Artykuł Autorski, XI Forum Inżynierskiego ProCAx cz. II,
Kraków, 16-18 października 2012 r.

Trzmiel Adam, mgr inż.
email: trzmiel.adam@gmail.com
Góra Marta, dr inż.
email: mgora@m6.mech.pk.edu.pl
Politechnika Krakowska, Wydział Mechaniczny

MODELOWANIE CHODU ROBOTA SZEŚCIONOŻNEGO W ŚRODOWISKU CATIA v5 I EMBARCADERO

Streszczenie: Przygotowano metody do analizy chodu robota sześcionożnego, które mogą zostać wykorzystane przy projektowaniu maszyn kroczących. W tym celu zamodelowano makietę cyfrową robota kroczącego w środowisku CATIA v5 oraz przygotowano autorską aplikację w środowisku Embarcadero umożliwiającą modelowanie chodu robota kroczącego, obserwację przemieszczenia poszczególnych kończyn i korpusu za pomocą trójwymiarowego modelu oraz dobór parametrów konstrukcyjnych, przy których chód maszyny spełnia postawione kryteria.

GAIT SIMULATION OF HEXAPODE ROBOT IN CATIA v5 AND EMBARCADERO ENVIROMENT

Abstract: Methods for gait analyses of hexapod robot, useful in designing process of stepping machines, are presented. A virtual prototype of the considered robot with six legs was prepared in CATIA v5. Own application formulated in Embarcadero environment enables the robot gait modeling, observing 3-dimensional displacements of the legs and the body. The analysis goal was to find some design parameters improving defined criteria for the robot gait.

1. Wprowadzenie

Z wielu typów robotów kroczących szczególną pozycję zajmują konstrukcje o sześciu odnóżach. Inspiracje konstrukcyjne czerpie się tu głównie z budowy i funkcjonowania insektów lub pajęczaków. Cechują się one redundancją kończyn, z której wynikają szczególne zalety tego typu mechanizmów, jak również pewne wady. Większa liczba punktów podparcia prowadzi do lepszej stabilności, umożliwia implementację różnych typów chodu oraz dostosowanie sposobu lokomocji w zależności od zmieniającego się terenu. Sposób poruszania określić można jako stabilny statycznie, bazujący na dobrze poznanych i stosunkowo łatwych do naśladowania zasadach ruchu owadów. Powoduje to uproszczenie matematycznego opisu ruchu (nie jest konieczne uwzględnianie zależności dynamicznych). Konieczny jest jednak bardziej rozbudowany system sterowania umożliwiający kontrolowanie większej ilości napędów, jak również przetwarzanie znacznych ilości danych sensorycznych związanych z każdym odnóżem. W konsekwencji koszt konstrukcji wzrasta, a budowa mechaniczna bardziej staje się złożona. Dodatkowe kończyny pozwalają jednak, na kontynuację pracy systemu, np. w przypadku uszkodzenia pewnej ilości napędów lub całych kończyn, co w dużym stopniu zwiększa niezawodność tego typu mechanizmów. Jeżeli wyposażyć odnóża w wystarczającą liczbę stopni swobody, posłużyć mogą ponadto do zadań manipulacyjnych, czy też technologicznych.

2. Analiza chodu

Jednym z najważniejszych aspektów związanych z projektowaniem systemów sterowania robotami kroczącymi jest implementacja rozwiązań umożliwiających zachowanie stabilności urządzenia podczas lokomocji. Dla układów regulacji automatycznej stabilność określić można ogólnie jako własność systemu umożliwiająca mu powrót do stanu równowagi w przypadku, gdy został on z tego stanu wytrącony. Dla maszyn kroczących oznacza to, iż zakłócenia zawierające się w określonym przedziale zmienności nie mogą powodować większych niż dopuszczalne odchyleń trajektorii rzeczywistej od zadanej grożąc tym samym przewróceniem się maszyny. Kryterium to nakłada na system sterowania dodatkowe zadanie polegające na każdorazowym sprawdzeniu warunków stabilności przed wykonaniem ruchu oraz odpowiednie dobieranie punktów podparcia i orientacji korpusu w celu unikania sytuacji, w której dojść może do destabilizacji układu. W celu opisu stabilności maszyn kroczących wprowadza się fundamentalne pojęcia [2, 8], takie jak:

- Stan podparcia (ang. support state) maszyny kroczącej o n odnóżach stanowi wektor binarny $\boldsymbol{k}(t)$ taki, że dla każdej chwili t, $k_i(t) = 1$, jeżeli i-ta noga znajduje się w stanie kontaktu z podłożem oraz $k_i(t) = 0$ w przypadku przeciwnym,
- Wielokąt podparcia (ang. support pattern) jest wielokątem wypukłym rozpiętym na punktach śladowych nóg będących projekcją punktów związanych z końcem odnóży, dla których $k_i(t) = 1$, wzdłuż osi oddziaływania grawitacji na płaszczyznę do niej prostopadłą.

Dodatkowo zakłada się, iż masy poszczególnych odnóży są niewielkie w porównaniu z masą korpusu. Uproszczenie to pozwala przyjąć, iż środek ciężkości maszyny nie przemieszcza się zbytnio w wyniku ruchu poszczególnych kończyn, a jego współrzędne wyrażone w układzie związanym z korpusem pozostają stałe. Założenie to nie sprawdza się w przypadku maszyn cięższych, gdzie ruch pojedynczej nogi wpływa na położenie środka ciężkości maszyny, a tym samym na stabilność układu. Położenie to musi być wówczas obliczane w każdej kolejnej iteracji. Pozycję środka ciężkości $\boldsymbol{q}(t)$ wyrażoną w układzie związanym z korpusem, określić można jako sumę iloczynów mas wszystkich członów mechanizmu i współrzędnych ich środków ciężkości podzieloną przez masę całkowitą urządzenia.

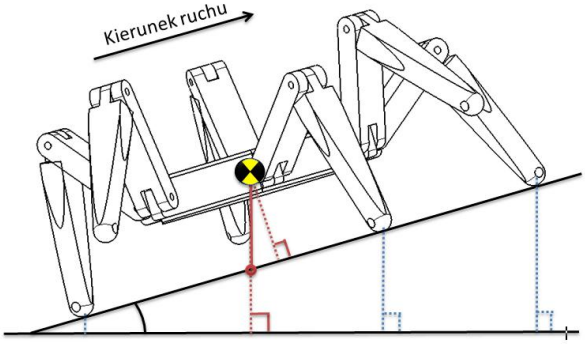
Mając na uwadze powyższe definicje, maszyny kroczące można podzielić biorąc pod uwagę rodzaj stabilności na:

- stabilne statycznie,
- stabilne quasi – statycznie,
- stabilnie dynamicznie.

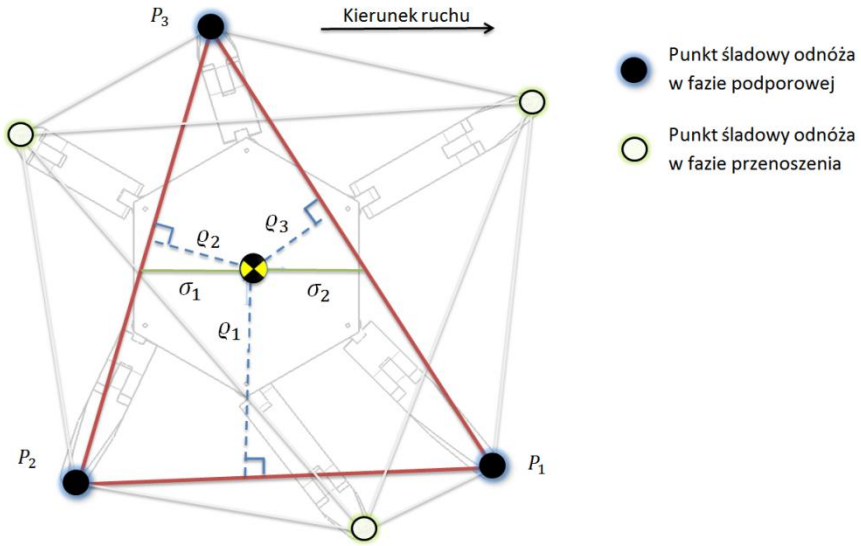
Ruch stabilny statycznie jest sposobem chodu występującym powszechnie w naturze, w szczególności w przypadku owadów. Mechanizmy budowane na ich podobieństwo cechują się dużą liczbą aktywnych stopni swobody związanych z dużą liczbą odnóży. Wynika to z faktu, iż w celu utworzenia wielokąta podparcia, koniecznego dla zaistnienia statycznej stabilności, co najmniej trzy odnóża muszą znajdować się w kontakcie z podłożem. W efekcie mechanizmy statycznie stabilne posiadają co najmniej cztery, a najczęściej sześć lub więcej odnóży. Jeżeli znane jest położenie środka ciężkości maszyny kroczącej $\boldsymbol{q}(t)$ oraz jej konfiguracja kinematyczna określająca położenia P_i punktów kontaktu odnóży z podłożem, dla których $k_i(t) = 1$, to maszyna krocząca po płaskim terenie jest statycznie stabilna w chwili t wtedy i tylko wtedy, gdy pionowy rzut $\boldsymbol{q}(t)$ jej zamrożonej konfiguracji na płaszczyznę poziomą leży wewnątrz wielokąta podparcia [2, 8]. Warunkiem zachowania statycznie stabilnej pozycji w przypadku działania wymuszeń jest, aby rzut środka ciężkości $\boldsymbol{q}(t)$ na płaszczyznę ruchu

nie znajdował się bliżej krawędzi budujących wielokąt podparcia niż zadana odległość ϱ, zwana marginesem stabilności [2, 8]. Warunkiem zachowania statycznie stabilnej pozycji w przypadku ruchu postępowego maszyny kroczącej wzdłuż kierunku określonego wektorem prędkości jej środka ciężkości $\boldsymbol{v}(t)$ jest, aby odległość od punktu stanowiącego rzut środka ciężkości $\boldsymbol{q}(t)$ na płaszczyznę ruchu do krawędzi budujących wielokąt podparcia, mierzona wzdłuż $\boldsymbol{v}(t)$, nie była mniejsza niż zadana odległość σ, zwana marginesem stabilności wzdłużnej [2, 8].

Modelując chód statycznie stabilny dąży się więc do tego, aby w czasie ruchu zachowany był zadany zapas stabilności statycznej umożliwiający zachowanie stateczności układu w przypadku wystąpienia nieprzewidzianych wymuszeń. Im większy margines stabilności, tym lepsze warunki stateczności całego systemu. Na rys. 1-2 umieszczono graficzną reprezentację powyższych definicji.



Rys. 1 Rzut środka ciężkości i punktów podporowych na płaszczyznę horyzontalną w ruchu po nachylonym terenie



Rys. 2 Wielokąt podparcia oraz margines stabilności $(\min(\varrho_1, \varrho_2, \varrho_3))$ i margines stabilności wzdłużnej $(\min(\sigma_1, \sigma_2))$

Zauważyć należy, iż o ile kryterium marginesu stabilności statycznej sprawdza się dobrze przy rozpatrywaniu ruchu maszyny kroczącej po płaskim terenie, to jednak zawodzi w przypadku podłoża o bardziej zróżnicowanej strukturze. Powodem jest brak uwzględnienia konkretnej konfiguracji mechanizmu takiej jak np. postura korpusu, jej wysokość nad powierzchnią ruchu, czy też wysokości, na jakiej podparcie znajdują odpowiednie kończyny.

3. Symulator maszyny kroczącej

Analiza strukturalna i kinematyczna oraz chodu mają wpływ na ogólny kształt modelu maszyny kroczącej. Przystępując do projektowania konstrukcji maszyny kroczącej problematyczne stać się mogę kwestie doboru wielkości charakterystycznych jak, np. wymiary geometryczne korpusu, sposób rozmieszczania odnóży, długości poszczególnych segmentów tworzących nogi urządzenia, czy też sposób technicznej realizacji połączeń ruchomych. Współcześnie coraz częściej budowę fizycznych prototypów poprzedza się wykonaniem ich wirtualnych odpowiedników, nazywanych makietami cyfrowymi (ang. digital mock-up). Umożliwia to przeprowadzenie za jej pomocą szeregu symulacji pozwalających nie tylko na opracowanie najlepszych zbiorów cech konstrukcyjnych, ale również na przeprowadzenie obszernych analiz numerycznych dotyczących własności kinematycznych i dynamicznych prototypu. Proces ten nazywany jest wirtualnym prototypowaniem (ang. virtual prototyping).

Wyróżnić można dwa główne elementy budujące symulator:

1) makieta cyfrowa – cyfrowy model maszyny kroczącej wizualizowany w środowisku trójwymiarowym,
2) program sterujący – autorska aplikacja posiadająca interfejs graficzny umożliwiający sterowanie robotem w czasie rzeczywistym.

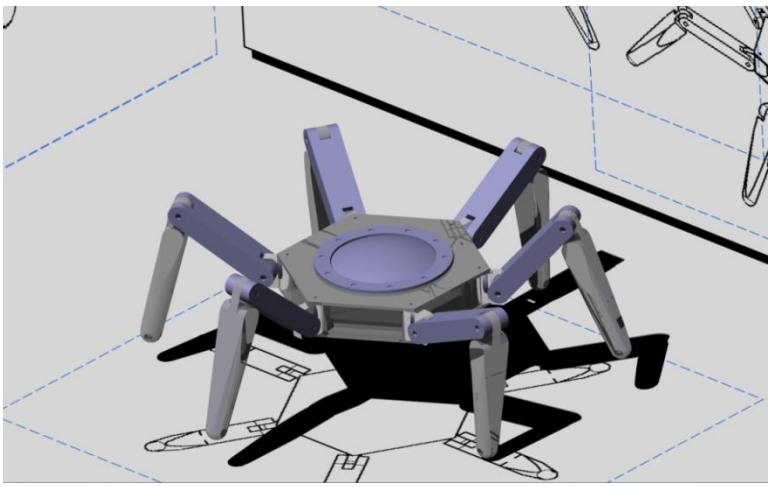
3.1. Cyfrowa makieta robota kroczącego

Do modelowania oraz symulacji kinematycznych makiety robota kroczącego wykorzystano środowisko CATIA v5 jest to powszechnie stosowana aplikacja CAx, która znalazła szerokie zastosowanie w przemyśle lotniczym, samochodowym, maszynowym, elektrotechnicznym i innych.

Budowanie makiety robota kroczącego rozpoczyna się w modułach przeznaczonych do modelowania bryłowego (Part Design) i powierzchniowego (Shape Design). Utworzone detale łączy się w zespoły w module złożeniowym (Assembly Design), natomiast generowanie dokumentacji umożliwia moduł służący do tworzenia rysunków technicznych. Więcej informacji na temat budowania modeli za pomocą tychże modułów znaleźć można w pracy [4].

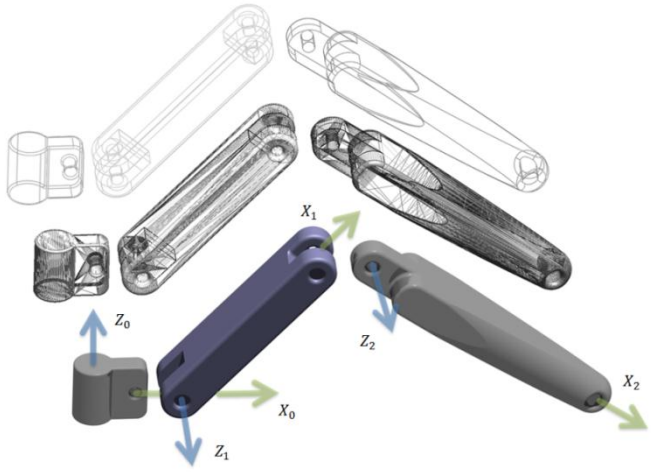
Zaproponowany model robota kroczącego ma charakter dość ogólny. Nie jest to model wiernie odzwierciedlający postać konstrukcyjną rzeczywistego obiektu. Uwzględniono w nim przede wszystkim analizę kinematyczną i reguły określające proces projektowania i sterowania robotami kroczącymi. Unikając zbędnego skomplikowania geometrii obiektu opracowany model zawiera daleko idące uproszczenia. Najważniejsze było zauważenie pewnych szczególnych cech postaci geometrycznej, niezbędnych do zastosowania więzów kinematycznych jak, np. otwory, osie, krawędzie czy płaszczyzny. Nadmiar elementów konstrukcyjnych mógłby ponadto powodować spowolnienie renderowania modelu, co uniemożliwiłoby symulację w czasie rzeczywistym. Zrezygnowano więc ze wszelkich elementów złącznych, a poszczególne segmenty odnóży tworzone są przez jednolite ogniwa. Nie zamodelowano również napędów ograniczając się do reprezentacji połączeń ruchomych za pomocą przegubów obrotowych.

Ogólny widok zamodelowanej makiety robota kroczącego przedstawia rys. 3.



Rys. 3 Ogólny widok zamodelowanej makiety robota kroczącego

Przyjęto strukturę korpusu typu holonomicznego. Odnóża rozmieszczone zostały na bazie koła w równych odstępach wynoszących $60°$. Każda z sześciu kończyn posiada trzy segmenty połączone szeregowo przegubami obrotowymi. Łączna suma par kinematycznych wynosi 18, przy czym każda z nich napędzana jest z osobna. Ostatni segment posiada zaokrąglenie imitujące kształtem stopę dopasowującą się do nierówności terenu. Wizualizacja modelu pojedynczej kończyny przedstawiona została na rys. 4.

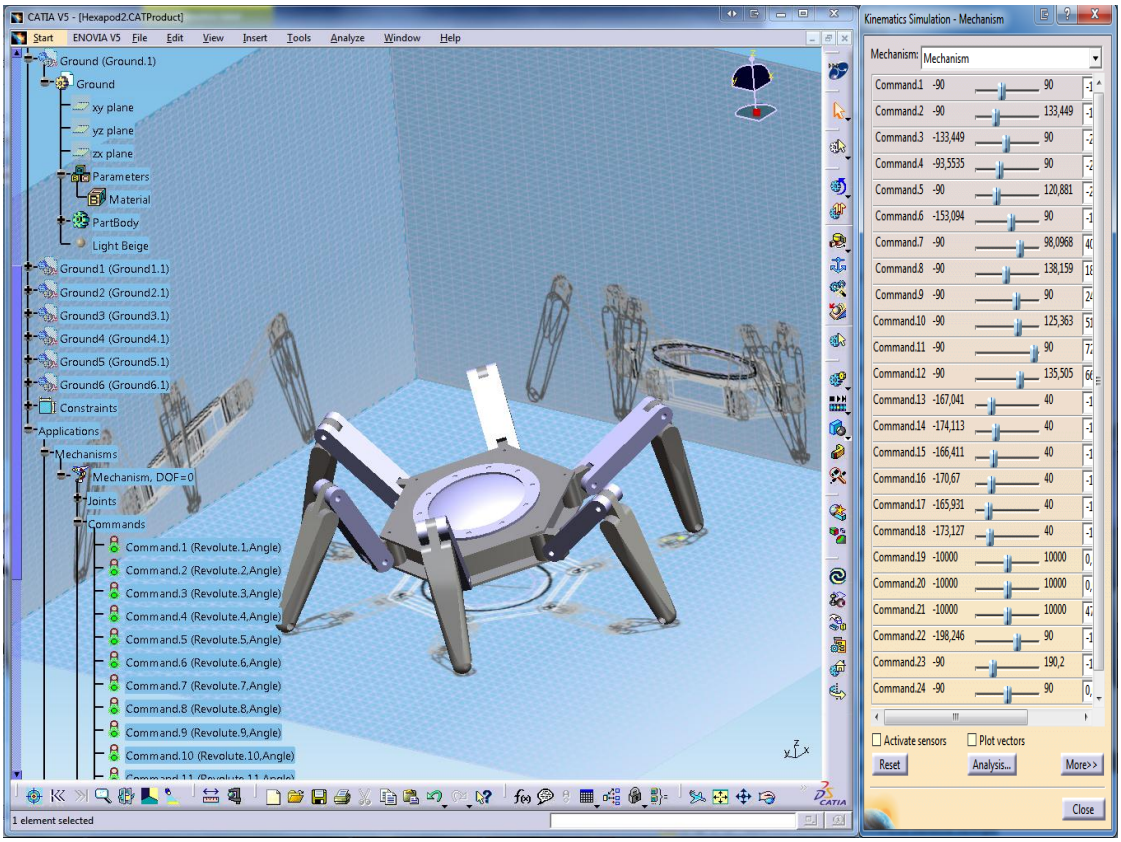


Rys. 4 Modele poszczególnych segmentów odnóża

Model kinematyczny

Do wprowadzenia zależności kinematycznych w modelu posłużył moduł przeznaczony do budowania makiet cyfrowych (DMU Kinematics). Jest to bardzo rozbudowana część składowa środowiska umożliwiająca przekształcenie więzów czysto konstrukcyjnych określających jedynie zależności złożeniowe pomiędzy detalami na więzy kinematyczne umożliwiające definiowanie wzajemnych przemieszczeń pomiędzy segmentami. Pod względem graficznego interfejsu użytkownika (ang. GUI – graphic user interface) moduł ten nie różni się znacząco od pozostałych modułów oferowanych przez środowisko. Trzeba jednak zwrócić uwagę na główne narzędzia umożliwiające budowę modeli kinematycznych, jak również ich szczegółową analizę [3].

Poszczególne segmenty każdego z odnóży połączone zostały ze sobą przy pomocy więzów Revolute Joint, przy jednoczesnym oznaczeniu wynikowych stopni swobody jako aktywne, sterowane niezależnie. Obliczona przez środowisko ogólna liczba stopni swobody (ang. DOF - deegres of freedom) wynosi zero (obliczenia nie uwzględniają aktywnych stopni swobody). Utworzono ponadto 18 poleceń służących do niezależnego sterowania każdym przegubem (rys. 5). Bezpośrednią kontrolę na przemieszczeniem kątowym w każdej parze kinematycznej daje narzędzie Simulation with Commands. Za pomocą suwaków ustawiona może zostać pozycja wszystkich przegubów, a rezultat tego ruchu przenoszony jest natychmiastowo na wszystkie połączone segmenty mechanizmu.



Rys. 5 Ogólny widok interfejsu graficznego modułu DMU Kinematics środowiska CATIA V5; widoczne pola analizy mechanizmu i interfejs narzędzia Simulation with Commands

Jak wspomniano wcześniej, aplikacja CATIA V5 umożliwia nagrywanie i odtwarzanie symulacji. Może to zostać zrealizowane na dwa sposoby. Pierwszą możliwością jest nagrywanie sekwencji ruchowych poprzez manualne ustawienie przemieszczeń w każdym z przegubów dla każdej chwili czasu trwania symulacji. Ze względu na dużą liczbę stopni swobody zadanie to jest bardzo żmudne, a wynik miałby charakter jednorazowy – wprowadzenie zmian w sekwencji symulacji nie jest możliwe do wykonania w prosty sposób. Drugim sposobem jest zapisanie przemieszczeń w każdym z przegubów jako funkcji czasu. Umożliwia to zaprogramowanie sekwencji ruchu mających charakter prezentacyjny. Po zakończeniu symulacji może być ona zmodyfikowana a następnie odtworzona ponownie. Celem niniejszej pracy jest natomiast stworzenie alternatywnej możliwości w postaci symulatora, pozwalającego na zmianę wszelkich parametrów i generowanie przemieszczeń w czasie rzeczywistym. Symulator taki wymaga utworzenia oddzielnego interfejsu graficznego umożliwiającego sterowanie modelem i procesora

obliczeniowego generującego ruch na poziomie warstw wykonawczych w zależności od komend wydawanych na poziomach wyższych. Zadanie to zostało zrealizowano poprzez napisanie autorskiej aplikacji opisanej w następnym rozdziale.

3.2. Aplikacja sterująca

Interfejs komunikacyjny

Aplikacja CATIA V5 napisana została w języku C++ i w pełni wykorzystuje obiektowe właściwości tego języka. Struktura środowiska posiada charakter obiektowy, będący połączeniem sposobów organizacji hierarchicznej oraz relacyjnej. Pewne jej elementy mogą być łączone ze sobą zgodnie z zasadą przynależności lub powiązania ideologicznego. Cały model obiektowy środowiska udostępniony został na zewnątrz aplikacji w postaci interfejsu COM (ang. Component Object Model). Jest to binarny interfejs opracowany przez firmę Microsoft umożliwiający komunikację pomiędzy procesami oraz dynamiczne tworzenie obiektów reprezentujących te procesy. Idea interfejsu COM polega na implementowaniu obiektów w sposób niezależny od języka, w którym została napisana aplikacja oraz na możliwości ich użycia w innym środowisku niż to, w którym zostały one stworzone, włączając w to użycie na różnych platformach sprzętowych oraz obsługę w sposób zdalny. Wymaga to od programistów tworzenia dobrze zdefiniowanych interfejsów odrębnych od właściwego kodu aplikacji. Z drugiej jednak strony COM umożliwia programistom niezależnym tworzenie własnych aplikacji korzystających z narzędzi i obiektów udostępnianych przez interfejsy abstrahując od właściwej ich implementacji. Należy również nadmienić, iż za pomocą interfejsów uzyskać można również dostęp do pewnych funkcji środowiska, które nie są udostępniane z poziomu interfejsu użytkownika.

Na potrzeby aplikacji sterującej napisano oddzielną bibliotekę implementującą klasę opakowującą (ang. wrapper) o nazwie $TCatia$. Klasa ta służy w większości jedynie przekształcaniu komend wewnętrznych aplikacji sterującej na wywołania zgodne z interfejsem COM poprzez użycie bibliotek OLEObject. Całość napisana została w języku C++.

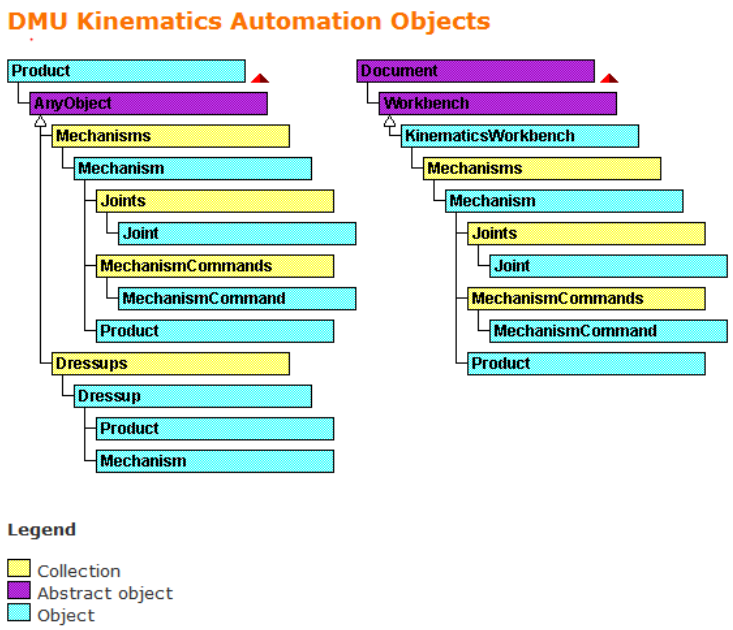
Jedną z najbardziej użytecznych funkcji interfejsu jest możliwość tworzenia automatycznego obiektów reprezentujących daną aplikację zarejestrowaną w systemie operacyjnym. Otwierając program w ten sposób zyskujemy ponadto w rezultacie wskaźnik do procesu tej aplikacji, który wykorzystywany jest do dalszej wymiany informacji z procesem.

```
fCatiaCom=Variant::CreateObject("catia.application");
```

$fCatiaCom$ w powyższym kodzie jest wskaźnikiem typu $Variant$ będącym uchwytem do głównego procesu otwieranej aplikacji CATIA V5. Po wykonaniu tej funkcji należy poczekać na pełne zainicjalizowanie aplikacji, po czym można od razu rozpocząć korzystanie z całej funkcjonalności interfejsu.

Poniższy kod powoduje wczytanie do środowiska dokumentu z przygotowanym modelem, do którego ścieżka przekazywana jest w zmiennej tekstowej doc:

```
CatiaCom.OlePropertyGet("Documents").OleFunction("open",doc.t_str());
```



Rys. 6 Model obiektowy struktur dotyczących zależności kinematycznych modelu [5]

Obiektami udostępnianymi przez interfejs CATIA V5 szczególnie przydatnymi na potrzeby symulacji ruchu modelu jest jeden z obiektów technologicznych uzyskiwanych poprzez enumerację z produktu modelu (Product) lub kolekcji modułów (Workbench): Mechanisms. Stanowi on kolekcję mechanizmów zdefiniowanych w modelu. Pobierając interesujący nas mechanizm uzyskać można z niego informację dotyczące wszystkich jego par kinematycznych (Joints), poleceń sterujących ruchem (MechanismCommands), a także reprezentacji ideologicznej działania mechanizmu (Dressups). Poniżej przedstawiono kod funkcji pobierającej z otwartego dokumentu obiekt technologiczny będący reprezentacją pierwszego mechanizmu w kolekcji:

```
Variant mechanism = Catia.
OlePropertyGet("ActiveDocument")
.OlePropertyGet("Product")
.OleFunction("GetTechnologicalObject","Mechanisms")
.OleFunction("Item",1);
```

Z uzyskanego obiektu $mechanism$ pobrać można następnie wartości odpowiadające wartościom konfiguracyjnym dla $dmDOF$ par kinematycznych w mechanizmie:

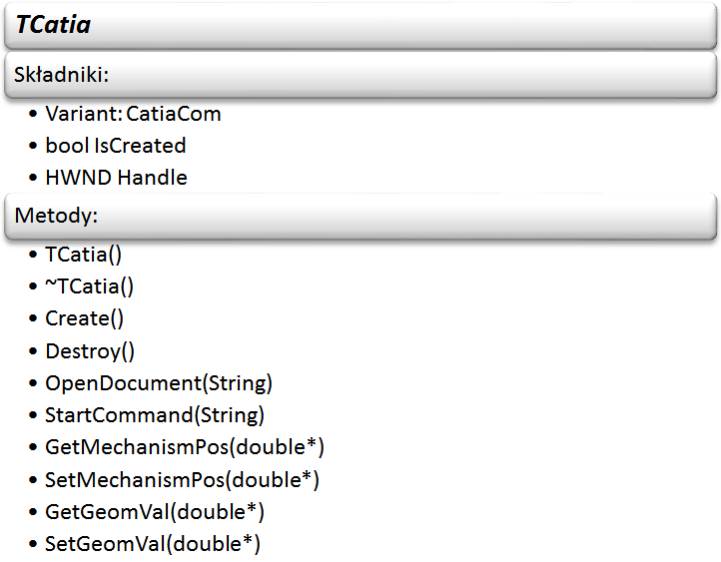
```
Variant varValues;
int bounds[2]={0,dmDOF};
varValues=VarArrayCreate(bounds,1,VT_VARIANT);
Mechanism .OleFunction("GetCommandValues",varValues);
```

Podobnie zrealizowano proces wysyłania komend do środowiska CATIA V5:

```
Mechanism.OleProcedure("PutCommandValues",varValues);
```

Po przypisaniu nowych wartości konfiguracyjnych CATIA automatycznie dokonuje ich przeliczenia i wizualizacji w przestrzeni trójwymiarowej. Jest to szczególnie istotna cecha tego środowiska zdejmująca z programisty konieczność zajmowania się generowaniem wizualizacji. Wystarczającym jest w tym przypadku skupienie się na obliczeniach kinematycznych pozwalających na zebranie zbioru zmiennych konfiguracyjnych w danej chwili symulacji oraz przesłanie ich do obiektu reprezentacji mechanizmu aplikacji CATIA. Za pomocą odpowiednich obiektów możliwe jest pobranie w podobny sposób wszystkich parametrów geometrycznych modelu. Dzięki temu zmiany wprowadzone w modelu nie wymagają rekompilacji programów lub przechowywania ich w osobnych plikach.

Wszystkie obliczenia wykonywane będą z uwzględnieniem aktualnych właściwości geometrycznych wczytanego modelu.



Rys. 7 Klasa opakowująca funkcję komunikacji z procesem CATIA V5

Procesor obliczeniowy

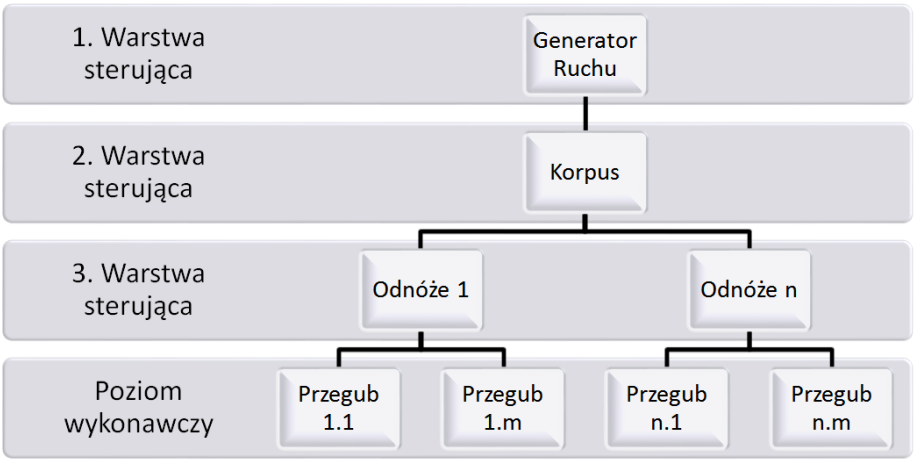
W celu zbudowania procesora obliczeniowego symulatora maszyny kroczącej zaimplementowano szereg klas enkapsulując je w osobnej bibliotece. Powinna ona spełniać przede wszystkim następujące funkcje:

1) Implementowanie struktur danych pozwalających na przechowywanie wszelkich informacji dotyczących modelu takich, jak:
   - Właściwości każdego z przegubów lub grup przegubów (zakresy maksymalnych przemieszczeń i prędkości, aktualne oraz zadane wartości przemieszczeń i prędkości),
   - Właściwości każdego z odnóży lub grup odnóży (ilość stopni swobody, długości segmentów, aktualne oraz zadane wartości przemieszczeń i prędkości, dane odczytywane z czujników związanych z odnóżami),
   - Właściwości związane z korpusem robota (wymiary korpusu, ilość odnóży, sposób ich rozmieszczenia, aktualne oraz zadane wartości przemieszczeń i prędkości, dane odczytywane z czujników związanych z korpusem),
   - Parametry determinujące typ, rodzaj i charakterystykę wykonywanych chodów maszyny (domyślna wysokość uniesienia korpusu, oddalenie odnóży od korpusu, długość kroku, wysokość kroku, współczynniki obciążenia odnóży, aktualne oraz zadane prędkości przemieszczania się urządzenia, typ wykonywanego chodu itp.).
2) Implementacja algorytmów do obliczeń matematycznych zależności związanych z modelem to:
   - Algorytmy rachunku macierzowego i wektorowego,
   - Obliczenia dotyczące generowania macierzy przekształceń jednorodnych,
   - Dokonywanie przekształceń za pomocą notacji Denavita-Hartenberga pomiędzy dowolnymi układami współrzędnych,

- Wykonywanie statycznych projekcji punktów związanych z maszyną wzdłuż dowolnego wektora,
- Obliczenia rozwiązań dla prostej i odwrotnej kinematyki pozycyjnej każdej z kończyn,
- Obliczenia rozwiązań kinematyki różnicowej,
- Obliczanie warunków statycznej i energetycznej stabilności urządzenia,
- Algorytmy generowania chodów sekwencyjnych i swobodnych.

3) Implementacja funkcji publicznych, za pomocą których wykonywane mogą być z zewnątrz następujące zadania:
   - Operowanie na strukturach danych tworzących zbiór wszystkich parametrów systemu,
   - Bezpośrednie sterowanie poszczególnymi warstwami modelu sterowania,
   - Sterowanie warstwami niższymi za pomocą warstw wyższego rzędu, a tym samym dokonywanie syntezy ruchu całego systemu.
4) Implementacja procesora czasu umożliwiającego:
   - Iteracyjne symulowanie realizacji zadań w czasie rzeczywistym,
   - Synchronizację działania poszczególnych obiektów wchodzących w skład systemu dążąc do realizacji zadanego celu.

Modelując strukturę biblioteki starano się przybliżyć ją do budowy hierarchicznego systemu sterowania, jaki mógłby potencjalnie zostać zastosowany przy budowie fizycznego prototypu. Strukturę takiego systemu sterowania przedstawiono na rys. 8.



Rys. 8 Przykład hierarchicznego systemu sterowania robotem kroczącym

Najniższą warstwę systemu tworzą przeguby w odnóżach robota. Każdy z nich posiada informacje na temat zakresów przemieszczeń jakie może wykonywać oraz maksymalne i minimalne wartości prędkości z jakimi wykonywany może być ruch. Istnieje przy tym możliwość bezpośredniego kontrolowania tychże przegubów.

Drugą licząc od końca warstwę stanowią odnóża urządzenia. Każde z nich posiada pewną ilość przegubów, nad którymi posiada ono pełną kontrolę. Odnóża posiadają informacje dotyczące swoich wymiarów geometrycznych, a także implementują algorytmy rozwiązania prostego i odwrotnego zadania kinematyki pozycyjnej umożliwiające wysterowanie odpowiednich przegubów w zależności od zadanych pozycji wejściowych.

Kolejną warstwę tworzy korpus maszyny, który przechowuje dotyczące go parametry geometryczne oraz kolekcję przyczepionych do niego odnóży. Posiada on również własne

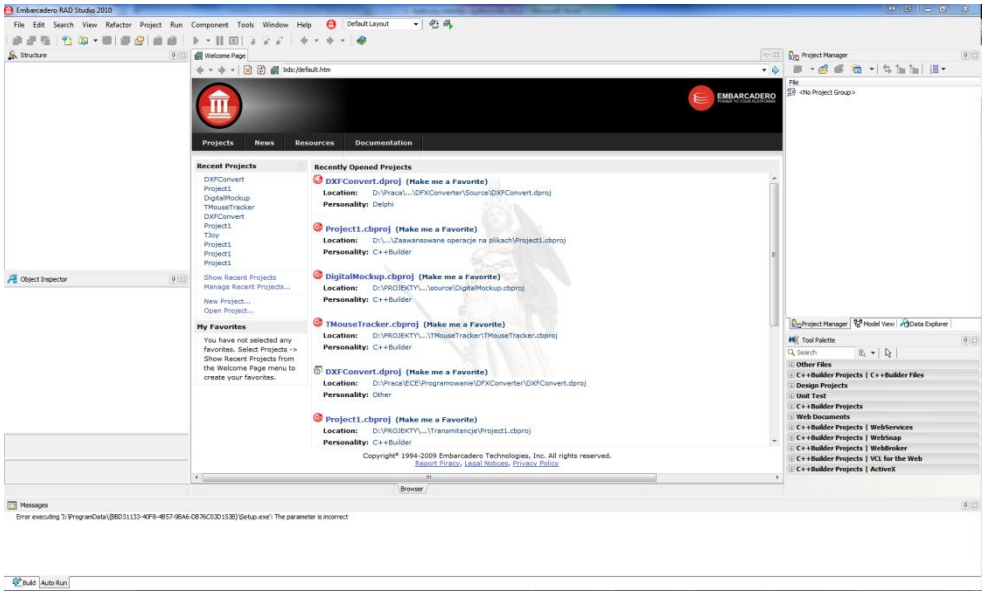
metody sterowania pozwalające na wydawanie komend warstwom niższym w zależności od zadanych położeń korpusu.

Najwyższy poziom w modelu stanowi generator ruchu przechowujący w swoich strukturach parametry dotyczące chodu urządzenia. Posiada on również kolekcję algorytmów generowania chodu. W zależności od wybranego rodzaju chodu przekazuje on odpowiednie rozkazy do warstw niższych. Skoordynowane działanie wszystkich poziomów systemu sterowania umożliwia wykonanie przez maszynę ruchu kroczącego. Szerzej na ten temat w pracy [6].

Graficzny interfejs użytkownika

Graficzny interfejs użytkownika napisany został również w języku C++. W celu stworzenia intuicyjnego, łatwego w obsłudze i dobrze wyglądającego interfejsu wykorzystano zintegrowane środowisko programistyczne (ang. IDE – integrated development environment). Środowiska takie udostępniają obszerną funkcjonalność obejmującą nie tylko edycję i kompilowanie kodu źródłowego, ale również wsparcie w niemalże każdym aspekcie cyklu życia oprogramowania takim, jak tworzenie zasobów programu, graficznych formatek, okien dialogowych, raportów, baz danych, komponentów, czy też dokumentacji oprogramowania.

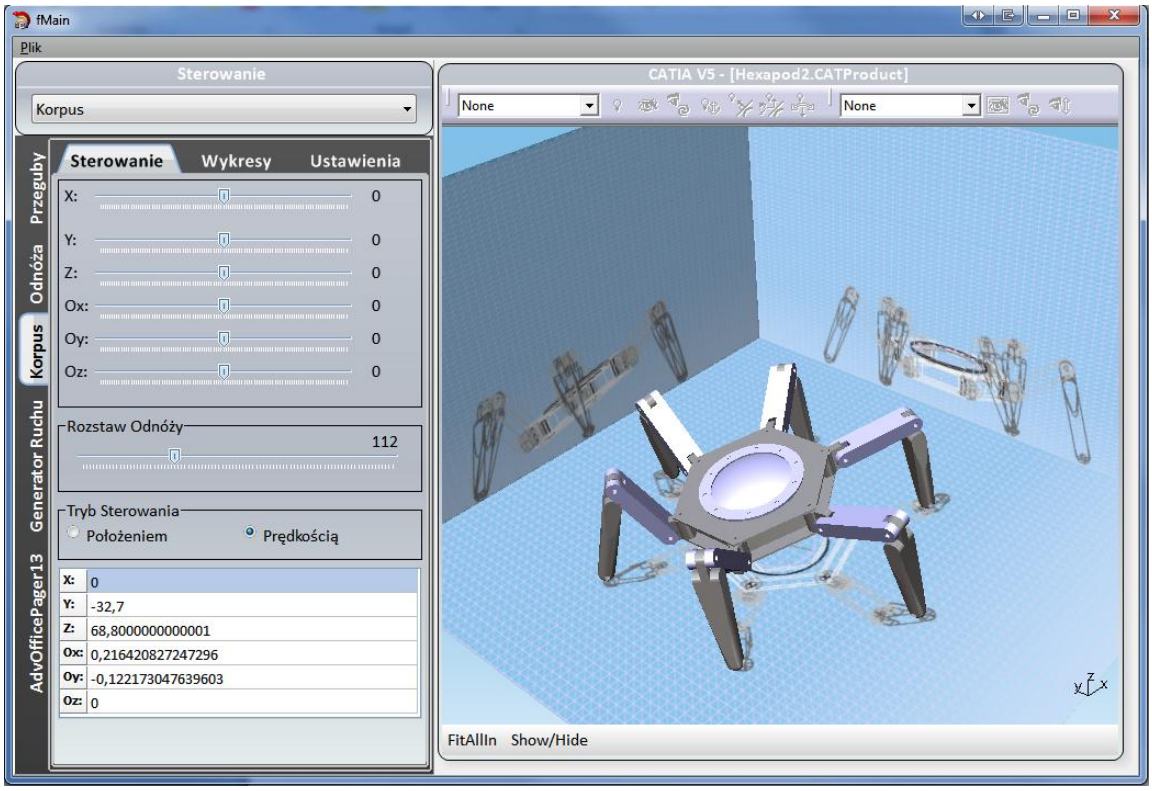
W niniejszej pracy zastosowano środowisko Embarcadero RAD Studio 2010 (rys. 9) stanowiące rozwinięcie koncepcji integracji środowiska w postaci idei szybkiego prototypowania aplikacji (ang. RAD – rapid application development) [7]. Środowisko to jest kolejną odsłoną jednej z najpopularniejszych aplikacji tego typu rozwijanej początkowo przez koncern Borland. Aplikacja korzysta z w pełni obiektowej biblioteki VCL napisanej początkowo na potrzeby Delphi. Komponenty tejże biblioteki implementują dziesiątki tysięcy obiektów tworzonych przez twórców środowiska oraz przez jego użytkowników. Pełnią one różnorodne funkcje poczynając od budowania reprezentacji graficznej, poprzez obsługę baz danych i urządzeń zewnętrznych, po implementacje gotowych rozwiązań jak algorytmy genetyczne czy elementy obsługa całych serwisów. Dzięki temu nie jest konieczne pisanie aplikacji każdorazowo od nowa, lecz możliwe jest skorzystanie z istniejących elementów składowych, a budowane na ich podstawie własne implementacje zamykać można w postaci kolejnych komponentów.



Rys. 9 Główne okno środowiska programistycznego Embarcadero RAD Studio 2010

Główne okno aplikacji podzielone zostało na dwie główne części (rys. 10). Pierwszą z nich zajmuje panel wizualizacji trójwymiarowej, druga natomiast służy przeglądania warstw systemu sterowania oraz bezpośredniego sterowania robotem w czasie rzeczywistym.

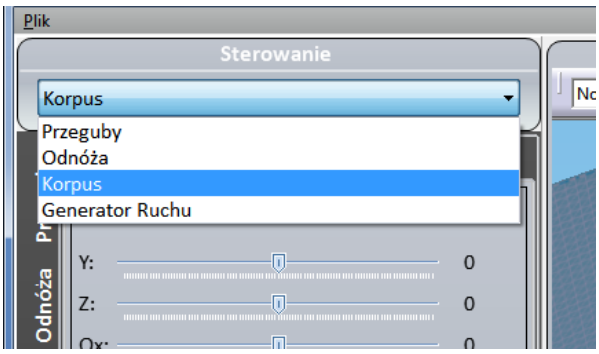
Do zagnieżdżenia wizualizacji w oknie symulatora wykorzystano metody oferowane przez WinApi systemu operacyjnego Microsoft Windows [1]. WinApi stanowi niezależny od języka interfejs umożliwiający wykorzystanie funkcjonalności systemu operacyjnego we własnych aplikacjach. Zastosowanie funkcji takich jak $EnumWindows, GetParent, GetWindowRecct, SetParent, SetWindowPos$ itp. pozwala na przejęcie kontroli nad oknem dowolnej aplikacji działającej w systemie. Możliwe jest również przechwytywanie wszelkich komunikatów wysyłanych do oraz od danego procesu, co pozwala na filtrowanie przez aplikację czynności wykonywanych przez dany proces. W powyżej opisany sposób przejęte zostało okno środowiska CATIA V5 z wizualizacją modelu.



Rys. 10 Główne okno autorskiego symulatora maszyny kroczącej

Druga część to główny panel sterowania umożliwiający wydawanie poleceń ruchu oraz analizę dokonywanych przemieszczeń. W górnej części panelu znajduje się lista rozwijana umożliwiająca wybór aktywnego obiektu sterowania (rys. 11). Sterowaniu podlegać mogą:

- pojedyncze przeguby – sterowanie na poziomie wykonawczym,
- odnóża – sterowanie na poziomie warstwy pojedynczego odnóża,
- korpus – odpowiada warstwie sterującej ruchem całego korpusu urządzenia,
- generator ruchu – umożliwia generowanie ruchu poprzez syntezę przemieszczeń korpusu oraz pojedynczych (lub grupy) odnóży.



Rys. 11 Lista rozwijana zawierająca możliwe obiekty sterowania

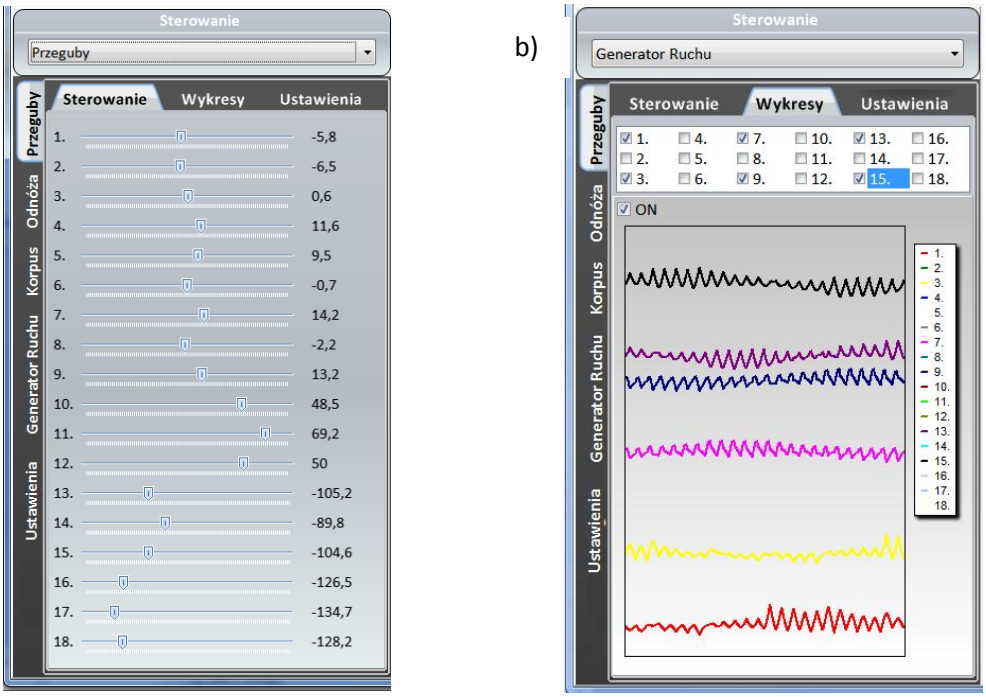
Dolna część panelu zawiera zakładki, w których umieszczone zostały narzędzia służące zmianie parametrów właściwych dla danej warstwy sterowania. Każda z zakładek pełni dwie role. Pierwsza z nich uaktywnia się w momencie, gdy warstwa systemu jest w danym momencie obiektem bezpośredniego sterowania. Istnieje wówczas możliwość wysyłania komend sterujących do procesora obliczeniowego, które zmieniają aktualny stan modelu. Wszelkie zmiany podlegają przy tym wizualizacji w czasie rzeczywistym. Druga funkcja aktywna jest w momencie, gdy dana warstwa nie jest aktualnym obiektem sterowania. Panel pozwala wówczas jedynie na podgląd zmieniających się wartości w czasie oraz analizę generowanych komend pochodzących z warstw wyższego poziomu. Funkcja ta zostanie szerzej omówiona przy okazji prezentacji poszczególnych warstw sterowania. Pierwszą zakładką panelu sterowania jest zakładka odpowiadająca warstwie związanej z przegubami robota (rys. 12). Zawiera ona ponumerowane suwaki umożliwiające zmianę pozycji w każdym z przegubów. Ilość suwaków generowana jest dynamicznie na podstawie informacji uzyskanych z procesora obliczeniowego.
W przypadku sterowania z poziomu warstw wyższych panel ten jest nieczuły na akcje użytkownika pozwalając jednak na obserwację zmian we wszystkich przegubach. Analizę tychże zmian dokonać można korzystając z funkcji drugiej części panelu. Umożliwia ona rysowanie wykresów przemieszczeń lub prędkości w funkcji czasu dla poszczególnych przegubów wskazanych przez użytkownika. Na rys. 12b przedstawione zostały przebiegi przemieszczeń w przegubach dwóch sąsiednich nóg podczas realizacji adaptacyjnego chodu trójpodporowego.

Kolejny panel odpowiada za sterowanie poszczególnymi nogami robota (rys. 13). Umożliwia on zadawanie położeń lub prędkości osobno dla każdego z odnóży. Możliwe jest tutaj sterowanie wybranym odnóżem zarówno w układzie związanym ze stawem biodrowym odnóża jak również w układzie związany z korpusem urządzenia. Na bieżąco wyświetlane są ponadto położenia końcówki odnóża względem wybranego układu odniesienia. Podobnie jak w przypadku przegubów także tutaj istnieje możliwość rysowania w czasie rzeczywistym wykresów przemieszczeń i prędkości końcówki odnóża.

Następny panel umożliwia sterowanie ruchem całego korpusu robota. Do dyspozycji oddanych zostało sześć suwaków odpowiadających sześciu stopniom swobody korpusu w przestrzeni. W momencie wywoływania ruchu komendy wysyłane są do procesora obliczeniowego, który rozsyła komendy dalej do niższych warstw sterowania – odnóży. Możliwe jest tutaj również bezpośrednie sterowanie położeniem lub prędkością poruszania korpusu. Dodatkowo możliwa jest zmiana parametrów związanych ze strukturą korpusu takich jak, np. domyślna odległość odnóży od korpusu. Zmiana tej wartości skutkuje natychmiastową reakcją w postaci wizualizacji. Umożliwia to dokonanie analizy ruchliwości korpusu w zależności od tego parametru oraz wypracowanie wartości quasi-optymalnych.

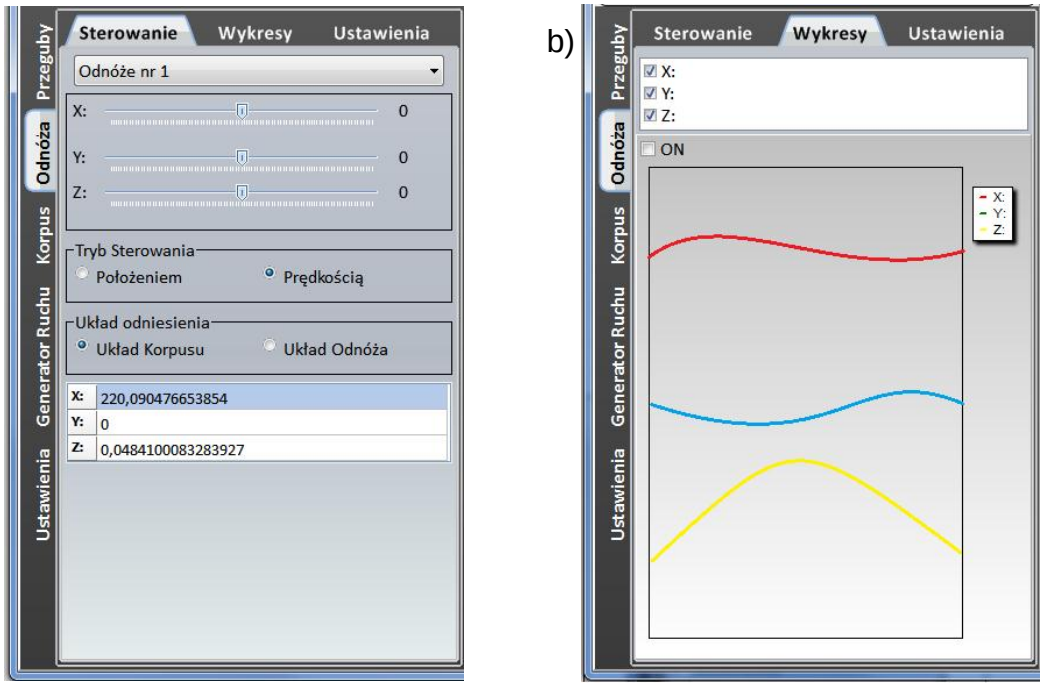
Aktualne położenie korpusu w przestrzeni jest dodatkowo wyświetlane w postaci tabelarycznej.

a) b)



Rys. 12 Panel sterowania warstwy wykonawczej związanej z przegubami; a) sterowanie poszczególnymi przegubami za pomocą suwaków, b) sporządzanie wykresów przemieszczeń i prędkości w wybranych przegubach

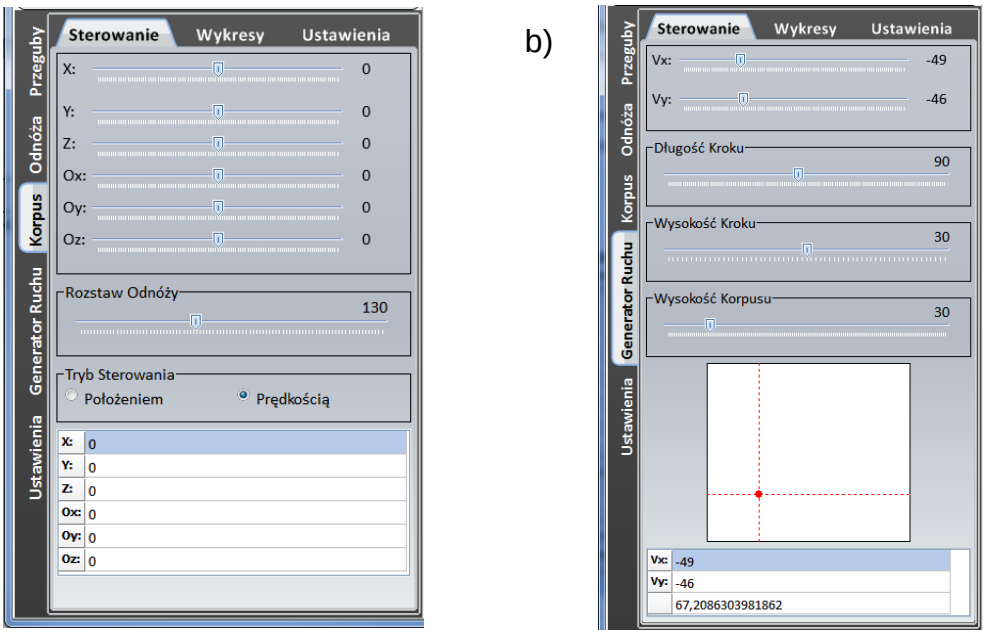
a) b)



Rys. 13 Panel sterowania warstwy związanej z odnóżami; a) – sterowanie odnóżami oraz związanymi z nimi parametrami, b) – wykresy położenia końcówki odnóża w trakcie jednej sekwencji ruchu robota

Ostatnią z zakładek panelu sterowania (rys. 14) stanowi panel sterowania warstwy generującej ruch całej maszyny. Zadanie kierunku ruchu możliwe jest za pomocą suwaków (definiujących prędkość poruszania osobno względem osi X_0 oraz Y_0 układu bazowego) oraz za pomocą „touchpad'a" pozwalającego na intuicyjne określanie kierunku poruszania, podczas gdy prędkość obliczana jest jako odległość od punktu wykrycia ruchu do centrum jego powierzchni. Panel umożliwia również wybór algorytmu generującego chód maszyny oraz określanie specyficznych dla niego parametrów. Pozwala to na bezpieczne (wirtualne) testowanie tychże parametrów oraz wykrywanie niebezpieczeństw związanych z np. kolizjami pomiędzy odnóżami, czy też utratą stabilności przez maszynę, poprzez rysowanie w czasie rzeczywistym wielokątów podparcia tworzonych przez punkty śladowe kończyn z zaznaczeniem projekcji środka ciężkości.

a) b)



Rys. 14 a) Panel sterowania ruchem korpusu modelu, b) panel generowania ruchu całej maszyny kroczącej

Opracowany symulator okazał się niezwykle pomocny przy opracowywaniu algorytmów generowania ruchu maszyny kroczącej. Pomimo, iż znajduje się on nadal w fazie rozwoju oraz testowania, pozwolił na zdobycie wiedzy dotyczącej metod sterowania urządzeniami tego typu oraz parametrów określających ogólną ich postać geometryczną i konstrukcyjną.

4. Podsumowanie

Na podstawie przeprowadzonej analizy strukturalnej i kinematycznej oraz chodu zamodelowano cyfrową makietę robota kroczącego. Model ten został poddany parametryzacji geometrycznej poprzez zdefiniowanie reguł opisujących wymiary, położenia i maksymalne zakresy ruchów poszczególnych jej elementów. Na połączenia elementów budujących makietę nałożono więzy kinematyczne umożliwiając symulowanie jej ruchu. System sterowania maszyny kroczącej przedstawiono w postaci wirtualnego modelu obiektowego, a następnie jego implementacji w języku C++ do postaci odrębnej biblioteki.

Cyfrowa makieta została połączona z wirtualnym systemem sterowania za pomocą autorskiej aplikacji. Środowisko posiada interfejs graficzny dający bezpośrednią kontrolę

nad systemem sterowania oraz umożliwiający wizualizację efektów jego działania w czasie rzeczywistym za pomocą wykonanej makiety.

Literatura

[1] Hollingworth J., Swart B., Cashman M., Gustavson P.: *C++ Builder. Vademecum Profesjonalisty.* HELION, Gliwice 2003.
[2] Koyachi N., Arai T., Adachi H., Asami K., Itoh, Y.: *Geometric design of hexapod with integrated limb mechanism of leg and arm.* Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference. Pittsburgh, PA 2002, s. 291 – 296 vol. 3.
[3] Palmer L.R., Diller E.D., Quinn, R.D.: *Design of a wall-climbing hexapod for advanced maneuvers.* Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference. St. Louis, MO 2009, s. 625 – 630.
[4] Qingsheng L., Hui Z., Baoling H., Xiaochuan Z.: *Research on biologically inspired hexapod robot's gait and path planning.* Information and Automation, 2009. ICIA '09. International Conference. Zhuhai, Macau 2009, s. 1546 – 1550.
[5] Stone H.W., Sanderson A.C.: *Statistical performance evaluation of the S-model arm signature identification technique.* Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference. Philadelphia 2002, s. 939 – 946 vol. 2.
[6] Trzmiel A.: *Modelowanie chodu robota sześcionożnego*. Praca inżynierska, Kraków 2011.
[7] *V5Automation.* Dokumentacja modelu automatyzacji środowiska CATIA V5 rozprowadzana wraz z instalacją aplikacji.
[8] Zielińska T.: *Maszyny kroczące.* PWN, Warszawa 2003.