



Technologia i rozwiązania

Nauka robotyki z językiem Python

Przekonaj się, jak fascynujące jest
programowanie robotów!



Lentin Joseph

[PACKT] open source*
PUBLISHING community experience distilled

Tytuł oryginału: Learning Robotics Using Python

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-2345-2

Copyright © Packt Publishing 2015.

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

Polish edition copyright © 2016 by Helion SA.
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/naropy>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

0 autorze	11
0 recenzentach	13
Przedmowa	15
Rozdział 1. Wprowadzenie do robotyki	19
Czym jest robot?	20
Historia terminu „robot”	20
Współczesna definicja robota	22
Skąd pochodzą roboty?	25
Co można znaleźć w robocie?	28
Ciało fizyczne	29
Sensory	29
Efektory	29
Kontrolery	30
Jak budujemy robota?	31
Sterowanie reaktywne	31
Sterowanie hierarchiczne (deliberatywne)	31
Sterowanie hybrydowe	32
Podsumowanie	33
Rozdział 2. Projekt mechaniki robota usługowego	35
Wymagania dla robota usługowego	36
Mechanizm napędowy robota	36
Wybór silników i kół	36
Podsumowanie projektu	38
Projekt podwozia robota	38

Instalacja oprogramowania LibreCAD, Blender i MeshLab	40
Instalacja programu LibreCAD	40
Instalacja programu Blender	40
Instalacja programu MeshLab	41
Tworzenie rysunku CAD 2D robota z wykorzystaniem programu LibreCAD	41
Projekt płyty bazowej	43
Projekt biegunów płyty bazowej	44
Projekt kół, silnika i uchwytów silnika	45
Projekt kół kastora	47
Projekt płyty środkowej	47
Projekt płyty górnej	48
Praca z modelem 3D robota z wykorzystaniem programu Blender	48
Wykorzystanie skryptów Pythona w programie Blender	49
Wprowadzenie do API Pythona programu Blender	50
Skrypt modelu robota w Pythonie	51
Pytania	56
Podsumowanie	56
Rozdział 3. Symulacja robota z wykorzystaniem systemów ROS i Gazebo	59
Symulacja robota	59
Modelowanie matematyczne robota	62
Wprowadzenie do ROS i Gazebo	69
Instalacja systemu ROS Indigo w systemie Ubuntu 14.04.2	72
Symulacja robotów ChefBot i TurtleBot w środowisku hotelu	96
Pytania	100
Podsumowanie	101
Rozdział 4. Projektowanie sprzętu robota ChefBot	103
Specyfikacje sprzętu robota ChefBot	104
Schemat blokowy robota	104
Silnik i enkoder	104
Sterownik silnika	106
Płyta wbudowanego kontrolera	109
Sensory ultradźwiękowe	110
Inercyjna jednostka pomiarowa	112
Kinect	113
Centralna jednostka obliczeniowa	114
Głośniki i mikrofon	115
Zasilacz (akumulator)	115
Opis działania sprzętu robota ChefBot	115
Pytania	118
Podsumowanie	118

Rozdział 5. Aktuatory i enkodery kół	119
Podłączenie motoreduktora DC z kontrolerem Tiva C LaunchPad	120
Robot kołowy z napędem różnicowym	122
Instalacja IDE Energia	123
Kod interfejsu	126
Podłączenie enkodera kwadraturowego do kontrolera Tiva C LaunchPad	130
Przetwarzanie danych enkodera	131
Kod interfejsu z enkoderem kwadraturowym	134
Praca z aktuatorami Dynamixel	137
Pytania	140
Podsumowanie	141
Rozdział 6. Wykorzystanie sensorów	143
Ultradźwiękowe sensory odległości	143
Podłączenie modułu HC-SR04 z kontrolerem Tiva C LaunchPad	144
Sensory odległości na podczerwień	149
Inercyjne jednostki pomiarowe (IMU)	152
Nawigacja inercyjna	152
Połączenie sensora MPU 6050 z kontrolerem Tiva C LaunchPad	154
Kod interfejsu w środowisku Energia	156
Kod interfejsu sensora MPU 6050 z kontrolerem Launchpad z wykorzystaniem DMP w środowisku Energia	159
Pytania	164
Rozdział 7. Programowanie sensorów wizji z wykorzystaniem języka Python i systemu ROS	165
Lista sensorów wizji dla robota i bibliotek przetwarzania obrazu	166
Wprowadzenie do OpenCV, OpenNI oraz PCL	169
Czym jest OpenCV?	170
Co to jest OpenNI?	174
Co to jest PCL?	175
Programowanie sensora Kinect za pomocą języka Python z wykorzystaniem ROS, OpenCV i OpenNI	176
Jak uruchomić sterownik OpenNI?	176
Interfejs ROS do biblioteki OpenCV	177
Przetwarzanie chmur punktów z wykorzystaniem sensora Kinect, systemu ROS oraz bibliotek OpenNI i PCL	182
Otwieranie urządzenia i generowanie chmury punktów	182
Konwersja chmury punktów na dane skanu laserowego	183
Wykorzystanie techniki SLAM z systemem ROS i sensorem Kinect	185
Pytania	186
Podsumowanie	186

Rozdział 8. Rozpoznawanie i synteza mowy z wykorzystaniem systemu ROS i języka Python	187
Rozpoznawanie mowy	188
Schemat blokowy systemu rozpoznawania mowy	188
Biblioteki rozpoznawania mowy	189
Windows Speech SDK	190
Synteza mowy	190
Biblioteki syntezy mowy	191
Rozpoznawanie i synteza mowy w systemie Ubuntu 14.04.2 z wykorzystaniem języka Python	192
Konfiguracja biblioteki Pocket Sphinx i jej wrapperów dla języka Python w systemie Ubuntu 14.04.2	192
Wykorzystanie wrappera biblioteki Pocket Sphinx do języka Python w systemie Ubuntu 14.04.2	193
Wyjście	194
Rozpoznawanie mowy w czasie rzeczywistym z wykorzystaniem biblioteki Pocket Sphinx, frameworka GStreamer i języka Python w systemie Ubuntu 14.04.2	195
Rozpoznawanie mowy za pomocą narzędzia Julius i języka Python w systemie Ubuntu 14.04.2	198
Instalacja narzędzia rozpoznawania mowy Julius i modułu języka Python	198
Kod klienta Python-Julius	199
Poprawianie dokładności rozpoznawania mowy w Pocket Sphinx i Julius	200
Konfiguracja syntezy eSpeak i Festival w systemie Ubuntu 14.04.2	201
Rozpoznawanie i synteza mowy z wykorzystaniem języka Python w systemie Windows	202
Instalacja pakietu Speech SDK	202
Rozpoznawanie mowy w systemie ROS Indigo z wykorzystaniem języka Python	203
Instalacja pakietu pocketsphinx w systemie ROS Indigo	203
Synteza mowy w systemie ROS Indigo z wykorzystaniem języka Python	204
Pytania	206
Podsumowanie	206
Rozdział 9. Zastosowanie mechanizmów sztucznej inteligencji w robocie ChefBot za pośrednictwem języka Python	207
Schemat blokowy systemu komunikacji w robocie ChefBot	208
Wprowadzenie do AIML	209
Wprowadzenie do znaczników AIML	209
Wprowadzenie do PyAIML	212
Instalacja modułu PyAIML w systemie Ubuntu 14.04.2	213
Instalacja modułu PyAIML z kodu źródłowego	213
Przetwarzanie formatu AIML z poziomu języka Python	213
Ładowanie pojedynczego pliku AIML za pośrednictwem argumentu wiersza polecenia	215
Wykorzystanie plików AIML robota A.L.I.C.E.	216
Ładowanie plików AIML do pamięci	216
Ładowanie plików AIML i zapisywanie ich do plików .brn	217
Ładowanie plików AIML i plików .brn za pomocą metody bootstrap	218

Integracja biblioteki PyAIML z systemem ROS	219
aiml_server.py	219
aiml_client.py	220
aiml_tts_client.py	221
aiml_speech_recog_client.py	221
start_chat.launch	223
start_tts_chat.launch	223
start_speech_chat.launch	223
Pytania	225
Podsumowanie	225
Rozdział 10. Integracja sprzętu i programowanie robota ChefBot z wykorzystaniem języka Python	227
<hr/>	
Budowa sprzętu robota ChefBot	228
Konfiguracja komputera PC robota ChefBot i ustawienie pakietów systemu ROS	232
Interfejs sensorów robota ChefBot z kontrolerem Tiva C LaunchPad	233
Wbudowany kod robota ChefBot	234
Sterownik systemu ROS dla robota ChefBot w języku Python	236
Pliki startowe systemu ROS dla robota ChefBot	241
Korzystanie z plików startowych i węzłów robota ChefBot z poziomu języka Python	242
Wykorzystanie algorytmu SLAM w systemie ROS	
do zbudowania mapy pomieszczenia	248
Lokalizacja i nawigacja w systemie ROS	250
Pytania	251
Podsumowanie	251
Rozdział 11. Projektowanie GUI dla robota za pomocą biblioteki Qt oraz języka Python	253
<hr/>	
Instalacja frameworka Qt w systemie Ubuntu 14.04.2 LTS	254
Korzystanie z wrappera frameworka Qt dla języka Python	254
PyQt	255
PySide	255
Korzystanie z wrapperów PyQt oraz PySide	256
Wprowadzenie do programu Qt Designer	256
Sygnały i gniazda Qt	258
Konwersja pliku UI na kod w języku Python	260
Dodawanie definicji gniazda do kodu PyQt	260
Uruchomienie aplikacji GUI Witaj świecie	262
Interfejs GUI sterowania robotem ChefBot	263
Instalacja i korzystanie z narzędzia rqt w systemie Ubuntu 14.04.2 LTS	269
Pytania	271
Podsumowanie	271

Rozdział 12. Kalibracja i testowanie robota ChefBot	273
Kalibrowanie sensora Xbox Kinect z wykorzystaniem systemu ROS	273
Kalibracja kamery RGB sensora Kinect	274
Kalibracja kamery podczerwieni sensora Kinect	277
Kalibracja odometrii kół	279
Analiza błędów odometrii kół	279
Korekcja błędów	280
Kalibracja sensora MPU 6050	281
Testowanie robota za pomocą interfejsu GUI	282
Zalety i wady nawigacji w systemie ROS	285
Pytania	285
Podsumowanie	285
Skorowidz	287

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.

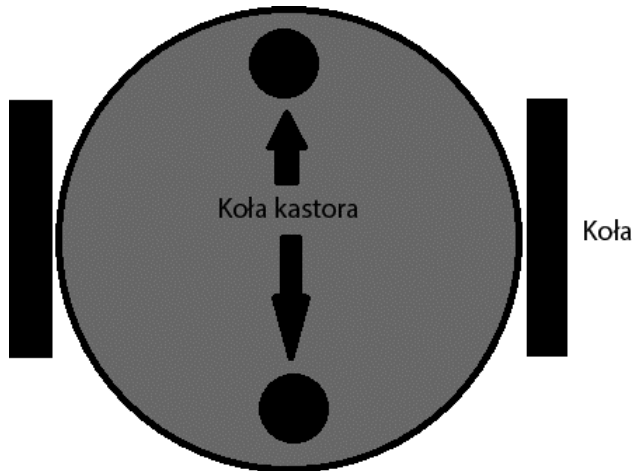
Mechanizm napędowy robota

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:

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

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

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 = ($W = mg$) = ($\sim 100 \text{ N} + \sim 50 \text{ N}$)
 $W = \sim 150 \text{ N}$, 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 jezdnią 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 wspomaganego 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 (modele 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

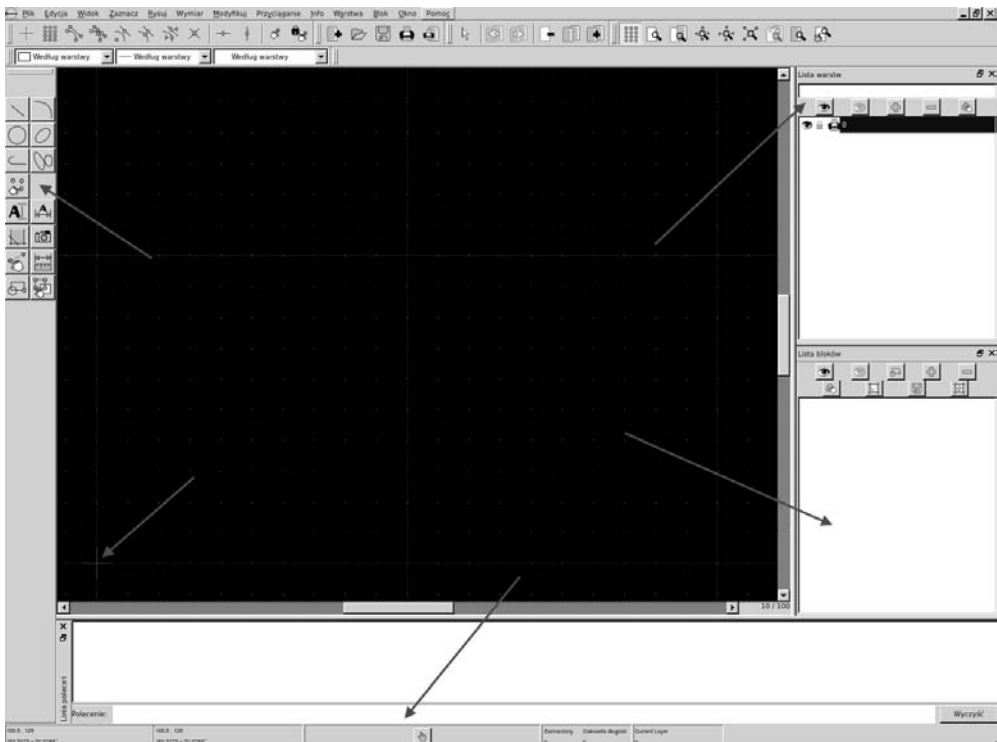
MeshLab jest dostępny dla wszystkich platform OS. Pod adresem <http://meshlab.sourceforge.net/> dostępne są łącza pobierania skompilowanych binariów i kodu źródłowego programu MeshLab.

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

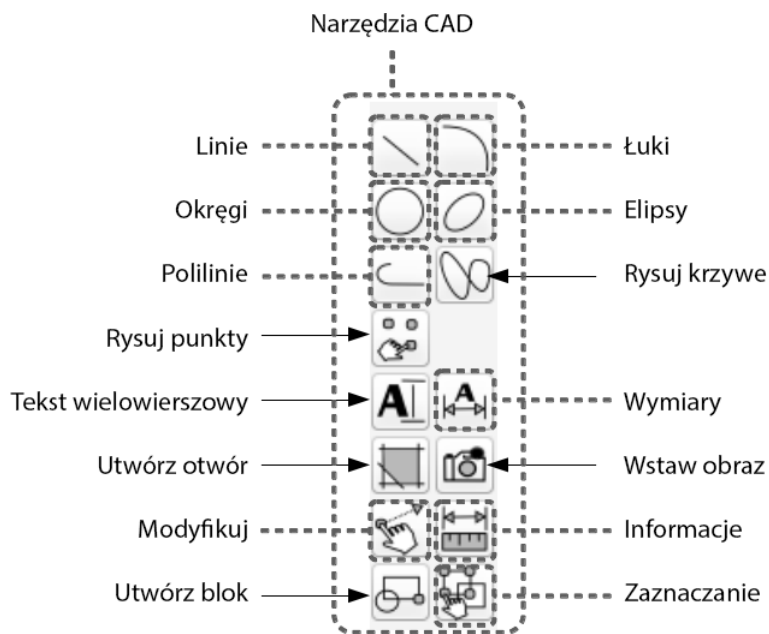
```
$sudo apt-get install meshlab
```

Tworzenie rysunku CAD 2D robota z wykorzystaniem programu LibreCAD

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



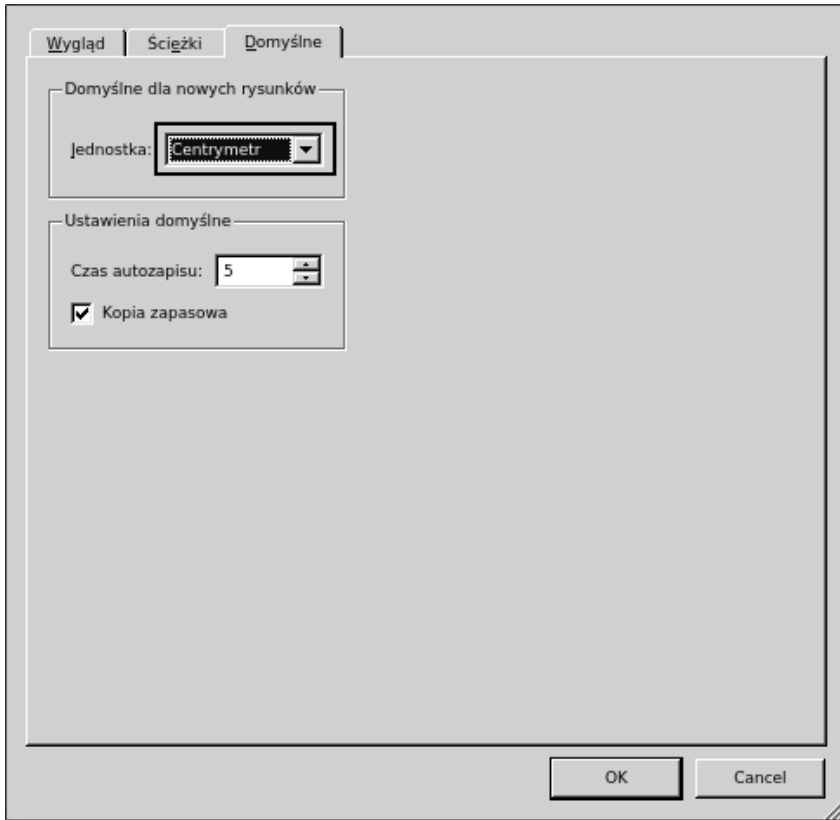
Pasek narzędzi CAD ma komponenty niezbędne do narysowania modelu. Szczegółowy widok paska narzędzi CAD zamieszczono na poniższym zrzucie ekranu:



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).

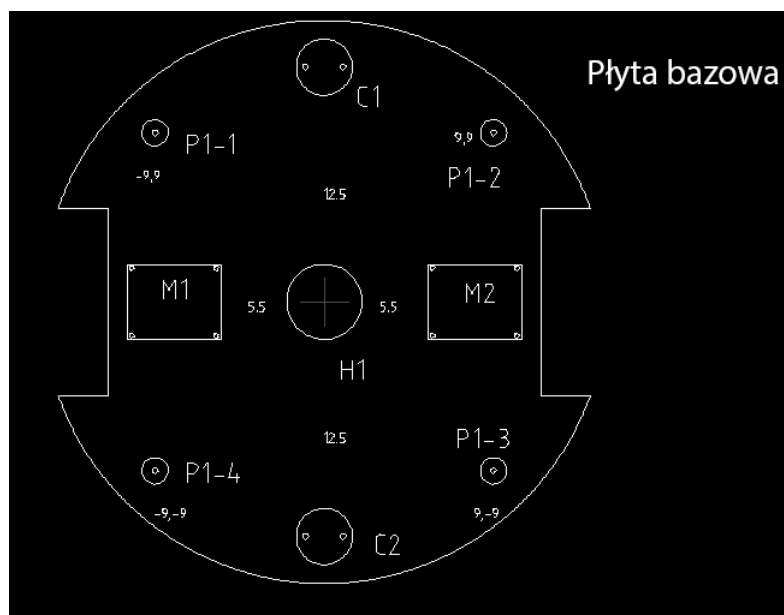
Teraz możemy zacząć szkicowanie. Najpierw ustawimy jednostki dla rysunku. W naszym przykładzie wykorzystamy centymetry. Otwórz program LibreCAD, wybierz polecenie *Edycja/Preferencje aplikacji*. Ustaw opcję *Jednostka* na *Centymetr*, jak pokazano na zrzucie ekranu na następnej stronie.



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.

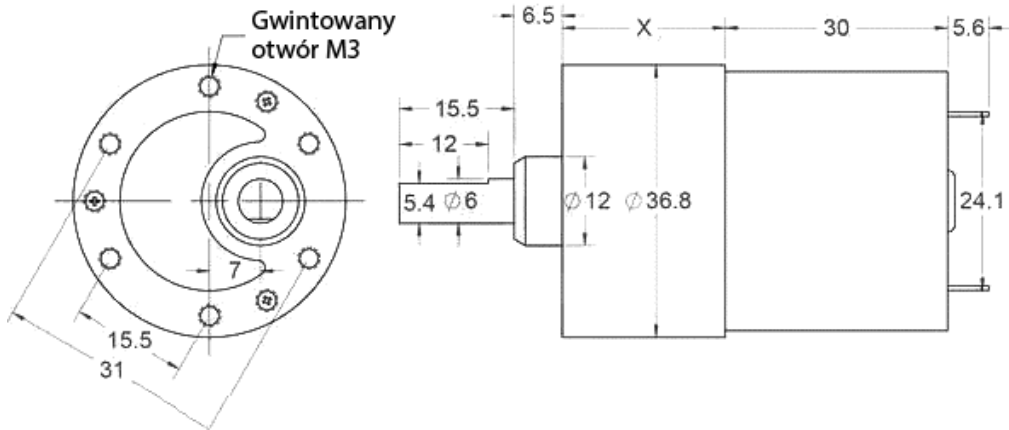
Projekt biegunów płyty bazowej

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

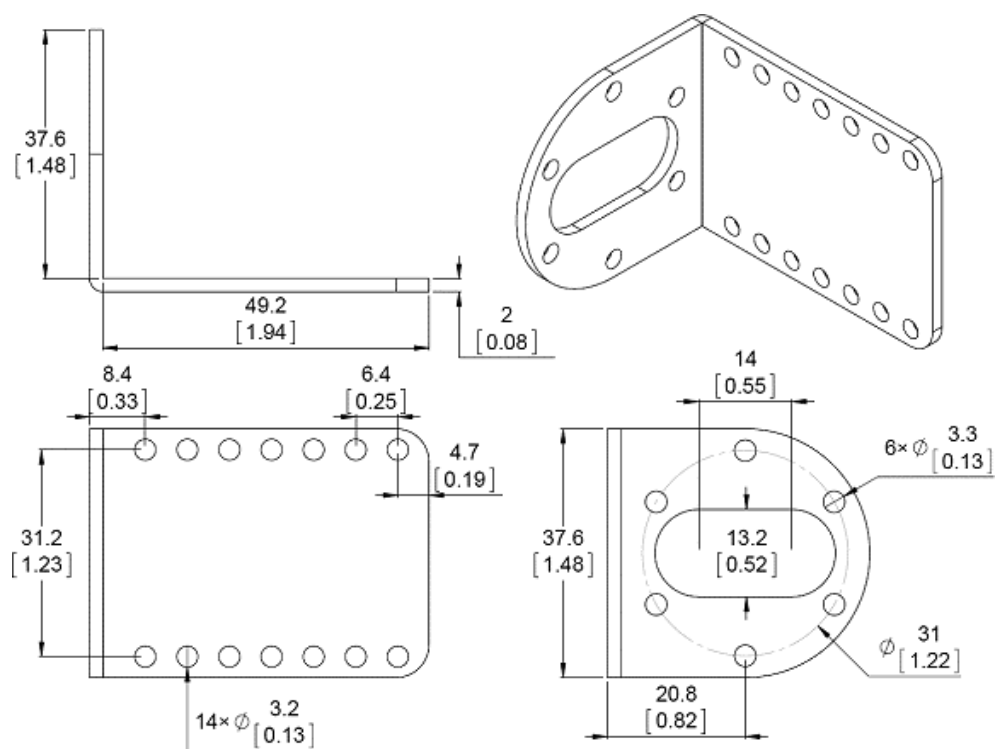
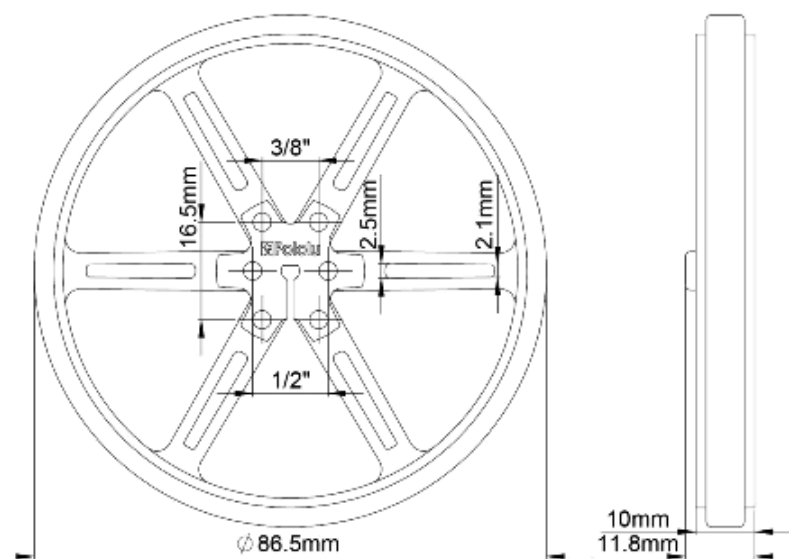
Musimy podjąć decyzję dotyczącą średnicy kół i obliczyć wymagania dla silnika. W naszym przykładzie użyjemy typowego silnika i koła, których będzie można użyć w przypadku wykonania odpowiedniego projektu.



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 mm. 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.

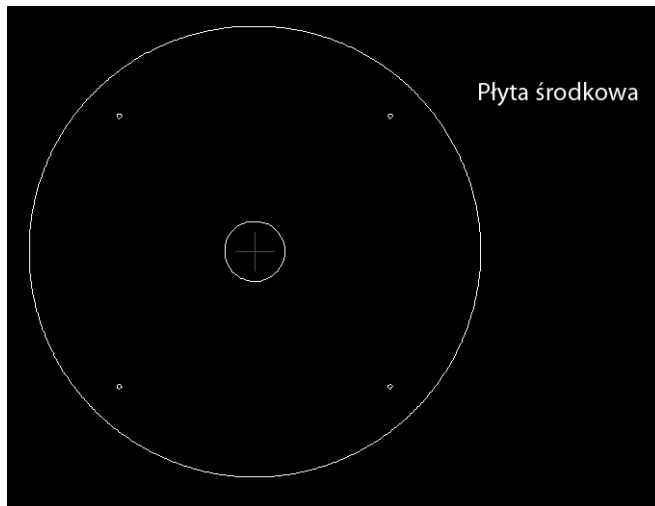


Projekt kół kastora

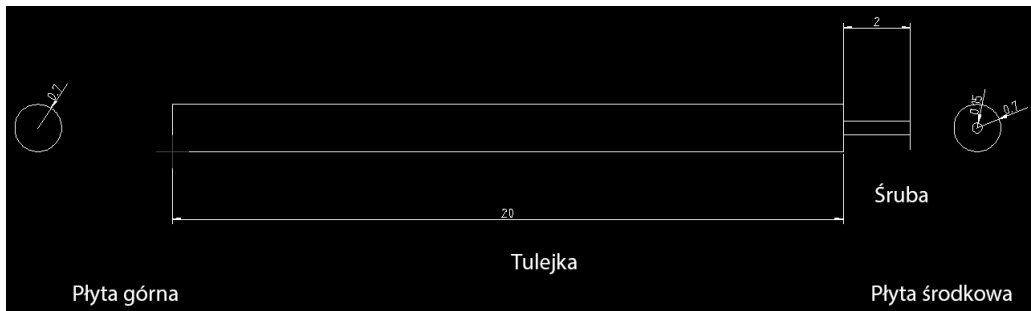
Koła kastora nie wymagają specjalnego projektu. Możemy użyć dowolnych kółek, które będą mogły toczyć się po podłożu. Listę pomocniczych kółek, które można wykorzystać na potrzeby tego projektu, można znaleźć pod adresem <http://www.pololu.com/category/45/pololu-ball-casters>.

Projekt płyty środkowej

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



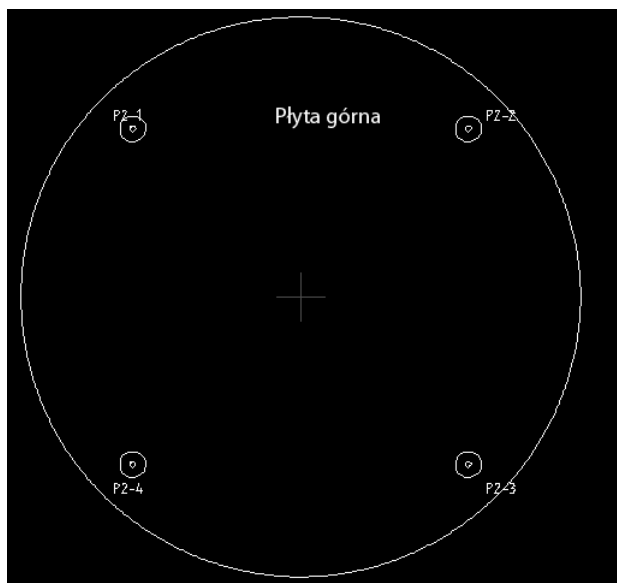
Płyta środkowa jest montowana nad tulejkami przymocowanymi na płycie bazowej. Konstrukcja ta jest połączona za pomocą innej tulejki wyprowadzonej z płyty środkowej. Tulejka z płyty środkowej będzie wyposażona w śrubę na spodzie w celu przymocowania tulejki z płyty bazowej do środkowej oraz otwór pozwalający 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.

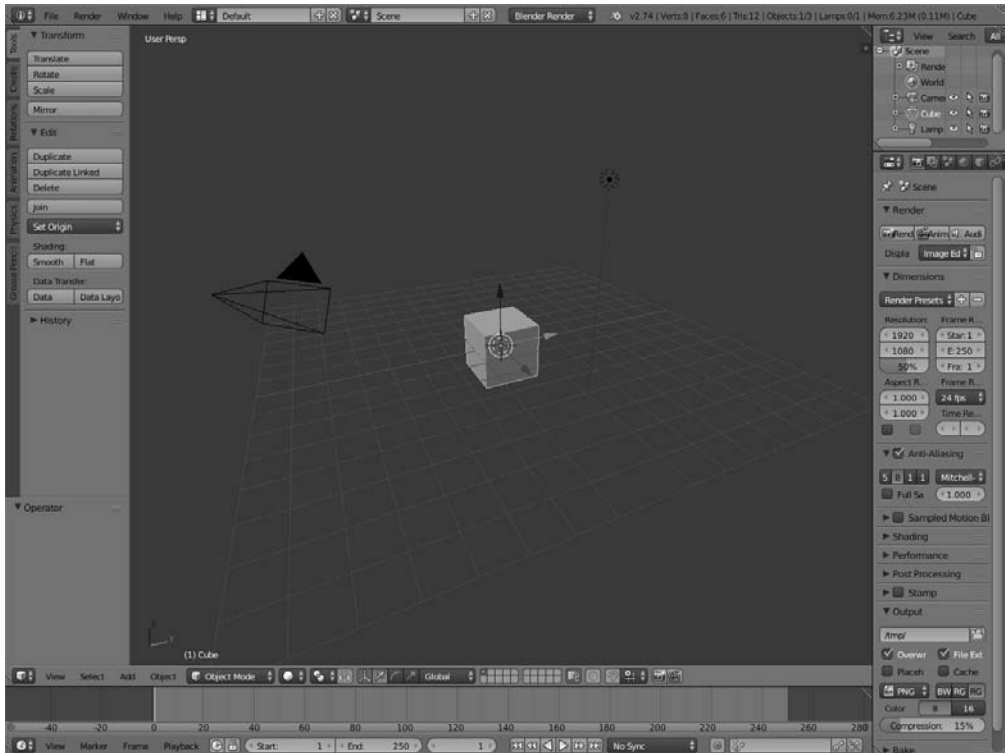


Po zaprojektowaniu płyty górnej projekt szkieletu robota jest prawie skończony. Spróbujmy przyjrzeć się modelowi 3D tego robota zbudowanego z wykorzystaniem programu Blender. Model 3D został zbudowany w celu symulacji, natomiast projekt 2D, który opracowaliśmy, jest podstawą produkcji.

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 motyłów.
- Tworzenie nowych narzędzi programu Blender.
- Rysowanie widoku 3D z wykorzystaniem poleceń OpenGL za pośrednictwem Pythona.

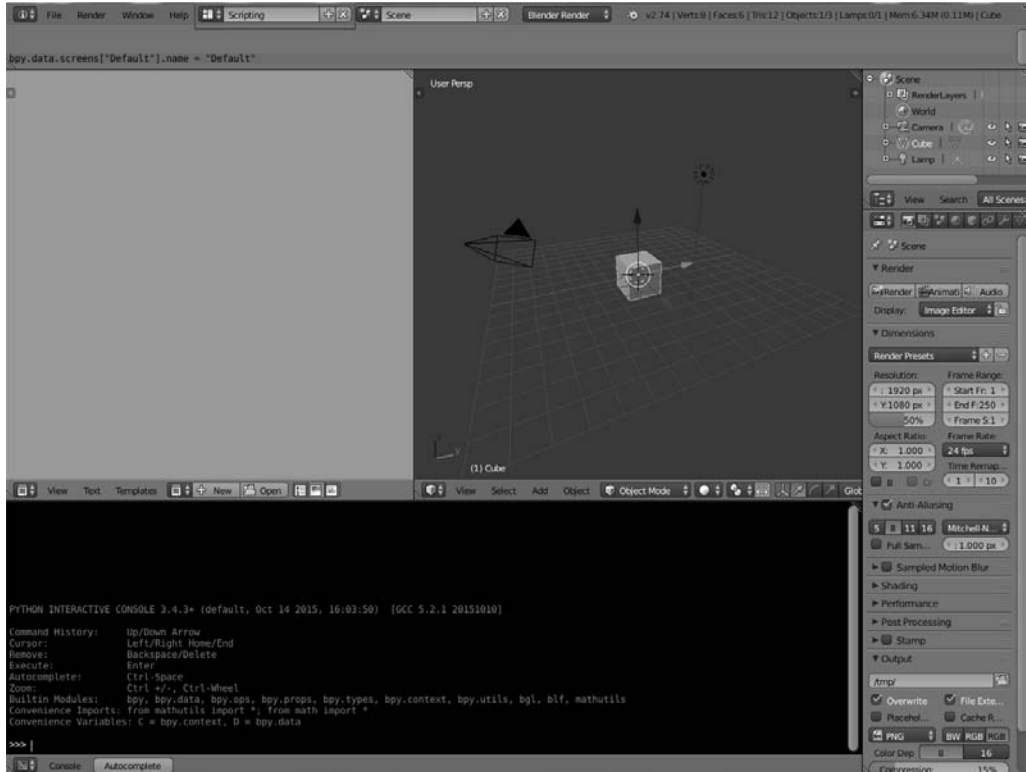
Blender dostarcza interpreterowi Pythona moduł *bpy*. Wystarczy zaimportować ten moduł w skrypcie, aby uzyskać dostęp do danych Blendera oraz klas i funkcji. Skrypty korzystające z danych Blendera muszą zaimportować ten moduł. Oto typowe zadania wykonywane za pomocą *bpy*:

- **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 cylindera 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)
```

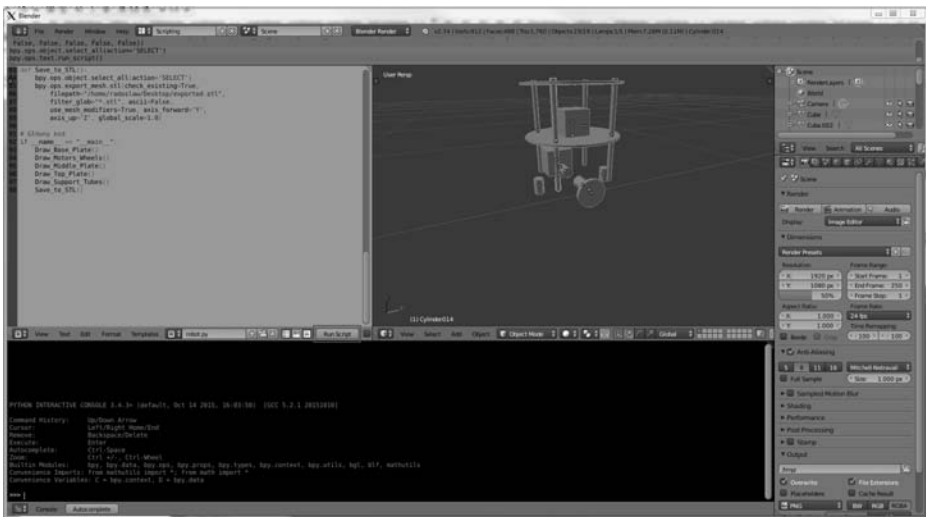
```
# Główny kod
```

```

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 zajrzemy na pulpit, to znajdziemy tam plik *exported.stl*, który wykorzystamy do symulacji:

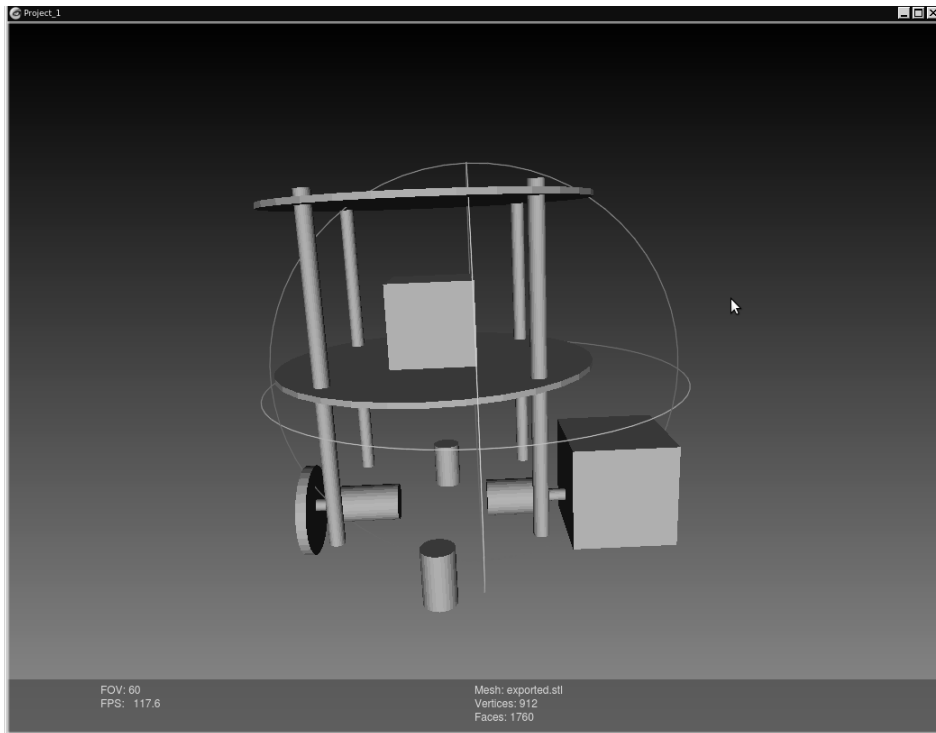


Plik *exported.stl* można otworzyć w programie MeshLab. Oto zrzut ekranu z programu MeshLab (patrz rysunek na następnej stronie).

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

adres IP, 246
AI, 23, 207, 208
AIML, 208, 209
 interpreter, 209
 ładowanie pliku, 215, 216, 217, 218
 znacznik, 209
 aiml, 209
 category, 210
 pattern, 210
 srai, 212
 star index, 211
 template, 210
akcelerometr, 154
aktuator, 29, 119
 Dynamixel, *Patrz:* Dynamixel
algorytm
 fuzyjny, 154
 nawigacji, 114
 SLAM, *Patrz:* SLAM
 wyszukiwania A*, 28
Alicebot, 209
architektura, 31
Archytas z Tarentu, 25
Arduino, 109
artificial intelligence, *Patrz:* AI
Artificial Intelligence Markup Language,
 Patrz: AIML
Asimov Isaac, 26
Asus Xtion PRO, 167, 168, 182

B

biblioteka
 actionlib, 265
 GTK+, 253
 obsługi wizji, 165
 OpenCV, *Patrz:* OpenCV
 OpenNI, *Patrz:* OpenNI
 PCL, *Patrz:* PCL
 pydynamixel, 138
 Qt, 253, 254, 258
 rozpoznawania mowy, 189, 190
 rqt, 269
 syntezy mowy, 191
Blender, 39, 40, 48
 API Pythona, 50
 instalacja, 40
 narzędzia, 49
 skrypt Pythona, 49, 50, 51
bot, *Patrz:* robot programowy

C

Čapek Karel, 20, 25
Carmine, 167, 169
caster wheel, *Patrz:* koło kastora
chmura punktów, 113, 175, 182
 generowanie, 182
 konwersja na dane skanu laserowego, 183
CMU Sphinx, *Patrz:* Sphinx
counts per revolution, *Patrz:* CPR

CPR, 131
czujnik, *Patrz:* sensor
Halla, 130

D

da Vinci Leonardo, 25
dane odometryczne, 67, 104, 112, 152
analiza błędów, 279
kalibracja, 279
korekcja błędów, 280
de Vaucanson Jacques, 26
dead reckoning, *Patrz:* nawigacja zliczeniowa
degrees of freedom, *Patrz:* DOF
detektor kolizji, 27
DOF, 62
doświadczenie użytkownika, 174
Dynamixel, 137
programowanie, 138
układ szeregowy, 137

E

efekt Halla, 130
efektor, 29, 31
Energia, 123, 281
enkoder, 104, 109, 117, 234, 279
dane, 131
koła, 67
kwadraturowy, 105, 130
liczba impulsów na obrót, *Patrz:*
liczba taktów na obrót, *Patrz:* CPR
odległość na takt, 279
rozdzielczość, 131
schemat kodowania, 134
eSpeak, 191
instalowanie, 201

F

Festival, 191, 204
instalowanie, 201
filtr spamu, 23
format
.brn, 217, 218
AIML, 208, 209
PCM, 189

G

Gazebo, 59, 61, 79
grafika 3D, 80
instalacja, 80
klient, 81
model robota, 80
plik
cheffbot_base.urdf.xacro, 94
cheffbot_base_gazebo.urdf.xacro, 90
CMakeList.txt, 89
kinect.urdf.xacro, 93
package.xml, 89
świata, 89
serwer, 81
symulacja
chmury, 80
dynamiczna, 79
transport TCP/IP, 80
uruchamianie, 81
wiersz polecenia, 80
wsparcie dla sensorów, 80
wtyczka, 80
głośniki, 115, 230
Gołąb, *Patrz:* ptak mechaniczny
grajek mechaniczny, 26

H

Halla efekt, *Patrz:* efekt Halla
H-bridge, *Patrz:* mostek typu H
Herbert Simon, 27
Hidden Markov Model, *Patrz:* model HMM
humanoid, 25

I

ICC, 64
IDE Energia, *Patrz:* Energia
IMU, 112, 117, 152, 154
Inertial Measurement Unit, *Patrz:* IMU
Instantaneous Center of Curvature, *Patrz:* ICC
inteligencja, 26
sztuczna, 27
interakcja naturalna, *Patrz:* NI
interfejs
GStreamer pocketsphinx, 203
GUI, 256, 263
sensorów, 233

IR, *Patrz*: sensor na podczerwień

J

jednostka pomiaru inercyjna, *Patrz*: IMU

Julius, 190, 192, 198
instalowanie, 198

K

kaczka trawiąca, 26

kamera, 166
internetowa, 166
rejestracja obrazu, 172

IR, 113
kalibracja, 277
RGB, 113
kalibracja, 274
telewizyjna, 27
TOF, 182

karta dźwiękowa, 189

Kinect, 113, 117, 143, 167, 168, 176, 182, 183
kalibracja, 273
przetwarzanie chmur punktów, 182
wyświetlanie obrazów, 178

kod Graya, 132, 133

koło kastora, 36, 47, 117, 122, 229

kompas, 280, 281

kompilator pyuic, 260

komputer, 232
adres IP, *Patrz*: adres IP

kontroler, 30, 31

Tiva C, 147

konwerter poziomu, 117, 121

L

leksykon, 189

LibreCAD, 39, 40
instalacja, 40
interfejs, 41
narzędzia, 42

lokomocja, 30

M

makro CMake, 75
manipulacja, 30

mapa środowiska, 113, 185, 242, 248, 282
współrzędne, 265

Mataric Maja, 22

McCarthy John, 27

mechanizm różnicowy, 36, 65, 104, 117, 122, 233

MeshLab, 39, 40

instalacja, 41

Microsoft Speech SDK, 190, 192

instalowanie, 202

miernik zasięgu, 27

mikrofon, 115, 189, 230

wielozakresowy, 114

Minsky Marvin, 27

model

akustyczny, 189, 200

HMM, 189, 190

języka, 189

napędowy różnicowy, *Patrz*: mechanizm
różnicowy

ukryty Markowa, *Patrz*: model HMM

moduł, *Patrz też*: pakiet

bpy, 50

PyQt, 255, 256

PySerial, 147

PySide, 255, 256

moment obrotowy, 36, 105

maksymalny, 38

obliczanie, 37

mostek typu H, 107, 109, 120

motoreduktor, 104

DC, 119, 120

mowa

algorytm wyszukiwania, 189

model akustyczny, 189, 200

rozpoznawanie, 188, 189, 198, 202, 209

biblioteka, 189

w czasie rzeczywistym, 195

synteza, 190, 191, 209

wyodrębnianie cech, 189

N

Natural Interaction, *Patrz*: NI

nawigacja, 285

autonomiczna, 245

inercyjna, 152

zliczeniowa, 154

Newell Allan, 27

Next Unit of Computing, *Patrz*: NUC

NI, 174
 NUC, 114

O

obraz
 3D, 113
 głębia, 166, 176, 277
 OpenCV, 165, 170, 177
 instalowanie, 171
 wrapper, 171
 OpenNI, 165, 174
 instalowanie, 175
 uruchamianie, 176

P

pakiet, *Patrz też:* moduł
 chefbot_bringup, 239
 chefbot_description, 86, 89
 chefbot_gazebo, 87
 pocketsphinx, 203
 sound_play, 204
 turtlebot_description, 86
 turtlebot_gazebo, 86
 paradygmat, *Patrz:* architektura
 PCL, 165, 175
 Pocket Sphinx, 190, 192, 200, 203
 podwozie, 38
 Point Cloud Library, *Patrz:* PCL
 pojazd powietrzny bezzałogowy, *Patrz:* UAV
 polecenie
 catkin_create_pkg, 75
 roslaunch, 86
 PPR, 131
 prawo robotyki, 26
 procesor
 NUC, *Patrz:* NUC
 ruchu, 154
 projektor IR, 113
 przekładnia różnicowa, *Patrz:* mechanizm
 różnicowy
 przestrzeń inercyjna, 152
 ptak mechaniczny, 25
 pulses per revolution, *Patrz:* PPR
 PyAIML, 208, 213
 instalowanie, 213
 integracja z ROS, 219
 Python skrypty w Blenderze, 49, 50, 51

Q

Qt Designer, 256

R

reactive control, *Patrz:* sterowanie reaktywne
 RoboLogix, 62
 robot, 20, 21, 22, 26, 28
 A.L.I.C.E., 216
 akumulator, 263
 ChefBot, 103, 104, 115, 207
 kod, 234
 sterowanie, 232
 symulacja, 96
 dane odometryczne, *Patrz:* dane
 odometryczne
 dynamika, 62
 fizyczny, 24, 29
 historia, 25
 kalibracja, 273, 279
 kinematyka, 62, 86
 odwrotna, 68
 w przód, 63
 komputer, *Patrz:* komputer
 model 3D, 48
 modelowanie, 60
 nieholonomiczny, 68
 oś obrotu, *Patrz:* ICC
 płyta
 bazowa, 43, 44
 dolna, 228
 górną, 48, 230
 łączenie, 229
 środkowa, 47, 229
 podwozie, 38
 prędkość kątowa, 64, 280
 programowy, 23
 prototyp wirtualny, 60
 Shakey, 27
 stan, *Patrz:* stan
 stopnie swobody, *Patrz:* DOF
 symulacja, 59, 79, 96
 system sterowania, *Patrz:* sterowanie
 środowisko, 60
 mapa, *Patrz:* mapa środowiska
 testowanie, 273, 282

TurtleBot, 38, 60
 instalacja, 82
 pakiet, *Patrz:* pakiet
 symulacja, 96
 wartość odometryczna, *Patrz:* dane odometryczne
 wydawanie poleceń, 264
 zasilacz, 115
 robotyka mobilna, 30
 ROS, 69, 176, 177, 236
 actionlib, 265
 dystrybucja, 72
 funkcje, 69
 graf obliczeniowy, 70
 Indigo, 83
 instalacja, 72
 komunikat, 70, 71, 76
 lista mailingowa, 72
 Master, 71
 pakiet, 70, 75
 plik manifestu, 70
 plik startowy, 241, 242
 repozytorium, 72
 serwer parametrów, 71
 stos nawigacyjny, 238
 system plików, 70
 torebka, 71
 usługa, 70, 71
 węzeł, 70, 71, 76
 Wiki, 72
 RViz, 247, 250, 265

S

sensor, 24, 31
 HC-SR04, 144, 145
 IMU, 104
 Kinect, *Patrz:* Kinect
 liczby obrotów, *Patrz:* enkoder
 MPU 6000, 154
 na podczerwień, 109, 149
 sposób pomiaru, 150
 odległości, 143, 234
 optyczny, 130
 ping, *Patrz:* sensor ultradźwiękowy
 ultradźwiękowy, 104, 109, 110, 117, 143, 144, 145, 234
 HC-SR04, 111

wizji 2D, *Patrz:* kamera
 wizji 3D, 113, 165, 166, 167, 168, 169
 Shannon Robert, 60
 siatka 3D, 39
 silnik, 36, 233
 kontroler, *Patrz:* sterownik
 liczba obrotów na minutę, 36
 moment obrotowy, *Patrz:* moment obrotowy
 Pololu, 105
 sterownik, *Patrz:* sterownik
 text to speech, *Patrz:* TTS
 siłownik, *Patrz:* aktuator, *Patrz:* aktuator
 Simultaneous Localization and Mapping, *Patrz:* SLAM
 skan laserowy, 247
 imitowanie, 183
 SLAM, 165, 185, 248
 słownik, *Patrz:* leksykon
 słowo kluczowe volatile, 134
 Sphinx, 189
 instalowanie, 192
 stan, 29, 263
 obserwowalny
 częściowo, 29
 w pełni, 29
 sterowanie, 236, 238, 263
 deliberatywne, *Patrz:* sterowanie
 hierarchiczne
 hierarchiczne, 31
 hybrydowe, 32
 reaktywne, 31
 sterownik, 104, 107, 121
 płyta, 109
 system
 catkin, 75
 nawigacji inercyjnej, 152
 nieholonomiczny, 62
 ROS, *Patrz:* ROS
 rozpoznawania mowy, *Patrz:* mowa
 rozpoznawanie
 sterowania, *Patrz:* sterowanie
 sztuczna inteligencja, *Patrz:* AI

T

Tesla Nikola, 26
 text to speech, *Patrz:* TTS
 Tiva C LaunchPad, 109
 TTS, 115

U

UAV, 152
układ
 DN2820FYKH, 114
 GP2D12, 149
 HC-SR04, 144, 145
 inercyjnej jednostki pomiarowej, *Patrz:* IMU
 MPU 6000, 154
 MPU 6050, 112, 154, 155, 159, 234
 kalibracja, 281
 napędowy, 119
Unmanned Aerial Vehicles, *Patrz:* UAV
URDF, 88, 91

V

V-REP, 61

W

wartość odometryczna, *Patrz:* dane odometryczne
Webots, 61
wheel encoder, *Patrz:* enkoder koła

X

Xacro, 88

Z

zasilacz, 115

Ż

żyroskop, 154, 280, 281

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

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.

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. Biegłe 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.

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

[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>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-2345-2



9 788328 323452

Informatyka w najlepszym wydaniu

cena: 49,00 zł