Wymagania:

1. Sterowanie pracą osi robota.
2. Konfigurowanie parametrów osi.
3. Komunikacje przez bluetooth.

Ad.1

Aplikacja umożliwia uruchomienie jednej osi w danym momencie (MOVE) lub kilku osi na raz (TASK).

Profil ruchu tworzy się na ekranie „Task Screen” na którym znajdują się odpowiednie kontrolki umożliwiające ustalić pozycje jakie mają zająć osie robota. Aby możliwy był ruch osi należy ją załączyć przyciskiem z prawej strony ekranu, załączona oś pojawi się w tabeli w dolnej części ekranu. Po wybraniu żądanych osi trzeba je dodać do listy ruchów poprzez kliknięcie przycisku „Dodaj do listy”. Jeśli wybrana zostanie tylko jedna oś utworzony zostanie tzw. „Ruch” (MOVE), natomiast w przypadku większej liczby osi utworzone zostanie „Zadanie” (TASK). Dodane do listy „Ruchy” oraz „Zadania” tworzą sekwencję zadań (JOB) którą wysyła się do robota. Wysyłanie sekwencji zadań możliwe jest na ekranie „Job Screen” na który dostać się można przez kliknięcie przycisku „Pokaż listę”. Po wysłaniu sekwencji ruchów odblokowany zostaje przycisk „Uruchom” który załączy wykonywanie zadań przez robota.

Ad.2

Konfigurowanie parametrów osi wykonuje się na ekranie „Config Screen”. Znajdują się tam suwaki którymi ustala się skrajne pozycje osi robota. Wysłanie konfiguracji do sterownika odbywa się poprzez kliknięcie przycisku „Wyślij ustawienia”. Możliwy jest również odczyt aktualnych ustawień robota klikając przycisk „Pobierz ustawienia”.

Ad.3

1. Wszystkie dane wysyłane są z aplikacji do sterownika robota poprzez bluetooth.
2. Telefon jest urządzeniem „Master” a sterownik robota „Slave”, oznacza to, że komunikacja inicjowana jest przez telefon. Wyjątkiem od tej reguły jest sytuacja gdy robot zakończy działanie, wtedy to sterownik bez żądania wyśle informację o tym fakcie.
3. W praktyce wygląda to tak że telefon wysyła żądanie a sterownik odpowiada.
4. Dane wymieniane między urządzeniami zorganizowane są w „Ramki”.
5. Aplikacja może wysłać kilka typów ramek, natomiast sterownik zawsze odpowiada wysyłając ramkę zawierającą status sterownika.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ogólny kształt ramki danych | | | | | | | |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |

**Ramki wysyłane do sterownika:**

1. Kod początku.
   1. „SF”.
2. Typ ramki.
   1. „STATUS” – żądanie wykonania akcji związanych z działaniem sterownika.
   2. „MOVE” – żądanie dodania ruchu jednej osi do listy ruchów w sterowniku.
   3. „TASK” – żądanie dodania ruchu osi do jednoczesnego uruchomienia.
   4. „ETASK” – żądanie dodania ostatniego ruchu osi do jednoczesnego uruchomienia.
   5. „JOB” – żądanie uruchomienia robota.
   6. „EJOB” – żądanie zatrzymania robota
3. Dane właściwe dla typu ramki.
4. Kod końca.
   1. „EF”

**Ramka statusu może zawierać następujące dane:**

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 1 | 0 | 0 | 0 | 0 | EF |

Jest to przygotowanie sterownika do odbioru danych ruchu.

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 1 | 0 | 0 | 0 | 0 | EF |

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 2 | 0 | 0 | 0 | 0 | EF |

Żądanie podłączenia sterownika.

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 2 | 0 | 0 | 0 | 0 | EF |

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 3 | 0 | 0 | 0 | 0 | EF |

Żądanie odłączenia sterownika.

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 3 | 0 | 0 | 0 | 0 | EF |

W powyższych przypadkach sterownik odpowiada taką samą ramką jaką otrzymał, pod warunkiem, że wszystko poszło dobrze, w przeciwnym wypadku „Dana1” będzie zawierać wartość 99.

Sytuacja wyjątkowa w której to sterownik samodzielnie inicjuje komunikację ma miejsce gdy robot zakończy działanie i wtedy sterownik wysyła ramkę jak poniżej.

Ramka informująca o zakończeniu pracy:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 6 | 0 | 0 | 0 | 0 | EF |

**Ramki typu „MOVE”, „TASK” oraz „ETASK” zawierają parametry ruchu robota i przyjmują formę jak poniżej:**

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | MOVE | Nazwa osi | Kąt osi | Prędkość osi | Kierunek osi | ------- | EF |

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 1 | 0 | 0 | 0 | 0 | EF |

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | TASK | Nazwa osi | Kąt osi | Prędkość osi | Kierunek osi | ------- | EF |

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 1 | 0 | 0 | 0 | 0 | EF |

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | ETASK | Nazwa osi | Kąt osi | Prędkość osi | Kierunek osi | ------- | EF |

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 1 | 0 | 0 | 0 | 0 | EF |

Różnica między tymi ramkami polega na odmiennym ich interpretowaniu przez sterownik robota.

Ramka MOVE powoduje, że do listy ruchów w sterowniku dodany zostanie ruch wskazanej osi który zostanie wykonany odrębnie. Typ TASK powoduje, że parametry kolejnych osi trafiają do listy jako ruchy które mają być wykonane jednocześnie. Natomiast ETASK informuje sterownik, że podana oś jest ostatnią która trafia do listy ruchów wykonywanych razem z poprzednimi. Ostatnie dwa typy ramek JOB oraz EJOB służą kolejno do uruchamiania i zatrzymywania robota.

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | JOB | 0 | 0 | 0 | 0 | 0 | EF |

Żądanie załączenia robota.

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 4 | 0 | 0 | 0 | 0 | EF |

Ramka wysłana:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 5 | 0 | 0 | 0 | 0 | EF |

Żądanie zatrzymania robota.

Odpowiedź sterownika:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Kod początku | Typ ramki | Dane1 | Dane2 | Dane3 | Dane4 | Dane5 | Kod końca |
| SF | STATUS | 3 | 0 | 0 | 0 | 0 | EF |

Moduły

1. AxisParameters
2. DataGridView
3. ImageButton
4. USART
5. AxisConfig
6. Main

**AxisParameters**

Klasa AxisParameters służy do gromadzenia informacji na temat pozycji oraz zakresu ruchu osi robota. Zawiera również metody, które wykonują wymagane obliczenia do określenia tych parametrów.

1. Pola klasy:
   1. CW\_Travel – zakres ruchu w prawo.
   2. CCW\_Travel – zakres ruchu w lewo.
   3. Actualposition – pozycja aktualna.
   4. TravelToGo – Droga do przebycia (zadany kąt obrotu).
   5. MaxTravel – Zakres pracy osi (0 – MaxTravel).
   6. Direction – Kierunek obrotu.
2. Metody
   1. Initialize – Konstruktor klasy.
   2. SetParameters – Metoda ustawia pola na wartości podane w konstruktorze klasy.
   3. RangeCalculate – Obliczanie, ile może się dana oś obrócić w każdą stroną, czyli obliczenie wartości dla CW\_Travel i CCW\_Travel. Obliczenia wykonywane są na podstawie wartości suwaka pozycji, która przekazywana jest do metody przez parametr Value.
   4. TravelToGoCalculate – Metoda oblicza, kąt o jaki ma obrócić się oś. Aby obliczenia mogły się wykonać należy podać wartość suwaka osi oraz typ osi. Wymagane wartości trafiają do metody za pośrednictwem parametrów: Value – wartość suwaka oraz RotaryAxis – czy oś jest obrotowa?
   5. ActualPositionCalculate – Działanie tej metody sprowadza się do obliczenia pozycji aktualnej osi robota.

**DataGridView**

DataGridView jest to klasa, która implementuje kontrolkę o wyglądzie tabeli i daje możliwość dynamicznego dodawania do niej kolejnych wierszy. Wykorzystywana jest na ekranie TaskScreen do gromadzenia parametrów osi, które mają zostać dodane do sekwencji zadań.

1. Pola klasy:
   1. Frontend:
      1. RowBackground – Kolor tła komórek tabelki.
      2. GapColor – Kolor linii tabeli.
      3. ItemTextColor – Kolor napisów w tabeli.
      4. HeaderBackground – Tło w nagłówkach tabeli.
      5. HeaderItemColor – Kolor napisów w nagłówkach.
   2. Backend:
      1. HeadersTop – Położenie nagłówków względem górnej krawędzi tabeli.
      2. ColumnWidth – Szerokość kolumny.
      3. ColumnGap – Odstęp między kolumnami.
      4. ColumnsCount – Ilość kolumn w tabeli.
      5. Rows – Dane zawarte w tabeli.
2. Metody:
   1. Initialize – Konstruktor klasy.
   2. DesignerCreateView – Metoda generująca elementy wizualne kontrolki, wywoływana jest przez „Designer”.
   3. GetBase – Metoda zwraca obiekt typu Panel, w którym znajduje się kontrolka.
   4. InsertHeaders – Wstawienie nagłówków tabeli.
   5. InsertRow – Metoda dodaje wiersz, do tabeli którego komórki wypełnione zostają danymi pochodzącymi z listy przekazanej za pośrednictwem parametru RowData.
   6. DeleteRow – Usunięcie wiersza, który zawiera nazwę osi podaną w parametrze Axis.
   7. GetItem – Pobranie elementu tabeli znajdującego się w miejscu określonym parametrami: Axis – nazwa osi oraz Column – nazwa kolumny.
   8. SetItemValue – Wstawienie do tabeli wartości z parametru Value w miejsce określone parametrami: Axis – nazwa osi oraz Column – nazwa kolumny.
   9. IsRowExist – Metoda sprawdza czy w tabeli znajduje się wiersz o nazwie podanej w parametrze Name. Jeśli wiersz istnieje to metoda zwraca wartość True w przeciwnym wypadku zwróci False.
   10. GetTable – Metoda zwraca wszystkie dane zawarte w tabeli.
3. Eventy:
   1. BlendChecked\_CheckedChange – Jest to metoda obsługi zdarzenia zaznaczenia/odznaczenia kontrolki CheckBox znajdującej się w ostatniej kolumnie tabeli, parametr Checked określa stan kontrolki, zaznaczona/odznaczona (True/False).

**ImageButton**

ImageButton to klasa implementująca przycisk z obrazkiem.

1. Pola klasy:
   1. Frontend:
      1. FrameColor – Kolor obramowania przycisku.
      2. BackgroundColor – Kolor tła.
      3. ButtonText – Napis w przycisku.
      4. ButtonTextPosX – Pozycja tekstu, współrzędna X.
      5. ButtonTextPosY – Pozycja teksty, współrzędna Y.
      6. ImageWidth – Szerokość obrazka w przycisku.
      7. ImageHeight – Wysokość obrazka w przycisku.
   2. Backend:
      1. ButtonPanel – Panel, w którym znajdują się elementy przycisku.
      2. Image – Obrazek umieszczany w przycisku
      3. ButtonLabel – Labelka do której wstawiany jest napis podany w polu ButtonText.
2. Metody:
   1. Initialize – Konstruktor klasy
   2. DesignerCreateView – Metoda generująca elementy wizualne kontrolki, wywoływana jest przez „Designer”.
   3. SetImage – Wstawienie obrazka do przycisku.
   4. SetText – Wstawienie tekstu.
   5. SetFrameColor – Ustawienie koloru ramki.
   6. SetBackground – Ustawienie koloru tła.
   7. Disable – Zablokowanie przycisku.
   8. Enable – Odblokowanie przycisku.
   9. GetBase – Pobranie obiektu typu Panel, na którym utworzona jest kontrolka.
3. Eventy:
   1. ButtonImageEvent\_Click – Metoda obsługi zdarzenia kliknięcia.

**USART**

USART jest modułem, który jest odpowiedzialny za komunikację ze sterownikiem robota. Aby zapewnić asynchroniczny tryb komunikacji dodatkowo moduł korzysta z klasy AsyncStreamsText w której zaimplementowane są metody realizujące asynchroniczne odbieranie i wysyłanie danych.

1. Pola klasy:
   1. AvailableDevices – Lista odnalezionych urządzeń.
   2. Adapter – Obiekt modułu bluetooth w telefonie.
   3. CommPort – Obiekt reprezentujący port szeregowy.
   4. AsyncText – Obiekt klasy do obsługi komunikacji asynchronicznej.
   5. ReceivedFrame – Ramka danych odebranych od sterownika.
   6. IsAdapterConnected – Zmienna informuje czy zewnętrzny adapter bluetooth został podłączony.
2. Metody:
   1. Initialize – Konstruktor kalsy.
   2. AskWindow – Metoda wyświetla okienko z pytaniem, którego treść podana jest w parametrze Question, wartość zwracana z metody to obiekt wyświetlonego okienka.
   3. ErrorWindow – Metoda wyświetla okienko wskazujące błąd podany w parametrze Message, wartość zwracana z metody to obiekt wyświetlonego okienka.
   4. InfoWindow – Metoda wyświetla okno z informacją podaną w parametrze Message, wartość zwracana z metody to obiekt wyświetlonego okienka.
   5. WarningWindow – Metoda wyświetla okno ostrzegawcze z treścią podaną w parametrze Message, wartość zwracana z metody to obiekt wyświetlonego okienka.
   6. Run – Uruchomienie wewnętrznego modułu bluetooth.
   7. Serch – Uruchomienie wyszukiwania urządzeń.
   8. Connect – Podłączenie do zewnętrznego adaptera. Parametr DeviceName określa nazwę urządzenia, do którego można się podłączyć.
   9. Send – Metoda wysyła przez bluetooth ciąg znaków podany w parametrze Message.
   10. SendStatusFrame – Wysłanie ramki danych z żądaniem ustawienia przez sterownik statusu, którego wartość podana jest w parametrze StatusValue.
   11. SendRunFrame – Wysłanie komendy uruchomienia robota.
   12. SendStopFrame – Wysłanie komendy zatrzymania robota.
   13. SendWorkFrame – Wysłanie ramki z danymi ruchu, które przekazane są za pomocą parametru FrameData. Natomiast parametr FrameType określa czy jest to ramka typu: MOVE, TASK czy ETASK.
   14. GetData – Metoda uzupełnia ramkę odbiorczą danymi przekazanymi przez parametr DataText, dane zawarte w tym parametrze to tekst odebrany z bluetooth.
   15. Adapter\_DiscoveryStarted – Procedura obsługi zdarzenia rozpoczęcia przez adapter bluetooth, wyszukiwania dostępnych urządzeń.
   16. Adapter\_DeviceFound – Procedura obsługi zdarzenia wykrycia urządzenia.
   17. Adapter\_DiscoveryFinished – Procedura obsługi zdarzenia zakończenia wyszukiwania.
   18. Adapter\_StateChanged – Procedura obsługi zdarzenia zmiany stanu adaptera.
   19. CommPort\_Connected – Procedura obsługi zdarzenia nawiązania połączenia.
   20. AsyncText\_Terminated – Procedura obsługi zdarzenia zerwania komunikacji.
   21. AsyncText\_NewText – Procedura obsługi zdarzenia odebrania tekstu.
3. Eventy:
   1. DataAck – Potwierdzenie przez sterownik odebrania danych.
   2. RunAck – Potwierdzenie uruchomienia robota.
   3. StopAck – Potwierdzenie zatrzymania robota.
   4. WorkDone – Potwierdzenie wykonania zadania.
   5. DriverNotReady – Zdarzenie braku odpowiedzi ze sterownika.
   6. AdapterConnected – Zdarzenie podłączenia zewnętrznego adaptera bluetooth.
   7. DriverConnected – Potwierdzenie podłączenia sterownika.
   8. DriverDisConnected – Potwierdzenie odłączenia sterownika.

**Main**

Główny moduł aplikacji, wiążę on ze sobą pozostałe moduły oraz realizuje komunikację z użytkownikiem.

1. Zmienne współdzielone z innymi modułami:
   1. Bluetooth – Obiekt modułu USART odpowiedzialny za komunikację.
   2. AutoSwitchScreenTimer – Czasówka służy do automatycznego przełączenia ekranu SplashSceen na TaskScreen.
   3. TimeoutConnectionTimer – Czasówka sprawdza czy nie został przekroczony czas na odpowiedź sterownika.
2. Zmienne globalne:
   1. Designer vars – Obiekty reprezentujące elementy interfejsu użytkownika.
   2. SendRowIndex – Indeks wysyłanego wiersza tabeli.
   3. SendTaskSequenceIndex – Indeks wysłanej sekwencji zadań.
   4. OldBlend – Wartośćłączenia ruchów w poprzednim zadaniu.
   5. IsTaskSending – Czy trwa wysyłanie zadania?
   6. IsTaskSequenceSending – Czy trwa wysyłanie sekwencji zadań?
   7. IsRunCommandSending – Czy wysłana została komenda uruchomienia?
   8. IsStopCommandSending – Czy została wysłania komenda zatrzymania?
   9. IsWorkInProgres – Czy robot jest w trakcie pracy?
   10. TaskSequence – Wirtualna tabela zadań dla robota.
   11. TaskSequenceIndex – Indeks tabeli zadań.
   12. ParametersAxisA – Aktualne parametry ruchu osi A.
   13. ParametersAxisB – Aktualne parametry ruchu osi B.
   14. ParametersAxisC – Aktualne parametry ruchu osi C.
   15. ParametersAxisZ – Aktualne parametry ruchu osi Z.
   16. ParametersAxisG – Aktualne parametry ruchu osi G.
   17. ParametersAxisT – Aktualne parametry ruchu osi T.
3. Procedury: