Technologia i rozwiązania

Nauka robotyki z językiem Python

Przekonaj się, jak fascynujące jest programowanie robotów!

Lentin Joseph

Helion

[PACKT] open source community experience distilled PUBLISHING

Tytuł oryginału: Learning Robotics Using Python

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-2345-2

First published in the English language under the title „Learning Robotics Using Python – (9781783287536)".

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Spis treści

ROZDZIAŁ 2

Projekt mechaniki robota usługowego

Dzięki tej książce nauczysz się robotyki poprzez projektowanie i budowanie robotów, a także programowanie ich za pomocą języka Python. Zastanówmy się, w jaki sposób od podstaw opracować projekt mechaniczny robota. Robot, którego zamierzamy zbudować, będzie wykorzystany jako robot usługowy do podawania żywności i napojów w hotelach i restauracjach.

W rozdziale tym zapoznamy się z różnymi komponentami mechanicznymi robota, ponadto nauczymy się montowania tych komponentów. Możemy zaprojektować i zestawić ze sobą komponenty składowe, używając narzędzia CAD, a także zbudować model 3D w celu symulacji robota.

Faktyczny robot, który mógłby być zastosowany w hotelu, może być duży, ale w tej książce skupimy się na zbudowaniu jego miniaturowej wersji, wyłącznie do testowania naszej technologii. Jeżeli jesteś zainteresowany budowaniem robotów od podstaw, to ten rozdział jest dla Ciebie. Jeśli nie jesteś zainteresowany budowaniem robotów od zera, to aby pracować z tą książką, możesz wybrać jakąś gotową platformę robota dostępną na rynku.

Aby zbudować ciało robota, trzeba najpierw poznać wymagania projektowania robotów. Po zebraniu tych wymagań możemy przy użyciu narzędzia CAD zaprojektować i narysować model 2D, który będzie potrzebny do produkcji części robota. Przydatny może być także model 3D wymagany do symulacji robota, czym zajmiemy się w następnym rozdziale.

Wymagania dla robota usługowego

Aby zaprojektować dowolnego robota, najpierw należy zidentyfikować dla niego wymagania. Oto zbiór wymagań sprzętowych, jakie powinien spełnić robot, którego budujemy:

- Robot musi mieć zdolność przenoszenia żywności.
- Robot powinien być w stanie unieść maksymalnie 5 kg.
- Robot powinien poruszać się z szybkością pomiędzy 0,25 m/s a 1 m/s.
- Odległość podwozia robota od podłoża powinna być większa niż 3 cm.
- Robot musi być w stanie pracować nieprzerwanie przez 2 godziny.
- Robot powinien być w stanie poruszać się i dostarczać żywność do każdego stołu, omijając przeszkody.
- Wzrost robota powinien mieścić się w przedziale od 40 cm do 1 metra.
- Robot powinien być tani.

Teraz możemy rozpoznać wymagania projektu mechanicznego robota, takie jak ładowność, prędkość poruszania się, wysokość podwozia, wzrost i koszty. Zaprojektujemy ciało robota i odpowiednio wybierzemy komponenty. Spróbujmy omówić mechanizm robota, który można wykorzystać, aby spełnić te wymagania.

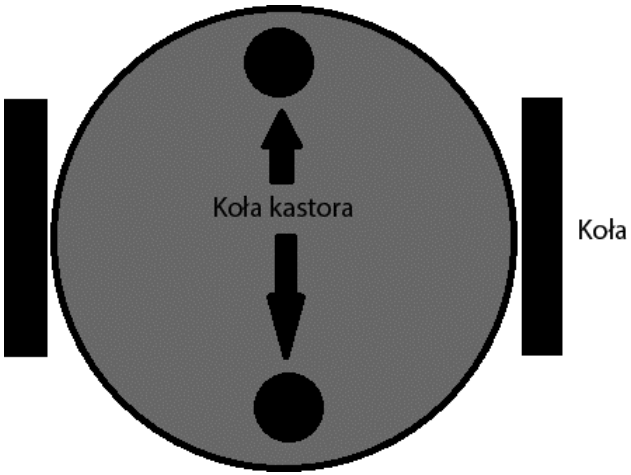
Mechanizm napędowy robota

Jednym ze skutecznych rozwiązań dla mobilnej nawigacji robotem są systemy sterowania bazujące na mechanizmie różnicowym. To jeden z najprostszych mechanizmów poruszania się mobilnego robota, który ma się przemieszczać głównie wewnątrz pomieszczeń. **Napęd robota bazujący na przekładni różnicowej** składa się z dwóch kół zamontowanych na wspólnej osi sterowanej przez dwa oddzielne silniki. Jest wyposażony w dwa koła pomocnicze nazywane **kołami kastora** (ang. *caster wheels*). Koła te zapewniają stabilność i równomierne rozłożenie wagi robota. Typowy mechanizm napędowy bazujący na przekładni różnicowej pokazano na rysunku na następnej stronie.

Kolejnym krokiem jest wybór komponentów mechanicznych dla systemu napędowego robota, tzn. silników, kół i podwozia. Bazując na wymaganiach, najpierw omówimy sposób wyboru silnika.

Wybór silników i kół

Silniki wybiera się po zapoznaniu się z ich specyfikacjami. Do najważniejszych parametrów decydujących o wyborze silnika należą moment obrotowy i liczba obrotów na minutę (rpm). Wartości te możemy obliczyć na podstawie podanych wymagań.



Mechaniczny napęd różnicowy

Określenie liczby obrotów na minutę silników

Załóżmy, że robot ma się poruszać z prędkością 0,35 m/s. Z wymagań wynika, że szybkość robota musi mieścić się w przedziale od 0,25 m/s do 1 m/s. Przyjmijmy, że średnica koła wynosi 9 cm, ponieważ według wymagań wysokość podwozia powinna być większa niż 3 cm. Korzystając z poniższego wzoru, możemy obliczyć obroty na minutę dla silnika:

$$rpm = 60 \times prędkość / (3{,}14 \times średnica\ koła)$$

$$rpm = (60 \times 0{,}35) / (3{,}14 \times 0{,}09) = 21 / 0{,}2826 = 74\ rpm$$

> Aby zapoznać się z obliczeniami dla robota, można wejść na stronę *http://www.robotshop.com/blog/en/vehicle-speed-rpm-and-wheel-diameterfinder-9786*.

Przy średnicy koła 9 cm i prędkości 0,35 m/s mamy 74 obroty na minutę. Możemy zaokrąglić tę liczbę do standardowej wartości 80 obrotów na minutę.

Obliczanie momentu obrotowego silnika

Pora obliczyć moment obrotowy potrzebny do poruszania się robota:

1. Liczba kół = 4 koła włącznie z 2 kołami pomocniczymi.
2. Liczba silników = 2.
3. Załóżmy, że współczynnik tarcia wynosi 0,6, a promień koła wynosi 4,5 cm.
4. Całkowita waga robota = waga robota + ładunek = (W = mg) = (~100 N + ~50 N) W = ~150 N, natomiast całkowita masa = 15 kg.
5. Nacisk na cztery koła można zapisać jako $2 \times N1 + 2 \times N2 = W$, czyli $N1$ oznacza ciężar oddziałujący na każde koło pomocnicze, a $N2$ na każde koło napędowe.

6. Załóżmy, że robot jest stacjonarny. Maksymalny moment obrotowy jest wymagany w chwili, gdy robot zaczyna się poruszać. Musi on także pokonać tarcie.
7. Dla nieporuszającego się robota możemy zapisać siłę tarcia jako moment obrotowy = 0.

Jeśli obliczymy moment obrotowy robota w takim stanie, to możemy uzyskać maksymalny moment obrotowy z następującego wzoru:

$\mu \times N \times r - T = 0$, gdzie μ jest współczynnikiem tarcia, N to średnia wartość siły działającej na każde koło, r oznacza promień koła, a T jego moment obrotowy.

$N = W / 4$ (zakładając, że waga robota jest rozłożona równo na wszystkie cztery koła).

W związku z tym otrzymujemy:

$0{,}6 \times (150/4) \times 0{,}045 - T = 0$

Stąd T = 1,0125 Nm lub 10,32 kg-cm.

Podsumowanie projektu

Wykonując projekt, obliczyliśmy następujące wartości:

- liczba obrotów silnika na minutę (rpm) = 80;
- moment obrotowy silnika = 10,32 kg-cm;
- średnica koła = 9 cm.

Projekt podwozia robota

Po obliczeniu parametrów silnika robota i kół możemy zaprojektować jego podwozie (czyli ciało). Zgodnie z wymaganiami robot powinien mieć możliwość przenoszenia żywności, powinien móc podnieść ładunek do 5 kg, wysokość podwozia robota od podłoża powinna być większa niż 3 cm i robot powinien być tani. Oprócz tego robot musi zapewniać możliwość montażu komponentów elektronicznych, na przykład **komputera PC**, sensorów i akumulatora.

Jednym z najprostszych projektów, który pozwala spełnić te wymagania, jest konstrukcja „stołopodobna". Przykład konstrukcji stołopodobnej można znaleźć na stronie projektu TurtleBot (*http://www.turtlebot.com/*). W projekcie tym ciało robota składa się z trzech warstw. Platforma o nazwie **Roomba** jest mechanizmem napędowym. Platforma Roomba posiada wbudowane silniki i sensory, dzięki czemu nie musimy się martwić o projekt osprzętu robota. Projekt podwozia robota **TurtleBot** pokazano na rysunku na następnej stronie.

Robot TurtleBot

Zaprojektujemy robota podobnego do TurtleBota, z własną platformą jezdną i własnymi komponentami. W naszym projekcie również zastosujemy architekturę trójwarstwową. Przed przystąpieniem do projektowania, spróbujmy się zastanowić, jakich narzędzi będziemy potrzebowali.

Przed rozpoczęciem projektowania podwozia robota musimy zapoznać się z narzędziami do komputerowego wspomagania projektowania (ang. *Computer-aided design — CAD*). Do popularnych narzędzi CAD należą:

- SolidWorks (*http://www.solidworks.com/default.htm*),
- AutoCAD (*http://www.autodesk.com/products/autocad/overview*),
- Maya (*http://www.autodesk.com/products/maya/overview*),
- Inventor (*http://www.autodesk.com/products/inventor/overview*),
- Google SketchUp (*http://www.sketchup.com/*),
- Blender (*http://www.blender.org/download/*),
- LibreCAD (*http://librecad.org/cms/home.html*).

Podwozie robota może zostać zaprojektowane przy użyciu dowolnego programu CAD. W tej książce model 2D zademonstrujemy za pomocą programu LibreCAD, a model 3D za pomocą programu Blender. Zaletą tych aplikacji jest to, że są one darmowe i dostępne dla wszystkich platform systemów operacyjnych. Do przeglądania i sprawdzania modelu 3D będziemy używać narzędzia podglądu siatki 3D MeshLab, a naszym głównym systemem operacyjnym będzie Ubuntu. Aby rozpocząć proces projektowania, zapoznamy się z procedurą instalacji tych aplikacji w systemie Ubuntu 14.04.2. Zostaną podane również linki do przewodników instalacji aplikacji na innych platformach.

Instalacja oprogramowania LibreCAD, Blender i MeshLab

LibreCAD to darmowa aplikacja CAD 2D typu open source dostępna dla systemów Windows, OS X i Linux. **Blender** to darmowe oprogramowanie open source do tworzenia komputerowej grafiki 3D (modeli 3D, animacji i gier wideo), dostępne na licencji GPL, zgodnie z którą użytkownicy mają prawo do współdzielenia, modyfikowania i dystrybucji aplikacji. **MeshLab** to przenośny i rozszerzalny program open source pozwalający na przetwarzanie i edytowanie trójkątnych i pozbawionych struktury siatek 3D.

Poniżej zamieszczono linki do instalacji programu LibreCAD w systemach Windows, Linux i OS X:

- Odwiedź stronę *http://librecad.org/cms/home.html*, aby pobrać oprogramowanie LibreCAD.
- Odwiedź stronę *http://librecad.org/cms/home/fromsource/linux.html*, by zbudować oprogramowanie LibreCAD z kodu źródłowego.
- Odwiedź stronę *http://librecad.org/cms/home/installation/linux.html*, aby zainstalować oprogramowanie LibreCAD w systemie Debian (Ubuntu).
- Odwiedź stronę *http://librecad.org/cms/home/installation/rpmpackages.html*, aby zainstalować oprogramowanie LibreCAD w systemie Fedora.
- Odwiedź stronę *http://librecad.org/cms/home/installation/osx.html*, aby zainstalować oprogramowanie LibreCAD w systemie OS X.
- Odwiedź stronę *http://librecad.org/cms/home/installation/windows.html*, aby zainstalować LibreCAD w systemie Windows.

> Dokumentacja oprogramowania LibreCAD jest dostępna pod adresem: *http://wiki.librecad.org/index.php/Main_Page*.

Instalacja programu LibreCAD

W tym podrozdziale zamieszczono procedurę instalacji dla wszystkich systemów operacyjnych. Użytkownicy systemu Ubuntu mogą zainstalować LibreCAD także za pośrednictwem systemu Ubuntu Software Centre.

Instalacja programu Blender

Aby zainstalować najnowszą wersję Blendera dla wybranej platformy OS, należy odwiedzić stronę *http://www.blender.org/download/*. Najnowszą wersję dokumentacji Blendera można pobrać pod adresem *http://wiki.blender.org/*.

Użytkownicy systemów Ubuntu (Linux) mogą zainstalować program Blender za pośrednictwem systemu Ubuntu Software Centre.

Instalacja programu MeshLab

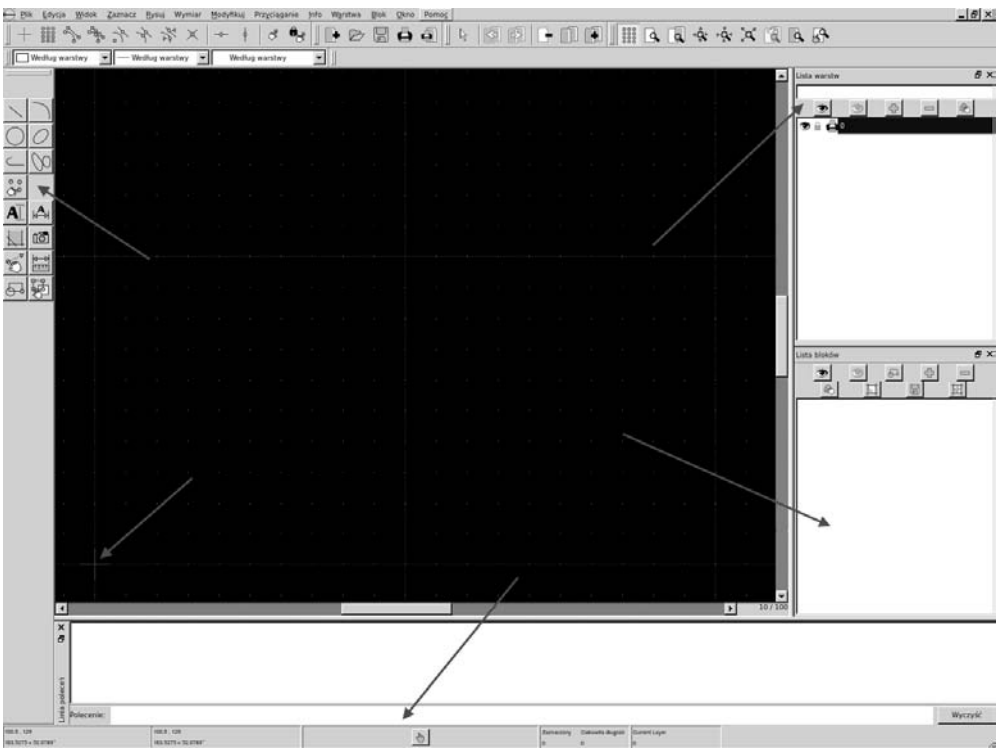
MeshLab jest dostępny dla wszystkich platform OS. Pod adresem *http://meshlab.sourceforge.net/* dostępne są łącza pobierania skompilowanych binariów i kodu źródłowego programu MeshLab.

Użytkownicy systemu Ubuntu mogą zainstalować program MeshLab za pośrednictwem menedżera pakietów apt, korzystając z następującego polecenia:

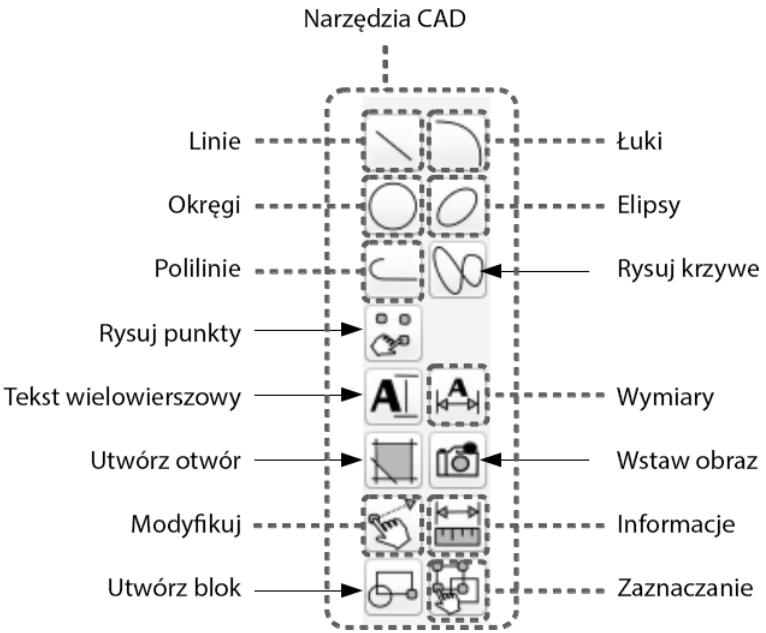
```
$sudo apt-get install meshlab
```

Tworzenie rysunku CAD 2D robota z wykorzystaniem programu LibreCAD

Przyjrzyjmy się podstawowemu interfejsowi programu LibreCAD. Zaprezentowano go na poniższym zrzucie ekranu:



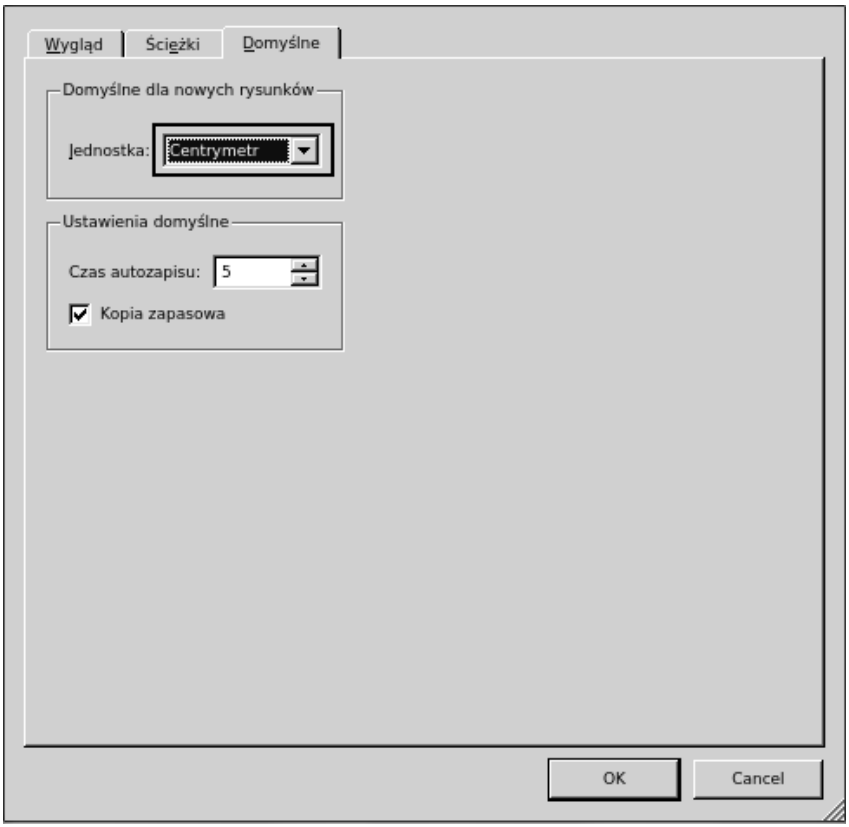
Pasek narzędzi CAD ma komponenty niezbędne do narysowania modelu. Szczegółowy widok paska narzędzi CAD zamieszczono na poniższym zrzucie ekranu:



Szczegółowy opis narzędzi LibreCAD jest dostępny pod adresem *http://wiki.librecad.org/index.php/LibreCAD_users_Manual*.

- **Ramka poleceń** (ang. *Command*) — służy do rysowania figur wyłącznie za pomocą rozkazów. Możemy rysować wykresy bez klikania pasków narzędzi. Szczegółowy opis wykorzystania ramki poleceń można znaleźć pod adresem *http://wiki.librecad.org/index.php/A_short_manual_for_use_from_the_command_line*.
- **Lista warstw** (ang. *Layer list*) — zawiera warstwy używane na bieżącym rysunku. Podstawową operacją w rysowaniu wspomaganym komputerowo jest wykorzystywanie warstw do organizowania rysunku. Szczegółowy opis warstw można znaleźć pod adresem *http://wiki.librecad.org/index.php/Layers*.
- **Bloki** (ang. *Blocks*) — to grupa encji, które można wstawić na tym samym rysunku więcej niż raz z różnymi atrybutami, w różnych miejscach, dla różnej skali i pod różnymi kątami obrotu. Szczegółowy opis zastosowania bloków można znaleźć pod adresem *http://wiki.librecad.org/index.php/Blocks*.
- **Zero absolutne** — początek układu współrzędnych rysunku, punkt (0,0).

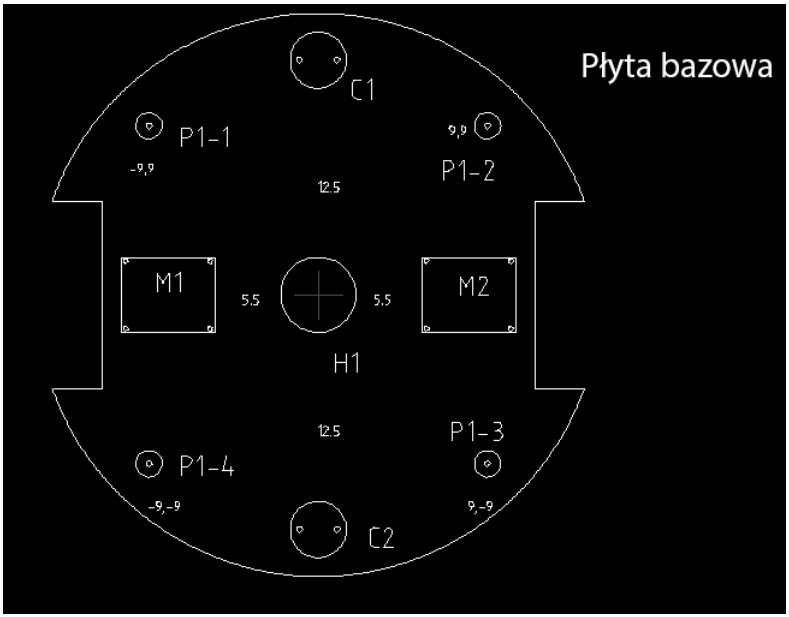
Teraz możemy zacząć szkicowanie. Najpierw ustawimy jednostki dla rysunku. W naszym przykładzie wykorzystamy centymetry. Otwórz program LibreCAD, wybierz polecenie *Edycja/Preferencje aplikacji*. Ustaw opcję *Jednostka* na *Centymetr*, jak pokazano na zrzucie ekranu na następnej stronej.



Rozpocznijmy od projektu płyty bazowej. Płyta bazowa będzie służyć do zamontowania silników, akumulatora oraz płyty sterującej.

Projekt płyty bazowej

Płytę bazową robota zaprezentowano na poniższym rysunku. Zawiera ona miejsce do zamontowania dwóch silników, napędu różnicowego oraz kółek pomocniczych na jej przedniej i tylnej stronie. Silniki na wykresie oznaczono jako **M1** i **M2**, natomiast kółka pomocnicze są reprezentowane jako **C1** i **C2**. Płyta zawiera także cztery bieguny umożliwiające podłączenie do kolejnych płyt. Bieguny są reprezentowane jako **P1-1**, **P1-2**, **P1-3** i **P1-4**. Śruby oznaczono literą **S**. W projekcie tym zastosujemy takie same śruby. W centralnej części płyty znajduje się otwór pozwalający na przeprowadzenie przewodów od silnika do jej górnej części. Płyta jest ścięta po lewej i po prawej stronie tak, aby można było przymocować koła do silnika. Odległość od środka do kółek pomocniczych wynosi **12,5** cm, natomiast odległość od środka do silników wynosi **5,5** cm. Środki biegunów są oddalone od środka płyty o **9** cm w długości i **9** cm w wysokości. Otwory we wszystkich płytach mają identyczne wymiary (patrz rysunek na następnej stronie).



Na diagramie nie podano wymiarów; wymieniono je w poniższej tabeli:

Części	Wymiar (cm) (długość · wysokość) (promień)
M1 i **M2**	5 × 4
C1 i **C2**	Promień = 1,5
S (Śruba)	0,15
P1-1, **P1-2**, **P1-3**, **P1-4**	Zewnętrzny promień 0,7; wysokość 3,5
Sekcje lewego i prawego koła	2,5 × 10
Płyta bazowa	Promień = 15

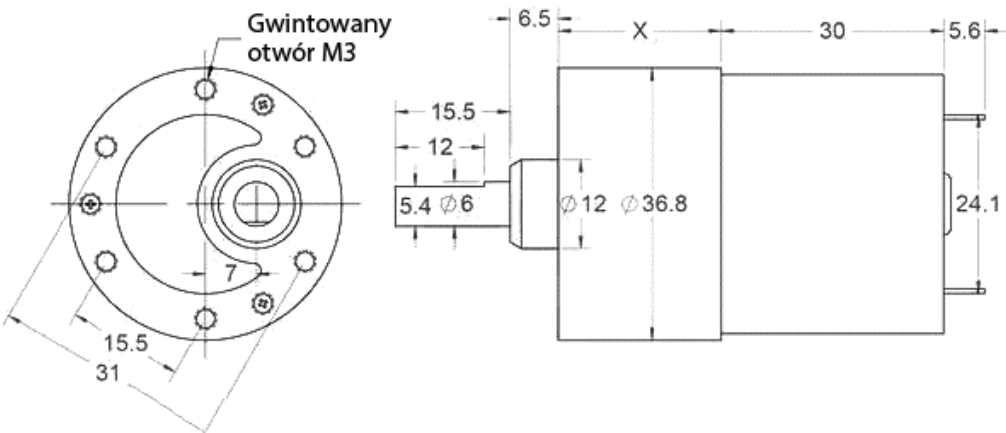
Więcej informacji na temat wymiarów silnika i zacisków będzie później.

Projekt biegunów płyty bazowej

Płyta bazowa ma cztery bieguny pozwalające na zamontowanie następnej płyty. Bieguny mają **3,5** cm długości i promień **0,7** cm. Rozszerzenie do następnej płyty można zrealizować poprzez przymocowywanie do biegunów tulejek. Na szczycie tulejki umieścimy twardy plastik, w którym wykonamy otwór na śrubę. Otwór ten będzie potrzebny do rozszerzenia do płyty górnej. Biegun płyty bazowej i tulejkę pokazano na poniższym rysunku. Każda tulejka ma promień **0,75** cm i długość **15** cm:

Projekt kół, silnika i uchwytów silnika

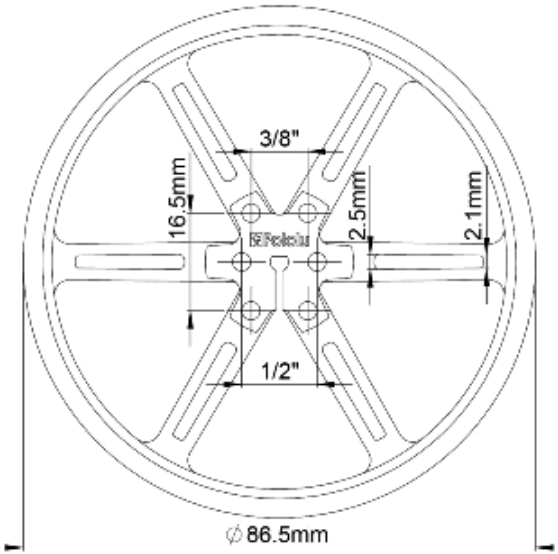
Musimy podjąć decyzję dotyczącą średnicy kół i obliczyć wymagania dla silnika. W naszym przykładzie użyjemy typowego silnika i koła, których będzie można użyć w przypadku wykonania odpowiedniego projektu.

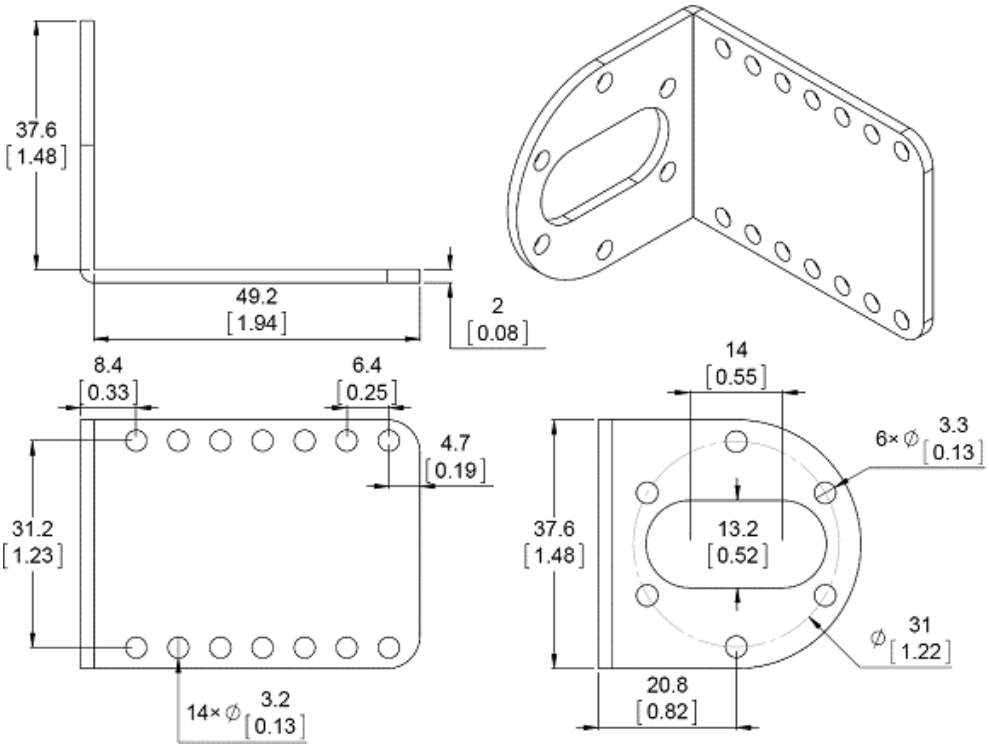


Projekt silnika może być różny w zależności od doboru silnika. Jeśli zajdzie taka konieczność, dobrany silnik można zmienić po wykonaniu symulacji. Wartość **X** na rysunku silnika może być różna w zależności od szybkości i momentu obrotowego. Jest to mechanizm napędowy silnika.

Na pierwszym rysunku na następnej stronie pokazano typowe koło, które można zastosować. Ma ono średnicę **90** cm. Koło o średnicy **86,5** mm po zamontowaniu pierścienia również będzie miało średnicę **90** mm.

Silnik trzeba zamontować na płycie bazowej. Aby można było to zrobić, potrzebujemy uchwytu, który można przykręcić do płyty, a także sposobu połączenia silnika z uchwytem. Na drugim rysunku na następnej stronie zaprezentowano typowy uchwyt, który można wykorzystać do tego celu. Jest to łącznik w kształcie litery L. Jedna część łącznika pozwala na przymocowanie do silnika, natomiast druga do płyty.



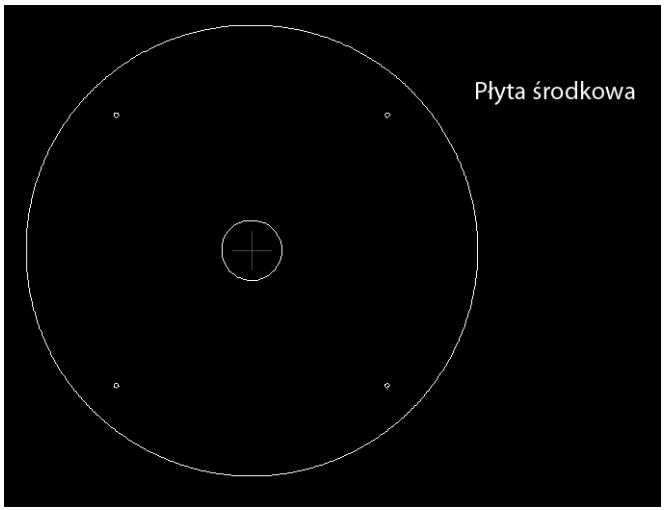


Projekt kół kastora

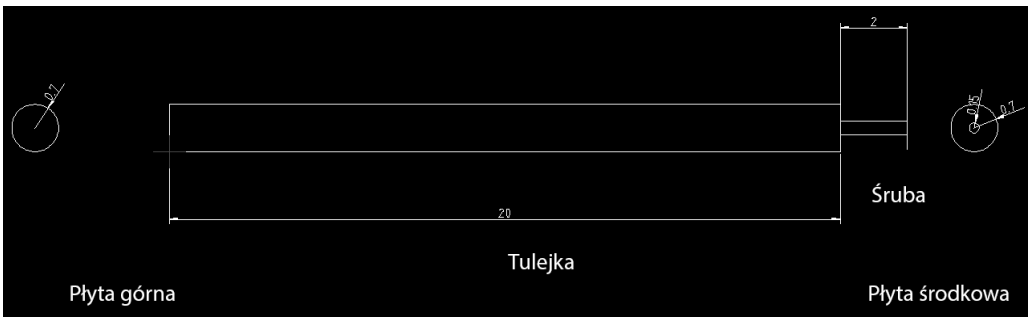
Koła kastora nie wymagają specjalnego projektu. Możemy użyć dowolnych kółek, które będą mogły toczyć się po podłożu. Listę pomocniczych kółek, które można wykorzystać na potrzeby tego projektu, można znaleźć pod adresem *http://www.pololu.com/category/45/pololu-ball-casters*.

Projekt płyty środkowej

Wymiary tej płyty są takie same jak wymiary płyty bazowej. Wielkość śruby także jest podobna.



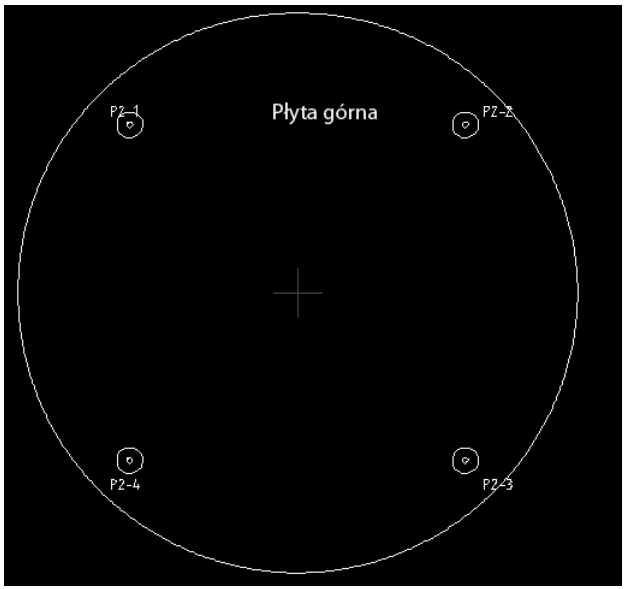
Płyta środkowa jest montowana nad tulejkami przymocowanymi na płycie bazowej. Konstrukcja ta jest połączona za pomocą innej tulejki wyprowadzonej z płyty środkowej. Tulejka z płyty środkowej będzie wyposażona w śrubę na spodzie w celu przymocowania tulejki z płyty bazowej do środkowej oraz otwór pozwalający na zamontowanie płyty górnej. Widok tulejki wychodzącej z płyty środkowej z góry i z boku pokazano na poniższym rysunku:



Tulejka ta łączy płytę środkową z bazową i równocześnie zapewnia możliwość zamontowania płyty górnej.

Projekt płyty górnej

Płyta górna jest podobna do innych płyt. Ma cztery niewielkie bieguny o średnicy 3 cm, podobne do tych na płycie bazowej. Bieguny te będą zamontowane na tulejkach wyprowadzonych z płyty środkowej. Cztery bieguny są podłączone do płyty w sposób pokazany poniżej.

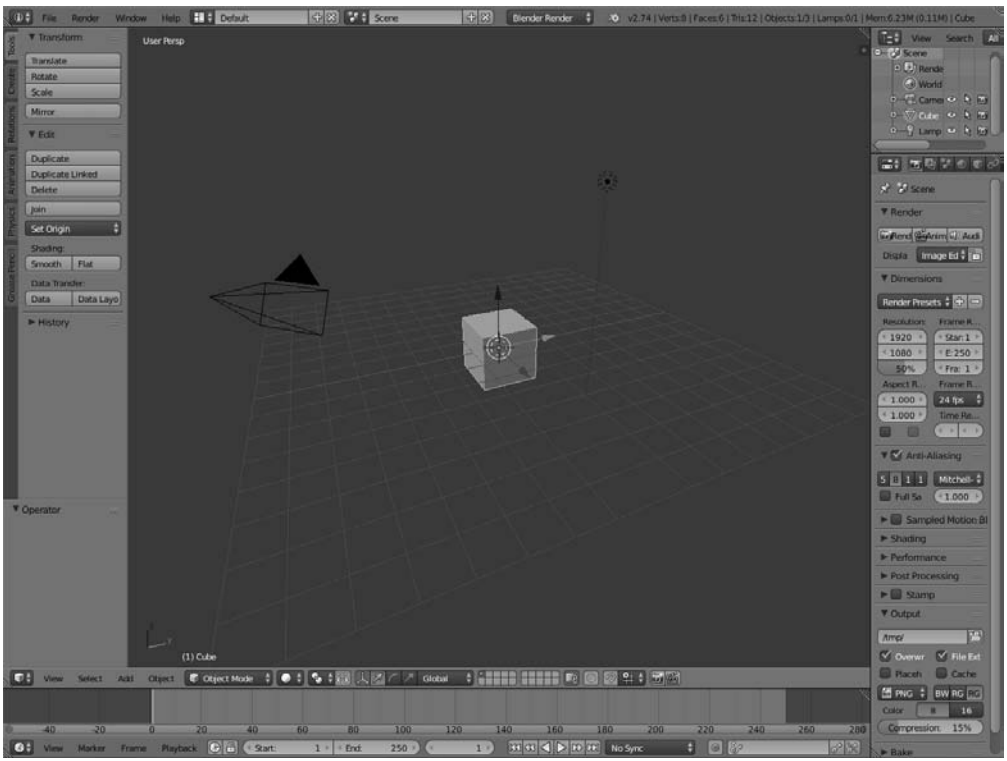


Po zaprojektowaniu płyty górnej projekt szkieletu robota jest prawie skończony. Spróbujmy przyjrzeć się modelowi 3D tego robota zbudowanego z wykorzystaniem programu Blender. Model 3D został zbudowany w celu symulacji, natomiast projekt 2D, który opracowaliśmy, jest podstawą produkcji.

Praca z modelem 3D robota z wykorzystaniem programu Blender

W tym podrozdziale wykonamy projekt 3D modelu robota. Model 3D służy głównie do symulacji. Modelowanie wykonamy za pomocą programu Blender. Należy zaopatrzyć się w wersję programu nowszą niż 2.6. Przewodniki były testowane tylko dla tych wersji.

Na poniższym zrzucie ekranu pokazano przestrzeń roboczą programu Blender oraz narzędzia, które można wykorzystać do pracy z modelami 3D.



Głównym powodem, dla którego skorzystaliśmy tu z Blendera, jest możliwość modelowania robota za pomocą skryptów w Pythonie. Blender jest wyposażony we wbudowany interpreter Pythona oraz edytor skryptów umożliwiający programowanie. Nie będziemy tu omawiać interfejsu użytkownika. Dobry opis Blendera można znaleźć na wymienionej poniżej witrynie internetowej. Aby zapoznać się z interfejsem użytkownika programu Blender, skorzystaj z poniższego linku:

http://www.blender.org/support/tutorials/.

Zacznijmy programować w programie Blender, używając Pythona.

Wykorzystanie skryptów Pythona w programie Blender

Blender napisano głównie w językach C, C++ i Python. Użytkownicy mogą pisać własne skrypty Pythona i korzystać ze wszystkich funkcjonalności programu Blender. Jeżeli jesteś

ekspertem w API Pythona programu Blender, możesz zamodelować całego robota za pomocą skryptów Pythona, zamiast robić to ręcznie.

Blender korzysta z Pythona 3.x. API Pythona programu Blender, ogólnie rzecz biorąc, jest stabilne, choć nadal są dodawane i usprawniane niektóre funkcjonalności. Dokumentacja API Pythona programu Blender jest dostępna pod adresem *http://www.blender.org/documentation/blender_python_api_2_69_7/*.

Spróbujmy omówić interfejs API Pythona programu Blender, który wykorzystamy do utworzenia skryptu na potrzeby modelu naszego robota.

Wprowadzenie do API Pythona programu Blender

API Pythona w programie Blender pozwala na uruchomienie większości funkcji programu. Oto najważniejsze zadania, które mogą być zrealizowane za pośrednictwem tego API:

- Edycja dowolnych danych w Blenderze, na przykład scen, siatek, fragmentów itp.
- Modyfikowanie preferencji użytkownika, map klawiszy i motywów.
- Tworzenie nowych narzędzi programu Blender.
- Rysowanie widoku 3D z wykorzystaniem poleceń OpenGL za pośrednictwem Pythona.

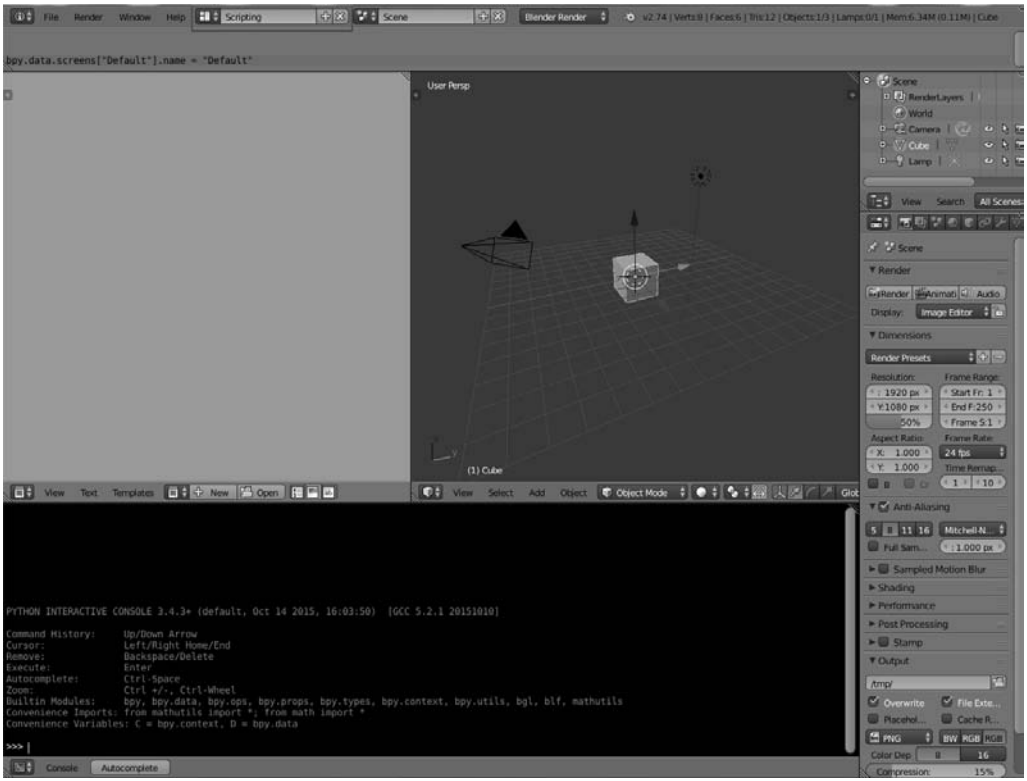
Blender dostarcza interpreterowi Pythona moduł *bpy*. Wystarczy zaimportować ten moduł w skrypcie, aby uzyskać dostęp do danych Blendera oraz klas i funkcji. Skrypty korzystające z danych Blendera muszą zaimportować ten moduł. Oto typowe zadania wykonywane za pomocą *bpy*:

- **Dostęp do kontekstu** — zapewnia dostęp do funkcji interfejsu użytkownika programu Blender za pośrednictwem skryptu (*bpy.context*).
- **Dostęp do danych** — zapewnia dostęp do wewnętrznych danych programu Blender (*bpy.data*).
- **Operatory** — dostęp z poziomu Pythona do wywoływania operatorów, w tym operatorów napisanych w C, Pythonie albo za pomocą makr (*bpy.ops*).

Aby skorzystać ze skryptów w Blenderze, należy zmodyfikować układ ekranu programu. Na zrzucie na następnej stronie pokazano opcję, która pomaga w przełączeniu się do układu *Scripting*.

Po wybraniu zakładki *Scripting* wyświetla się edytor tekstowy oraz konsola Pythona. W edytorze tekstowym możemy programować, używając API Blendera. Możemy także korzystać z poleceń Pythona za pośrednictwem jego konsoli. Kliknij przycisk *New*, aby utworzyć nowy skrypt Pythona; nadaj mu nazwę *robot.py*. Teraz możemy zaprojektować model 3D robota, używając wyłącznie skryptów Pythona. W następnym podrozdziale zamieszczono kompletny skrypt pozwalający zaprojektować model naszego robota. Omówimy ten kod przed próbą jego uruchomienia. Oczekujemy, że Czytelnik zapoznał się z opisem API Pythona dla programu Blender.

Wspomniany kod podzielono na sześć funkcji Pythona. Funkcje te są odpowiedzialne za narysowanie trzech płyt robota, silników i kół, narysowanie czterech tulejek wspierających oraz wyeksportowanie modelu do formatu **3D STL** (ang. *STereoLithography*) w celu symulacji.



Skrypt modelu robota w Pythonie

Poniżej zamieszczono skrypt w Pythonie modelu projektowanego robota:

1. Zanim zaczniemy pisanie skryptu Pythona w programie Blender, musimy zaimportować moduł *bpy*. Moduł *bpy* zawiera wszystkie funkcjonalności Blendera; dostęp do niego można uzyskać tylko za pośrednictwem Blendera:

```
import bpy
```

2. Poniższa funkcja narysuje płytę bazową robota. Funkcja narysuje cylinder o promieniu 5 cm i przytnie części po obu stronach tak, aby można było zamocować silniki. Do tego celu zostanie wykorzystany w Blenderze modyfikator `Boolean`:

```
#Ta funkcja narysuje płytę bazową
def Draw_Base_Plate():
```

3. Poniższe dwie komendy utworzą dwa sześciany o promieniu 0,05 metra po obu stronach płyty bazowej. Celem tych komend jest utworzenie modyfikatora, który wycina fragmenty z płyty bazowej. Tak więc uzyskamy płytę bazową z dwoma wcięciami. Po wykonaniu wcięć z obu stron usuwamy sześciany:

```
bpy.ops.mesh.primitive_cube_add(radius=0.05,location=(0.175,0,0.09))
bpy.ops.mesh.primitive_cube_add(radius=0.05,location=(-0.175,0,0.09))
##################################################
##################################################

#Dodanie płyty bazowej
bpy.ops.mesh.primitive_cylinder_add(radius=0.15,depth=0.005,
location=(0,0,0.09))

#Dodanie modyfikatora różnicy Boolean na podstawie pierwszego sześcianu
bpy.ops.object.modifier_add(type='BOOLEAN')
bpy.context.object.modifiers["Boolean"].operation =
    'DIFFERENCE'
bpy.context.object.modifiers["Boolean"].object =
    bpy.data.objects["Cube"]
bpy.ops.object.modifier_apply(modifier="Boolean")
##################################################
##################################################

#Dodanie modyfikatora różnicy Boolean na podstawie drugiego sześcianu

bpy.ops.object.modifier_add(type='BOOLEAN')
bpy.context.object.modifiers["Boolean"].operation =
    'DIFFERENCE'
bpy.context.object.modifiers["Boolean"].object =
    bpy.data.objects["Cube.001"]
bpy.ops.object.modifier_apply(modifier="Boolean")

##################################################
##################################################

#Anulowanie zaznaczenia cylindra i usunięcie sześcianów
bpy.ops.object.select_pattern(pattern="Cube")
bpy.ops.object.select_pattern(pattern="Cube.001")
bpy.data.objects['Cylinder'].select = False
bpy.ops.object.delete(use_global=False)
```

4. Poniższa funkcja narysuje silniki i koła przymocowane do płyty bazowej:

```
#Ta funkcja narysuje silniki i koła
def Draw_Motors_Wheels():
```

5. Poniższe polecenia narysują cylinder o promieniu 0,045 metra i głębokości 0,01 metra tworzący koła. Utworzone koła zostaną obrócone i przesunięte do wyciętego fragmentu płyty bazowej:

```
# Utworzenie pierwszego koła
bpy.ops.mesh.primitive_cylinder_add(radius=0.045,
    depth=0.01, location=(0,0,0.07))
# Obrót
bpy.context.object.rotation_euler[1] = 1.5708
# Przesunięcie
bpy.context.object.location[0] = 0.135
# Utworzenie drugiego koła
bpy.ops.mesh.primitive_cylinder_add(radius=0.045,
    depth=0.01, location=(0,0,0.07))
# Obrót
bpy.context.object.rotation_euler[1] = 1.5708
# Przesunięcie
bpy.context.object.location[0] = -0.135
```

6. Poniższy kod doda dwie atrapy silników do płyty bazowej. Wymiary silników zostały podane w projekcie 2D. Silnik jest w zasadzie cylindrem, który zostanie obrócony i zamontowany na płycie bazowej:

```
# Dodanie silników

bpy.ops.mesh.primitive_cylinder_add(radius=0.018,
    depth=0.06, location=(0.075,0,0.075))
bpy.context.object.rotation_euler[1] = 1.5708

bpy.ops.mesh.primitive_cylinder_add(radius=0.018,
    depth=0.06, location=(-0.075,0,0.075))
bpy.context.object.rotation_euler[1] = 1.5708
```

7. Poniższy kod, podobnie jak w przypadku modelu silnika, doda do silników wał. Wał także jest cylindrem, który zostanie obrócony i umieszczony wewnątrz modelu silnika:

```
# Dodanie wału silnika
bpy.ops.mesh.primitive_cylinder_add(radius=0.006,
    depth=0.04, location=(0.12,0,0.075))
bpy.context.object.rotation_euler[1] = 1.5708

bpy.ops.mesh.primitive_cylinder_add(radius=0.006,
    depth=0.04, location=(-0.12,0,0.075))
bpy.context.object.rotation_euler[1] = 1.5708

######################################################
######################################################
```

8. Poniższy kod doda dwa koła kastora do płyty bazowej. W tym momencie dodajemy cylinder, który pełni funkcję koła. Podczas symulacji możemy go przypisać jako koło:

```
# Dodawanie koła kastora
bpy.ops.mesh.primitive_cylinder_add(radius=0.015,
    depth=0.05, location=(0,0.125,0.065))
```

```
bpy.ops.mesh.primitive_cylinder_add(radius=0.015,
    depth=0.05, location=(0,-0.125,0.065))
```

9. Poniższy kod dodaje atrapę sensora Kinect:

```
# Dodanie sensora Kinect

bpy.ops.mesh.primitive_cube_add(radius=0.04,
    location=(0,0,0.26))
```

10. Ta funkcja narysuje płytę środkową robota:

```
# Rysowanie płyty środkowej
def Draw_Middle_Plate():
    bpy.ops.mesh.primitive_cylinder_add(radius=0.15,
        depth=0.005, location=(0,0,0.22))

# Dodawanie płyty górnej
def Draw_Top_Plate():
    bpy.ops.mesh.primitive_cylinder_add(radius=0.15,
        depth=0.005, location=(0,0,0.37))
```

11. Ta funkcja narysuje wszystkie cztery tulejki wspierające dla wszystkich trzech płyt:

```
# Dodawanie tulejek wspierających
def Draw_Support_Tubes():
##################################################################
##########################

    #  Cylindry
    bpy.ops.mesh.primitive_cylinder_add(radius=0.007,
        depth=0.30, location=(0.09,0.09,0.23))
    bpy.ops.mesh.primitive_cylinder_add(radius=0.007,
        depth=0.30, location=(-0,09,0.09,0.23))
    bpy.ops.mesh.primitive_cylinder_add(radius=0.007,
        depth=0.30, location=(-0.09,-0.09,0.23))
    bpy.ops.mesh.primitive_cylinder_add(radius=0.007,
        depth=0.30, location=(0,09,-0.09,0.23))
```

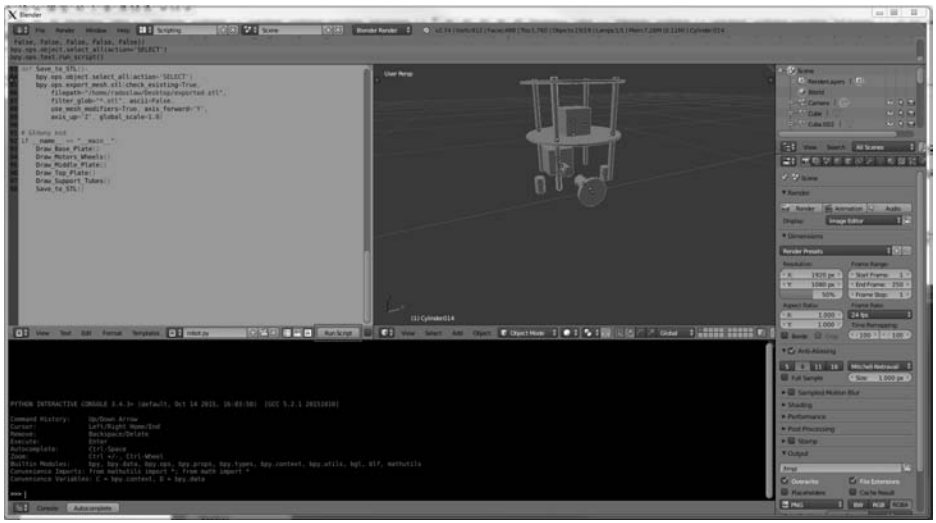
12. Ta funkcja wyeksportuje zaprojektowanego robota do formatu STL. Przed uruchomieniem skryptu trzeba zmienić ścieżkę pliku STL:

```
# Eksport do STL
def Save_to_STL():
    bpy.ops.object.select_all(action='SELECT')
# Należy ustawić ścieżkę filepath na istniejący folder
    bpy.ops.export_mesh.stl(check_existing=True,
        filepath="/home/radoslaw/Desktop/exported.stl",
        filter_glob="*.stl", ascii=False,
        use_mesh_modifiers=True, axis_forward='Y',
        axis_up='Z', global_scale=1.0)

# Główny kod
```

```
if __name__ == "__main__":
    Draw_Base_Plate()
    Draw_Motors_Wheels()
    Draw_Middle_Plate()
    Draw_Top_Plate()
    Draw_Support_Tubes()
    Save_to_STL()
```

13. Po wprowadzeniu kodu do edytora tekstu uruchamiamy skrypt poprzez kliknięcie przycisku *Run Script*, jak pokazano na poniższym zrzucie ekranu. Wynikowy model 3D będzie wyświetlony w widoku 3D w programie Blender. Jeśli zajrzymy na pulpit, to znajdziemy tam plik *exported.stl*, który wykorzystamy do symulacji:

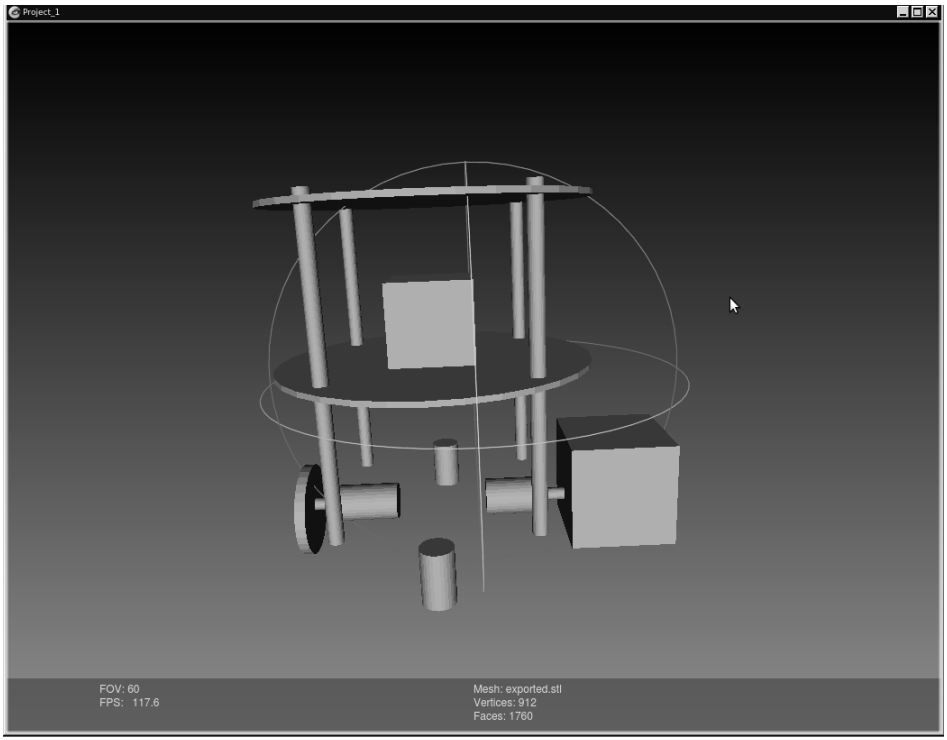


Plik *exported.stl* można otworzyć w programie MeshLab. Oto zrzut ekranu z programu MeshLab (patrz rysunek na następnej stronie).

Pobieranie przykładowego kodu

Przykładowe pliki kodu można pobrać z serwisu FTP wydawnictwa Helion: *ftp://ftp.helion.pl/przyklady/naropy.zip*.

Przykładowe pliki kodu dołączonego do oryginalnej wersji książki, a także pliki kodu do innych książek wydanych przez Packt, które Czytelnicy kupili na portalu tego wydawnictwa, można pobrać, logując się na swoim koncie pod adresem *http://www.packtpub.com/*. Jeżeli nabyłeś tę książkę gdzie indziej, możesz zarejestrować się na witrynie *http://www.packtpub.com/support*. Pliki zostaną przesłane na Twój e-mail.



Pytania

1. Czym jest modelowanie robota i jakie przynosi korzyści?
2. Jaką rolę odgrywa model 2D robota?
3. Jaką rolę odgrywa model 3D robota?
4. Na czym polega wyższość stosowania skryptów Pythona nad projektowaniem ręcznym?

Podsumowanie

Niniejszy rozdział poświęcono projektowaniu mechaniki robota. Zawiera obliczenia parametrów robota i projekt jego szkieletu. Podczas projektowania najpierw należy określić wymagania wstępne. Gdy będą gotowe, możemy ustalić wymagania dla komponentów wykorzystywanych w robocie. Następnie projektujemy szkielet robota według określonych wymagań. Szkielet wymaga projektu 2D wszystkich części potrzebnych do budowy robota. Po wykonaniu projektu 2D pokazano, jak zbudować model 3D za pomocą programu Blender i skryptu w Pythonie.

Model 3D zbudowaliśmy z wykorzystaniem wymiarów, których użyliśmy w projekcie 2D. Omówiono także skrypt Pythona programu Blender do budowy całego modelu 3D. Opracowaliśmy projekt, który może zostać użyty do produkcji robota, a także utworzyliśmy model 3D dla celów symulacji. W następnym rozdziale zostanie omówiona symulacja utworzonego modelu robota oraz będą przedstawione wybrane popularne narzędzia do przeprowadzania symulacji.

Skorowidz

A

B

C

S

T

U

V

W

X

Z

Ż

Nauka robotyki z językiem Python

Roboty wkraczają do różnych dziedzin naszego życia, więc robotyka nabiera coraz większego znaczenia. Nauka o robotach, ich budowaniu i programowaniu jest dość złożoną, ale fascynującą dziedziną. Jej opanowanie wymaga wysiłku, jednak aby zaprojektować łatwy do wykorzystania interfejs, wystarczy posłużyć się kilkoma programami narzędziowymi oraz językiem Python. W ten sposób można zaprojektować zachowania robota, określić, w jaki sposób będzie zmierzał do celu, reagował na sygnały otaczającego świata, czy sprawić, by oczekiwał na instrukcje.

Dzięki tej książce można się nauczyć, jak z wykorzystaniem języka Python oraz kilku popularnych frameworków stosowanych w robotyce, takich jak system ROS, budować autonomiczne roboty mobilne. Omówiono w niej inne frameworki programistyczne, w tym również te dla Pythona. Aby równocześnie pokazać praktyczne wykorzystanie przedstawianego materiału, omówiono krok po kroku proces budowania robota-służącego ChefBot, który na przykład może podawać posiłki w domu, hotelu czy restauracji.

W tej książce przedstawiono:

- zwięzłe podstawy robotyki i zasady projektowania oprogramowania robotów
- aspekty projektowania CAD 2D i 3D z wykorzystaniem programów LibreCAD i Blender
- budowanie modeli 3D z wykorzystaniem API Blender dla Pythona
- zagadnienia sprzętowej warstwy projektowania robota
- zagadnienia testowania i kalibrowania robota

Lentin Joseph — inżynier elektroniki, entuzjasta robotyki i ekspert w dziedzinie systemów wbudowanych. Szczególnie interesuje się robotyką, przetwarzaniem obrazu i zastosowaniem języka Python w programowaniu robotów. Jest również znawcą wielu platform oprogramowania robotów, takich jak system ROS (ang. Robot Operating System), V-REP i Actin. Biegle posługuje się bibliotekami przetwarzania obrazu, w tym OpenCV, OpenNI i PCL. Specjalizuje się również w dziedzinie projektowania 3D i programowania systemów wbudowanych na platformach Arduino i Launchpad Stellaris. Jest właścicielem firmy Qbotics Labs zajmującej się rozwijaniem robotyki i jej zastosowaniami w wielu dziedzinach.

[PACKT] open source PUBLISHING community experience distilled

Helion

44317 numer katalogowy

księgarnia internetowa

http://helion.pl

zamówienia telefoniczne

0 801 339900

0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ

KOD KORZYŚCI

ISBN 978-83-283-2345-2

cena: 49,00 zł