykonywanie pomiarów czasu - rysunek 7.43b.

(a) Okno podglądu kodu programu. (b) Okno stopera.

Rysunek 7.43: Okna dodatkowych aplikacji, c.d.

Wizualizacja

Każdy program może posiadać własną wizualizację. Tworzenie wizualizacji omówione jest w instrukcji przeznaczonej dla modułów aplikacji EScontrol. Widżety wizualizacji wyświetlane są po wyborze programu na środku ekranu - rysunek 7.44. Umożliwiają one graficzne odwzorowanie danego procesu oraz wprowadzanie danych do programu jak i wyświetlanie ich za pomocą odpowiednich widżetów. Wyświetlane za pomocą widżetów wartości są przypisane do tzw. zmiennych współdzielonych.



Rysunek 7.44: Wizualizacja dla celi spawalniczej.

Tworzenie wizualizacji

Tworzenie wizualizacji dostosowanej do potrzeb konkretnego stanowiska pracy odbywa się za pomocą modułu dodatkowego udostępnionego w każdym robocie bezpłatnie. Sposób tworzenia wizualizacji oraz zasady ich funkcjonowania zostały omówione w dokumentacji o nazwie "Instrukcja obsługi modułu do tworzenia wizualizacji pracy robota". Zachęcamy do zapoznania się z nią.

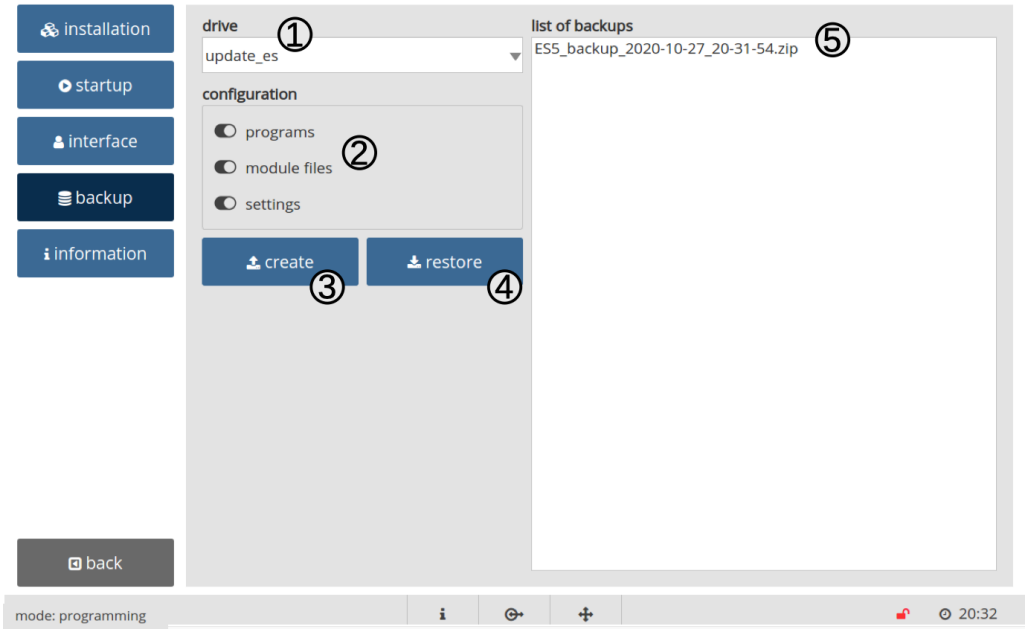
7.6 Wykonywanie i przywracanie kopii zapasowej

Funkcjonalność

Oprogramowanie EScontrol pozwala na wykonywanie kopii zapasowych:

- programów,
- plików modułów,
- ustawień.

Aby przejść do ekranu przywracania/tworzenia kopii zapasowych (rys 7.45) należy w głównym oknie wybrać przycisk settings , a następnie zakładkę backup .



Rysunek 7.45: Ekran przywracania i tworzenia kopii zapasowych

Opis

Okno kopii zapasowej składa się z następujących elementów:

1. lista rozwijana **drive** - wybór nośnika USB, na którym będzie tworzona kopia zapasowa (lub z którego kopia zostanie przywrócona),
2. panel konfiguracji - definicja rodzaju danych, które mają zostać zaimportowane / wyeksportowane,
3. przycisk create - tworzenie pliku kopii zapasowej,
4. przycisk restore - przywracanie danych z wykorzystaniem kopii zapasowej wskazanej w polu **list of backups** ,
5. lista plików kopii zapasowych.

Pliki kopii zapasowych, przechowywane są w postaci plików archiwów .zip, które można przywrócić. Nazwa każdego pliku zawiera datę oraz godzinę wykonania kopii.

8 Programowanie

8.1 Język programowania

Język skryptowy

Podstawowym językiem programowania robota jest język Python w wersji 3.7. Wszelkie zasady i zależności obowiązujące w tym języku mają zastosowanie w skryptach przeznaczonych dla robota. Podczas tworzenia skryptu sugeruje się jednak używać podstawowego zestawu słów kluczowych, których opis znajduje się w tym rozdziale.

Stosowanie w skrypcie robota niektórych, zaawansowanych funkcjonalności języka Python, takich jak konstrukcja "try..except" może wywołać niepożądane efekty działania modułu nadzorującego przepływ programu robota.

Blok kodu

Skrypt składa się z tzw. bloków. Przez "blok kodu" rozumie się funkcje, instrukcje warunkowe, pętle itd. Bloki kodu w języku Python nie posiadają nazwanych początków i końców oraz żadnych nawiasów służących do zaznaczania, gdzie dany blok się zaczyna, a gdzie kończy. Jedynym separatorem jest dwukropek (:) i wcięcia kodu.

Wcięcia

Bloki kodu definiowane są poprzez wcięcia. Wstawiając wcięcie zaczynamy blok, a kończymy go zmniejszając wielkość wcięcia do poprzedniej wartości. Nie ma żadnych nawiasów, klamer czy słów kluczowych. Oznacza to, że białe znaki (spacje itp.) mają znaczenie i ich stosowanie musi być konsekwentne.

Listing 8.1: Przykładowy skrypt

```
1 def funkcja():
2     zmienna=3
3     for i in range(zmienna):
4         pass
```

Tworzenie skryptu

Minimalna wersja skryptu robota zapewnia dostęp do wszystkich funkcjonalności biblioteki robota oraz porządkuje tworzony kod. Zadaniem programisty jest stworzenie programu robota w ciele funkcji "main", która jest domyślnie wyświetlana w edytorze.

Listing 8.2: Główna funkcja programu

```
1 def main():
2     pass
```

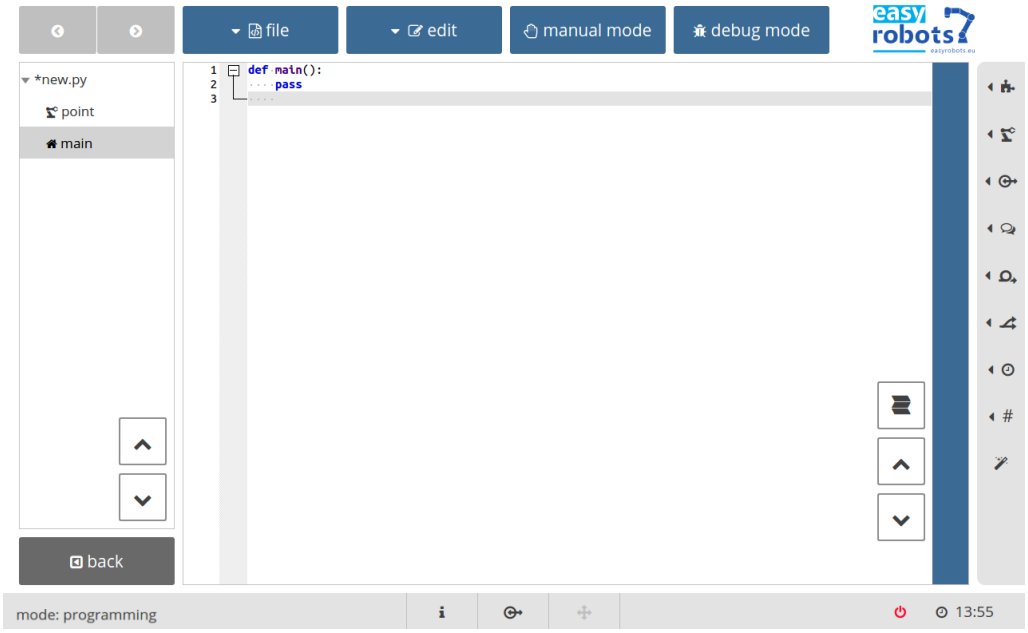
Skrypt początkowy

Pomimo, że w edytorze wyświetlana jest jedynie funkcja "main", skrypt początkowy zawiera więcej elementów. Szablon ten przedstawiono na poniższym listingu.

Listing 8.3: Szablon podstawowy programu

```
1 import robotLib.es5 as es
2 def point(name):
3     data = {}
4     return data[name]
5
6 def main():
```

```
7       pass
8
9   if (__name__ == '__main__'):
10      es.init('new')
11      main()
```



Rysunek 8.46: Widok głównej funkcji programu robota

8.2 Podstawy programowania ruchu

Zadawanie punktów

Podawanie punktów dojazdu manipulatora w przestrzeni odbywa się za pośrednictwem przeznaczonych do tego obiektów. Zmienne te są tworzone poprzez użycie parametrów liczbowych punktu jako danych wejściowych dla klasy reprezentującej dany typ punktu. Istnieje możliwość stworzenia punktu będącego:

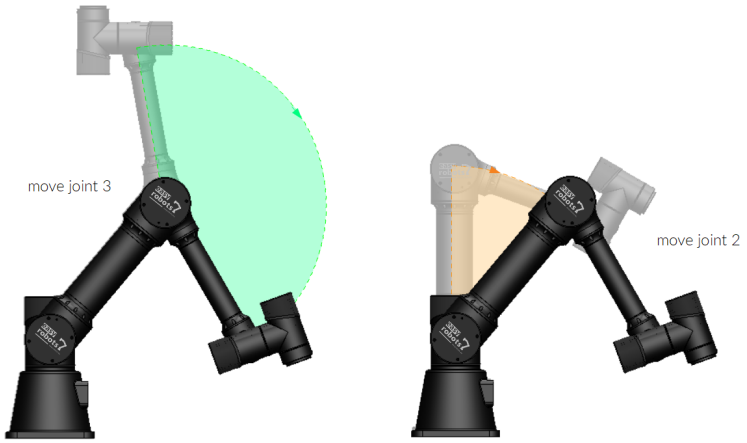
- zestawieniem pozycji kątowych poszczególnych przegubów (współrzędne przegubowe; wewnętrzne) wyrażonych w stopniach
- pozycją w metrach i orientacją w stopniach punktu TCP w przestrzeni względem określonego początku układu.

Współrzędne przegubowe

Reprezentacją punktu zapisanego w postaci współrzędnych przegubowych w skrypcie robota jest obiekt klasy InternalPos. Klasa ta jest częścią biblioteki robota. Definicja punktu odbywa się poprzez stworzenie obiektu klasy InternalPos wraz z podaniem parametrów liczbowych punktu. Stworzony punkt może być w późniejszym czasie edytowany lub może posłużyć do zadania pozycji dla manipulatora.

Listing 8.4: Przykład definiowania punktu we współrzędnych przegubowych

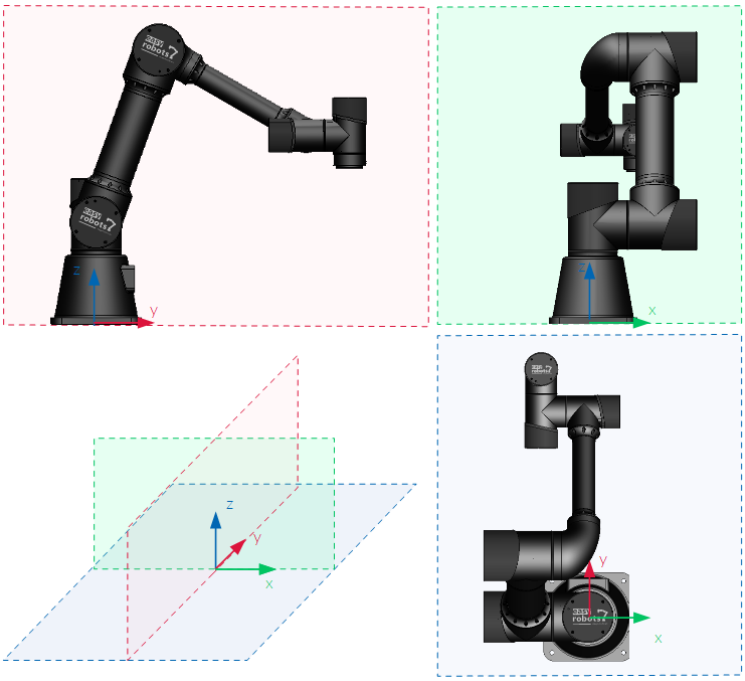
```
1 point = es.InternalPos([214.6, 108.5, 85.8, 75.6, 90.0, 34.6])
```



Rysunek 8.47: Ruch we współrzędnych przegubowych

Współrzędne kartezjańskie

Analogicznie do InternalPos, reprezentacją punktu zapisanego w postaci współrzędnych kartezjańskich jest obiekt klasy CartesianPos. W przypadku ruchu manipulatora do punktu zapisanego w takiej formie, należy myśleć o ruchu ostatniego punktu łańcucha kinematycznego, tzw. TCP względem określonego początku układu współrzędnych. Domyślnie jest to punkt znajdujący się w podstawie robota. Poszczególne pozycje przegubowe manipulatora są wtedy wyliczane na podstawie odwrotnego zadania kinemtyki.



Rysunek 8.48: Domyślny punkt odniesienia układu kartezjańskiego

Dodatkowe informacje

W związku z występowaniem kilku rozwiązań kinematyki, wymaga się, aby oprócz punktu zdefiniowanego we współrzędnych kartezjańskich, określone zostało przybliżone rozwiązanie kinematyki w postaci punktu zdefiniowanego we współrzędnych przegubowych.

Listing 8.5: Przykład definiowania punktu we współrzędnych kartezjańskich

```
point = es.CartesianPos([0.4, 0.4, 0.2], [90, 0, 180],
    es.InternalPos([214.6, 108.5, 85.8, 75.6, 90.0, 34.6])
```

Zalety

Pomimo konieczności podawania danych punktu w postaci dwóch różnych reprezentacji, wykorzystanie współrzędnych kartezjańskich w programowaniu ruchu manipulatora niesie za sobą szereg korzyści:

- zwiększona czytelność prezentowanych danych,
- możliwość ruchu wzdłuż osi układu współrzędnych bez konieczności definiowania każdego punktu z osobna (ręczna edycja współrzędnych),
- możliwość przesunięcia układu odniesienia przy pomocy zdefiniowanych układów odniesienia,
- możliwość modyfikacji parametrów narzędzia bez konieczności redefiniowania punktów w programie,
- i wiele innych..

8.3 Instrukcje ruchu manipulatora

Definiowanie ruchu

W celu zdefiniowania ruchu do punktu niezbędne są parametry celu, czyli pozycja, ale również parametry stanu przejściowego, takie jak rodzaj ruchu, prędkość przejazdu oraz przyspieszenia. Aby zawrzeć te informacje w skrypcie, należy wywołać funkcję z biblioteki o nazwie "move" wraz z parametrami wejściowymi dotyczącymi rodzaju ruchu, parametrami celu, przyspieszeniem oraz prędkością przejazdu.

Ruch do punktu

Podstawowym rodzajem ruchu manipulatora jest ruch do punktu. Charakteryzuje się on brakiem synchronizacji pomiędzy poszczególnymi przegubami manipulatora. Czas trwania ruchu każdego przegubu jest dostosowany do czasu ruchu przegubu, którego ruch osobno wykonany zająłby najwięcej czasu. Każdy staw dąży do zadanej pozycji niezależnie od ruchu innych stawów, najrótszą drogą.

Listing 8.6: Przykład definiowania ruchu do punktu

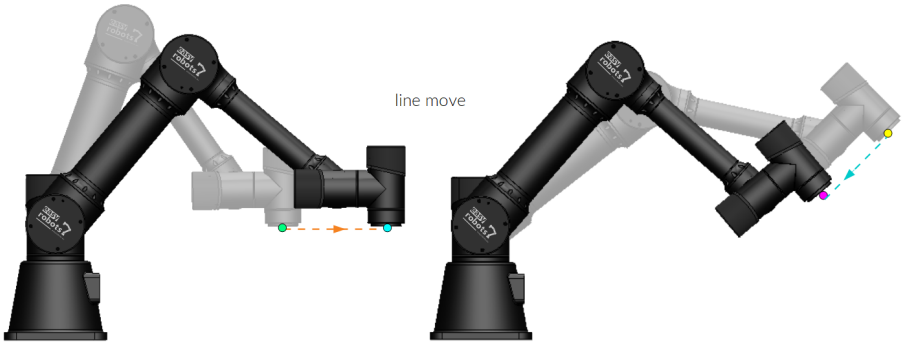
```
es.move(type=es.Point, end=point, speed=50, acc=50)
```

Ruch po linii

Ruch, w którym aktualny punkt TCP (tool center point - ang. punkt centralny narzędzia) przemieszcza się po najkrótszej ścieżce pomiędzy punktem startowym a celem, nazywa się ruchem po linii. Charakteryzuje się on synchronizacją położenia wszystkich stawów manipulatora podczas ruchu w celu zapewnienia odpowiedniego położenia punktu TCP. Dlatego też poszczególne stawy mogą podczas jednego ruchu wykonywać szereg przejazdów w różnym kierunku. Ruch ten znajduje zastosowanie w przypadkach, gdy wymagane jest określone zachowanie narzędzia podczas ruchu, np. podjazd do elementu, obróbka elementu, spawanie.

Listing 8.7: Przykład definiowania ruchu po linii

```
es.move(type=es.Line, end=point, speed=50, acc=50)
```



Rysunek 8.49: Ruch po linii

Wykonywanie warunkowe ruchu

Istnieje możliwość uzależnienia wykonywanego ruchu po linii od stanu wejścia I/O robota. W tym celu należy podać parametry warunkujące wykonywanie ruchu. W momencie gdy warunek przestaje być spełniony, ruch zostaje przerwany.

Listing 8.8: Przykład definiowania warunkowego ruchu po linii

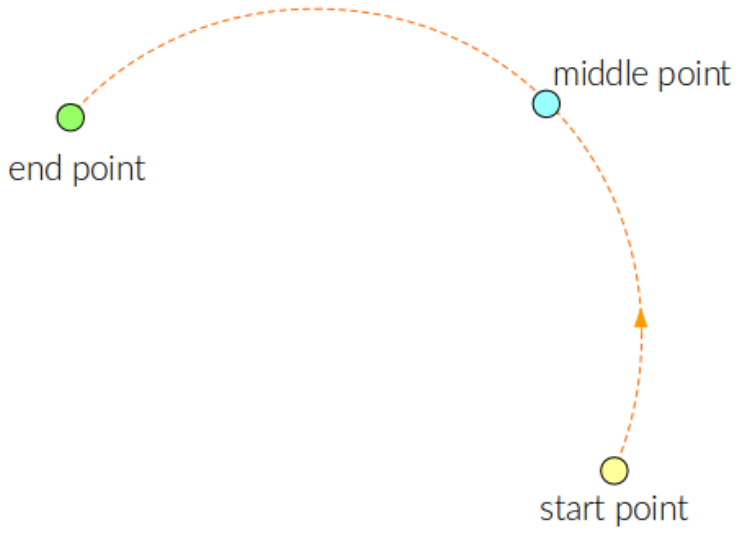
```
1 es.move(type=es.Line, end=point, speed=50, acc=50, until_input=1,
2     until_state=True )
```

Ruch po łuku

Ruch po łuku definiuje trajektorię, w której punkt TCP zakreśla w przestrzeni fragment okręgu. Zachowanie napędów jest analogiczne do ruchu po linii. W celu zdefiniowania tego ruchu potrzebne są dwa punkty orientacyjne. Pierwszy z nich jest punktem docelowym, natomiast drugi - punktem pośrednim, należącym do łuku.

Listing 8.9: Przykład definiowania ruchu po łuku

```
1 es.move(type=es.Arc, end=point1, middle=point2, speed=50, acc=50)
```



Rysunek 8.50: Ruch po łuku

Wykonywanie polecenia ruchu

Aby zdefiniowany uprzednio ruch został wykonany przez robota, w skrypcie musi znaleźć się polecenie "execute_move". Robot podejmuje działanie dopiero w momencie, gdy zostanie ono wykonane. Gdy funkcja ta zostanie wywołana, bez parametrów, oznacza to, że podane trajektorie zostaną wykonane w domyślnym układzie odniesienia oraz z domyślnym narzędziem. Jeśli istnieje potrzeba zmiany jednego z tych parametrów, należy uwzględnić te zmiany podczas definiowania funkcji execute.

Listing 8.10: Polecenie wykonywania podanych ruchów

```
1 es.execute_move()
```

Uwzględnienie układu odniesienia

Dany ruch/sekwencja ruchów może zostać wykonana w innym, niż domyślny, układzie odniesienia. W tym celu w momencie definiowania funkcji "execute_move" należy uwzględnić parametr o nazwie "coord_sys". Jako wartość należy podać nazwę układu odniesienia, z uwzględnieniem którego zostanie wykonany ruch.

Listing 8.11: Przykład wykonywania ruchu w podanym układzie odniesienia

```
1 es.execute_move(coord_sys='nazwa_ukladu')
```

Uwzględnienie geometrii narzędzia

W celu zmiany geometrii narzędzia na czas wykonywania pojedynczego ruchu/sekwencji ruchów, należy podczas definiowania funkcji "execute_move", uwzględnić parametr o nazwie "tool_geom". Jako wartość należy podać nazwę geometrii narzędzia z uwzględnieniem którego zostanie wykonany ruch.

Listing 8.12: Przykład wykonywania ruchu z podaną geometrią narzędzia

```
1 es.set_tool_geometry(name='nazwa_geometrii')
```

Ustawienia globalne

Domyślny układ odniesienia oraz geometria narzędzia, w ramach skryptu, może zostać zmieniony za pomocą funkcji "set_default_coordinate_system" oraz "set_default_tool_geom z biblioteki robota.

Listing 8.13: Przykład zmiany domyślnego układu odniesienia i geometrii narzędzia

```
1 es.set_coordinate_system(name='nazwa_ukladu')
2 es.set_tool_geometry(name='nazwa_geometrii')
```

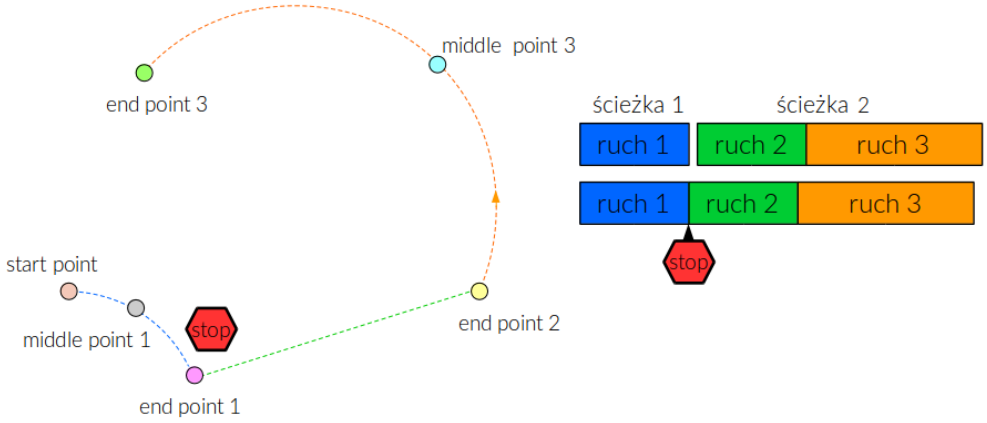
Zmieniony domyślny układ współrzędnych obowiązuje w całym skrypcie. Dotyczy to także importowanych, stworzonych wcześniej, skryptów.

Łączenie ruchów

Zdefiniowane ruchy manipulatora mogą być łączone w ścieżkę. Wystarczy stworzyć kilka poleceń "move", a dopiero po nich wstawić polecenie "execute_move". Dzięki temu robot spróbuje stworzyć płynne połączenia pomiędzy ruchami; nie będzie zatrzymywał się w poszczególnych punktach docelowych.

Listing 8.14: Przykład łączenia kilku ruchów w ścieżkę

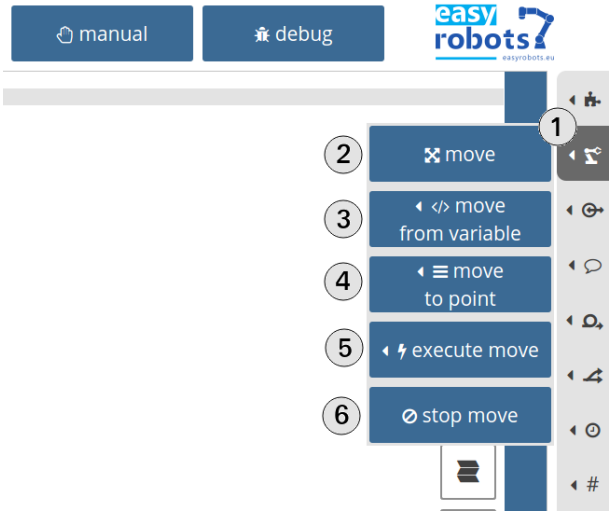
```
1 es.move(type=es.Point, end=point1, speed=30, acc=50)
2 es.move(type=es.Point, end=point2, speed=50, acc=50)
3 es.move(type=es.Point, end=point3, speed=40, acc=50)
4 es.execute_move()
```



Rysunek 8.51: Składanie ruchów w ścieżkę

Dodawanie komend ruchu

Ręczne definiowanie instrukcji ruchu jest możliwe, lecz nieintuicyjne i trudne. W celu dodania komend ruchu do skryptu przy pomocy kreatora, należy nacisnąć przycisk move z rozwijanego menu panelu bocznego.



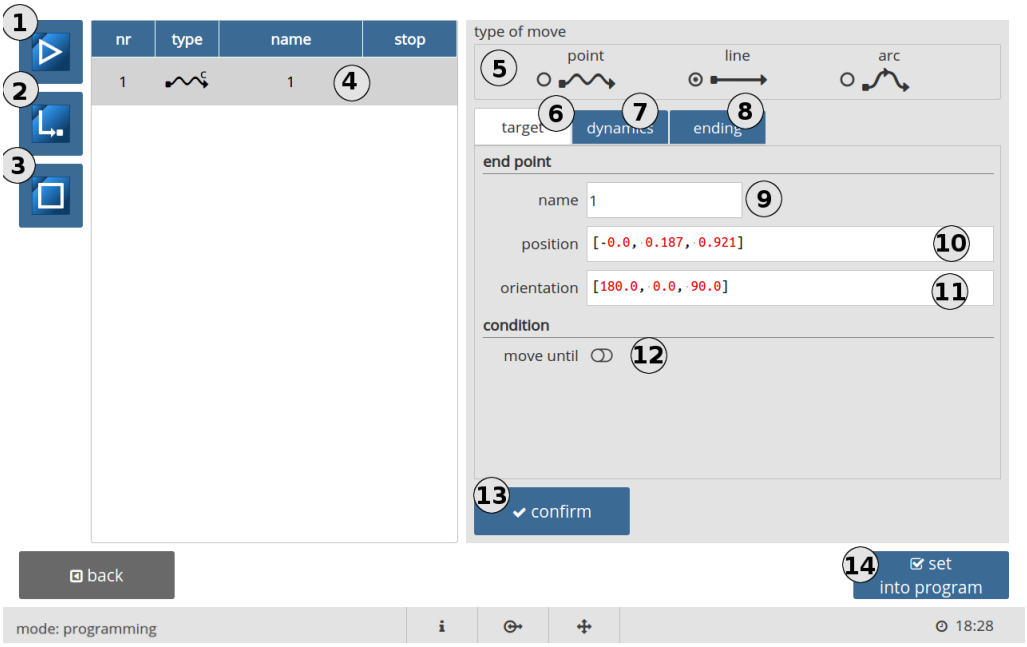
Rysunek 8.52: Dodawanie komend ruchu

1. Aktywacja menu ruchu.
2. Dodawanie ruchu za pomocą kreatora.
3. Dodawanie pojedynczego ruchu przy pomocy pozycji zapisanej w zmiennej.
4. Dodawanie pojedynczego ruchu przy pomocy pozycji zapisanej w tablicy punktów.
5. Polecenie wykonania zaplanowanego ruchu.
6. Polecenie zatrzymania ruchu do wykorzystania w osobnym wątku (8.13).

Kreator ruchu

Kreator ruchu pozwala dodawać kolejne punkty oraz określać parametry statyczne i dynamiczne planowanego ruchu. Udostępnia opcje takie jak:

- Testowanie tworzonej ścieżki
- Wyświetlanie tworzonej ścieżki z możliwością jej modyfikacji
- Wyświetlanie parametrów, takich jak: rodzaj ruchu, punkt docelowy, dynamika (prędkość, przyspieszenie, czas ruchu) oraz geometrie referencyjne



Rysunek 8.53: Kreator dodawania ścieżki ruchu

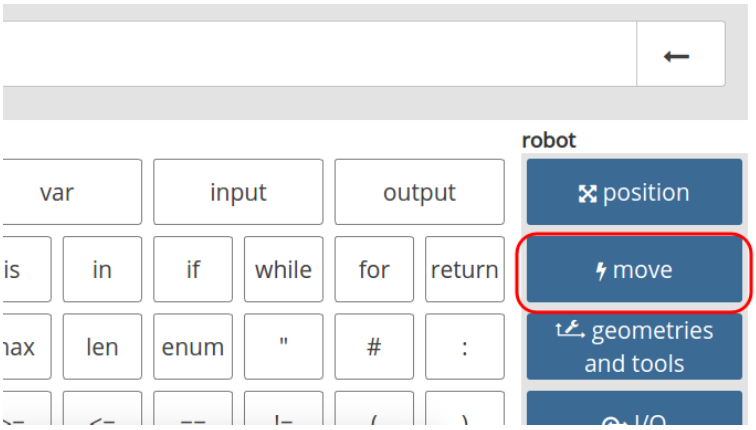
1. Przycisk uruchomienia/wznowienia testu ścieżki.
2. Uruchomienie pojedynczego ruchu w ścieżce.
3. Natychmiastowe przerwanie ruchu.
4. Lista punktów pośrednich ścieżki ruchu.
5. Typ ruchu do punktu pośredniego.
6. Karta punktu docelowego punktu ruchu oraz ewentualnych punktów pośrednich.
7. Karta dynamiki ruchu. Zawiera takie parametry jak prędkość i przyspieszenie.
8. Karta kończenia ruchu. Zawiera informacje o tym, czy ruch zakończy się zatrzymaniem robota.
9. Nazwa punktu docelowego. Pod tą nazwą punkt docelowy będzie przechowywany w bazie punktów (funkcja *point*).
10. Współrzędne punktu docelowego.
11. Orientacja TCP w punkcie docelowym.
12. Warunek ruchu (tylko w ruchu liniowym).
13. Zatwierdzenie zmian w kartach.

14. Zatwierdzenie ścieżki i dodanie jej do skryptu.

8.4 Ruch względem zdefiniowanych punktów. Shift i approach

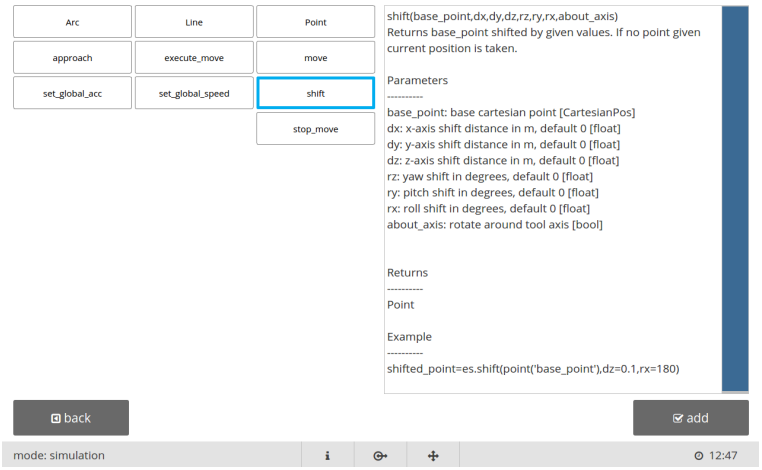
Polcecenie shift

Polecenie shift pozwala na utworzenie nowego punktu na podstawie jednego z wcześniej zdefiniowanych i zapisanie go w postaci zmiennej lub w bazie punktów. W celu wykorzystania polecenia shift należy wejść do okna tworzenia komend niestandardowych 7.28 a następnie przejść do poleceń z klasy move (8.54).



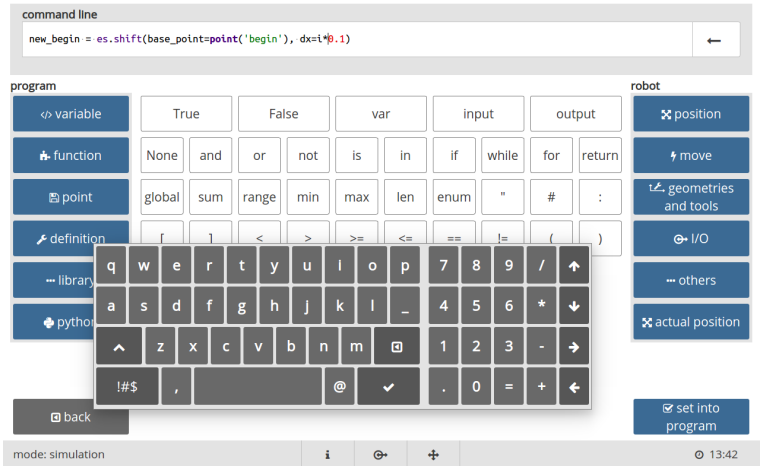
Rysunek 8.54: Dostęp do polecenia shift

Polecenie shift należy wybrać z listy dostępnych polceń (8.55). W tym oknie można również zapoznać się z opisem polecenia shift oraz przejrzeć listę dostępnych parametrów.



Rysunek 8.55: Sposób użycia polecenia shift

. Po dodaniu polcecenia shift do linii komend (8.56) należy wypełnić listę parametrów. Parametr $base_point$ określa punkt odniesienia. Za pomocą parametrów dx, dy, dz określa się przesunięcie w poszczególnych osiach a rx, ry i rz określają rotację punktu TCP względem orientacji w punkcie bazowym. Parametr $about_axis$ pozwala przesuwać i rotować względem osi związanych z narzędziem.



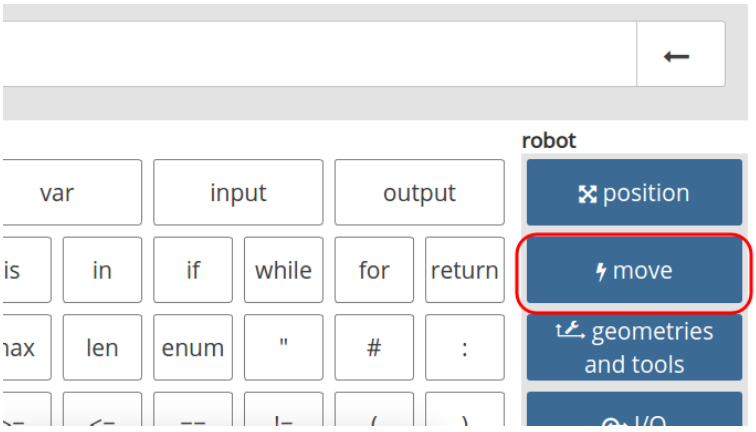
Rysunek 8.56: Polecenie shift po dodaniu do linii komend

Listing 8.15: Przykład użycia polecenia shift w kodzie programu

```
for i in range(10):
    es.move(type=es.Point, end=point('base'))
    es.execute_move()
    new_begin = es.shift(base_point=point('begin'), dx=i * 0.1)
    es.move(type=es.Point, end=new_begin, speed=20.0, acc=20.0)
    es.execute_move()
```

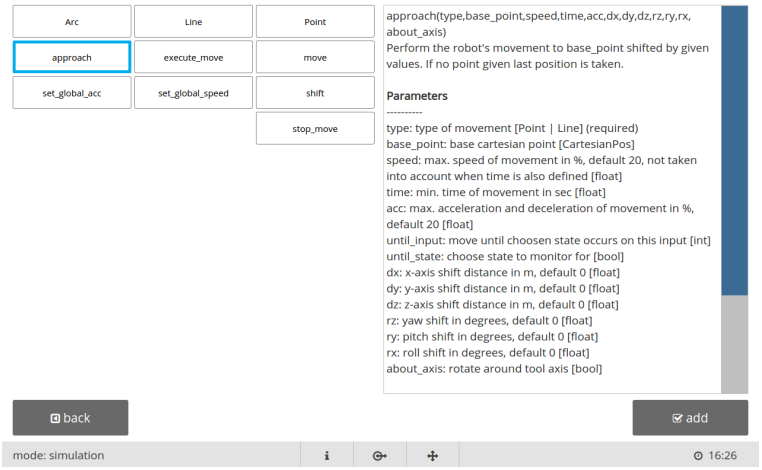
Polcecenie approach

Polecenie approach umożliwia poruszanie robotem o określoną wartość w każdej z osi kartezjańskiego układu współrzędnych względem wybranego, zdefiniowanego uprzednio punktu lub względem aktualnej pozycji punktu TCP robota. W celu wykorzystania polecenia approach należy wejść do okna tworzenia komend niestandardowych 7.28 a następnie przejść do poleceń z klasy move (8.57).



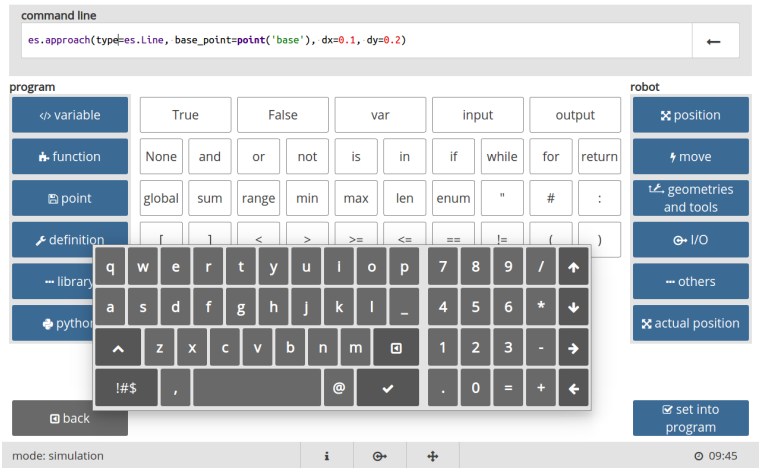
Rysunek 8.57: Dostęp do polecenia approach

Polecenie approach należy wybrać z listy dostępnych polceń (8.58). W tym oknie można również zapoznać się z opisem polecenia approach oraz przejrzeć listę dostępnych parametrów.



Rysunek 8.58: Sposób użycia polecenia approach

. Po dodaniu polcecenia approach do linii komend (8.59) należy wypełnić listę parametrów. Parametr $type$ określa typ ruchu, $base_point$ określa punkt odniesienia (brak punktu bazowego powoduje ruch względem aktualnej pozycji robota, pobieranej na początku realizowania danej ścieżki). Za pomocą parametrów dx, dy, dz określa się przesunięcie w poszczególnych osiach a rx, ry i rz określają rotację punktu TCP względem orientacji w punkcie bazowym.



Rysunek 8.59: Polecenie approach po dodaniu do linii komend

Listing 8.16: Przykład użycia polecenia approach w kodzie programu

```
1 es.move(type=es.Point, end=point('base'), speed=20.0, acc=20.0)
2 es.approach(type=es.Line, base_point=point('base'), dx=0.1, dy=0.2)
3 es.execute_move()
```

8.5 Funkcje

Zastosowanie

Często używane fragmenty skryptu robota można wydzielić i zgrupować w postaci funkcji. Stworzoną funkcję można wywoływać wielokrotnie w skrypcie. Funkcje, po zaimportowaniu, mogą być także wywoływane przez inne skrypty robota.

> *i* Używanie funkcji wewnątrz niej samej jest możliwe. Takie wykonywanie skryptu nazywa się rekurencją.

Definiowanie funkcji

Funkcja w skrypcie posiada charakterystyczną składnię. Wszystkie elementy umieszczone w jej ciele posiadają przynajmniej jedno wcięcie więcej niż sama definicja funkcji.

Listing 8.17: Definiowanie prostej funkcji

```
1 def funkcja1():
2     komenda1
3     komenda2
4     ...
```

Funkcja parametryzowana

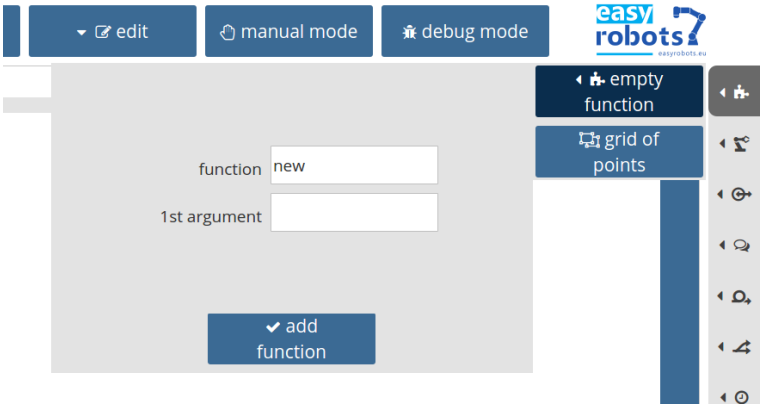
Istnieje możliwość stworzenia funkcji, do której przekazywane są parametry. Podane wartości są przypisywane do zdefiniowanych podczas tworzenia funkcji zmiennych. Parametry mogą być obligatoryjne lub opcjonalne.

Listing 8.18: Definiowanie funkcji przyjmującej parametry

```
1 def funkcja2(parametr1,parametr2=2):
2     komenda1
3     komenda2
4     ...
```

Dodawanie funkcji

Dodawanie funkcji za pomocą edytora odbywa się poprzez uzupełnienie formularza znajdującego się w menu panelu bocznego.



Rysunek 8.60: Dodawanie nowej funkcji

Używanie funkcji w skrypcie

W celu wywołania funkcji w skrypcie należy podać jej nazwę z nawiasami okrągłymi, co świadczy o chęci jej wywołania. Jeśli wywoływana funkcja przyjmuje parametry, należy podać je w nawiasach okrągłych. Podane dane są kolejno przypisywane do zmiennych zdefiniowanych przy tworzeniu funkcji.

Listing 8.19: Przykład użycia stworzonych funkcjij

```
1 funkcja1()
2 funkcja2(1)
```

Funkcja "point"

Nowo stworzony program posiada predefiniowaną funkcję parametryzowaną o nazwie "point". Służy ona do przechowywania punktów zapisanych w programie. Aby funkcja zwróciła dany punkt należy wywołać ją, jako parametr podając nazwę tego punktu.

8.6 Biblioteki

Importowanie modułów

Stworzone wcześniej funkcje można importować do bieżącego skryptu. Daje to możliwość przygotowania sobie bibliotek zapisanych w pamięci, używanych w różnych skryptach. Istnieje również możliwość importowania standardowych bibliotek języka Python w wersji 3.7.

Sposób używania

Importowanie modułu odbywa się za pośrednictwem słowa kluczowego "import". Istnieje kilka możliwości zaimportowania modułu.

Listing 8.20: Sposoby importowania bibliotek

```
1 import numpy
2 import numpy as np
3 from numpy import rad2deg
```

Biblioteka robota

Sterowanie robotem odbywa się za pośrednictwem przygotowanej do tego celu biblioteki. Jest ona domyślnie zaimportowana w skrypcie z aliasem "es".

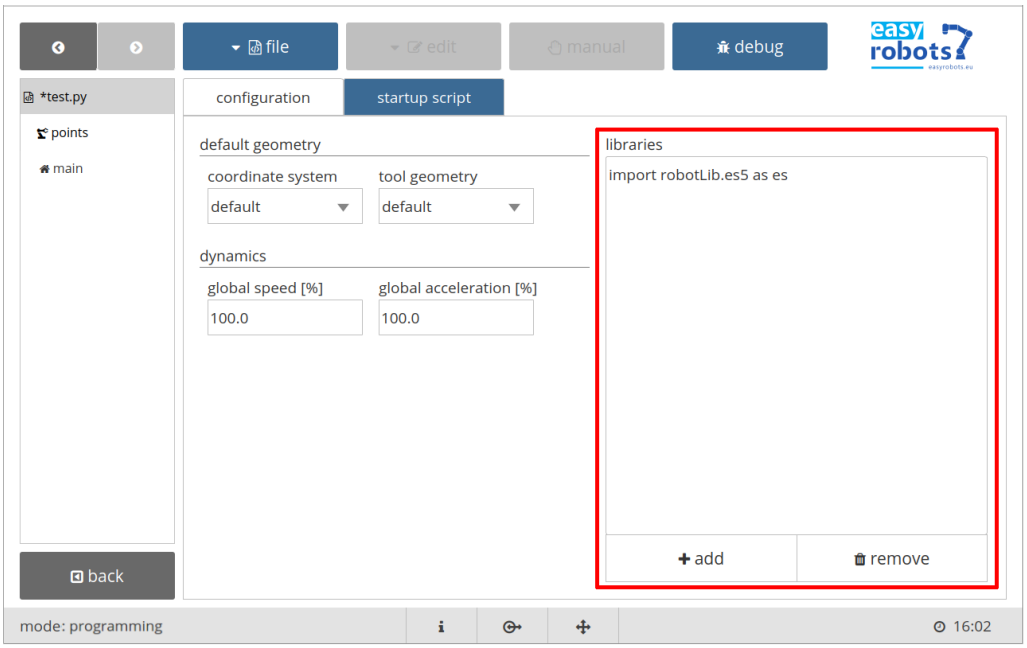
Listing 8.21: Import biblioteki robota

```
1 import robotLib.es5 as es
```

Zarządzanie bibliotekami

Po kliknięciu na nazwę tworzonego skryptu uaktywnia się okno konfiguracji. Jednym z jego elementów jest okno zarządzania bibliotekami. Domyślnie znajduje się

w nim biblioteka standardowa robota. Listę tę można modyfikować dodając rozszerzenia Python'a, a także własne biblioteki oraz skrypty w celu wykorzystania stworzonych w nich funkcji.



Rysunek 8.61: Panel zarzadzania bibliotekami

8.7 Instrukcje kontroli nad robotem

Moduł I/O

W celu komunikacji robota ze światem zewnętrznym, szafa sterownicza została wyposażona w moduł cyfrowych wejść i wyjść (I/O). Ich ilość zależy od wersji zamówionego robota i mogą być one rozszerzane. Minimalna ilość wyjść to 4, natomiast minimalna ilość wejść to 8. Moduły rozszerzeń posiadają po 8 wejść i 8 wyjść. Sterowanie wyjściami odbywa się za pomocą funkcji "set_output". Odczyt wyjścia jest możliwy przy pomocy funkcji "get_output", natomiast odczyt wejścia: "get_input". Jako argumenty funkcji należy podać numer wejścia/wyjścia oraz nowy stan w przypadku ustawiania go na wyjściu.

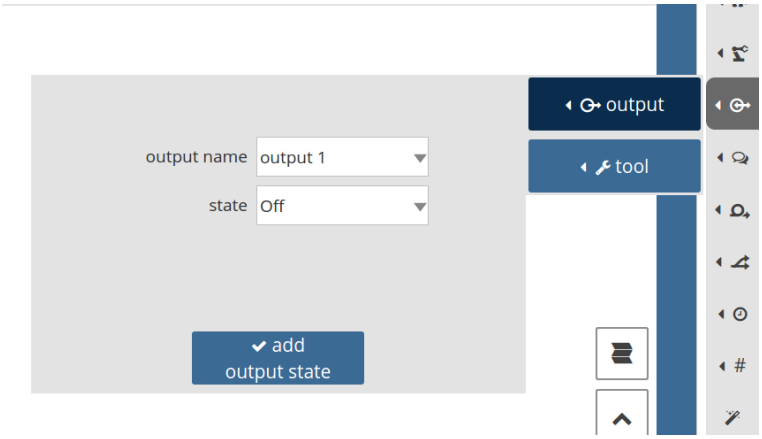
Listing 8.22: Kontrola modułu wejść i wyjść

```
1 es.set_output(1,True)
2 es.get_output(1)
3 es.get_input(1)
```

Należy pamiętać, że część wejść I/O jest fabrycznie przypisana do kontroli stanu obwodów elektrycznych robota przez system sterowania. W związku z tym ilość wejść przeznaczonych do użytku w aplikacji jest mniejsza.

Ustawianie wyjścia

Ustawianie bądź zerowanie pojedynczego wyjścia można łatwo zrealizować za pośrednictwem formularza znajdującego się w menu panelu bocznego.



Rysunek 8.62: Formularz dodawania komendy ustalającej stan wyjścia I/O

Obsługa narzędzi

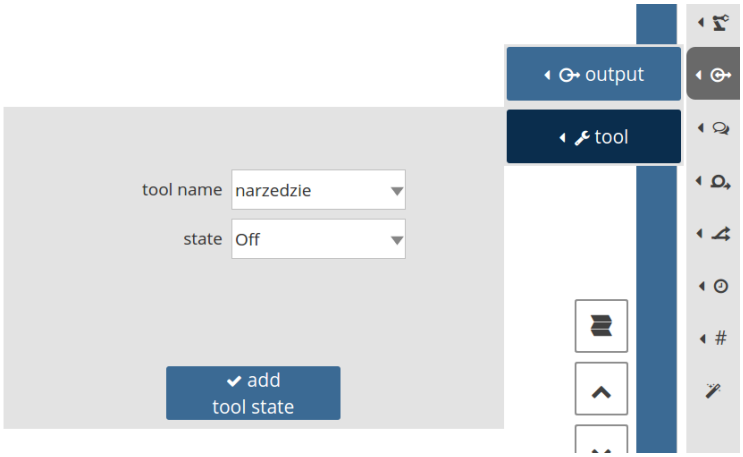
Zdefiniowane wcześniej metody obsługi narzędzi można w prosty sposób przywołać przy pomocy funkcji "set_tool_state". Jako argumenty przyjmuje ona nazwę narzędzia oraz jego nowy stan.

Listing 8.23: Zmiana stanu narzędzia

```
1 es.set_tool_state("narzedzie1", True)
2 es.set_tool_state("narzedzie1", False)
```

Zmiana stanu narzędzia

Zmiana stanu zdefiniowanego narzędzia może zostać dodana za pomocą formularza z menu bocznego edytora.



Rysunek 8.63: Formularz dodawania komendy zmiany stanu narzędzia

Stan hamulców

Aby przetestować manipulator pod kątem stanu hamulców należy użyć funkcji "get_brakes_state". Funkcja ta zwraca listę 6 wartości odpowiadających przegubom robota, począwszy od joint1 aż do joint6. Wartość "True" oznacza, że hamulec został zwolniony. Za pomocą funkcji "set_brakes_state" można zablokować/zwolnić poszczególne hamulce robota.

Listing 8.24: Kontrola hamulców stawów manipulatora

```
1 state = es.get_brakes_state()
2 es.set_brakes_state([False,False,False,False,False,False])
```

8.8 Typy danych

Tworzenie zmiennych

W celu stworzenia zmiennej i przypisania jej nazwy należy wykorzystać znak przypisania "=". Po lewej stronie przypisania znajduje się nazwa zmiennej, natomiast po prawej - wyrażenie tworzące zmienną.

Dane liczbowe

Zmienna liczbowa, zgodnie z powyższą definicją, tworzona jest poprzez przypisanie liczby, bądź wyniku wyrażenia do nazwy.

Listing 8.25: Przykłady tworzenia zmiennych liczbowych.

```
1 variable1 = 1
2 variable2 = variable1+1
```

Listy

Dane liczbowe mogą być grupowane w listę. Definiuje się ją poprzez podanie ciągu liczb (lub dowolnych innych typów danych), oddzielonych przecinkami i zamkniętych w nawiasy kwadratowe. Wielkość listy może być zmieniana w trakcie jej życia.

Listing 8.26: Definiowanie i modyfikacja listy

```
1 listVariable = [1,2,3,4,5]
2 listVariable[0] = 0
3 listVariable.append(6)
```

Metody zmiennych

Język Python jest obiektowy. Oznacza to, między innymi, że wszystkie zmienne są obiektami posiadającymi pewne metody. W przypadku list przydatnymi metodami są:

- append(e) - dodanie jednego elementu po ostatnim,
- clear() - usuniecie wszystkich elementów,
- copy() - płytka kopia listy,
- count(e) - zliczanie wystąpień elementu e,
- index(e) - znajdowanie położenia pierwszego wystąpienia e
- insert(p,e) - wstawienie elementu e przed elementem na pozycji p,
- pop([p]) - usunięcie i zwrócenie ostatniego elementu lub elementu na opcjonalnej pozycji p
- remove(e) - usunięcie pierwszego wystąpienia elementu e
- reverse() - odwrócenie kolejności elementów

Wycinanie

Listy posiadają możliwość wycinania. W wyniku wycinania zwrócona zostaje cześć pierwotnej listy. Operacje tą definiuje się za pomocą składni:

Listing 8.27: Składnia wycinania

```
1 nowaLista = lista[a:b:c]
```

przy czym *a* oznacza indeks pierwszego elementu wyciętej listy, *b*- indeks pierwszego elementu (gdy ujemny- indeks liczony od końca listy), który nie będzie zawierał się w nowej liście (gdy ujemny- indeks liczony od końca listy), *c*- krok, z jakim będą wycinane elementy (co który)(gdy ujemny- elementy będą ułożone w odwrotnej kolejności).
W przypadku pominięcia elementu *a* wycięty zostanie fragment począwszy od elementu 0, w przypadku pominięcia elementu *b* wycięty zostanie fragment od elementu *a* do końca bazowej listy. W przypadku pominięcia elementu *c* wycięty zostanie każdy element bazowej listy z podanego przedziału.

Listing 8.28: Przykłady wycinania listy

```
lista = [1,2,3,4,5]
nowa1 = lista[1:-1] #[2,3,4]
nowa2 = lista[:3] #[1,2,3]
nowa3 = lista[::2] #[1,3,5]
```

Pozycja manipulatora

Pozycja manipulatora jest zbiorem danych liczbowych określających położenie poszczególnych przegubów w przestrzeni. Położenie może być zdefiniowane jako zbiór wartości kątowych, w tzw. współrzędnych wewnętrznych (przegubowych; ang. internals) lub położenie końcówki TCP w przestrzeni współrzędnych kartezjańskich. W obu przypadkach położenie to jest definiowane za pomocą specjalnie stworzonej do tego celu klasy. W przypadku współrzędnych przegubowych jest to "InternalPos":

Listing 8.29: Definicja współrzędnych przegubowych

```
data = es.InternalPos([180.0, 90.0, 0.0, 90.0, 180.0, 0.0])
```

W przypadku współrzędnych kartezjańskich jest to "CartesianPos":

Listing 8.30: Definicja współrzędnych kartezjańskich

```
data = es.CartesianPos([0.0, 0.187, 0.921], [180.0, 0.0, 90.0])
```

Dane tekstowe

Dane tekstowe w skrypcie robota są używane m.in. do tworzenia wiadomości wyświetlanych w oknach dialogowych dla operatorów. Ich tworzenie polega na umieszczeniu tekstu pomiędzy dwoma cudzysłowami " lub apostrofami '.

Listing 8.31: Tworzenie zmiennej tekstowej

```
variable = "Text␣for␣the␣operator"
```

Dane logiczne

Wyrażenia logiczne w skrypcie robota są możliwe przy pomocy typu logicznego (*bool*). Przyjmuje on jedną z dwóch wartości: *True* lub *False*. Wyrażenia logiczne zawsze zwracają daną typu *bool*.

W języku Python, który jest natywnym językiem programowania robota, istnieją zasady określające tłumaczenie innych typów na typ logiczny bool, dlatego też: wartość 0 jest traktowana jako *False*, podczas gdy każda inna liczba jako *True*; pusta lista jest traktowana jako *False*, a zawierająca elementy jako *True*.

Zmienne lokalne i globalne

Zmienne stworzone w funkcji są dostępne jedynie w obrębie tej funkcji. Są to tzw. zmienne lokalne. Aby móc stworzyć lub modyfikować zmienną globalną w języku Python, należy wcześniej zadeklarować taką chęć poprzez użycie słowa kluczowego *global*. Zmienna taka jest dostępna w dowolnej funkcji w ramach danego pliku skryptu.

Listing 8.32: Tworzenie i modyfikacja zmiennej globalnej

```
global g_var

g_var = 1
g_var = g_var+10
```

Deklaracja *global* musi znaleźć się w każdej funkcji modyfikującej zmienną globalną. W przeciwnym wypadku Python stworzy lokalny odpowiednik zmiennej globalnej zamiast ją modyfikować.

8.9 Logika

Porównywanie danych

Aby móc sterować przepływem programu, konieczna jest możliwość porównywania danych ze sobą. Efektem porównania jest zawsze wynik typu *bool*. W celu porównania, podobnie jak w innych językach programowania, można wykorzystać operatory porównania, takie jak:

- == równy,
- ! = nie równy,
- > większy,
- < mniejszy,
- >= większy lub równy,
- <= mniejszy lub równy.

Listing 8.33: Przykład porównania danych

```
1 1==2 # False
2 1!=2 # True
3 1>2 # False
4 1<2 # True
5 variable = 3>=1
```

Operatory *not and* i *or*

Operatory *and* oraz *or* służą do łączenia warunków. Jeśli przed przystąpieniem do wykonywania danej części skryptu należy sprawdzić więcej niż jeden warunek, można je sprawdzić przy pomocy jednej instrukcji warunkowej. W tym celu pomiędzy warunkami należy wstawić słowa kluczowe *and* jeżeli oba warunki muszą być spełnione, lub *or* jeśli co najmniej jeden z nich musi być spełniony aby wykonać daną część skryptu. Operator *not* neguje wynik działania logicznego.

Listing 8.34: Przykłady łączonych operacji logicznych

```
1 1>2 and 2>1 #False
2 1>2 or 2>1 #True
3 1>2 or not 2>1 #False
```

8.10 Instrukcje kontroli przepływu programu

Oczekiwanie czasowe

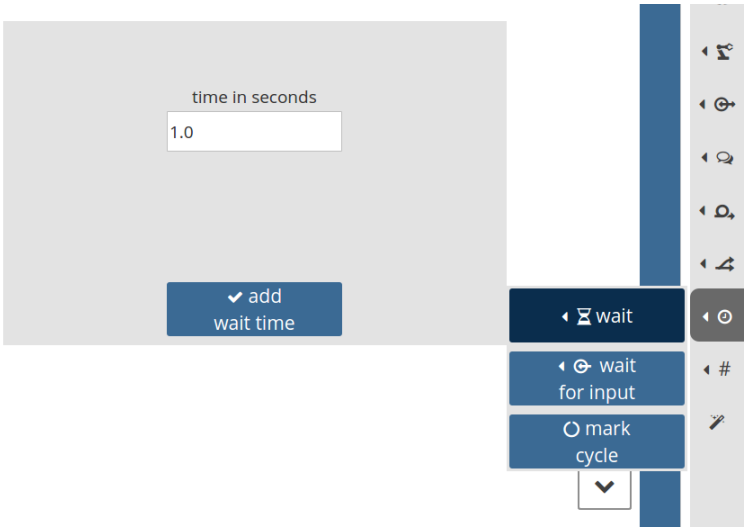
Jeśli wartość czasu oczekiwania jest znana przed jego rozpoczęciem, w celu wprowadzenia oczekiwania należy użyć funkcji z modułu robota o nazwie "wait". Jako argument należy podać czas oczekiwania wyrażony w sekundach.

Listing 8.35: Instrukcja oczekiwania przez 0.5 sekundy

```
1 es.wait(0.5)
```

Dodawanie oczekiwania

Dodawanie oczekiwania przez jakiś czas można zrealizować za pomocą formularza w panelu bocznym edytora.



Rysunek 8.64: Formularz dodawania komendy oczekiwania czasowego

Oczekiwanie na stan zmiennej

Oczekując na pojawienie się określonego stanu zmiennej (gdy jej wartość zmieniana jest przez inny wątek programu) można wykorzystać element języka Python o nazwie: pętla *while*. Pętla sprawdza podany warunek i wykonuje podane w jej ciele instrukcje do momentu, gdy warunek nie zostanie spełniony. Jeśli blok skryptu ma zadanie jedynie sprawdzać warunek, ciało pętli należy wypełnić komendą *pass* (nie rób nic).

Listing 8.36: Oczekiwanie na wartość 1 zmiennej o nazwie data

```
1 while data != 1:
2     pass
```

Oczekiwanie na stan wejścia

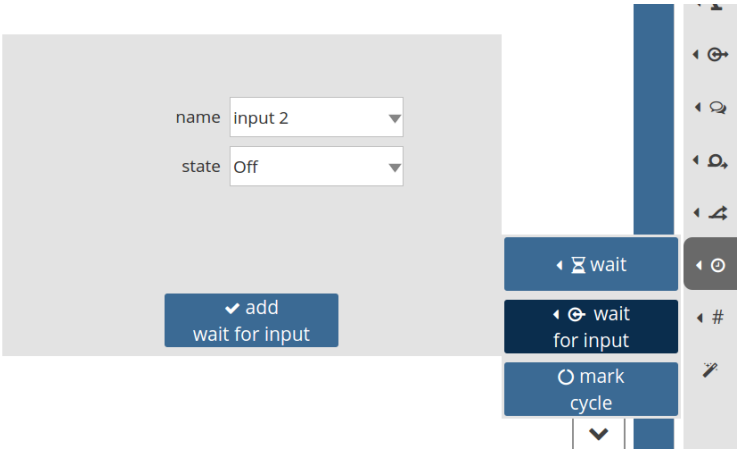
Sprawdzanie stanu wejść I/O robota bardzo często wiąże się z oczekiwaniem na określony stan jednego z nich. W tym celu odpowiednie jest użycie funkcji "get_input" w połączeniu z pętlą *while*.

Listing 8.37: Oczekiwanie na pojawienie się stanu wysokiego na wejściu nr 1

```
1 while es.get_input(1) is False:
2     pass
```

Dodawanie oczekiwania na wejście

Realizacja oczekiwania na stan wejścia wymaga stworzenia pętli sprawdzającej jego stan. Można tego dokonać wypełniając formularz znajdujący się w panelu bocznym edytora.



Rysunek 8.65: Formularz dodawania sekwencji oczekiwania na stan wejścia I/O

Instrukcje warunkowe

Warunkowe wykonywanie bloków programu umożliwia instrukcja *if*. Jeśli podany warunek, po wykonaniu, zwraca wartość *True*, to ciało instrukcji zostaje wykonane. Jeśli nie, zostaje pominięte.

Alternatywa

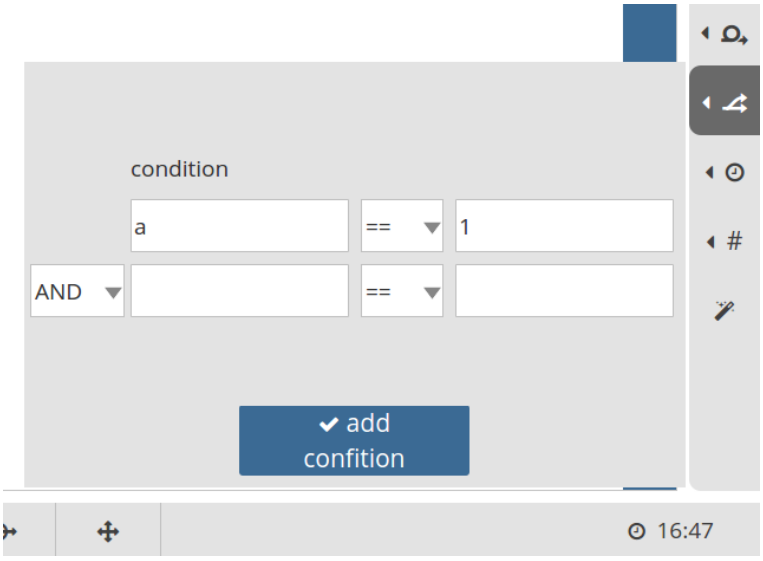
Jeśli warunek podany w instrukcji *if* jest niespełniony, istnieje możliwość wykonania alternatywnego bloku instrukcji, stworzonego po słowie kluczowym *else*.

Listing 8.38: Instrukcja warunkowa

```
1 if data == True:
2     ans = 1 # data is True
3 else:
4     ans = 2 # data is not True
```

Dodawanie warunków

Proste wyrażenia warunkowe można umieścić w skrypcie przy pomocy formularza z bocznego menu edytora. Wyrażenie alternatywy *else* można dodać dopiero po stworzeniu warunku *if*. Dokonuje się tego przy pomocy menu podręcznego komendy *if* i przycisku else .



Rysunek 8.66: Formularz dodawania wyrażeń warunkowych

Instrukcja alternatywy *else* nie może istnieć bez poprzedzającej jej instrukcji warunku *if*!

Pętla for

Zamiast powielać elementy programu w celu kilkukrotnego ich wykonania, można wykorzystać funkcjonalność pętli *for*. Zadaniem tej instrukcji jest iteracyjne wykonywanie bloku programu. Alias podany po słowie kluczowym *for* jest nadawany charakterystycznemu dla danej iteracji elementowi, będącemu częścią zbioru podanego po słowie kluczowym *in*.

Listing 8.39: Pętla *for* wykonująca blok programu 5 razy

```
1 for i in range(5):
2     data = data+1 # i= 0...4
```

Pętla *for* wykonuje swoje ciało tyle razy, ile elementów liczy podany zbiór. W celu wykonania najprostszej pętli wykonującej ciało *x* razy, należy użyć funkcji "range", tworzącej zbiór (listę) elementów przyjmujących wartości od 0 do x-1.

Dodawanie pętli *for*

Pętla for w najprostszej postaci może zostać stworzona za pomocą przygotowanego w tym celu formularza z menu bocznego.

Rysunek 8.67: Formularz dodawania pętli for

8.11 Interakcja z otoczeniem

Okna dialogowe

Jednym ze sposobów interakcji z operatorem, podczas wykonywania programu, jest wstrzymanie programu oraz wyświetlenie okna dialogowego na ekranie panelu dotykowego. Po zatwierdzeniu przez operatora, program jest kontynuowany. Celem wyświetlenia okna może być samo wstrzymanie wykonywania programu, aby operator potwierdził wykonanie jakiejś operacji, podjął decyzję lub podał wartość. Aby wyświetlić okno dialogowe, należy wywołać funkcję o nazwie "view_dialog". Wygląd okna zależy od podanych argumentów:

- 'information' - okno informacyjne z pojedynczym przyciskiem "OK",
- 'question' - okno zapytania z przyciskami "Tak" i "Nie",
- 'value' - okno podawania wartości z polem edycyjnym oraz przyciskami "Akceptuj" i "Anuluj".

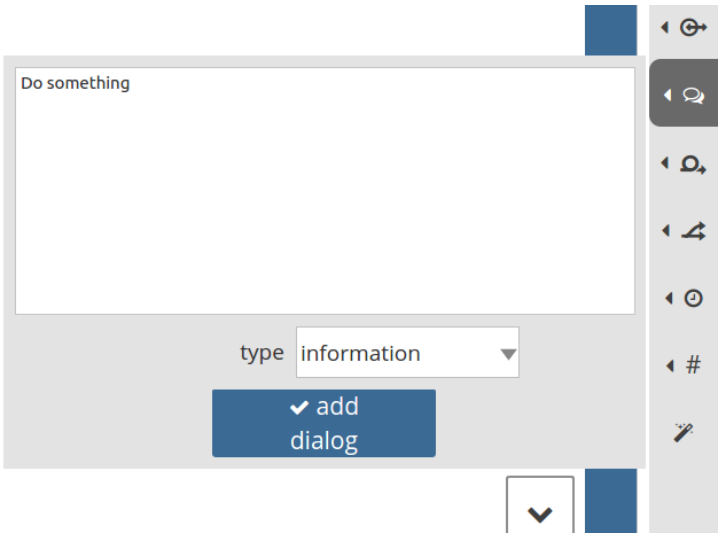
Funkcja "view_dialog" umożliwia wyświetlenie własnego tekstu okna. W przypad-

ku zwracania wartości liczbowej, należy także podać wartość domyślną (zwracaną po wciśnięciu przycisku "Anuluj").

Okno informacyjne

Listing 8.40: Okno informacyjne

```
1 ans = es.view_dialog('information','Do something')
```

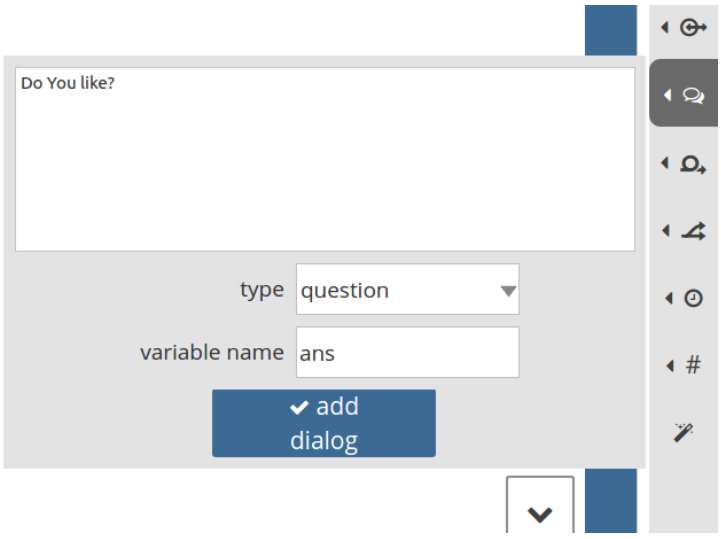


Rysunek 8.68: Formularz dodawania okna informacyjnego do skryptu

Okno zapytania

Listing 8.41: Okno zapytania

```
1 ans = es.view_dialog('question','Do You like?')
```

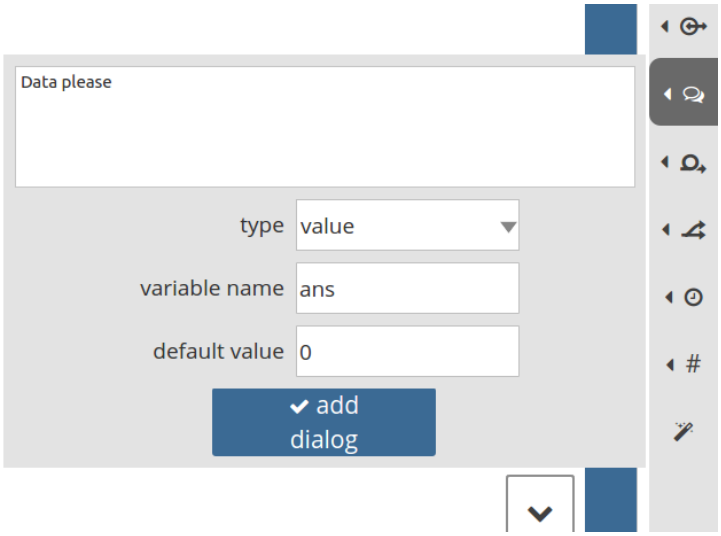


Rysunek 8.69: Formularz dodawania okna zapytania do skryptu

Okno podania wartości

Listing 8.42: Okno podawania wartości

```
1 ans = es.view_dialog('value','Data, please',0)
```



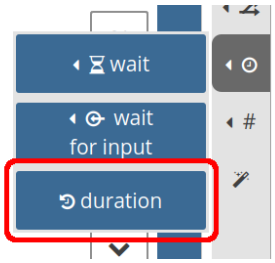
Rysunek 8.70: Formularz dodawania okna pobierania wartości do skryptu

Pomiar wydajności cyklu

Ważnym wskaźnikiem pracy robota jest czas cyklu. W bibliotekach robota znajduje się mechanizm do jego pomiaru. Aby go uruchomić należy dodać tzw. marker w skrypcie robota. Zadaniem markera jest określenie miejsca w programie, w którym znajduje się koniec aktualnego cyklu i jednocześnie początek następnego. Aby mechanizm ten działał poprawnie, zaleca się dodanie tylko jednego markera na cały wykonywany skrypt.

Listing 8.43: Komenda wskazująca miejsce zakończenia cyklu oraz zwracająca czas od ostatniego wywołania.

```
1 var.duration = es.get_duration()
```



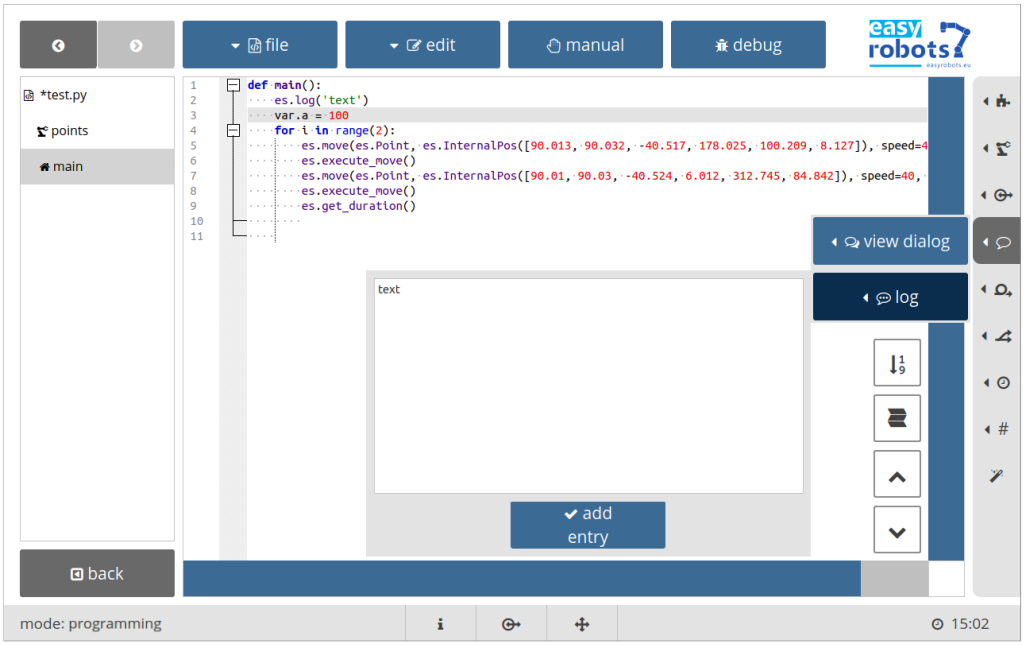
Rysunek 8.71: Dodawanie markera cyklu pracy robota wraz z pobraniem czasu od ostatniego wywołania

logowanie do konsoli

Wykonujący się program ma możliwość wysyłania komunikatów do specjalnej konsoli dostępnej na panelu oteratora w celu logowania informacji o przebiegu programu. Dostęp do konsoli znajduje się w bocznym pasku okna debuggowania oraz w bocznym pasku okna uruchamiania programu. W celu dostepu do konsoli stworzona została komenda "log".

Listing 8.44: Komenda wyświetlająca w przygotowanej konsoli kumunikat o zadanej treści lub wartość podanej zmiennej.

```
1 es.log("Wykonano␣cykl:",x)
```



Rysunek 8.72: Dodawanie komendy do programu

Zmienne współdzielone

Jednym ze sposobów wymiany danych z otoczeniem jest współdzielenie zmiennych. Wartości zmiennych mogą być modyfikowane przez inne urządzenia podłączone do sieci robota, a także za pośrednictwem wizualizacji dostępnej na panelu operatorskim. Tworzenie, modyfikowanie i odczyt zmiennych współdzielonych w robotach Easy Robots odbywa się za pomocą obiektu o nazwie "var", tworzonego w programie robota zaraz po uruchomieniu.

Odczyt i zapis

W celu utworzenia/zmodyfikowania zmiennej należy utworzyć/zmodyfikować atrybut obiektu o nazwie "var" poprzez przypisanie mu wartości. Odczyt zmiennej odbywa się poprzez odczyt atrybutu.

Listing 8.45: Definiowanie i modyfikacja zmiennych współdzielonych

```
1 var.x = 10
2 var.x = var.x+10
```

Odczyt wartości zmiennej współdzielonej, która nie została wcześniej zainicjowana spowoduje zwrócenie liczby 0.

Każdy zapis i odczyt zmiennej współdzielonej wiąże się z koniecznością wykonania kilku operacji tzw. "zakulisowych" co powoduje generowanie opóźnienia. W celu utrzymania szybkości działania skryptu należy użycie zmiennych współdzielonych ograniczyć do koniecznego minimum.

8.12 Zapis danych do pamięci trwałej

Dualizm zmiennych współdzielonych

Zmienne współdzielone, opisywane w poprzedniej sekcji, zapisywane są w pamięci trwałej robota. Dzięki temu ich wartości są dostępne po ponownym uruchomieniu programu oraz po ponownym uruchomieniu robota.

8.13 Alternatywne wątki

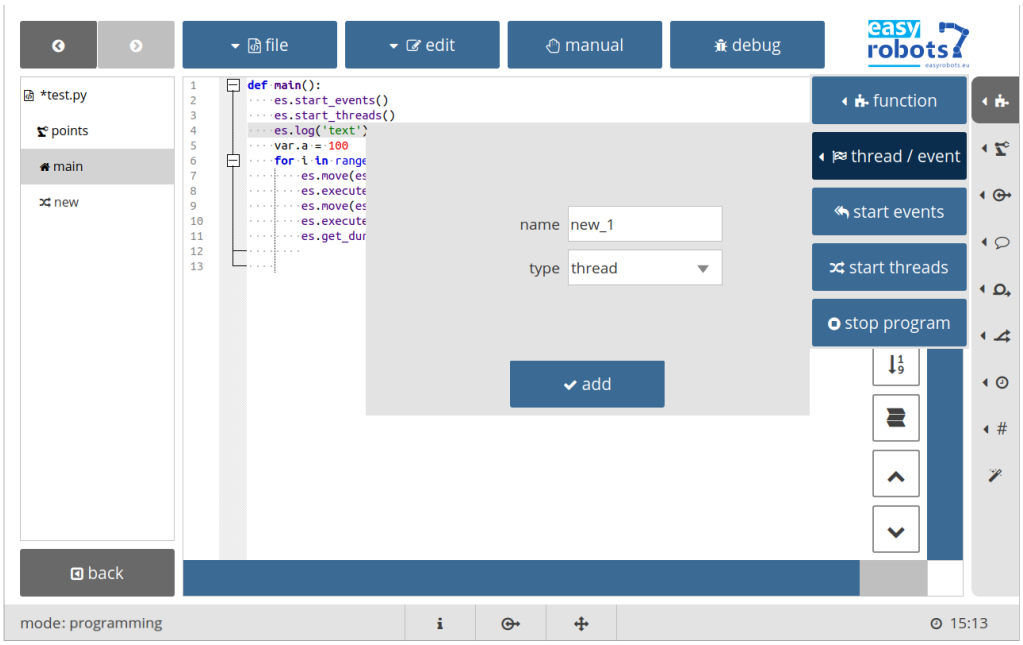
Dodawanie wątku

Wygodnym i często niezbędnym rozwiązaniem, służącym do kontroli otoczenia i urządzeń współpracujących, jest stworzenie skryptu wykonywanego równolegle do głównego wątku. W tym celu wymagane jest stworzenie funkcji nieparamatryzowanej, opatrzonej dekoratorem "es.thread". Uruchommienie wszystkich wątków jest kontrolowane przez programistę i odbywa się po wykonaniu komendy "es.start_threads". Interfejs udostępnia przyciski i formularz ułatwiający dodawanie do skryptu robota wątków oraz uruchamianie ich.

Listing 8.46: Tworzenie i uruchamianie wątku

```
@es.therad
def thread_function():
    pass

es.start_threads()
```



Rysunek 8.73: Formularz dodający funkcję wykonywaną w oddzielnym wątku programu

Przebieg

Wątek rozpoczyna swoją pracę w trakcie wykonywania polecenia "start_threads" oraz kończy wraz z końcem pracy głównego wątku, lub w wyniku zakończenia wykonywania funkcji przypisanej do wątku.

Zatrzymanie manipulatora

W skrypcie wątku nie zaleca się używania poleceń służących do zmiany położenia manipulatora. Można jednak wpływać bezpośrednio na jego pracę poprzez wymuszenie jego zatrzymania za pomocą polecenia "stop_move".

Listing 8.47: Zatrzymanie ruchu manipulatora

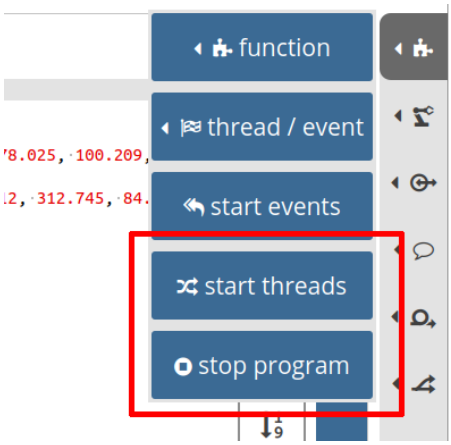
```
es.stop_move()
```

Zatrzymanie programu

W celu zatrzymania działania skryptu robota (wszystkich jego wątków) można użyć polecenia "stop_program".

Listing 8.48: Zatrzymanie wykonywania programu

```
1 es.stop_program()
```



Rysunek 8.74: Przyciski dodające komendy zatrzymania manipulatora oraz zatrzymania programu

Komunikacja pomiędzy wątkami

Wątki w języku Python mają wspólną przestrzeń zmiennych. Oznacza to, że zmienne globalne stworzone w wątku głównym są dostępne w wątkach i na odwrót.

> *i* W celu wymiany danych można polegać na zmiennych współdzielonych, lecz ze względu na opóźnienia czasowe związane z ich zapisem i odczytem, nie jest to rozwiązanie polecane.

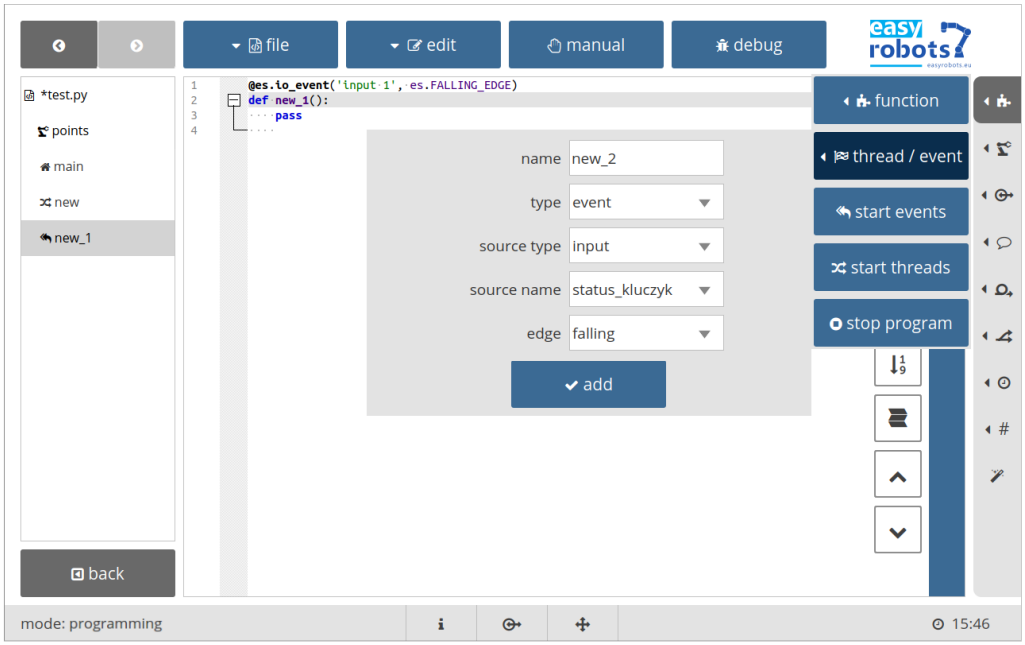
8.14 Zdarzenia

Dodawanie zdarzenia

Wyspecjalizowanym typem wątku, odpowiedzialnym za kontrolę parametrów robota jest zdarzenie. Pozwala ono reagować na pojawienie się pewnych okoliczności otoczenia lub zarejestrować ich fakt, podczas gdy wątek gówny skrypu jest zajęty wykonywaniem innych czynności. W tym celu wymagane jest stworzenie funkcji nieparamatryzowanej, opatrzonej dekoratorem "es.io_event","es.time_event","es.variable_e lub "es.end_event". Uruchommienie obsługi wszystkich eventów jest kontrolowane przez programistę i odbywa się po wykonaniu komendy "es.start_events". Interfejs udostępnia przyciski i formularz ułatwiający dodawanie eventów do skryptu robota oraz uruchamianie ich.

Listing 8.49: Uruchomienie obsługi zdarzenia

```
1 @es.io_event('input␣1', es.FALLING_EDGE)
2 def in_falling_event():
3     pass
4
5 @es.io_event('input␣1', es.RISING_EDGE)
6 def in_rising_event():
7     pass
8
9 es.start_events()
```



Rysunek 8.75: Formularz dodający funkcję pełniącą obsługę zdarzenia

Rodzaje zdarzeń

W oprogramowaniu ESControl uwzględniono kilka rodzajów zdarzeń przechwytywanych przez oprogramowanie. Są to:

- "io_event" -zdarzenie wywoływane zmianą stanu wejścia lub wyjścia IO na konkretną wartość podaną w definicji zdarzenia
- "time_event" -cyklicznie wywoływane zdarzenie
- "variable_event" -zdarzenie wywoływane poprzez zmianę stanu zmiennej współdzielonej o nazwie podanej w definicji eventu
- "end_event" -zdarzenie wykonywane po zakończeniu wykonywania programu (gdy wątek główny już nie pracuje)

Przebieg

W momencie pojawienia się określonego stanu zasobu, wywołana zostaje zdefiniowana przez programistę funkcja, mająca na celu zareagować na pojawienie się zdarzenia. Zdarzenie jest wykonywane każdorazowo po wykryciu zmiany stanu na taki, który jest podany w jego deklaracji.

8.15 Opis programu

Komentarze

Niezależnie od używanego języka programowania, należy dbać o czytelność napisanego kodu. W osiągnięciu tego celu pomocne są komentarze. W języku Python komentarz oznaczony jest symbolem #.

Listing 8.50: Możliwości komentowania napisanego skryptu

```
1 a = 1 #a now has the value 1
2 #free comment
```

Dodawanie komentarza

Samodzielne komentarze mogą być dodawane przy pomocy formularza znajdującego się w panelu bocznym edytora.

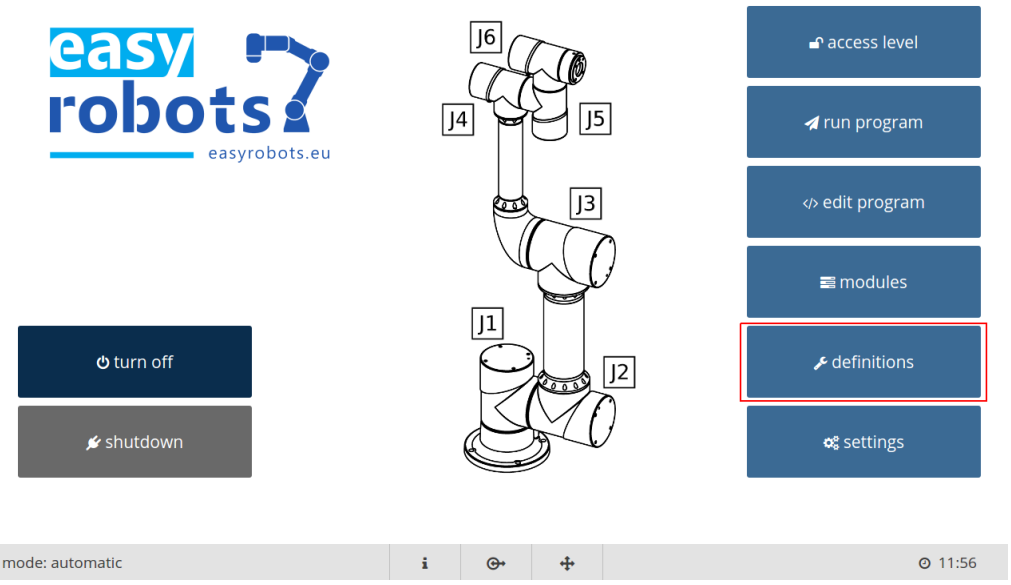
Rysunek 8.76: Dodawanie komentarza do skryptu

Komentarze opisujące operacje dokonywane w poszczególnych liniach, mogą być również dodawane w oknie tworzenia komend programu.

9 Konfiguracja

Okno definicji

Układy odniesienia, geometrie narzędzi, sterowanie narzędziami oraz ograniczenia przestrzeni robota konfigurowane są w oknie definicji, uruchamianym z okna startowego aplikacji EScontrol - rysunek 9.77.



Rysunek 9.77: Uruchomienie okna definicji.

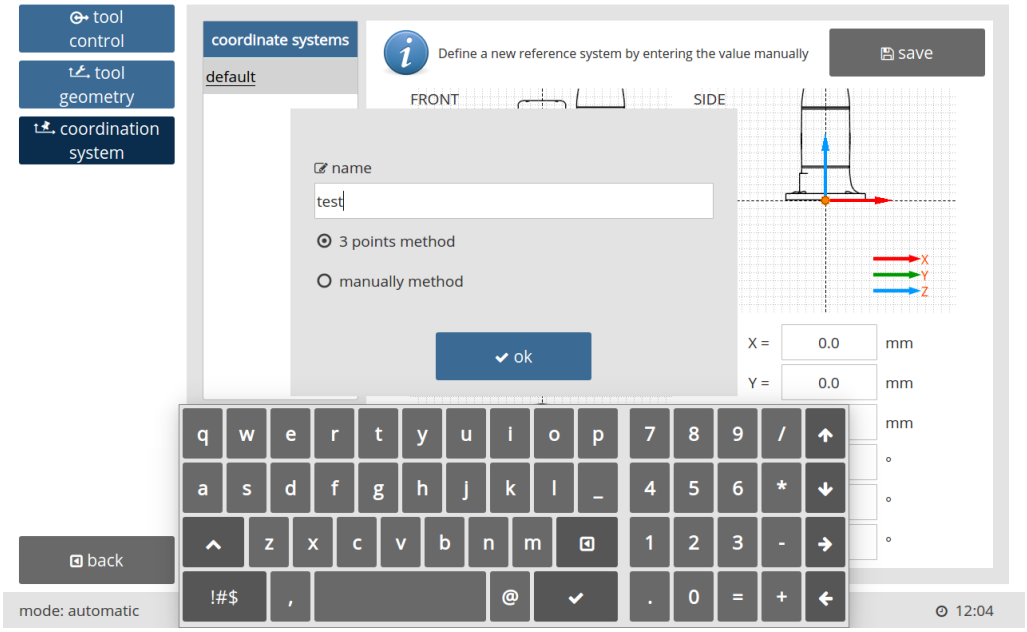
9.1 Układy odniesienia

Zastosowanie

Układ referencyjny jest definiowany za pomocą punktu w przestrzeni kartezjańskiej, odniesionego do podstawowego układu współrzędnych. Wszelkie ruchy robota wykonywane w przestrzeni kartezjańskiej mogą być wykonywane względem podstawowego układu współrzędnych, lub innego, zdefiniowanego przez operatora, układu odniesienia.

Zarządzanie

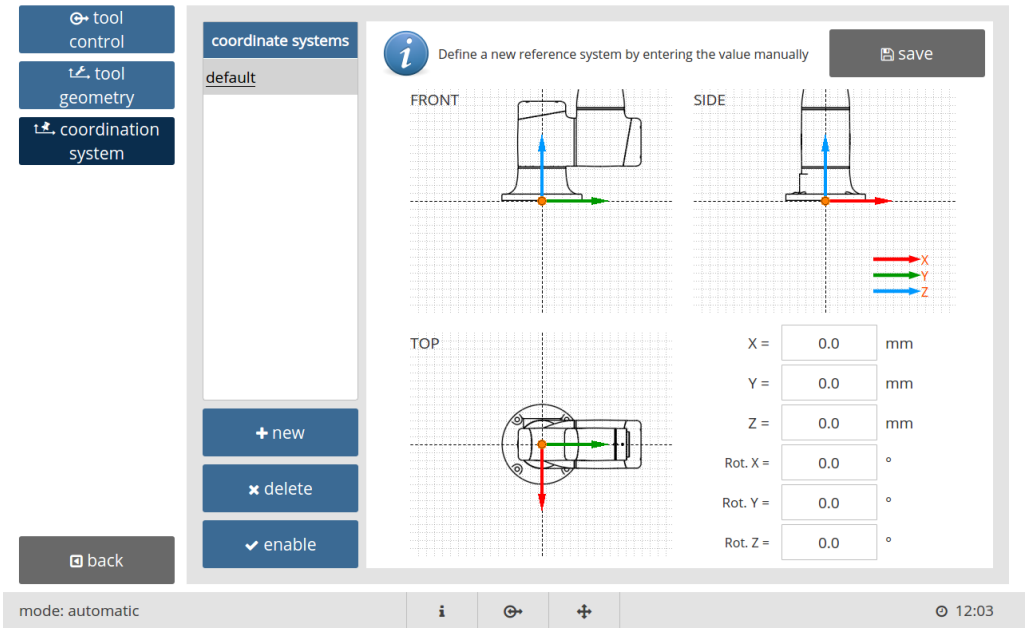
Zarządzanie zdefiniowanymi układami odniesienia odbywa się za pomocą karty w oknie definicji, dostępnej pod przyciskiem coordination system . W celu dodania nowego układu odniesienia, należy nacisnąć przycisk new . Aby utworzyć układ, należy następnie wpisać jego unikalną nazwę oraz zdecydować, czy będzie definiowany metodą 3-punktową czy manualną.



Rysunek 9.78: Okno dodawania nowego układu odniesienia.

Metoda manualna

Definiowanie układu odniesienia metodą manualną polega na podaniu wartości punktu zerowego nowego układu w odniesieniu do układu podstawowego. Punkt ten określa przesunięcia i rotacje nowego układu względem układu podstawowego. Podane wartości należy zatwierdzić przyciskiem save .



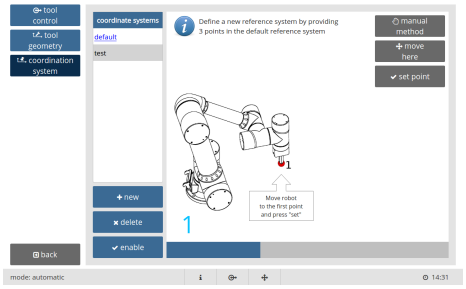
Rysunek 9.79: Okno definicji układów odniesienia metodą manualną.

Metoda 3-punktowa

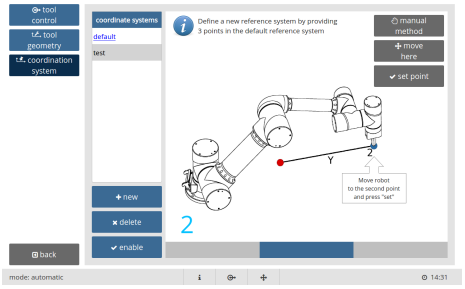
Metoda 3-punktowa polega na zdefiniowaniu układu odniesienia robota poprzez ustawienie manipulatora w trzech pozycjach, za każdym razem zatwierdzając swój wybór przyciskiem set point . Każdy z uzyskanych punktów posiada unikalną funkcję:

1. Pierwszy z punktów wyznacza początek nowego układu współrzędnych.
2. Drugi, w odniesieniu do pierwszego, wyznacza kierunek i dodatni zwrot osi Y tworzonego układu.
3. Trzeci punkt, w odniesieniu do pierwszego i drugiego, wyznacza dodatni zwrot osi X nowego układu.

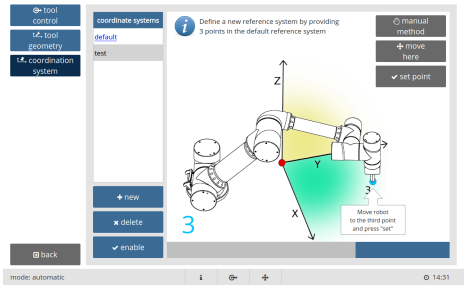
Oś Z w tak wyznaczonym układzie jest jednoznaczna i nie ma potrzeby jej wskazywać. Istnieje możliwość przywrócenia zapisanej pozycji punktu poprzez naciśnięcie przycisku move here . Po zapisaniu wszystkich trzech punktów pojawi się komunikat mówiący o tym, czy układ został poprawnie zdefiniowany. W celu zapisania nowego układu odniesienia należy nacisnąć przycisk save . Usuwanie zdefiniowanych układów jest możliwe poprzez naciśnięcie przycisku delete . Przycisk enable ustawia zaznaczony układ współrzędnych jako aktywny.



(a) Dodawanie pierwszego punktu.



(b) Dodawanie drugiego punktu.



(c) Dodawanie trzeciego punktu.

Rysunek 9.80: Metoda 3-punktowa.

Wszystkie układy odniesienia w robotach ES są prawoskrętne.

Przy definiowaniu punktów musi być aktywny domyślny układ odniesienia oraz zdefiniowana wcześniej geometria dla wykorzystywanego narzędzia.

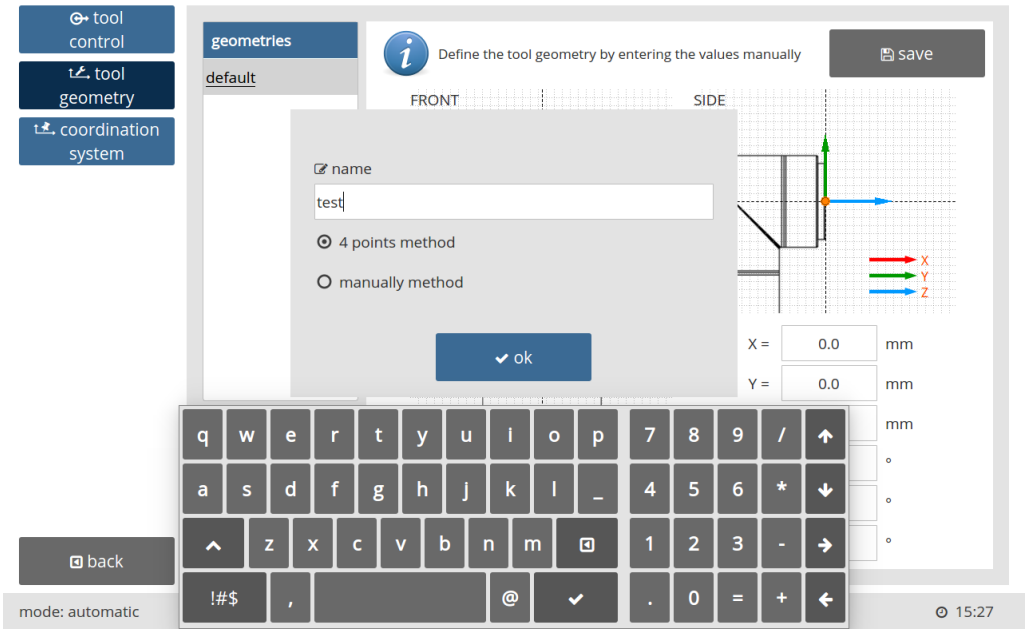
9.2 Geometrie narzędzi

Zastosowanie

Geometria narzędzia, podobnie jak układ odniesienia, jest definiowany za pomocą punktu w przestrzeni kartezjańskiej, odniesionego do punktu domyślnego punktu TCP manipulatora. Pozwala to na uwzględnienie narzędzia w łańcuchu kinematycznym robota.

Zarządzanie

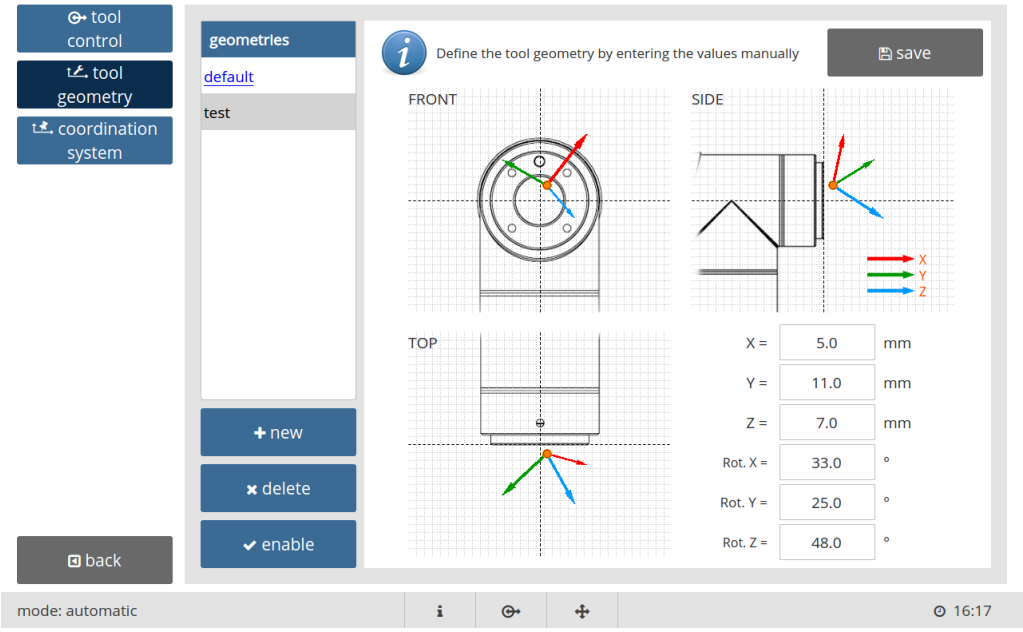
Zarządzanie zdefiniowanymi geometriami narzędzi odbywa się za pomocą karty w oknie definicji, dostępnej pod przyciskiem tool geometry . W celu dodania nowej geometrii narzędzia, należy nacisnąć przycisk new , wpisać jego unikalną nazwę oraz zdecydować, czy będzie definiowany metodą 4-punktową czy manualną.



Rysunek 9.81: Okno dodawania nowej geometrii narzędzia.

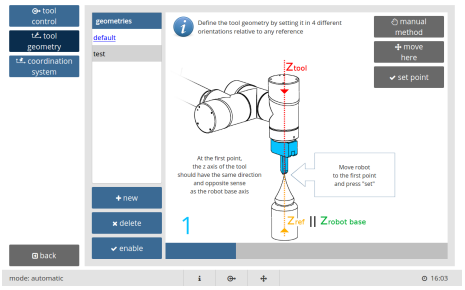
Metoda manualna

Definiowanie geometrii narzędzia metodą manualną polega na podaniu odległości oraz orientacji punktu roboczego narzędzia względem podstawowego punktu TCP manipulatora. Podane wartości należy zatwierdzić przyciskiem save .

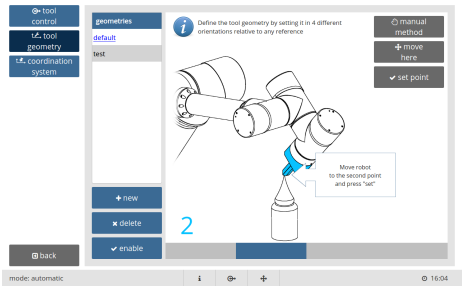


Rysunek 9.82: Manualna definicja geometrii narzędzia.

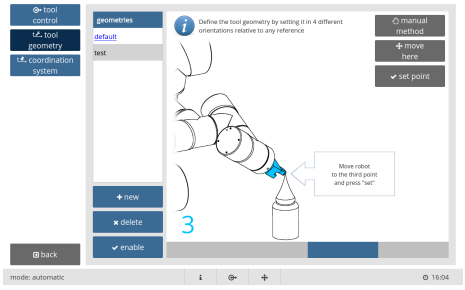
Metoda 4-punktowa



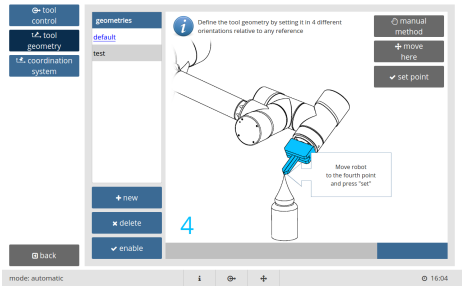
(a) Dodawanie pierwszego punktu.



(b) Dodawanie drugiego punktu.



(c) Dodawanie trzeciego punktu.



(d) Dodawanie czwartego punktu.

Rysunek 9.83: Metoda czteropunktowa.

Metoda 4-punktowa polega na zdefiniowaniu geometrii narzędzia robota poprzez ustawienie manipulatora w czterech pozycjach, za każdym razem zatwierdzając swój wybór przyciskiem set point . Należy przy tym pamiętać, że w celu zwiększenia dokładności określenia nowej geometrii narzędzia, podane punkty muszą się w sposób znaczny różnić od siebie. Istnieje możliwość przywrócenia zapisanej pozycji punktu poprzez naciśnięcie przycisku move here . Po zdefiniowaniu ostatniego punktu wyświetlony zostanie komunikat, czy geometria została zdefiniowana poprawnie. W celu zapisania geometrii narzędzia należy nacisnąć przycisk save . Zapisane geometrie można aktywować za pomocą przycisku

enable lub usunąć poprzez naciśnięcie przycisku delete

> Geometria narzędzia musi być definiowana przy włączonej domyślnej geometrii narzędzia. W przeciwnym razie obliczona geometria będzie nieprawidłowa.

9.3 Sterowanie narzędziami

Zastosowanie

Metody obsługi narzędzia są to sekwencje uruchamiania i zwalniania narzędzia. Zbudowane są z operacji dokonywanych na wyjściach I/O oraz komend oczekiwania. Używanie stworzonych metod w skrypcie robota polega na użyciu funkcji o nazwie "set_tool_state" dostępnej w bibliotece robota. Takie rozwiązanie posiada szereg zalet:

- zwiększenie czytelności skryptu robota,
- zwolnienie operatora z konieczności zapamiętywania sekwencji sterującej narzędziem,
- możliwość łatwej zmiany sposobu sterowania narzędziem.

Zarządzanie

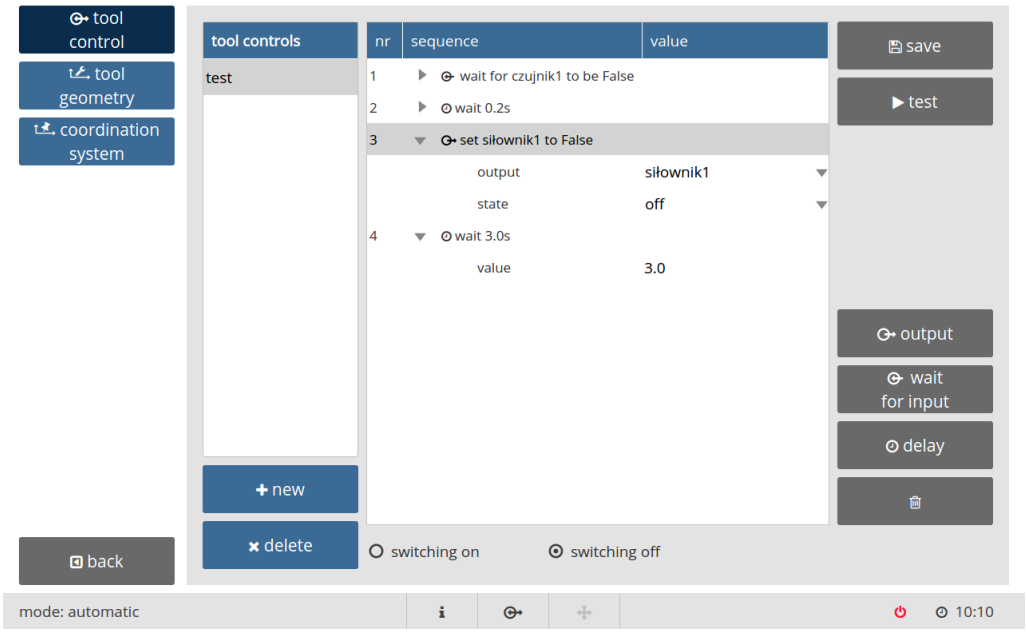
Zarządzanie zdefiniowanymi geometriami narzędzi odbywa się za pomocą karty w oknie definicji, dostępnej pod przyciskiem tool control . W celu dodania nowych metod sterowania narzędzia, należy nacisnąć przycisk new oraz wpisać unikalną nazwę narzędzia. Za pomocą przycisku delete możliwe jest usunięcie wcześniej stworzonej sekwencji.

Rysunek 9.84: Okno dodawania nowej sekwencji sterowania narzędziem.

Definiowanie sekwencji

Dla narzędzia możliwe jest stworzenie osobnych sekwencji dla jego włączania oraz wyłączania. Wybór sekwencji jest dokonywany za pomocą przycisków "switching on" dla sekwencji włączania oraz "switching off" dla sekwencji wyłączania. Podczas budowania sekwencji dostępne są trzy typy komend:

- output , ustawianie określonego stanu na dane wyjście I/O,
- wait for input , oczekiwanie na wystąpienie danego stanu na wybranym wejściu I/O,
- delay , oczekiwanie przez zadaną liczbę sekund.



Rysunek 9.85: Okno definicji sekwencji sterowania narzędziem.

Dodanie komendy do sekwencji wymaga zaznaczenia miejsca dodania oraz wciśnięcie przycisku komendy. Nowe komendy zawsze dodawane są nad aktualnie zaznaczoną linią. Zapisanie stworzonych sekwencji umożliwia przycisk save . Przycisk test pozwala na przetestowanie widocznej sekwencji.

Edycja

Edycja danej komendy dostępna jest poprzez jej rozwinięcie. Poszczególne rodzaje komend posiadają różne pola edycyjne:

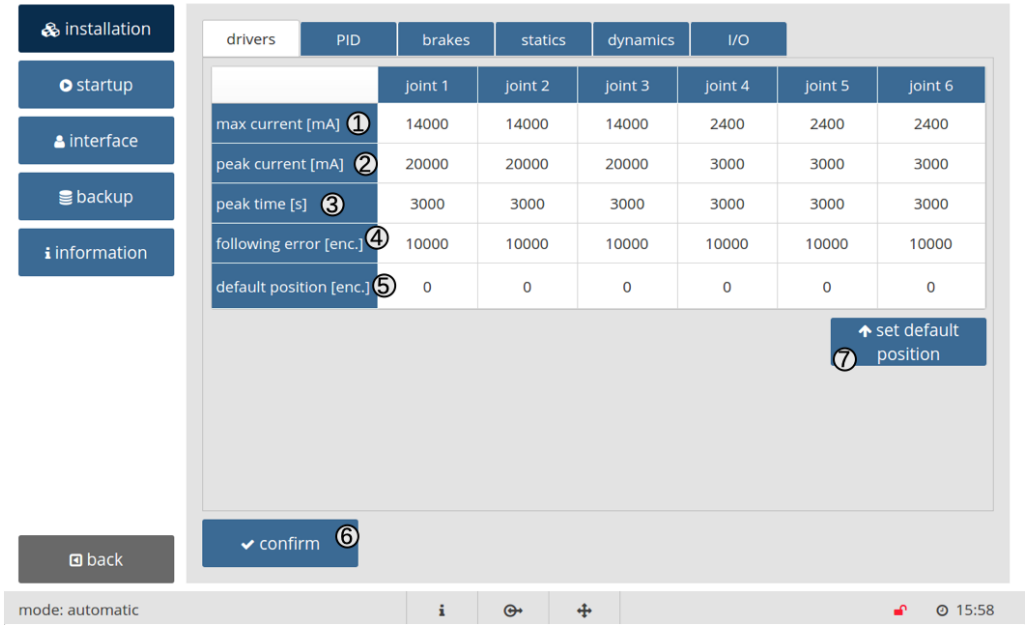
- Polecenie "wait for input" przyjmuje dwa argumenty "input", czyli nazwa wejścia oraz "state", czyli stan na jaki robot ma oczekiwać na danym wejściu.
- Komenda "output" przyjmuje parametry "output", czyli nazwa wejścia które chcemy wysterować oraz "state", czyli stan jaki chcemy podać na dane wyjście.
- Polecenie "wait" przyjmuje argument "value", czyli czas oczekiwania.

Aktualnie zaznaczoną komendę można usunąć poprzez naciśnięcie przycisku z ikoną kosza. Przycisk test umożliwia przetestowanie widocznej sekwencji narzędzia.

9.4 Ustawienia parametrów statycznych i dynamicznych ruchu

Serwonapędy

Każda z osi robota napędzana jest przez serwomotor połączony przekładnią z odpowiednim przegubem robota. Robot dostarczany jest do klienta po wstępnej kalibracji napędów. Może jednak zdarzyć się sytuacja, w której pewne parametry napędów będą wymagały kalibracji.



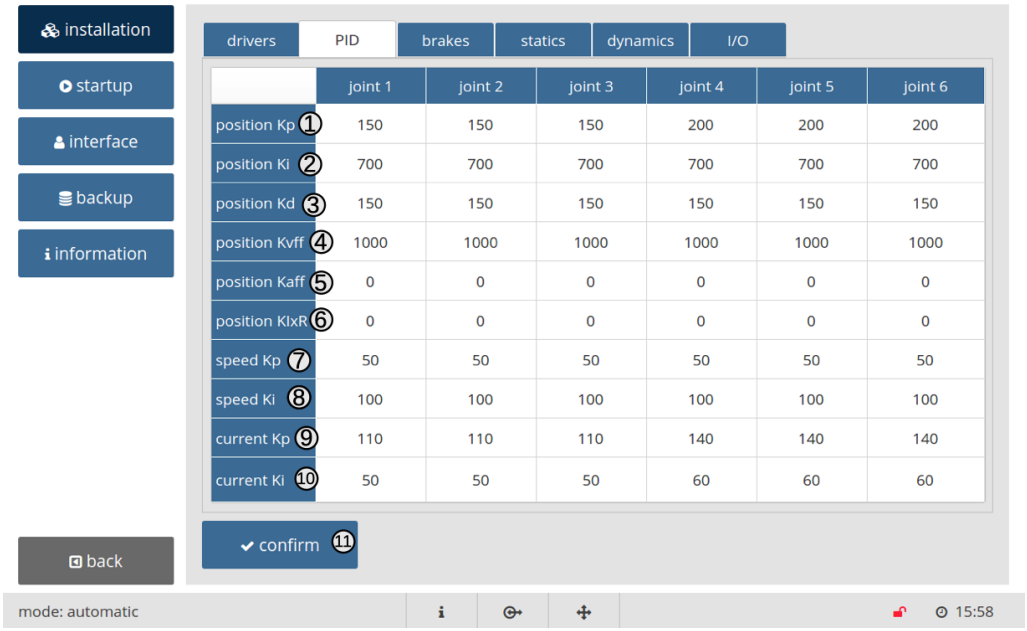
Rysunek 9.86: Okno konfiguracji podstawowych parametrów napędów

Zakładka drivers przedstawiona na rysunku 9.86 pozwala na konfiguracje:

1. Konfiguracja limitu prądu ciągłego.
2. Konfiguracja limitu prądu chwilowego oraz maksymalnego czasu jego trwania.
3. Konfiguracja błędu nadążania wyrażona w ilości impulsów enkodera.
4. Konfiguracja domyślnej pozycji wyrażona w ilości impulsów enkodera.
5. Zapis nastawianych parametrów.
6. Pobiera aktualne pozycje przegubów i ustawia jako domyślne.

PID

Zakładka PID przedstawiona na rysunku 9.87 pozwala na konfiguracje regulatorów PID dla każdego z napędów. Układ regulacji tworzy kaskadowy regulator PID pozycji, PI prędkości i PI prądu.



Rysunek 9.87: Okno konfiguracji nastaw i parametrów regulatorów PID

Poszczególne parametry regulatora to:

1. Nastawa członu proporcjonalnego regulatora pozycji.
2. Nastawa członu całkującego regulatora pozycji.
3. Nastawa członu różniczkującego regulatora pozycji.
4. Parametr definiuje sprzężenie zwrotne prędkości.
5. Parametr definiuje sprzężenie zwrotne przyspieszenia.
6. Parametr pozwala na kompensacje wpływu rezystancji uzwojeń i przewodów zasilania na pracę układu regulacji.
7. Nastawa członu proporcjonalnego regulatora prędkości.
8. Nastawa członu całkującego regulatora prędkości.
9. Nastawa członu proporcjonalnego regulatora prądu.
10. Nastawa członu całkującego regulatora prądu.
11. Zapis nastaw i parametrów regulatora.

Parametry statyczne

Statyczne parametry ruchu można ustawić na karcie static w sekcji installation w oknie ustawień. Poniżej przedstawiono okno parametrów statycznych dla układu napędowego manipulatora.

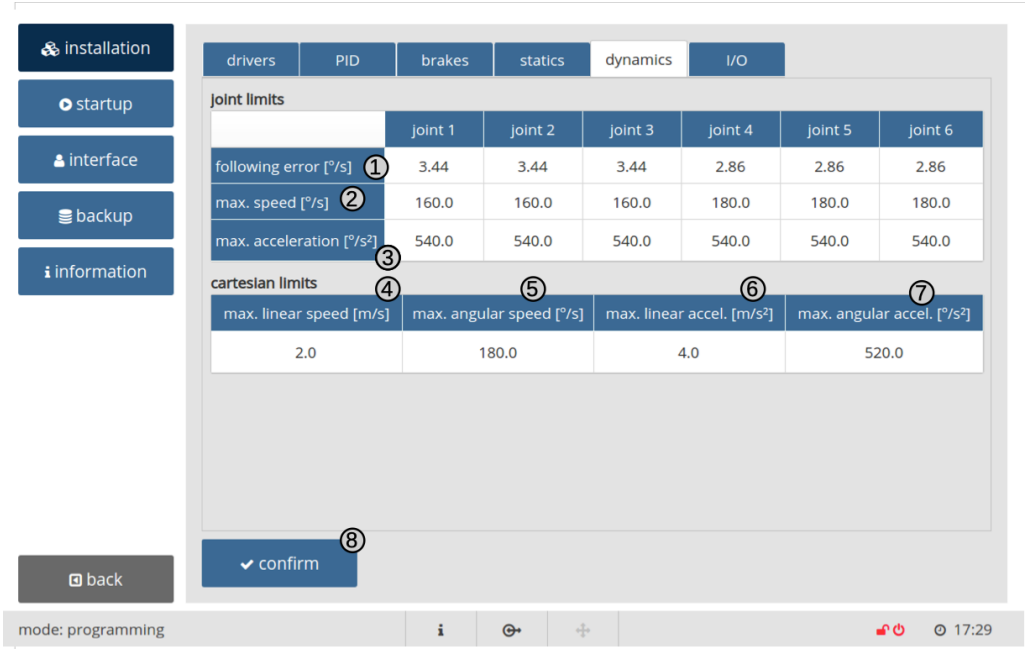
Rysunek 9.88: Okno ustawień parametrów statycznych

Na parametry statyczne układu napędowego robota składają się limit pozycji dla poszczególnych stawów (pozycje kątowe) oraz dla pozycji efektora robota (przemieszczenia liniowe). Okno ustawień parametrów statycznych pozwala:

1. Aktywować lub dezaktywować limit na wybranym przegubie robota.
2. Ustawić dolny limit pozycji na wybranym przegubie robota.
3. Ustawić górny limit pozycji na wybranym przegubie robota.
4. Aktywować lub dezaktywować limit dla wybranej osi kartezjańskiego układu współrzędnych (x,y,z).
5. Ustawić dolny limit pozycji dla wybranej osi kartezjańskiego układu współrzędnych (x,y,z).
6. Ustawić górny limit pozycji dla wybranej osi kartezjańskiego układu współrzędnych (x,y,z).
7. Zatwierdzić wprowadzone parametry.

Parametry dynamiczne

Parametry dynamiczne również dostępne są w menu ustawień w sekcji installation



Rysunek 9.89: Okno ustawień parametrów dynamicznych

Okno ustawień dynamicznych parametrów pozwala zdefiniować:

1. Wartość błędu nadążania na wybranym przegubie robota.
2. Maksymalną prędkość kątową na wybranym przegubie robota.
3. Maksymalne przyspieszenie kątowe na wybranym przegubie robota.
4. Maksymalną prędkość liniową efektora.
5. Maksymalną prędkość kątową rotacji efektora.
6. Maksymalne przyspieszenie liniowe efektora.
7. Maksymalne przyspieszenie kątowe rotacji efektora.

Przycisk confirm służy do zatwierdzenia wprowadzonych parametrów.

9.5 Automatyczne uruchamianie

Zastosowanie

W przypadku, gdy robot wykonuje na swoim stanowisku jeden program, wygodnym dla obsługi stanowiska rozwiązaniem jest ustawienie automatycznego uruchamiania napędów oraz programu po włączeniu zasilania. Jeśli na stanowisku istnieje sterownik nadrzędny, uruchomienie programu robota można uzależnić od stanu wejścia I/O.

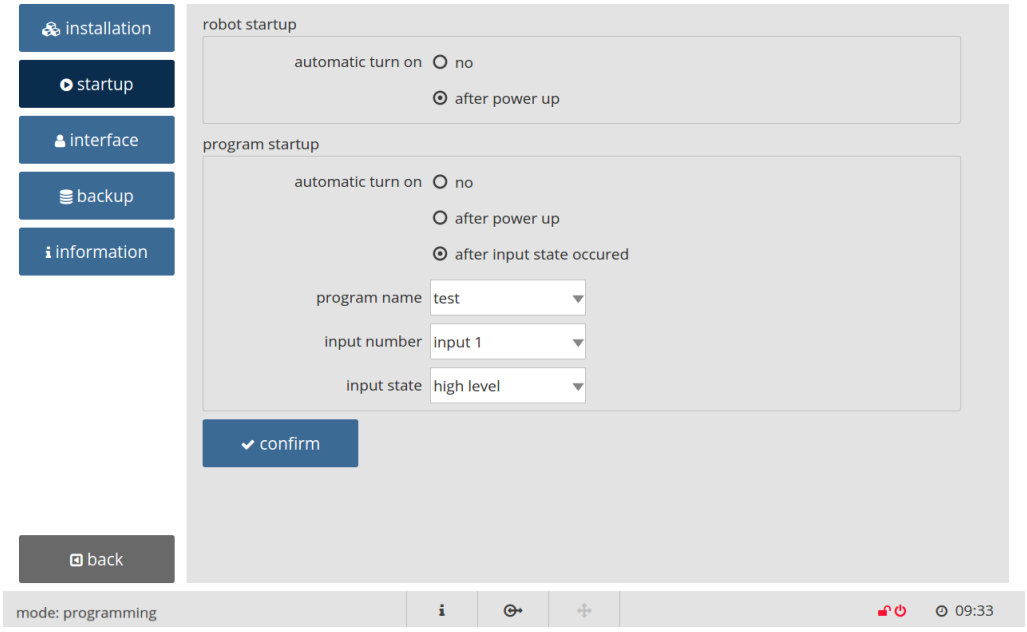
Auto-uruchamianie robota

Sekcja "robot startup" służy do skonfigurowania uruchamiania napędów robota automatycznie po włączeniu zasilania. Ustawienie opcji "after power up" włącza opcję auto-uruchamiania napędów, natomiast ustawienie opcji "no" powoduje standardowe działanie.

Auto-uruchamianie programu

Sekcja "program startup" służy do skonfigurowania uruchamiania programu robota po pojawieniu się określonej konfiguracji wejścia I/O, lub po włączeniu zasilania. Poszczególne opcje to:

1. standardowy tryb- bez auto-uruchamiania,
2. uruchamianie programu po włączeniu zasilania,
3. nazwa uruchamianego programu,
4. wejście skonfigurowane do uruchamiania wskazanego programu robota (aktywne tylko w przypadku aktywnej opcji 3),
5. stan wejścia uruchamiający program (aktywne tylko w przypadku aktywnej opcji 3).



Rysunek 9.90: Karta ustawień auto-uruchamiania napędów oraz programu robota.

10 Serwis

10.1 Diagnostyka

Cel

Zadaniem tego rozdziału jest wskazanie sposobu postępowania w razie wystąpienia problemów z użytkowaniem produktu. Jeżeli źródło zaistniałego problemu nie pokrywa się z żadnym z poniżej wymienionych przypadków, zaleca się kontakt z serwisem Easy Robots.

Rozpoznanie

Opis problemu	Rozwiązanie
Stanowisko nie włącza się	Sprawdzić zasilanie szafy sterowniczej oraz pozycję przełącznika zasilania
Błąd napędu (czerwone podświetlenie w oknie głównym)	Wyłączyć i włączyć robota
Panel się nie uruchamia (zbyt długo wyświetla się ikona oczekiwania (10.1))	Sprawdzić podłączenie robota do szafy sterowniczej oraz połączenia przewodów komunikacyjnych wewnątrz szafy (pomiędzy modułem wejść/wyjść a gniazdem przyłączeniowym)

Rysunek 10.1: Ikona oczekiwania na uruchomienie panelu

10.2 Pomoc techniczna

Wstęp

Niniejsza dokumentacja zawiera informacje na temat eksploatacji i obsługi produktu oraz usuwania zakłóceń. W przypadku dalszych pytań, zapraszamy do kontaktu.

Informacje

W celu wysłania pytania serwisowego potrzebne są następujące informacje:

- Typ i numer seryjny robota
- Wersja oprogramowania EScontrol
- Archiwum oprogramowania

- Opis problemu, czas, częstotliwość występowania usterki

Dane kontaktowe

Easy Robots Sp. z o. o.
ul. Gdyńska 32
26-600 Radom
Polska
tel.: 48 377 99 99
email: service@easyrobots.eu