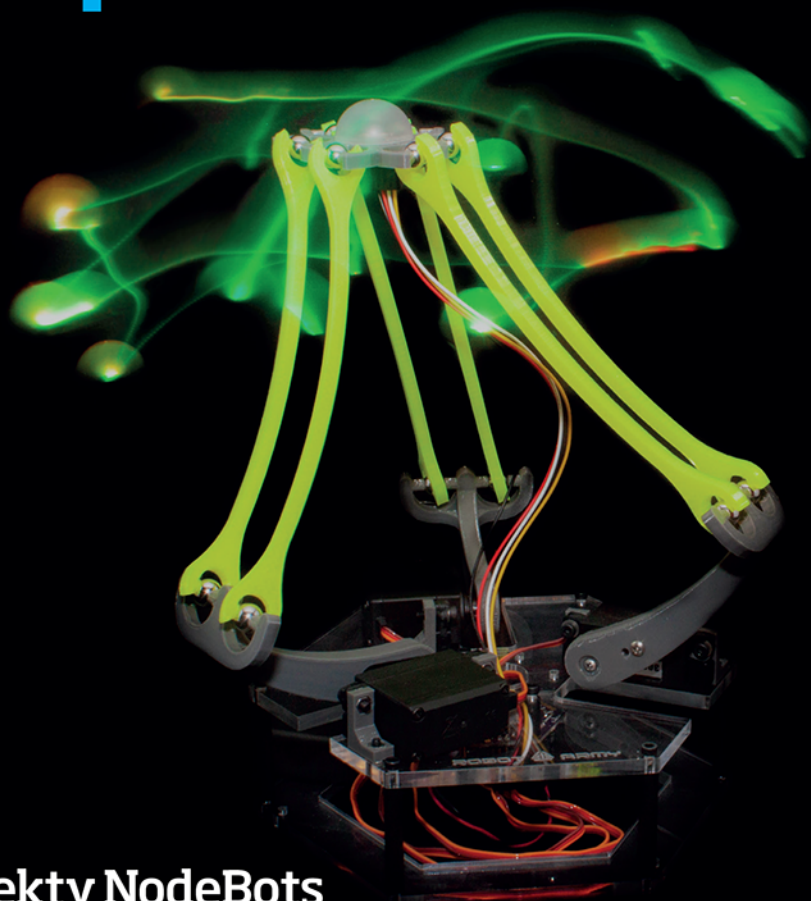


# Roboty JavaScript od podstaw



**Projekty NodeBots  
dla platformy Johnny-Five  
z wykorzystaniem płytek Raspberry Pi,  
Arduino oraz BeagleBone**

**Rick Waldron, Backstop Media**

**Helion** 

Tytuł oryginału: Make: JavaScript Robotics: Building NodeBots with Johnny-Five, Raspberry Pi, Arduino, and BeagleBone

Tłumaczenie: Andrzej Watrak

ISBN: 978-83-283-2054-3

© 2016 Helion S.A.

Authorized Polish translation of the English edition of Make: JavaScript Robotics, ISBN 9781457186950 © 2015 Backstop Media, LLC, published by Maker Media Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

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](mailto:helion@helion.pl)  
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:  
<ftp://ftp.helion.pl/przyklady/roboty.zip>

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

Printed in Poland.

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

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

---

# Spis treści

---

<b>Przedmowa</b> .....	<b>11</b>
<b>1. Budowanie robotów ze zwykłych materiałów</b> .....	<b>15</b>
Budowa robota SimpleBot .....	16
Wykaz materiałów .....	16
Etapy budowy .....	17
Instalacja pakietów Node.js .....	20
Sprawdzenie urządzenia za pomocą podstawowego programu .....	20
Diagnostyka problemów .....	21
Prosty program sterujący .....	21
Diagnostyka problemów .....	23
Odcięcie przewodu .....	24
Budowa bezprzewodowego robota SimpleBot .....	24
Połączenia .....	24
Sterowanie robotem SimpleBot .....	25
Diagnostyka problemów .....	28
Co dalej? .....	29
<b>2. Robot piszący TypeBot</b> .....	<b>31</b>
Wykaz materiałów .....	31
Informacje o serwowoatorach .....	32
Anatomia ramienia robota .....	33
Konstrukcja ramienia .....	33
Warunki związane z ruchem ramienia .....	34

Budowa części mechanicznej .....	34
Podstawa i ramię .....	34
Łokieć .....	35
Nadgarstek .....	36
Palec .....	37
Mózg .....	37
Utworzenie programu .....	38
Utworzenie plików projektu .....	38
Sterowanie serwomotorami .....	38
Inicjalizacja .....	41
Określenie sekwencji naciśnięć klawiszy .....	43
Pierwsze uruchomienie .....	46
Dokładna regulacja ramienia .....	46
Co dalej? .....	47
<b>3. Pierwsze kroki z robotami NodeBoat .....</b>	<b>49</b>
Wykaz materiałów .....	50
Narzędzia .....	50
Zasobnik z silnikiem .....	51
Po co stosować sterownik silnika? .....	51
Komponenty zasobnika silnika .....	51
Modyfikacja silnika .....	52
Montaż silnika .....	53
Sprawdzenie silnika .....	53
Przygotowanie silnika .....	53
Otwór w zasobniku na silnik .....	54
Zamknięcie zasobnika na silnik .....	55
Uszczelnienie przewodów .....	55
Konfiguracja modułu Spark .....	56
Test modułu Spark .....	56
Twój pierwszy program Spark .....	57
Przylutowanie sterownika silnika .....	58
Połączenie komponentów łodzi .....	58
Zasilanie sterownika silnika .....	59
Połączenie modułu Spark i sterownika silnika .....	60
Podłączenie silnika .....	61
Sterowanie silnikiem: kod .....	62
Obsługa zdarzenia naciśnięcia klawisza .....	62
Zapisywanie stanu klawiszy .....	63
Wodowanie łodzi .....	64

Sterowanie łodzią za pomocą serwomotoru .....	66
Programowanie serwomotoru .....	66
Montaż steru .....	68
Pierwszy rejs .....	69
Co dalej? .....	69
<b>4. Mobilna platforma piDuino5 .....</b>	<b>71</b>
Wykaz materiałów .....	72
Instalacja platformy Node.js na płytce Raspberry Pi .....	73
Pobranie kodu piDuino5 i zależności .....	73
Narzędzia .....	73
Konfiguracja płytek i instalacja oprogramowania .....	73
Podłączenie Arduino .....	74
Przegląd kodu app.js .....	74
Sprawdzenie platformy Johnny-Five za pomocą protokołu WebSocket .....	74
Inicjalizacja platformy Johnny-Five .....	75
Sterowanie urządzeniami .....	75
Sterowanie z małymi opóźnieniami za pomocą protokołu WebSocket .....	75
Połączenie z każdego miejsca .....	76
Montaż komponentów .....	76
Wysłanie adresu do aplikacji WWW .....	78
Sterowanie za pomocą smartfona .....	78
Pobranie aplikacji WWW do sterowania robotem piDuino5 .....	78
Sprawdź aplikację WWW na telefonie .....	79
Przegląd plików app.js i index.html .....	79
Zapisanie lokalnego adresu IP .....	79
Udostępnienie interfejsu użytkownika .....	79
Dotykowy joystick .....	80
Nawiązanie połączenia .....	80
Wysyłanie poleceń .....	80
Co dalej? .....	81
<b>5. Sterowanie heksapodem za pomocą platformy Johnny-Five .....</b>	<b>83</b>
Wykaz materiałów .....	83
Sterowanie robotem za pomocą wiersza poleceń .....	84
Wprowadzenie do programu phoenix.js .....	85
Montaż robota .....	86
Przygotowanie szkieletu .....	87
Montaż komponentów elektronicznych .....	87
Przygotowanie serwomotorów .....	87
Montaż stawów biodrowych .....	88

Montaż segmentów udowych .....	89
Montaż segmentów puszczelowych .....	89
Układ współrzędnych .....	90
Regulacja serwowatorów .....	91
Regulacja stawów biodrowych .....	91
Regulacja segmentów udowych .....	92
Regulacja segmentów puszczelowych .....	92
Określenie zakresów ruchu serwowatorów .....	92
Chodzenie jest trudne! .....	93
Poznaj klasę Animation .....	94
Tabela serwowatorów jako grupa docelowa .....	94
Obiekt Servo.Array jako grupa docelowa .....	94
Tabela obiektów Servo.Array jako grupa docelowa .....	94
Pierwszy segment animacyjny .....	95
Chodzenie .....	98
Wiosłowanie (row) .....	98
Marsz (walk) .....	99
Bieg (run) .....	100
Lista poleceń .....	100
Obracanie robota .....	100
Co dalej? .....	101
<b>6. Budowanie robota NodeBot sterowanego głosem .....</b>	<b>103</b>
Wykaz materiałów .....	103
Płytką BeagleBone Black .....	105
Tworzenie projektu .....	106
Budowa obwodu przekaźnika .....	106
Sterowanie obwodem za pomocą platformy Johnny-Five .....	107
Budowa obwodu mikrofonu .....	108
Obsługa mikrofonu i przekaźnika za pomocą kodu Johnny-Five .....	108
Tworzenie serwera poleceń .....	110
Prosty kontroler głosowy wykorzystujący interfejs Web Speech API .....	112
Integracja serwera poleceń z obwodem przekaźnika .....	112
Zaawansowany kontroler głosowy wykorzystujący gadżet z Androidem .....	115
Aplikacja przenośna dla systemu Android .....	115
Aplikacja dla gadżetu z Androidem .....	120
Co dalej? .....	123
<b>7. Pokojowy zegar słoneczny .....</b>	<b>125</b>
Wykaz materiałów .....	126
Elementy z pianki PCV .....	126

Budowanie zegara słonecznego .....	128
Wycinanie elementów i przygotowanie głównej konstrukcji .....	129
Przygotowanie płyty montażowej .....	129
Podłączenie i konfiguracja serwomotorów .....	130
Podłączenie serwomotorów .....	130
Konfiguracja serwomotorów .....	131
Przygotowanie tarcz .....	132
Przygotowanie ścianek i podparcia tarczy podstawowej .....	133
Przygotowanie ścianek .....	133
Umieszczenie tarcz .....	134
Przygotowanie podparcia tarczy .....	134
Wykonanie ostatnich elementów i montaż zegara .....	136
Przygotowanie ramienia azymutu .....	136
Przygotowanie łuku wysokości .....	137
Wycięcie gnomonu .....	138
Podłączenie „słońca” .....	138
Kod, który wszystkim steruje .....	139
Program sundial.js .....	140
Dane konfiguracyjne w programie sundial.js .....	140
Szczegóły programu sundial.js .....	141
Złóż wszystko razem! .....	143
Włącz zegar! .....	144
Co dalej? .....	144
<b>8. Światelka-strazydelka .....</b>	<b>145</b>
Wykaz materiałów .....	145
Dobór kontrolera matrycy .....	146
Teoria .....	146
Co to jest matryca LED? .....	147
Budowa łańcucha światełek .....	147
Potrzebne narzędzia .....	148
Przygotowanie matrycy .....	149
Przygotowanie kabla .....	150
Diagnostyka .....	153
Sterowanie matrycą LED .....	153
Przygotowanie płytki Arduino .....	153
Dekorowanie łańcucha światełek .....	153
Opcje konstruktora klasy Matrix .....	154
Rysowanie wzorów na matrycy .....	154
Uruchomienie programu testującego .....	154

Utworzenie aplikacji WWW .....	155
Narzędzia programistyczne .....	155
Korzystanie z generatora express .....	155
Utworzenie interfejsu API .....	156
Utworzenie interfejsu użytkownika .....	157
Rozbudowa aplikacji .....	158
Co dalej? .....	158
<b>9. Lampa CheerfulJ5 .....</b>	<b>159</b>
Wykaz materiałów .....	159
Zbudowanie obwodu .....	160
Kod do lampy CheerfulJ5 .....	161
Połączenie z płytką Arduino .....	161
Sterowanie diodą RGB .....	162
Użycie platformy Node.js i pętli REPL .....	163
Zdefiniowanie mapy kolorów usługi CheerLights .....	163
Dostęp do usługi CheerLights za pomocą interfejsu ThingSpeak API .....	164
Korzystanie z interfejsu Twitter Streaming API .....	165
Łączność bezprzewodowa dzięki zestawowi Spark WiFi Development Kit .....	169
Umieszczenie modułu Spark w obudowie .....	169
Zastosowanie wtyczki Spark-io .....	170
Przełączenie na zasilanie z akumulatora .....	170
Umieszczenie obwodu w obudowie .....	171
Co dalej? .....	172
<b>10. Interaktywny panel LED RGB z płytką BeagleBone Black .....</b>	<b>173</b>
Wykaz materiałów .....	174
Płytką BeagleBone Black .....	174
Adapter WiFi (opcjonalny) .....	174
Zewnętrzny zasilacz 5 V (opcjonalny) .....	174
Panel LED RGB .....	175
Czujniki .....	175
Inne komponenty .....	175
Pierwsze kroki: oprogramowanie .....	175
Oprogramowanie LEDScope .....	175
Podłączenie panelu RGB LED .....	176
Podłączenie panelu RGB LED .....	176
Czas na kod! Użyj języka JavaScript .....	178
Uruchomienie skryptu testowego .....	179
Zastosowanie platformy Johnny-Five i obudowy beaglebone-io .....	181



Podłączenie fotorezystora .....	183
Zmianianie kolorów za pomocą przyspieszeniomierza .....	185
Co dalej? .....	187
<b>11. Osobiste bezpieczeństwo, JavaScript i Ty .....</b>	<b>189</b>
Prosty czujnik ultradźwiękowy .....	190
Implementacja .....	190
Urządzenie ultradźwiękowe wysyłające SMS-y .....	193
Implementacja .....	193
System monitoringu .....	194
Implementacja .....	194
Laser robi wrażenie na wrogach i przyjaciółach .....	195
Implementacja .....	195
Wskaźnik stanu, przyciski i diody LED .....	196
Implementacja .....	198
Co dalej? .....	201
<b>12. Sztuczna inteligencja: robot BatBot .....</b>	<b>203</b>
Sztuczna inteligencja — podstawy .....	203
Roboty semiautonomiczne .....	204
Roboty autonomiczne .....	204
Robot BatBot .....	204
Roboty zdalnie sterowane .....	204
Wykaz materiałów .....	205
Uwagi dotyczące materiałów .....	206
Montaż .....	206
Krok 1. Zdalne sterowanie .....	207
Poruszanie robotem .....	208
Sterowanie robotem .....	210
Ustawianie czujnika ultradźwiękowego i odczyt sygnału .....	210
Krok 2. Autonomia .....	211
Implementacja algorytmu .....	213
Diagnostyka .....	215
Co dalej? .....	216
<b>13. Roboty delta i kinematyka .....</b>	<b>217</b>
Wykaz materiałów .....	218
Anatomia robota delta .....	219
Budowa robota Junky Delta .....	220
Wprawienie robota w ruch .....	223
Określanie położenia w kinematyce .....	225
Bardziej zaawansowane roboty delta .....	229

Robot TapsterBot .....	229
Robot Army .....	229
Co dalej? .....	230
<b>14. Interaktywne buty .....</b>	<b>231</b>
Wykaz materiałów .....	231
Opis elementów .....	231
Przygotowanie czujników .....	234
Przylutowanie przewodów do czujników .....	234
Umieszczenie czujników w butach .....	235
Wyprowadzenie przewodów na zewnątrz buta .....	235
Połączenie butów .....	235
Przygotowanie przewodu spiralnego .....	235
Podłączenie butów do płytki Arduino .....	237
Przygotowanie przewodów .....	237
Przylutowanie płytki Arduino .....	238
Przymocowanie płytki Arduino do prawego buta .....	238
Tworzenie kodu opartego na platformie Johnny-Five .....	238
Podłączenie do platformy Johnny-Five .....	238
Konfiguracja czujników .....	239
Wyświetlanie danych z czujników .....	239
Przykładowe zachowania .....	240
Co dalej? .....	241
<b>A. Dodatek .....</b>	<b>243</b>
Instalacja biblioteki Node.js .....	243
Instalacja platformy Johnny-Five .....	243
Diagnostyka .....	244
Dodatkowe informacje .....	244
Konfigurowanie sprzętu .....	244
Arduino .....	244
BeagleBone Black .....	244
Raspberry Pi .....	244
Zestaw Particle WiFi Development Kit .....	245
Programowanie w systemie Android .....	247
Instalacja oprogramowania Android Studio .....	247
Tworzenie projektu Android .....	247
Instalacja biblioteki Volley .....	248
<b>Skorowidz .....</b>	<b>249</b>

# Sterowanie heksapodem za pomocą platformy Johnny-Five

# 5

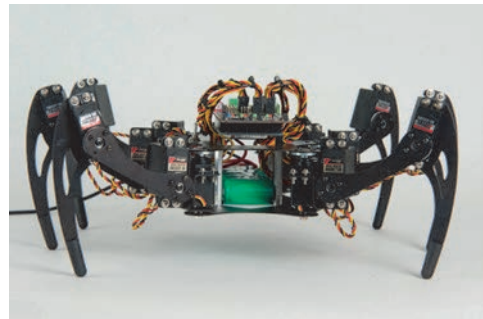
*Donovan Buck*

W tym rozdziale dowiesz się, jak zbudować prostego kroczącego heksapoda, czyli sześcionogiego robota. Opisany tu heksapod, pokazany na [rysunku 5.1](#), będzie miał po trzy stawy w każdej nodze. Robot będzie sterowany za pomocą klasy `Animation` z platformy Johnny-Five. Klasa ta jest wykorzystywana do tworzenia skryptów sterujących serwomotorami w czasie. Oferuje funkcjonalności takie jak harmonogram, kluczowe ramki, płynne przejście jednej ramki w drugą (ang. *tweening*) i funkcje wygładzające (ang. *easing functions*). Na koniec otrzymasz doskonałą platformę do budowania w przyszłości złożonych, ciekawych robotów.

## Wykaz materiałów

Materiały użyte w tym rozdziale wymienione są w [tabeli 5.1](#).

Sześć serwomotorów Hitec HS-485HB będzie tworzyć stawy biodrowe, sześć modeli Hitec HS-645MG —



**Rysunek 5.1.** Gotowy heksapod

stawy segmentów udowych, a sześć Hitec HS-5685 MH — stawy segmentów piszczelowych. Firma Lynxmotion zaleca zastosowanie sześciu serwomotorów HS-485HB i dwunastu HS-645MG, jednak modele HS-5685MH oferują większy moment obrotowy niezbędny w stawach segmentów udowych. Do poruszania robotem będzie potrzebny maksymalny moment obrotowy, jaki silniki mogą osiągnąć.

Tabela 5.1. Wykaz materiałów do budowy robota

Liczba	Element	Przybliżona cena	Numer/źródło
1	Heksapod Phoenix 3DOF (bez serwomotorów i elementów elektronicznych)	950 zł	LM PHOE
1	Arduino Mega 2560	175 zł	MS MKSP5, AF 191, SF DEV-11061
1	Nakładka DFRobot Mega Sensor	75 zł	AZ B0098SJ1RS
6	Serwomotor HiTec HS-485HB	65 zł	LM S485HB, AZ B00944TF72
6	Serwomotor HiTec HS-645MG	120 zł	LM S645MG, AZ B003T6RSVQ
6	Serwomotor HiTec HS-5685MH	150 zł	LM S5685MH, AZ B003X6KT7C
1	Ładowarka 6 V – 12 V NiMH/NiCd	85 zł	LM USC-02, AZ B001DHC2LO
1	Akumulator 6 V / 2800 mAh Ni-MH	105 zł	LM BAT-05

Lista potrzebnych narzędzi jest skromna. Do podstawowego montażu będą potrzebne:

- zestaw małych wkręteków,
- szczypce precyzyjne,
- wiertarka,
- wiertło o średnicy 3 mm,
- wiertło o średnicy 6 mm,
- opaski zaciskowe (kilkadziesiąt sztuk).

## Sterowanie robotem za pomocą wiersza poleceń

Podczas konstruowania robota będziesz musiał sterować serwomotorami i zmieniać ich pozycje, wykorzystując układ współrzędnych. Pętla REPL oferowana przez platformę Node.js doskonale nadaje się do tego celu. Możliwość wysyłania poleceń do serwomotorów i obserwowania ich reakcji w czasie rzeczywistym jest bardzo fajna i przydatna.

Do sterowania robotem za pomocą pętli REPL będzie potrzebny program *phoenix.js*. Wykonaj następujące czynności:

1. Pobierz zawartość katalogu *Buck.Animation* z serwisu GitHub (<https://github.com/rwaldron/javascript-robotics>).
2. Zapisz zawartość katalogu w folderze roboczym na komputerze.
3. Przejdź do katalogu roboczego, a następnie użyj polecenia `npm` do zainstalowania zależności:

```
npm install
```

Powyższe polecenie powinno dostarczyć wszystkich danych potrzebnych do skonfigurowania robota i sterowania nim. Jeżeli nie zainstalowałeś jeszcze platformy Node.js na swoim komputerze, zajrzyj do części „Instalacja platformy Node.js” w dodatku.

### Pętla REPL

Skrót REPL oznacza pętlę odczytaj-przetwórz-wyświetl (ang. *read-eval-print loop*). Umożliwia ona wprowadzanie informacji za pomocą wiersza poleceń, następnie przetwarzanie wprowadzonych danych za pomocą kodu i na koniec wyświetlenie wyników i oczekiwanie na kolejne polecenia.

## Wprowadzenie do programu phoenix.js

Plik *phoenix.js* ze skryptem JavaScript zawiera pełny kod potrzebny do konfiguracji robota i sterowania nim. W dalszej części rozdziału określisz w tym kodzie pewne wartości konfiguracyjne, ale na razie tylko przejrzyj go i zapoznaj się z jego kilkoma podstawowymi funkcjonalnościami.

Plik składa się z dwóch głównych sekcji, w których może być konieczne wprowadzenie zmian. Pierwsza z nich — sekcja konfiguracyjna — znajduje się w pobliżu początku pliku. Są w niej zdefiniowane obiekty opisujące wszystkie pozycje, jakie przyjmują serwomotory podczas wykonywania przez heksapoda kroków przesuających go naprzód i obracających na boki. Te dane będziesz musiał zmienić tylko w przypadku, gdy zastosujesz inne serwomotory niż wymienione w liście.

Druga sekcja, w której koniecznie musisz wprowadzić zmiany, zawiera instrukcje tworzące obiekty reprezentujące serwomotory poruszające nogami robota. Sekcja ta znajduje się wewnątrz funkcji zwrotnej w metodzie `five.Board().on("ready")`. Za pomocą właściwości `offset` musisz ustawić w środkowej pozycji wszystkie 18 serwomotorów. Szczegółowe instrukcje, jak to zrobić, zawarte są w dalszej części rozdziału. Ponadto, jeżeli używasz innego typu serwomotorów niż wymienione w tabeli, będziesz musiał zmienić wartości właściwości `range` serwomotorów.

Definicja serwomotoru w obiekcie `phoenix` jest następująca:

```
phoenix.r1c = new five.Servo({
  pin:40, ①
  offset: 0, ②
  startAt: 1.c, ③
  range: [50, 180], ④
  isInverted: true ⑤
});
// + 17 kolejnych takich definicji...
```

- ① właściwość `pin` określa numer pinu nakładki Mega Sensor, do którego podłączony jest serwomotor.
- ② Właściwość `offset` służy do regulacji pozycji serwomotoru.

- ③ Właściwość `startAt` określa pozycję początkową serwomotoru.
- ④ Właściwość `range` określa minimalną i maksymalną wartość kąta, o jaki może obrócić się serwomotor.
- ⑤ Właściwość `isInverted` odwraca wartość kąta obrotu wysyłaną do serwomotoru (np. wartość 180 jest zamieniana na 0, 45 na 135 itd.).

Tabela 5.2 przedstawia funkcje serwomotorów oraz piny wykorzystane przez wszystkie serwomotory poruszające stawami robota.

Gdy przyjrzyś się dokładnie kodowi, zauważysz, że w konfiguracjach serwomotorów tylnych stawów biodrowych właściwość `isInverted` przyjmuje przeciwne wartości niż w przypadku serwomotorów w pozostałych stawach biodrowych z tej samej strony robota. Jest tak dlatego, ponieważ podczas przesuwania przedniej nogi w przód dwie pozostałe nogi muszą być przesuwane wstecz. W dalszej części kodu dla każdej nogi zdefiniowana jest tabela `Servo.Array`:

```
phoenix.l1 = new five.Servo.Array([
  phoenix.l1c,
  phoenix.l1f,
  phoenix.l1t
]);
// Pięć kolejnych podobnych definicji
```

Po wywołaniu metody klasy `Servo.Array`, na przykład z obiektu `phoenix.l1`, w każdym obiekcie typu `Servo` zawartym w powyższej klasie wywoływana jest ta sama metoda z takimi samymi argumentami. Klasa `Servo.Array` jest również wykorzystywana do łączenia serwomotorów w grupy animacyjne. Więcej informacji na ten temat znajduje się w dalszej części rozdziału.

W pliku *phoenix.js* tworzone są również następujące obiekty typu `Servo.Array`:

- `coxa`: sześć serwomotorów tworzących stawy biodrowe,
- `femur`: sześć serwomotorów w segmentach udowych,
- `tibia`: sześć serwomotorów w segmentach piszczelowych,
- `legs`: wszystkie serwomotory (osiemnaście sztuk)

Tabela 5.2. Lista serwomotorów

Serwomotor	Skrót	Model	Pin
Staw biodrowy lewy 1	L1C	HS-485HB	27
Segment udowy lewy 1	L1F	HS-645MG	26
Segment piszczelowy lewy 1	L1T	HS-5685MH	25
Staw biodrowy lewy 2	L2C	HS-485HB	23
Segment udowy lewy 2	L2F	HS-645MG	21
Segment piszczelowy lewy 2	L2T	HS-5685MH	20
Staw biodrowy lewy 3	L3C	HS-485HB	19
Segment udowy lewy 3	L3F	HS-645MG	18
Segment piszczelowy lewy 3	L3T	HS-5685MH	17
Staw biodrowy prawy 1	R1C	HS-485HB	40
Segment udowy prawy 1	R1F	HS-645MG	39
Segment piszczelowy prawy 1	R1T	HS-5685MH	38
Staw biodrowy prawy 2	R2C	HS-485HB	49
Segment udowy prawy 2	R2F	HS-645MG	48
Segment piszczelowy prawy 2	R2T	HS-5685MH	47
Staw biodrowy prawy 3	R3C	HS-485HB	45
Segment udowy prawy 3	R3F	HS-645MG	44
Segment piszczelowy prawy 3	R3T	HS-5685MH	43

Ponadto tworzona jest tabela zawierająca tabele serwomotorów, która będzie użyta później podczas animowania uśpienia i przebudzenia robota:

```

    phoenix.joints = new five.Servo.Array([
        phoenix.coxa,
        phoenix.femur,
        phoenix.tibia
    ]);

```

Jak widać, w pliku *phoenix.js* tworzonych jest wiele obiektów typu *Servo.Array*, zawierających różne kombinacje serwomotorów. Dzięki takiej konfiguracji poruszanie segmentami nóg robota będzie łatwiejsze. Zadaniem kodu w tym pliku jest wysyłanie poleceń tylko do wybranych grup serwomotorów.

## Montaż robota

Zanim poznasz klasę *Animation*, musisz zmontować robota. Większość czynności wykonaj według instrukcji dostarczonej przez Lynxmotion. Są jednak trzy miejsca, na które musisz zwrócić szczególną uwagę. Oto najważniejsze etapy montażu:

1. Przygotowanie szkieletu.
2. Montaż komponentów elektronicznych.
3. Przygotowanie serwomotorów.
4. Montaż stawów biodrowych.
5. Montaż segmentów udowych.
6. Montaż segmentów piszczelowych.

## Przygotowanie szkieletu

Jednym z problemów, które musisz rozwiązać, jest montaż komponentów elektronicznych. Firma Lynxmotion zaprojektowała heksapoda Phoenix pod kątem płytki BotBoarduino. Płytkę tę ma otwory montażowe w innych miejscach niż Arduino Mega. Ponadto płytkę Arduino Mega jest duża. Jej montaż wewnątrz szkieletu robota Phoenix jest idealnym, ale trudnym rozwiązaniem. Musisz odsunąć platformy szkieletu na tyle, aby dopasować je do szerokości płytki Mega. Na razie zamontuj płytkę Arduino po prostu na górnej platformie szkieletu.

Na [rysunku 5.2](#) widoczne są dwa małe słupki dystansowe z tworzywa sztucznego, dopasowane do otworów montażowych w płytce Arduino Mega. Zwróć uwagę, że nie są one dopasowane idealnie. Płytkę po zamontowaniu będzie obrócona w stosunku do szkieletu pod niewielkim kątem. Gdyby Ci to nie odpowiadało, wywierć dodatkowe otwory montażowe.



Rysunek 5.2. Zmontowany szkielet robota

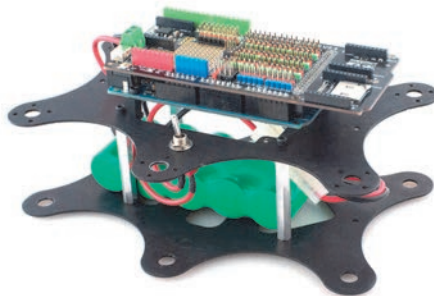
Od tego momentu do oznaczenia nóg heksapoda będą stosowane skrótory R1, R2 i R3 (nogi prawe) oraz L1, L2 i L3 (nogi lewe).

## Montaż komponentów elektronicznych

Aby zamontować elementy elektroniczne, wykonaj następujące czynności:

1. Przycmóć płytkę Arduino do słupków dystansowych 1/2".

2. Załóż nakładkę silnikową na płytkę Arduino.
3. W zestawie Phoenix znajduje się wyłącznik zasilania serwowymotorów. Aby go zamontować, trzeba wykonać otwór o średnicy 6 mm, ponieważ istniejące otwory w szkielecie zostaną zasłonięte przez komponenty elektroniczne. Wywierć więc nowy otwór pod wyłącznik, który zalecam zamontować w pobliżu nogi R2. Zamontuj go w szkielecie w sposób pokazany na rysunku, przytnij przewody na odpowiednią długość i podłącz je do złączy zasilania nakładki silnikowej.
4. Akumulator Ni-MH można ściśle wpasować pomiędzy słupki dystansowe i nie trzeba go mocować żadnymi dodatkowymi elementami. Wsuń akumulator obrócony o 45° i umieść go we właściwym położeniu. [Rysunek 5.3](#) przedstawia urządzenie na obecnym etapie montażu.

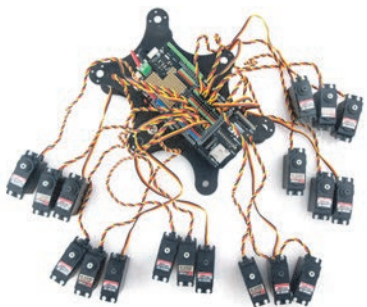


Rysunek 5.3. Szkielet robota z zamontowanymi komponentami elektronicznymi

## Przygotowanie serwowymotorów

Teraz przygotuj serwowymotory. Wykonaj następujące czynności:

1. Podłącz wszystkie serwowymotory stawów biodrowych i segmentów udowych do nakładki Mega Sensor zgodnie z tabelą 5.2. Podłączenie serwowymotorów segmentów piszczelowych nie jest na razie konieczne, ale jeżeli chcesz, możesz to zrobić.
2. Podłącz płytkę Arduino do portu USB komputera, a przewody zasilające do nakładki. [Rysunek 5.4](#) przedstawia widok wszystkich komponentów na obecnym etapie montażu.



Rysunek 5.4. Podłączone wszystkie serwowmotory

- Nasadki wszystkich serwowmotorów muszą być ustawione jak najdokładniej w środkowej pozycji. W tym celu przed założeniem nasadki na serwowmotor musisz ustawić go w środkowej pozycji. W katalogu, w którym znajduje się plik *phoenix.js*, wpisz następujące polecenie:
 

```
node phoenix
```
- Serwowmotory po zainicjowaniu zostaną ustawione w pozycjach odpowiadających wartości właściwości `startAt`. Wszystkie trzeba ustawić w pozycji 90°, więc w pętli REPL wpisz następujące polecenie:
 

```
ph.joints.to(90);
```
- Wszystkie serwowmotory zostaną ustawione w pozycji 90°. Teraz załóż nasadki na serwowmotory stawów biodrowych i serwowmotory segmentów udowych tak, aby otwory montażowe były równoległe do osi. Nie zakładaj nasadek na serwowmotory segmentów piszczelowych. Nasadki powinny być założone tak, aby otwory znalazły się w położeniu pokazanym na rysunku 5.5.



Rysunek 5.5. Serwowmotor ustawiony w pozycji 90° z poprawnie założoną nasadką

- Wykonaj powyższą czynność dla wszystkich 12 serwowmotorów stawów biodrowych i segmentów udowych. Nasadki załóż dopiero po wyłączeniu zasilania nakładki.



Serwowmotory stawów biodrowych trzeba przygotować inaczej niż serwowmotory segmentów udowych i piszczelowych. Uważaj, aby nie pomylić wkrętów i nie uszkodzić serwowmotorów.

## Montaż stawów biodrowych

W celu zamontowania serwowmotorów stawów biodrowych wykonaj następujące czynności:

- Złóż wszystkie elementy stawów biodrowych i segmentów udowych zgodnie z instrukcjami producenta. Zwróć uwagę, że trzy elementy są lustrzanymi odbiciami trzech innych (patrz rysunek 5.6).



Rysunek 5.6. Zmontowane stawy biodrowe

- Teraz przymocuj do szkieletu elementy stawów biodrowych i segmentów udowych dwóch środkowych nóg (R2 i L2). Nie mocuj na razie nasadek do szkieletu.
- Przeciagnij przewody od serwowmotorów pomiędzy nakładką Mega Sensor a szkieletem robota.
- Podłącz cztery serwowmotory do nakładki Mega Sensor. Tabela 5.2 zawiera informacje, gdzie należy podłączyć każdy serwowmotor.



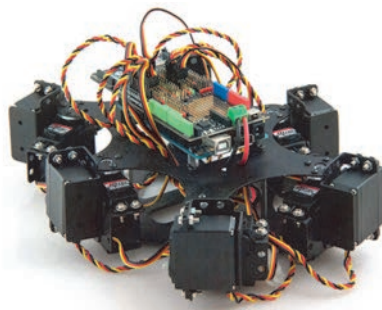
5. Podłącz płytkę Arduino do komputera i akumulatory do nakładki, uruchom program *phoenix.js* i ustaw wszystkie serwomotory w pozycji 90°. Pamiętaj, że na razie zajmujesz się tylko stawami biodrowymi.
6. Obróć serwomotory stawów biodrowych tak, aby nasadki na serwomotorach segmentów udowych były skierowane w przód, a otwory montażowe w nasadkach były dopasowane do otworów w rusztowaniu. Serwomotory stawów biodrowych nie muszą być umieszczone pod idealnym kątem w stosunku do szkieletu. Nie przejmuj się, dopasujesz je później.
7. Połącz nasadki ze szkieletem za pomocą wkrętów, ale jeszcze ich nie dokręcaj.
8. Odłącz przewód USB i przewody zasilające, a następnie dokręć wszystkie cztery wkręty w każdej nasadce serwomotoru.
9. Wykonaj kroki 1. – 8. dla nóg przednich (R1 i L1). Zwróć uwagę, że otwory montażowe w nasadce serwomotoru stawu biodrowego są obrócone pod kątem 45°. Oba segmenty udowe będą ustawione pod kątem 45° w kierunku na wprost heksapoda.
10. Na koniec powtórz kroki 1. – 8. dla nóg tylnych (R3 i L3). Otwory montażowe również będą obrócone o 45°. Oba segmenty udowe będą ustawione pod kątem 45° w kierunku do tyłu heksapoda.
11. Uruchom program *phoenix.js* i przymocuj nasadki do serwomotorów segmentów udowych ustawionych w pozycji 90°.
12. Odłącz przewód USB i przewody zasilające.
13. Dokręć cztery wkręty w każdej nasadce serwomotoru.

Rysunek 5.7 przedstawia efekt dotychczasowej pracy.

## Montaż segmentów udowych

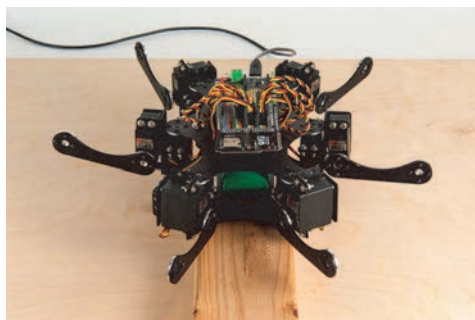
Aby zamontować segmenty udowe, wykonaj następujące czynności:

1. Uruchom program *phoenix.js* i ustaw wszystkie serwomotory w pozycji 90°.



Rysunek 5.7. Szkielet robota ze wszystkimi sześcioma stawami biodrowymi

2. Do segmentów udowych powinny już być przymocowane nasadki. Załóż je na serwomotory segmentów udowych wszystkich sześciu nóg. Segmenty udowe będą ustawione na wprost, niemal równoległe do podłoża.
3. Zatrzymaj program *phoenix.js*, naciskając klawisze *Ctrl+C*, i odłącz płytkę Arduino od komputera. Teraz możesz bezpiecznie przykręcić segmenty udowe do serwomotorów. Rysunek 5.8 przedstawia zamontowane segmenty.



Rysunek 5.8. Wszystkie segmenty udowe już zamontowane

## Montaż segmentów piszczelowych

Montaż jest prawie zakończony, pozostały tylko segmenty piszczelowe. Wykonaj następujące czynności:

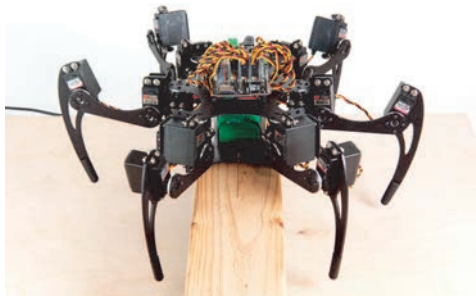
1. Przymocuj sześć segmentów piszczelowych do serwowymotorów. Zwróć uwagę, że trzy elementy są lustrzanymi odbiciami trzech innych.
2. Podłącz przewody od serwowymotorów segmentów piszczelowych do nakładki, ale nie mocuj jeszcze serwowymotorów do segmentów udowych.



Po ponownym uruchomieniu programu *phoenix.js* poruszają się serwowymotory wszystkich stawów. Jeżeli robot jest położony płasko na stole, może zmienić położenie i spowodować poważne zamieszanie. Od tej chwili przed uruchomieniem programu *phoenix.js* musisz trzymać robota nad podłożem. Możesz go trzymać w ręce, albo lepiej — zbudować specjalne stanowisko testowe i z niego korzystać. Bądź ostrożny, jeżeli postanowisz trzymać robota w dłoni — potrafi uszczyplnąć.

3. Uruchom program i ustaw wszystkie serwowymotory w pozycji 90°.
4. Przymocuj serwowymotory segmentów piszczelowych do segmentów udowych. Końcówki nóg powinny znajdować się pod nasadkami serwowymotorów segmentów piszczelowych. Ustaw segmenty pod niewielkim kątem w kierunku do wewnątrz szkieletu.
5. Odłącz przewód USB i zasilanie i przykręć serwowymotory segmentów piszczelowych do segmentów udowych.

Teraz jest odpowiedni moment, aby za pomocą opasek zaciskowych uporządkować przewody i przymocować je w odpowiednich miejscach. Pamiętaj o pozostawieniu odpowiedniego zapasu, aby każda noga mogła poruszać się w pełnym zakresie. Nie pozostawiaj zbyt dużego zapasu, aby przewody nie zaczepiały o nogi. **Rysunek 5.9** przedstawia robota z zamontowanymi segmentami piszczelowymi.



**Rysunek 5.9.** Zamontowane segmenty piszczelowe i serwowymotory wszystkich stawów ustawione pod kątem 90°

## Układ współrzędnych

W tym projekcie dla każdej nogi będzie zastosowany osobny, lokalny układ współrzędnych. Zamiast definiowania położenia końcówek nóg w przestrzeni za pomocą trzech współrzędnych X, Y, Z będą stosowane wartości kątów obrotów serwowymotorów. Jest to tzw. **przegubowy układ współrzędnych**. Układ ten nie pozwala stosować tak zaawansowanych metod animacji jak np. kinematyka odwrotna w globalnym układzie współrzędnych, ale lepiej nadaje się do poznania klasy `Animation`.

Ponieważ w programie *phoenix.js* wykorzystany jest przegubowy układ współrzędnych, każdy serwowymotor ustawiany jest pod określonym kątem. Obracany jest również lokalny układ współrzędnych tak, aby osie były ustawione równolegle. Ponadto odwracane są kierunki obrotów po lewej i prawej stronie robota, jak również po stronie przedniej i tylnej.

Taka konfiguracja powoduje, że poruszanie robotem jest maksymalnie ułatwione. Jeżeli zamierzasz programować własny sposób poruszania robotem, przydatne jest narysowanie układu współrzędnych na stanowisku testowym. Środek układu przyjmij w środku szkieletu robota, a jako jednostkę miary ustal jeden centymetr.

## Regulacja serwowatorów

Do tej pory starałeś się jak najdokładniej dopasować nasadki serwowatorów, jednak wypustki ograniczają położenia przekładni i wszystkie nasadki mogą być obrócone o kilka stopni. Do zniwelowania tego efektu wykorzystaj właściwość `offset` w obiektach konfiguracyjnych serwowatorów.

Kalibracja wszystkich serwowatorów ułatwi proces programowania animacji robota. Jeżeli podczas wykonywania kroku w przód kąt obrotu każdego stawu biodrowego będzie dokładnie taki sam, wtedy za każdym razem będziesz musiał wyliczać tylko jeden parametr ruchu. W dłuższej perspektywie opłaci się poświęcić czas na wykonanie tej wstępnej pracy.

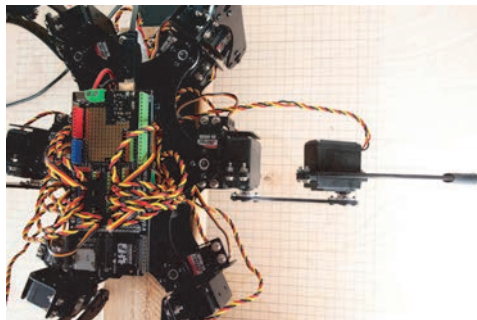
## Regulacja stawów biodrowych

Najpierw wyreguluj serwowator stawu biodrowego L2C:

1. Gdy popatrzysz na robota z góry (rysunek 5.10), segment udowy powinien być ustawiony pod kątem prostym w stosunku do robota (równoległe do osi x układu współrzędnych). Jeżeli używasz stanowiska testowego z narysowanym układem współrzędnych, powinieneś móc to łatwo sprawdzić. Jeżeli serwowator L2C ma obrócić się zgodnie ze wskazówkami zegara, należy ustawić go pod kątem większym niż 90°. Jeżeli ma obrócić się w przeciwnym kierunku, zmniejsz powyższą wartość. Zwiększaj za każdym razem kąt o kilka stopni, aby znaleźć wartość, przy której serwowator będzie ustawiony najdokładniej. Nie używaj ułamkowych wartości, ponieważ platforma Johnny-Five wysyła do serwowatorów tylko wartości całkowite:

```
ph.joints.to(91); // Lepsze ustawienie
ph.joints.to(94); // O, za daleko!
ph.joints.to(93); // Super!
```

2. Po określeniu dokładnej wartości musisz od niej odjąć liczbę 90, aby obliczyć wartość właściwości `offset`. W tym przypadku będzie to działanie  $93 - 90 = 3$ , więc właściwość `offset` otrzyma wartość 3. Wartość ta pełni dokładnie taką samą rolę jak trymer w nadajniku do zdalnego stero-



Rysunek 5.10. Wyregulowany staw biodrowy

wania. Jej przeznaczeniem jest ustawienie w środkowej pozycji serwowatora obróconego o kilka stopni.

3. W kodzie `phoenix.js` odszukaj wiersz, w którym tworzony jest obiekt L2c, i zmień wartość właściwości `offset`:

```
l2c = new five.Servo({
  pin: 27,
  invert: true,
  offset: 3,
  startAt: 90,
  range: [50, 130]
})
```

4. Powtórz kroki 1. i 2. dla serwowatora R2C. Pamiętaj, że ten serwowator jest odwrócony w stosunku do serwowatora po lewej stronie. Będziesz musiał odwrócić wartości kąta obrotu.
5. Teraz wyreguluj serwowator L1C. Segment jest ustawiony pod kątem ok. 45° w stosunku do osi x, ale musi być do niej równoległy, więc ustaw serwowator pod kątem 45°. Segment powinien ustawić się dość dokładnie, jednak będziesz musiał wyregulować jego położenie. Cały proces wygląda następująco:  

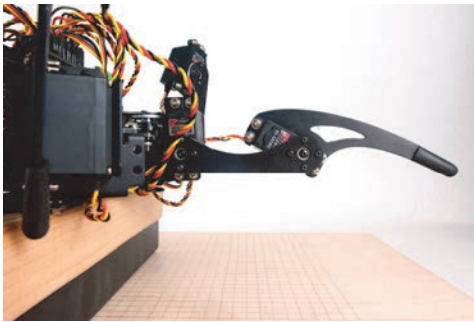
```
ph.joints.to(45); // Znacznie lepiej
ph.joints.to(47); // Źle, nie tak!
ph.joints.to(44); // Super!
```
6. Po wyregulowaniu odejmij od uzyskanej wartości liczbę 90. Będzie to wartość (ujemna) właściwości `offset`.
7. Zmień wartość właściwości `offset` w obiekcie L1c. Często zdarza się, że jest to duża liczba, np. 20.

8. Powtórz kroki 4. – 6. dla pozostałych trzech serwowymotorów stawów biodrowych, a następnie zatrzymaj i uruchom ponownie program *phoenix.js*. Teraz wszystkie serwomotory stawów biodrowych po ustawieniu pod kątem 90° powinny być równoległe do osi *x*.

## Regulacja segmentów udowych

Regulacja segmentów udowych jest podobna do opisanej wyżej regulacji stawów biodrowych. Jak poprzednio, segmenty udowe po ustawieniu serwowymotorów w pozycji 90° powinny być równoległe do osi *x*:

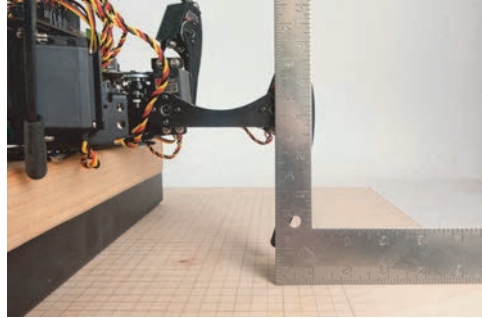
1. Patrząc na robota od przodu lub od tyłu, wyreguluj serwowymotory segmentów udowych tak, aby segmenty były skierowane równoległe do podłoża.
2. Od otrzymanych wartości odejmij liczbę 90, aby obliczyć wartości właściwości *offset*.
3. Ustaw właściwości *offset* dla wszystkich sześciu obiektów serwowymotorów segmentów udowych *l1f*, *r1f*, *l2f*, *r2f*, *l3f* oraz *r3f*. [Rysunek 5.11](#) przedstawia wyregulowane segmenty.



Rysunek 5.11. Wyregulowany segment udowy

## Regulacja segmentów piszczelowych

Na koniec wyreguluj segmenty piszczelowe. Tym razem musisz wyregulować każdy serwowymotor tak, aby po ustawieniu go w pozycji 90° linia poprowadzona od środka nasadki serwowymotora segmentu piszczelowego do jego końca tworzyła z podłożem kąt prosty (patrz [rysunek 5.12](#)).



Rysunek 5.12. Wyregulowany segment piszczelowy

Po wyczeniu wszystkich wartości właściwości *offset* wpisz je w pliku *phoenix.js* w definicjach obiektów *l1t*, *r1t*, *l2t*, *r2t*, *l3t* i *r3t*.

## Określenie zakresów ruchu serwowymotorów

Teraz określ zakres ruchu każdego serwowymotora. Większość serwowymotorów nie obraca się w pełnym zakresie 180°. Jest to normalne. Aby uzyskać pełny zakres, może być wymagane przeprogramowanie serwowymotora lub włączenie pomiędzy serwowymotorem a nakładką silnikową układu zwiększającego zakres obrotu. W przypadku heksapoda nie będzie potrzebny pełny zakres, więc nie musisz się tym przejmować. W rzeczywistości będziesz musiał zakres jeszcze bardziej ograniczyć, aby serwowymotory nie ustawiały się w pozycjach powodujących zaczepianie nóg o siebie.

W tej części będziesz za każdym razem sterował tylko jednym serwowymotorem. Wykonaj następujące czynności:

1. Jak poprzednio, zacznij od stawu biodrowego nogi L2. Aby określić właściwe granice, musisz obracać serwowymotorem w obu kierunkach. Serwowymotor jest w stanie obracać się w większym zakresie, niż jest to konieczne.
2. Metodą prób i błędów określ zakres obrotów w obu kierunkach. Gdy przy pewnej wartości serwowymotor przestanie reagować, będzie to znak, że wykroczyłeś poza zakres ruchu. Jeżeli na przykład serwowymotor obróci się przy wartości 165,

ale nie będzie reagował dla wartości 166, będzie to oznaczać, że górną granicę zakresu stanowi wartość 165. Proces regulacji wygląda następująco:

```
ph.12c.to(60); // Wciąż jest miejsce
ph.12c.to(58); // To dobra pozycja
ph.12c.to(122); // Segmenty udowe zaczepiają
                // o siebie
ph.12c.to(119); // SUPER!
```

3. W pliku *phoenix.js* odszukaj wiersz, w którym tworzony jest obiekt 12c, i we właściwości range wpisz określone przed chwilą wartości:

```
12c = new five.Servo({
  pin:27,
  invert: true,
  offset: 3,
  startAt: 90,
  range: [58, 119]
})
```

4. Powtórz kroki 1. – 3. dla pozostałych pięciu serwowymotorów stawów biodrowych i zmień właściwości range w obiektach 11c, 13c, r1c, r2c oraz r3c.

W przypadku serwowymotorów segmentów udowych i piszczelowych zakres ruchu jest znacznie większy, ponieważ istnieje mniejsze prawdopodobieństwo, że podczas obracania serwowymotorów nogi będą zaczepiać o siebie. W praktyce prawdopodobnie będziesz wykorzystywał pełny zakres ruchu serwowymotorów zarówno segmentów udowych, jak i piszczelowych.

5. Poeksperymentuj i znajdź wartości, przy których serwowymotory już się nie obracają. Możesz zastosować tę samą metodę prób i błędów jak w przypadku serwowymotorów stawów biodrowych. Pamiętaj, że zakresy ruchu wszystkich serwowymotorów segmentów udowych powinny być takie same (lub podobne).
6. Zmień właściwości range dla wszystkich obiektów serwowymotorów segmentów udowych: 11f, 12f, 13f, r1f, r2f i r3f.
7. Zakresy ruchu wszystkich serwowymotorów segmentów piszczelowych również muszą być takie same. Określ te zakresy.
8. W pliku *phoenix.js* zmień właściwości range w obiektach 11t, 12t, 13t, r1t, r2t, r3t.

Korzyści z określenia rzeczywistego zakresu ruchu serwowymotorów mogą nie być oczywiste. Gdybyś ustawił kąt obrotu na 175° dla serwowymotora, który może ustawić się pod kątem 165°, wtedy końce nóg nie zostaną ustawione w pozycjach, których oczekuje program *phoenix.js*, i klasa *Animation* przyjmie, że robot wciąż się porusza, mimo że serwowymotor się nie obraca. Nie będzie to pożądanym efektem.

Po zakończeniu konfiguracji obiekty serwowymotorów segmentów udowego i piszczelowego nogi L2 będą wyglądały jak poniżej:

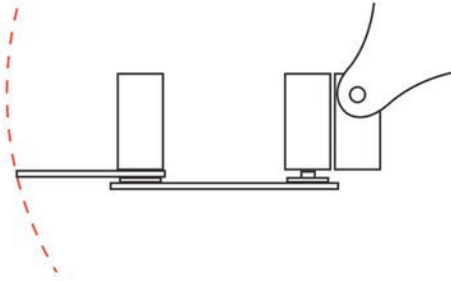
```
12f = new five.Servo({
  pin:22,
  invert: true,
  offset: -2,
  startAt: 180,
  range: [25, 165]
}),
12t = new five.Servo({
  pin:21,
  invert: true,
  offset: 4,
  startAt: 180,
  range: [21, 159]
}),
```

Po ustawieniu wartości właściwości `offset` i `range` dla wszystkich 18 serwowymotorów czas na zabawę! Uruchoom robota!

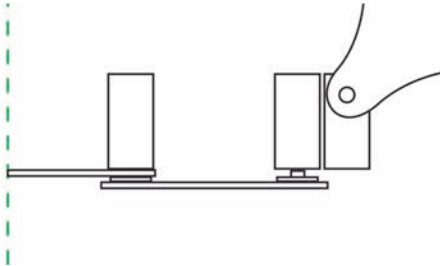
## Chodzenie jest trudne!

Oczywiście jest możliwe poruszanie robotem bez korzystania z klasy *Animation*, ale wymaga to włożenia mnóstwa pracy. Być może wydaje Ci się, że przesunięcie robota w przód wymaga jedynie postawienia końca nogi na podłożu i obrócenia stawu biodrowego wstecz. Problem jednak polega na tym, że końcówka nogi zakreśla łuk o środku znajdującym się w osi obrotu stawu biodrowego, jak pokazuje [rysunek 5.13](#). Każda noga zakreśla wtedy własny łuk i utrudnia funkcjonowanie innych nóg. Z tego powodu chodzenie wygląda bardzo niezgrabnie.

Aby zniwelować ten efekt i aby końcówka nogi poruszała się po linii prostej równoległej do kierunku ruchu robota, jak pokazuje [rysunek 5.14](#), trzeba segment



Rysunek 5.13. Ten robot chodzi, jakby był pijany



Rysunek 5.14. Taka sekwencja ruchów zapewnia płynne chodzenie

piszczelowy przesuwać do wewnątrz i na zewnątrz. Ale zaczekaj! Podczas przesuwania segmentu piszczelowego do wewnątrz i na zewnątrz końcówka nogi porusza się w górę i w dół w stosunku do podłoża, więc trzeba jeszcze podnosić i opuszczać segment udowy, aby skompensować ten efekt.

Aby robot poruszał się płynnie, wszystkie 18 serwo-motorów musi nieustannie ze sobą współpracować w zgrany sposób jak w orkiestrze. To zadanie może wydawać się bardzo trudne, ale platforma Johnny-Five i klasa `Animation` pozwala nad wszystkim zapanować.

## Poznaj klasę `Animation`

Klasa `Animation` jest wykorzystywana do sterowania robotami za pomocą skryptów definiujących ich ruchy. Klasa ta stanowi opakowanie klasy `Servo`. Oferuje harmonogramy, kluczowe ramki, płynne przejścia z jednej ramki do drugiej (ang. *tweening*) i punkty kontrolne (ang. *cue points*). Grupę docelową stanowi

sterowany serwomotor lub grupa serwomotorów. Grupą może być również tabela serwomotorów, czyli obiekt `Servo.Array`, lub tabela serwomotorów i obiektów `Servo.Array`. Ponieważ może to wydawać się skomplikowane, poznaj kilka przykładów.

## Tabela serwomotorów jako grupa docelowa

W listingu 5.1 trzy serwomotory wchodzące w skład nogi L1 są poruszane niezależnie.

Listing 5.1. Tabela serwomotorów

```
board = new five.Board().on("ready",
function() {
  var l1c = new five.Servo(27);
  var l1f = new five.Servo(26);
  var l1t = new five.Servo(25);
  var myAnimation =
    new five.Animation([ l1c, l1f, l1t
  ]);
  // ...
});
```

Taki kod pozwala wystać do każdego z serwomotorów inną wartość w dowolnym punkcie kontrolnym harmonogramu.

## Obiekt `Servo.Array` jako grupa docelowa

Wynik wykonania kodu przedstawionego w listingu 5.2 jest taki sam jak poprzedniego kodu. Zwróć jednak uwagę, że jako domyślna grupa docelowa przekazywany jest jeden obiekt `Servo.Array`. Podobnie jak poprzednio, grupa jest jednowymiarową tabelą.

## Tabela obiektów `Servo.Array` jako grupa docelowa

W ostatnim listingu 5.3 serwomotory są zgrupowane według typów stawów. Dzięki temu możliwe jest sterowanie serwomotorami stawów biodrowych jako jedną grupą, serwomotorami segmentów udowych

### Listing 5.2. Obiekt Servo.Array

```
board = new five.Board().on("ready", function() {
  var l1c = new five.Servo(27);
  var l1f = new five.Servo(26);
  var l1t = new five.Servo(25);
  var l1 = new Servo.Array([ l1c, l1f, l1t ]);
  var myAnimation = new five.Animation(l1);
  // ...
});
```

jako drugą grupą i serwowymi segmentów piszczałkowych jako trzecią. Przekazanie wartości kątowej w argumencie metody obiektu reprezentującego całą grupę powoduje przekazanie tej samej wartości w argumentach metod wszystkich obiektów serwowymotorów w danej grupie.

### Listing 5.3. Tabela obiektów Servo.Array

```
board = new five.Board().on("ready",
↳function() {
  var l1c = new five.Servo(27);
  var l1f = new five.Servo(26);
  var l1t = new five.Servo(25);
  var l2c = new five.Servo(23);
  var l2f = new five.Servo(22);
  var l2t = new five.Servo(21);
  var coxa = new Servo.Array([ l1c,
↳l2c ]);
  var femur = new Servo.Array([ l1f,
↳l2f ]);
  var tibia = new Servo.Array([ l1t,
↳l2t ]);
  // Dwuwymiarowa tabela tabel
  var myAnimation = new five.Animation([
↳coxa, femur, tibia ]);
  // ...
});
```

Dzięki zastosowaniu klasy `Servo.Array` można w jednej grupie docelowej umieszczać serwowymotory i inne obiekty `Servo.Array`. Grupą serwowymotorów w tabeli `Servo.Array` można sterować tak samo jak każdym innym urządzeniem.

Teraz przyjrzyj się wierszowi pliku `phoenix.js`, w którym tworzona jest instancja klasy `Animation` i definiowana domyślna grupa docelowa:

```
var legsAnimation = new
five.Animation(phoenix.legs);
```

W powyższym wierszu domyślna grupa docelowa `phoenix.legs` jest obiektem typu `Servo.Array` zawierającym 18 serwowymotorów robota. Ta domyślna grupa wymaga określenia wartości dla każdego serwowymotora w każdym punkcie kontrolnym harmonogramu.

## Pierwszy segment animacyjny

Po utworzeniu obiektu animacyjnego można zdefiniować segmenty animacyjne. Segment animacyjny to krótka sekwencja ruchów. Segmenty to miejsca, w których zdefiniowane są wszystkie kąty obrotów serwowymotorów, punkty kontrolne, czasy trwania ruchu i funkcje wygładzające. Krótko mówiąc, jest to miejsce, w którym dzieje się cała magia animacyjna.

Segmenty animacyjne są wykonywane synchronicznie. Po umieszczeniu w kolejce animacyjnej wykonywane są zgodnie z zasadą „pierwszy wchodzi, pierwszy wychodzi”. Po uruchomieniu programu `phoenix.js` segmenty można wywoływać z pętli REPL. Poniższa seria poleceń powoduje, że robot powstaje, idzie, zatrzymuje się i zasypia:

```
> node phoenix
ph.stand();
ph.walk();
ph.stop();
ph.sleep();
```

Powyższe segmenty zostały już zdefiniowane w programie *phoenix.js*, więc nie musisz ich tworzyć ani edytować. Poniżej przedstawione są fragmenty programu, które ułatwią Ci poznanie zawartości segmentów animacyjnych.

Przyjrzyj się pokazanemu w [listingu 5.4](#) pierwszemu zdefiniowanemu w pliku *phoenix.js* segmentowi animacyjnemu.

- 1 Opcjonalna właściwość `target` definiuje grupę animowanych serwowatorów. Po utworzeniu obiektu typu `Animation` domyślną wartością tej właściwości jest grupa docelowa `phoenix.legs`. W tym segmencie jest ona zastępowana grupą `phoenix.altJoints`. Wpisując inną wartość tej właściwości, można sterować inną grupą serwowatorów. Wszystkie serwowatory w grupie są skojarzone z tym samym harmonogramem i kolejną animacyjną. Jeżeli we właściwości `target` nie wskażesz własnej grupy, wtedy w danym segmencie zostanie użyta domyślna grupa docelowa.
- 2 W opcjonalnej właściwości `duration` określanej jest w milisekundach czas trwania animacji. Zwróć uwagę, że zmiana prędkości animacji podczas odtwarzania segmentu powoduje wydłużenie lub skrócenie jego czasu trwania (domyślnie właściwość ta ma wartość 1000 ms).

- 3 Opcjonalna właściwość `cuePoints` (punkty kontrolne) jest jednowymiarową tabelą wartości z zakresu od 0,0 do 1,0. Każdy segment animacyjny ma własny harmonogram. W harmonogramie można zdefiniować dowolną liczbę punktów kontrolnych, w których urządzenia są sterowane za pomocą kluczowych ramek. Punkty kontrolne i kluczowe ramki nie muszą być rozmieszczone w regularnych odstępach czasu. W tym przypadku, gdy czas trwania segmentu jest równy 500 ms, punkty kontrolne odpowiadają wartościom 50 ms, 150 ms, 350 ms i 500 ms. Zmiana czasu trwania animacji powoduje zmianę skali czasu, w którym realizowane są wszystkie punkty kontrolne (domyślnie są to punkty [0, 1]).
- 4 Ta funkcja jest wywoływana po zakończeniu wykonywania segmentu animacyjnego.
- 5 Właściwość `keyFrames` (kluczowe ramki) jest właściwością obowiązkową. Jest to dwuwymiarowa tabela. Pierwszy wymiar odpowiada urządzeniom w grupie docelowej i jego długość musi być równa długości grupy. Aby określić urządzenia powiązane z każdym elementem pierwszego wymiaru, przyjrzyj się kodowi przedstawionemu w [listingu 5.5](#), w którym zdefiniowana jest grupa docelowa `phoenix.altJoints`.

**Listing 5.4.** Segment animacyjny powodujący powstanie robota

```
var stand = {
  target: phoenix.altJoints, ❶
  duration: 500, ❷
  cuePoints: [0, 0.1, 0.3, 0.7, 1.0], ❸
  oncomplete: function() { ❹
    phoenix.state = "stand";
  },
  keyFrames: [ //5
    [null, { degrees: 90 }],
    [null, { degrees: 66 }],
    [null, false, false, { degrees: 120, easing: easeOut},
     { degrees: 94, easing: easeIn}],
    [null, false, { degrees: 106}, false, { degrees: 93 } ]
  ]
};
```



### Listing 5.5. Grupa docelowa phoenix.altJoints

```
altJoints = new five.Servo.Array([
  phoenix.innerCoxa,
  phoenix.outerCoxa,
  phoenix.femurs,
  phoenix.tibia
]),
```

W grupie phoenix.altJoints znajdują się obiekty reprezentujące cztery urządzenia: phoenix.midCoxa, phoenix.outerCoxa, phoenix.femur i phoenix.tibia. Każdy z nich jest obiektem typu Servo.Array, zawierającym dwa, cztery lub sześć obiektów reprezentujących serwomotory:

- element keyFrames[0] jest skojarzony z serwomotorami środkowych stawów biodrowych,
- element keyFrames[1] jest skojarzony z serwomotorami przednich i tylnych stawów biodrowych,
- element keyFrames[2] jest skojarzony z serwomotorami segmentów udowych,
- element keyFrames[3] jest skojarzony z serwomotorami segmentów piszczelowych.

Każda tabela keyFrames zawiera od 1 do  $n$  elementów, gdzie  $n$  oznacza liczbę punktów kontrolnych. Przeanalizujemy kilka tabel z segmentu stand powodującego powstanie robota. Najpierw przyjrzymy się elementowi keyFrames[0] odpowiadającemu grupie innerCoxa:

```
[null, false, { degrees: 88 }],
```

W elemencie keyFrames[0][0] zdefiniowana jest wartość dla pierwszego urządzenia w grupie docelowej, wykorzystywana w punkcie kontrolnym w chwili 0 ms. W elemencie keyFrames[0][1] zdefiniowana jest wartość wykorzystywana w punkcie kontrolnym 50 ms. Do określenia wartości pośrednich w przedziale czasu od 0 ms do 50 ms wykorzystywana jest funkcja wygładzająca.

Pierwsza wartość w pierwszym elemencie keyFrames jest równa null. Oznacza ona, że animacja powinna rozpoczynać się od bieżącej pozycji urządzenia. Jeżeli urządzenie jest zdefiniowane za pomocą obiektu Servo.Array, wtedy bieżąca pozycja jest

odczytywana z pierwszego elementu tego obiektu. Jeżeli wartość null znajduje się w innym elemencie tabeli keyFrames, wówczas klasa Animation pomija ramkę dla danego urządzenia. W tej tabeli zdefiniowane są wartości tylko pierwszych dwóch punktów kontrolnych. We wszystkich pozostałych wykorzystywana jest ostatnia zdefiniowana wartość (tj. 70).

Teraz przyjrzymy się elementowi keyFrames[3] (odpowiadającemu serwowmotorom segmentów piszczelowych):

```
[null, { degrees: 110}, false,
 ↪{ degrees: 110}]
```

Jeżeli element w tabeli keyFrames zawiera wartość false, wówczas klasa Animation kopiuje wartość obliczoną dla poprzedniego elementu. Umieszczenie w pierwszych dwóch elementach wartości null i false powoduje, że klasa Animation w ogóle nie będzie sterować urządzeniami, dopóki nie minie drugi punkt kontrolny.

Jeżeli w elemencie zostanie podana liczba, wtedy klasa Animation doda ją do poprzedniej wartości.

Ten segment animacyjny powoduje, że heksapod przechodzi z pozycji uśpienia w pozycję wyjściową (pozycja wyjściowa jest przyjmowana jako pozycja startowa w większości segmentów animacyjnych).



Gry porównasz kod z listingu 5.5 z kodem w pliku phoenix.js, zauważysz różnice. W pliku zamiast jawnych wartości liczbowych są użyte elementy z tabeli h. Tabela ta, zdefiniowana na początku pliku, zawiera informacje o pozycjach serwowmotorów dla wszystkich podstawowych kroków robota. Dzięki zapisaniu tych wartości w jednym miejscu łatwiej je stosować i modyfikować w dalszej części kodu. Ponieważ zastosowanie elementów tabeli powoduje, że kod jest bardziej skomplikowany, w opisywanych przykładach zostały użyte liczby.

## Chodzenie

Sekwencje kroków robota dzielą się na dwie główne kategorie: statyczne i dynamiczne. Podczas wykonywania kroków statycznych środek ciężkości robota w każdym momencie jest utrzymywany na stabilnym oparciu (przynajmniej na trzech nogach). Innymi słowy robot nie przewróciłby się, gdyby w dowolnym momencie został zatrzymany w pół kroku. Podczas wykonywania kroku dynamicznego przez pewien czas występuje stan nierównowagi. Program *phoenix.js* wykonuje tylko kroki statyczne.

Jak wspomniałem wcześniej, sekwencja kroków heksapoda jest bardziej skomplikowana, niż mogłoby się to wydawać. Aby przesunąć robota w przód, nie wystarczy poruszyć jedynie stawami biodrowymi, ponieważ spowodowałoby to obrócenie się końcówek nóg wokół osi serwowatorów stawów biodrowych. Kończówka nogi musi poruszać się wzdłuż linii prostej.

Przesuwanie końcówki nogi wzdłuż linii prostej wymaga skoordynowania ruchów wszystkich trzech serwowatorów nogi. Przesuwanie w przód całego szkieletu robota wymaga współpracy wszystkich 18 serwowatorów, które muszą ustawiać końcówki nóg w określonych miejscach na podłożu. Jeżeli serwo-

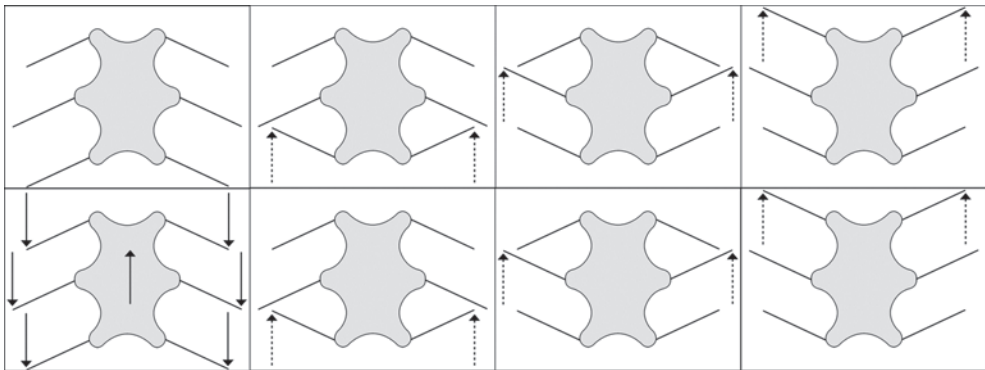
motory nie będą współpracować, robot będzie powłóczył nogami po podłożu i kroki nie będą płynne.

W programie *phoenix.js* zdefiniowane są cztery rodzaje kroków: wiosłowanie (*row*), kraul (*crawl*), marsz (*walk*) i bieg (*run*). W każdym rodzaju kroku przyjęty jest odcinek przesunięcia (odległość pomiędzy początkową a końcową pozycją końcówki nogi) równy 8 cm. Teraz przyjrzyjmy się dokładniej każdemu rodzajowi kroku.

### Wiosłowanie (*row*)

Sekwencja ruchów w tym kroku (zwanym również falą) powoduje przesuwanie nóg parami do przodu, a następnie przesuwanie szkieletu do przodu. Nie jest to elegancki krok (patrz [rysunek 5.15](#)), ale prosty w implementacji, ponieważ wymaga umieszczenia w tabeli `keyFrames` tylko ośmiu elementów.

Do zdefiniowania ruchu nogi potrzebne są tylko trzy punkty kontrolne. Pierwszy z nich zawiera wartość `null` lub `false`, więc ruch nogi rozpoczyna się od bieżących pozycji urządzeń, niezależnie od tego, jakie one są. Drugi punkt kontrolny definiuje moment zgięcia nogi, natomiast trzeci — jej końcowy ruch. Ilustruje to [listing 5.6](#).



Rysunek 5.15. Fala

Listing 5.6. Wiosłowanie

```
var row = {
  target: phoenix.jointPairs,
  duration: 1500,
```

```

cuePoints: [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.85, 1.0],
loop: true,
fps: 100,
onstop: function() { phoenix.att(); },
oncomplete: function() { },
keyFrames: [
  [null, null, null, null, false, null, {degrees: 56},
    false, {degrees: 70}, {degrees: 91}],
  [null, null, null, null, false, { step: 30, easing: easeOut },
    {degrees: 116}, false, {degrees: 120}, {degrees: 119}],
  [null, null, null, null, false, { step: -20, easing: easeOut },
    {degrees: 97}, false, {degrees: 110}, {degrees: 116}],
  // ...dwie następne pary nóg
]
}

```

## Marsz (walk)

W tym kroku (patrz [listing 5.7](#)) cztery nogi pozostają przez cały czas w kontakcie z podłożem. Krok rozpoczyna się od przesunięcia jednej nogi do przodu. Gdy noga zostanie zgięta, do przodu przesuwana jest druga noga. Pierwsza noga oprze się na podłożu w tym samym momencie, w którym zostanie zgięta druga noga. Jednocześnie do przodu przesuwana jest trzecia noga.

Graficzne przedstawienie kluczowych ramek dla tego kroku jest trudne, więc ich opis został pominięty. Gdybyś chciał zbadać, jak jest realizowany ten krok, ustaw właściwość `duration` segmentu na 20 000 ms. Krok będzie wtedy wykonywany w zwolnionym tempie.

Listing 5.7. Marsz

```

var walk = {
  duration: 2000,
  cuePoints: [0, 0.071, 0.143, 0.214, 0.286, 0.357, 0.429, 0.5,
    0.571, 0.643, 0.714, 0.786, 0.857, 0.929, 1],
  loop: true,
  loopback: 0.5,
  fps: 100,
  onstop: function() { phoenix.att(); },
  oncomplete: function() { },
  keyFrames: [
    [null, null, {degrees: 82}, null, null, null, null, {degrees: 82}, null,
      {56}, null, null, null, null, {degrees: 82}], //r1c
    [null, { step: 30, easing: easeOut }, {degrees: 119, easing: easeIn}, null,
      null, null, null, {degrees: 119}, { step: 30, easing: easeOut },
      {degrees: s.f.f[0], easing: easeIn}, null, null, null, null,
      {degrees: 119}],
    [null, { step: -20, easing: easeOut }, {degrees: 119, easing: easeIn}, null,
      null, null, null, {degrees: 119}, { step: -20, easing: easeOut },
      {degrees: 97, easing: easeIn}, null, null, null, null, {degrees: 119}],
    // ...pięć następnych nóg
  ]
};

```

## Bieg (run)

Ten krok (zwany popularnie trójnogiem, zdefiniowany w [listingu 5.8](#)) powoduje najszybsze przemieszczanie się robota, ale obciąża serwomotory segmentów udowych i piszczelowych. Stosuj go rzadko. W tym kroku jednocześnie poruszane są trzy nogi, natomiast trzy inne pozostają w kontakcie z podłożem i muszą one utrzymać większy ciężar niż w przypadku pozostałych kroków.

## Obracanie robota

Obracanie robota jest trudniejsze niż wykonywanie kroków. Kończówki nóg nie poruszają się wtedy po liniach prostych, ale zakreślają koncentryczne łuki o środkach znajdujących się w środku szkieletu robota. Promień łuku dla nóg R2 i L2 jest inny niż dla pozostałych czterech nóg.

Krok obracający robota jest bardzo podobny do kroku biegowego. Trzy nogi pozostają w kontakcie z podłożem, a pozostałe obracają się o kąt pomiędzy  $15^\circ$  a  $30^\circ$ .

W programie *phoenix.js* zdefiniowane są dodatkowe, nieopisane tutaj segmenty animacyjne: `sleep` (uśpienie), `waveRight` (przesunięcie przedniej prawej

nogi), `waveLeft` (przesunięcie przedniej lewej nogi) i `crawl` (pełzanie). Wypróbuj je po wyregulowaniu robota i uzyskaniu prawidłowego kroku.

## Lista poleceń

Wszystkie poniższe polecenia można wpisywać w pętli REPL podczas ruchów robota:

- `ph.stand()` (powstanie)
- `ph.sleep()` (uśpienie)
- `ph.walk()` (marsz)
- `ph.crawl()` (pełzanie)
- `ph.row()` (wiosłowanie)
- `ph.badRow()` (inne wiosłowanie)
- `ph.run()` (bieg)
- `ph.run("rev")` (bieg do tyłu)
- `ph.turn()` (obróć)
- `ph.turn("left")` (obróć w lewo)
- `ph.stop()` (zatrzymanie)
- `ph.waveLeft()` (przesunięcie przedniej prawej nogi)
- `ph.waveRight()` (przesunięcie przedniej lewej nogi)
- `ph.att()` (powrót do pozycji wyjściowej)

Listing 5.8. Bieg

```
var run = {
  duration: 1000,
  cuePoints: [0, 0.25, 0.5, 0.75, 1.0],
  loop: true,
  fps: 100,
  onstop: function() { phoenix.att(); },
  oncomplete: function() { },
  keyFrames: [
    [ null, {degrees: 70}, {degrees: 56}, null, {degrees: 91}],
    [ null, {degrees: 120}, {degrees: 116}, { step: 30, easing: easeOut },
      {degrees: 119, easing: easeIn}],
    [ null, {degrees: 110}, {degrees: 97}, { step: -20, easing: easeOut },
      {degrees: 116, easing: easeIn}],
    // ...5 następnych nóg
  ]
};
```

Wypróbuj śmiało powyższe polecenia i sprawdź, do czego służą.

## Co dalej?

---

Teraz Twój heksapod powinien na komendę chodzić, obracać się i robić kilka innych rzeczy. Ale to dopiero początek. Możesz go wykorzystać do tworzenia nowych, niesamowitych konstrukcji. Poniżej wymienionych jest kilka pomysłów, które możesz zrealizować:

### *Utwórz inne segmenty animacyjne.*

Użyj podstawy testowej i swojej wyobraźni. Podziel się na forum <http://forums.nodebots.io> efektami z innymi konstruktorami heksapodów.

### *Dodaj kontroler.*

Wpisywanie poleceń w terminalu jest uciążliwe. Powinieneś zbadać inne możliwości sterowania heksapodem. Dostępne są interfejsy API platformy Node.js do następujących kontrolerów:

- Leap Motion Controller (<http://bit.ly/19LY4nm>),
- nakładka SparkFun Joystick — po prostu użyj platformy Johnny-Five,
- Kinect (<http://bit.ly/1bQRz4l>),
- Wii Motion Controller — po prostu użyj platformy Johnny-Five,
- Brainwaves (<http://bit.ly/19LYoT4>).

### *Dodaj czujniki.*

Dodaj kamerę i czujniki ultradźwiękowe, aby robot mógł orientować się w swoim otoczeniu. Naucz go podejmować decyzje oparte na informacjach odbieranych z tych urządzeń.

### *Zbuduj innego heksapoda.*

Przedstawiony w tym rozdziale kod nadaje się nie tylko do sterowania robotem Phoenix. Dostępnych jest wiele innych robotów, pod adresem <http://www.thingiverse.com> można znaleźć nawet ich ogólnodostępne projekty do wykonania na drukarce 3D. Będziesz musiał zmienić wartości w tabelach h, t, s oraz l, ale jeżeli zbudujesz stanowisko testowe i wyregulujesz wszystkie serwomotory, będzie to całkiem proste zadanie.

Decyzja o zastosowaniu platform Node.js i Johnny-Five okazała się słuszna. Wszystkie pakiety npm znajdują się w zasięgu palców. Dzięki temu możesz podążać dowolnie wybraną ścieżką. Co więcej, Johnny-Five jest najbardziej elastyczną i najlepiej obsługiwaną platformą wykorzystywaną podczas budowania robotów. Realizuj więc śmiało wszystkie pomysły, jakie Ci przyjdą do głowy.

Rozwój społeczności majsterkowiczów oraz potęga platformy Node.js sprawiają, że bardzo przyjemne jest być robotykiem amatorem!



---

# Skorowidz

---

## A

adapter WiFi, 174, 245  
adres  
  I2C, 149  
  IP, 79  
akumulator, 18  
algorytm sterowania robotem,  
  208  
anatomia  
  ramienia robota, 33  
  robota delta, 219  
animacja, 95  
  Android, 115  
aplikacja WWW, 78, 79, 155

## B

BatBot  
  algorytm, 214  
  czujnik ultradźwiękowy, 210  
  diagnostyka, 215  
  implementacja algorytmu,  
    213  
  montaż, 206  
  poruszanie robotem, 208  
  sterowanie robotem, 210  
  wykaz materiałów, 205  
  zdalne sterowanie, 207

bezpieczeństwo, 189  
bezwodowodowe sterowanie  
  robotem, 27  
bezwodowodowy robot  
  SimpleBot, 24  
biblioteka  
  Johnny-Five, 156  
  Node.js, 243  
  Twilio, 193  
  Volley, 248  
  wscat, 74  
bieg, 100  
budowa  
  łańcucha światełek, 147  
  todzi, 50  
  obwodu mikrofonu, 108  
  obwodu przekaźnika, 106  
  robota bezwodowodowego, 24  
  robota SimpleBot, 16  
  zegara słonecznego, 128  
budowanie robotów, 15

## C

chodzenie, 93, 98, 175  
czujnik, 234  
  HC-SR04, 190  
  ultradźwiękowy, 190, 206  
  wody, 69

## D

dekorowanie łańcucha światełek,  
  153  
diagram pinów sterownika  
  silnika, 60  
dioda  
  LED, 196  
  RGB, 162  
dobór kontrolera matrycy, 146

## E

elementy  
  elektroniczne lampy  
    CheerfulJ5, 160  
  interaktywnych butów, 232  
  panelu LED RGB, 174  
  robota  
    BatBot, 205  
    heksapod, 84  
    Junky Delta, 218  
    NodeBoat, 50  
    NodeBot, 104  
    piDuino5, 72  
    SimpleBot, 16  
    TypeBot, 32  
  zegara słonecznego, 126

## F

- fala, 98
- fotorezystor, 183
- fritzing, 37
- funkcja, *patrz także* metoda
  - callback, 41
  - delta\_calcAngleYZ, 227
  - getLatestColor, 165
  - min(), 67
  - setInterval(), 165
  - sunPositionInDegrees, 143
  - tick, 141
  - ultraData(), 200

## G

- generator express, 155
- gnomon, 125

## H

- heksapod
  - chodzenie, 98
  - lista poleceń, 100
  - montaż robota, 86
  - obracanie robota, 100
  - program phoenix.js, 85
  - regulacja serwowatorów, 91
  - sterowanie robotem, 84
  - układ współrzędnych, 90
  - wykaz materiałów, 83
  - zakres ruchu serwowatorów, 92

## I

- informacje o serwowatorach, 32
- inicjalizacja
  - modułu, 41
  - platformy Johnny-Five, 75
- instalacja
  - biblioteki Node.js, 243
  - biblioteki Volley, 248
  - oprogramowania, 73
  - oprogramowania Android Studio, 247
  - platformy Node.js, 73

- instancja platformy Johnny-Five, 243
- interaktywne buty, 231
  - czujniki, 235
  - kod, 238
  - podłączenie do Arduino, 237
  - połączenie butów, 235
  - stan lewego buta, 240
  - wykaz materiałów, 231
  - wyświetlanie danych, 239
- interaktywny panel LED RGB, 173
- interfejs
  - API, 156
  - ThingSpeak API, 164, 166
  - Twitter Streaming API, 165
  - użytkownika, 79, 157
  - Web Speech API, 112

## J

- joystick, 79
  - dotykowy, 80
- Junky Delta
  - anatomia robota, 219
  - budowa, 220
  - ruch robota, 223
  - wykaz materiałów, 218

## K

- kabel, 150
- kinematyka, 217, 225
- klasa
  - Animation, 83, 90, 94, 95
  - AsyncTask, 122
  - Led, 75
  - Led.Matrix, 158
  - Matrix, 154
  - Servo.Array, 95
  - WebSocketServer, 75
- kod
  - do lampy CheerfulJ5, 161
  - funkcji tick, 141
  - servo-test.js, 20
  - simplebot.js, 22
  - sterujący kolorami, 184
  - sterujący obwodem, 109
- komponenty zasobnika silnika, 51

- komunikaty Firmata, 27
- konfiguracja
  - adaptera WiFi, 245
  - Arduino, 26
  - czujników, 239
  - modułu Spark, 56
  - modułu WiFi, 25
  - serwowatorów, 41, 130, 131
  - sprzętu, 244
- konsola DualShock, 210
- konstrukcja ramienia, 33
- kontroler
  - głosowy, 112, 115
  - matrycy, 146

## L

- lampa CheerfulJ5, 159
  - kod, 161
  - łączność bezprzewodowa, 169
  - moduł Spark, 171
  - obwód, 160
  - umieszczenie obwodu, 171
  - warianty obudowy, 161
  - wykaz materiałów, 159
- laser, 195
- lista serwowatorów, 86
- lutowanie sterownika silnika, 58

## Ł

- łączenie
  - przewodów, 150
  - serwowatorów, 37
- łączność bezprzewodowa, 24, 169
- łokieć, 35
- tuk wysokości, 137

## M

- mapa kolorów, 163
- marsz, 99
- matryca
  - LED, 147
  - świetlna, 188
- menedżer Homebrew, 27



metoda, *patrz też* funkcja  
  callback, 41  
  forward(), 76  
  init, 39  
  joystick.left(), 80  
  joystick.up(), 80  
  led.color(), 165  
  motor.forward(), 63  
  move, 40  
  sonarServo.center(), 210  
  sonarServo.max(), 210  
  temporal.queue(), 68  
  tick, 45  
mikrofon, 108  
mobilna platforma, 71  
mocowanie  
  akumulatora, 18, 76  
  efektora, 222  
  przewodów, 37  
  ramion, 222, 223  
  segmentu nadgarstka, 37  
  serwomotoru, 18  
  łokcia, 36  
  nadgarstka, 36  
  zasobnika z silnikiem, 65  
modulacja szerokości impulsu, 60  
moduł  
  dualshock-controller, 210  
  keypress, 63  
  Spark, 56, 58, 59  
  Spark Core, 58, 169  
  Spark-IO, 58  
  temporal, 68  
  WiFi, 24  
  WIFI232, 27  
modyfikacja silnika, 52  
monitoring, 194  
montaż  
  kadłuba łodzi, 64  
  komponentów, 76  
  komponentów  
    elektronicznych, 87  
  segmentów puszczelowych, 89  
  segmentów udowych, 89  
  silnika, 53  
  stawów biodrowych, 88  
  steru, 68  
  zasobnika na silnik, 55  
  zegara, 136

## N

nadgarstek, 36  
napęd  
  różnicowy, 23  
  Tamiya, 52  
narzędzia programistyczne, 155  
nawiązanie połączenia, 80  
NodeBoat  
  łączenie komponentów, 58  
  moduł Spark, 56  
  sterowanie łodzią, 66  
  sterownik silnika, 58, 62  
  wodowanie łodzi, 64  
  wykaz materiałów, 50  
  zasobnik z silnikiem, 51  
NodeBot  
  aplikacja Android, 115  
  kontroler głosowy, 112, 115  
  mikrofon, 108  
  obwód przekaźnika, 106  
  płytką BeagleBone Black, 105  
  serwer poleceń, 110  
  tworzenie projektu, 106  
  wykaz materiałów, 103

## O

obiekt  
  Board, 58  
  MatrixView, 158  
  Motor, 62, 66  
  Servo, 66  
  Servo.Array, 94, 95  
obliczenie kąta, 226  
obracanie robota, 100  
obsługa zdarzeń, 62, 63  
obudowa Beaglebone-io, 181, 182  
obwód  
  mikrofonu, 108  
  przedwzmacniacza  
    mikrofonu, 108  
  przekaźnika, 106  
odwrócona kinematyka, 226  
opcje konstruktora klasy, 154  
oprogramowanie  
  Firmata, 20  
  LEDScape, 175  
  VoodooSpark, 170

## P

palec, 33, 37  
panel LED RGB, 173, 175  
  czujniki, 175  
  kod, 178, 180  
  komponenty, 175  
  obudowa Beaglebone-io, 182  
  oprogramowanie, 175  
  schemat podłączenia, 177  
  wykaz materiałów, 174  
  zmienianie kolorów, 185  
parametr nonblock, 28  
Particle WiFi Development Kit, 245  
pętla  
  REPL, 84, 163  
  sterująca  
    otwarta, 213  
    zamknięta, 213  
pianka PCV, 135  
piDuino5  
  montaż komponentów, 76  
  sterowanie smartfonem, 78  
  sterowanie urządzeniami, 75  
  wykaz materiałów, 72  
pierwsze uruchomienie, 46  
platforma  
  Johnny-Five, 38, 75, 181, 243  
  Node.js, 73  
  ThingSpeak API, 164  
plik  
  activity\_main.xml, 116  
  align.js, 36  
  AndroidManifest.xml, 120  
  app.js, 79, 156  
  boat.js, 62  
  circuit.js, 113  
  CommandRequest.java, 115,  
    118  
  index.html, 79, 111, 112  
  lights.js, 156  
  ListenerService.java, 119  
  MainActivity.java, 116  
  moveBot.js, 209  
  package.json, 57, 68, 193  
  phoenix.js, 88  
  rect\_activity\_wear\_main.xml,  
    123

- plik
    - relay.js, 107
    - server.js, 110, 112, 114
    - servos.js, 131
    - sundial.js, 139
    - typebot.js, 41
    - WearMainActivity.java, 120
  - plytka
    - Arduino, 244
    - Arduino Uno, 31
    - BeagleBone Black, 105, 173, 244
    - Raspberry Pi, 73, 244
  - podłączenie
    - Arduino, 74
    - butów, 237
    - czujnika HC-SR04, 190
    - diody laserowej, 196
    - fotorezystora, 183
    - matrycy, 151, 152
    - modułu Spark, 57
    - modułu WiFi, 25
    - panelu RGB LED, 176
    - plytki Arduino, 151
    - serwomotorów, 19, 88, 130
    - silnika, 61
    - sterownika silnika, 60
    - zasilania, 59
  - podparcie tarczy, 134
  - podstawa, 34
  - pokoju zegar słoneczny, 125
  - połączenie
    - butów, 235
    - modułu Spark i silnika, 60
    - z każdego miejsca, 76
    - z modulem WiFi, 26
    - z serwomotorem, 66
  - pozycja początkowa robota, 42
  - prezentacja urządzenia, 189
  - prędkość obrotowa, 33
  - program
    - cheerful.js, 162, 166
    - cheerful-twit.js, 168
    - matrix-test.js, 154
    - open-pixel-test.js, 179
    - phoenix.js, 85
    - socat, 27
    - Spark, 57
    - sterujący, 21
    - sundial.js, 139, 140, 141
    - VSPE, 28
    - warmup.js, 224
  - programowanie, 247
    - serwomotoru, 66
  - projekt VoiceController, 247
  - protokół WebSocket, 74, 75
  - prototyp systemu
    - bezpieczeństwa, 201
  - przedwzmacniacz mikrofonu, 108
  - przeglądarka, 75
  - przełącznik, 106
  - przeliczanie odczytów, 186
  - przewód mikro-USB, 50
  - przycisk resetowania, 27
  - przygotowanie
    - czujników, 234
    - kabla, 150
    - matryc, 149
    - serwomotorów, 87
  - przyspieszoniomierz, 185
  - PWM, pulse-width modulated, 33, 60
- ## R
- ramię, 38
    - azymutu, 128, 136
    - robota, 32, 33
  - regulacja
    - ramienia, 46
    - segmentów puszczelowych, 92
    - segmentów udowych, 92
    - serwomotorów, 91
    - stawów biodrowych, 91
  - rejs, 69
  - resetowanie
    - Arduino, 28
    - układów, 28
  - robot
    - Army, 217, 229, 230
    - BatBot, 203
    - Junky Delta, 217
    - NodeBoat, 49
    - NodeBot, 15, 103
    - piDuino5, 71, 76
    - SimpleBot, 15
    - TapsterBot, 217, 229
    - TypeBot, 31
  - roboty
    - autonomiczne, 204
    - delta, 217
    - piszące, 31
    - semiautonomiczne, 204
    - sterowane głosem, 103
    - zdalnie sterowane, 204
  - rozpoznawanie mowy, 112
  - ruch ramienia, 34
  - rysowanie prostokąta, 228
  - rzepy, 76
- ## S
- schemat
    - ideowy podłączenia czujnika, 207
    - obwodu przełącznika, 107
    - podłączenia fotorezystora, 184, 186
    - połączeń robota SimpleBot, 19
    - ruchów robota, 208
    - stanów, 43
  - segment
    - animacyjny, 95
    - palca, 37
  - sekwencja naciśnięć klawiszy, 43
  - serwer
    - poleceń, 110
    - WWW, 75
  - serwis GitHub, 201
  - serwomotor, 17, 18, 21, 86
  - silnik, 53
  - SimpleBot
    - budowa robota, 16
    - diagnostyka problemów, 21, 28
    - łączność bezprzewodowa, 24
    - sterowanie robotem, 25
    - wykaz materiałów, 16
  - Spark WiFi Development Kit, 169
  - sprawdzenie
    - sieci, 26
    - silnika, 53

- stan
  - buta, 240
  - klawiszy, 63
  - robota, 212
- staw
  - biodrowy, 88
  - łokciowy, 35
- ster, 68
- sterowanie, 23
  - smartfonem, 78
  - diodą RGB, 162
  - heksapodem, 83
  - łodzią, 66
  - matrycami, 157
  - matrycą LED, 153
  - obwodem przekaźnika, 107
  - robotem, 210
  - robotem SimpleBot, 25
  - serwomotorami, 38
- sterowanie
  - serwomotorem, 67
  - silnikiem, 62
  - urządzeniami, 75
- sterownik silnika, 51, 58, 59
- sygnał o modulowanej szerokości, 33
- system
  - bezpieczeństwa, 201
  - monitoringu, 194
- szkic StandardFirmata.ino, 26
- sztuczna inteligencja, 203
- szyna I2C, 148, 150

## Ś

- śruba, 54
- światelka-strazydełka, 145
  - aplikacja WWW, 155
  - budowa, 147
  - diagnostyka, 153
  - interfejs użytkownika, 157
  - komponenty, 146

- program testujący, 154
- sterowanie matrycą LED, 153
- wykaz materiałów, 145

## T

- tabela
  - obiektów Servo.Array, 94
  - serwomotorów, 94
- tag <div>, 80
- tarcza, 132
- tarcze gnomonu, 133
- test modułu Spark, 56, 57
- tworzenie
  - interfejsu API, 156
  - interfejsu użytkownika, 157
  - plików projektu, 38
  - projektu Android, 247
  - serwera poleceń, 110
- TypeBot
  - budowa
    - części mechanicznej, 34
  - inicjalizacja, 41
  - pierwsze uruchomienie, 46
  - program, 38
  - wykaz materiałów, 31

## U

- układ współrzędnych, 90
- umieszczenie tarcz, 134
- urządzenie ultradźwiękowe, 193
- usługa CheerLights, 163, 164
- ustawianie
  - czujnika ultradźwiękowego, 210
- uszczelnienie przewodów, 55

## W

- wartość null, 97
- wiercenie otworów, 222

- wiosłowanie, 98
- wodowanie łodzi, 64
- wskaźnik stanu, 196
- wtyczka Spark-io, 170
- wycięcie gnomonu, 138
- wyliczanie położenia efektora, 227
- wysyłanie
  - poleceń, 80
  - SMS-ów, 193
- wyświetlacz, 173

## Z

- zakładanie łączówek, 152
- zakres ruchu serwomotorów, 92
- zapisywanie stanu klawiszy, 63
- zasilacz 5 V, 174
- zasilanie sterownika silnika, 59
- zasobnik z silnikiem, 51, 54, 55
- zatrask, 52
- zdalne sterowanie, 207
- zdarzenie
  - message, 76
  - naciśnięcia klawisza, 62
- zegar słoneczny, 125
  - budowanie, 128
  - kod, 139
  - konfiguracja serwomotorów, 130
  - konstrukcja główna, 129
  - montaż, 136
  - podparcie tarczy, 133
  - przygotowanie tarcz, 132
  - ścianki, 133
  - włączanie, 144
  - wycinanie elementów, 129
  - wykaz materiałów, 126
- zmiana
  - kolorów, 185
  - rezystancji czujnika, 233



# 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**

# Przekonaj się, jak fascynujące jest budowanie i programowanie robotów!

Powszechnie JavaScript jest uważany za doskonałe narzędzie do programowania aplikacji internetowych. Język ten świetnie nadaje się również do programowania robotów. Umożliwia zaprojektowanie zachowania robota, określenie, w jaki sposób będzie reagował na sygnały otaczającego świata czy omijał przeszkody. Dużą wygodę pracy zapewnia też platforma Johnny-Five, która umożliwi programowanie robotów zbudowanych z płytek Arduino, Raspberry Pi i BeagleBone.

Dzięki tej książce nauczysz się budować roboty o rozmaitych funkcjach i zastosowaniach. Do pisania tej książki Rick Waldron zaprosił zespół inżynierów, nauczycieli i popularyzatorów JS. Z ich pomocą dowiesz się, w jaki sposób zbudować poszczególne elementy robota, jakie części zastosować i jak napisać potrzebny kod. Podpowiedzą Ci, jak przygotować sobie warsztat pracy oraz skąd zdobyć materiały. Autorzy podali mnóstwo wskazówek pomocnych przy planowaniu, budowie i testowaniu już wykonanych projektów. Poszczególne zadania zostały tak dobrane, aby po ich wykonaniu można było zabrać się do trudniejszych projektów o wyższym stopniu złożoności.

## Zbuduj i oprogramuj:

- » roboty kroczące, piszące, pływające, a nawet tańczące!
- » robota sterowanego głosem oraz do sterowania oświetleniem
- » robota ze sztuczną inteligencją
- » robota delta

**Rick Waldron** — twórca Johnny-Five, platformy open source służącej do programowania robotów w języku JavaScript. Obecnie zajmuje się głównie Arduino, BeagleBone, Raspberry Pi, Pinoccio, Spark Core, Intel Galileo i Intel Edison. Jest również twórcą Idiomatic.js, przetłumaczonego na 12 języków przewodnika po JavaScript.

<b>Helion</b>			
43384	numer katalogowy	KOD KORZYŚCI	
	księgarnia internetowa	ISBN 978-83-283-2054-3	
<a href="http://helion.pl">http://helion.pl</a>			
	zamówienia telefoniczne	9 788328 320543	
	<b>0 801 339900</b>	cena: 59,00 zł	
	<b>0 601 339900</b>	<b>Make:</b> makezine.com	
Informatyka w najlepszym wydaniu			