

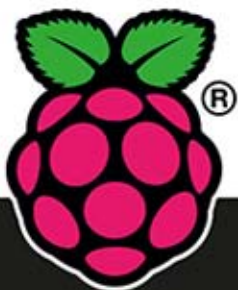
16 praktycznych
projektów programowych
i sprzętowych opartych
na Raspberry Pi



Raspberry Pi[®]

Najlepsze projekty

Wykorzystaj ogromny potencjał miniaturowego komputera!



Andrew Robinson

Mike Cook

WILEY



Tytuł oryginału: Raspberry Pi Project

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-246-9221-7

© 2014 John Wiley & Sons, Ltd.

All Rights Reserved. Authorised translation from the English language edition published by John Wiley & Sons Limited. Responsibility for the accuracy of the translation rests solely with Helion S.A. and is not the responsibility of John Wiley & Sons Limited.

No part of this book may be reproduced in any form without the written permission of the original copyright holder, John Wiley & Sons Limited.

Translation copyright © 2014 by Helion S.A.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book. This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Wiley and the John Wiley & Sons, Ltd. logo are trademarks or registered trademarks of John Wiley and Sons, Ltd. and/ or its affiliates in the United States and/or other countries, and may not be used without written permission. Raspberry Pi is a trademark of the Raspberry Pi Foundation. All other trademarks are the property of their respective owners. John Wiley & Sons, Ltd. is not associated with any product or vendor mentioned in the book.

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/rasppn>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

Wprowadzenie	13
Historia powstania	13
Informatyka użytkowa	14
Dlaczego wszyscy powinni uczyć się informatyki?	14
Poznaj Raspberry Pi	15
O tej książce	16
Jak korzystać z tej książki?	17
Przyszłość	19

Część I: Wprowadzenie do Raspberry Pi **21**

ROZDZIAŁ 1

Uruchamianie Raspberry Pi	23
System operacyjny	24
Podłączanie Raspberry Pi	30
Proces ładowania	38
Uruchamianie interfejsu graficznego	39
Uruchamianie terminala w środowisku X	39
Rozwiązywanie problemów	40
Pora rozpocząć zabawę!	42

ROZDZIAŁ 2

Wprowadzający projekt oprogramowania — generator obelg	43
Uruchamianie pierwszego programu w Pythonie	44
Zapisywanie programu	46
Generowanie obelg	48
Imienne obrazanie znajomych	51
Tworzenie strumienia obelg	53
Łączenie wszystkich elementów	56

Część II: Projekty oprogramowania **59**

ROZDZIAŁ 3

Kółko i krzyżyk	61
Błędy	62
Początek	63
Gra dla dwóch zawodników	67
Gra przeciw komputerowi	70
Twoja kolej	79

ROZDZIAŁ 4

Oto przegląd najnowszych wiadomości	81
Pierwsze telepromptery	82
Pi Prompter	83

Czego potrzebujesz?	83	
Krok bliżej do użytecznego programu	88	
Gotowy kod programu Pi Prompter	92	
Fizyczna budowa telepromptera	96	
Twoja kolej	99	
ROZDZIAŁ 5		
Ping	101	
Wczesne produkty komercyjne	102	
Gra Ping	103	
Ulepszanie gry Ping	107	
Gra dla jednej osoby	111	
Wersja dla dwóch graczy	117	
Twoja kolej	123	
ROZDZIAŁ 6		
Pie Man	125	
Pie Man	126	
Tworzenie zasobów	127	
Przygotowywanie planszy	131	
Przebieg gry	135	
Wyświetlanie zawartości ekranu	142	
Ostatnia funkcja	144	
Twoja kolej	149	
ROZDZIAŁ 7		
Generowanie labiryntów w Minecraftcie	151	
Instalowanie Minecrafta	152	
Uruchamianie Minecrafta	153	
Gra w Minecrafta	154	
Przygotowania do użycia Pythona	156	
Używanie modułu Minecrafta	156	
Twoja kolej	172	
Część III: Projekty sprzętowe		173
ROZDZIAŁ 8		
Kolorowa wersja gry Snap	175	
Implementowanie gry	176	
Teoria	176	
Oprogramowanie do testowania gry	189	
Oprogramowanie gry	192	
Twoja kolej	196	
ROZDZIAŁ 9		
Sprawdź swój czas reakcji	197	
Witaj w świecie systemów wbudowanych!	198	
Pozyskiwanie komponentów	199	
Konfigurowanie karty PiFace Digital	200	
Podłączanie karty PiFace Digital	204	

Używanie emulatora	204
Komunikowanie się z Pythonem	205
Stoper do pomiaru czasu reakcji	208
Co podłączysz do komputera?	218
ROZDZIAŁ 10	
Twittująca zabawka	219
„Hakowanie” zabawki	220
Jak zmusić zabawkę do mówienia?	224
Poruszanie zabawką	227
Łączenie się z Twitterem	230
Łączenie wszystkich elementów	235
Podsumowanie	238
ROZDZIAŁ 11	
Światła dyskotekowe	241
Definiowanie sekwencji świateł	242
Kod wykonujący inne operacje	244
Trochę teorii	246
Projektowanie sekwensera	247
Implementowanie sekwensera	248
Oświetlenie	254
Używanie dłuższych taśm LED	256
Ruszające się światła	258
Uruchamianie obwodu	262
Twoja kolej	262
ROZDZIAŁ 12	
Zamek do drzwi	263
Ogólne omówienie systemu	264
Systemy krytyczne ze względu na bezpieczeństwo	265
Zamek do drzwi	266
Wstępna wysokopoziomowa symulacja programowa	267
Blok wyjściowy	268
Blok do pobierania danych wejściowych	271
Blok uwierzytelniający użytkownika	271
Odblokowywanie drzwi bez ich dotknięcia	273
Testowanie programu i mocowanie zamka	278
Łączenie wielu drzwi w sieci	279
Twoja kolej	280
Sztuka programowania	281
ROZDZIAŁ 13	
Automatyzowanie domu	283
IOT	284
Projekt 1. Jak zbudować czujnik ruchu oraz wyłącznik drzwiowy?	284
Projekt 2. Monitorowanie domu za pomocą kamery internetowej	290
Projekt 3. Budowanie termometru	296
Projekt 4. Wysyłanie e-maili z powiadomieniami	300
Projekt 5. Wysyłanie e-maili za pomocą pilota bezprzewodowego	306
Twoja kolej	311

ROZDZIAŁ 14

Komputerowe sterowanie elektroniką toru z samochodzikami	313
Kupno toru z samochodzikami	314
Podłączanie toru z samochodzikami	314
Testowanie połączeń z torem	316
Pobieranie danych od graczy	316
Oprogramowanie	325
Kod gry	327
Twoja kolej	332

ROZDZIAŁ 15

Generowanie grafiki i zamieszczanie jej na Facebooku	333
Pomysł	334
Rodzaje enkoderów obrotowych	334
Dane wyjściowe z enkodera	335
Przesyłanie rysunków do Facebooka	342
Ostateczna wersja programu Roto-Sketch	346
Tworzenie symetrycznych wzorów	351
Twoja kolej	355

ROZDZIAŁ 16

Harmonograf	357
Pomysł	359
Efekt Halla	359
Poznaj Arduino	361
Łączenie różnych elementów	362
Programowanie Arduino	372
Programowanie Raspberry Pi	383
Używanie programu Pendulum Pi	388
Twoja kolej	388

ROZDZIAŁ 17

Zaawansowana budka dla ptaków — obserwowanie przyrody	391
Budowanie niewidocznych czujników promieni	393
Montowanie czujników	397
Zapisywanie aktywności ptaków w pliku	400
Przetwarzanie danych	410
Radzenie sobie z szumem z czujników	415
Rysowanie wykresu	420
Włączanie budki	422
Twoja kolej	423
Możliwości są nieograniczone	424
Skorowidz	425

Rozdział **7**

Generowanie labiryntów w Minecraftcie

Sean McManus

W tym rozdziale:

- Instalowanie Minecrafta
- Poznawanie świata Minecrafta
- Manipulowanie światem Minecrafta w Pythonie
- Generowanie losowego labiryntu Minecrafta za pomocą Pythona

Minecraft jest popularny wśród fanów klocków Lego z całego świata. Pozwala tworzyć z bloków wciągające trójwymiarowe światy i rozpala wyobraźnię do tego stopnia, że zgodnie z szacunkami sprzedano około 20 milionów egzemplarzy tej gry na różne platformy (w tym na komputery PC i konsole Xbox).

Dostępna jest także wersja alfa tej gry przeznaczona na Raspberry Pi. Dostępny jest w niej tylko tryb tworzenia, w którym można bezpiecznie budować świat bez zagrożenia głodem lub atakami potworów. Wersja ta ma też ciekawą cechę — można ją programować za pomocą różnych języków, w tym Pythona. Oznacza to, że da się zbudować wielki pałac bez konieczności ręcznego dodawania wszystkich bloków. Można też, co przedstawiam w tym rozdziale, pisać programy wymyślające oryginalne, nowe struktury, a następnie zwiedzać i eksplorować te obiekty.

W tym projekcie utworzysz w Pythonie program generujący labirynty w Minecraftcie. Przy każdym uruchomieniu program generuje nowy labirynt, a użytkownik może kontrolować, jak duża ma być struktura i z jakich materiałów należy ją zbudować. W trakcie prac nad projektem zobaczysz, jak za pomocą Pythona dodawać i usuwać bloki w Minecraftcie. Dzięki temu zdobędziesz umiejętności, które pozwolą Ci pisać własne programy sterujące budową.

W czasie gdy powstawała ta książka, gra Minecraft: Pi Edition była dostępna w wersji alfa. Jest to bardzo wczesna wersja testowa (jeszcze mniej zaawansowana niż wersja beta). Jednak wykryłem w niej tylko kilka problemów: okno i jego zawartość były źle wyrównane przy rozdzielczości 1024×768 (dlatego przełączyłem ją na 1280×1024), a kursor działał nieprawidłowo po zmaksymalizowaniu okna. Mimo to zachęcam do tego, aby przy korzystaniu z oprogramowania w wersji alfa na wypadek problemów zawsze archiwizować ważne dane z Raspberry Pi.

WSKAZÓWKA

Najłatwiejszy sposób archiwizowania plików z Raspberry Pi polega na podłączeniu urządzenia pamięci masowej i wykorzystaniu do skopiowania danych menedżera plików dostępnego w środowisku okienkowym.

Kod do tego rozdziału można pobrać z mojej witryny: <http://www.sean.co.uk>.

Instalowanie Minecrafta

Choć Minecraft na inne platformy jest oprogramowaniem komercyjnym, wersję alfa na Raspberry Pi można pobrać bezpłatnie. Aby móc uruchomić tę wersję, trzeba zainstalować w Raspberry Pi dystrybucję Raspbian Wheezy Linuksa. Jest to wersja systemu zalecana przez fundację Raspberry Pi. Można ją zainstalować za pomocą obrazu NOOBS dostępnego w sekcji z plikami do pobrania z witryny <http://www.raspberrypi.org>. Informacje na temat instalowania systemu operacyjnego znajdziesz w rozdziale 1., „Uruchamianie Raspberry Pi”.

Aby zainstalować Minecrafta, wykonaj następujące czynności:

1. Upewnij się, że Raspberry Pi jest podłączony do internetu. Minecrafta pobierzesz przez połączenie internetowe.
2. Uruchom Raspberry Pi, wpisz polecenie `startx` i wciśnij klawisz *Enter*, aby utworzyć środowisko okienkowe.
3. Kliknij dwukrotnie ikonę Midori w środowisku okienkowym lub użyj menu *Programs* w lewym dolnym rogu, aby uruchomić przeglądarkę internetową.
4. Otwórz stronę <http://pi.minecraft.net> i kliknij odnośnik, aby pobrać Minecrafta. W oknie dialogowym kliknij przycisk *Save As* i zapisz plik w katalogu *pi*. W przeglądarce plików katalog ten powinien być wyróżniony w widocznej po lewej stronie sekcji *Places*. Kliknij przycisk *Save* i zamknij przeglądarkę Midori (nie będziesz już jej potrzebował).
5. Kliknij dwukrotnie ikonę programu LXTerminal na pulpicie, aby uruchomić sesję terminala.
6. Wprowadź instrukcję `tar -zxvf minecraft-pi-0.1.1.tar.gz`, aby wypakować zawartość pobranego pliku. W Linuksie wielkość znaków ma znaczenie, dlatego koniecznie użyj samych małych liter. Ostatni człon polecenia to nazwa pobranego pliku. Może się ona zmienić (zwłaszcza końcowe numery), gdy pojawią się nowe wersje Minecrafta. Wystarczy wpisać kilka pierwszych liter nazwy i wcisnąć klawisz *Tab*, aby komputer automatycznie ją uzupełnił. W trakcie wypakowywania plików ich nazwy będą pojawiać się na ekranie. W celu wyświetlenia zawartości katalogu *pi* wpisz instrukcję `ls`.
7. Wprowadź polecenie `cd mcpi`, aby przejść do katalogu z wypakowanymi plikami Minecrafta.
8. Wpisz instrukcję `./minecraft-pi` i wciśnij klawisz *Enter*, aby uruchomić grę.

Jeśli wszystko przebiegło prawidłowo, powinieneś zobaczyć ekran tytułowy Minecrafta.

Uruchamianie Minecrafta

Gdy po zainstalowaniu Minecrafta zechcesz ponownie go uruchomić, przejdź do środowiska okienkowego (krok 2. w instrukcjach instalacji w podrozdziale „Instalowanie Minecrafta”), rozpocznij sesję terminala (krok 5.) i wpisz polecenie `cd mcpi`, a następnie `./minecraft-pi`. Nie można uruchomić Minecrafta z poziomu wiersza poleceń bez wcześniejszego otwarcia środowiska okienkowego.

Jeśli zamkniesz okno programu LXTerminal, natychmiast zakończysz sesję Minecrafta. Dlatego spróbuj ignorować to okno, które na pozór nic nie robi — jest ono potrzebne.

WSKAZÓWKA

Gra w Minecrafta

Gdy uruchomisz Minecrafta w Raspberry Pi, na ekranie powitalnym będziesz mógł wybrać jedną z dwóch opcji:

- *Start Game* (rozpoczęcie gry). Z tej opcji będziesz korzystał w tym rozdziale do generowania własnych światów, które można eksplorować. Za jej pomocą możesz też wybrać wcześniej wygenerowany świat, gdy ponownie uruchomisz Minecrafta. Aby wybrać jeden ze światów, kliknij je i przeciągnij, tak aby potrzebny świat znalazł się pośrodku. Następnie kliknij na nim, aby go otworzyć.
- *Join Game* (dołączanie do gry). Ta opcja pozwala dołączyć do innych graczy w sieci lokalnej. Omawianie tego trybu wykracza poza zakres rozdziału. Ta opcja umożliwia współpracę lub rywalizację w świecie Minecrafta.

Kliknij przycisk *Start Game*, a następnie wybierz opcję *Create New*. Minecraft wygeneruje wtedy nowy świat z wyjątkowym układem gór, lasów i oceanów. Po zakończeniu tego procesu zobaczysz świat z perspektywy pierwszej osoby (rysunek 7.1).

Rysunek 7.1.
Microsoft
w Raspberry Pi



WSKAZÓWKA

Możesz zmienić perspektywę i zobaczyć postać gracza. W tym celu wciśnij klawisz *Esc*, aby otworzyć menu gry, a następnie kliknij ikonę obok ikony głośników w lewym górnym rogu.

Po zakończeniu gry możesz ją zamknąć z poziomu menu gry (aby je wyświetlić, wciśnij klawisz *Esc*).

Poruszanie się po świecie

Przy grze w Minecrafta najlepiej jest używać obu rąk. Jedna powinna znajdować się na myszy, a druga — na klawiaturze. Za pomocą myszy możesz się rozglądać i zmieniać kierunek. Przesunięcie myszy w lewo lub prawo pozwala się obrócić, a ruchy do przodu i do tyłu powodują spoglądanie w dół i w górę. Do poruszania się służą klawisze *W* (do przodu), *S* (do tyłu), *A* (w lewo) i *D* (w prawo). Są one zgrupowane w jednym miejscu na klawiaturze, dzięki czemu posługiwanie się nimi jest łatwe.

Postać automatycznie skacze na niżej położone obszary, jeśli nad nie przejdiesz. Możesz też celowo wykonać skok (służy do tego spacja).

Aby uzyskać najlepszy obraz świata, kliknij dwukrotnie spację. Ujrzysz wtedy świat z lotu ptaka. W tym trybie wciśnięcie spacji pozwala wznieść się wyżej, a lewy klawisz *Shift* służy do obniżania pozycji. Ponowne dwukrotne kliknięcie spacji umożliwia powrót na ziemię. W tej wersji Minecrafta nie ma żyć ani zagrożeń, dlatego możesz bezpiecznie skakać z dowolnej wysokości.

Tworzenie i niszczenie obiektów

Jeśli chcesz zniszczyć blok, wskaż go myszą, a następnie kliknij i przytrzymaj lewy przycisk myszy. Niektóre bloki są łatwiejsze do usunięcia niż inne. Aby zniszczyć obiekt, musisz być odpowiednio blisko. Dlatego jeśli przy próbie usunięcia bloku nie widzisz, jak się rozpada, stań bliżej niego.

Panel w dolnej części okna przedstawia bloki, które możesz umieścić w świecie (zobacz rysunek 7.1). Wybierać bloki można za pomocą kółka przewijania myszy lub klawiszy z cyframi od 1 do 8 (blokowi pierwszemu od lewej odpowiada cyfra 1 itd.). Wciśnij *E*, aby otworzyć pełną listę materiałów. Do poruszania się po niej służą klawisze sterujące ruchem (*W*, *A*, *S*, *D*). Jeśli chcesz wybrać dany blok, wciśnij klawisz *Enter* lub kliknij go myszą.

Aby umieścić blok w danym miejscu, kliknij docelową lokalizację prawym przyciskiem myszy. Blok możesz umieścić na innym tylko wtedy, gdy widzisz górą część tego ostatniego. Dlatego przy tworzeniu wysokich budowli czasem trzeba spojrzeć na świat z lotu ptaka.

Możesz budować wieże i jednocześnie wspinać się po nich. W tym celu należy spoglądać w dół i jednocześnie podskakiwać oraz dodawać bloki.

WSKAZÓWKA

Choć budowanie obiektów za pomocą Pythona jest znacznie łatwiejsze, zachęcam do zapoznania się z tym, jak gracze czują się w świecie Minecrafta. Warto zwłaszcza poeksperymentować z interakcją między blokami. Bloki kamienne potrafią utrzymać się bez bezpośredniego podparcia w powietrzu, natomiast bloki z piasku spadają na powierzchnię. Nie można sadzić kaktusów na trawie, można jednak

umieścić je na piasku. Jeśli usuniesz blok na brzegu jeziora, opróżnioną przestrzeń zapełni woda. W grze nie można umieszczać źródeł wody ani lawy, choć można dodać je programowo za pomocą Pythona. Woda i lawa spływają kaskadami w dół i zapełniają duże obszary. Gdy te żywioły zetkną się ze sobą, woda schładza lawę i zamienia ją w kamień.

Przygotowania do użycia Pythona

Jedną z zaskakujących cech Minecrafta jest to, że gra przejmuje kontrolę nad myszą. Dlatego trzeba wcisnąć klawisz *Tab*, aby wrócić do używania innych programów. Jeśli później chcesz ponownie zacząć używać myszy w Minecraftcie, kliknij okno gry. Wkrótce przyzwyczaisz się do wciskania klawisza *Tab* przed rozpoczęciem programowania. Teraz wciśnij ten klawisz, aby wyjść z Minecrafta, i przenieś kursor na pulpit.

Do tworzenia programów związanych z Minecraftem posłużysz środowisko IDLE. Kliknij dwukrotnie jego ikonę na pulpicie, aby je uruchomić. Możliwe, że najpierw będziesz musiał kliknąć górną krawędź okna Minecrafta i przeciągnąć je, żeby zobaczyć potrzebną ikonę.

Jedną z pierwszych rzeczy, jakie zauważysz, jest to, że Minecraft znajduje się nad innymi oknami, dlatego środowisko IDLE może być niewidoczne. Niezbędna jest więc odpowiednia reorganizacja pulpitu. Aby przenieść okno, przeciągnij je za pasek tytułu. Zmianianie wielkości okien odbywa się w wyniku przeciągania krawędzi i rogów. Zachęcam, aby tak uporządkować okna, aby wszystkie były jednocześnie widoczne. Na monitorze o standardowych wymiarach, jakiego używam, umieszczam Minecrafta w lewym górnym rogu, małe okno z powłoką Pythona w prawym górnym narożniku, a okno z kodem programu w dolnej części ekranu. Nie należy zmieniać wielkości okna Minecrafta. W wersji, z której korzystam, gra przestaje wtedy reagować na ruchy myszą. Okno programu LXTerminal możesz zignorować (ale go nie zamykaj).

Używanie modułu Minecrafta

Teraz możesz napisać swój pierwszy program w Pythonie powiązany ze światem Minecrafta. Program ten będzie wysyłał komunikaty do dostępnego w grze czata.

W powłoce Pythona otwórz menu *File* i wybierz opcję *New*, aby otworzyć nowe okno edytora. Wpisz w nim poniższy kod, a następnie zapisz plik w katalogu *pi* za pomocą menu *File* i wciśnij klawisz *F5*, aby uruchomić nowy program. Do działania wymaga on, aby uruchomiona była sesja gry w Minecraftcie.

```
import sys, random
sys.path.append("./mcp/api/python/mcpi")
import minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Witamy w labiryntach Minecrafta!")
```

Pierwszy wiersz importuje moduły `sys` i `random`. Moduł `random` będzie później potrzebny do budowania losowych labiryntów. Moduł `sys` jest używany od razu do poinformowania środowiska IDLE, gdzie znajduje się moduł Pythona związany z Minecraftem (pozwala on na przekazywanie instrukcji wykonywanych w Minecraftcie). Po wskazaniu środowisku IDLE lokalizacji tego modułu należy go zaimportować.

W Pythonie do przesyłania instrukcji do Minecrafta służy polecenie `minecraft`. `Minecraft.create()`, po którym należy podać potrzebną instrukcję. Na przykład aby wyświetlić powitanie w oknie czata, zastosuj poniższy kod:

```
minecraft.Minecraft.create().postToChat("Witamy w labiryntach Minecrafta!")
```

Przy dłuższych instrukcjach taki kod jest nieczytelny, dlatego w przedstawionym programie utworzyłem zmienną `mc`, którą można stosować jako skrót polecenia `minecraft.Minecraft.create()`. Dzięki temu można przysyłać komunikaty za pomocą krótszych wierszy kodu, tak jak w przykładowym programie.

Jeśli kod nie działa, zwróć uwagę na wielkość znaków. W Pythonie ma ona znaczenie, dlatego musisz stosować małe i wielkie litery dokładnie w taki sposób jak w przykładowym kodzie. Zwróć uwagę na notację wielbłądzia w nazwie `postToChat` i wielką literę `M` w poleceniu `minecraft.Minecraft.create()`.

WSKAZÓWKA

Współrzędne w Minecraftcie

Jak łatwo się domyślić, każdy punkt w świecie Minecrafta ma współrzędne. Do określenia pozycji punktu potrzebne są współrzędne w trzech wymiarach:

- Oś `x`. Jest równoległa do powierzchni i przyjmuje wartości od $-127,7$ do $127,7$.
- Oś `y`. Jest ustawiona pionowo i określa wysokość. Możesz wznieść się aż na wysokość 500 jednostek, jednak już przy wysokości 70 ziemia nie jest widoczna, dlatego dalsze wznoszenie się nie ma sensu. Poziom morza to 0. Możesz rozbijać bloki, aby drażyć tunele pod wodą. Mnie udało się zejść do poziomu -70 jednostek, zanim wypadłem ze świata i zginąłem. Jest to jedyny możliwy sposób zabicia postaci w Minecraftcie na Raspberry Pi, jaki udało mi się znaleźć.
- Oś `z`. Także jest równoległa do powierzchni i przyjmuje wartości od $-127,7$ do $127,7$.

Celowo podałem współrzędne w tej właśnie kolejności, ponieważ tak są podawane w Minecraftcie. Jeśli — podobnie jak ja — często używasz współrzędnych `x` i `y` do określania pozycji w świecie dwuwymiarowym (na przykład punktów na ekranie), będziesz musiał przyzwyczaić się do tego, że `y` reprezentuje wysokość. W tym rozdziale zwykle używam współrzędnych `x` i `z` do określania pozycji ścian (zmienia się ona w zależności od ściany), a współrzędnej `y` — do opisywania jej wysokości (jest ona niezależna od lokalizacji ściany).

W trakcie poruszania się po planszy w lewym górnym rogu okna Minecrafta zmieniają się współrzędne gracza. Jeśli spróbujesz wyjść poza świat gry, natrafisz na niewidzialną ścianę, której nie da się przebić — podobnie jak w filmie *Truman Show*, gdzie jednak bohater znalazł drzwi.

Zmienianie pozycji gracza

Za pomocą poniższego polecenia można przenieść postać w dowolne miejsce w świecie Minecrafta:

```
mc.player.setTilePos(x, y, z)
```

Aby na przykład zrzucić postać z nieba w środek świata, użyj następującego wywołania:

```
mc.player.setTilePos(0, 100, 0)
```

WSKAZÓWKA

Nie musisz umieszczać tego polecenia w aplikacji i uruchamiać jej. Jeśli już uruchomiłeś program w celu włączenia modułu Minecrafta, możesz używać powłoki Pythona do wprowadzania instrukcji służących do przenoszenia postaci i dodawania bloków.

Jeśli nie jest włączony tryb latania, postać spadnie z nieba w środku świata. Jeżeli tryb latania jest aktywny, kliknij okno Minecrafta i dwukrotnie wciśnij spację, aby wyłączyć ten tryb i rozpocząć opadanie.

Postać można umieścić w dowolnym miejscu świata gry. Czasem oznacza to, że postać pojawi się w środku góry lub innej struktury, w której ruch jest niemożliwy. W takiej sytuacji zmień pozycję gracza za pomocą kodu. Dobrym rozwiązaniem jest zwykle umieszczenie postaci na dużej wysokości, ponieważ można z niej spaść na najwyżej położony punkt.

Dodawanie bloków

Aby dodać blok do świata, zastosuj poniższe polecenie:

```
mc.setBlock(x, y, z, blockTypeId)
```

Parametr `blockTypeId` to liczba reprezentująca materiał, z jakiego zbudowany jest dodawany blok. Pełną listę materiałów znajdziesz na stronie [http://www.minecraft-wiki.net/wiki/Data_values_\(Pocket_Edition\)](http://www.minecraft-wiki.net/wiki/Data_values_(Pocket_Edition)). Ważne są liczby z kolumny *Dec* z tabeli z tej strony, ponieważ potrzebujesz liczby dziesiętnej, a nie szesnastkowej. Poprawne są wartości z przedziału od 0 do 108, a także kilka większych liczb. W tabeli 7.1 opisuję wybrane materiały, które prawdopodobnie okażą się najbardziej przydatne w tym projekcie oraz w trakcie wykonywania eksperymentów.

Tabela 7.1. Materiały z gry Minecraft: Pi Edition

blockTypeId	Typ bloku
0	Powietrze
1	Kamień
2	Trawa
3	Błoto
5	Deska
8	Woda
10	Lawa
12	Piasek
20	Szklana cegła
24	Piaskowiec
41	Złota cegła
45	Cegła
47	Półka na książki
53	Drewniane schody
57	Diamentowy blok
64	Drewniane drzwi
81	Kaktus

Gdy używasz bloków wody lub lawy, możesz wywołać powódź. Dlatego na potrzeby eksperymentów utwórz nowy świat.

WSKAZÓWKA

Dostępne jest też inne polecenie, które pozwala tworzyć duże prostopadłościany z bloków określonego typu. Aby zastosować to polecenie, podaj współrzędne dwóch przeciwległych narożników oraz materiał, z którego bryła ma być zbudowana:

```
mc.setBlocks(x1, y1, z1, x2, y2, z2, blockTypeId)
```

Aby szybko zbudować ceglany schron, utwórz duży prostopadłościan z cegły, a następnie umieść w nim prostopadłościan z powietrza. Powietrze zastępuje dowolne inne bloki, co powoduje ich usunięcie ze świata. Oto przykład:

```
mc.setBlocks(0, 0, 0, 10, 5, 7, 45) # Cegła
mc.setBlocks(1, 0, 1, 9, 5, 6, 0) # Powietrze
```

Ten kod w punkcie o współrzędnych (0, 0, 0) tworzy schron o powierzchni 10×7 bloków i wysokości 5 bloków. Ściany mają grubość jednego bloku, ponieważ obszar od pierwszego do dziewiątego bloku na osi x, od pierwszego do szóstego bloku na osi z i od zerowego do piątego bloku na osi y są wypełniane powietrzem. Po wszystkich stronach pozostaje więc jedna ściana o grubości jednego bloku, a góra budowli jest odkryta.

WSKAZÓWKA

Pamiętaj, że symbol # reprezentuje komentarz dla programisty. Komputer ignoruje zawartość wiersza znajdującą się po tym symbolu.

Choć przy określaniu pozycji postaci można posługiwać się współrzędnymi z częściami ułamkowymi (na przykład 1,7), to przy dodawaniu bloków współrzędne są zaokrąglane do najbliższej liczby całkowitej.

Uniemożliwianie graczom modyfikowania świata

Wiem, że nigdy byś nie oszukiwał w grze, jednak szukanie drogi w labiryncie, w którym można *przypadkowo* wyrąbać sobie przejście, nie byłoby ciekawe, prawda? Aby uniemożliwić graczom usuwanie lub dodawanie bloków w świecie, zastosuj następujący wiersz:

```
mc.setting("world.immutable",True)
```

Słowo *immutable* jest często stosowane w kontekście programowania i oznacza „niezmienny”.

Ustawianie parametrów labiryntu

Skoro wiesz już, jak dodawać bloki i używać bloków z powietrzem do opróżniania przestrzeni, możesz przystąpić do pisania programu generującego labirynty. W tym programie wykorzystasz zestaw stałych przeznaczonych na ważne informacje o labiryncie. Stałe to rodzaj zmiennych, których wartości *zdecydowałeś się* nie zmieniać w trakcie działania programu. Nazwy stałych zwykle zapisuje się wielkimi literami, aby zasignalizować ich przeznaczenie dla osób czytających kod programu i ułatwić samemu sobie pamiętanie o tym, że program nie powinien zmieniać wartości tych elementów. Zastąpienie w programie liczb stałymi pozwala w przyszłości łatwiej wprowadzać zmiany, a także znacznie poprawia czytelność kodu i pomaga zrozumieć, co reprezentują poszczególne wartości.

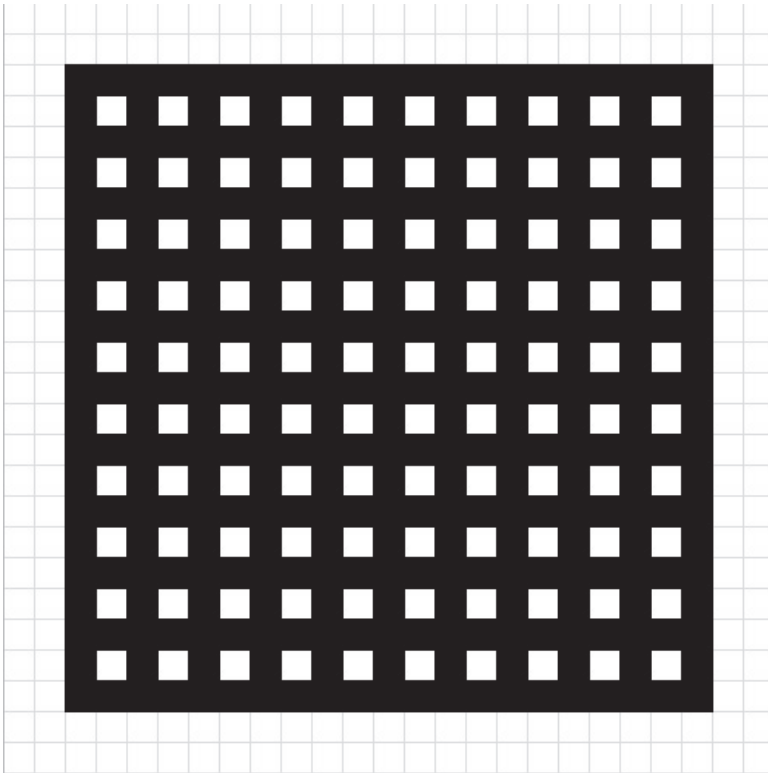
WSKAZÓWKA

W nazwach zmiennych wielkość liter ma znaczenie, dlatego dla Pythona `SIZE` i `size` to dwie różne zmienne (jednak stosowanie obu tych nazw w jednym programie nie jest najlepszym pomysłem).

Program najpierw ustawia stałe:

```
SIZE = 10
HEIGHT = 2
MAZE_X = 0
GROUND = 0
MAZE_Z = 0
MAZE_MATERIAL = 1 # Kamień
GROUND_MATERIAL = 2 # Trawa
CEILING = False
```

Budowanie labiryntu zaczniesz od siatki ścian z jednoblokowymi przestrzeniami (komórkami) między nimi. Efekt wyglądem przypomina gofry (rysunek 7.2). Każda komórka ma początkowo cztery ściany, a program usuwa je, dzięki czemu powstają ścieżki między komórkami, a ostatecznie — labirynt. Tu jest on kwadratowy, a parametr SIZE określa jego wielkość w komórkach. Labirynt z parametrem SIZE równym 10 ma po 10 komórek w wymiarach x i z, jednak w świecie Minecrafta zajmuje dwukrotnie więcej miejsca (20 na 20 bloków), ponieważ między komórkami znajdują się jednoblokowe ściany. Stanie się to zrozumiałe, gdy zaczniesz budować labirynt. Próbowałem tworzyć labirynty o parametrze SIZE równym 40, jednak ich budowanie zajmuje wiele czasu, a eksploracja — całe wieki. Na razie wystarczy wartość 10. Jeśli w świecie nie ma wystarczająco dużo miejsca na labirynt, program zakończy pracę i zgłosi błąd.



Rysunek 7.2.
Początkowa
siatka

Parametr HEIGHT określa w blokach wysokość ścian labiryntu. Wybrałem wartość 2, ponieważ przy wartości 1 można przeskakiwać ściany labiryntu (gracz automatycznie wchodzi na bloki o tej wysokości). Wyższe wartości powodują, że nie można zobaczyć widocznych w oddali gór, które są dla użytkownika cenną wskazówką wizualną.

Stałe MAZE_X, GROUND i MAZE_Z określają współrzędne początku labiryntu. Materiał, z którego zbudowany jest labirynt (MAZE_MATERIAL), to kamień (1), a materiałem podłoża (GROUND_MATERIAL) jest trawa (2). Dostępna jest też opcja, która pozwala dodać sufit, co uniemożliwia graczowi wzniesienie się ponad labirynt. Na razie jednak opcja ta jest wyłączona, co pozwala swobodnie eksplorować labirynt w trakcie jego tworzenia.

WSKAZÓWKA

Doskonale wygląda labirynt z półek z książkami (MAZE_MATERIAL=47).

Przygotowywanie podłoża

Jedną z pierwszych rzeczy, jakie trzeba zrobić, jest upewnienie się, że labirynt znajduje się na równym stałym łądzie. Ponieważ świat Minecrafta jest generowany dynamicznie, może się okazać, że stawiasz budowlę w środku góry lub na morzu.

Oprócz powierzchni przeznaczonej na labirynt należy oczyścić obszar 10 bloków wokół budowli, aby gracz mógł wygodnie się do niej zbliżyć i obejść ją dookoła. Najpierw opróżnij obszar przez wypełnienie go blokami powietrza. Spowoduje to usunięcie wszystkich innych obiektów z danego miejsca.

Labirynt zajmuje mierzony w blokach obszar od MAZE_X do MAZE_X+(SIZE*2) i od MAZE_Z do MAZE_Z+(SIZE*2). Liczba bloków jest dwa razy większa od liczby komórek (SIZE), ponieważ na prawo i poniżej każdej komórki znajduje się ściana. Środkowy punkt labiryntu w świecie Minecrafta ma współrzędne MAZE_X+SIZE, MAZE_Z+SIZE.

Trzeba też opróżnić obszar 10 bloków w każdym kierunku od labiryntu. Poniższy kod usuwa wszystkie bloki na wysokość 150 bloków od poziomu labiryntu. Dzięki temu nie istnieje ryzyko, że nieusunięte bloki góry spadną z nieba do labiryntu i będą leżeć w korytarzach:

```
mc.setBlocks(MAZE_X-10, GROUND, MAZE_Z-10, MAZE_X+(SIZE*2)+10, GROUND+150,
↳MAZE_Z+(SIZE*2)+10, 0)
mc.setBlocks(MAZE_X-10, GROUND, MAZE_Z-10, MAZE_X+(SIZE*2)+10, GROUND,
↳MAZE_Z+(SIZE*2)+10, GROUND_MATERIAL)
```

Zalecam, aby dodać blok wyznaczający początkowy róg labiryntu (współrzędne MAZE_X, MAZE_Z). Jest to przydatne przy pisaniu i debugowaniu programu, ponieważ pozwala określić położenie labiryntu przy obserwowaniu go z lotu ptaka. Blok można dodać za pomocą poniższego kodu:

```
mc.setBlock(MAZE_X, GROUND+HEIGHT+1, MAZE_Z, MAZE_MATERIAL)
```

Umieść postać nad środkiem labiryntu, abyś mógł patrzeć w dół i obserwować jego powstawanie. Jeśli nie masz włączonego trybu latania, spadniesz na ścianę labiryntu, jednak zawsze możesz ponownie uruchomić ten tryb.

```
mc.player.setTilePos(MAZE_X+SIZE, GROUND+25, MAZE_Z+SIZE)
```

Dodawanie ścian labiryntu

Aby utworzyć przypominającą gofra siatkę, zastosuj poniższy kod:

```
for line in range(0, (SIZE+1)*2, 2):
    mc.setBlocks(MAZE_X+line, GROUND+1, MAZE_Z, MAZE_X+line, GROUND+HEIGHT,
↳MAZE_Z+(SIZE*2), MAZE_MATERIAL)
    mc.setBlocks(MAZE_X, GROUND+1, MAZE_Z+line, MAZE_X+(SIZE*2), GROUND+HEIGHT,
↳MAZE_Z+line, MAZE_MATERIAL)
```

Pętla for przypisuje do zmiennej line liczby parzyste od 0 do SIZE*2. Przed podwojeniem wartości trzeba dodać do SIZE liczbę 1, ponieważ funkcja range nie zwraca ostatniej wartości w sekwencji (na przykład wywołanie range(1, 10) zwraca liczby od 1 do 9). Liczba 2 na końcu wywołania range oznacza wielkość kroku, tak więc pętla w każdym powtórzeniu przyjmuje kolejne wartości parzyste. W efekcie między ścianami pozostają puste miejsca (komórki). Przy każdym powtórzeniu pętli używany jest prostopadłościan do narysowania dwóch ścian, które biegną przez całą szerokość labiryntu wzdłuż osi x i z. To, że w miejscach przecięcia się ścian blok jest dodawany dwukrotnie, nie ma znaczenia. Ściana jest budowana od poziomu GROUND+1, dlatego gdy usuniesz ściany w celu utworzenia ścieżek, nadal będzie widoczna trawa.

Nie zapomnij o dwukropku po instrukcji for. Ponadto dwa następne wiersze muszą być wyróżnione wcięciem (jest to dla Pythona informacja, że należą do pętli).

WSKAZÓWKA

Powinieneś uzyskać siatkę wyglądającą tak, jak na rysunku 7.3.



Rysunek 7.3.
Siatka
w Minecraftie

Algorytm generowania labiryntu

Przed rozpoczęciem analizowania kodu, który przekształca gofra w labirynt, wyjaśniam, jak ten kod działa. Celem jest utworzenie labiryntu doskonałego (w sensie technicznym — nie są to moje przechwałki). Oznacza to, że w labiryncie ma nie być żadnych pętli ani niedostępnych miejsc. Między dowolnymi dwoma punktami istnieje tylko jedna ścieżka.

Program działa w następujący sposób:

1. Punktem wyjścia jest utworzony gofr, w którym każda komórka jest otoczona z czterech stron ścianami.
2. Należy losowo wybrać początkową komórkę labiryntu.
3. Trzeba zbadać wszystkie sąsiednie komórki i utworzyć listę tych, które mają wszystkie cztery ściany. Są to komórki, które nie zostały jeszcze odwiedzone.
4. Jeśli istnieją nieodwiedzone sąsiednie komórki, należy losowo wybrać jedną z nich, usunąć ścianę między bieżącą i wybraną komórką, a następnie przejść do tej ostatniej. W tym momencie to ona staje się bieżącą komórką.
5. Jeżeli wszystkie komórki sąsiadujące z bieżącą zostały już odwiedzone, należy cofnąć się o jedną komórkę i ustawić ją jako bieżącą.
6. Trzeba powtarzać kroki od 3. do 5. do momentu odwiedzenia wszystkich komórek.

Konfigurowanie zmiennych i list

Aby zaimplementować ten algorytm, należy zastosować następujące zmienne:

- Zmienna `numberOfCells`. Przechowuje łączną liczbę komórek labiryntu (czyli `SIZE*SIZE`).
- Zmienna `numberOfVisitedCells`. Przechowuje liczbę komórek odwiedzonych do tej pory. Jeśli ta liczba jest identyczna z wartością zmiennej `numberOfCells`, oznacza to, że algorytm odwiedził każdą komórkę i usunął jedną z jej ścian. Można więc dotrzeć do każdej komórki i labirynt jest ukończony.
- Zmienna `xposition`. Zapamiętuje pozycję na wymiarze „x” w czasie poruszania się po labiryncie i generowania go. Pozycja jest mierzona w komórkach i początkowo przyjmuje losową wartość z przedziału od 1 do `SIZE`.
- Zmienna `zposition`. Zapamiętuje pozycję na wymiarze „z” w czasie poruszania się po labiryncie i generowania go. Pozycja jest mierzona w komórkach i początkowo przyjmuje wartość losową.
- Lista `cellsVisitedList[]`. Przechowuje ścieżkę, dzięki czemu program może się cofać. Przy tworzeniu listy należy zapisać w niej początkową pozycję za pomocą metody `append()`.
- Zmienne `playerx` i `playerz`. Służą do zapamiętywania początkowej pozycji, dzięki czemu można umieścić w niej gracza po zbudowaniu labiryntu.

W kodzie algorytmów tego rodzaju (jest to **algorytm generowania labiryntów metodą DSF**) często potrzebna jest lista lub podobna struktura danych do przechowywania lokalizacji ścian. Tutaj jest to niepotrzebne, ponieważ w Minecraftie istnieją widoczne ściany. Jeśli chcesz, możesz zapisać labirynt w świecie gry.

Poniższy kod ustawia początkowe wartości zmiennych:

```
numberOfCells = SIZE*SIZE
numberOfVisitedCells = 1
cellsVisitedList = []
xposition = random.randint(1, SIZE)
zposition = random.randint(1, SIZE)
playerx = xposition
playerz = zposition
cellsVisitedList.append((xposition, zposition))
```

Tworzenie funkcji

W programie potrzebnych jest kilka podstawowych funkcji. Oto one:

- Funkcje `realx(x)` i `realz(z)`. Przekształcają współrzędne labiryntu (podawane w komórkach) na współrzędne świata Minecrafta (mierzone w blokach i przesunięte względem początkowej pozycji labiryntu).
- Funkcje `showMaker(x,z)` i `hideMaker(x,z)`. Dodają i ukrywają złoty blok pokazujący, do której komórki program doszedł w trakcie budowania labiryntu. Ciekawe jest obserwowanie zmian tych komórek z lotu ptaka, a ponadto funkcje te są przydatne w trakcie tworzenia i debugowania programu.
- Funkcja `demolish(realx,realz)`. Służy do usuwania ścian w labiryncie. Jako parametry przyjmuje rzeczywiste współrzędne ze świata Minecrafta.
- Funkcja `testAllWalls(cellx, cellz)`. Sprawdza, czy wszystkie cztery ściany komórki pozostały nietknięte. Jeśli tak jest, zwraca wartość `True` (w przeciwnym razie zwraca `False`). W tej funkcji używane jest polecenie `mc.getBlock(x, y, z)`, które określa typ bloku (`blockTypeId`) stojącego w danym miejscu. W standardowy sposób (za pomocą dwóch znaków równości) należy sprawdzić, czy ten typ jest taki sam jak typ materiału ścian (`MAZE_MATERIAL`). Jeśli tak jest, oznacza to, że w danym miejscu stoi ściana.

Dodaj poniższe definicje funkcji w początkowej części programu, po instrukcji ustawiającej moduł Minecrafta:

```
def realx(x):
    return MAZE_X+(x*2)-1

def realz(z):
    return MAZE_Z+(z*2)-1

def showMaker(x, z):
    mc.setBlock(realx(x), GROUND+1, realz(z), 41) # 41=złoto
```

```

def hideMaker(x, z):
    mc.setBlock(realx(x), GROUND+1, realz(z), 0)

def demolish(realx, realz):
    mc.setBlocks(realx, GROUND+1, realz, realx, HEIGHT+GROUND, realz, 0)

def testAllWalls(cellx, cellz):
    if mc.getBlock(realx(cellx)+1, GROUND+1, realz(cellz))==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx)-1, GROUND+1, realz(cellz))==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx), GROUND+1, realz(cellz)+1)==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx), GROUND+1, realz(cellz)-1)==MAZE_MATERIAL:
        return True
    else:
        return False

```

WSKAZÓWKA Jeśli wystąpił błąd, sprawdź, czy nie brakuje dwukropków po instrukcjach `def` i `if`.

Tworzenie głównej pętli

Algorytm tworzący labirynt działa do momentu odwiedzenia wszystkich komórek. Dlatego rozpoczyna się od poniższej instrukcji:

```
while numberOfVisitedCells < numberOfCells:
```

Trzeba sprawdzić, czy ściany w komórkach sąsiadujących z bieżącą pozostają nietknięte. W tym celu należy zbadać po kolei każdy kierunek za pomocą funkcji `testAllWalls(x, z)`. Po znalezieniu komórki z wszystkimi ścianami należy za pomocą metody `append()` dodać jej kierunek do listy `possibleDirections[]`. Tak wygląda implementacja trzeciego kroku algorytmu (pamiętaj, że cały poniższy kod jest wcięty względem nadrzędnej instrukcji `while`):

```

possibleDirections = []

if testAllWalls(xposition-1, zposition):
    possibleDirections.append("left")

if testAllWalls(xposition+1, zposition):
    possibleDirections.append("right")

if testAllWalls(xposition, zposition-1):
    possibleDirections.append("up")

if testAllWalls(xposition, zposition+1):
    possibleDirections.append("down")

```

Określenia `up`, `down`, `left` i `right` (czyli góra, dół, lewa, prawa) są w trójwymiarowym świecie dość nieprecyzyjne. Zastosowałem je, ponieważ są łatwe do zrozumienia. Jeśli w trakcie generowania labiryntu spojrzysz na niego z lotu ptaka i ustawisz się tak, aby początkowy narożnik labiryntu (o współrzędnych `MAZE_X`, `MAZE_Z`) znajdował się w lewym górnym rogu, wymienione wyżej określenia będą wskazywały właściwe kierunki.

Może zauważyłeś, że kod nie sprawdza, czy komórki znajdują się na krawędziach labiryntu. Co się stanie przy sprawdzaniu komórek, które wychodzą poza lewą lub dolną krawędź labiryntu? Nie stanowi to problemu. Kod programu automatycznie zatrzymuje się przed krawędziami. Ponieważ sprawdzane „komórki” poza krawędzią nie mają wszystkich czterech ścian (ich jedyną ścianą jest bok labiryntu), tak więc algorytm nigdy do nich nie przechodzi.

W kroku czwartym algorytm losowo wybiera jedną z nieodwiedzonych sąsiadujących komórek. Usuwa ścianę między bieżącą i wybraną komórką, po czym przechodzi do tej ostatniej. Aby ustalić, czy istnieje choć jeden nieodwiedzony sąsiad, należy sprawdzić długość listy `possibleDirections`. Jeśli jest ona różna od 0 (!=0), należy wykonać odpowiednie operacje. Cały omawiany fragment powinien być wyróżniony wcięciem względem nadrzędnej pętli `while`. Jeśli masz trudności z ustaleniem właściwego poziomu wcięcia, zapoznaj się z pełnym kodem z listingu 7.1 zamieszczonego w końcowej części rozdziału.

Przed zmianą pozycji należy ukryć złotą cegłę wskazującą pozycję w labiryncie:

```
hideMaker(xposition, zposition)
if len(possibleDirections)!=0:
    directionChosen=random.choice(possibleDirections)

    if directionChosen=="left":
        demolish(realx(xposition)-1, realz(zposition))
        xposition -= 1

    if directionChosen=="right":
        demolish(realx(xposition)+1, realz(zposition))
        xposition += 1

    if directionChosen=="up":
        demolish(realx(xposition), realz(zposition)-1)
        zposition -= 1

    if directionChosen=="down":
        demolish(realx(xposition), realz(zposition)+1)
        zposition += 1
```

Po przejściu do nowej komórki trzeba zwiększyć liczbę odwiedzonych komórek o jeden i dodać nową komórkę do listy przechowującej ścieżkę. Jest to także dobry moment na wyświetlenie w komórce złotego bloku pokazującego proces budowania labiryntu:

```
numberOfVisitedCells += 1
cellsVisitedList.append((xposition, zposition))
showMaker(xposition, zposition)
```

Warto wyjaśnić sposób przechowywania listy odwiedzonych komórek. Zmienne `xposition` i `zposition` znajdują się w nawiasach oznaczających krotkę. **Krotka** to sekwencja danych. Przypomina ona nieco listę, jednak różni się tym, że nie umożliwia modyfikowania wartości (jest niezmienna). Tak więc `cellsVisitedList` to lista

z krotkami, które zawierają pary współrzędnych x i z . Do sprawdzenia zawartości listy można wykorzystać powłokę Pythona. Oto przykład wzięty z jednego z przebiegów programu. Widać tu ścieżkę przejścia przez labirynt:

```
>>> print cellsVisitedList
[(6, 6.), (6, 7), (6, 8), (5, 8), (4, 8), (3, 8), (3, 7)]
```

Jeśli dana komórka nie ma nieodwiedzonych sąsiadów, w piątym kroku algorytmu należy wrócić do wcześniejszej pozycji na ścieżce. Wymaga to pobrania ostatniego elementu z listy ze ścieżką. Zadanie to można wykonać za pomocą metody `pop()` przeznaczonej dla list. Metoda ta pobiera ostatni element z listy i usuwa go z niej. W programie element ten jest przypisywany do zmiennej `retrace`, która zapisuje krotkę ze współrzędnymi x i z pozycji w labiryncie. Do wskazywania poszczególnych wartości krotki (podobnie jak w przypadku listy) służy indeks. Indeksowanie rozpoczyna się od zera, dlatego wywołanie `retrace[0]` zwraca współrzędną x poprzedniej komórki, a wywołanie `retrace[1]` zwraca współrzędną z . Oto potrzebny kod (obejmuje też wiersz wyświetlający złoty blok w poprzedniej komórce):

```
else: # Ten kod należy wywołać, jeśli nie istnieją nieodwiedzone sąsiednie komórki
    retrace = cellsVisitedList.pop()
    xposition = retrace[0]
    zposition = retrace[1]
    showMaker(xposition, zposition)
```

Warto zauważyć, że ta instrukcja `else` powinna być wyrównana względem powiązanej instrukcji `if` (sprawdzającej, czy można przejść dalej w jednym z kierunków).

Krok szósty algorytmu został już zaimplementowany, ponieważ pętla `while` po odwiedzeniu każdej komórki powtarza wyróżniony wcięciem kod.

Dodawanie sufitu

Osobiście uważam, że więcej zabawy daje rezygnacja z sufitu. Dzięki temu można się wznieść, podziwiać labirynt i spaść w jego dowolne miejsce. Jeśli jednak chcesz zbudować opartą na labiryncie grę i uniemożliwić użytkownikom oszukiwanie, dodaj sufit za pomocą poniższego kodu. Należy też zmienić wartość zmiennej `CEILING` w początkowej części programu na `True`. Tu sufit budowany jest ze szklanych bloków, dzięki czemu w korytarzach nie jest ciemno:

```
if CEILING == True:
    mc.setBlocks(MAZE_X, GROUND+HEIGHT+1, MAZE_Z, MAZE_X+(SIZE*2),
                ↪GROUND+HEIGHT+1, MAZE_Z+(SIZE*2), 20)
```

Określanie pozycji gracza

W ostatnim kroku należy umieścić gracza w losowym miejscu, od którego program rozpoczął generowanie labiryntu. Możesz umieścić postać w dowolnym miejscu, jednak to jest równie dobre jak każde inne i pozwala wykorzystać wygenerowane wcześniej losowe współrzędne:

```
mc.player.setTilePos(realx(playerx), GROUND+1, realz(playerz))
```

Teraz możesz rozpocząć grę! Rysunek 7.4 przedstawia labirynt od środka.



Rysunek 7.4. Znajdowanie drogi w labiryncie

Ostateczna wersja kodu

Listing 7.1 przedstawia gotowy i kompletny kod.

Listing 7.1. Generowanie labiryntów do gry Minecraft

```
#!/usr/bin/env python

"""
Minecraft Maze Maker
Autor: Sean McManus
Z książki Raspberry Pi. Najlepsze projekty
"""

import sys, random
sys.path.append("./mcp/api/python/mcpi")
import minecraft
mc = minecraft.Minecraft.create()

mc.postToChat("Witamy w generatorze labiryntów do Minecrafta!")

def realx(x):
    return MAZE_X+(x*2)-1

def realz(z):
    return MAZE_Z+(z*2)-1

def showMaker(x, z):
    mc.setBlock(realx(x), GROUND+1, realz(z), 41) # 41=zloto

def hideMaker(x, z):
```

```

mc.setBlock(realx(x), GROUND+1, realz(z), 0)

def demolish(realx, realz):
    mc.setBlocks(realx, GROUND+1, realz, realx, HEIGHT+GROUND, realz, 0)

def testAllWalls(cellx, cellz):
    if mc.getBlock(realx(cellx)+1, GROUND+1, realz(cellz))==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx)-1, GROUND+1, realz(cellz))==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx), GROUND+1, realz(cellz)+1)==MAZE_MATERIAL
        ↪and mc.getBlock(realx(cellx), GROUND+1, realz(cellz)-1)==MAZE_MATERIAL:
        return True
    else:
        return False

mc.setting("world_immutable", True)

# Konfigurowanie labiryntu
SIZE = 10
HEIGHT = 2
MAZE_X = 0
GROUND = 0
MAZE_Z = 0
MAZE_MATERIAL = 1 # 1=kamień
GROUND_MATERIAL = 2 # 2=trawa
CEILING = False

mc.setBlocks(MAZE_X-10, GROUND, MAZE_Z-10, MAZE_X+(SIZE*2)+10, GROUND+150,
    ↪MAZE_Z+(SIZE*2)+10, 0) # powietrze
mc.setBlocks(MAZE_X-10, GROUND, MAZE_Z-10, MAZE_X+(SIZE*2)+10, GROUND,
    ↪MAZE_Z+(SIZE*2)+10, GROUND_MATERIAL)
# Przygotowywanie podłoża

mc.setBlock(MAZE_X, GROUND+HEIGHT+1, MAZE_Z, MAZE_MATERIAL)
# Znacznik początku labiryntu

mc.player.setTilePos(MAZE_X+SIZE, GROUND+25, MAZE_Z+SIZE)
# Przenoszenie gracza nad środek labiryntu

mc.postToChat("Trwa budowanie labiryntu...")

# Tworzenie siatki ze ścianami
for line in range(0, (SIZE+1)*2, 2):
    mc.setBlocks(MAZE_X+line, GROUND+1, MAZE_Z, MAZE_X+line, GROUND+HEIGHT,
        ↪MAZE_Z+(SIZE*2), MAZE_MATERIAL)
    mc.setBlocks(MAZE_X, GROUND+1, MAZE_Z+line, MAZE_X+(SIZE*2), GROUND+HEIGHT,
        ↪MAZE_Z+line, MAZE_MATERIAL)

# Konfigurowanie zmiennych używanych przy tworzeniu labiryntu
numberOfCells = SIZE*SIZE
numberOfVisitedCells = 1 # Jeden odpowiada początkowej komórce
cellsVisitedList = []

xposition = random.randint(1, SIZE)
zposition = random.randint(1, SIZE)
playerx = xposition
playerz = zposition
showMaker(xposition, zposition)

```

```

cellsVisitedList.append((xposition, zposition))

while numberOfVisitedCells < numberOfCells:
    possibleDirections = []

    if testAllWalls(xposition-1, zposition):
        possibleDirections.append("left")

    if testAllWalls(xposition+1, zposition):
        possibleDirections.append("right")

    if testAllWalls(xposition, zposition-1):
        possibleDirections.append("up")

    if testAllWalls(xposition, zposition+1):
        possibleDirections.append("down")

    hideMaker(xposition, zposition)

    if len(possibleDirections)!=0:
        directionChosen=random.choice(possibleDirections)

        # Usuwanie ściany w wybranym kierunku
        if directionChosen=="left":
            demolish(realx(xposition)-1, realz(zposition))
            xposition -= 1

        if directionChosen=="right":
            demolish(realx(xposition)+1, realz(zposition))
            xposition += 1

        if directionChosen=="up":
            demolish(realx(xposition), realz(zposition)-1)
            zposition -= 1

        if directionChosen=="down":
            demolish(realx(xposition), realz(zposition)+1)
            zposition += 1

        numberOfVisitedCells += 1
        cellsVisitedList.append((xposition, zposition))
        showMaker(xposition, zposition)

    else: # Ten kod należy wykonać, jeśli nie istnieją nieodwiedzone sąsiednie komórki
        retrace = cellsVisitedList.pop()
        xposition = retrace[0]
        zposition = retrace[1]
        showMaker(xposition, zposition)

if CEILING == True:
    mc.setBlocks(MAZE_X, GROUND+HEIGHT+1, MAZE_Z, MAZE_X+(SIZE*2),
↳GROUND+HEIGHT+1, MAZE_Z+(SIZE*2), 20)

mc.postToChat("Labirynt jest gotowy!")
mc.postToChat("Udanego eksplorowania!")
mc.player.setTilePos(realx(playerx), GROUND+1, realz(playerz))

```

Twoja kolej

Po zbudowaniu labiryntu złota cegła pozostaje widoczna, dlatego możesz spróbować ją znaleźć. Możesz też dodać inne obiekty do znalezienia w labiryncie i określić czas na dotarcie do nich. Polecenie `mc.player.getTilePos()` sprawdza, w którym miejscu świata Minecrafta znajduje się gracz, i zwraca wynik w postaci współrzędnych x , y i z . Kod do tworzenia zegarów znajdziesz w rozdziale 9., „Sprawdź swój czas reakcji”.

Możesz dodać wejście i wyjście w losowych punktach bocznej ściany labiryntu, a następnie jako cel postawić przejście z wejścia do wyjścia. Aby ułatwić przechodzenie dużych labiryntów, dodaj punkty orientacyjne; wykorzystaj różne materiały lub postaw bloki na niektórych ścianach. Po wygenerowaniu labiryntu można usunąć losowe ściany, aby utworzyć skróty. Inna możliwość to zastąpienie wybranych ścian szklanymi blokami, co pozwoli zaglądać w inne korytarze. A co powiesz na wielopoziomowy labirynt ze schodami między poziomami? Możliwości są naprawdę niesamowite.

Skorowidz

A

Abrams Laurence, 82
adapter
 DVI, 33
 HDMI, 33
 HDMI/VGA, 33
 pasywny, 33
 Pi-View, 33
adres IP, 291, 294, 295
alarm, 264, 284
Alcorn Allan, 102
algorytm
 generowania labiryntów
 metodą DSF, 165
 rekurencyjny, 80
analiza danych, *Patrz:* dane
 analiza
Android, 24, 25
aplikacja, *Patrz:* program
Arch, 24
archiwum tar, 231
Arduino, 361, 368
 programowanie, 372, 377,
 381
arkusz kalkulacyjny
 Calc, 420
 Excel, 420
atak
 przez odtwarzanie, 275
 przez przepełnienie
 bufora, 265
 słownikowy, 296
automat skończony, *Patrz:*
 maszyna stanowa
automatyzowanie
 domu, 284, 290

B

Baer Ralph, 102
bajt, 246
barwa, *Patrz też:* kolor
 addytywna, 179
 subtraktywna, 180
baza danych, 196
bezpieczeństwo, 265
 atak
 przez odtwarzanie, 275
 przez przepełnienie
 bufora, 265
 słownikowy, 296

biblioteka, *Patrz też:* pakiet
 Pygame, 84, 87, 105
 Twittera, 220
bit, 246
bit banging, *Patrz:*
 manipulowanie bitami
blitting, 84
błąd
 logiczny, 62, 74
 składni, 62
Braben David, 16
breadboard, *Patrz:* płytką
 prototypowa
budka dla ptaków, 392, 422
 budowa, 397, 399
 oprogramowanie, 402,
 410
buforowanie podwójne, 246
Bushnell Nolan, 102

C

Carpintero Angel, 291
Cellan-Jones Rory, 16
color key, *Patrz:* kolor
 przezroczystości
Compu=Prompt, 82
czcionka, 84
 o stałej szerokości
 znaków, 88
 proporcjonalna, 88
czujnik, 284, 300
 do śledzenia
 małych ssaków, 393
 ptaków, 392
DS18B20, 296, 297
Halla, 360
PIR, *Patrz:* czujnik ruchu
promieni, 393
 detektor, 394, 398
 emiter, 394, 398
rotacji, 368
ruchu, 284, 286
 alarm, 288
 biegun dodatni, 288
 masa, 288
szum, *Patrz:* szum
 temperatury, 296
 oprogramowanie, 297,
 299, 300
czytnik RFID, 264, 273, 274,
 275, 276

D

dane
 analiza, 400
 nadmiarowe, 401
 rejestrowanie, 400
Debian, 24
 pakiet, 201
detektor szczytowy, 258
diagram, 412, 420
 przepływu, 74
dioda
 LED, 176, 200, 206, 217,
 227, 246, 368
 anoda, 181, 183, 217,
 317, 394
 czerwona, 177, 317
 jasność światła, 179
 katoda, 317, 394
 napięcie przebicia, 177
 natężenie prądu, 177
 niebieska, 177
 podczerwona, 394
 SFH484-2 IR, 394
 spadek napięcia, 177
 taśma, *Patrz:* taśma LED
 z równoległym
 układem pinów, 317
 z układem
 przeciwstawnym,
 317
 ze wspólną anodą, 180
 ze wspólną katodą, 180
 zielona, 177, 317
 LED RGB, 176, 179, 180
display memory, *Patrz:*
 pamięć układu
 graficznego
dysk twardy, 25
dziennik
 błędów, 344
 diagnostyczny, 344
dziura, 359
dźwięk, 110, 122, 128, 192,
 196, 224, 225, 242, 245
 częstotliwość, 258
 kompresja, 401
 mono, 128
 sterowanie światłem, 258
 sterownik, 39
dźwójstik, 316
 testowanie, 324

E

edytor tekstu
 Leafpad, 325
 nano, 292

efekt
 dźwiękowy, 127, 128
 Halla, 359

elektron, 210, 359

elektryczność, 210
 masa, *Patrz:* masa
 napięcie, *Patrz:* napięcie
 opór, *Patrz:* opór
 prąd, *Patrz:* prąd

e-mail, 300, 301, 309, 343,
Patrz też: poczta
 elektroniczna
 z raportem, 306
 załącznik, 304

enkoder obrotowy, 334, 359
 absolutny wykorzystujący
 efekt Halla, 359, 360
 AS5040, 368, 373
 błąd, 337
 dane wyjściowe, 335
 drganie styków, 335,
 336, 337
 oprogramowanie, 339
 optyczny, 334
 przyrostowy, 335
 testowanie, 341, 342
 wyjście kwadraturowe,
 335
 względny, 360
 z przełącznikami, 334,
 335
 ząbek, 334, 337

Excel, 420

F

Facebook, 334, 342
 przesyłanie rysunków, 344

Fail2Ban, 296

Fedora, 24

figura Lissajous, 359

FileZilla, 294

filtr
 częstotliwościowa graniczna,
 258
 dolnoprzepustowy, 258
 górnoprzepustowy, 258

flat file, *Patrz:* plik
 jednorodny

Flickr, 343
 logowanie, 344, 345

powiązanie
 z Facebookiem, 345

przesyłanie rysunków,
 343, 344

format

.avi, 294
 .jpg, 294
 .ogg, 110, 122, 128
 .png, 128
 .wav, 128
 MJPEG, 294
 MP3, 401
 tar.gz, 231
 zip, 401

fotodetektor, 394

fototranzystor, 394
 emiter, 395
 kolektor, 395
 na podczerwień, 394
 OSW113, 395

fundacja Raspberry Pi, 16

funkcja, 50, 51
 argument, 50
 choice, 51
 definiowanie, 53
 drawBox, 245
 generująca liczby losowe,
 192
 histogram, 422
 importowanie, 51
 input, 52
 mouseGet, 246
 nazwa, 54
 open, 406
 pakiet, *Patrz:* pakiet
 parametr, *Patrz:* funkcja
 argument
 raw_input, 52
 shuffle, 71

G

general purpose
 input/output, *Patrz:*
 złącze GPIO

generator obwiedni, 260

głośnik, 225, 227

Gmail, 302, 304

gniazdo, *Patrz:* złącze

Goodin Courtney, 82

Google Docs Spreadsheets,
 420

gra
 kółko i krzyżyk, 62
 Magnavox Odyssey, 102
 Minecraft, 152, 154, 155
 czat, 156
 gracz, 158, 160, 168

instalowanie, 152, 153

labirynt, 160, 161, 163,
 164, 166

materiał, 158

uruchamianie, 153

współrzędne, 157, 160

Pac-Man, 126
 ekran, 142
 tryb mocy, 126

Pie Man, 126, 135, 137,
 140
 plansza, 131
 sprite, *Patrz:* sprite
 warstwa, 131

Ping, 103
 dla dwóch graczy, 117
 dla jednej osoby, 111

ping-pong, 102

punktacja, 111

Snap, 176
 dźwięki, 192
 elektronika, 179
 oprogramowanie, 192
 poziom trudności, 192
 pudełko, 184
 testowanie, 189

grafika, 83
 współrzędne, 105

H

hakowanie, 220, 222
 komputerów, 265

Hall Edwin, 359

Halla efekt, *Patrz:* efekt
 Halla

Hancock Tony, 82

harmonograf, 358
 elektronika, 368
 oprogramowanie, 383

hasło domyślne, 39

Higginbotham Willy, 102

histogram, 420

I

IDLE, *Patrz:* środowisko
 IDLE

image, *Patrz:* obraz

informatyka, 400

instrukcja
 def, 226
 git, 232
 if, 72
 import, 206
 ls, 406

lsmod, 202
 minecraft.Minecraft.create,
 157
 pass, 72
 piface/scripts/
 piface-emulator, 205
 print, 50, 228
 startx, 39
 sudo, 28, 301
 sudo reboot, 202
 tar, 231
 tar -zxvf, 153
 time.sleep, 149
 Integrated Development
 Environment, *Patrz:*
 środowisko IDE
 interfejs
 graficzny, 39
 PiFace Digital, 16
 SPI, 360
 internet, 36
 rzeczy, *Patrz:* IOT
 internet of things, *Patrz:*
 IOT
 IOT, 284

J

jarzmo testowe, 66
 język
 C, 361
 C++, 361
 Python, *Patrz:* Python

K

kamera internetowa, 290,
 291, 294, 392
 obraz w internecie, 295
 karta
 interfejsowa, 200
 PiFace, 176, 179
 bufor, 254
 dioda LED, 246
 emulator, 204, 207, 208
 inicjowanie, 228
 konfigurowanie, 254
 przełącznik, 245,
 254, 314
 wejście, 207, 208
 wyjście, 254, 256
 PiFace Digital, 200, 204,
 222, 227, 268, 394
 emulator, 395
 konfiguracja, 200,
 201, 202

oprogramowanie, 201,
 202
 podłączanie, 204
 przekaźnik, 211
 tranzystor, 212
 wejście, 211
 wyjście, 211
 wyjście otwartego
 kolektora, 211, 217
 złącze, 209, 211
 SD, 24, 25, 26
 tworzenie, 26
 katalog, 406
 klasa, 228, 280
 metoda, *Patrz:* metoda
 szczególności IP, 399
 klawiatura
 sterownik, 39
 ze złączem USB, 32
 klawisz
 Home, 122
 Page Down, 122
 Page Up, 122
 klient
 FTP, 294
 SSH, 291, 296
 klucz, 277
 kod, *Patrz:* program
 Graya, 337
 kolektor otwarty, 212, 217,
 218, 288
 kolor, 245, *Patrz też:* barwa
 przezroczystości, 131
 komentarz, 107, 202
 komparator, 258
 kompas elektroniczny, 359
 komponent, 199, 214
 do montażu
 powierzchniowego,
 180, 182
 przewlekany, 180, 181

kompresja
 bezstratna, 401
 stratna, 401
 zip, 401
 komunikat, 228
 konsola tekstowa, 63
 kontaktron, 285
 krotka, 167, 245
 kurczak twittujący, 220

L

lampka, 40
 Lang Jack, 16
 latencja, *Patrz:* opóźnienie
 Lavrsen Kenneth, 291

LeafPad, 44
 LED, *Patrz:* dioda LED
 LibreOffice, 420
 light emitting diode, *Patrz:*
 dioda LED
 lightweight directory access
 protocol, *Patrz:* protokół
 LDAP
 Linux, 24, 25, 27, 28, 39,
 280, 361
 Arch, *Patrz:* Arch
 Debian, *Patrz:* Debian
 dystrybucja, 24
 Fedora, *Patrz:* Fedora
 instrukcja sudo, *Patrz:*
 instrukcja sudo
 openSUSE, *Patrz:*
 openSUSE
 Raspbian Wheezy, 152
 Red Hat, *Patrz:* Red Hat
 Ubuntu, *Patrz:* Ubuntu
 ustawienia lokalne, 405
 Lissajous figura, 359
 lista, 50, 51, 63, 67, 167
 list, 65
 sortowanie, 196
 Lomas Pete, 16
 lutowanie, 215
 LXTerminal, 39

Ł

łańcuch znaków, 49
 łożysko, 367

M

magistrala SPI, 200, 202
 uprawnienia, 204
 magnes, 367
 cyldryczny
 namagnetyzowany
 wzdłuż średnicy, 360,
 365
 manipulowanie bitami, 373
 masa, 210, 211
 maszyna stanowa, 412
 implementacja, 412
 metoda, 228
 __init__, 228
 inicjująca, 228
 pfi0, 228
 Microsoft Windows, *Patrz:*
 Windows
 miernik prądu bezstykowy,
 359

mikrofon, 259
 monitor, 32, 96
 VGA, 33, 34
 Motion, 291
 demon, 293
 instalowanie, 292
 konfiguracja, 293
 skrypt Pythona, 294
 Mullins Rob, 16
 Mycroft Alan, 16
 mysz, 32, 156

N

nadajnik podczerwieni, 394
 napięcie, 210
 notacja wielbłądzia, 54, 157

O

obiekt, 228
 pygame, 244
 obraz, 26, 29, 38
 przesyłanie, 26
 obwód
 równoległy, 222
 szeregowy, 221, 222
 odbiornik astabilny, 307
 odpytywanie, 339
 Ohma prawo, *Patrz:* prawo
 Ohma
 okna otwieranie, 242
 openSUSE, 24
 opór, 210
 opóźnienie, 110
 Oppenheimer Jess, 82
 oprogramowanie Motion,
 Patrz: Motion
 OS X, 24, 27, 29
 oscyloskop, 102

P

pakiet
 folders2flickr, 343, 344
 piface, 245
 pygame, 242
 random, 51
 pamięć, 25
 adresowanie, 25
 blok, 25, 26
 bufor, 246, 408
 obraz, *Patrz:* obraz
 robocza, 83
 suma kontrolna, *Patrz:*
 suma kontrolna

układu graficznego, 83
 para klucz – wartość, 277
 pendrive, 25
 pętla, 54, 412
 for, 54, 55, 64
 nieskończona, 56
 while, 54, 55, 56
 z licznikiem, 55
 piksel, 83
 analiza, 126
 pilot bezprzewodowy, 306
 oprogramowanie, 309
 plik
 .flickrToken, 345
 archiwizacja, 152
 dziennik błędów, 344
 dziennik diagnostyczny,
 344
 history, 344
 jednorodny, 272
 otwieranie, 406
 tryb, 408
 wielkość bufora, 408
 ścieżka, 406
 wykonywalny, 237
 płytka
 prototypowa, 214, 261
 stripboard, 261
 uniwersalna, *Patrz:*
 płytki prototypowa
 poczta elektroniczna, 291,
 301, 392, *Patrz też:* e-mail
 pole
 elektryczne, 359
 magnetyczne, 359
 polecenie, *Patrz:* instrukcja
 port, *Patrz:* złącze
 potencjometr, 334
 prawo Ohma, 210
 prąd, 210
 problemy, 40, 41
 procesor, 24
 program
 Audacity, 110
 błąd, *Patrz:* błąd
 debugowanie, 228
 działający w czasie
 rzeczywistym, 400
 espeak, 224, 225
 LXTerminal, 153, *Patrz:*
 LXTerminal
 łączenie, 414
 Modprobe, 297
 Motion, *Patrz:* Motion
 NOOBS, 26, 152
 Pendulum Pi, 383, 388
 Photoshop Elements, 128
 przerywanie pracy, 56

Putty, 291
 sendmail.py, 301
 struktura, 51, 229
 testowanie, 230, 273
 w tle, 293
 X Server, *Patrz:* X Server
 zapisywanie, 46
 programowanie obiektowe,
 228, 229, 234, 278
 prompter, 92
 budowa, 96
 protokół
 LDAP, 280
 SPI, 373
 SSH, 278
 przeglądarka, 291, 295
 Dillo, 344
 Internet Explorer, 294
 Midori, 344
 przekaźnik, 211, 270
 dwustanowy, *Patrz:*
 przekaźnik przełączny
 NC, 316
 NO, 314, 316
 przełączny, 270
 przełącznik, 176, 180, 184,
 200, 211, 216
 bezstykowy, 359
 drżenie styków, 335, 336
 filtrowanie, 416
 eliminacja drgań, 336
 przechyłny, 339
 ręciovowy, 339
 z wbudowaną diodą LED,
 217
 przerwanie, 339
 przetwornik
 analogowo-cyfrowy, 334
 przezroczystość, 131
 pulpit graficzny, 39
 Python, 44, 49, 50, 152, 156
 bufor, 265
 czytnik RFID, 276
 konsola tekstowa, *Patrz:*
 konsola tekstowa
 moduł, 232
 budowanie, 230
 httplib2, 230, 232
 instalowanie, 202, 204
 matplotlib, 420
 numpy, 420, 422
 odblokowywanie, 202
 python-oauth2, 230, 232
 python-twitter, 230, 232
 random, 157
 shelve, 196
 simplejson, 230, 231
 sys, 157

- program
 espeak, 225
 wywoływanie, 225, 226
skrypt, 294
stała, 404
wcięcia, 53, 54
wersja, 206
- Q**
- QuickTime, 105
- R**
- radio frequency
 identification, *Patrz:*
 RFID
Raspbian, 26, 280
Red Hat, 24
rekurencja, 80
rezystor
 obniżający, 286
 podciągający, 286, 288
RFID, 264, 273, 274, 275
Roto-Sketch, 334
 oprogramowanie, 346, 351
 symetria, 351, 354
ruch, 103
 ką, 103
 kierunek, 105
 odbicie, 104, 111, 112
 zderzenie, *Patrz:*
 zderzenie
ruter, 291, 295
 adres IP, 295
- S**
- Schlafly Hubert, 82
sekwenser, 247, 248
Serial Peripheral Interface,
 Patrz: magistrala SPI
serial protocol interface,
 Patrz: protokół SPI
serwer
 katalogowy, 280
 LDAP, 280
 poczty elektronicznej, 301
 szyfrowanie, 302, 303,
 304
 WWW, 24, 25, 36
sieć, 24
 sterownik, 39
skrypt, 294
 sendmail.py, 294, 309
słownik, 277
- słowo
 def, 53
 elif, 73
 immutable, 160
sprite, 127
 tworzenie, 128, 129
 wielkość, 127
sprzężenie zwrotne
 dodatnie, 259
 ujemne, 259
stała, 160, 228, 403
 Python, 404
sterownik, 201
 dźwięku, *Patrz:* dźwięk
 sterownik
 klawiatury, *Patrz:*
 klawiatura sterownik
 sieci, *Patrz:* sieć
 sterownik
 SPI, 201
stoper, 208, 217
strumień
 standardowy, 414
 stderr, 414
 stdin, 414
 stdout, 414
suma kontrolna, 27, 28, 30
 MD5, 401
sygnał
 analogowy, 33, 34
 cyfrowy, 33, 34
syntezytor mowy, 224, 225
system
 Active Directory, 280
 Git, 203
 kontroli wersji, 203, 343
 krytyczny ze względu na
 bezpieczeństwo, 265
 nawigacji satelitarnej, 225
 oparty na sprite'ach, 84
 operacyjny, 24, 361
 Android, *Patrz:* Android
 Linux, *Patrz:* Linux
 ładowanie, 36, 38
 OS X, *Patrz:* OS X
 Windows, *Patrz:*
 Windows
plików, 25
 ext, 25
 FAT, 25, 26
 HFS Plus, 25
 hierarchiczny, 406
 katalog, 279
 NTFS, 25
 subwersion, 343
 transmodulacji, 176
 wbudowany, 198
sztuczna inteligencja, 70
- szum, 400, 415
 filtrowanie, 416
- Ś**
- środowisko
 IDE, 44
 Arduino, 361, 372
 IDLE, 44, 156, 157, 242
 funkcja, 52
 uruchamianie, 45
 wersja, 45, 206
 zapisywanie kodu, 46
 IDLE 3, 45, 52
 okienkowe, 153, 242
- T**
- tabela rekordów, 196
tablica
 asocjacyjna, 277
 wyszukiwań, 84
 z haszowaniem, 277
taśma LED, 254, 255, 256, 257
teleprompter, 82, 83, *Patrz*
 też: prompter
teoria informacji, 400
terminal, 39, 153
Terminal, 29
test harness, *Patrz:* jarzmo
 testowe
tor z samochodzikami, 314
 kod gry, 327
 oprogramowanie, 325
 testowanie, 316
Torvalds Linus, 24, 203
 tranzystor, 212
 baza, 308
 bipolarny, 308
 emiter, 308
 kolektor, 308
 NPN, 212
Twitter, 220, 230, 392
komunikacja, 233, 234
token, 233
uwierzytelnianie, 233
- U**
- Ubuntu, 24
Upton Eben, 16
urządzenie
 analogowe, 200
 cyfrowe, 200
 nieliniowe, 176
 Wi-Fi, 295

- usługa
 - katalogowa, 279
 - SSH, 291, 296
 - ustawienia lokalne, 405
 - użytkownik
 - domyślny, 39
 - root, 28, 291
- V**
- Vreeken Jeroen, 291
- W**
- wahadło, 358, 359, 361, 362, 367, 388
 - amplituda, 359
 - masa, 362
 - okres, 362
 - współczynnik tłumienia, 362
 - wartość logiczna, 245
 - wiersz poleceń, 28, 29, 235
 - tekstowy, *Patrz:* terminal
 - Windows, 24, 27, 280
 - wyjście kwadraturowe, 335
 - wykres, 420
 - czasowania, 335
 - wyłącznik
 - drzwiowy, 284, 285
 - normalnie otwarty, 285, 314
 - wzmacniacz operacyjny, 259, 260
- X**
- X Server, 39
- Y**
- Yahoo!, 302, 304
- Z**
- zacisk śrubowy, 214
 - zamek do drzwi, 264
 - elektromagnetyczny, 266
 - hasło, 271, 272
 - kontrola drzwi, 268
 - oprogramowanie, 270, 271, 276
 - schemat, 264
 - symulacja programowa, 267
 - testowanie, 278
 - uwierzytelnianie, 271, 272, 276, 280
 - typu wyzwanie – odpowiedź, 275
 - z elektromechanicznym zatrzaskiem, 266
 - zapora, 296
 - zasilanie, 38, 41
 - problemy, 41
 - zdarzenie, 243
 - wciśnięcia przycisku myszy, 246
 - zamknięcia programu, 243
 - zderzenie, 104, 111, 116
 - określenie miejsca, 122
 - zegar, 36
 - złącze
 - DVI, 33
 - ethernetowe, 36
 - GPIO, 200, 284, 285, 286, 308
 - HDMI, 32, 225
 - jack, 225
 - mikro USB, 41
 - RCA, 32, 35
 - USB, 32
 - zmienna, 49, 412
 - bajtowa, 246
 - globalna, 134
 - mc, 157
 - znak
 - _, 230
 - |, 414
 - cudzysłów, 49
 - delta, 103
 - komentarza, 202
 - równości, 49
 - theta, 104

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**

Zbiór najlepszych projektów dla Pi!

Raspberry Pi to prawdziwy komputer o rozmiarach lekko przerośniętej karty kredytowej. Ma ogromny potencjał, całkiem sporą moc obliczeniową, a do tego kosztuje naprawdę niewiele. Platforma ta była projektowana z myślą o nauce programowania dla dzieci. Jednak, jak to często bywa, historia potoczyła się zupełnie inaczej. Obecnie **Raspberry Pi** znajduje zastosowanie jako serwer WWW, odtwarzacz filmów lub sterownik urządzeń. Masz dobry pomysł, żeby wykorzystać Pi w projekcie? A może jeszcze nie wiesz, co chciałbyś zrobić z **Raspberry Pi**?

Ta książka rozwieje wszystkie Twoje wątpliwości i podsunie pomysły na atrakcyjne projekty. W trakcie lektury poznasz budowę **Raspberry Pi** oraz dowiesz się, jak go podłączyć i uruchomić system Linux. Na podstawie kolejnych rozdziałów przygotujesz grę kółko i krzyżyk oraz stworzysz własny teleprompter. Jeżeli potrzebny Ci jest elektroniczny zegar do pomiaru czasu reakcji lub marzy Ci się twittująca zabawka, to trzymasz w ręku właściwą książkę! Jeżeli pragniesz zamieszkać w inteligentnym domu za rozsądne pieniądze — zainteresuje Cię rozdział poświęcony jego automatyzacji. Sprawdź, jakie to proste! Jest to obowiązkowa lektura dla wszystkich pasjonatów, chcących w pełni wykorzystać możliwości **Raspberry Pi**.

Sięgnij po tę książkę i:

- poznaj budowę Raspberry Pi i podstawy korzystania z tej platformy
- wygeneruj labirynt w Minecraft
- zbuduj światła dyskotekowe
- przygotuj własny czujnik ruchu
- skonstruuj swoje elektroniczne marzenia



helion.pl
księgarnia
internetowa



sięgnij po **WIĘCEJ**



KOD KORZYŚCI

Nr katalogowy: 20858



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>

ISBN 978-83-246-9221-7



9 788324 692217

CENA: 69,00 zł

Informatyka w najlepszym wydaniu