

O'REILLY®



Raspberry Pi Receptury

NAJLEPSZE PRZEPISY DLA MINIATUROWEGO KOMPUTERA!



Simon Monk

Tytuł oryginału: Raspberry Pi Cookbook

Tłumaczenie: Konrad Matuk

ISBN: 978-83-246-9625-3

© 2014 Helion S.A.

Authorized Polish translation of the English edition of Raspberry Pi Cookbook,
ISBN 9781449365226 © 2014 Simon Monk.

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 jakiejkolwiek 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 właścicielami.

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, zatrzymaj pod adres
http://helion.pl/user/opinie/raspre_ebook
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

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

- [Poleć książkę na Facebook.com](#)
- [Kup w wersji papierowej](#)
- [Oceń książkę](#)

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

Spis treści

Wstęp	11
1. Podłączanie i konfiguracja	15
1.0. Wprowadzenie	15
1.1. Wybór modelu Raspberry Pi	15
1.2. Zamknięcie Raspberry Pi w obudowie	17
1.3. Wybór zasilacza	18
1.4. Wybór dystrybucji systemu operacyjnego	19
1.5. NOOBS — zapis na kartę SD	20
1.6. Ręczny zapis karty SD (komputery Macintosh)	22
1.7. Ręczny zapis karty SD (system Windows)	23
1.8. Ręczny zapis karty SD (Linux)	25
1.9. Podłączanie urządzeń zewnętrznych do Raspberry Pi	26
1.10. Podłączanie monitora wyposażonego w interfejs DVI lub VGA	27
1.11. Korzystanie z telewizora lub monitora podłączonego za pośrednictwem złącza composite video	28
1.12. Korzystanie z całej pojemności karty SD	29
1.13. Zmiana rozmiaru obrazu wyświetlanego na monitorze	30
1.14. Maksymalizacja wydajności	32
1.15. Zmiana hasła	34
1.16. Uruchamianie Raspberry Pi bezpośrednio w trybie graficznego interfejsu użytkownika	35
1.17. Wyłączanie Raspberry Pi	36
1.18. Instalacja modułu kamery	37
2. Praca w sieci	41
2.0. Wprowadzenie	41
2.1. Łączenie z siecią przewodową	41
2.2. Ustalanie własnego adresu IP	43
2.3. Łączenie z siecią przewodową	44

2.4. Zmiana nazwy, pod którą Raspberry Pi jest widoczne w sieci	45
2.5. Nawiązywanie połączenia z siecią bezprzewodową	46
2.6. Korzystanie z kabla konsolowego	48
2.7. Zdalne sterowanie Raspberry Pi za pomocą protokołu SSH	50
2.8. Sterowanie Raspberry Pi za pomocą VNC	51
2.9. Udostępnianie plików w sieci komputerów Macintosh	52
2.10. Udostępnianie ekranu Raspberry Pi na komputerze Macintosh	54
2.11. Używanie Raspberry Pi jako magazynu NAS	56
2.12. Drukowanie sieciowe	59
3. System operacyjny	61
3.0. Wprowadzenie	61
3.1. Przenoszenie plików w interfejsie graficznym	61
3.2. Uruchamianie sesji Terminala	63
3.3. Przeglądanie plików i folderów za pomocą Terminala	64
3.4. Kopiowanie plików i folderów	66
3.5. Zmiana nazwy pliku lub folderu	67
3.6. Edycja pliku	68
3.7. Oglądanie zawartości pliku	70
3.8. Tworzenie plików bez użycia edytora	71
3.9. Tworzenie katalogów	71
3.10. Kasowanie plików i katalogów	72
3.11. Wykonywanie zadań z uprawnieniami administratora	73
3.12. Co oznaczają atrybuty plików?	74
3.13. Modyfikacja atrybutów plików	75
3.14. Zmiana właściciela pliku	76
3.15. Wykonywanie zrzutów ekranu	77
3.16. Instalacja oprogramowania za pomocą polecenia apt-get	78
3.17. Usuwanie zainstalowanego oprogramowania za pomocą polecenia apt-get	79
3.18. Pobieranie plików za pomocą wiersza poleceń	79
3.19. Pobieranie kodu źródłowego za pomocą polecenia git	80
3.20. Automatyczne uruchamianie programu lub skryptu podczas startu Raspberry Pi	81
3.21. Automatyczne uruchamianie programu lub skryptu w regularnych odstępach czasu	83
3.22. Wyszukiwanie	84
3.23. Korzystanie z historii wiersza poleceń	85
3.24. Monitorowanie aktywności procesora	86
3.25. Obsługa archiwów	88
3.26. Wyświetlanie listy podłączonych urządzeń USB	89
3.27. Zapisywanie w pliku komunikatów wyświetlanych w wierszu poleceń	89
3.28. Obsługa archiwów	90
3.29. Korzystanie z potoków	90

3.30. Ukrywanie danych wyjściowych wyświetlanych w oknie Terminala	91
3.31. Uruchamianie programów w tle	92
3.32. Tworzenie aliasów poleceń	93
3.33. Ustawianie daty i godziny	93
3.34. Ustalanie ilości wolnego miejsca na karcie pamięci	94
4. Oprogramowanie	95
4.0. Wprowadzenie	95
4.1. Tworzenie multimedialnego centrum rozrywki	95
4.2. Instalowanie oprogramowania biurowego	98
4.3. Instalowanie innych przeglądarek internetowych	99
4.4. Korzystanie z Pi Store	101
4.5. Uruchamianie serwera kamery internetowej	102
4.6. Uruchamianie emulatora klasycznej konsoli do gier	104
4.7. Uruchamianie gry Minecraft	105
4.8. Uruchamianie gry Open Arena	106
4.9. Raspberry Pi jako nadajnik radiowy	107
4.10. Uruchamianie edytora grafiki GIMP	109
4.11. Radio internetowe	110
5. Podstawy Pythona	113
5.0. Wprowadzenie	113
5.1. Wybór pomiędzy Pythonem 2 a 3	113
5.2. Pisanie aplikacji Pythona za pomocą IDLE	114
5.3. Korzystanie z konsoli Pythona	116
5.4. Uruchamianie programów napisanych w Pythonie za pomocą Terminala	117
5.5. Zmienne	117
5.6. Wyświetlanie danych generowanych przez program	118
5.7. Wczytywanie danych wprowadzonych przez użytkownika	119
5.8. Działania arytmetyczne	119
5.9. Tworzenie łańcuchów	120
5.10. Scalanie (łączenie) łańcuchów	121
5.11. Konwersja liczb na łańcuchy	122
5.12. Konwersja łańcuchów na liczby	122
5.13. Ustalanie długości łańcucha	123
5.14. Ustalanie pozycji łańcucha w łańcuchu	124
5.15. Wydobywanie fragmentu łańcucha	124
5.16. Zastępowanie fragmentu łańcucha innym łańcuchem	125
5.17. Zamiana znaków łańcucha na wielkie lub małe litery	126
5.18. Uruchamianie poleceń po spełnieniu określonych warunków	127
5.19. Porównywanie wartości	128
5.20. Operatory logiczne	129

5.21. Powtarzanie instrukcji określona liczbę razy	130
5.22. Powtarzanie instrukcji do momentu, w którym zostanie spełniony określony warunek	131
5.23. Przerywanie działania pętli	131
5.24. Definiowanie funkcji	132
6. Python — listy i słowniki	135
6.0. Wprowadzenie	135
6.1. Tworzenie list	135
6.2. Uzyskiwanie dostępu do elementu znajdującego się na liście	136
6.3. Ustalanie długości listy	136
6.4. Dodawanie elementów do listy	137
6.5. Usuwanie elementów z listy	138
6.6. Tworzenie listy w wyniku przetwarzania łańcucha	138
6.7. Iteracja listy	139
6.8. Numerowanie elementów listy	140
6.9. Sortowanie listy	141
6.10. Wycinanie fragmentu listy	141
6.11. Przetwarzanie elementów listy przez funkcję	142
6.12. Tworzenie słownika	143
6.13. Uzyskiwanie dostępu do elementów znajdujących się w słowniku	144
6.14. Usuwanie elementów ze słownika	145
6.15. Iteracja słownika	146
7. Python — zaawansowane funkcje	147
7.0. Wprowadzenie	147
7.1. Tworzenie multimedialnego centrum rozrywki	147
7.2. Formatowanie dat	148
7.3. Zwracanie więcej niż jednej wartości	149
7.4. Definiowanie klasy	149
7.5. Definiowanie metody	151
7.6. Dziedziczenie	152
7.7. Zapis danych w pliku	153
7.8. Odczytywanie pliku	154
7.9. Serializacja	154
7.10. Obsługa wyjątków	155
7.11. Stosowanie modułów	157
7.12. Liczby losowe	158
7.13. Wysyłanie żądań do sieci Web	159
7.14. Argumenty Pythona w wierszu poleceń	159
7.15. Wysyłanie wiadomości pocztą elektroniczną z poziomu aplikacji Pythona	160
7.16. Prosty serwer sieci Web napisany w Pythonie	161

8. Podstawowe zagadnienia dotyczące złącza GPIO	163
8.0. Wprowadzenie	163
8.1. Styki złącza GPIO	163
8.2. Bezpieczne korzystanie ze złącza GPIO	164
8.3. Instalacja biblioteki RPi.GPIO	165
8.4. Konfiguracja magistrali I2C	166
8.5. Korzystanie z narzędzi I2C	167
8.6. Przygotowanie do pracy interfejsu SPI	169
8.7. Zwalnianie portu szeregowego	170
8.8. Instalowanie biblioteki PySerial pozwalającej na korzystanie z portu szeregowego przez aplikacje Pythona	171
8.9. Testowanie portu szeregowego za pomocą aplikacji Minicom	172
8.10. Łączenie Raspberry Pi z płytą prototypową za pomocą przewodów połączeniowych	173
8.11. Łączenie modułu Pi Cobbler z płytą prototypową	174
8.12. Zmniejszanie napięcia sygnałów z 5 do 3,3 V za pomocą dwóch rezystorów	175
8.13. Korzystanie z modułu przetwornika obniżającego napięcie sygnałów z 5 do 3,3 V	177
8.14. Zasilanie Raspberry Pi za pomocą baterii	178
8.15. Zasilanie Raspberry Pi za pomocą akumulatora litowo-polimerowego (LiPo)	179
8.16. Rozpoczynanie pracy z płytą PiFace	181
8.17. Rozpoczynanie pracy z płytą Gertboard	184
8.18. Rozpoczynanie pracy z płytą RaspiRobot	186
8.19. Używanie płytki prototypowej Humble Pi	188
8.20. Używanie płytki prototypowej Pi Plate	190
8.21. Podłączanie płytki drukowanej z zaciskami sprężynowymi	194
9. Sterowanie sprzętem elektronicznym	197
9.0. Wprowadzenie	197
9.1. Podłączanie diody LED	197
9.2. Regulacja jasności diody LED	200
9.3. Generowanie brzęczącego dźwięku	202
9.4. Sterowanie pracą urządzenia o dużej mocy zasilanego prądem stałym za pośrednictwem tranzystora	204
9.5. Włączanie urządzeń o dużej mocy za pomocą przekaźnika	206
9.6. Sterowanie urządzeniami zasilanymi prądem przemiennym o wysokim napięciu	208
9.7. Tworzenie graficznego interfejsu pozwalającego na włączanie i wyłączanie elektroniki podłączonej do Raspberry Pi	210
9.8. Tworzenie graficznego interfejsu użytkownika pozwalającego na sterowanie mocą diod i silników za pomocą modulacji czasu trwania impulsu	211
9.9. Zmiana koloru diody RGB LED	213

9.10. Tworzenie multimedialnego centrum rozrywki	215
9.11. Stosowanie analogowego woltomierza w charakterze wyświetlacza wskazówkowego	218
9.12. Tworzenie programów korzystających z przerwań	220
9.13. Sterowanie złączem GPIO za pomocą sieci Web	223
10. Silniki	227
10.0. Wprowadzenie	227
10.1. Sterowanie pracą serwomotoru	227
10.2. Sterowanie pracą wielu serwomotorów	230
10.3. Sterowanie prędkością obrotową silnika zasilanego prądem stałym	233
10.4. Zmienianie kierunku obrotów silnika zasilanego prądem stałym	235
10.5. Używanie unipolarnych silników krokowych	240
10.6. Korzystanie z bipolarnych silników krokowych	244
10.7. Sterowanie pracą bipolarnego silnika krokowego za pośrednictwem płytki RaspiRobot	246
10.8. Budowa prostego jeżdżącego robota	248
11. Cyfrowe wejścia	253
11.0. Wprowadzenie	253
11.1. Podłączanie przełącznika chwilowego	253
11.2. Korzystanie z przełącznika chwilowego	256
11.3. Korzystanie z dwupozycyjnego przełącznika dwustabilnego lub suwakowego	258
11.4. Korzystanie z przełącznika trójpozycyjnego	259
11.5. Redukcja stuków powstających podczas wciskania przycisku	261
11.6. Korzystanie z zewnętrznego rezystora podwyższającego	263
11.7. Korzystanie z (kwadratowego) kodera obrotowego	265
11.8. Korzystanie z bloku klawiszy	268
11.9. Wykrywanie ruchu	271
11.10. Raspberry Pi i moduł GPS	272
11.11. Wprowadzanie danych z klawiatury	275
11.12. Przechwytywanie ruchów myszy	277
11.13. Korzystanie z modułu zegara czasu rzeczywistego	278
12. Czujniki	283
12.0. Wprowadzenie	283
12.1. Korzystanie z czujników rezystancyjnych	283
12.2. Pomiar jasności światła	287
12.3. Wykrywanie metanu	288
12.4. Pomiar napięcia	291
12.5. Stosowanie dzielnika napięcia	293
12.6. Podłączanie rezystancyjnego czujnika do przetwornika analogowo-cyfrowego	295

12.7. Pomiar temperatury za pomocą przetwornika analogowo-cyfrowego	297
12.8. Pomiar przyspieszenia	299
12.9. Pomiar temperatury za pomocą cyfrowego czujnika	302
12.10. Pomiar odległości	304
12.11. Wyświetlanie mierzonych wielkości	307
12.12. Zapisywanie danych do dziennika utworzonego w pamięci USB	308
13. Wyświetlacze	311
13.0. Wprowadzenie	311
13.1. Korzystanie z czterocyfrowego wyświetlacza LED	311
13.2. Wyświetlanie komunikatów za pomocą wyposażonego w interfejs I2C wyświetlacza składającego się z matrycy diod LED	314
13.3. Korzystanie z płytki Pi-Lite	316
13.4. Wyświetlanie komunikatów na alfanumerycznym wyświetlaczu LCD	318
14. Raspberry Pi i Arduino	323
14.0. Wprowadzenie	323
14.1. Programowanie Arduino za pośrednictwem Raspberry Pi	324
14.2. Komunikacja z Arduino za pośrednictwem monitora portu szeregowego	326
14.3. Sterowanie Arduino za pomocą biblioteki PyFirmata zainstalowanej na Raspberry Pi	328
14.4. Sterowanie pracą cyfrowych wyjść Arduino za pomocą Raspberry Pi	330
14.5. Sterowanie Arduino za pomocą biblioteki PyFirmata za pośrednictwem portu szeregowego	331
14.6. Odczytywanie danych z cyfrowych wejść Arduino za pomocą biblioteki PyFirmata	334
14.7. Odczytywanie danych z analogowych wejść Arduino za pomocą biblioteki PyFirmata	336
14.8. Obsługa wyjść analogowych (PWM) za pomocą biblioteki PyFirmata	337
14.9. Sterowanie pracą serwomotoru za pomocą biblioteki PyFirmata	339
14.10. Komunikacja pomiędzy Raspberry Pi a Arduino za pośrednictwem interfejsu szeregowego bez użycia biblioteki PyFirmata	341
14.11. Tworzenie programu komunikującego się z Arduino za pośrednictwem magistrali I2C	345
14.12. Podłączanie do Raspberry Pi mniejszych płyt Arduino	349
14.13. Podłączanie płytki aLaMode do Raspberry Pi	350
14.14. Korzystanie z shieldów Arduino i płytki aLaMode podłączonej do Raspberry Pi	353
14.15. Stosowanie płytki Gertboard w roli interfejsu Arduino	354
A Komponenty i dystrybutory	355
Skorowidz	361

Wstęp

Raspberry Pi od wprowadzenia na rynek w 2011 roku sprawdza się znakomicie w roli bardzo taniego komputera opartego na Linuksie, pełniącego funkcję platformy systemu wbudowanego. Potwierdza to wielu pedagogów i hobbytów. Sprzedano już ponad 2 miliony tych urządzeń.

W niniejszej książce znajdziesz wiele receptur dotyczących korzystania z Raspberry Pi. Receptury te wiążą się między innymi z rozpoczętym pracą z Raspberry Pi, korzystaniem z języka programowania (Python), obsługą czujników, wyświetlaczy, silników itp. W książce znajdziesz ponadto rozdział poświęcony współpracy Raspberry Pi z płytą Arduino.

Książka ta jest napisana w taki sposób, by można było czytać kolejno wszystkie jej rozdziały (tak jak w przypadku zwyczajnej książki), ale możesz również zaglądać do konkretnych receptur umieszczonej w różnych rozdziałach. W celu znalezienia właściwej receptury możesz się posiliwać spisem treści i indeksem. W podobny sposób korzysta się z książki kucharskiej — przed przygotowaniem dania możesz się najpierw przyjrzeć recepturze samego sosu.

Świat Raspberry Pi zmienia się błyskawicznie. Z urządzeniem tym jest związana duża i aktywna społeczność. Cały czas powstają nowe oprogramowanie i płytki interfejsów. Książka poza przykładami ilustrującymi wykorzystanie konkretnej płytki lub oprogramowania opisuje również pewne podstawowe reguły, które pozwolą Ci lepiej zrozumieć zastosowanie nowych technologii oraz rozwój ekosystemu Raspberry Pi.

Jak się zapewne domyślasz, książka zawiera wiele kodów źródłowych (są to głównie programy napisane w Pythonie). Większość kodu to oprogramowanie otwarte, które znajdziesz w witrynie internetowej książki pod adresem <http://www.helion.pl/ksiazki/raspre.htm>. Na stronie internetowej oryginalnego wydania książki, pod adresem <http://razzpisampler.oreilly.com/>, znajdziesz natomiast związane z zawartością książki dodatkowe materiały (w języku angielskim), w tym filmy instruktażowe.

Aby skorzystać z receptur dotyczących oprogramowania, będziesz musiał posiadać moduł Raspberry Pi. Polecam Raspberry Pi model B. W recepturach dotyczących sprzętu komunikującego się z Raspberry Pi skoncentrowałem się głównie na prefabrykowanych modułach, a także projektach opartych na płytach uniwersalnych niewymagających lutowania obwodów.

Osoby, które chcą, aby budowane przez nie układy były bardziej trwałe, powinny z łatwością zamontować elementy obwodu na płytce prototypowej o połowę mniejszej od płytki uniwersalnej. Płytkę taką można kupić w każdym sklepie sprzedającym podzespoły elektroniczne.

Konwencje typograficzne przyjęte w tej książce

Czcionka pochylona

Adresy internetowe, a także nazwy plików i ich rozszerzeń zapisujemy czcionką pochyloną.

Czcionka o stałej szerokości znaków

Fragmenty kodu zapisujemy czcionką o stałej szerokości znaków.

Pogrubiona czcionka o stałej szerokości znaków

Treść komend wpisywanych samodzielnie przez użytkownika zapisujemy czcionką pogrubioną o stałej szerokości znaków.

Pochylona czcionka o stałej szerokości znaków

Treści abstrakcyjne wpisywane przez użytkownika zapisujemy czcionką pochyloną o stałej szerokości znaków.



Tekst oznaczony tą ikoną jest podpowiedzią, sugestią lub ogólną uwagą.



Tekst oznaczony tą ikoną jest ostrzeżeniem lub pouczeniem.

Korzystanie z przykładowych kodów

Przykładowe kody możesz pobrać ze strony internetowej książki: <http://www.helion.pl/ksiazki/raspre.htm>.

Książka ta powstała, aby Ci pomóc. Zwykle, gdy książka zawiera przykłady kodu, można je wykorzystywać we własnych programach i dokumentach. Nie musisz występować o pozwolenie do autora książki do momentu, w którym chcesz dokonać reprodukci znacznej części kodu zamieszczonego w publikacji. Na przykład nie musisz występować o pozwolenie, jeżeli piszesz program zawierający kilka fragmentów kodu przedstawionego w książce. Jednak sprzedawanie bądź też wprowadzanie do obrotu płyty CD-ROM zawierającej przykładowe kody przedstawione w książce wymaga już zezwolenia. Udzielanie odpowiedzi za pomocą cytatu lub kodu z książki nie wymaga żadnego zezwolenia, ale umieszczenie znacznej ilości przykładowego kodu z tej książki w dokumentacji własnego produktu — wymaga.

Doceniamy, ale nie wymagamy dodawania przypisu. Przypis zawiera zwykle imię i nazwisko autora, tytuł, nazwę wydawnictwa, miejsce oraz datę wydania. Na przykład Simon Monk, *Raspberry Pi. Receptury*, Helion, Gliwice 2014.

Jeżeli obawiasz się, że korzystasz z przykładowych kodów w sposób przekraczający dopuszczalne ramy użytkowania, skontaktuj się z nami: helion@helion.pl.

Podziękowania

Jak zawsze dziękuję Lindzie za cierpliwość i wspieranie mnie podczas pracy nad tą książką.

Dziękuję również korektorom merytorycznym. Są to: Duncan Amos, Chaim Krause i Steve Suehring. Ich komentarze okazały się bardzo przydatne.

Rachel Roumeliotis wspaniale pełniła obowiązki redaktora tego projektu. Jej pragmatyzm sprawił, że nasza współpraca przebiegała bez najmniejszych zakłóceń, co miało pozytywny wpływ na pracę nad tym interesującym projektem.

Chciałbym podziękować także zespołowi wydawnictwa O'Reilly, a zwłaszcza osobom, które spotkałem w biurze w Cambridge — dziękuję wam za ciepłe przyjęcie mojej osoby. Oczywiście dziękuję również Nan Reinhart za sumienne pełnienie obowiązków adiustatora.

Podłączanie i konfiguracja

1.0. Wprowadzenie

Kupując Raspberry Pi, tak naprawdę dokonujesz zakupu tylko gotowej płytki układu — nie posiada ona nawet zasilacza ani nie zainstalowano na niej systemu operacyjnego.

Receptury zawarte w tym rozdziale dotyczą konfigurowania oraz przygotowywania do pracy Twojego Raspberry Pi.

Do Raspberry Pi możesz podłączyć standardową klawiaturę oraz mysz (urządzenia te muszą być wyposażone w interfejs USB). Podłączenie ich jest dość proste i niczym się nie różni od podłączania tych urządzeń do komputera. Skoncentrujemy się na pewnych aspektach charakterystycznych dla Raspberry Pi.

1.1. Wybór modelu Raspberry Pi

Problem

Istnieją dwa modele Raspberry Pi — A i B. Nie wiesz, który z nich wybrać.

Rozwiążanie

Jeżeli chcesz stosować Raspberry Pi do czegoś więcej niż tylko jednego projektu, to powinieneś zakupić model B (w najnowszej wersji rev2). Model ten wyposażono w dwa razy więcej pamięci niż model A, a więc będzie on sprawdzał się o wiele lepiej w większości zastosowań.

Jeżeli planujesz zastosować Raspberry Pi w tylko jednym projekcie, to zakup modelu A pozwoli Ci zaoszczędzić trochę pieniędzy.

Omówienie

Na rysunku 1.1 porównano modele A i B.



Rysunek 1.1. Raspberry Pi: model A (po stronie lewej) i model B (po stronie prawej)

Jak widać na rysunku 1.1, oba modele zbudowano w oparciu o tę samą płytę drukowaną obwodu. Model A wyposażono w pojedyncze złącze USB i nie umieszczono na nim gniazda sieci Ethernet RJ45. Za gniazdem USB widoczne są punkty, do których powinien być przyłutowany układ kontrolera sieci Ethernet.

Różnice pomiędzy modelami przedstawiono w tabeli 1.1. W tabeli umieszczone również model B we wcześniejszej wersji rev1, która została szybko zastąpiona wersją rev2. Wersje modelu B można rozpoznać po kolorze gniazda audio — w modelu wcześniejszym było ono koloru czarnego, a w późniejszym zastąpiono je niebieskim gniazdem.

Tabela 1.1. Modele Raspberry Pi

Model	Pamięć RAM	Gniazda USB	Gniazdo sieci Ethernet
A	256 MB	1	nie
B rev2	512 MB	2	tak
B rev1	256 MB	2	tak

Brak interfejsu sieciowego w modelu A nie jest problematyczny, ponieważ do Raspberry Pi można podłączyć kontroler Wi-Fi za pośrednictwem gniazda USB (zobacz receptura 2.5). W związku z tym, że taki kontroler zająłby jedyne złącze USB, w celu zwiększenia liczby dostępnych portów niezbędne byłoby zastosowanie dodatkowego koncentratora USB. Raspberry Pi model B wyposażono w dwa gniazda USB, a więc do jednego z nich możesz podłączyć kontroler sieci bezprzewodowej, a do drugiego — odbiornik sygnału nadawanego przez bezprzewodową klawiaturę i mysz.

Zobacz również

Więcej informacji na temat różnic pomiędzy poszczególnymi modelami Raspberry Pi znajdziesz na stronie http://pl.wikipedia.org/wiki/Raspberry_Pi.

1.2. Zamknięcie Raspberry Pi w obudowie

Problem

Chcesz umieścić swoje Raspberry Pi w obudowie.

Rozwiązanie

Jeżeli nie kupiłeś Raspberry Pi w formie większego zestawu, to z pewnością nie posiada ono obudowy. Dlatego też trzeba się z nim obchodzić dość ostrożnie — na spodniej stronie płytki znajdują się złącza, które mogą zostać z łatwością zwarte po umieszczeniu płytki na czymś metalowym.

Warto zakupić obudowę chroniącą Raspberry Pi.

Omówienie

Istnieje wiele obudów, spośród których możesz wybierać. Są to między innymi:

- proste dwuczęściowe obudowy plastikowe wyposażone w zatrzaski,
- obudowy VESA (przeznaczone do montażu na tylnej ścianie monitora lub telewizora),
- obudowy przypominające kształtem klocki lego,
- obudowy wydrukowane za pomocą drukarki 3D,
- obudowy wycięte z akrylu za pomocą lasera.

Wybór obudowy zależy głównie od Twojego gustu, powinieneś jednak rozważyć pewne kwestie:

- Czy potrzebujesz dostępu do złącza GPIO? Dostęp ten przyda Ci się, jeżeli planujesz podłączać zewnętrzne komponenty elektroniczne do swojego Raspberry Pi.
- Czy obudowa jest należycie wentylowana? Jest to ważne, jeżeli planujesz przetaktować swoje Raspberry Pi (receptura 1.14) lub obciążać je odtwarzaniem filmów bądź graniem w gry — spowoduje to wydzielanie większej ilości ciepła przez urządzenie.

Zobacz również

Firma Adafruit dostarcza sporo obudów przeznaczonych dla Raspberry Pi. Możesz je obejrzeć w internecie na stronie http://www.adafruit.com/index.php?main_page=adasearch&q=raspberry+pi+enclosure.

Różne obudowy przeznaczone dla Raspberry Pi znajdziesz również na serwisach aukcyjnych takich jak Allegro i eBay.

1.3. Wybór zasilacza

Problem

Potrzebujesz zasilacza do swojego Raspberry Pi.

Rozwiążanie

Urządzenie zasilające Raspberry Pi powinno dostarczać regulowany prąd stały o napięciu 5 V i maksymalnym natężeniu 700 mA. Zasilacz powinien być wyposażony w kabel zakończony wtyczką micro USB.

W przypadku zakupu zasilacza w tym samym miejscu, w którym dokonałeś zakupu Raspberry Pi, sprzedawca powinien poinformować Cię o tym, który ze sprzedawanych zasilaczy może współpracować z tym urządzeniem.

Jeżeli zamierzasz korzystać z urządzeń USB pobierających dużą ilość prądu, takich jak kontroler sieci bezprzewodowej Wi-Fi, to powinieneś zaopatrzyć się w zasilacz mogący dostarczyć prąd o natężeniu 1,5 A lub nawet 2 A. Wystrzegaj się bardzo tanich zasilaczy, które mogą nie być w stanie stale dostarczać prądu o właściwym napięciu (5 V).

Omówienie

Tak naprawdę takie same zasilacze jak te służące do zasilania Raspberry Pi stosuje się w roli ładowarek wielu smartfonów. Jeżeli dysponujesz takim zasilaczem, a dodatkowo posiada on wtyczkę micro USB, to niemal z pewnością generuje on prąd o napięciu 5 V (musisz to jednak sprawdzić). Jedynym problemem może być dostarczanie przez zasilacz zbyt małego natężenia prądu.

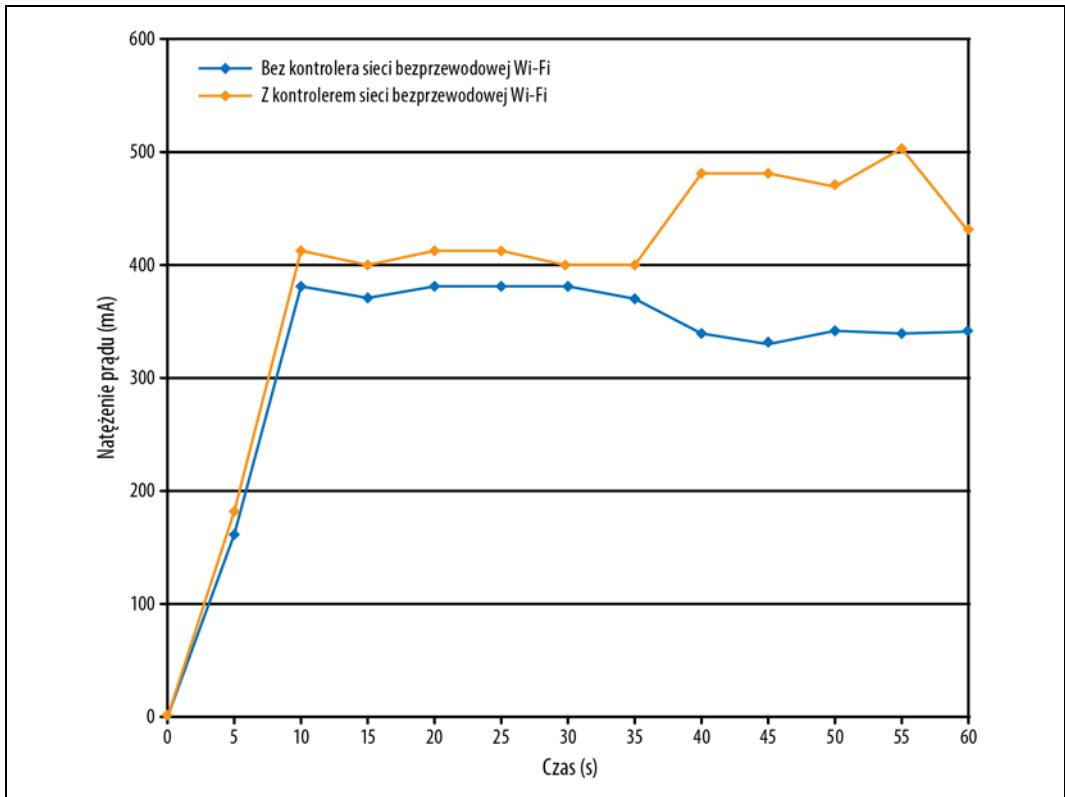
Jeżeli zasilacz nie jest w stanie dostarczyć prądu o odpowiednim natężeniu, to:

- może się on nadmiernie nagrzewać i stwarzać zagrożenie pożarowe,
- może on ulec uszkodzeniu,
- podczas dużego obciążenia zasilacza (np. gdy Raspberry Pi korzysta z kontrolera sieci bezprzewodowej) napięcie prądu zasilającego może spadać, co może prowadzić do samoczynnego ponownego uruchamiania się Raspberry Pi.

Szukaj zasilacza mogącego dostarczyć prąd o natężeniu co najmniej 0,7 A (700 mA). Jeżeli na obudowie zasilacza podano jego moc wyrażoną w watach (W), to podziel tę wartość przez 5 i otrzymasz wartość natężenia prądu wyrażoną w amperach (A). Zasilacz generujący prąd o napięciu 5 V, charakteryzujący się mocą 10 W, dostarcza prąd o natężeniu 2 A (2000 mA).

Korzystanie z zasilacza generującego prąd o maksymalnym natężeniu 2 A nie spowoduje większego zużycia prądu niż zastosowanie zasilacza generującego prąd o maksymalnym natężeniu 700 mA. Raspberry Pi będzie pobierało tylko tyle prądu, ile potrzebuje.

Na rysunku 1.2 przedstawiono wyniki pomiarów natężenia prądu pobieranego przez Raspberry Pi (model B rev2) podczas uruchamiania z kontrolerem sieci bezprzewodowej Wi-Fi oraz bez niego. Podczas testów korzystano ze złącza wideo HDMI i systemu Raspbian Wheezy.



Rysunek 1.2. Natężenie prądu pobieranego przez Raspberry Pi podczas uruchamiania

Możesz zobaczyć, że natężenie prądu rzadko przekracza 500 mA. Tak naprawdę jednak procesor nie jest tutaj szczególnie mocno obciążony. Odtworzenie wideo w rozdzielczości HD obciążałoby procesor znacznie bardziej i urządzenie pobierałoby o wiele większy prąd. Warto, aby zasilacz dysponował pewną rezerwą mocy.

Zobacz również

Na stronie <http://www.pi-supply.com/> możesz kupić zasilacz, który jest automatycznie wyłączany wraz z Raspberry Pi.

1.4. Wybór dystrybucji systemu operacyjnego

Problem

Istnieje wiele różnych dystrybucji Raspberry Pi. Nie wiesz, którą z nich wybrać.

Rozwiążanie

Odpowiedź na to pytanie zależy od tego, do czego chcesz stosować swoje Raspberry Pi.

Jeżeli Raspberry Pi ma być sprzętowym elementem Twojego projektu, to korzystaj z dystrybucji Raspbian lub Occidental (od Adafruit). Raspbian jest oficjalnym i najpopularniejszym systemem, pod kontrolą którego pracuje Raspberry Pi, jednak Occidental da się skonfigurować szybciej w celu rozpoczęcia hakowania sprzętu.

Jeżeli planujesz stosować Raspberry Pi w roli odtwarzacza multimedialnego, możesz znaleźć dystrybucję zoptymalizowaną właśnie pod tym kątem (zobacz receptura 4.1).

W niniejszej książce korzystamy prawie wyłącznie z dystrybucji Raspbian, jednak większość receptur będzie się sprawdzać w dowolnej dystrybucji opartej na Debianie.

Omówienie

Karty SD, a zwłaszcza karty SD o pojemności 4 GB, które są zalecane w celu zainstalowania większości dystrybucji systemu Linux, są tanie. Kup kilka takich kart i wypróbuj parę dystrybucji. W takim przypadku dobrym pomysłem może się okazać trzymanie własnych plików na zewnętrznym dysku flash wyposażonym w interfejs USB — nie będziesz musiał kopować ich na każdą kartę SD.

Zwróć uwagę na to, że aby skorzystać z kolejnych receptur (samodzielnego zapisu na karcie SD), musisz mieć komputer wyposażony w gniazdo kart SD (wiele laptopów je posiada) lub czeka Cię zakup taniego czytnika kart SD wyposażonego w interfejs USB.

Zobacz również

Oficjalna lista dystrybucji Raspberry Pi: <http://www.raspberrypi.org/downloads/>.

Occidental: <https://learn.adafruit.com/adafruit-raspberry-pi-educational-linux-distro/occidental-v0-dot-2>.

1.5. NOOBS — zapis na kartę SD

Problem

Chcesz dokonać zapisu na karcie SD za pomocą oprogramowania NOOBS.

Rozwiązanie

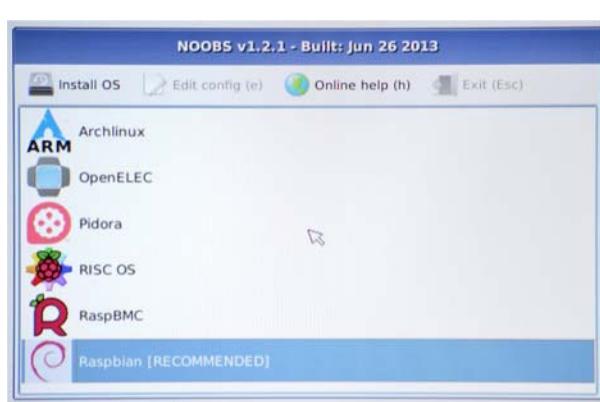
Ściagnij archiwum z oprogramowaniem NOOBS ze strony <http://www.raspberrypi.org/downloads/>, rozpakuj je i przenieś jego zawartość na kartę SD.



NOOBS pobiera niezbędne oprogramowanie z sieci. Jeżeli korzystasz z modelu A Raspberry Pi, który nie posiada złącza sieciowego, to mogą Ci się przydać receptury przedstawiające instalację całego systemu operacyjnego (receptury 1.6, 1.7 i 1.8).

Po pobraniu archiwum NOOBS rozpakuj je. Rozpakowaną zawartość archiwum skopiuj na kartę SD. Archiwum zostanie rozpakowane do folderu o nazwie *NOOBS_v1_2_1* lub podobnej. Pliki i katalogi znajdujące się w tym folderze należy skopiować bezpośrednio do głównego katalogu karty SD.

Kartę SD zawierającą rozpakowane pliki NOOBS włożyć do Raspberry Pi, a następnie podłącz zasilanie do tego urządzenia. Po uruchomieniu wyświetli się okienko widoczne na rysunku 1.3. Z tego ekranu możesz wybrać dystrybucję, którą chcesz zainstalować. Domyślną dystrybucją jest Raspbian. Jest to najlepszy wybór na początek pracy z Raspberry Pi.



Rysunek 1.3. Ekran powitalny NOOBS

Wyświetli się komunikat ostrzegający o tym, że karta SD zostanie nadpisana (tego właśnie chcemy), a następnie dystrybucja zostanie zainstalowana na karcie SD. W tym czasie wyświetlany będzie komunikat informujący o przebiegu instalacji. Będzie on zawierał również przydatne informacje na temat instalowanej dystrybucji (zobacz rysunek 1.4).



Rysunek 1.4. NOOBS nadpisujący zawartość karty SD

Po zakończeniu kopiowania plików wyświetli się komunikat o treści *Image applied successfully*, który informuje Cię o poprawnie zakończonym procesie instalowania obrazu. Wciśnięcie klawisza *Enter* spowoduje ponowne uruchomienie Raspberry Pi i automatyczne załadowanie narzędzia *raspi-config*, które posłuży do skonfigurowania świeżo zainstalowanego systemu (receptura 1.12).

Omówienie

W celu przeprowadzenia poprawnej instalacji NOOBS karta SD musi być sformatowana w systemie plików FAT. Większość kart SD jest fabrycznie formatowana w tym systemie. Jeżeli korzystasz z używanej karty i musisz ją sformatować, użyj do tego narzędzia do formatowania dysków wymiennych dostępnego w Twoim systemie operacyjnym.

Możesz w tym celu skorzystać również z narzędzia formatującego SD Association, które jest zalecane przez producenta Raspberry Pi i jest dostępne w wersji na komputery pracujące pod kontrolą systemu Mac OS X oraz Windows. Możesz je pobrać ze strony https://www.sdcard.org/downloads/formatter_4/.

Zobacz również

Zobacz receptury 1.6, 1.7 i 1.8 — zawierają one informacje dotyczące instalowania obrazu pełnego systemu operacyjnego na karcie SD.

1.6. Ręczny zapis karty SD (komputery Macintosh)

Problem

Raspberry Pi nie jest dostarczane z fabrycznie zainstalowanym systemem operacyjnym. Chcesz zapisać dystrybucję systemu operacyjnego na karcie SD z komputera typu Macintosh.

Rozwiążanie

Ściagnij obraz *iso* wybranej dystrybucji (zobacz receptura 1.4). Następnie zapisz go na karcie SD za pomocą skryptu narzędziowego.

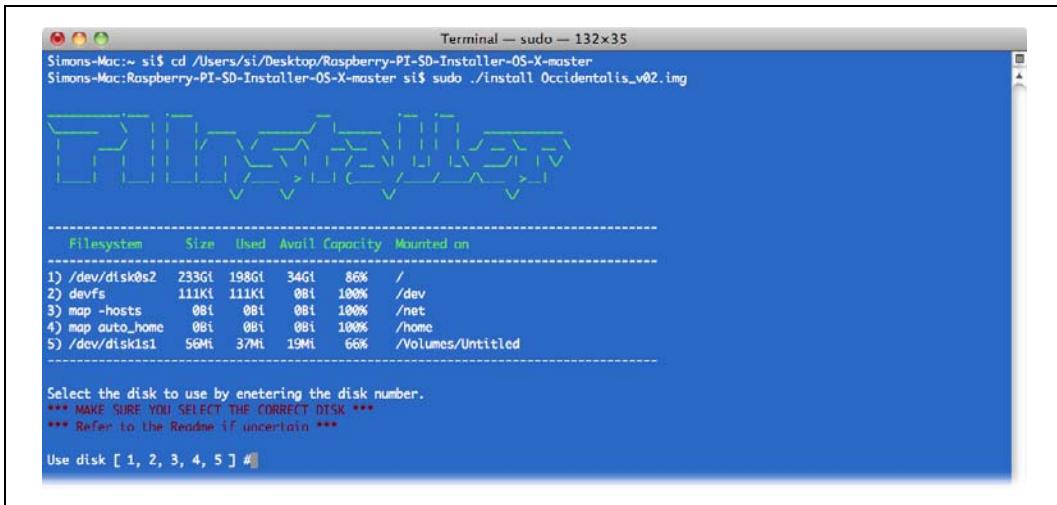
Istnieje wiele programów narzędziowych stworzonych po to, by pomóc Ci w zapisie kart SD. Popularne oprogramowanie tego typu znajdziesz pod adresem <http://bit.ly/viljoen-installer>.

1. Pobierz i rozpakuj (do dowolnej lokacji) folder znajdujący się w archiwum zip.
2. Skopiuj do tego folderu pobrany wcześniej plik *iso* zawierający obraz systemu. Załóżmy, że plik ten ma nazwę *Occidentalis_v02.iso*.
3. Odłącz od gniazd USB komputera wszelkie zewnętrzne dyski twarde lub pamięci flash. Włożyć kartę SD.
4. Otwórz sesję Terminala, zmień bieżący katalog (polecenie `cd`) na folder instalatora i uruchom następujące polecenie:

```
sudo ./install Occidentalis_v02.iso
```

Jeżeli korzystasz z innego obrazu *iso*, to po prostu zastosuj jego nazwę. Powyższe polecenie uruchamia instalator kart SD. Na ekranie wyświetli się treść podobna do zawartości rysunku 1.5.

Teraz musisz wybrać numer dysku, który odpowiada karcie SD. Dobrym pomysłem jest odłączenie wszystkich dysków przenośnych od komputera. Jeżeli wybierzesz niewłaściwą opcję, program skasuje zawartośćomyłkowo wybranego dysku. Zanim wykonasz ten krok, upewnij się, że wiesz, który numer odpowiada karcie SD.



Rysunek 1.5. Zapis na karcie SD (Macintosh)

Po wybraniu karty, jeżeli wszystko przebiegło poprawnie, nastąpi zapis danych. Trwa on zwykle kilka minut.

Omówienie

Możesz również zakupić gotowe karty SD, na których fabrycznie zainstalowano dystrybucję jakiegoś systemu operacyjnego. Niestety nie zawsze system taki będzie w najnowszej wersji, a więc warto umieć samodzielnie zapisywać dane na karcie SD.

Wybierając kartę SD, upewnij się, że będziesz korzystać z karty o pojemności przynajmniej 4 GB.

Zobacz również

Zobacz recepturę 1.5. Opisano w niej proces tworzenia karty SD za pomocą narzędzi NOOBS.

Istnieją alternatywne narzędzia służące do zapisywania danych na kartach SD w systemie Mac OS X, takie jak np. <http://alltheware.wordpress.com/2012/12/11/easiest-way-sd-card-setup/>.

Więcej informacji (w języku angielskim) na temat zapisu na kartach SD znajdziesz pod adresem: http://elinux.org/RPi_Easy_SD_Card_Setup.

1.7. Ręczny zapis karty SD (system Windows)

Problem

Raspberry Pi nie jest dostarczane z fabrycznie zainstalowanym systemem operacyjnym. Chcesz zapisać dystrybucję systemu operacyjnego na karcie SD za pomocą komputera PC pracującego pod kontrolą systemu Windows.

Rozwiążanie

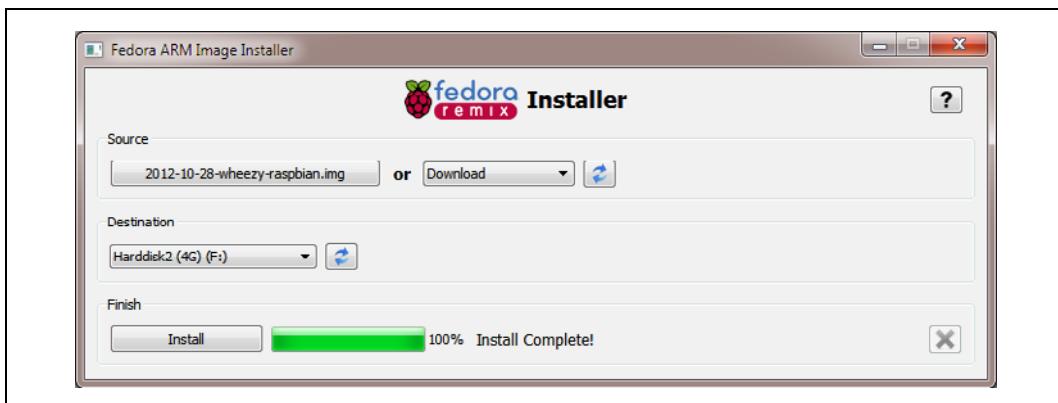
Ściagnij obraz *iso* wybranej dystrybucji (zobacz receptura 1.4). Następnie zapisz go na karcie SD za pomocą programu Fedora ARM Installer.

Zacznij od pobrania tego programu ze strony http://fedoraproject.org/wiki/Fedora_ARM_Installer - *Windows_Vista_.26_7*.

Program ten będzie działał tylko w systemie Windows Vista lub nowszym.

- Wypakuj folder znajdujący się w archiwum zip do wybranej lokalizacji (np. na pulpit).
- Odłącz od komputera wszystkie zewnętrzne dyski twardy i pamięci flash wyposażone w interfejs USB, a następnie włożyć kartę SD.
- Uruchom plik *fedora-arm-installer.exe*. Musisz być zalogowany jako użytkownik posiadający uprawnienia administratora.

Narzędzie to ma przyjazny interfejs (zobacz rysunek 1.6).



Rysunek 1.6. Zapis na kartę SD (Windows)

W sekcji *Source* wybierz pobrany wcześniej plik *iso*, a w sekcji *Destination* wskaż docelowy napęd, na który ma zostać zapisany obraz systemu. Upewnij się, że wybrałeś napęd będący kartą SD, a nie jakimś innym dyskiem. Teraz wystarczy, że klikniesz przycisk *Install*.

Omówienie

Możesz również zakupić gotowe karty SD, na których fabrycznie zainstalowano dystrybucję jakiegoś systemu operacyjnego. Niestety nie zawsze system taki będzie w najnowszej wersji, a więc warto umieć samodzielnie zapisywać dane na karcie SD.

Wybierając kartę SD, upewnij się, że będziesz korzystać z karty o pojemności przynajmniej 4 GB.

Zobacz również

Zobacz recepturę 1.5. Opisano w niej proces tworzenia karty SD za pomocą narzędzi NOOBS.

Więcej informacji na temat zapisu kart SD znajdziesz pod adresem http://elinux.org/RPi_Easy_SD_Card_Setup.

1.8. Ręczny zapis karty SD (Linux)

Problem

Raspberry Pi nie jest dostarczane z fabrycznie zainstalowanym systemem operacyjnym. Chcesz zapisać dystrybucję systemu operacyjnego na karcie SD za pomocą komputera PC pracującego pod kontrolą systemu Linux.

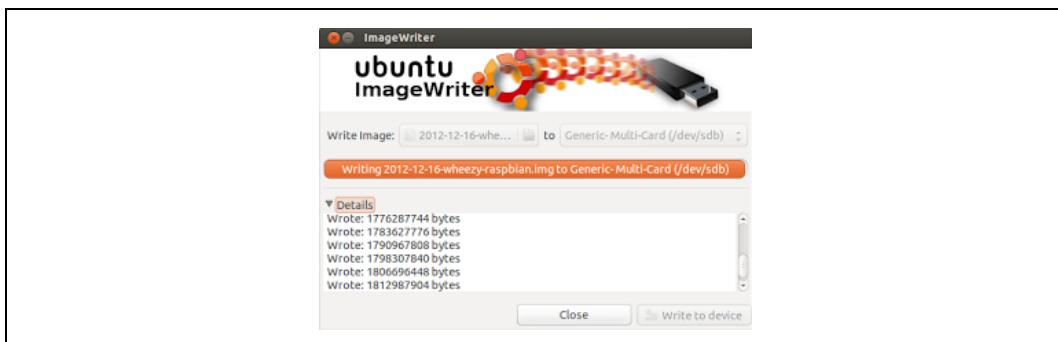
Rozwiążanie

Ściągnij obraz *iso* wybranej dystrybucji (zobacz receptura 1.4). Następnie zapisz go na karcie SD za pomocą narzędzia *ImageWriter* (dystrybucja Ubuntu). Jeżeli posiadasz inną dystrybucję Linuksa, to możesz skorzystać z narzędzi NOOBS (receptura 1.5).

Pracę zacznij od zainstalowania narzędzia *ImageWriter* — otwórz sesję Terminala i wprowadź następujące polecenie:

```
$ sudo apt-get install usb-imagewriter
```

Narzędzie to ma przyjazny interfejs (zobacz rysunek 1.7).



Rysunek 1.7. Zapis na kartę SD (Linux)

W sekcji *Write Image* wybierz pobrany wcześniej plik *iso*, a w sekcji *to* wybierz docelowy napęd, na który ma zostać zapisany obraz systemu. Upewnij się, że wybrałeś napęd będący kartą SD, a nie jakimś innym dyskiem. Teraz wystarczy, że klikniesz przycisk *Write to device*.

Omówienie

Możesz również zakupić gotowe karty SD, na których fabrycznie zainstalowano dystrybucję jakiegoś systemu operacyjnego. Niestety nie zawsze system taki będzie w najnowszej wersji, a więc warto umieć samodzielnie zapisywać dane na karcie SD.

Wybierając kartę SD, upewnij się, że będziesz korzystać z karty o pojemności przynajmniej 4 GB.

Zobacz również

Zobacz recepturę 1.5. Opisano w niej proces tworzenia karty SD za pomocą narzędzi NOOBS.

Więcej informacji na temat zapisu kart SD znajdziesz pod adresem http://elinux.org/RPi_Easy_SD_Card_Setup.

1.9. Podłączanie urządzeń zewnętrznych do Raspberry Pi

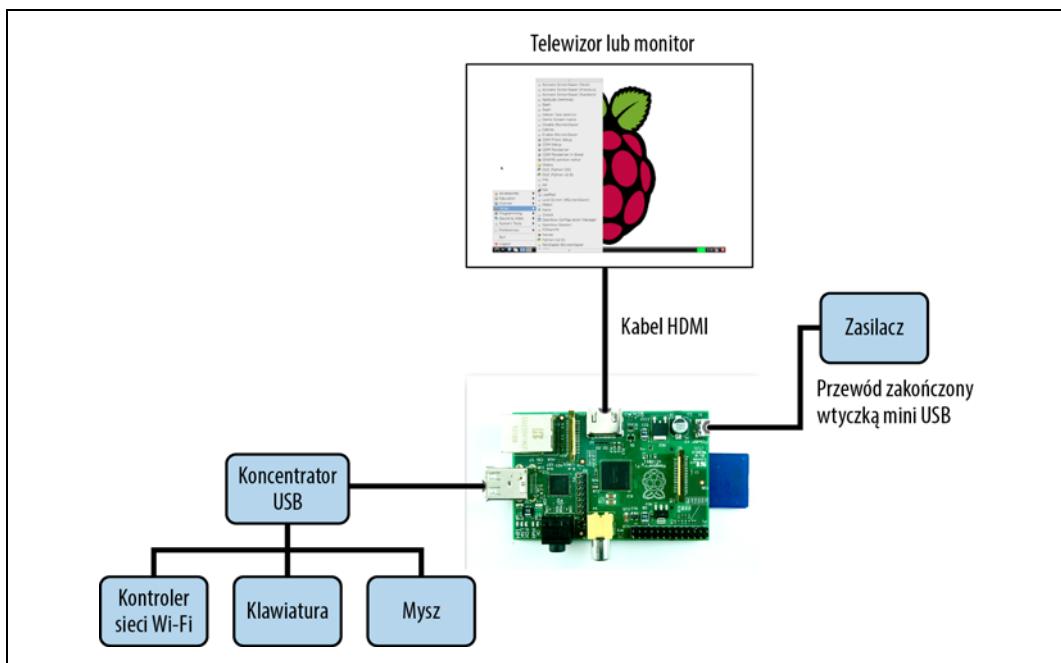
Problem

Posiadasz już wszystkie rzeczy potrzebne Ci do pracy i chcesz podłączyć je do Raspberry Pi.

Rozwiążanie

Najprawdopodobniej podłączysz do swojego Raspberry Pi klawiaturę, mysz, monitor i kontroler sieci bezprzewodowej Wi-Fi. Jeżeli Twoje Raspberry Pi będzie pracować w roli odtwarzacza multimedialnego lub podzespołu składowego jakiegoś większego projektu, to możesz zrezygnować z tych urządzeń. W celu ich podłączenia będziesz potrzebował trzech gniazd USB. Nawet jeżeli posiadasz Raspberry Pi model B, to będziesz potrzebował do tego koncentratora USB zapewniającego odpowiednią liczbę portów.

Na rysunku 1.8 przedstawiono typowy system Raspberry Pi.



Rysunek 1.8. Typowy system Raspberry Pi

Omówienie

Jeżeli korzystasz z zestawu składającego się z bezprzewodowej klawiatury i myszy, który posiada jeden klucz sprzętowy (zajmuje jedno gniazdo USB), to drugie wolne gniazdo USB (zakładam, że posiadasz model B Raspberry Pi) może być użyte do podłączenia kontrolera sieci Wi-Fi. Najprawdopodobniej jednak i tak przyda Ci się koncentrator USB — pozwoli Ci on na podłączenie zewnętrznego dysku USB lub pamięci USB.

Raspberry Pi może współpracować z praktycznie dowolnymi klawiaturami i myszami, niezależnie od tego, czy są one przewodowe, czy bezprzewodowe. Wyjątek stanowią tu urządzenia wyposażone w bezprzewodowy interfejs Bluetooth — tego typu klawiatury i myszy nie będą współpracować z Raspberry Pi.

Zobacz również

Oficjalny poradnik dla osób rozpoczynających przygodę z Raspberry Pi: <http://www.raspberrypi.org/help/quick-start-guide/>.

1.10. Podłączanie monitora wyposażonego w interfejs DVI lub VGA

Problem

Monitor, który chcesz podłączyć do Raspberry Pi, nie posiada złącza HDMI.

Rozwiążanie

Z problemem tym ma do czynienia wielu użytkowników. Na szczęście możliwy jest zakup adapterów pozwalających na konwersję sygnału HDMI na sygnał akceptowany przez złącza DVI lub VGA.

Adaptery DVI są najprostsze i najtańsze. Takie przejściówki można znaleźć na serwisach aukcyjnych już za kilkanaście złotych — wystarczy wpisać w wyszukiwarce hasło „przejściówka HDMI na DVI”. Adapter powinien posiadać męskie złącze HDMI i żeńskie DVI.

Omówienie

Konwersja sygnału HDMI na sygnał VGA jest bardziej złożona — proces ten wymaga zastosowania podzespołów elektronicznych. Wystrzegaj się przewodów niewypozażonych w układy elektroniczne. Oficjalny adapter nosi nazwę **Pi-View** i można go nabyć w punktach sprzedaży Raspberry Pi. Pi-View zostało przetestowane i możesz mieć pewność, że będzie współpracować z Raspberry Pi. Oczywiście możesz znaleźć tańsze odpowiedniki tego układu, jednak nie można mieć pewności co do tego, czy będą współpracować z Raspberry Pi.

Zobacz również

W serwisie elinux można znaleźć pewne wskazówki dotyczące wyboru właściwego adaptera: http://elinux.org/RPi_VerifiedPeripherals#HDMI-3EVGA_converter_boxes.

1.11. Korzystanie z telewizora lub monitora podłączonego za pośrednictwemłącza composite video

Problem

Tekst wyświetlany na ekranie o niskiej rozdzielcości jest nieczytelny. Musisz dostosować rozdzielcość, z jaką pracuje Raspberry Pi, do wymagań małego ekranu.

Rozwiążanie

Raspberry Pi poza złączem HDMI posiada również złącze zespolonego sygnału wizyjnego. Złącze HDMI zapewnia obraz o znacznie wyższej jakości. Jeżeli chcesz łączyć Raspberry Pi z głównym ekranem za pośrednictwem złącza composite video, to przemyśl to jeszcze raz.

Jeżeli korzystasz z tego złącza, ponieważ — powiedzmy — potrzebujesz naprawdę małego ekranu, to powinieneś dokonać kilku zmian w generowanym sygnale wideo. Musisz zmodyfikować plik `/boot/config.txt`. Modyfikacji tej możesz dokonać bezpośrednio z Raspberry Pi. Wystarczy wpisać poniższe polecenie w konsoli:

```
$ sudo nano /boot/config.txt
```

Jeżeli tekst wyświetlany na ekranie jest na tyle mały, że nie możesz go odczytać, a nie posiadasz monitora wyposażonego w złącze HDMI, to możesz wyjąć kartę SD z Raspberry Pi i edytować plik po włożeniu karty do komputera. Plik będzie się znajdował w głównym katalogu karty SD. Możesz go zmodyfikować na komputerze za pomocą edytora plików tekstowych.

Musisz wiedzieć, jaką rozdzielcość ma Twój ekran. Wiele małych ekranów charakteryzuje się rozdzielcością 320 na 240 pikseli. Odnajdź w edytowanym pliku poniższe wiersze:

```
#framebuffer_width=1280  
#framebuffer_height=720
```

Usuń symbol półtka (#) z początków tych dwóch linii i zmień wartości odpowiadające za szerokość (width) i wysokość (height) ekranu. Poniżej znajduje się modyfikacja tych wierszy dostosowująca Raspberry Pi do pracy z ekranem o rozdzielcości 320 na 240 pikseli:

```
framebuffer_width=320  
framebuffer_height=240
```

Zapisz plik i uruchom ponownie Raspberry Pi. Teraz tekst wyświetlany na ekranie powinien być o wiele łatwiejszy do odczytania. Wokół ekranu będzie się znajdowała gruba czarna ramka. Aby zmodyfikować jej wielkość, zajrzyj do receptury 1.13.

Omówienie

Istnieje wiele tanich monitorów CCTV, które mogą się świetnie sprawdzić z Raspberry Pi, gdy ma ono udawać konsolę do gier w stylu retro (receptura 4.6). Niestety monitory tego typu zwykle charakteryzują się bardzo niską rozdzielcością.

Zobacz również

Pod adresem <https://learn.adafruit.com/using-a-mini-pal-ntsc-display-with-a-raspberry-pi> znajdziesz inny poradnik dotyczący korzystania z wyjścia zespołonego sygnału wideo Raspberry Pi.

Aby wyregulować obraz w przypadku korzystania z wyjścia HDMI, zajrzyj do receptury 1.13 i przyjrzyj się rysunkowi 1.10.

1.12. Korzystanie z całej pojemności karty SD

Problem

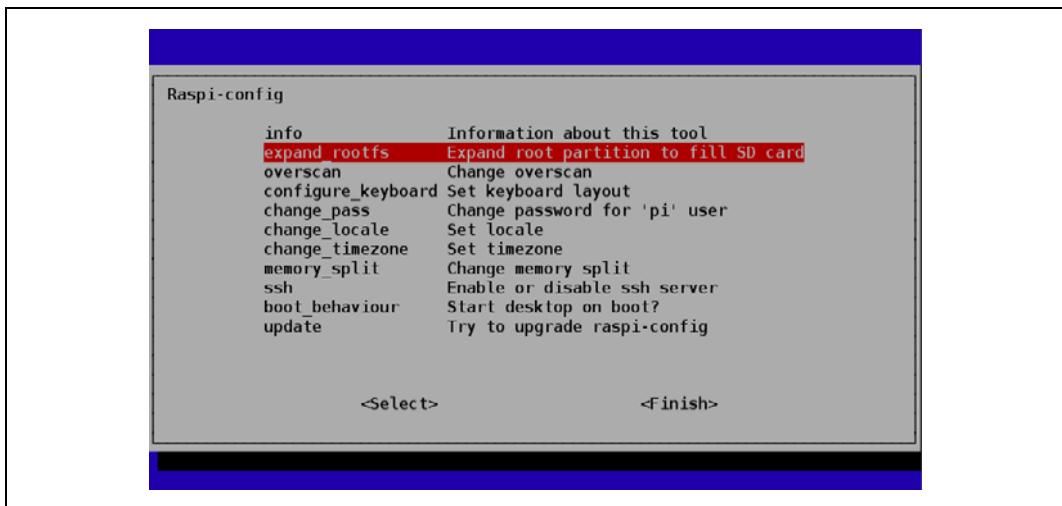
Gdy na karcie pamięci zapisywany jest system operacyjny, rozmiar tworzonej partycji jest determinowany rozmiarem obrazu, z którego go instalujemy. W wyniku tego dysponujemy bardzo małym obszarem pamięci przeznaczonym na nasze własne pliki.

Rozwiążanie

Aby rozwiązać ten problem, musisz uruchomić narzędzie `raspi-config`. Program ten jest wyświetlany automatycznie podczas pierwszego uruchomienia Raspberry Pi z nową kartą SD. Możesz go uruchomić, wpisując w wierszu poleceń:

```
$ sudo raspi-config
```

Z dostępnych opcji wybierz `expand_rootfs`, a następnie za pomocą klawiszy ustaw kursor na przycisku *Select* (zobacz rysunek 1.9).



Rysunek 1.9. Modyfikacja rozmiaru partycji głównej

Zostanie wyświetlony komunikat potwierdzający wykonanie operacji. Następnie musisz ponownie uruchomić Raspberry Pi, aby zobaczyć zmiany.

Teraz powinieneś mieć możliwość korzystania z całej dostępnej pojemności karty.

Omówienie

Zmiana rozmiaru głównej partycji jest czymś, co z pewnością warto zrobić. Umożliwi to tymczasowe przechowywanie plików na karcie. Jednak dokumenty i własne pliki lepiej przechowywać w pamięci USB, co pozwoli uniknąć konieczności przenoszenia ich przy okazji aktualizacji systemu operacyjnego.

Zobacz również

Więcej informacji na temat narzędzia `raspi-config` znajdziesz pod adresem http://elinux.org/RPi_raspi-config.

1.13. Zmiana rozmiaru obrazu wyświetlanyego na monitorze

Problem

Po podłączeniu Raspberry Pi do monitora po raz pierwszy może się okazać, że część wyświetlanego tekstu jest niemożliwa do odczytania, ponieważ znajduje się poza obszarem wyświetlonym na ekranie. Istnieje również prawdopodobieństwo, że wyświetlany obraz nie będzie zajmował całej powierzchni ekranu.

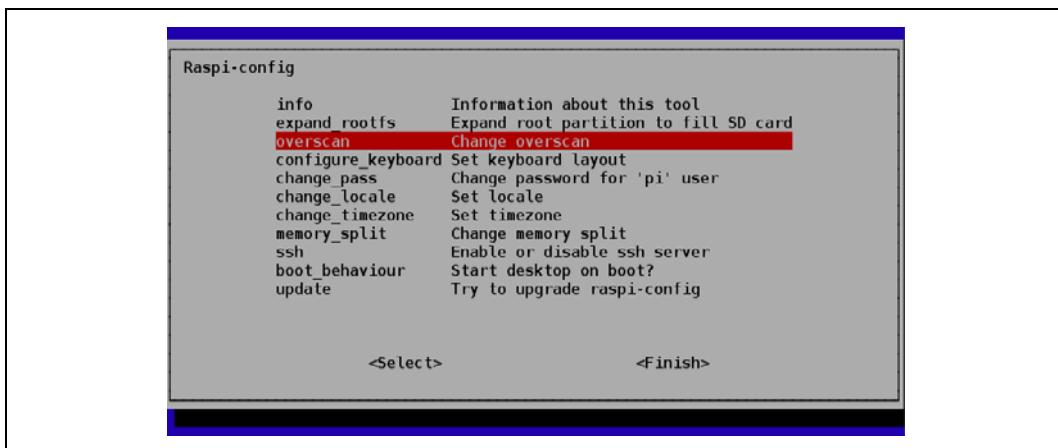
Rozwiążanie

Jeżeli wyświetlany tekst nie mieści się na ekranie, skorzystaj z narzędzia `raspi-config` i wyłącz funkcję `overscan`.

Uruchom `raspi-config`, wpisując w wierszu poleceń:

```
$ sudo raspi-config
```

Za pomocą klawiszy strzałek wybierz opcję `overscan`, a następnie ją wyłącz (zobacz rysunek 1.10).



Rysunek 1.10. Wyłączanie funkcji overscan

Jeżeli wokół obrazu wyświetlanego na ekranie widzisz grubą czarną ramkę, to możesz ją zmniejszyć (a prawdopodobnie całkowicie wyeliminować), edytując plik `/boot/config.txt`. W tym celu zastosuj poniższe polecenie.

```
$ sudo nano /boot/config.txt
```

Poszukaj w tym pliku sekcji dotyczącej funkcji overscan. Cztery linie, które musisz edytować, pokazano w środkowej części rysunku 1.11.

```
GNU nano 2.2.6          File: /boot/config.txt

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
# overscan.
#framebuffer_width=1280_
#framebuffer_height=720

AG Get Help  AO WriteOut  AR Read File  AY Prev Page  AK Cut Text  AC Cur Pos
AX Exit  AJ Justify  AW Where Is  AV Next Page  AU Uncut Text  AT To Spell
```

Rysunek 1.11. Dostosowywanie funkcji overscan

Najpierw usuń znaki `#` znajdujące się na początku tych wierszy. W ten sposób linie te przestaną być traktowane jak komentarz do kodu.

Następnie metodą prób i błędów zmieniaj wartości do momentu, w którym ekran będzie zajmował możliwie największą część powierzchni ekranu. Pamiętaj o tym, że wartości te muszą być ujemne. Na początek zmień je wszystkie na `-20`.

Omówienie

Konieczność wielokrotnego ponownego uruchamiania Raspberry Pi może być trochę uciążliwa. Na szczęście regulację tę będziesz musiał wykonać tylko raz. Wiele monitorów i telewizorów wyświetla obraz poprawnie bez konieczności wprowadzania opisanych tutaj modyfikacji.

Zobacz również

Więcej informacji na temat narzędzia `raspi-config` znajdziesz pod adresem http://elinux.org/RPi_raspi-config.

1.14. Maksymalizacja wydajności

Problem

Twoje Raspberry Pi działa zbyt wolno — chcesz je przetaktować w celu uzyskania większej wydajności.

Rozwiązanie

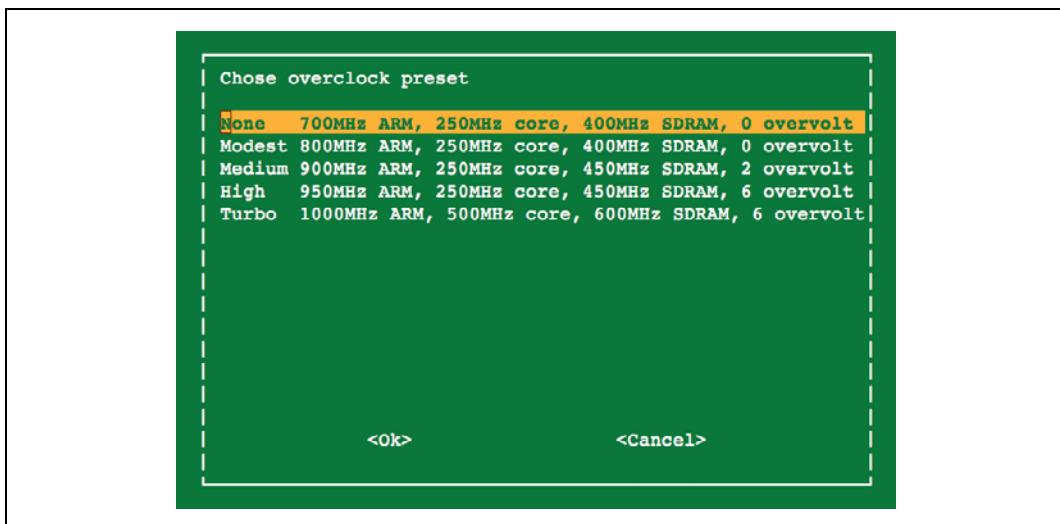
Możesz zwiększyć częstotliwość pracy zegara Raspberry Pi — dzięki temu będzie ono działało trochę szybciej. Po takim zabiegu urządzenie będzie pobierało trochę więcej prądu i wydzielało więcej ciepła (zobacz sekcja „Omówienie” poniżej).

Metoda przetaktowywania opisana tutaj jest metodą **dynamczną** — temperatura Raspberry Pi jest stale monitorowana, a częstotliwość zegara automatycznie obniżana, gdy układ rozgrzeje się zbyt mocno.

W celu zwiększenia częstotliwości pracy układu uruchom narzędzie `raspi-config`, wpisując następujące polecenie w Terminalu:

```
$ sudo raspi-config
```

W menu wybierz opcję `overclock`, a zostanie wyświetlona lista widoczna na rysunku 1.12.



Rysunek 1.12. Opcje przetaktowywania układu

Wybierz interesującą Cię częstotliwość pracy zegara. Jeżeli Raspberry Pi zacznie pracować niestabilnie albo będzie się niespodziewanie zawieszać, to niestety możesz być zmuszony do wybrania niższej częstotliwości pracy układu lub nawet wyłączenia funkcji przetaktowywania poprzez wybranie opcji None.

Omówienie

Przetaktowanie układu może spowodować wyraźny wzrost jego wydajności. Aby dokonać pomiaru wzrostu wydajności, skorzystałem z modelu B rev2. Raspberry Pi znajdowało się bez obudowy w pomieszczeniu, w którym panowała temperatura 15°C.

Do testowania wydajności zastosowałem poniższy skrypt napisany w języku Python. Program ten po prostu obciąża procesor i tak naprawdę nie odzwierciedla wydajności operacji takich jak zapis na karcie SD czy generowanie grafiki. Może on jednak dość dobrze służyć do testowania sprawności wykonywania obliczeń przez główny procesor, a więc możemy tym narzędziem sprawdzić efekt przetaktowania Raspberry Pi.

```
import time

def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

before_time = time.clock()
for i in range(1, 10000):
    factorial(200)
after_time = time.clock()

print(after_time - before_time)
```

Rezultaty przeprowadzonego testu przedstawiono w tabeli 1.2.

Tabela 1.2. Efekty przetaktowania

	Test szybkości	Prąd	Temperatura (°C)
700 MHz	15,8 sekundy	360 mA	27
1 GHz	10,5 sekundy	420 mA	30

Jak widzisz, wydajność wzrosła o 33%, jednak urządzenie zaczęło pobierać więcej prądu i wydzielać więcej ciepła.

W celu obniżenia temperatury głównego układu scalonego Raspberry Pi możesz nakleić na niego radiator wyposażony w specjalną taśmę samoprzylepną. Niestety nie wszystkie radiatory posiadają taśmę o właściwościach termoprzewodzących. Temperaturę układu z pewnością skuteczniej obniży dobrze wentylowana obudowa (receptura 1.2). Niektórzy użytkownicy zaprojektowali układy chłodzące Raspberry Pi za pomocą wody. Szczerze mówiąc, to już przesada.

Zobacz również

Więcej informacji na temat narzędzia `raspi-config` znajdziesz pod adresem http://elinux.org/RPi_raspi-config.

1.15. Zmiana hasła

Problem

Standardowym hasłem Raspberry Pi jest słowo *raspberry*. Chcesz je zmienić.

Rozwiązanie

Do zmiany hasła możesz skorzystać z narzędzia `raspi-config`. Uruchom je, wpisując w otwartą sesję Terminala (zobacz receptura 3.2) poniższe polecenie:

```
$ sudo raspi-config
```

Następnie w menu wybierz opcję `change_pass` — wyświetli się komunikat informujący Cię o tym, że za chwilę zostaniesz poproszony o podanie nowego hasła (zobacz rysunek 1.13).



Rysunek 1.13. Zmiana hasła

Zmiana hasła jest czynnością, która nie wymaga ponownego uruchamiania Raspberry Pi.

Omówienie

Hasło możesz również zmienić, korzystając bezpośrednio z otwartej sesji Terminala, za pomocą polecenia `passwd`:

```
$ sudo passwd
Changing password for pi.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

Zobacz również

Więcej informacji na temat narzędzia `raspi-config` znajdziesz pod adresem http://elinux.org/RPi_raspi-config.

1.16. Uruchamianie Raspberry Pi bezpośrednio w trybie graficznego interfejsu użytkownika

Problem

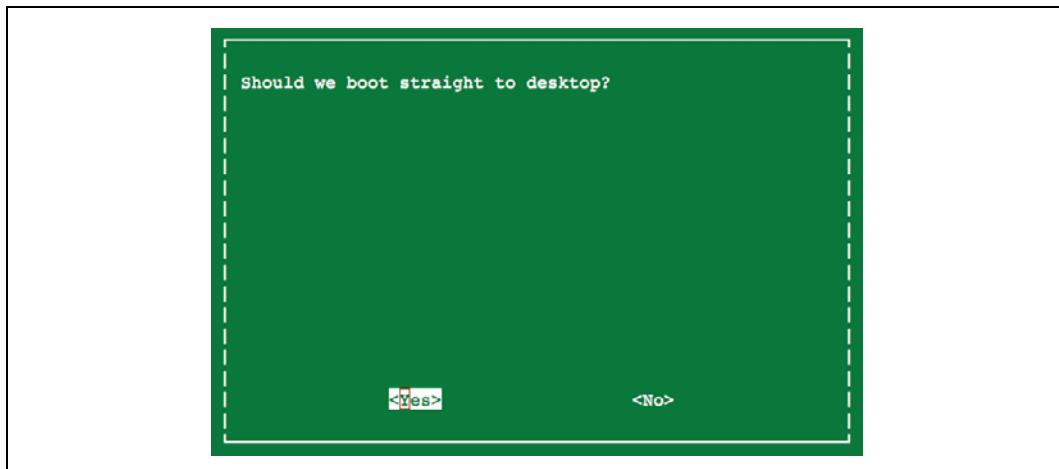
Za każdy razem gdy włączasz Raspberry Pi, musisz się zalogować, a następnie ręcznie uruchomić pulpit. Chcesz zautomatyzować ten proces.

Rozwiązanie

Proces rozruchu Raspberry Pi możesz skonfigurować za pomocą narzędzia `raspi-config`. Urządzenie może Cię automatycznie zalogować i wyświetlić pulpit. Uruchom to narzędzie, wpisując w otwartą sesję Terminala następujące polecenie:

```
$ sudo raspi-config
```

Następnie w menu wybierz opcję `boot_behaviour` — wyświetli się komunikat z pytaniem, czy podczas rozruchu Raspberry Pi pulpit ma być ładowany automatycznie (zobacz rysunek 1.14).



Rysunek 1.14. Automatyczne ładowanie pulpitu

Po zmianie opcji rozruchu zostaniesz poproszony o ponowne uruchomienie Raspberry Pi w celu zastosowania zmian.

Omówienie

Oczywiście włączenie automatycznego logowania się do pulpitu niesie ze sobą pewne zagrożenia. Raspberry Pi jest zwykle użytkowane w roli komputera osobistego i nie jest współdzielone przez kilku użytkowników, a więc zalety takiego rozwiązania przeważają nad jego wadami.

Zobacz również

Więcej informacji na temat narzędzia `raspi-config` znajdziesz pod adresem http://elinux.org/RPi_raspi-config.

1.17. Wyłączanie Raspberry Pi

Problem

Chcesz wyłączyć swoje Raspberry Pi.

Rozwiążanie

Kliknij czerwony przycisk *Logout* (wyloguj), który znajduje się w prawym dolnym rogu pulpitu. Wyświetla się opcje widoczne na rysunku 1.15.



Rysunek 1.15. Wyłączanie Raspberry Pi

Wyłącz

Wyłącza Raspberry Pi. W celu ponownego uruchomienia urządzenia będziesz musiał odłączyć je choćby na chwilę od zasilania.

Uruchom ponownie

Uruchamia ponownie urządzenie.

Wyloguj

Wylogowuje użytkownika i wyświetla ekran logowania w celu ponownego zalogowania użytkownika.

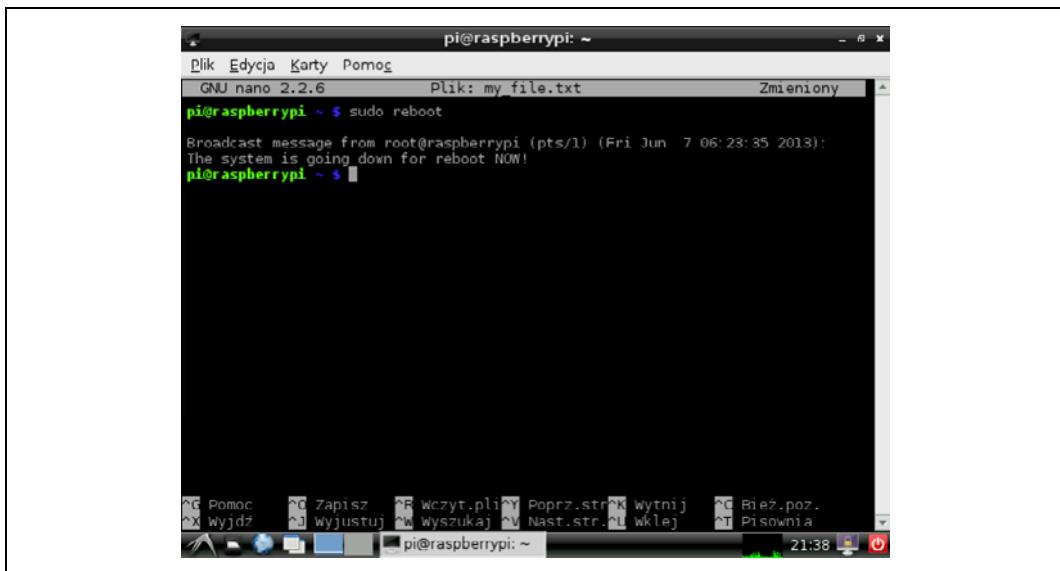
Anuluj

Pozwala na anulowanie kliknięcia czerwonego przycisku i dalsze korzystanie z Raspberry Pi.

Raspberry Pi może zostać uruchomione ponownie również z poziomu wiersza poleceń. Wystarczy wpisać w nim polecenie:

```
sudo halt
```

Czynność tę będziesz musiał wykonać po zainstalowaniu pewnych programów. Podczas ponownego uruchamiania Raspberry Pi zostanie wyświetlony komunikat informujący Cię o tym, że system jest właśnie zamazywany w celu ponownego uruchomienia (zobacz rysunek 1.16). Komunikat ten wynika z natury systemu Linux, gdyż jest to system, z którego może korzystać wielu użytkowników. System ostrzega wszystkich użytkowników korzystających z Raspberry Pi.



Rysunek 1.16. Wyłączanie Raspberry Pi z poziomu Terminala

Omówienie

Raspberry Pi po wyłączeniu nie zostanie automatycznie odłączone od prądu, jak to bywa w większości komputerów. Urządzenie przełącza się w tryb niskiego poboru prądu, chociaż nie jest ono energochłonne nawet podczas normalnej pracy. Raspberry Pi nie posiada żadnej sprzętowej kontroli nad zasilaczem.

Zobacz również

Tutaj możesz zakupić moduł automatycznie odłączający wyłączone Raspberry Pi od zasilania: <http://www.pi-supply.com/>.

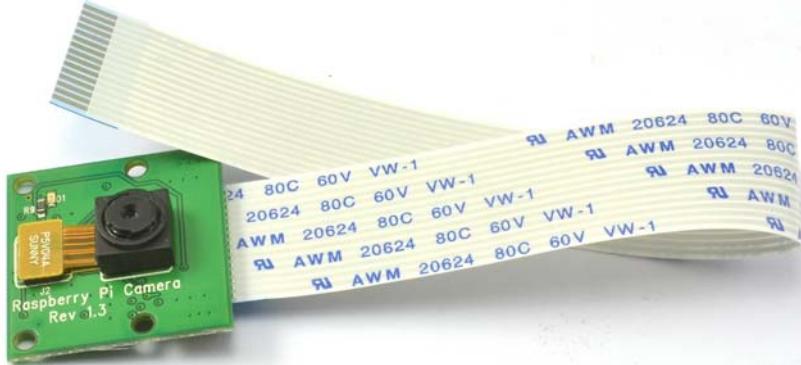
1.18. Instalacja modułu kamery

Problem

Chcesz podłączyć moduł kamery do Raspberry Pi (zobacz sekcja „Moduły” w dodatku A).

Rozwiążanie

Moduł kamery jest podłączany do Raspberry Pi za pomocą kabla taśmowego (zobacz rysunek 1.17).



Rysunek 1.17. Moduł kamery przeznaczonej do Raspberry Pi

Kabel ten należy podłączyć do złącza znajdującego się za gniazdem sieci Ethernet. W celu włożenia kabla pociagnij za dźwigienki znajdujące się po bokach złącza — w ten sposób zostanie ono otwarte. Następnie włóż kabel do złącza, tak aby styki znajdujące się na końcu kabla były zwrócone w kierunku przeciwnym do gniazda sieci Ethernet. Wciśnij dźwigienki, aby unieruchomić kabel w gnieździe (zobacz rysunek 1.18).



Rysunek 1.18. Podłączanie modułu kamery

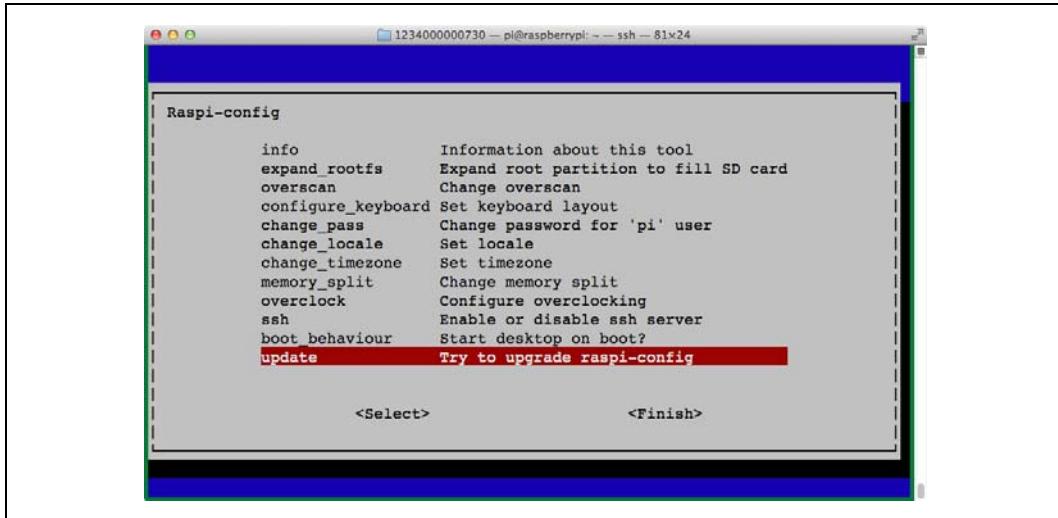


Na opakowaniu modułu kamery umieszczono napis informujący użytkownika o tym, że urządzenie jest wrażliwe na ładunki elektrostatyczne. Przed wyjęciem kamery z opakowania odprowadź ładunki zgromadzone na ciele, dotykając jakiegoś uziemionego przedmiotu, np. obudowy komputera.

Moduł kamery wymaga skonfigurowania oprogramowania. Najłatwiej to zrobić przy użyciu narzędzia `raspi-config` (receptura 1.12). W celu uruchomienia tego programu wpisz w otwartej sesji Terminala poniższe polecenie.

```
$ sudo raspi-config
```

Na ekranie zostanie wyświetlona lista opcji pokazana na rysunku 1.19.



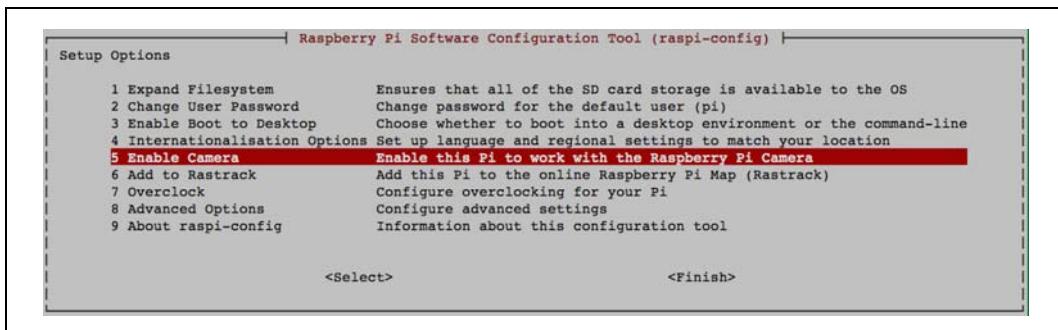
Rysunek 1.19. Narzędzie konfiguracyjne `raspi-config`

Jeżeli na liście nie ma opcji *Camera*, to znaczy, że musisz dokonać aktualizacji systemu operacyjnego, wpisując w sesji Terminala następujące polecenia (zobacz receptura 3.2):

```
$ sudo apt-get update  
$ sudo apt-get upgrade
```

Do przeprowadzenia aktualizacji niezbędne będzie połączenie z internetem. Wykonanie drugiego z powyższych poleceń może zająć kilka minut. Gdy operacje zostaną zakończone, uruchom ponownie Raspberry Pi (receptura 1.17).

Teraz po uruchomieniu narzędzia `raspi-config` powinieneś mieć do dyspozycji opcję włączającą obsługę kamery — *Enable Camera* (zobacz rysunek 1.20).



Rysunek 1.20. Zaktualizowane narzędzie konfiguracyjne `raspi-config`

W celu wykonania zdjęcia lub nagrania materiału wideo możesz skorzystać z poleceń `raspistill` lub `raspivid`.

W celu wykonania zdjęcia kamerą zastosuj polecenie `raspistill`:

```
$ raspistill -o image1.jpg
```

Najpierw przez pięć sekund na ekranie będzie wyświetlany podgląd obrazu, a następnie zostanie wykonane zdjęcie. Będzie ono zapisane w pliku o nazwie `image1.jpg` w bieżącym katalogu.

W celu nagrania materiału wideo zastosuj polecenie `raspivid`:

```
$ raspivid -o video.h264 -t 10000
```

Liczba na końcu polecenia określa czas trwania nagrania wyrażony w milisekundach. W tym przypadku nagranie będzie trwało 10 sekund.

Omówienie

Polecenia `raspistill` oraz `raspivid` można stosować z różnymi parametrami. Jeżeli wpiszesz któreś z tych poleceń bez parametrów, na ekranie zostanie wyświetlona pomoc informująca Cię o opcjach, jakie możesz zastosować.

Moduł kamery może wykonywać zdjęcia oraz rejestrować filmy w wysokiej rozdzielczości.

Kamera charakteryzuje się:

- matrycą o rozdzielczości 5 megapikseli,
- stałoogniskowym obiektywem o liczbie aperturowej $f/2$,
- rozdzielczością 1920×1080 pikseli,
- możliwością rejestrowania obrazu w rozdzielczości 1080p przy 30 klatkach na sekundę.

Do Raspberry Pi możesz podłączyć również kamerę internetową za pośrednictwem gniazda USB (zobacz receptura 4.5).

Zobacz również

Dokumentację kamery, w której omówiono polecenia `raspistill` i `raspivid`, znajdziesz na stronie <http://www.farnell.com/datasheets/1730389.pdf>.

Praca w sieci

2.0. Wprowadzenie

Konstrukcja Raspberry Pi pozwala na podłączenie tego urządzenia do internetu. Możliwość komunikacji sieciowej jest jedną z ważniejszych zalet tej konstrukcji. Umożliwia ona wiele nowych zastosowań, takich jak automatyzacja domu, obsługa sieci Web, monitoring sieciowy itd.

Połączenie sieciowe może być realizowane poprzez kabel Ethernet (w przypadku modelu B). Do Raspberry Pi możesz również podłączyć za pośrednictwem gniazda USB moduł Wi-Fi.

Dzięki połączeniu sieciowemu możliwe jest zdalne sterowanie Raspberry Pi z innego komputera. Przydaje się to zwłaszcza w sytuacji, w której nie mamy bezpośredniego dostępu do urządzenia lub nie podłączono do niego klawiatury, myszy ani monitora.

W tym rozdziale omówimy receptury pozwalające na podłączenie Raspberry Pi do internetu i zdalne sterowanie nim za pośrednictwem sieci.

2.1. Łączenie z siecią przewodową

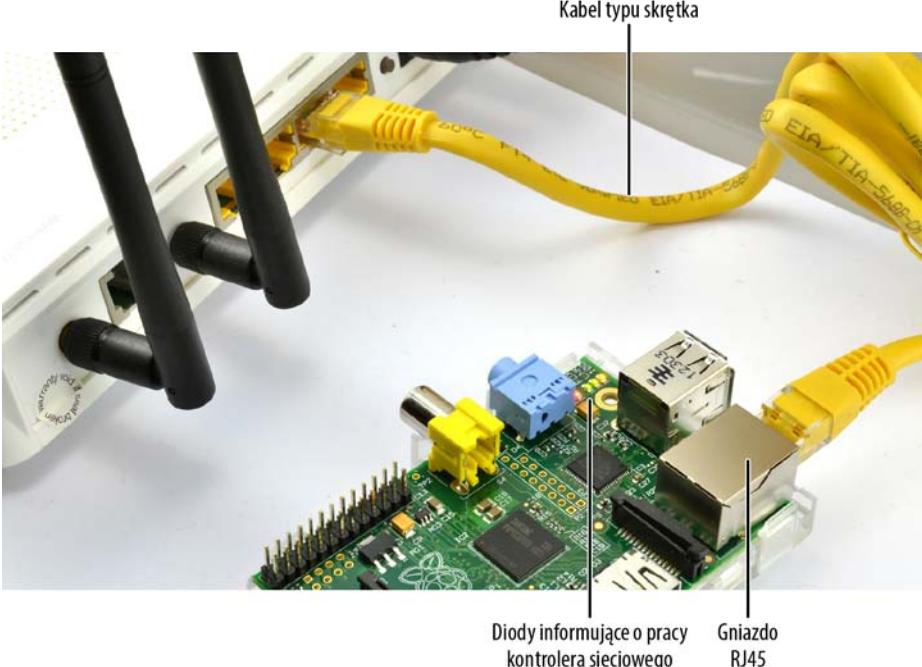
Problem

Chcesz podłączyć Raspberry Pi do internetu za pomocą kabla sieciowego.

Rozwiązanie

Jeżeli masz Raspberry Pi model A, które nie posiada gniazda RJ45 sieci Ethernet, to najprościej podłączyć je do internetu za pośrednictwem kontrolera sieci bezprzewodowej wyposażonego w gniazdo USB (zobacz receptura 2.5).

Jeżeli posiadasz Raspberry Pi model B, to podłącz do niego kabel typu skrętka zakończony wtyczką RJ45 (końcówkę przewodu włóż do gniazda sieci Ethernet). Drugi koniec tego samego kabla podłącz do domowego routera lub koncentratora (zobacz rysunek 2.1).



Rysunek 2.1. Podłączanie Raspberry Pi do koncentratora sieciowego

Diody informujące o pracy kontrolera sieciowego Raspberry Pi powinny zacząć migać, gdy urządzenie zacznie się łączyć z Twoją siecią.

Omówienie

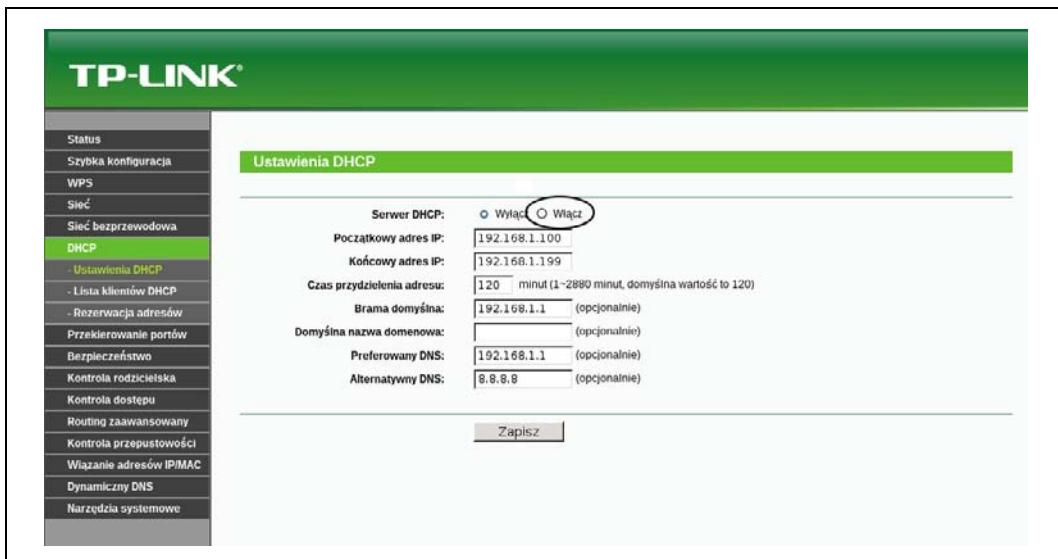
Raspbian i Occidental to dystrybucje systemów operacyjnych, które są skonfigurowane tak, aby były w stanie połączyć się z dowolną siecią korzystającą z protokołu dynamicznej konfiguracji węzłów (DHCP). Tak naprawdę większość dystrybucji systemów operacyjnych Raspberry Pi będzie skonfigurowana właśnie w ten sposób. Jeżeli w sieci włączono obsługę DHCP, to urządzeniu zostanie automatycznie przypisany adres IP.

Jeżeli diody informujące o pracy kontrolera sieciowego Raspberry Pi nie zapalą się po połączeniu urządzenia z koncentratorem, sprawdź, czy nie podłączyłeś wtyczki RJ45 do gniazda koncentratora oznaczonego napisem *Uplink*.

W przypadku gdy diody LED migają, ale nie możesz się połączyć z internetem za pośrednictwem przeglądarki zainstalowanej na Raspberry Pi, sprawdź, czy włączono obsługę DHCP w konsoli zarządzającej routerem. Poszukaj opcji podobnej do tej, którą pokazano na rysunku 2.2.

Zobacz również

Jeżeli chcesz połączyć Raspberry Pi z siecią bezprzewodową, to zobacz recepturę 2.5.



Rysunek 2.2. Włączanie obsługi DHCP w sieci domowej

2.2. Ustalanie własnego adresu IP

Problem

W celu nawiązania komunikacji z Raspberry Pi potrzebny Ci jest adres IP tego urządzenia. Jest to informacja przydatna, gdy Pi pełni rolę serwera sieci Web lub serwera wymiany plików, a także gdy chcesz sterować nim zdalnie za pośrednictwem SSH (receptura 2.7) lub VNC (receptura 2.8).

Adres IP jest ciągiem czterech liczb oddzielonych od siebie kropkami. Służy do identyfikacji komputera w sieci.

Rozwiążanie

Aby ustalić adres IP Raspberry Pi, wpisz w oknie Terminala poniższe polecenie:

```
$ sudo ifconfig
```

```
eth0      Link encap:Ethernet HWaddr b8:27:eb:d5:f4:8f
          inet addr:192.168.1.16 Bcast:192.168.255.255 Mask:255.255.0.0
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:1114 errors:0 dropped:1 overruns:0 frame:0
              TX packets:1173 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:76957 (75.1 KiB) TX bytes:479753 (468.5 KiB)
lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
              UP LOOPBACK RUNNING MTU:16436 Metric:1
              RX packets:0 errors:0 dropped:0 overruns:0 frame:0
              TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
```

```
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
wlan0 Link encap:Ethernet HWaddr 00:0f:53:a0:04:57
      inet addr:192.168.1.13 Bcast:192.168.255.255 Mask:255.255.0.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:38 errors:0 dropped:0 overruns:0 frame:0
        TX packets:28 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:6661 (6.5 KiB) TX bytes:6377 (6.2 KiB)
```

Przyglądając się temu, co wyświetlono na ekranie, po uruchomieniu polecenia `ifconfig` możesz stwierdzić, że danemu egzemplarzowi Raspberry Pi w sieci przewodowej (`eth0`) przypisano adres IP 192.168.1.16, a w pierwszej sieci bezprzewodowej (`wlan0`) adres 192.168.1.13.

Omówienie

Jak widzisz, Raspberry Pi może mieć kilka adresów IP (urządzenie podczas pracy w różnych sieciach może mieć różne adresy). Jeżeli Raspberry Pi będzie jednocześnie podłączone do sieci przewodowej oraz bezprzewodowej, to będzie miało dwa adresy IP. Zwykle jednak będziesz łączył je tylko z jedną siecią, a nie symultanicznie z dwoma.

Innym sposobem ustalenia adresu IP Raspberry Pi jest zalogowanie się do konsoli zarządzającej routerem i przejście to tabeli zawierającej adresy IP, która znajduje się w sekcji LAN. Na liście urządzeń powinieneś odnaleźć nazwę `raspberrypi`, obok której będzie podany adres IP.

Zobacz również

Więcej informacji na temat adresu IP znajdziesz w Wikipedii — http://pl.wikipedia.org/wiki/Adres_IP.

2.3. Łączenie z siecią przewodową

Problem

Chcesz przypisać określony adres IP na stałe.

Rozwiążanie

W celu określenia adresu IP (niezależnie od tego, czy korzystasz z sieci przewodowej, czy bezprzewodowej) musisz zmodyfikować plik konfiguracyjny `/etc/network/interfaces`.

Plik ten możesz wyświetlić, korzystając z poniższego polecenia:

```
$ more /etc/network/interfaces
```

Będzie on wyglądał w następujący sposób:

```
iface lo inet loopback

iface eth0 inet dhcp

iface wlan0 inet dhcp
  wpa-ssid "identyfikatorssid"
  wpa-psk "hasło"
```

Aby edytować ten plik, skorzystaj z następującego polecenia:

```
$ sudo nano /etc/network/interfaces
```

Jeżeli korzystasz z sieci przewodowej, zmodyfikuj sekcję `eth0`, a jeżeli używasz sieci bezprzewodowej — `wlan`.

Najpierw powinieneś zdecydować się na adres IP, którego chcesz używać. Wybierz adres, który nie jest używany przez inne urządzenia w sieci, a jednocześnie znajduje się w puli adresów IP obsługiwanych przez Twój router.

Edytuj zawartość pliku i zmień słowo `dhcp` na `static`, a następnie dodaj do pliku poniższe linie:

```
address 192.168.1.16
netmask 255.255.255.0
gateway 192.168.1.1
```

Taka modyfikacja pliku spowoduje przypisanie adresu IP 192.168.1.16 do interfejsu `eth0`.

```
iface lo inet loopback

iface eth0 inet static
    address 192.168.1.16
    netmask 255.255.255.0
    gateway 192.168.1.1

iface wlan0 inet dhcp
    wpa-ssid "identyfikatorssid"
    wpa-psk "hasło"
```

W większości sieci maska powinna mieć adres 255.255.255.0, a brama adres IP routera, którego używasz. Logując się do konsoli administracyjnej, korzystałeś już z tego adresu.

Po zmodyfikowaniu pliku zapisz go i uruchom ponownie Raspberry Pi.

Omówienie

Wewnętrzne adresy IP zwykle mają postać 192.168.1.16 — urządzenia podłączone do tej samej sieci będą posiadały adresy IP różniące się ostatnią liczbą. Inną popularną konwencją jest przypisywanie komputerom wewnętrznych adresów IP typu 10.0.0.16.

Zobacz również

Więcej informacji na temat adresu IP znajdziesz w Wikipedii — http://pl.wikipedia.org/wiki/Adres_IP.

2.4. Zmiana nazwy, pod którą Raspberry Pi jest widoczne w sieci

Problem

Chcesz zmienić nazwę, pod którą Raspberry Pi jest widoczne w sieci. Nie chcesz korzystać z domyślnej nazwy „raspberrypi”.

Rozwiązanie

Zmiana nazwy jest dość prosta. Musisz zmodyfikować tylko dwa pliki.

Najpierw edytuj plik `/etc/hostname`. Możesz to zrobić, otwierając okno Terminala i wpisując w nim polecenie:

```
$ sudo nano /etc/hostname
```

Zastąp „raspberrypi” wybraną przez siebie nazwą. Powinna ona być jednym słowem i nie może zawierać znaków specjalnych ani przestankowych. Dotyczy to również znaku podkreślenia `_`.

Następnie otwórz w edytorze plik `/etc/hosts` za pomocą polecenia:

```
$ sudo nano /etc/hosts
```

Plik powinien wyglądać mniej więcej tak:

```
127.0.0.1      localhost
::1            localhost ip6-localhost ip6-loopback
fe00::0        ip6-localnet
ff00::0        ip6-mcastprefix
ff02::1        ip6-allnodes
ff02::2        ip6-allrouters

127.0.1.1      raspberrypi
```

Na końcu pliku zmień starą nazwę (`raspberrypi`) na nową.

Uruchom ponownie Raspberry Pi. Teraz będzie ono widoczne w sieci pod nową nazwą.

Omówienie

Zmiana adresu Raspberry Pi może się okazać bardzo praktyczna, zwłaszcza gdy w jednej sieci pracuje kilka takich urządzeń.

Zobacz również

Zobacz również recepturę 2.3. Opisano w niej zmianę adresu IP Raspberry Pi.

2.5. Nawiązywanie połączenia z siecią bezprzewodową

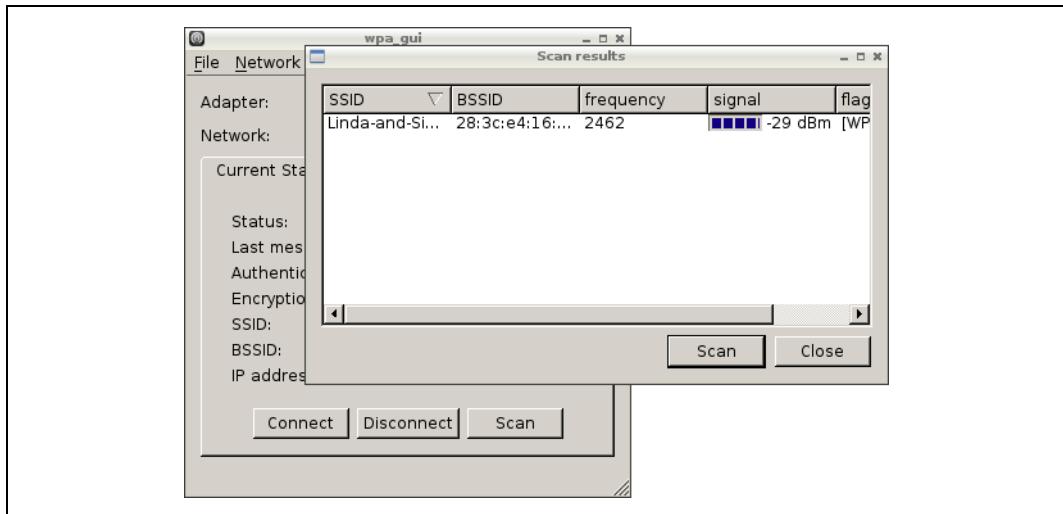
Problem

Chcesz się połączyć z internetem za pośrednictwem karty sieci bezprzewodowej wyposażonej w interfejs USB.

Rozwiązanie

Połączenie z siecią bezprzewodową jest bardzo łatwe do nawiązania, jeżeli korzystasz z aktualnej wersji systemu Raspbian, która posiada narzędzie **WiFi Config**. Skrót do tego narzędzia będzie się znajdował na pulpicie. Jeżeli korzystasz ze starej wersji systemu, zaktualizuj go (zobacz receptury od 1.4 do 1.8).

Do jednego z gniazd USB Raspberry Pi podłącz kompatybilny kontroler sieci bezprzewodowej (większość z nich jest kompatybilna). Następnie uruchom narzędzie WiFi Config (zobacz rysunek 2.3) i kliknij przycisk *Scan*, aby wyszukać dostępne sieci bezprzewodowe. Kliknij dwukrotnie sieć (symbolizującą tak naprawdę Twój domowy router), z którą chcesz nawiązać połączenie. Następnie w odpowiednim polu wpisz hasło.



Rysunek 2.3. Narzędzie WiFi Config

Na koniec kliknij przycisk *Connect* — uzyskasz połączenie z siecią.

Omówienie

Kontrolery sieci Wi-Fi pobierają dużo prądu. Jeżeli Raspberry Pi niespodziewanie się resetuje lub uruchamia się w sposób nieprawidłowy, to musisz się liczyć z koniecznością zakupu lepszego zasilacza. Poszukaj takiego, który jest w stanie dostarczyć prąd o natężeniu przynajmniej 1,5 A.

Jeżeli jednocześnie podłączasz do Raspberry Pi mysz oraz klawiaturę, może brakować Ci wolnych portów USB. Rozwiążaniem tego problemu jest zastosowanie koncentratora USB. Zakup koncentratora wyposażonego we własny zasilacz rozwiąże również problemy związane z zasilaniem.

Zobacz również

Receptura 2.1 opisuje podłączanie Raspberry Pi do sieci przewodowej. Listę kontrolerów sieci bezprzewodowej zgodnych z Raspberry Pi znajdziesz pod adresem http://www.elinux.org/RPi_VerifiedPeripherals.

2.6. Korzystanie z kabla konsolowego

Problem

Nie możesz skorzystać z połączenia sieciowego, ale mimo to chcesz sterować Raspberry Pi zdalnie z innego komputera.

Rozwiązanie

W celu uzyskania połączenia z Raspberry Pi skorzystaj z **kabla konsolowego**.

Kable konsolowe są przydatne, gdy Raspberry Pi pracuje samodzielnie bez klawiatury, myszy i monitora. Kabel konsolowy widoczny na rysunku 2.4 możesz zakupić za pośrednictwem firmy Adafruit.



Rysunek 2.4. Kabel konsolowy

Kabel należy podłączyć w następujący sposób:

1. Podłącz czerwony przewód do dolnego prawego pinu, tak jak pokazano na rysunku 2.4.
2. Pozostaw jeden pin wolny i do kolejnego podłącz czarny przewód (uziemienie).
3. Podłącz biały przewód po lewej stronie czarnego przewodu.
4. Podłącz zielony przewód po lewej stronie białego przewodu.

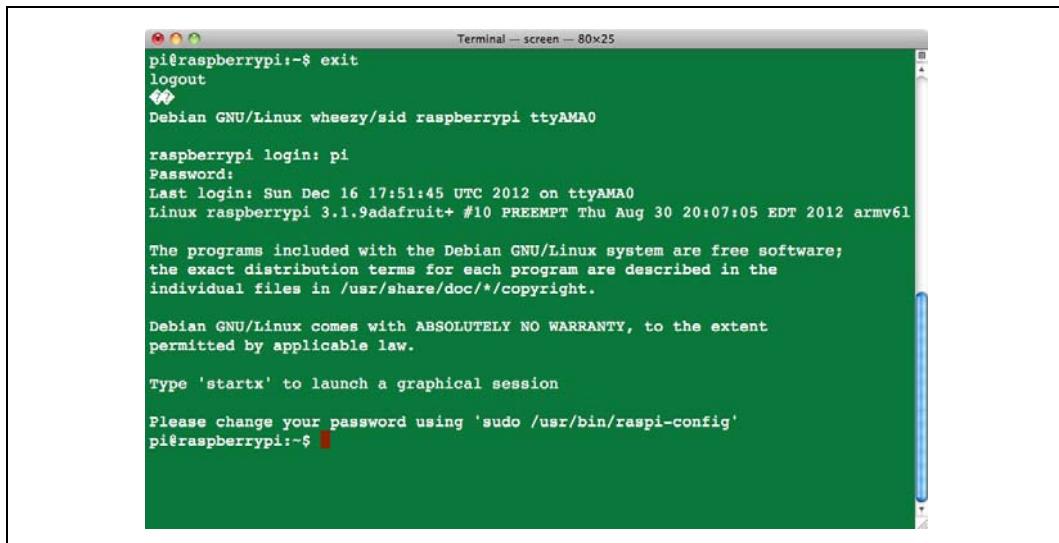
Zauważ, że przewód dostarcza do Raspberry Pi prąd o napięciu 5 V. Możesz zasilać w ten sposób samo Raspberry Pi bez urządzeń peryferyjnych.

Jeżeli używasz systemu Mac OS X lub Windows, to w celu skorzystania z kabla będziesz musiał zainstalować odpowiednie sterowniki: http://www.prolific.com.tw/US>ShowProduct.aspx?p_id=225&pcid=41 (Windows) i <http://sourceforge.net/projects/osx-pl2303/> (OS X).

Aby połączyć się z Raspberry Pi z komputera typu Mac, uruchom Terminal i wprowadź następujące polecenie:

```
$ screen /dev/cu.PL2303-00001004 115200
```

Dokładna nazwa urządzenia będzie inna, jednak wcisnięcie klawisza *Tab* po frazie *cu.P* spowoduje automatyczne uzupełnienie wpisywanej nazwy. Po uzyskaniu połączenia wcisnij klawisz *Enter*. Na ekranie pojawi się zapytanie o nazwę użytkownika i hasło (zobacz rysunek 2.5). Domyślną nazwą użytkownika jest *pi*, a hasłem *raspberry*.



Rysunek 2.5. Logowanie się za pośrednictwem kabla konsolowego

W celu połączenia Raspberry Pi z komputerem pracującym pod kontrolą systemu Windows będziesz musiał zastosować program Putty. Możesz go pobrać ze strony <http://www.putty.org/>.

Po uruchomieniu tego programu zmień typ połączenia na szeregowego (Serial), a prędkość transmisji danych ustaw na 115200. Być może będziesz musiał ustawić również numer używanego portu (Serial line). Prawdopodobnie będzie to port COM7. Numer portu możesz sprawdzić w menedżerze urządzeń.

Po kliknięciu przycisku *Open* i wcisnięciu klawisza *Enter* powinna się otworzyć sesja Terminala z zapytaniem o nazwę użytkownika i hasło.

Omówienie

Kabel konsolowy pozwala na zdalne korzystanie z Raspberry Pi w wygodny i prosty sposób.

Zobacz również

Na stronie internetowej firmy Adafruit (<https://learn.adafruit.com/adafruits-raspberry-pi-lesson-5-using-a-console-cable>), która sprzedaje kable konsolowe, znajduje się obszerny zbiór informacji na ich temat.

2.7. Zdalne sterowanie Raspberry Pi za pomocą protokołu SSH

Problem

Chcesz się zdalnie połączyć z Raspberry Pi (z innego komputera) za pomocą protokołu SSH (ang. *Secure Shell*).

Rozwiązanie

Obsługę SSH najłatwiej skonfigurować za pomocą narzędzia `raspi-config` — tego samego, które zostaje wyświetcone na ekranie po pierwszym uruchomieniu Raspbiana. Narzędzie to możesz uruchomić również później za pomocą następującego polecenia, które należy wpisać w Terminalu:

```
$ sudo raspi-config
```

Za pomocą klawiszy kierunkowych ustaw kursor niżej i włącz obsługę SSH.

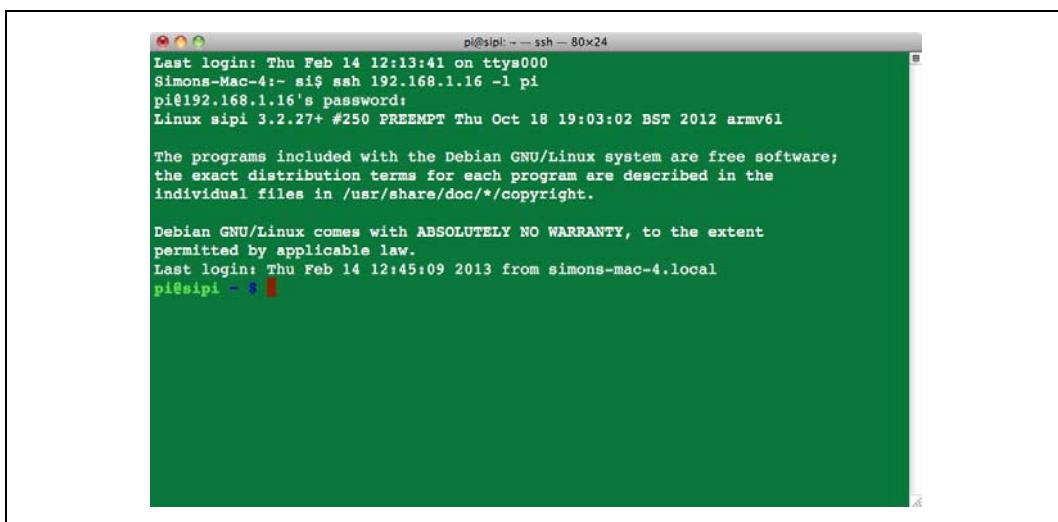


W nowszych wersjach Raspbiana SSH jest włączone automatycznie. Wtedy w opisywanym narzędziu nie ma opcji pozwalającej na wyłączenie tego protokołu.

Jeżeli na komputerze, za pośrednictwem którego chcesz się połączyć z Raspberry Pi, zainstalowano system operacyjny Mac OS X lub Linux, to wystarczy, że otworzysz okno Terminala i wpiszesz w nim polecenie:

```
$ ssh 192.168.1.12 -l pi
```

Oczywiście w poleceniu tym musisz podać faktyczny adres IP Twojego Raspberry Pi (zobacz receptura 2.2). Zostaniesz poproszony o podanie nazwy użytkownika i hasła, a następnie zalogowany do urządzenia (zobacz rysunek 2.6).



Rysunek 2.6. Logowanie za pośrednictwem SSH

W systemie Windows sesję SSH będziesz musiał otworzyć w programie Putty (zobacz rysunek 2.5).

Omówienie

Podczas zdalnego łączenia się z komputerami często korzysta się z protokołu SSH. Za jego pośrednictwem możesz stosować wszystkie polecenia obsługiwane przez Raspberry Pi. Połączenie wykonane za pomocą protokołu SSH jest szyfrowane.

Być może jego jedyną wadą jest to, że jest to interfejs tekstowy, a nie graficzny. Jeżeli potrzebujesz dostępu do pulpitu Raspberry Pi, musisz skorzystać z programu VNC (zobacz receptura 2.8).

Zobacz również

Zobacz również poradnik zamieszczony na stronie firmy Adafruit — <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh/overview>.

2.8. Sterowanie Raspberry Pi za pomocą VNC

Problem

Chcesz mieć dostęp do interfejsu graficznego Raspberry Pi z komputera typu PC lub Macintosh.

Rozwiązanie

Zainstaluj serwer VNC (ang. *Virtual Network Connection*).

Otwórz sesję Terminala (lub SSH) na Raspberry Pi i wprowadź poniższe komendy:

```
$ sudo apt-get update  
$ sudo apt-get install tightvncserver
```

Po zainstalowaniu serwera VNC uruchom go za pomocą polecenia:

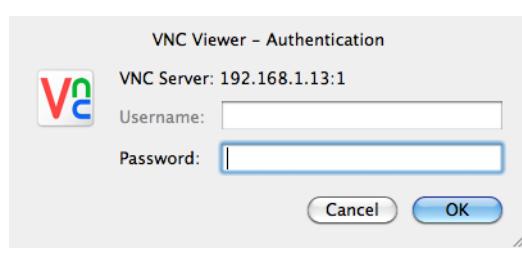
```
$ vncserver :1
```

Podczas pierwszego uruchomienia tego programu zostaniesz poproszony o podanie hasła, jakie będzie musiała wpisać osoba próbująca uzyskać zdalny dostęp do Raspberry Pi.

Aby połączyć się z Raspberry Pi, będziesz musiał zainstalować na komputerze klienta VNC. Popularną aplikacją tego typu jest *RealVNC*. Program ten dobrze współpracuje z *tightvnc*.

Po uruchomieniu programu będącego klientem VNC na komputerze typu PC lub Macintosh zostaniesz poproszony o podanie adresu IP komputera pełniącego funkcję serwera VNC, z którym chcesz się połączyć. W naszym przypadku będzie to adres IP Raspberry Pi. Po adresie IP wpisz :1 w celu połączenia się z ekranem o numerze 1.

Następnie zostaniesz poproszony o podanie hasła. Wpisz tutaj hasło, które podałeś po instalacji *tightvncserver* (zobacz rysunek 2.7). Hasło to może być inne niż główne hasło Twojego Raspberry Pi.



Rysunek 2.7. Logowanie się za pośrednictwem VNC

Omówienie

Za pośrednictwem SSH możesz zrobić większość rzeczy, czasem jednak może Ci się również przydać dostęp do graficznego interfejsu Raspberry Pi.

Jeżeli chcesz, aby serwer VNC był włączany przy każdym uruchomieniu Raspberry Pi, skorzystaj z poniższych instrukcji.

```
$ cd /home/pi  
$ cd .config  
$ mkdir autostart  
$ cd autostart  
$ nano tightvnc.desktop
```

W oknie edytora wpisz poniższy kod.

```
[Desktop Entry]  
Type=Application  
Name=TightVNC  
Exec=vncserver :1  
StartupNotify=false
```

Serwer VNC będzie teraz uruchamiany automatycznie, o ile Raspberry Pi będzie skonfigurowane tak, aby podczas uruchamiania dochodziło do automatycznego zalogowania i uruchomienia graficznego interfejsu użytkownika.

Zobacz również

Möżesz się również zapoznać z poradnikiem zamieszczonym na stronie firmy Adafruit — <https://learn.adafruit.com/adafruit-raspberry-pi-lesson-7-remote-control-with-vnc>.

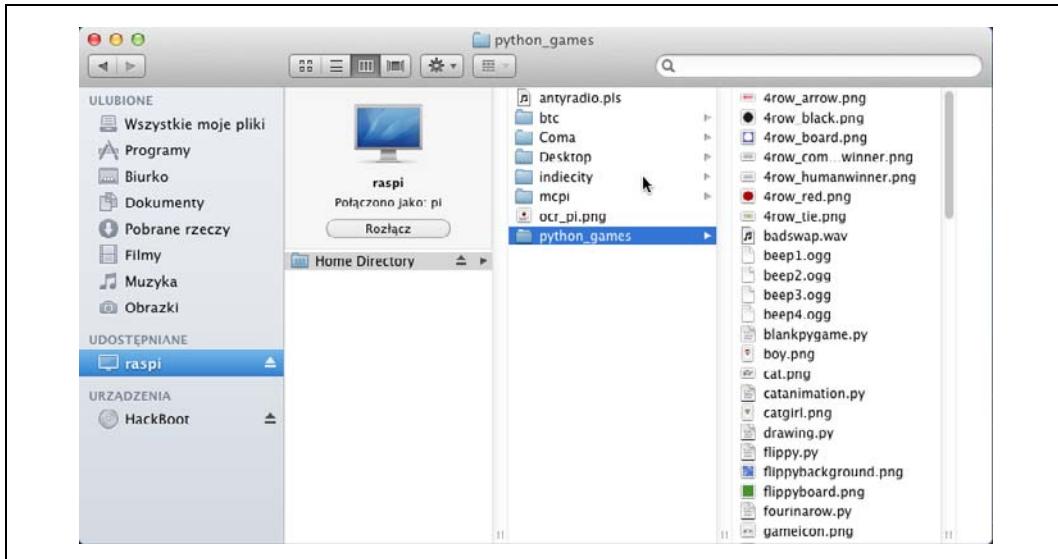
2.9. Udostępnianie plików w sieci komputerów Macintosh

Problem

Chcesz, żeby Raspberry Pi pojawiało się na liście komputerów w Finderze — pozwoli Ci to na uzyskanie połączenia i przeglądanie plików.

Rozwiążanie

Finder, będący częścią systemu Mac OS X, pozwala na przeglądanie plików poprzez sieć (zobacz rysunek 2.8). Aby skorzystać z tej możliwości, musisz odpowiednio skonfigurować swoje Raspberry Pi.



Rysunek 2.8. Raspberry Pi widoczne w Finderze (Mac OS X)

Zacznij od ustalenia adresu IP swojego Raspberry Pi (receptura 2.2).

Następnie zainstaluj na swoim Raspberry Pi narzędzie netatalk za pomocą polecenia:

```
$ sudo apt-get install netatalk
```

W dalszej kolejności na komputerze wybierz opcję *Połącz z serwerem* w menu *Idź*. Jako adres serwera wprowadź **afp://192.168.1.16** (podano tutaj przykładowy adres IP, w rzeczywistości musisz wprowadzić adres IP swojego Raspberry Pi). Następnie kliknij przycisk *Połącz*. Teraz powinien się wyświetlić komunikat z prośbą o podanie nazwy użytkownika i hasła. Komunikat ten nie wyświetla się, dopóki Raspberry Pi nie zostanie ponownie uruchomione.

Podaj nazwę użytkownika (domyślnie *pi*) i hasło (domyślnie *raspberry*). Finder wyświetli zawartość folderu *home* znajdującego się na karcie pamięci Raspberry Pi.

Teraz musisz dokonać jeszcze kilku zmian ustawień Raspberry Pi.

```
$ sudo apt-get install avahi-daemon  
$ sudo update-rc.d avahi-daemon defaults
```

Następnie wpisz polecenie:

```
$ sudo nano /etc/avahi/services/afpd.service
```

W pliku umieść poniższy kod:

```
<?xml version="1.0" standalone='no'?><!--*-nxml-*-->  
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">  
<service-group>  
    <name replace-wildcards="yes">%h</name>
```

```
<service>
  <type>_afpovertcp._tcp</type>
  <port>548</port>
</service>
</service-group>
```

Aby uruchomić proces *daemon*, wpisz polecenie:

```
$ sudo /etc/init.d/avahi-daemon restart
```

Wróć do swojego Macintosha. Teraz Raspberry Pi powinno być widoczne w oknie Findera.

Omówienie

Możliwość łatwego przesyłania plików pomiędzy Macintoshem a Raspberry Pi jest bardzo przydatna. Możesz korzystać z plików znajdujących się na Raspberry Pi i nie musisz w tym celu podłączać do niego klawiatury, myszy ani monitora.

Pliki znajdujące się w pamięci Raspberry Pi możesz otwierać tak, jakby były one w pamięci Twojego komputera. Możesz je edytować za pomocą programu TextMate lub dowolnego innego edytora tekstowego.

Jeżeli korzystasz z systemu Windows lub Linux, możesz skonfigurować Raspberry Pi, tak by pracowało ono jako serwer NAS (ang. *Network Attached Storage*) — zobacz receptura 2.11.

Zobacz również

Powyższe instrukcje napisano w oparciu o artykuł <http://4dc5.com/2012/06/12/setting-up-vnc-on-raspberry-pi-for-mac-access/>, który odwołuje się do książki *Getting Started with Raspberry Pi* napisanej przez Matta Richardsona i Shawna Wallace'a (<http://shop.oreilly.com/product/0636920023371.do>).

2.10. Udostępnianie ekranu Raspberry Pi na komputerze Macintosh

Problem

Zainstalowałeś VNC, ale chcesz udostępniać ekran Raspberry Pi, tak jakby był to kolejny komputer Macintosh w Twojej sieci.

Rozwiążanie

Najpierw wykonaj czynności opisane w recepturze 2.8 — zainstaluj VNC. Musisz również skorzystać z receptury 2.9.

Później uruchom polecenie:

```
$ sudo nano /etc/avahi/services/rfb.service
```

a następnie w edytorze umieść poniższy kod:

```
<?xml version="1.0" standalone='no'?>
<!DOCTYPE service-group SYSTEM "avahi-service.dtd">
<service-group>
  <name replace-wildcards="yes">%h</name>
```

```
<service>
  <type>_rfb._tcp</type>
  <port>5901</port>
</service>
</service-group>
```

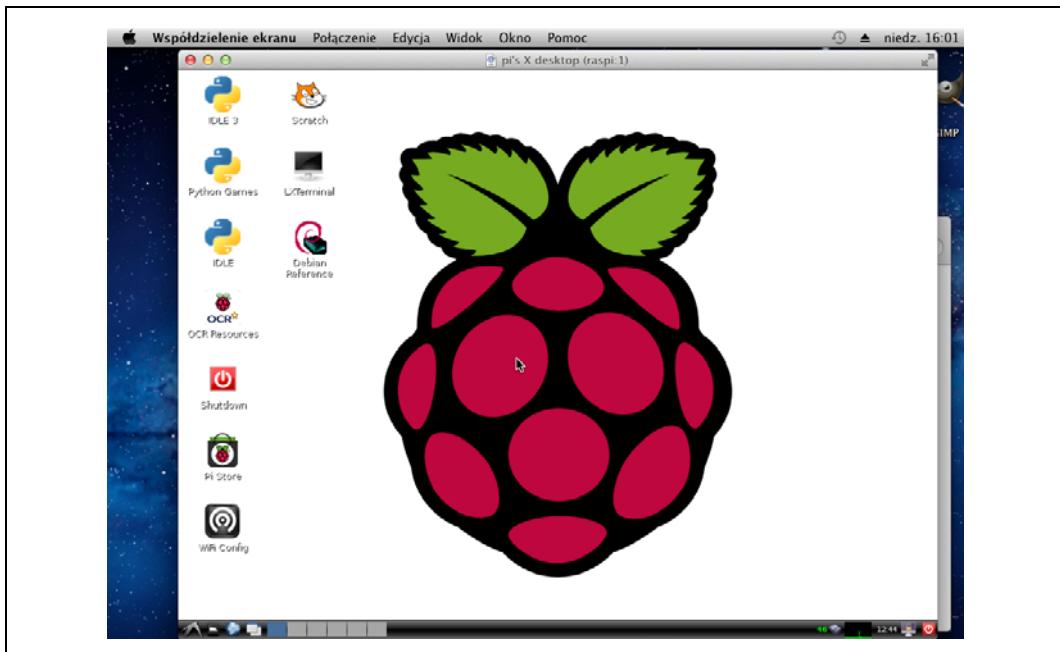
Na koniec uruchom polecenie:

```
$ sudo /etc/init.d/avahi-daemon restart
```

Następnie na komputerze wybierz opcję *Połącz z serwerem* w menu *Idź*. Jako adres serwera wprowadź **vnc://192.168.1.16:5901** (podano tutaj przykładowy adres IP, w rzeczywistości musisz wprowadzić adres IP swojego Raspberry Pi i port, na którym działa VNC, jak na rysunku 2.9). Kliknij przycisk *Połącz*. Teraz powinieneś się wyświetlić komunikat z prośbą o podanie hasła, które ustaliłeś w recepturze 2.8, podczas konfiguracji serwera *tightvnc*. Jeśli wszystko pójdzie dobrze, zobaczysz ekran przedstawiony na rysunku 2.10.



Rysunek 2.9. Łączenie się ze zdalnym ekranem Raspberry Pi w Finderze



Rysunek 2.10. Wygląd zdalnego pulpitu Raspberry Pi po udanym zalogowaniu się

Omówienie

Jeżeli masz więcej niż jedno Raspberry Pi podłączone do tej samej sieci, to powinieneś nadać im różne nazwy. Pozwoli to na ich identyfikację (zobacz receptura 2.4).

Jeżeli korzystasz z systemu Windows lub Linux, to wciąż będziesz w stanie połączyć się z Raspberry Pi za pomocą VNC (receptura 2.8).

Zobacz również

Powyższe instrukcje napisano w oparciu o artykuł <http://4dc5.com/2012/06/12/setting-up-vnc-on-raspberry-pi-for-mac-access/>, który odwołuje się do książki *Getting Started with Raspberry Pi* napisanej przez Matta Richardsona i Shawna Wallace'a (<http://shop.oreilly.com/product/0636920023371.do>).

2.11. Używanie Raspberry Pi jako magazynu NAS

Problem

Chcesz używać Raspberry Pi jako magazynu NAS (ang. *Network Attached Storage*) — uzyskiwać ze swojego komputera dostęp do dysku twardego podłączonego do Pi za pośrednictwemłącza USB.

Rozwiążanie

W tym celu musisz zainstalować i skonfigurować narzędzie Samba. Aby to zrobić, skorzystaj z następujących poleceń:

```
$ sudo apt-get install samba  
$ sudo apt-get install samba-common-bin
```

Teraz podłącz do Raspberry Pi dysk twardy za pośrednictwem interfejsu USB. Zostanie on automatycznie zamontowany w folderze */media*. Sprawdź to przy użyciu następujących poleceń:

```
$ cd /media  
$ ls
```

Dysk powinien się znajdować na liście. Zostanie wyświetlona nazwa, jaką nadałeś mu podczas formatowania. Dysk będzie automatycznie montowany przy każdym uruchomieniu Raspberry Pi.

Teraz musisz skonfigurować narzędzie Samba, tak aby dysk był udostępniany w sieci. Zaczni od dodania do programu użytkownika (*pi*). Wprowadź poniższe polecenia i wpisz hasło:

```
$ sudo smbpasswd -a pi  
New SMB password:  
Retype new SMB password:  
Added user pi.
```

Następnie musisz zmodyfikować plik */etc/samba/smb.conf*. W tym celu wpisz polecenie:

```
$ sudo nano /etc/samba/smb.conf
```

W górnej części pliku znajduje się pierwsza z linii, których szukamy:

```
workgroup = WORKGROUP
```

Wiersz ten musisz zmodyfikować tylko wtedy, gdy chcesz się łączyć z Raspberry Pi za pomocą komputera działającego pod kontrolą systemu Windows. W tym miejscu powinna się znajdować nazwa Twojej grupy roboczej. Domyślną nazwą grupy roboczej w systemie Windows XP jest *MSHOME*, a w nowszych wersjach systemu Windows *HOME* (sprawdź to w ustawieniach sieciowych systemu).

Kolejny fragment pliku, który musisz zmodyfikować, znajduje się w sekcji *Autentication*:

```
# security = user
```

Usunięcie znaku `#` spowoduje włączenie ochrony.

Gdy już to zrobisz, na końcu pliku dodaj poniższy fragment kodu:

```
[USB]
path = /media/NAS
comment = NAS Drive
valid users = pi
writeable = yes
browseable = yes
create mask = 0777
public = yes
```

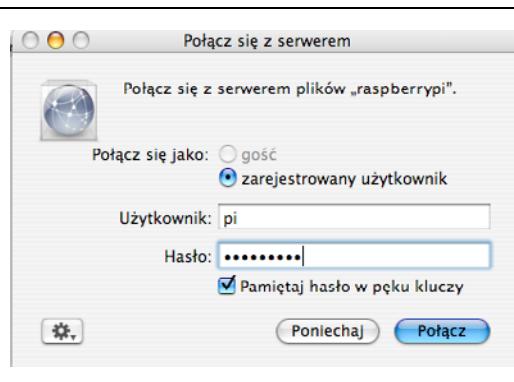
Zapisz zmodyfikowany plik i uruchom ponownie narzędzie Samba za pomocą polecenia:

```
$ sudo /etc/init.d/samba restart
```

Jeżeli wszystko zostało wykonane poprawnie, Twój dysk USB powinien być już udostępniany w sieci.

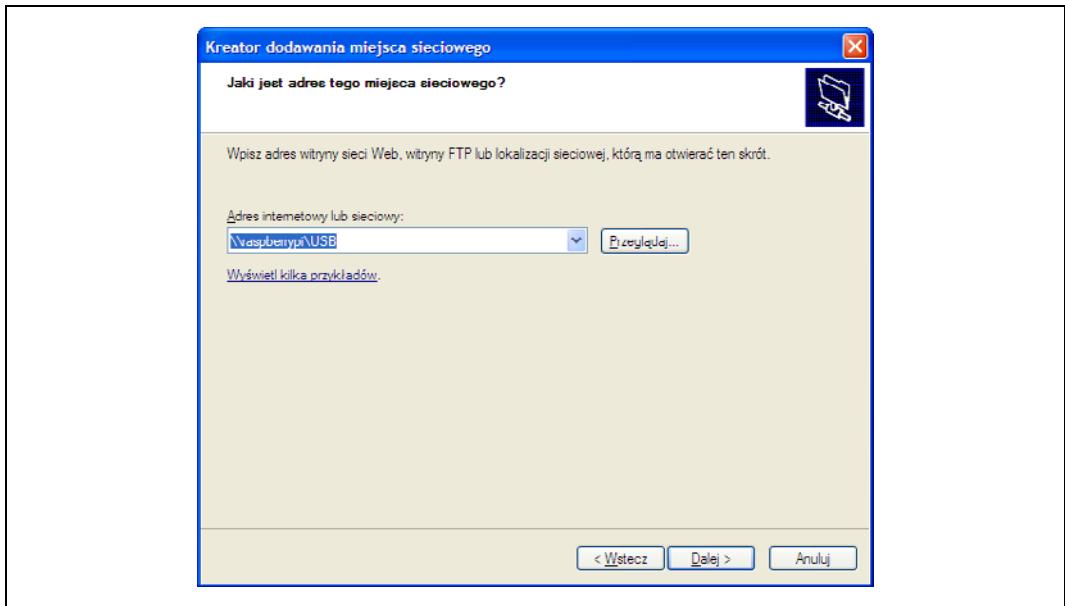
Omówienie

Aby przeglądać zawartość dysku z komputera Macintosh, wybierz *Połącz z serwerem* w menu *Idź*. Jako adres serwera wprowadź `smb://raspberrypi/USB`. Teraz powinien się wyświetlić komunikat z prośbą o podanie nazwy użytkownika i hasła. Jako nazwę użytkownika wpisz `pi` (zobacz rysunek 2.11).



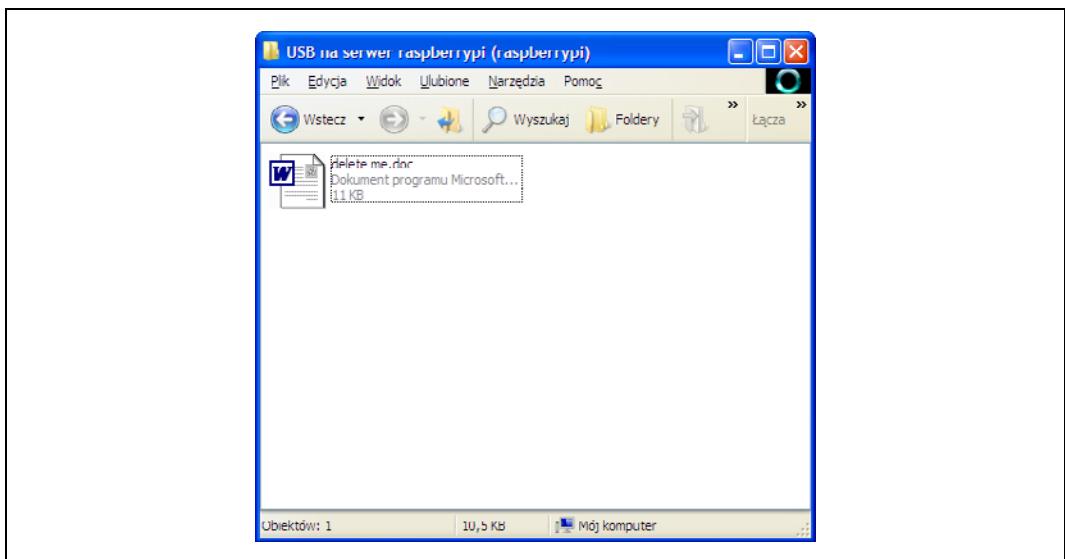
Rysunek 2.11. Łączenie z serwerem NAS w systemie Mac OS X

Jeżeli łączysz się z serwerem NAS z systemu Windows, to dokładna procedura uzyskania dostępu będzie zależeć od wersji systemu, który posiadasz. Będzie się ona jednak sprowadzać do podania adresu sieciowego `\\\raspberrypi\USB` (zobacz rysunek 2.12).



Rysunek 2.12. Łączenie z serwerem NAS w systemie Windows

Po podaniu nazwy użytkownika i hasła będziesz mógł korzystać z zawartości dysku serwera NAS (zobacz rysunek 2.13).



Rysunek 2.13. Przeglądanie zawartości serwera NAS w systemie Windows

Jeżeli pracujesz w systemie Linux, to w celu zamontowania napędu NAS zastosuj poniższe polecenia:

```
$ sudo mkdir /pishare  
$ sudo smbmount -o username=pi,password=raspberry //192.168.1.16/USB /pishare
```

Zobacz również

Możesz nadać bardziej adekwatną nazwę sieciową urządzeniu Raspberry Pi, np. *PiNAS* (zobacz receptura 2.4).

2.12. Drukowanie sieciowe

Problem

Chcesz drukować na drukarce sieciowej ze swojego Raspberry Pi.

Rozwiążanie

Zastosuj system obsługi urządzeń drukujących CUPS.

Zainstaluj go, wpisując w Terminalu poniższe polecenie. Wykonanie go może zająć trochę czasu.

```
$ sudo apt-get install cups
```

Nadaj sobie uprawnienia administracyjne w systemie CUPS za pomocą poniższego polecenia:

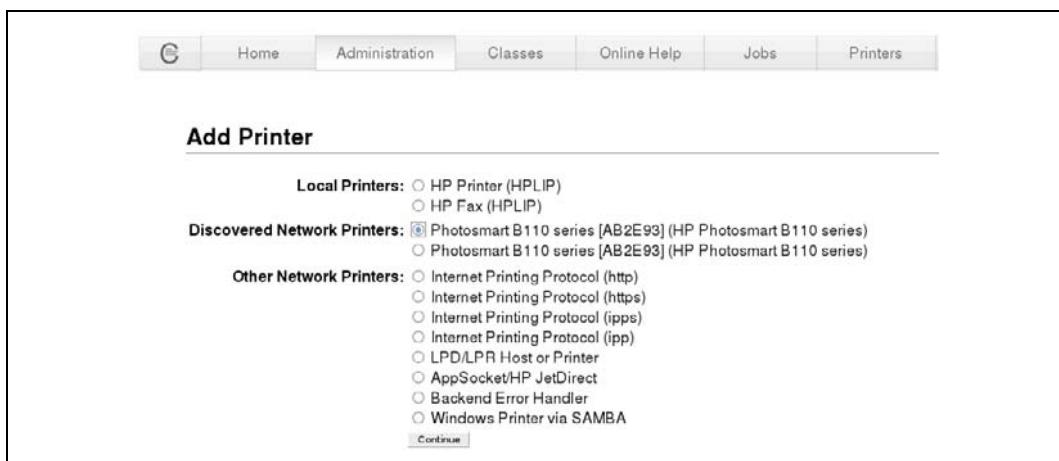
```
$ sudo usermod -a -G lpadmin pi
```

System CUPS jest konfigurowany za pośrednictwem interfejsu sieciowego. Przeglądarki Midori i Dillo niezbyt dobrze sobie radzą z obsługą tego panelu administracyjnego, warto więc zainstalować przeglądarkę Iceweasel. W tym celu wpisz następujące polecenie:

```
$ sudo apt-get install iceweasel
```

Uruchom zainstalowaną właśnie przeglądarkę — skrót do niej znajduje się w menu *Start* w grupie *Internet*. W polu adresu przeglądarki wpisz **http://localhost:631**.

Przejdź na zakładkę *Administration* i wybierz opcję *Add Printer*. Zostanie wyświetlona lista drukarek, które są dostępne w sieci lub są podłączone bezpośrednio do portu USB Raspberry Pi (zobacz rysunek 2.14).

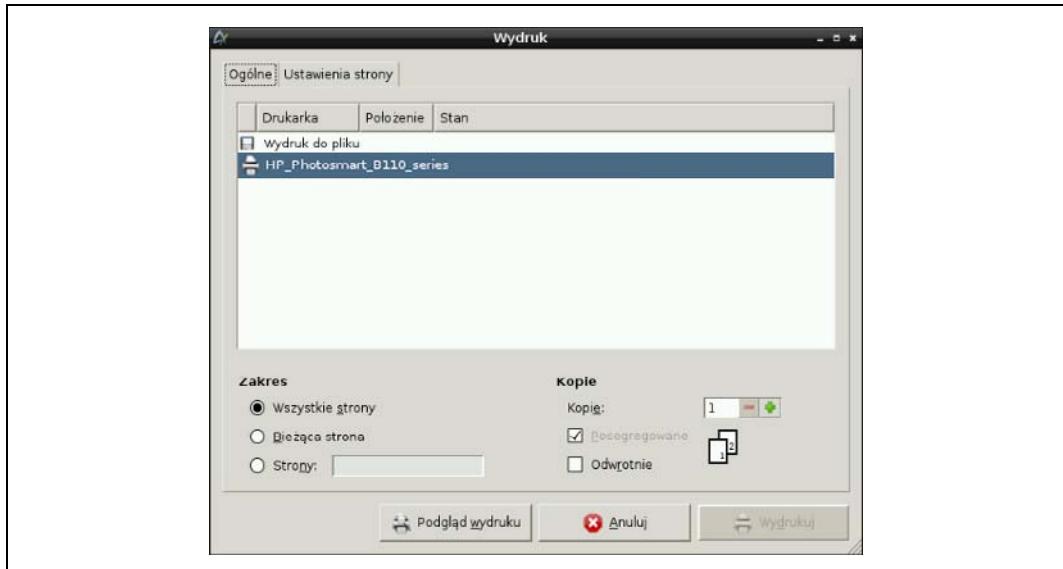


Rysunek 2.14. Wyszukiwanie drukarek za pomocą systemu CUPS

Kolejne okna dialogowe przeprowadzą Cię przez proces konfiguracji drukarki.

Omówienie

Teraz możesz sprawdzić działanie drukarki za pomocą programu AbiWord (zobacz receptura 4.2). W oknie programu wpisz dowolny tekst i spróbuj go wydrukować. Ostatnio dodane drukarki powinny być widoczne w oknie dialogowym pokazanym na rysunku 2.15.



Rysunek 2.15. Drukowanie z programu AbiWord

Zobacz również

Zajrzyj na oficjalną witrynę systemu CUPS — <http://www.cups.org/>.

System operacyjny

3.0. Wprowadzenie

W rozdziale tym opisano wiele spraw związanych z systemem operacyjnym Linux, pod kontrolą którego działa Raspberry Pi. Wiele czynności w tym systemie wykonuje się za pomocą wiersza poleceń.

3.1. Przenoszenie plików w interfejsie graficznym

Problem

Chcesz zarządzać plikami za pomocą graficznego interfejsu użytkownika, tak jak na komputerze Macintosh lub PC.

Rozwiążanie

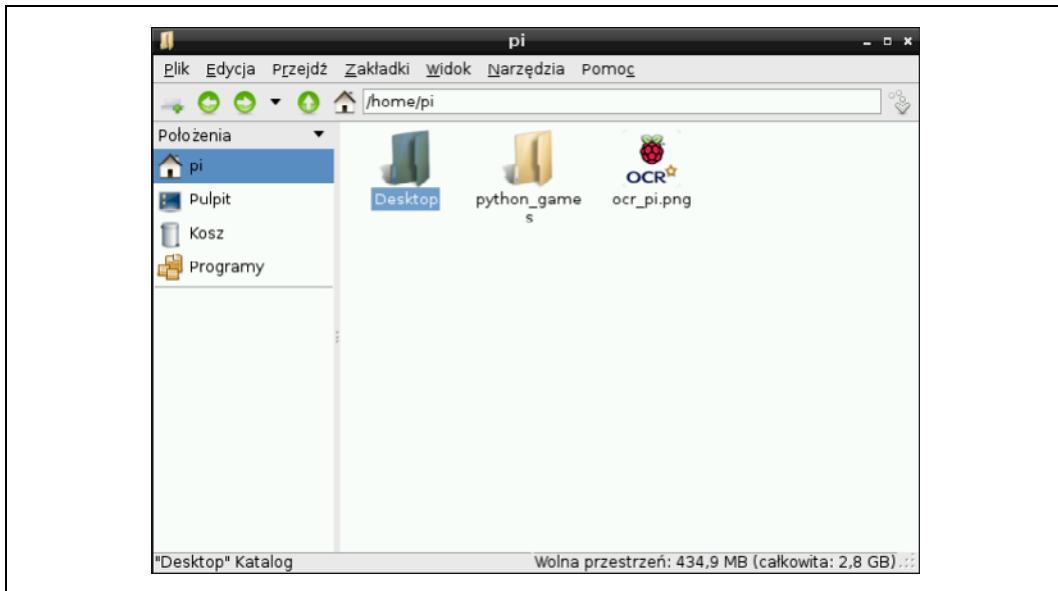
Skorzystaj z Menedżera plików.

Program ten znajdziesz w menu *Start* w grupie *Akcesoria* (zobacz rysunek 3.1).

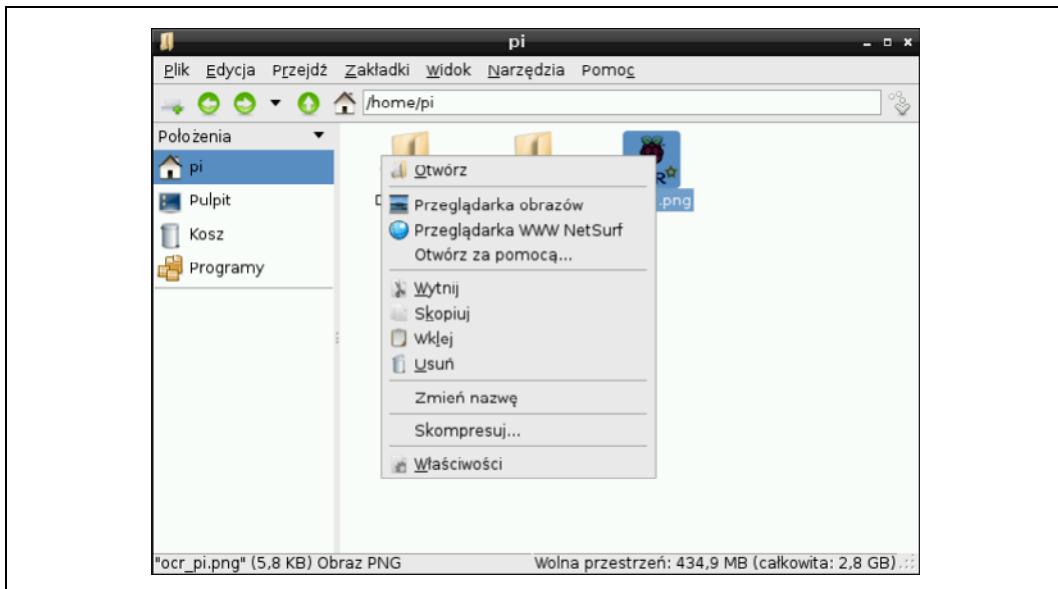
Omówienie

Po lewej stronie Menedżera plików znajdują się zamontowane woluminy. Właśnie tutaj pojawi się dysk podłączony przez Ciebie do interfejsu USB.

Na środku wyświetlane są pliki znajdujące się w bieżącym katalogu. Po strukturze katalogów możesz się przemieszczać za pomocą przycisków umieszczonego na pasku narzędziowym lub poprzez wpisanie ścieżki dostępu do pliku w polu w górnej części okna. Kliknij dowolny plik widoczny w oknie prawym przyciskiem myszy, a wyświetli się menu opcji widoczne na rysunku 3.2.



Rysunek 3.1. Menedżer plików



Rysunek 3.2. Menedżer plików

Zobacz również

Zobacz również recepturę 3.4.

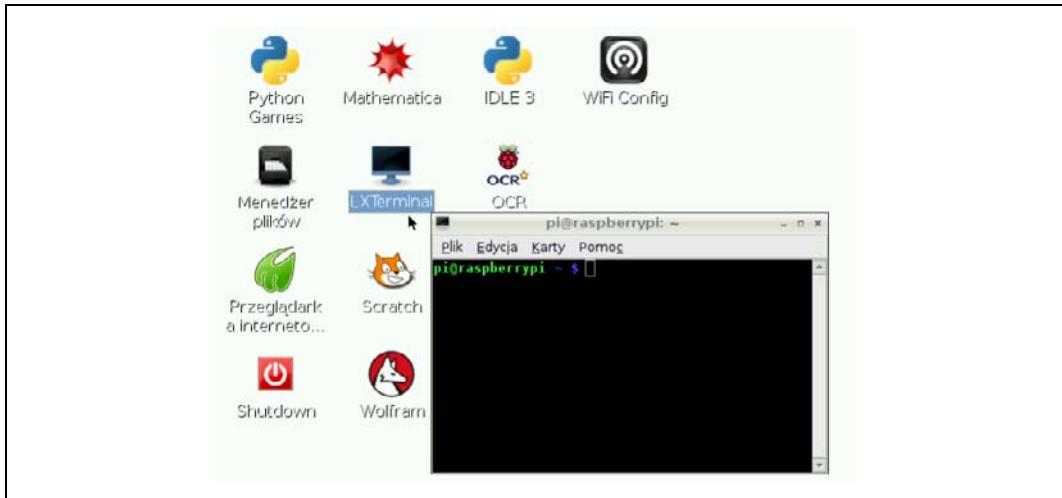
3.2. Uruchamianie sesji Terminala

Problem

Korzystając z Raspberry Pi, spotykasz się z koniecznością wpisywania polecień w Terminalu.

Rozwiążanie

Kliknij ikonę *LXTerminal* znajdującą się na pulpicie (zobacz rysunek 3.3). Jeżeli nie masz skrótu do tego narzędzia na pulpicie, możesz je również odnaleźć w menu *Start* w grupie *Akcesoria*.



Rysunek 3.3. Otwieranie okna LXTerminal

Omówienie

LXTerminal uruchomi się w Twoim katalogu domowym (*/home/pi*).

Możesz otworzyć dowolną liczbę sesji Terminala. Często przydaje się możliwość otwarcia sesji z kilkoma różnymi folderami. W ten sposób nie będziesz musiał ciągle przechodzić pomiędzy nimi za pomocą polecenia *cd* (receptura 3.3).

Zobacz również

W kolejnej sekcji (receptura 3.3) omówimy przeglądanie struktury plików i folderów za pomocą Terminala.

3.3. Przeglądanie plików i folderów za pomocą Terminala

Problem

Powinieneś potrafić przeglądać pliki i foldery przy użyciu Terminala.

Rozwiązanie

Głównym poleceniem stosowanym podczas nawigowania przez system plików jest `cd` (zmień folder). Po komendzie tej należy określić folder, do którego chcesz wejść. Możesz podać zarówno ścieżkę **względną** do katalogu znajdującego się wewnątrz aktualnie przeglądanej folderu, jak i ścieżkę **bezwzględną** do dowolnego innego katalogu. Zajrzyj do poniższej sekcji „Omówienie”.

Po wpisaniu komendy `pwd` Raspberry Pi wyświetli nazwę aktualnie przeglądanego folderu.

Omówienie

Sprawdźmy działanie tego polecenia w praktyce. Otwórz sesję Terminala. Na ekranie pojawi się napis podobny do poniższego:

```
pi@raspberrypi ~ $
```

Przypomina Ci on, że Twoja nazwa użytkownika to `pi`, nazwa komputera to `raspberrypi`, a znak `~` to skrót od Twojego katalogu domowego (`/home/pi`). A zatem przeglądając dowolny folder, możesz wrócić do katalogu domowego za pomocą polecenia:

```
$ cd ~
```



W książce każda linia, w której masz wpisać polecenie, rozpoczyna się od znaku `$`. Informacje wyświetlane w wierszu poleceń nie będą się zaczynały od tego prefiksu. Będzie on stosowany w ten sam sposób, w jaki jest wyświetlany na ekranie Raspberry Pi.

To, czy podane wcześniej polecenie przeniosło Cię do katalogu domowego, możesz sprawdzić za pomocą komendy `pwd`:

```
$ pwd  
/home/pi
```

Jeżeli chcesz przejść do katalogu nadzielnego, po poleceniu `cd` możesz zastosować specyfikator `..` (dwie kropki):

```
$ cd ..  
$ pwd  
/home
```

Jak już pewnie zauważyłeś, ścieżki dostępu do plików i folderów składają się ze słów oddzielonych znakiem `/`. A zatem główny katalog całego systemu plików to `/`, a katalog o nazwie `home` ma adres `/home/`. Katalog `pi` znajdujący się w katalogu `home` ma adres `/home/pi/`. Ostatni znak ścieżki `(/)` może zostać pominięty.

Ścieżki mogą być bezwzględne (zaczynają się od `/` i określają pełną ścieżkę dostępu do pliku lub folderu z katalogu głównego) lub względne (dostęp jest określany względem bieżącego katalogu, w którym akurat pracujesz; nie mogą się one rozpoczynać od znaku `/`).

Masz nieograniczone możliwości zapisu i odczytu plików znajdujących się w Twoim katalogu domowym, jednak w miejscach, w których są umieszczone pliki systemowe, dostęp do niektórych plików może być ograniczony tylko do możliwości ich odczytu. Możesz obejść te ograniczenia (receptura 3.11), ale należy zachować przy tym ostrożność.

Sprawdź zawartość katalogu głównego, wpisując polecenia:

```
$ cd /
$ ls
bin dev home lost+found mnt proc run selinux sys usr
boot etc lib media opt root sbin srv tmp var
```

Polecenie `ls` wyświetli listę plików i folderów znajdujących się w katalogu głównym. Na liście zobaczysz katalog `home`, z którego właśnie wyszedłeś.

Teraz wejdźmy do któregoś z folderów widocznych na liście za pomocą polecenia:

```
$ cd etc
$ ls
adduser.conf hosts.deny polkit-1
alternatives hp profile
apm iceweasel profile.d
apparmor.d idmapd.conf protocols
apt ifplugd pulse
asound.conf init python
```

Zwróć uwagę na kilka rzeczy. Po pierwsze, na liście znajduje się więcej plików i folderów, niż może być wyświetlonych jednocześnie na ekranie. Aby zapoznać się z całością listy, możesz skorzystać z paska przewijania znajdującej się w bocznej części okna.

Po drugie, pliki i foldery oznaczono odpowiednimi kolorami. Pliki wyświetlane są w kolorze białym, a foldery niebieskim.

Jeżeli nie lubisz wpisywać pełnych nazw, wciśnięcie klawisza `Tab` spowoduje automatyczne uzupełnienie nazwy pliku, którą zacząłeś wpisywać. Wyobraź sobie na przykład, że chcesz wejść do katalogu `network`. Wpisujesz polecenie `cd netw` i wciskasz klawisz `Tab`. Fraza `netw` pozwala na jednoznaczną identyfikację folderu, a więc wciśnięcie klawisza `Tab` spowoduje automatyczne uzupełnienie nazwy.

Jeżeli wpisany ciąg znaków nie pozwala na jednoznaczną identyfikację pliku lub folderu, to po ponownym wciśnięciu klawisza `Tab` zobaczysz listę możliwych opcji dokończenia nazwy. Jeżeli wpisałeś tylko frazę `net` i wcisnąłłeś klawisz `Tab`, wyświetli się:

```
$ cd net
netatalk/ network/
```

Po poleceniu `ls` możesz podać argument zawężający listę. Zmień katalog na `/etc` i uruchom następujące polecenia:

```
$ ls f*
fake-hwclock.data fb.modes fstab fuse.conf

fonts:
conf.avail conf.d fonts.conf fonts.dtd

foomatic:
defaultspooler direct filter.conf

fstab.d:
pi@raspberrypi /etc $
```

Znak * nazywamy wieloznacznikiem. Stosując argument `f*` po poleceniu `ls`, wskazujemy, że chcemy zobaczyć listę wszystkich elementów, których nazwy rozpoczynają się literą `f`.

W wynikach wyszukiwania najpierw zostaną wyświetcone pliki znajdujące się w folderze `/etc`, których nazwy zaczynają się literą `f`. Następnie pojawią się pliki o nazwach rozpoczynających się od litery `f`, które znajdują się w podkatalogach folderu `/etc`.

Wieloznacznik jest często używany do wyświetlania plików z danym rozszerzeniem (np. `ls *.docx`).

W systemie Linux oraz w wielu innych systemach operacyjnych stosuje się konwencję rozpoznawania kropką nazw plików, które powinny być ukryte przed użytkownikiem. Pliki i foldery o takich nazwach nie będą uwzględniane w listach generowanych za pomocą polecenia `ls`, chyba że użyjesz specyfikatora `-a`. Poniżej znajduje się przykład zastosowania go w praktyce.

```
$ cd ~  
$ ls -a  
.  
..  
Adafruit-Raspberry-Pi-Python-Code  
.advance  
.AppleDB  
.AppleDesktop  
.AppleDouble  
Asteroids.zip  
atari_roms  
.bash_history  
.bash_logout  
.bashrc  
.cache  
.config  
.dbus  
Desktop  
.dillo  
.dmrc  
.emulationstation  
.fltk  
.fontconfig  
.gstreamer-0.10  
.gvfs  
indiecity  
.local  
.motor.py  
.mozilla  
mydocument.doc  
Network Trash Folder  
.pulse  
.pulse-cookie  
python_games  
sales_log  
servo.py  
.stella  
stepper.py.save  
switches.txt.save  
Temporary Items  
thermometer.py  
.thumbnails  
.vnc  
.Xauthority  
.xsession-errors  
.profile  
.xsession-errors.old
```

Jak widzisz, większość plików i folderów znajdujących się w Twoim katalogu domowym jest ukryta.

Zobacz również

Zobacz również recepturę 3.11.

3.4. Kopiowanie plików i folderów

Problem

Chcesz skopiować plik, korzystając z sesji Terminala.

Rozwiążanie

Aby skopiować plik lub folder, zastosuj polecenie `cp`.

Omówienie

Oczywiście możesz skopiować pliki, korzystając z Menedżera plików oraz opcji kopiowania i wklejania znajdujących się w jego menu (receptura 3.1).

Najprostszym przykładem zastosowania Terminala w celu skopiowania pliku jest wykonanie kopii pliku znajdującego się w Twoim bieżącym katalogu. Po poleceniu `cp` należy podać nazwę pliku, który chcesz skopiować, oraz nazwę nowego pliku, który zostanie utworzony w wyniku operacji kopiowania.

W poniższym przykładzie tworzymy plik o nazwie `myfile.txt`, a następnie kopujemy go, nadając kopii nazwę `myfile2.txt`. Więcej informacji o tworzeniu plików i poleceniu `>` znajdziesz w recepturze 3.8.

```
$ echo "witaj" > myfile.txt  
$ ls  
myfile.txt  
$ cp myfile.txt myfile2.txt  
$ ls  
myfile.txt  myfile2.txt
```

W przytoczonym przykładzie ścieżki obu plików znajdują się w bieżącym katalogu. Możesz jednak posługiwać się ścieżkami prowadzącymi do dowolnego miejsca, w którym możesz dokonywać zapisu. W kolejnym przykładzie skopijemy plik do katalogu `/tmp`, który jest przeznaczony do przechowywania plików tymczasowych. Nie zapisuj w nim niczego ważnego.

```
$ cp myfile.txt /tmp
```

Zauważ, że tym razem nie określiliśmy nazwy tworzonego pliku. Podaliśmy tylko miejsce, w którym ma być on zapisany. Kopia pliku `myfile.txt` zostanie umieszczona w folderze `/tmp` i będzie miała taką samą nazwę jak oryginał.

Czasami zamiast kopiować pojedyncze pliki, będziesz musiał wykonać kopię całego folderu zawierającego wiele plików i podfolderów. W celu skopiowania folderu zastosuj argument `-r` (*rekursywny*). Pozwoli to na skopiowanie folderu z całą zawartością.

```
$ cp -r mydirectory mydirectory2
```

Jeżeli nie masz prawa skopiować danego pliku lub folderu, to zostaniesz o tym poinformowany przez wiersz poleceń. Będziesz musiał zmienić prawa dostępu do danego pliku bądź folderu (receptura 3.12) lub skopiować go, korzystając ze specjalnych uprawnień użytkownika (zobacz receptura 3.11).

Zobacz również

Zobacz również receptury 3.5 i 3.12.

3.5. Zmiana nazwy pliku lub folderu

Problem

Chcesz zmienić nazwę pliku, korzystając z sesji Terminala.

Rozwiązanie

Zmień nazwę pliku lub folderu za pomocą polecenia `mv`.

Omówienie

Polecenie `mv` (ang. *move* — przenieś) jest stosowane w sposób podobny do polecenia `cp`, jednak pliki i foldery nie są kopiowane, a zmieniane są jedynie ich nazwy.

Żeby zmienić nazwę pliku `my_file.txt` na `my_file.rtf`, należy zastosować następujące polecenie:

```
$ mv my_file.txt my_file.rtf
```

Zmiana nazwy katalogu przebiega w ten sam sposób. Nie trzeba korzystać z parametru `-r` stosowanego przy kopiowaniu folderów w recepturze 3.4. Po zmianie nazwy katalogu będą w nim te same pliki i podkatalogi co przed zmianą nazwy.

Zobacz również

Zobacz również receptury 3.4 i 3.12.

3.6. Edycja pliku

Problem

Chcesz uruchomić edytor tekstowy w celu edycji pliku konfiguracyjnego.

Rozwiązanie

Skorzystaj z edytora nano, który wchodzi w skład większości systemów operacyjnych używanych z Raspberry Pi.

Omówienie

Żeby skorzystać z edytora nano, wystarczy wpisać polecenie `nano`, a następnie podać ścieżkę do pliku, który chcesz edytować. Jeżeli takowy plik nie istnieje, zostanie utworzony przez edytor w momencie zapisu. Pamiętaj o tym, że musisz posiadać uprawnienia do zapisu w danym katalogu.

Będąc w katalogu `home`, wpisz polecenie `nano my_file.txt` — w ten sposób utworzysz lub będziesz edytować już istniejący plik `my_file.txt` za pomocą edytora nano. Na rysunku 3.4 pokazano uruchomiony edytor.

Kursora nie możesz ustawać za pomocą myszy. W tym celu musisz korzystać z klawiszy kierunkowych znajdujących się na klawiaturze.

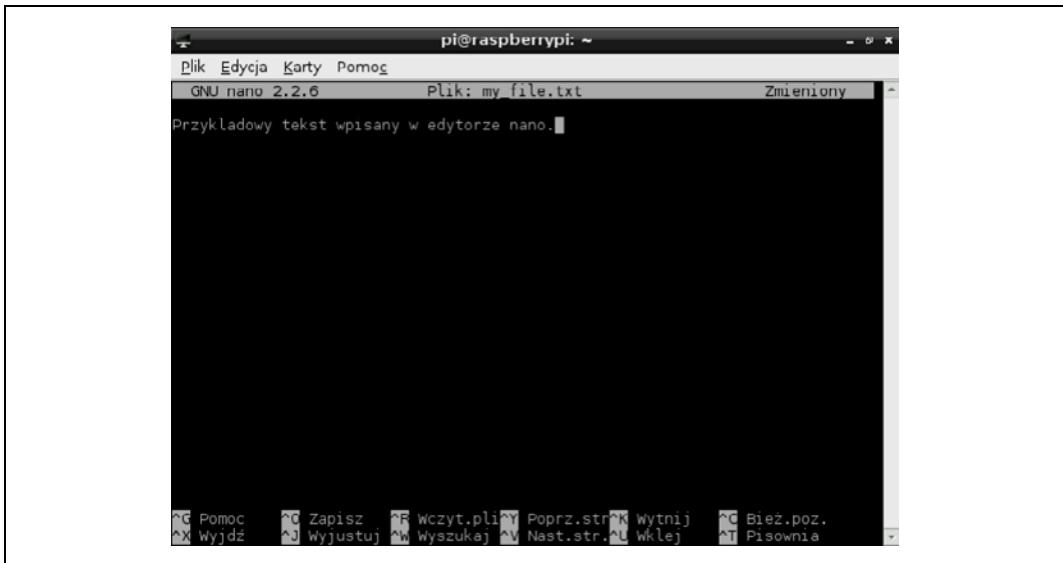
U dołu ekranu wyświetlna jest lista poleceń, z których możesz korzystać, przytrzymując klawisz `Ctrl`, a następnie wciskając właściwą literę na klawiaturze. Z większości tych funkcji nie będziesz jednak korzystać zbyt często. Najbardziej przydatne są:

`Ctrl+X`

Wyjdź. Przed wyjściem z edytora zostaniesz zapytany o to, czy chcesz zapisać plik.

`Ctrl+V`

Następna strona. Funkcja ta ma działanie podobne do strzałki w dół. Pozwala na przelączanie pomiędzy wyświetlonymi na ekranie partiami długiego pliku tekowego.



Rysunek 3.4. Edycja pliku za pomocą nano

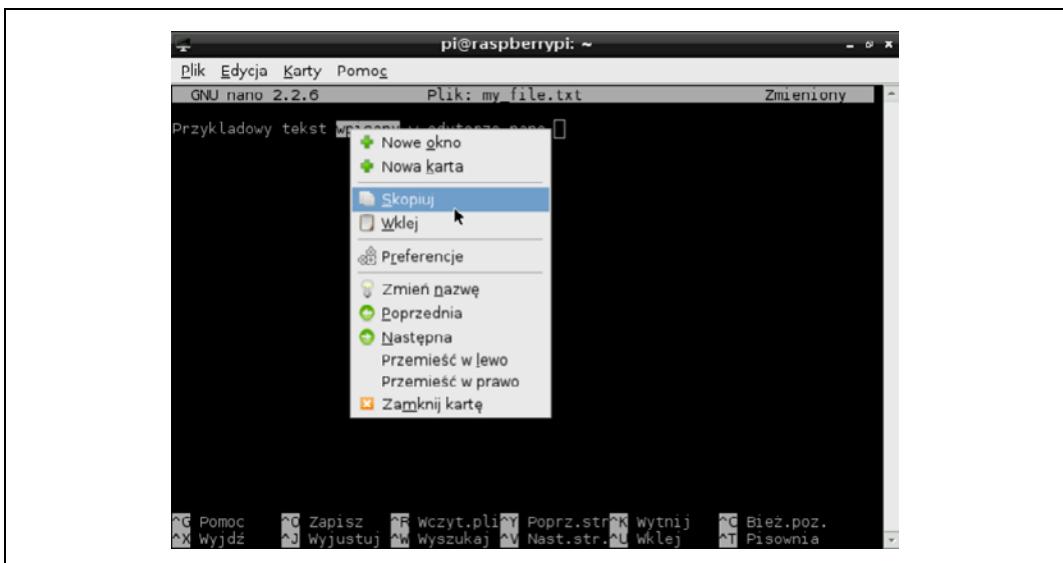
Ctrl+Y

Poprzednia strona.

Ctrl+W

Wyszukaj. Pozwala na znalezienie fragmentu tekstu.

Program zawiera dość prymitywne opcje typu „kopij i wklej”, jednak w praktyce łatwiej korzystać z normalnego schowka opartego o menu, które jest wywoływanie kliknięciem prawym przyciskiem myszy (zobacz rysunek 3.5).



Rysunek 3.5. Korzystanie ze schowka w edytorze tekstowym

Korzystanie ze schowka pozwala na kopiowanie i wklejanie tekstu do innych aplikacji — np. do przeglądarki internetowej.

Jeżeli jesteś gotowy, by zapisać zmiany w pliku i wyjść z edytora, wciśnij kombinację klawiszy *Ctrl+X*. Wciśnij *Y*, aby potwierdzić, że chcesz zapisać zmiany w pliku. Następnie możesz zmienić domyślną nazwę pliku. Wciśnij klawisz *Enter*, aby zapisać plik i wyjść z programu.

Jeżeli nie chcesz zapisywać dokonanych zmian, to zamiast klawisza *Y* wciśnij *N*.

Zobacz również

Wybór edytora tekstowego zależy od indywidualnych preferencji użytkownika. Wiele edytów tekstowych dostępnych dla systemu Linux może być uruchomionych na Raspberry Pi. Bardzo popularnym edytorem w świecie użytkowników Linuksa jest **vim** (ang. *vi improved*). Program ten jest dołączany do wielu popularnych dystrybucji systemów operacyjnych przeznaczonych dla Raspberry Pi. Niestety opanowanie tego edytora może się okazać dość trudne dla początkującego użytkownika. Możesz go uruchomić w sposób podobny do nano, jednak zamiast polecenia *nano* zastosuj polecenie *vi*. Więcej informacji dotyczących programu vim znajdziesz w internecie — http://newbiedoc.sourceforge.net/text_editing/vim.html.en.

3.7. Oglądanie zawartości pliku

Problem

Chcesz obejrzeć zawartość małego pliku bez edytowania go.

Rozwiążanie

W celu wyświetlenia zawartości pliku zastosuj polecenia *cat* lub *more*.

Na przykład:

```
$ more my_file.txt  
Przykładowy tekst wpisany w edytorze nano
```

Omówienie

Polecenie *cat* wyświetla całą zawartość pliku, nawet gdy nie mieści się ona na ekranie.

Polecenie *more* wyświetla tyle tekstu, ile mieści się na ekranie. W celu obejrzenia dalszej części pliku należy wciągnąć spację.

Zobacz również

Polecenie *cat* może być również stosowane do scalania (łączenia) kilku plików (zobacz receptura 3.28).

Innym popularnym poleceniem działającym podobnie do *more* jest *less*. Pozwala ono na przemieszczanie się w pliku do przodu oraz na cofanie się.

3.8. Tworzenie plików bez użycia edytora

Problem

Chcesz utworzyć mały plik bez użycia edytora.

Rozwiązanie

W celu zapisania do pliku treści wpisywanych w wierszu poleceń skorzystaj z poleceń > oraz echo.

Na przykład:

```
$ echo "tu wpisz zawartość pliku" > test.txt  
$ more test.txt  
tu wpisz zawartość pliku
```

Polecenie > nadpisuje obecną zawartość pliku, a więc korzystaj z niego ostrożnie.



Omówienie

To przydatny sposób na błyskawiczne tworzenie plików.

Zobacz również

Polecenie cat może być stosowane do oglądania zawartości plików bez otwierania ich w edytorze (zobacz receptura 3.7).

Inne zastosowanie polecenia > opisano w recepturze 3.27.

3.9. Tworzenie katalogów

Problem

Chcesz utworzyć nowy katalog za pomocą Terminala.

Rozwiązanie

Nowy katalog możesz utworzyć za pomocą polecenia mkdir.

Omówienie

Aby utworzyć nowy folder, skorzystaj z polecenia mkdir. Sprawdź działanie poniższego kodu:

```
$ cd ~  
$ mkdir my_directory  
$ cd my_directory  
$ ls
```

Musisz mieć uprawnienia do zapisu w katalogu, w którym tworzysz nowy podkatalog.

Zobacz również

Ogólne informacje na temat stosowania Terminala do przeglądania plików i folderów omówiono w recepturze 3.3.

3.10. Kasowanie plików i katalogów

Problem

Chcesz skasować plik lub katalog za pomocą Terminala.

Rozwiążanie

W celu usunięcia pliku lub katalogu wraz z zawartością skorzystaj z polecenia `rm` (ang. *remove* — usuń). Polecenie to powinno być stosowane z rozwagą.

Omówienie

Skasowanie pojedynczego pliku jest proste i bezpieczne. Poniżej przedstawiono ciąg komend usuwających plik `my_file.txt` z katalogu domowego:

```
$ cd ~  
$ rm my_file.txt  
$ ls
```

Musisz mieć uprawnienia do zapisu w katalogu, w którym chcesz przeprowadzić operację kasowania.

W celu usunięcia plików możesz stosować również wieloznacznik (*). Poniższy kod usuwa wszystkie pliki, których nazwy zaczynają się od fraz `my_file`, a które jednocześnie znajdują się w bieżącym katalogu.

```
$ rm my_file.*
```

Wszystkie pliki znajdujące się w katalogu mógłbyś skasować za pomocą polecenia:

```
$ rm *
```

Jeżeli chcesz skasować folder wraz z całą zawartością (wszystkimi plikami i podkatalogami), możesz skorzystać z atrybutu -r:

```
$ rm -r mydir
```



Kasując pliki za pomocą okna Terminala, pamiętaj o tym, że operacji tych nie można cofnąć — nie chroni Cię żaden mechanizm bezpieczeństwa w postaci kosza. Ponadto nie zostaniesz zapytany o potwierdzenie wykonywanej operacji — pliki będą kasowane natychmiast po wprowadzeniu polecenia. Jeżeli operację kasowania połączysz z poleceniem `sudo` (receptura 3.11), to skutki takich komend mogą być druzgocące.

Zobacz również

Zobacz również recepturę 3.3.

Jeżeli martwisz się, że możesz przypadkowo usunąć potrzebny Ci plik lub folder, możesz wymusić potwierdzanie polecenia `rm`, tworząc alias zastępujący tę komendę (receptura 3.32).

3.11. Wykonywanie zadań z uprawnieniami administratora

Problem

Niektóre polecenia nie działają, bo nie masz uprawnień do ich wykonywania. Musisz wydawać polecenia z uprawnieniami administratora.

Rozwiążanie

W celu wykonania poleceń z uprawnieniami administratora należy poprzedzić je komendą sudo (ang. *superuser do* — wykonaj jako administrator).

Omówienie

Większość zadań wykonasz z poziomu wiersza poleceń bez uprawnień administratora. Najczęściej spotykanymi czynnościami wymagającymi takich uprawnień są instalacja nowego oprogramowania i modyfikacja plików konfiguracyjnych.

Jeżeli np. spróbujesz uruchomić polecenie apt-get update, zobaczysz wiele wiadomości o braku właściwych uprawnień:

```
$ apt-get update
E: Nie udało się otworzyć pliku blokady /var/lib/apt/lists/lock - open (13: Brak dostępu)
E: Nie udało się zablokować katalogu /var/lib/apt/lists/
E: Nie udało się otworzyć pliku blokady /var/lib/dpkg/lock - open (13: Brak dostępu)
E: Nie udało się zablokować katalogu administracyjnego (/var/lib/dpkg/), czy to uprawnień
↳ administratora?
```

Ostatni komunikat informuje Cię, że prawdopodobną przyczyną problemu jest brak uprawnień administratora. Jeżeli uruchomisz to samo polecenie z prefiksem sudo, wszystko będzie działało poprawnie.

```
$ sudo apt-get update
Get:1 http://mirrordirector.raspbian.org wheezy InRelease [12.5 kB]
Hit http://archive.raspberrypi.org wheezy InRelease
Get:2 http://mirrordirector.raspbian.org wheezy/main Sources [6,241 kB]
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
40% [2 Sources 2,504 kB/6,241 kB 40%]
```

Zobacz również

Więcej informacji na temat atrybutów plików znajdziesz w recepturze 3.12.

Proces instalacji oprogramowania za pomocą polecenia apt-get opisano w recepturze 3.16.

3.12. Co oznaczają atrybuty plików?

Problem

Widzisz, że na listach plików przy ich nazwach pojawiają się tajemnicze znaki. Chcesz wiezieć, co one oznaczają.

Rozwiążanie

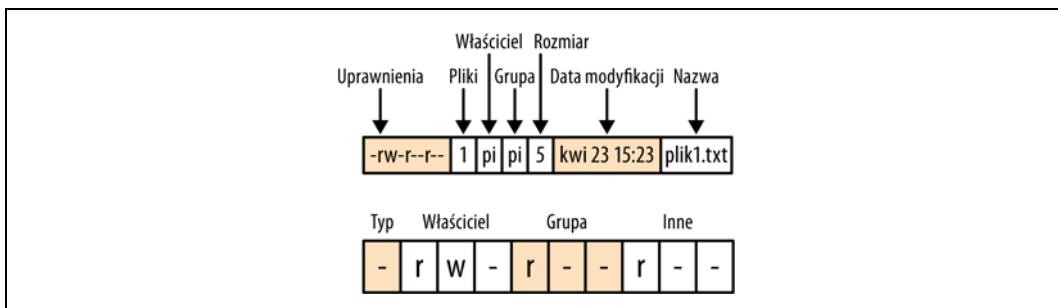
Aby zobaczyć informacje dotyczące atrybutów i własności plików, zastosuj polecenie `ls` z parametrem `-l`.

Omówienie

Uruchom polecenie `ls -l`, a wyświetli się lista podobna do poniższej.

```
$ ls -l
total 16
-rw-r--r-- 1 pi pi      5 kwi 23 15:23 plik1.txt
-rw-r--r-- 1 pi pi      5 kwi 23 15:23 plik2.txt
-rw-r--r-- 1 pi pi      5 kwi 23 15:23 plik3.txt
drwxr-xr-x 2 pi pi 4096 kwi 23 15:23 mójfolder
```

Na rysunku 3.6 przedstawiono kolumny, na jakie podzielone są dane na temat plików. Pierwsza kolumna zawiera uprawnienia. Liczba podana w drugiej kolumnie (podpisanej *Pliki*) informuje, ilu plików dotyczy dany wiersz. Pole to ma sens tylko wtedy, gdy opisywany element jest folderem. Jeżeli element jest plikiem, to pole to będzie zwykle zawierało liczbę 1. Następne dwa pola (oba zawierające nazwę *pi*) informują o właściwcu i grupie. Kolejna (piąta) kolumna podaje rozmiar pliku w bajtach. Data modyfikacji będzie ulegała zmianie za każdym razem, kiedy plik będzie edytowany lub modyfikowany. Ostatnia kolumna zawiera nazwę pliku.



Rysunek 3.6. Atrybuty plików

Łańcuch uprawnień jest podzielony na cztery sekcje: typ, właściciel, grupa i inne. Pierwsza sekcja określa, czy dany element jest plikiem (d), czy folderem (a).

Kolejna sekcja składa się z trzech znaków definiujących właściciela uprawnień do pliku. Każde z tych trzech miejsc może być wypełnione pewnym znakiem lub nie. Jeżeli właściciel może odczytywać plik, to na pierwszym miejscu będzie się znajdowała litera r. Jeżeli ma on

uprawnienia do zapisu, to na drugim miejscu będzie się znajdowała litera *w*. Jeżeli użytkownik ma uprawnienia do uruchomienia jakiegoś skryptu lub programu, to trzecie miejsce będzie zawierało literę *x*.

Trzecia sekcja również składa się z trzech znaków. Definiuje ona uprawnienia użytkowników należących do grupy. Użytkownicy mogą być przydzieleni do różnych grup. W przytoczonym przykładzie użytkownikiem pliku jest *pi*, a grupa, do której przynależy plik, również nazywa się *pi*. Użytkownicy należący do grupy *pi* będą mieli uprawnienia opisane atrybutami znajdującymi się właśnie w tej sekcji.

Ostatnia sekcja określa uprawnienia użytkowników niebędących użytkownikiem *pi* i nienależących do grupy o nazwie *pi*.

W związku z tym, że większość użytkowników Raspberry Pi będzie korzystała z urządzenia jako użytkownik *pi*, interesują nas głównie uprawnienia opisane w pierwszej sekcji.

Zobacz również

Zmianę atrybutów plików opisuje receptura 3.13.

3.13. Modyfikacja atrybutów plików

Problem

Chcesz zmodyfikować atrybuty pliku.

Rozwiążanie

Atrybuty plików można zmodyfikować za pomocą polecenia `chmod`.

Omówienie

Atrybuty pliku często zmienia się w przypadku pliku oznaczonego jako „tylko do odczytu”. Modyfikacje wykonuje się również po to, by nadać atrybut pozwalający danemu użytkownikowi na uruchamianie pliku będącego programem lub skryptem.

Polecenie `chmod` pozwala na dodanie lub usunięcie uprawnień dotyczących danego pliku. Operację tę można przeprowadzić na dwa sposoby — stosując różną składnię poleceń. Pierwszy sposób wymaga zastosowania systemu ósemkowego, a drugi można wykonać wyłącznie w trybie tekstowym. Zastosujemy drugą metodę, ponieważ jest łatwiejsza do zrozumienia.

Pierwszy parametr komendy `chmod` określa zmianę, jaką chcemy wykonać, a drugi definiuje plik lub folder, na którym chcemy dokonać modyfikacji. Parametry te dotyczą atrybutów i stosuje się z nimi znaki + (dodaj), - (usuń) i = (ustaw), a następnie należy podać typ uprawnienia.

Poniższy przykład doda uprawnienia wykonawcze (*x*) dla właściciela pliku o nazwie *plik2.txt*.

```
$ chmod u+x plik2.txt
```

Jeżeli teraz wyświetlimy listę plików i folderów, zobaczymy, że do pliku dodano atrybut wykonawczy x.

```
$ ls -l
total 16
-rw-r--r-- 1 pi pi    5 kwi 23 15:23 plik1.txt
-rwxr--r-- 1 pi pi    5 kwi 23 15:23 plik2.txt
-rw-r--r-- 1 pi pi    5 kwi 23 15:23 plik3.txt
drwxr-xr-x 2 pi pi 4096 kwi 23 15:23 mójfolder
```

Gdybyśmy chcieli nadać uprawnienia wykonawcze dla grupy lub innych użytkowników, zastosowalibyśmy odpowiednio atrybuty g lub o. Atrybut a nada uprawnienia każdemu użytkownikowi.

Zobacz również

Podstawowe wiadomości o atrybutach plików znajdziesz w recepturze 3.12.

W recepturze 3.14 przedstawiono sposób na zmianę właściciela pliku.

3.14. Zmiana właściciela pliku

Problem

Chcesz zmienić właściciela pliku.

Rozwiązanie

W celu zmiany właściciela pliku lub folderu skorzystaj z polecenia chown.

Omówienie

W recepturze 3.12 wspomniałem o tym, że każdemu plikowi i katalogowi przypisano właściciela i grupę. W związku z tym, że z Raspberry Pi zwykle korzysta tylko jeden użytkownik (*pi*), nie musimy zaprzątać sobie głowy grupami.

Czasami mogą Ci się trafić pliki zainstalowane przez użytkownika innego niż *pi*. W takim przypadku zastosuj polecenie chown w celu zmiany właściciela.

Żeby zmienić właściciela pliku, po poleceniu chown podaj nowego właściciela i grupę (informacje te powinny być oddzielone dwukropkiem), a następnie dodaj nazwę pliku.

W celu zmiany właściciela pliku potrzebne będą Ci uprawnienia administratora, a więc całe polecenie powinno się rozpoczynać prefiksem sudo (zobacz receptura 3.11).

```
$sudo shown root:root plik2.txt
$ ls -l
total 16
-rw-r--r-- 1 pi pi    5 kwi 23 15:23 plik1.txt
-rwxr--r-- 1 root root  5 kwi 23 15:23 plik2.txt
-rw-r--r-- 1 pi pi    5 kwi 23 15:23 plik3.txt
drwxr-xr-x 2 pi pi 4096 kwi 23 15:23 mójfolder
```

Zobacz również

Podstawowe wiadomości o atrybutach plików znajdziesz w recepturze 3.12.

W recepturze 3.13 przedstawiono sposób na zmianę atrybutów pliku.

3.15. Wykonywanie zrzutów ekranu

Problem

Chcesz wykonać zrzut ekranu i zapisać go w postaci pliku.

Rozwiążanie

Zainstaluj program `scrot`, który służy do wykonywania zrzutów ekranu.

Omówienie

Aby zainstalować wspomniany program, wpisz w Terminalu polecenie:

```
$ sudo apt-get install scrot
```

Zrzut ekranu możesz wykonać w bardzo prosty sposób. Wystarczy, że wpiszesz polecenie `scrot`. Spowoduje to natychmiastowe wykonanie zrzutu z pierwszego ekranu i zapisanie go w pliku o nazwie w rodzaju `2013-04-25-080116_1024x768_scrot.png`. Plik ten zostanie umieszczony w bieżącym katalogu.

Czasami istnieje potrzeba wykonania zrzutu, na którym widoczne będzie jakieś menu lub coś innego, co znika, gdy dane okno przestaje być aktywne. W takiej sytuacji możesz wykonać zrzut ekranu z opóźnieniem. Opóźnienie wyraża się w sekundach i podaje za parametrem `-d`.

```
$ scrot -d 5
```

Ze zrzutu całego ekranu będziesz mógł wyciąć potrzebne Ci elementy za pomocą edytora grafiki, takiego jak np. GIMP (receptura 4.10). Wygodniejsze jest jednak wykonanie zrzutu tylko wybranej części ekranu. Możesz to zrobić, korzystając z parametru `-s`.

Po wpisaniu polecenia z tym parametrem zaznacz myszką obszar ekranu, który ma się znajdować na zrzucie.

```
$ scrot -s
```

Nazwa pliku będzie zawierała informację na temat wymiaru zaznaczonego obszaru.

Zobacz również

Z poleceniem `scrot` możesz stosować wiele innych parametrów, które pozwalają na wykonywanie zrzutów z wielu ekranów i zmianę formatu zapisywanej pliku. Więcej informacji na temat tego programu uzyskasz, wpisując polecenie:

```
$ man scrot
```

Receptura 3.16 zawiera więcej informacji na temat instalowania oprogramowania za pomocą polecenia `apt-get`.

3.16. Instalacja oprogramowania za pomocą polecenia apt-get

Problem

Chcesz zainstalować nowe oprogramowanie za pomocą wiersza poleceń.

Rozwiążanie

W celu zainstalowania oprogramowania za pomocą sesji Terminala stosuje się narzędzie apt-get.

Składnię polecenia uruchamiającego to narzędzie podano poniżej. Należy je uruchamiać z uprawnieniami administratora.

```
$ sudo apt-get install <nazwa programu>
```

W celu zainstalowania edytora tekstu — programu AbiWord — należy wpisać polecenie:

```
$ sudo apt-get install abiword
```

Omówienie

Menedżer pakietów apt-get korzysta z listy dostępnego oprogramowania. Lista ta jest składnikiem zainstalowanego systemu operacyjnego i może być już nieaktualna. A zatem jeżeli podczas próby instalacji za pomocą polecenia apt-get zostanie wyświetlony komunikat „nie udało się odnaleźć pakietu”, to znaczy, że powinieneś zaktualizować tę listę, wpisując w wierszu polecenie:

```
$ sudo apt-get update
```

Zarówno lista, jak i oprogramowanie są pobierane z internetu, a więc polecenia te nie będą działać, jeżeli Raspberry Pi nie jest podłączone do sieci.



Jeżeli podczas aktualizacji wyświetli Ci się komunikat o treści podobnej do „E: Problem with MergeList /var/lib/dpkg/status”, spróbuj uruchomić poniższe polecenia:

```
sudo rm /var/lib/dpkg/status  
sudo touch /var/lib/dpkg/status
```

Proces instalacji może być długotrwały, ponieważ pliki przed instalacją są pobierane z internetu. Czasami skróty do zainstalowanych programów zostaną automatycznie umieszczone na pulpicie lub w menu *Start*.

Za pomocą polecenia apt-get search można wyszukiwać programy. Po tym poleceniu należy podać nazwę programu, którego szukasz (np. abiword). W wyniku zastosowania tego polecenia zostanie wyświetlona lista pakietów odpowiadających szukanej nazwie, które możesz zainstalować.

Zobacz również

W recepturze 3.17 przedstawiono proces usuwania niepotrzebnych już programów.

W recepturze 3.19 omówiono pobieranie kodu źródłowego z serwisu GitHub.

3.17. Usuwanie zainstalowanego oprogramowania za pomocą polecenia apt-get

Problem

Po zainstalowaniu wielu programów za pomocą polecenia apt-get chcesz usunąć niektóre z nich.

Rozwiążanie

Narzędzie apt-get pozwala na usunięcie pakietów, które zostały zainstalowane za pomocą polecenia apt-get install. W tym celu należy skorzystać z parametru remove.

Załóżmy, że chcesz usunąć edytor AbiWord. W tym celu możesz zastosować polecenie:

```
$ sudo apt-get remove abiword
```

Omówienie

Usuwając pakiety w ten sposób, nie skasujemy wszystkich elementów związanych z danym programem. Niektóre pakiety wymagają zainstalowania innych pakietów. Aby je usunąć, możesz zastosować parametr autoremove:

```
$ sudo apt-get autoremove abiword  
$ sudo apt-get clean
```

Polecenie apt-get clean spowoduje usunięcie jeszcze większej liczby elementów — dodatkowo zostaną skasowane nieużywane pliki instalacyjne pakietu.

Zobacz również

Zobacz również recepturę 3.16, w której opisano instalację pakietów za pomocą polecenia apt-get.

3.18. Pobieranie plików za pomocą wiersza poleceń

Problem

Chcesz pobrać plik z internetu bez użycia przeglądarki.

Rozwiążanie

W celu pobrania pliku z internetu zastosuj polecenie wget.

Na przykład:

```
$ wget http://www.icrobotics.co.uk/wiki/images/c/c3/Pifm.tar.gz  
--2013-06-07 07:35:01-- http://www.icrobotics.co.uk/wiki/images/c/c3/Pifm.tar.gz  
Translacja www.icrobotics.co.uk (www.icrobotics.co.uk)... 155.198.3.147  
Łączenie się z www.icrobotics.co.uk (www.icrobotics.co.uk)|155.198.3.147|  
:80... połączono.  
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
```

```
Długość: 5521400 (5.3M) [application/x-gzip]
Zapis do: `Pifm.tar.gz'
100%[=====] 5,521,400 601K/s
2013-06-07 07:35:11 (601 KB/s) - zapisano `Pifm.tar.gz' [5521400/5521400]
```

Jeżeli adres pliku, który chcesz pobrać, zawiera jakieś znaki specjalne, warto ująć go w cudzysłów. Adres zastosowany w powyższym przykładzie pochodzi z receptury 4.9.

Omówienie

Z pewnością nieraz zetkniesz się z instrukcjami instalacji oprogramowania, w których pliki będą pobierane za pomocą polecenia `wget`. Operację taką łatwiej wykonać za pomocą wiersza poleceń niż przeglądarki internetowej (co wymagałoby odnalezienia pliku, pobrania go i skopiowania we właściwe miejsce).

Korzystając z polecenia `wget`, w roli jego argumentu podajemy adres URL pliku, który chcemy pobrać do bieżącego folderu. Zwykle komendę tę stosuje się do pobierania archiwów, ale można wykorzystać ją również do pobrania dowolnej strony sieci Web.

Zobacz również

Zobacz również recepturę 3.16, w której opisano instalację pakietów za pomocą polecenia `apt-get`.

Receptura 4.3 zawiera informacje na temat wyboru przeglądarki oraz korzystania z niej.

3.19. Pobieranie kodu źródłowego za pomocą polecenia git

Problem

Czasami biblioteki Pythona oraz oprogramowanie mogą być dostarczone w formie odnośnika do zewnętrznego magazynu Git. Musisz pobrać takie dane na swoje Raspberry Pi.

Rozwiążanie

W celu pobrania danych z takiego serwisu musisz zainstalować narzędzie Git, a następnie za pomocą polecenia `git clone` uzyskać dostęp do plików.

Omówienie

Git jest systemem zarządzającym kodami źródłowymi. Aby go zainstalować, wpisz polecenie:

```
$ sudo apt-get install git
```

Po zainstalowaniu powyższej aplikacji możesz zacząć korzystać z polecenia `clone`, które służy do pobierania kodu źródłowego. Jeżeli pobierasz kody do folderu, w którym możesz zapisywać dane, to nie musisz wykonywać tej operacji jako administrator.

Na przykład poniższe polecenie spowoduje pobranie wszystkich kodów źródłowych omawianych w tej książce.

```
$ git clone https://github.com/simonmonk/raspberrypi_cookbook.git
```

Zobacz również

Więcej o usłudze Git dowiesz się z <http://www.git-scm.com/>, a więcej o usłudze hostingowej GitHub przeczytasz na <https://github.com/>.

Zobacz również recepturę 3.16.

3.20. Automatyczne uruchamianie programu lub skryptu podczas startu Raspberry Pi

Problem

Chcesz, żeby jakiś program lub skrypt był uruchamiany automatycznie podczas każdego startu Raspberry Pi.

Rozwiązanie

Linux Debian, na którym oparta jest większość systemów przeznaczonych dla Raspberry Pi, używa mechanizmu automatycznego uruchamiania poleceń rozruchu, który jest oparty na zależnościach. Korzystanie z tego rozwiązania jest stosunkowo trudne i wymaga tworzenia plików konfiguracyjnych w folderze o nazwie *init.d* dla skryptów lub programów, które chcesz uruchomić.

Omówienie

Poniższy skrypt pokazuje sposób uruchomienia skryptu Pythona znajdującego się w folderze *home*. Skrypt może wykonywać dowolną czynność, ale w przykładzie zastosowałem skrypt uruchamiający prosty serwer sieci Web napisany w Pythonie. Skrypt ten opisano bardziej szczegółowo w recepturze 7.16.

Operację tę można podzielić na trzy etapy:

1. Tworzenie skryptu *init*.
2. Sprawienie, że skrypt *init* będzie wykonywalny.
3. Poinformowanie systemu o nowym skrypcie *init*.

Pracę zacznij od stworzenia skryptu *init*. Należy go utworzyć w folderze */etc/init.d/*. Skrypt może mieć dowolną nazwę, ale w tym przykładzie nazwaliśmy go *my_server*.

Za pomocą edytora nano utwórz nowy plik. Zastosuj w tym celu polecenie:

```
$ sudo nano /etc/init.d/my_server
```

Wklej poniższy kod w oknie edytora i zapisz plik.

```
### BEGIN INIT INFO
# Provides: my_server
# Required-Start: $remote_fs $syslog $network
# Required-Stop: $remote_fs $syslog $network
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Simple Web Server
# Description: Simple Web Server
### END INIT INFO

#!/bin/sh
# /etc/init.d/my_server

export HOME
case "$1" in
    start)
        echo "Uruchamianie my_server"
        sudo /usr/bin/python /home/pi/myserver.py 2>&1 &
        ;;
    stop)
        echo "Zatrzymywanie my_server"
        PID=`ps auxwww | grep myserver.py | head -1 | awk '{print $2}'`^
        kill -9 $PID
        ;;
    *)
        echo "Używam: /etc/init.d/my_server {start|stop}"
        exit 1
        ;;
    esac
    exit 0
```

Zautomatyzowanie uruchamiania skryptu jest dość pracochłonne, jednak większość pracy polega na stosowaniu szablonowego kodu. Aby uruchomić inny skrypt, musisz go zmodyfikować, zmieniając opis i nazwę pliku Pythona, który chcesz uruchomić.

Teraz musisz sprawić, że ten plik będzie mógł być uruchamiany przez właściciela. W tym celu zastosuj kod:

```
$ sudo chmod +x /etc/init.d/my_server
```

Teraz, gdy program jest ustawiony tak, żeby działał jako usługa, możesz sprawdzić, czy wszystko zostało wykonane poprawnie. Zrób to przed umieszczeniem programu na liście aplikacji uruchamianych automatycznie podczas startu systemu. W celu sprawdzenia poprawności działania programu zastosuj poniższe polecenie:

```
$ /etc/init.d/my_server start
Uruchamianie my_server
Bottle v0.11.4 server starting up (using WSGIRefServer())...
Listening on http://192.168.1.16:80/
Naciśnij Ctrl-C, aby zakończyć.
```

Jeżeli wszystko działa poprawnie, to za pomocą podanego niżej polecenia poinformuj system o nowo zdefiniowanej usłudze:

```
$ sudo update-rc.d my_server defaults
```

Zobacz również

Więcej informacji na temat zmiany atrybutów plików i folderów znajdziesz w recepturze 3.13.

3.21. Automatyczne uruchamianie programu lub skryptu w regularnych odstępach czasu

Problem

Chcesz, żeby skrypt był uruchamiany raz dziennie lub w regularnych odstępach czasu.

Rozwiązanie

Zastosuj polecenie `crontab`.

Raspberry Pi do wykonania takiej operacji musi mieć dostęp do informacji o aktualnej dacie i godzinie. Potrzebny Ci będzie zatem zegar czasu rzeczywistego lub połączenie z internetem (zobacz receptura 11.13).

Omówienie

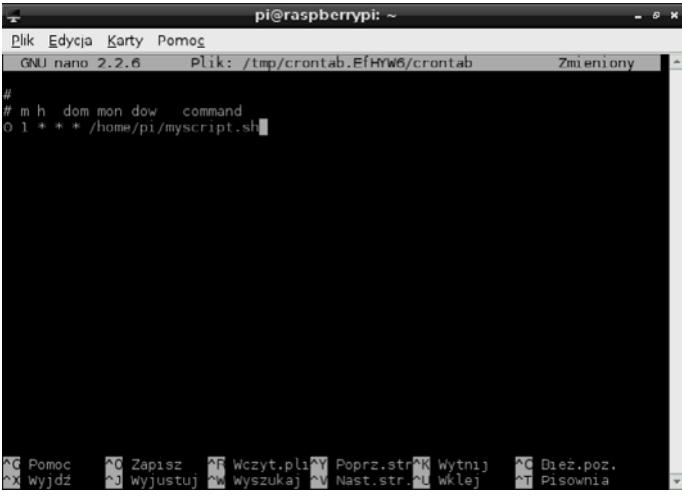
Polecenie `crontab` pozwala na zaplanowanie zadań, które mają być wykonywane w określonych odstępach czasu. Programy mogą być uruchamiane określonego dnia i o określonych godzinach. Istnieje możliwość zaplanowania uruchomiania każdego dnia innego programu. Pozwala to np. na uruchomienie w środku nocy procesu tworzenia kopii zapasowej.

Zaplanowane zadania możesz zmieniać za pomocą polecenia:

```
$ crontab -e
```

Jeżeli skrypt lub program, który ma być automatycznie uruchamiany, wymaga uprawnień administratora, to polecenie `crontab` poprzedź prefiksem `sudo` (receptura 3.11).

Komentarz widoczny na rysunku 3.7 informuje użytkownika o składni, jaką należy stosować w edytowanym pliku. Kolejno podawane cyfry definiują odpowiednio minutę, godzinę, dzień miesiąca, miesiąc i dzień tygodnia. Dalszy ciąg linii kodu określa polecenie, które ma zostać uruchomione.



```
#  
# m h dom mon dow   command  
0 1 * * * /home/pi/myscript.sh
```

Rysunek 3.7. Edycja harmonogramu automatycznie uruchamianych programów i skryptów

Gwiazdka (*), którą można wstawić w miejscu cyfr, stanowi synonim słów „w każdy” lub „w każdą”. Podając cyfry, możemy określić dokładnie datę i godzinę uruchomienia programu lub skryptu.

Na rysunku 3.7 widać linię kodu, która spowoduje codzienne uruchamianie skryptu o godzinie 1 w nocy.

Możesz również stosować zakresy, muszą być one jednak definiowane pojedynczymi cyframi. W poniższym przykładzie skrypt będzie uruchamiany tylko od poniedziałku do piątku.

```
0 1 * * 1-5 /home/pi/myscript.sh
```

Jeżeli Twój skrypt musi być uruchomiony z jakiegoś konkretnego katalogu, to polecenie informujące o tym oddziel średnikiem:

```
0 1 * * * cd /home/pi; python mypythoncode.py
```

Zobacz również

Zobacz pełną dokumentację narzędzia crontab, wpisując polecenie:

```
$ man crontab
```

3.22. Wyszukiwanie

Problem

Chcesz znaleźć plik, a nie znasz jego dokładnej lokalizacji.

Rozwiążanie

Skorzystaj z linuksowego polecenia find.

Omówienie

Polecenie find rozpoczęcie poszukiwania pliku w określonym katalogu, a po znalezieniu go wyświetli jego dokładną lokalizację. Na przykład:

```
$ find /home/pi -name gemgem.py  
/home/pi/python_games/gemgem.py
```

Wyszukiwanie plików możesz rozpocząć od folderu znajdującego się wyżej w hierarchii. Istnieje możliwość przeszukania całego systemu plików (/). Zajmie to jednak sporo czasu i może spowodować wyświetlanie komunikatów o błędach. Jeśli nie chcesz, by były one wyświetlane, dodaj na końcu polecenia 2>/dev/null.

Aby przeszukać cały system plików w poszukiwaniu konkretnego pliku, można zastosować następujące polecenie:

```
$ find / -name gemgem.py 2>/dev/null  
/home/pi/python_games/gemgem.py
```

Z poleceniem find możesz stosować również wieloznacznik (*):

```
$ find /home/pi -name match*
/home/pi/python_games/match4.wav
/home/pi/python_games/match2.wav
/home/pi/python_games/match1.wav
/home/pi/python_games/match3.wav
/home/pi/python_games/match0.wav
/home/pi/python_games/match5.wav
```

Zobacz również

Polecenie `find` ma również wiele innych zaawansowanych funkcji wyszukiwania. Aby do- wiedzieć się więcej na ich temat, wpisz polecenie:

```
$ man find
```

3.23. Korzystanie z historii wiersza poleceń

Problem

Chcesz skorzystać ponownie z wpisanych już wcześniej polecień bez potrzeby ponownego ich wpisywania.

Rozwiązanie

Jeśli chcesz przejrzeć wpisane wcześniej polecenie, skorzystaj z klawiszy strzałek (góra i dół). Aby przeglądać polecenia wpisane jeszcze wcześniej, użyj polecień `history` i `grep`.

Omówienie

Żeby uzyskać dostęp do listy wpisanych wcześniej polecień, wciśnij klawisz strzałki zwróconej ku górze. Dalsze wciskanie tego klawisza będzie powodować wyświetlanie kolejnych wpisywanych przez Ciebie polecień. Jeżeli przeskoczysz za daleko, zawsze możesz wrócić do uprzednio wyświetlonego polecenia za pomocą klawisza strzałki w dół.

Jeżeli nie chcesz wykonać wyświetlonego polecenia, wciśnij kombinację klawiszy `Ctrl+C`.

Z czasem historia polecień będzie tak dłużna, że znalezienie konkretnego wpisu będzie zajmowało wieki. W celu wyszukania polecenia możesz zastosować również polecenie `history`:

```
$ history
1 sudo nano /etc/init.d/my_server
2 sudo chmod +x /etc/init.d/my_server
3 /etc/init.d/my_server start
4 cp /media/4954-5EF7/sales_log/server.py myserver.py
5 /etc/init.d/my_server start
6 sudo apt-get update
7 sudo apt-get install bottle
8 sudo apt-get install python-bottle
```

Polecenie to wyświetli całą historię wpisywanych przez Ciebie polecień — prawdopodobnie będzie to niewygodnie dłuża lista. Problem ten możesz rozwiązać, łącząc ze sobą polecenia `history` i `grep`. Pozwoli to na wyświetlanie polecień zawierających wyszukiwany przez Ciebie ciąg znaków. W celu znalezienia wszystkich wpisywanych przez Ciebie wcześniej polecień związanych z narzędziem `apt-get` (receptura 3.16) możesz zastosować następujące polecenie:

```
$ history | grep apt-get
6 sudo apt-get update
7 sudo apt-get install bottle
8 sudo apt-get install python-bottle
55 history | grep apt-get
```

Każdy element historii jest oznaczony numerem. Jeżeli widzisz linię, którą chcesz ponownie uruchomić, wystarczy wpisać !, a następnie podać numer polecenia, a zostanie ono uruchomione. Ilustruje to poniższy przykład.

```
$ !6
sudo apt-get update
Hit http://mirrordirector.raspbian.org wheezy InRelease
Hit http://mirrordirector.raspbian.org wheezy/main armhf Packages
Hit http://mirrordirector.raspbian.org wheezy/contrib armhf Packages
....
```

Zobacz również

Jeżeli chcesz wyszukiwać pliki, a nie polecenia, zajrzyj do receptury 3.22.

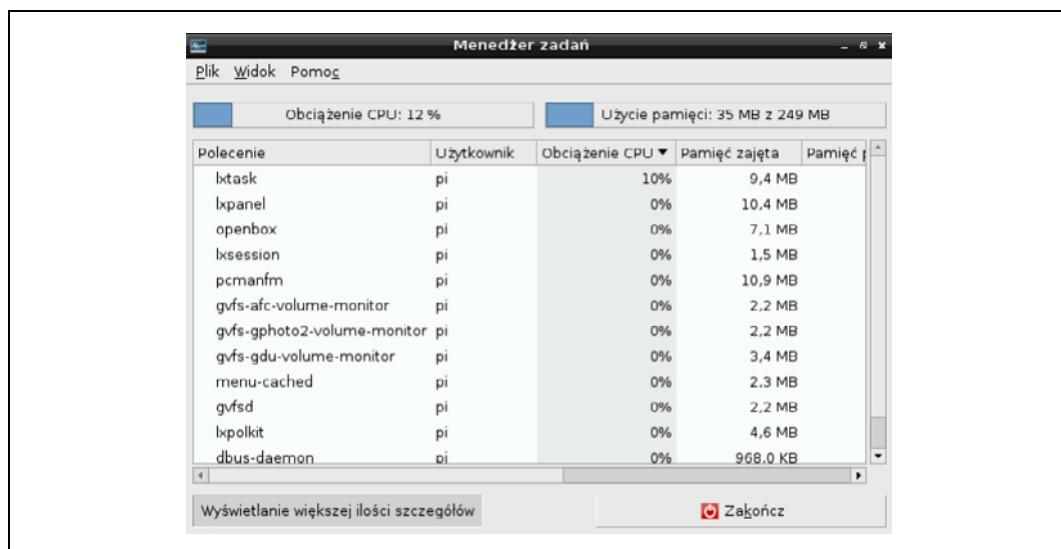
3.24. Monitorowanie aktywności procesora

Problem

Czasami wydajność pracy Raspberry Pi wyraźnie spada. Chcesz się dowiedzieć, jaki proces obciąża procesor.

Rozwiążanie

Skorzystaj z Menedżera zadań. Narzędzie to znajdziesz w menu *Start* w grupie *Narzędzia systemowe* (zobacz rysunek 3.8).



Rysunek 3.8. Menedżer zadań

Menedżer zadań informuje Cię, jak bardzo obciążony jest procesor. Ponadto narzędzie to pokazuje ilość zajmowanej pamięci operacyjnej. Klikając proces prawym przyciskiem myszy, możesz wybrać odpowiednią opcję i go wyłączyć.

W górnej części okna znajdują się wykresy ilustrujące obciążenie procesora i zajętą przestrzeń pamięci operacyjnej. Pod wykresami jest wyświetlana lista procesów, która informuje o tym, jak bardzo dany proces obciąża procesor.

Omówienie

Jeżeli wolisz, to taką samą czynność możesz wykonać za pomocą wiersza poleceń. Polecenie `top` wyświetla bardzo podobne dane na temat procesora, pamięci operacyjnej, zasobów sprzętowych i aktywnych programów (zobacz rysunek 3.9). W celu wyłączenia danego procesu zastosuj polecenie `kill`. Potrzebne Ci będą do tego uprawnienia administratora.

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2447	pi	20	0	9748	5216	2052	R	97.0	1.2	0:11.07	python
2206	lightdm	20	0	92132	11m	9248	S	1.3	2.6	0:54.98	lightdm-gtk-gre
2192	root	20	0	44972	11m	3172	S	0.7	2.7	0:25.25	Xorg
2448	pi	20	0	4656	1352	1028	R	0.7	0.3	0:00.09	top
2396	pi	20	0	9800	1640	1008	S	0.3	0.4	0:00.32	sshd
1	root	20	0	2144	728	620	S	0.0	0.2	0:01.89	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthread
3	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/u:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:00.33	kworker/u:0
7	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/u:0H
8	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khelper
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
10	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	bdi-default
13	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kblockd
14	root	20	0	0	0	0	S	0.0	0.0	0:00.35	khubd
15	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	rpciod
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khungtaskd

Rysunek 3.9. Wyświetlanie informacji dotyczących obciążenia zasobów sprzętowych za pomocą polecenia `top`

Założmy, że program napisany w Pythonie zajmuje 97% mocy obliczeniowej procesora, a numer identyfikacyjny tego procesu (PID) to 2447. Aby wyłączyć ten proces, wpisz polecenie:

```
$ kill 2447
```

Może się zdarzyć, że w ten sposób wyłączysz jakiś proces składowy systemu operacyjnego. W takim przypadku należy odłączyć Raspberry Pi od prądu i ponownie je włączyć — wszystko wróci do normy.

Czasami trudno znaleźć interesujący nas proces po zastosowaniu polecenia `top`. Możesz wówczas przeszukać wszystkie aktywne procesy za pomocą polecień `ps` i `grep` (zobacz receptura 3.29). W ten sposób interesujące Cię pozycje zostaną wyróżnione.

Na przykład w celu ustalenia numeru identyfikacyjnego (ID) procesu Pythona nadmiernie obciążającego procesor moglibyśmy skorzystać z poniższego ciągu polecen.

```
$ ps -ef | grep "python"
pi      2447 2397 99 07:01 pts/0 00:00:02 python speed.py
pi      2456 2397 0 07:01 pts/0 00:00:00 grep --color=auto python
```

W tym przykładzie numer identyfikacyjny programu Pythona o nazwie *speed.py* to 2447. Drugi proces na wyświetlanej liście to wynik wywołania polecenia *ps*.

Poleceniem podobnym do *kill* jest *killall*. Należy je jednak stosować z rozwagą, ponieważ wyłącza one wszystkie procesy, do których odnoszą się zastosowane argumenty. Poniższe przykładowe polecenie spowoduje wyłączenie wszystkich programów Pythona uruchomionych na Twoim Raspberry Pi.

```
$ sudo killall python
```

Zobacz również

Zapoznaj się również z informacjami o poleceniach *top*, *ps*, *grep*, *kill* i *killall*, które zostaną wyświetcone po zastosowaniu polecenia *man*.

Aby wyświetlić takie strony, należy wpisać *man*, a następnie nazwę interesującego nas polecenia, tak jak pokazano w poniższym przykładzie.

```
$ man top
```

3.25. Obsługa archiwów

Problem

Pobrałeś z internetu skompresowany plik. Chcesz go rozpakować.

Rozwiążanie

W zależności od typu pliku będziesz musiał zastosować polecenia *tar* lub *gunzip*.

Omówienie

Jeżeli plik, który chcesz rozpakować, ma rozszerzenie *.gz*, to możesz go rozpakować za pomocą polecenia:

```
$ gunzip myfile.gz
```

Z pewnością często natkniesz się na pliki (nazywane **tarballami**) zawierające foldery zarchiwizowane za pomocą narzędzia *tar*, które później zostały spakowane programem *gzip*. Takie pliki będą miały nazwy podobne do *mójplik.tar.gz*.

Takie archiva możesz rozpakować za pomocą polecenia *tar*.

```
$ tar -xzf mójplik.tar.gz
```

Zobacz również

Więcej informacji na temat archiwizatora *tar* znajdziesz po wpisaniu w Terminalu polecenia *man tar*.

3.26. Wyświetlanie listy podłączonych urządzeń USB

Problem

Chcesz się upewnić, że Linux rozpoznaje podłączone urządzenie USB.

Rozwiążanie

Polecenie `lsusb` wyświetli listę urządzeń podłączonych do Raspberry Pi za pośrednictwem portów USB.

```
$ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 15d9:0a41 Trust International B.V. MI-2540D [Optical mouse]
```

Omówienie

Wywołując to polecenie, możesz sprawdzić, czy dane urządzenie jest podłączone, czy nie. Nie masz jednak gwarancji, że działa ono poprawnie. Może Cię czekać proces instalacji sterowników lub konfiguracji podłączonego sprzętu.

Zobacz również

W recepturze 4.5 znajdziesz przykład zastosowania polecenia `lsusb` podczas podłączania zewnętrznej kamery internetowej.

3.27. Zapisywanie w pliku komunikatów wyświetlanych w wierszu poleceń

Problem

Chcesz szybko stworzyć plik zawierający jakiś tekst lub dane, takie jak np. lista plików i podkatalogów znajdujących się w danym folderze.

Rozwiążanie

Aby zapisać w pliku dane, które będą wyświetlane w oknie Terminala, zastosuj polecenie `>`.

Jeżeli chcesz zapisać w pliku o nazwie `mojepliki.txt` listę elementów znajdujących się w folderze, możesz skorzystać z poniższego kodu:

```
$ ls > mojepliki.txt
$ more mojepliki.txt
Desktop
indicecity
master.zip
mcpi
```

Omówienie

Polecenia > możesz używać z dowolnymi innymi poleceniami systemu Linux — nawet uruchamiając aplikację Pythona.

Możesz również korzystać z odwrotnego polecenia — <. Służy ono do zapisywania treści poleceń wprowadzanych przez użytkownika, a więc w praktyce nie jest tak często używane jak polecenie >.

Zobacz również

Polecenie > może być również stosowane do łączenia ze sobą plików (zobacz receptura 3.28).

3.28. Obsługa archiwów

Problem

Chcesz połączyć kilka małych plików tekstowych w jeden duży plik.

Rozwiążanie

W celu scalenia plików możesz zastosować polecenie cat. Poniżej znajduje się przykład jego użycia.

```
$ cat plik1.txt plik2.txt plik3.txt > pełny_plik.txt
```

Omówienie

Polecenie cat zostało stworzone właśnie do scalania ze sobą plików. Możesz scaić ze sobą dowolną liczbę plików. Zostaną one zapisane w jednym pliku, do którego skierujesz ten strumień danych. Jeżeli nie wskażesz pliku, w którym mają być zapisane dane ze scalonych plików, to zostaną one wyświetcone w oknie Twojego Terminala. Jeżeli scalasz ze sobą stosunkowo duże pliki, wyświetlenie całej ich zawartości może zająć trochę czasu.

Zobacz również

Polecenie cat może służyć również do wyświetlania zawartości pliku (zobacz receptura 3.7).

3.29. Korzystanie z potoków

Problem

Chcesz zastosować dane wyjściowe jednego polecenia jako dane wejściowe innego polecenia.

Rozwiążanie

Skorzystaj z polecenia pipe, które należy wprowadzać za pomocą symbolu pionowej kreski znajdującego się na Twojej klawiaturze (|). W ten sposób skierujesz dane wyjściowe jednego polecenia do innego polecenia. Na przykład:

```
$ ls -l *.py | grep lip  
-rw-r--r-- 1 pi pi 226 lip 7 06:49 speed.py
```

Powyższy przykład służy do wyszukania plików mających rozszerzenie *py*, które na liście plików posiadają oznaczenie *lip* — były ostatnio modyfikowane w lipcu.

Omówienie

Na pozór zapis ten może przypominać przekierowanie wyjścia za pomocą polecenia > (receptura 3.27). Nie będzie tu jednak działało polecenie +>_ — obiektem docelowym jest bowiem inny program. Polecenie to będzie działało tylko w przypadku skierowania danych do pliku.

W taki łańcuch możesz połączyć wiele programów, co zademonstrowano poniżej. W praktyce jednak rzadko będziesz wykonywać tę czynność.

```
$ polecenie1 | polecenie2 | polecenie3
```

Zobacz również

Zobacz również recepturę 3.24, w której przedstawiono przykład zastosowania funkcji grep do wyszukiwania procesów, a także recepturę 3.23, w której omówiono kierowanie wyników wyszukiwania historii wpisywanych poleceń do funkcji grep.

3.30. Ukrywanie danych wyjściowych wyświetlanych w oknie Terminala

Problem

Chcesz uruchomić jakieś polecenie, ale nie chcesz, żeby generowane przez nie dane wyjściowe były wyświetlane na ekranie.

Rozwiążanie

Przekieruj dane wyjściowe do */dev/null* za pomocą polecenia >.

Na przykład:

```
$ ls > /dev/null
```

Omówienie

Powyższy przykład dobrze ilustruje składnię polecenia, jednak w praktyce jest on bezużyteczny. Polecenie to będzie stosowane, gdy będziesz używał programu, w którym programista pozostawił w wielu miejscach elementy kodu wyświetlające dane dotyczące wykonywania

programu. Zwykle komunikaty tego typu nie będą Ci do niczego potrzebne. W poniższym przykładzie ukryliśmy dużą ilość danych wyjściowych wygenerowanych przez polecenie `find` (zobacz receptura 3.22).

```
$ find / -name gemgem.py 2>/dev/null  
/home/pi/python_games/gemgem.py
```

Zobacz również

Więcej informacji na temat przekierowywania standardowych danych wyjściowych znajdziesz w recepturze 3.27.

3.31. Uruchamianie programów w tle

Problem

Chcesz uruchomić program, ale jednocześnie pracować nad innym zadaniem.

Rozwiązanie

Uruchom program lub polecenie w tle za pomocą polecenia `&`.

Na przykład:

```
$ python speed.py &  
[1] 2528  
$ ls
```

Zamiast czekać na zakończenie działania programu, wiersz poleceń wyświetla numer identyfikacyjny procesu (druga liczba) i pozwala Ci na uruchamianie kolejnych programów. Podany numer identyfikacyjny programu przyda Ci się później do jego wyłączenia (zobacz receptura 3.24).

Aby wyświetlić proces wykonywany w tle, zastosuj polecenie `fg`.

```
$ fg  
python speed.py
```

Program dotychczas wykonywany w tle zostanie wyświetlony, a wiersz poleceń będzie czekać na jego zamknięcie.

Omówienie

Komunikaty generowane przez proces wykonywany w tle będą nadal wyświetlane w Terminalu.

Alternatywą uruchamiania procesu w tle jest równoczesne otwarcie więcej niż jednego okna Terminala.

Zobacz również

Więcej informacji dotyczących zarządzania procesami znajdziesz w recepturze 3.24.

3.32. Tworzenie aliasów poleceń

Problem

Chcesz utworzyć aliasy często używanych poleceń.

Rozwiązanie

Edytuj plik `~/.bashrc` za pomocą nano (zobacz receptura 3.6). Ustaw kursor na końcu pliku, a następnie dodaj dowolną liczbę linii o składni przedstawionej poniżej.

```
alias l='ls -a'
```

Przedstawiony fragment kodu tworzy alias o nazwie `l`, który po wprowadzeniu w wierszu poleceń będzie przetwarzany przez system interpretowany jako polecenie `ls -a`.

Zapisz plik za pomocą kombinacji klawiszy `Ctrl+X`, po czym wcisnij klawisz `Y`. Następnie uaktualnij listę aliasów obsługiwanych przez Terminal, wpisując polecenie:

```
$ source .bashrc
```

Omówienie

Wielu użytkowników systemu Linux ustawia alias polecenia `rm`, aby usuwanie plików wymagało dodatkowego potwierdzenia.

```
$ alias rm='rm -i'
```

To całkiem dobry pomysł. Pamiętaj o tym, że nie każdy użytkownik ustawia taki alias. Uważaj na to, pracując na innych komputerach!

Zobacz również

Więcej informacji na temat polecenia `rm` znajdziesz w recepturze 3.10.

3.33. Ustawianie daty i godziny

Problem

Chcesz ręcznie ustawić datę i godzinę na swoim Raspberry Pi.

Rozwiązanie

Zastosuj w tym celu polecenie `date`.

Datę i czas należy wprowadzić w następującym formacie: `MMDDHHMMYYYY`. W miejsce liter `MM` należy wprowadzić liczbę określającą miesiąc, a `DD` to dzień miesiąca. Litery `HH` i `MM` służą do podania odpowiednio godziny i minut. W miejsce liter `YYYY` należy podać rok.

Na przykład:

```
$ sudo date 010203042013
Śro, 2 sty 2013 03:04:00 CET
```

Omówienie

Jeżeli Raspberry Pi jest połączone z internetem, to informacje o aktualnej dacie i godzinie zostaną automatycznie pobrane z serwera czasu.

Aktualną datę i godzinę możesz wyświetlić, wpisując samo polecenie date.

```
$ date  
Śro, 2 sty 2013 03:08:14 CET
```

Zobacz również

Jeżeli chcesz, aby Raspberry Pi automatycznie ustawiało właściwą datę i godzinę (nawet bez połączenia z internetem), zastosuj w tym celu moduł zegara czasu rzeczywistego (zobacz receptura 11.13).

3.34. Ustalanie ilości wolnego miejsca na karcie pamięci

Problem

Chcesz się dowiedzieć, ile wolnego miejsca pozostało na karcie SD.

Rozwiążanie

W tym celu możesz zastosować polecenie df.

```
$ df -h  
System plików rozm. użyte dost. %uż. zamont. na  
rootfs 3.6G 1.7G 1.9G 48% /  
/dev/root 3.6G 1.7G 1.9G 48% /  
devtmpfs 180M 0 180M 0% /dev  
tmpfs 38M 236K 38M 1% /run  
tmpfs 5.0M 0 5.0M 0% /run/lock  
tmpfs 75M 0 75M 0% /run/shm  
/dev/mmcblk0p1 56M 19M 38M 34% /boot
```

Omówienie

Analizując pierwszą linię wyświetlonych informacji, możesz dostrzec, że karta pamięci charakteryzuje się pojemnością 3,6 GB, z czego zajęto 1,7 GB.

Gdy skończy się dostępna pamięć, Twoje Raspberry Pi może zacząć działać nieprawidłowo i np. wyświetlać komunikaty mówiące o tym, że plik nie może być zapisany.

Zobacz również

Więcej informacji na temat komendy df uzyskasz, wpisując w Terminalu polecenie man df.

Oprogramowanie

4.0. Wprowadzenie

Receptury przedstawione w tym rozdziale wiążą się z używaniem gotowego oprogramowania dostępnego na platformę Raspberry Pi.

Niektóre sekcje dotyczą zamiany Raspberry Pi w wyspecjalizowane urządzenie służące jednemu, konkretnemu celowi, a niektóre są związane z aplikacjami użytkowymi przeznaczonymi dla Raspberry Pi.

4.1. Tworzenie multimedialnego centrum rozrywki

Problem

Chcesz zamienić swoje Raspberry Pi w wypasione centrum rozrywki.

Rozwiążanie

Raspberry Pi całkiem dobrze sprawdza się w roli multimedialnego centrum rozrywki. Na rysunku 4.1 pokazano uruchomiony program XBMC (Xbox Media Center).

Raspberry Pi potrafi odtwarzać filmy w rozdzielcości full HD, strumieniować muzykę zapisaną w formacie MP3, a także obsługiwać radia internetowe.

XBMC jest oprogramowaniem otwartym, które powstało w celu zamiany konsoli Xbox w odtwarzacz multimedialny. Program ten został przystosowany do pracy na wielu różnych platformach, w tym na Raspberry Pi.

Aby zamienić Raspberry Pi w odtwarzacz multimedialny, musisz utworzyć nową kartę SD zawierającą system operacyjny oraz XBMC. Posłużymy się dystrybucją Raspbmc.



Rysunek 4.1. Raspberry Pi w roli multimedialnego centrum rozrywki

1. Pobierz z internetu obraz dysku.

W celu zapisania systemu na karcie pamięci niezbędny będzie komputer z systemem Windows, Mac OS X lub Linux, wyposażony w kontroler kart SD.

Pobierz plik-obraz z systemem Raspbmc ze strony <http://www.raspbmc.com/download>.

Na dole strony znajdź sekcję *Just want an image without a fancy installer?* i pobierz plik podpisany hasłem *Standalone Image*.

2. Zapisz obraz na karcie SD.

W celu zapisania systemu na karcie SD wykonaj czynności opisane w recepturze 1.6. Korzystaj z karty pamięci o pojemności co najmniej 4 GB.

3. Włóż kartę SD do Raspberry Pi i uruchom je.

Podczas pierwszego uruchomienia systemu będziesz musiał go skonfigurować. Po chwili będzie gotowy do pracy.

Omówienie

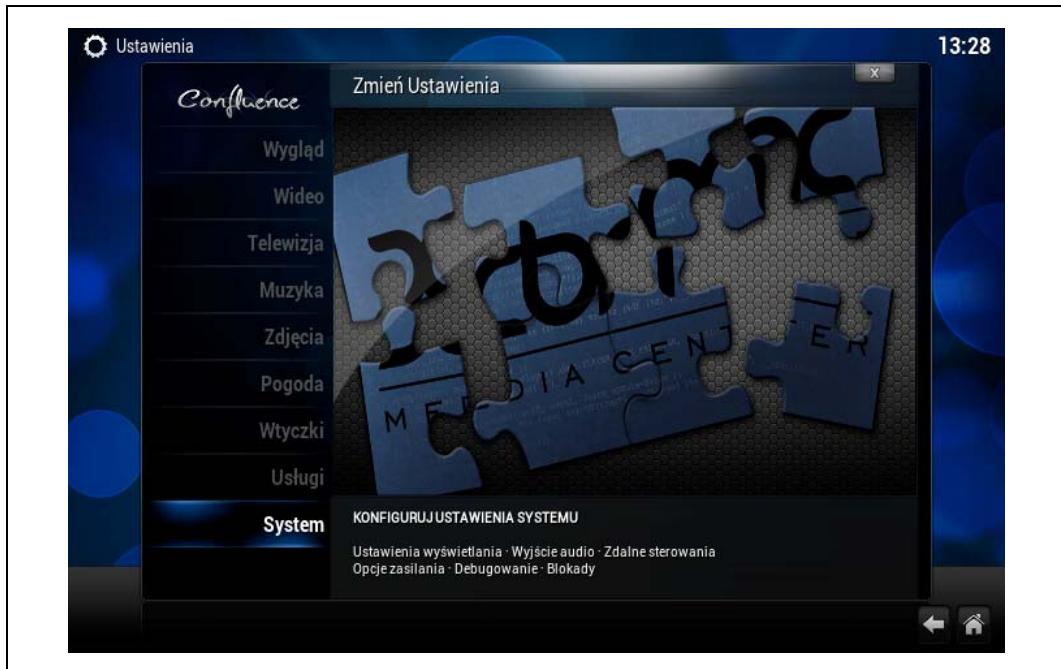
XBMC jest oprogramowaniem mającym wiele możliwości. Jego działanie najłatwiej sprawdzić, zapisując pliki z muzyką i filmami na dysku USB. Taki dysk należy podłączyć do Raspberry Pi. XBMC powinno odtworzyć pliki.

Prawdopodobnie używasz Raspberry Pi w sąsiedztwie telewizora. Może być on wyposażony w gniazdo USB, które może dostarczać prąd o natężeniu pozwalającym na zasilenie Raspberry Pi, a w takim przypadku nie musisz mieć zasilacza.

Warto rozważyć zakup zestawu składającego się z bezprzewodowej klawiatury i myszy. Odbiornik takiego zestawu będzie zajmował jedno gniazdo USB. Dzięki takiemu rozwiązaniu nie będziesz mieć w swoim pokoju płataniiny kabli. Innym dobrym rozwiązaniem może się okazać zakup klawiatury wyposażonej w gładzik.

Podłączenie Raspberry Pi do internetu za pomocą kabla jest rozwiązaniem pewniejszym, pozwalającym na uzyskanie większego transferu. Jednak urządzenie to nie zawsze można położyć w pobliżu gniazdka sieci Ethernet. Wtedy musisz skorzystać z kontrolera sieci Wi-Fi. XBMC obsługuje sieci bezprzewodowe po odpowiednim skonfigurowaniu.

Skonfigurowanie sieci radiowej w XBMC i Raspbmc jest łatwiejsze niż w systemie Raspbian lub Occidental (receptura 2.5) — XBMC ma przejrzysty interfejs użytkownika. W celu skonfigurowania połączenia sieciowego musisz przejść do menu *Ustawienia* (zobacz rysunek 4.2).



Rysunek 4.2. Konfiguracja systemu Raspbmc

Po przejściu do ustawień sieci Wi-Fi będziesz musiał podać jej nazwę i hasło dostępu.

Zobacz również

Zapoznaj się z poradnikiem użytkownika programu XBMC — <http://wiki.xbmcc.org/>.

Raspbmc nie jest jedyną dystrybucją systemu operacyjnego przeznaczoną do użytkowania w roli domowego centrum rozrywki. Istnieją jeszcze dwie inne dystrybucje:

- OpenElec (<http://www.openelec.tv/>),
- XBian (<http://www.xbian.org/>).

XBMCC może być obsługiwane za pomocą pilota — <https://learn.adafruit.com/using-an-ir-remote-with-a-raspberry-pi-media-center>.

4.2. Instalowanie oprogramowania biurowego

Problem

Chcesz otwierać dokumenty tekstowe i arkusze kalkulacyjne na swoim Raspberry Pi.

Rozwiązanie

Tak naprawdę Raspberry Pi jest komputerem działającym pod kontrolą systemu Linux, a więc istnieje kilka aplikacji biurowych, które możesz zainstalować w celu edycji dokumentów tekstowych i arkuszy kalkulacyjnych.

Raspberry Pi pobiera oprogramowanie z internetu, a zatem będziesz potrzebował połączenia z siecią.

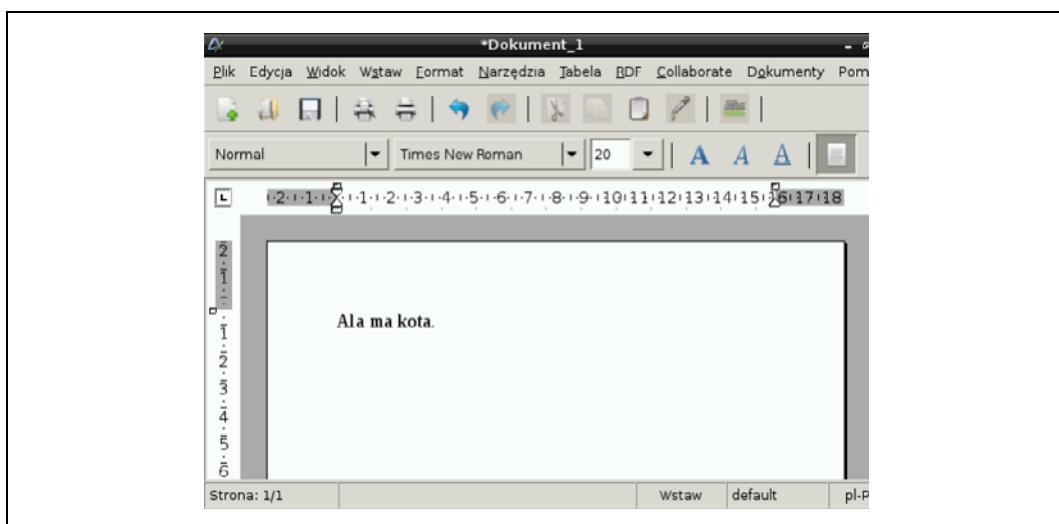
Przed instalacją jakiegokolwiek nowego oprogramowania warto otworzyć Terminal i wpisać w nim następujące polecenie:

```
$ sudo apt-get update
```

Aby zainstalować edytor AbiWord, uruchom polecenie:

```
$ sudo apt-get install abiword
```

Zostaniesz poproszony o potwierdzenie chęci zainstalowania tego programu. Wciśnij przycisk Y. Instalacja zajmie kilka chwil. Teraz w Twoim menu *Start* pojawi się sekcja *Biuro*. To w niej znajdziesz skrót do programu AbiWord (zobacz rysunek 4.3).



Rysunek 4.3. Edytor AbiWord

AbiWord potrafi otwierać popularne formaty dokumentów, takie jak np. *.doc* i *.docx*.

Jeżeli potrzebujesz również arkusza kalkulacyjnego, polecam program Gnumeric. Zainstalujesz go za pomocą poniższego polecenia.

```
$ sudo apt-get install gnumeric
```

Omówienie

Jeżeli aplikacje biurowe działają zbyt wolno, spróbuj przetaktować swoje Raspberry Pi (zobacz receptura 1.14). Zabieg ten z pewnością przyspieszy działanie aplikacji.

Zobacz również

Istnieją próby przeniesienia pakietu LibreOffice (wywodzącego się z pakietu OpenOffice) na platformę Raspberry Pi. Poszukaj w internecie najnowszych informacji dotyczących tego projektu.

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia apt-get.

4.3. Instalowanie innych przeglądarek internetowych

Problem

Chcesz korzystać z przeglądarki innej niż Midori.

Rozwiążanie

Na Raspberry Pi możesz zainstalować różne przeglądarki internetowe. Niestety nie jest to komputer dysponujący dużą mocą obliczeniową i oprogramowanie tego typu może go znacznie obciążać. Oznacza to, że działanie przeglądarki wymaga odpowiedniego kompromisu pomiędzy funkcjonalnością a wydajnością.

Przeglądarka Chromium (zobacz rysunek 4.4), tak jak sugeruje jej nazwa, przypomina przeglądarkę Google Chrome. Obsługuje ona praktycznie wszystko, ale przewijanie (góra – dół) rozbudowanych stron może być dość powolne. Poniżej przedstawiono polecenie, dzięki któremu zainstalujesz tę przeglądarkę. Skrót do niej zostanie umieszczony w menu *Start* w sekcji *Internet*.

```
$ sudo apt-get install chromium-browser
```

Inną popularną alternatywą dla Midori jest przeglądarka Iceweasel (zobacz rysunek 4.5). Program ten jest oparty na przeglądarce Firefox i będzie działał szybciej od Chromium, ponieważ jeśli tylko ma taką możliwość, korzysta z mobilnych wersji witryn, które nie mają zwykle bardzo złożonej budowy. Poniżej przedstawiono polecenie, dzięki któremu zainstalujesz tę przeglądarkę.

```
$ sudo apt-get install iceweasel
```

Omówienie

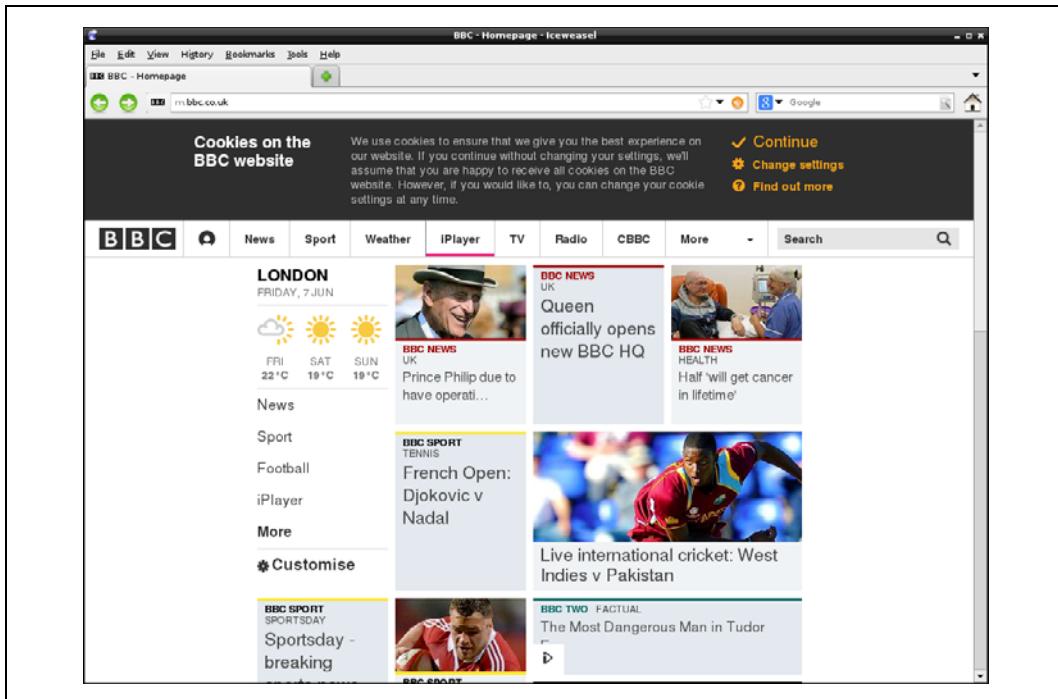
Przeglądarki internetowe wymagają dość dużej mocy obliczeniowej. Zakup Raspberry Pi model B wyposażonego w 512 MB pamięci podręcznej pozwoli na przeglądanie internetu w sposób wyraźnie szybszy. Dodatkowo warto przetaktować zegar procesora Raspberry Pi (zobacz receptura 1.14).

Zobacz również

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia apt-get.



Rysunek 4.4. Przeglądarka internetowa Chromium



Rysunek 4.5. Przeglądarka internetowa Iceweasel

4.4. Korzystanie z Pi Store

Problem

Chcesz instalować gry i oprogramowanie za pośrednictwem Pi Store.

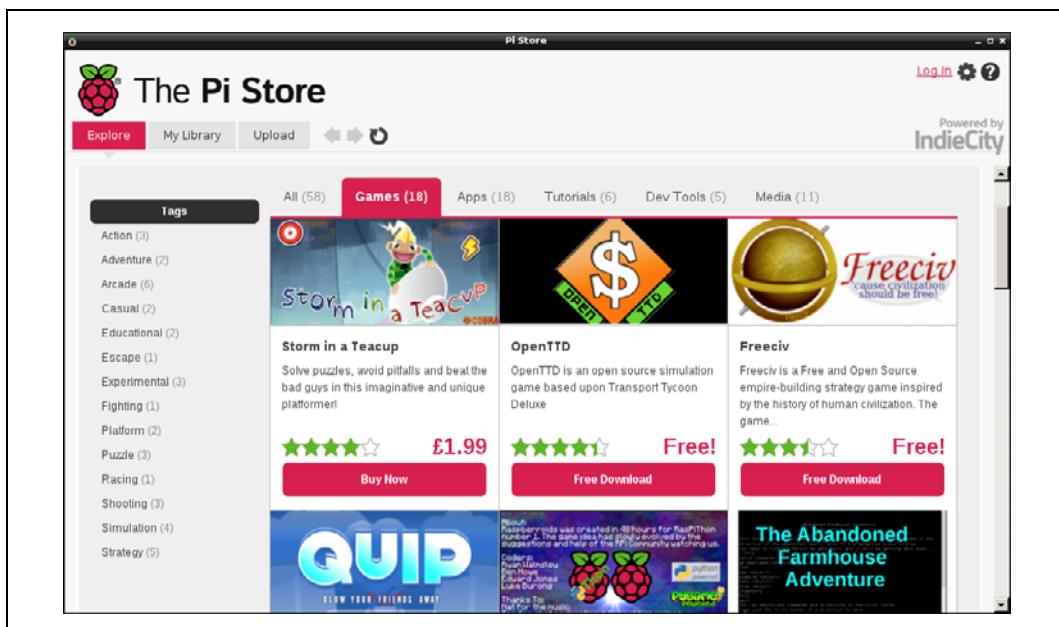
Rozwiążanie

Pi Store jest odpowiednikiem App Store (firmy Apple) i Play Store (firmy Google) — jest to miejsce, z którego możesz pobierać, instalować i uruchamiać różne aplikacje (zarówno darmowe, jak i płatne).

Jeżeli chcesz korzystać z Pi Store, najpierw musisz pobrać aplikację kliencką, która po uruchomieniu na Twoim Raspberry Pi pozwoli Ci na przeglądanie dostępnych aplikacji. Aby zainstalować ten program, otwórz sesję Terminala i uruchom poniższe polecenie.

```
$ sudo apt-get install pistore
```

Skrót do zainstalowanego oprogramowania zostanie automatycznie umieszczony na pulpicie (zobacz rysunek 4.6).



Rysunek 4.6. Pi Store

Przy pierwszej próbie pobrania jakieś aplikacji zostaniesz poproszony o rejestrację. Następnie aplikacja zostanie pobrana i wyświetlona w zakładce *My Library*. Kliknij ją dwukrotnie w celu jej uruchomienia.

Omówienie

Korzystanie z Pi Store jest wygodnym sposobem na znalezienie interesujących programów, które mogą zostać uruchomione na Raspberry Pi. Pi Store stale poszerza swój asortyment.

Zobacz również

Zajrzyj na oficjalną witrynę Pi Store — <http://store.raspberrypi.com/>.

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia apt-get.

4.5. Uruchamianie serwera kamery internetowej

Problem

Chcesz, żeby Raspberry Pi funkcjonowało jako serwer kamery sieciowej.

Rozwiążanie

Pobierz oprogramowanie motion, które pozwoli Ci na obsługę kamery podłączonej do gniazda USB — obraz widziany przez kamerę będziesz mógł oglądać za pomocą przeglądarki internetowej zainstalowanej na innym komputerze.

Aby zainstalować to oprogramowanie, wpisz w oknie Terminala poniższe polecenie.

```
$ sudo apt-get install motion
```

Podłącz kamerę internetową do gniazda USB i sprawdź, czy jest ona wykrywana przez Raspberry Pi, za pomocą polecenia lsusb:

```
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 3538:0059 Power Quotient International Co., Ltd
Bus 001 Device 006: ID eb1a:299f eMPIA Technology, Inc
```

Jeżeli patrząc na wyświetlzoną listę, nie jesteś pewien, czy kamera jest widziana przez Raspberry Pi, to wyciągnij wtyczkę kamery z gniazda USB i zobacz, czy jakiś element zniknął z listy. W przytoczonym przykładzie kamera jest ostatnim elementem wymienionym na liście.

Teraz musisz przeprowadzić pewne konfiguracje. Zacznij od otwarcia w edytorze pliku /etc/motion/motion.conf za pomocą polecenia:

```
$ sudo nano /etc/motion/motion.conf
```

Jest to dość obszerny plik konfiguracyjny. W górnej części pliku powinieneś znaleźć linię deamon off. Zmień ją na deamon on.

Pozostałe zmiany będą dokonywane w dużo dalszych częściach pliku. Zmień linię webcam_localhost = on na webcam_localhost = off.

Musisz dokonać zmian w jeszcze jednym pliku. Uruchom polecenie:

```
$ sudo nano /etc/default/motion
```

Zmień linię `start_motion_daemon=no` na `start_motion_daemon=yes`.

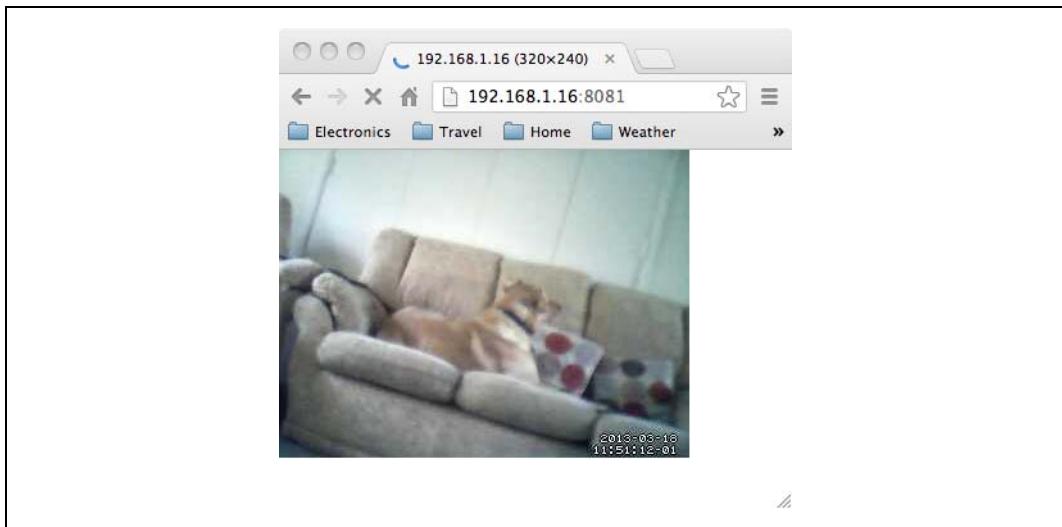
Uruchom usługę sieciową za pomocą polecenia:

```
$ sudo service motion start
```

Teraz powinieneś już mieć możliwość oglądania obrazu z kamery za pośrednictwem przeglądarki internetowej. W tym celu musisz znać adres IP swojego Raspberry Pi (zobacz receptura 2.2).

Uruchom przeglądarkę internetową na innym komputerze pracującym w tej samej sieci. W przeglądarce wpisz adres `http://192.168.1.16:8081/`. Oczywiście we wpisywanym adresie musisz podać adres IP Twojego Raspberry Pi. Na końcu adresu pozostaw :8081.

Jeżeli wszystko działa poprawnie, w przeglądarce powinieneś widzieć obraz z kamery (zobacz rysunek 4.7).



Rysunek 4.7. Obraz widziany przez kamerę podłączoną do Raspberry Pi

Omówienie

Oprogramowanie `motion` jest tak naprawdę bardzo rozbudowane i posiada wiele funkcji, które mogą wpływać na pracę Twojej kamery.

Domyślnie obraz z kamery będzie mógł być odbierany tylko przez komputery podłączone do Twojej sieci domowej. Jeżeli chcesz, aby dowolny komputer podłączony do internetu miał dostęp do Twojej kamery, to musisz ustawić odpowiednie *przekierowanie portu* na swoim domowym routerze. W tym celu zaloguj się do konsoli administracyjnej routera, znajdź opcję przekierowania portów i przekieruj port o numerze 8081 na adres IP Twojego Raspberry Pi.

Teraz będziesz mógł oglądać obraz rejestrowany przez kamerę, korzystając z zewnętrznego adresu IP Twojego routera. Adres ten jest zwykle widoczny na głównej stronie panelu administracyjnego. Uwaga! Jeżeli nie posiadasz stałego adresu IP, to adres ten będzie ulegał zmianie po każdorazowym uruchomieniu routera lub modemu.

Zobacz również

Dokładną dokumentację programu motion znajdziesz pod adresem <http://www.lavrsen.dk/foswiki/bin/view/Motion/MotionGuide>.

Istnieje specjalna kamera dedykowana dla Raspberry Pi (zobacz rysunek 1.18). W chwili pisania tej książki nie jest ona jeszcze kompatybilna z programem motion, jednak w chwili gdy ją czytasz, być może program jest już w stanie obsługiwać wspomnianą kamerę.

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia apt-get.

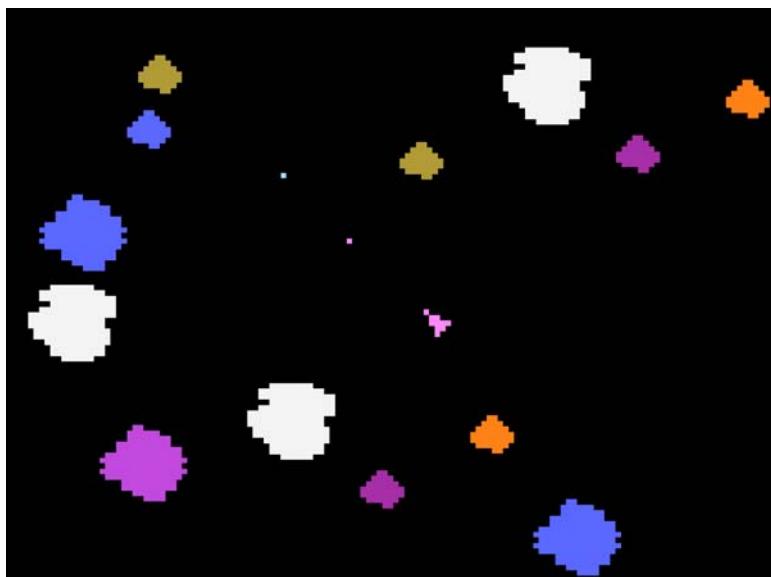
4.6. Uruchamianie emulatora klasycznej konsoli do gier

Problem

Chcesz korzystać z emulatora uruchamiającego gry przeznaczone na klasyczne konsole.

Rozwiążanie

Istnieje wiele emulatorów konsol do gier popularnych w latach 80. ubiegłego wieku. Jednym z najpopularniejszych programów tego typu jest **Stella** — emulator Atari 2600 (zobacz rysunek 4.8).



Rysunek 4.8. Gra Asteroids uruchomiona w programie Stella — emulatorze Atari 2600



Pamiętaj o tym, że ktoś posiada prawa autorskie nawet do tak starych gier. Pliki ROM z grami przeznaczone na emulatory takie jak Stella są łatwe do znalezienia w internecie. Nie oznacza to jednak, że ich używanie jest zgodne z prawem. Przestrzegaj prawa!

Aby zainstalować emulator Stella w oknie Terminala, wprowadź następujące polecenie:

```
$ sudo apt-get install stella
```

Po zainstalowaniu w menu *Start* w grupie *Gry* znajdziesz skrót do tej aplikacji. Nie uruchamiaj jeszcze tego programu. Najpierw musisz znaleźć pliki ROM zawierające obrazy gier.

Jeżeli mieszkasz poza terytorium Stanów Zjednoczonych i masz oryginalną grę, to w większości krajów masz prawo do posiadania jej kopii zapasowej w postaci pliku ROM. Możesz znaleźć również obrazy gier, do których prawa nie zostały zastrzeżone.

Skoro masz już obraz poszukiwanej przez Ciebie gry, stwórz folder o nazwie *romy*, w którym będziesz przechowywać pliki ROM. Teraz możesz uruchomić program Stella.

Aby uruchomić grę, kliknij plik obrazu. Domyślnie sterowanie odbywa się za pomocą klawiszy strzałek, a spacji przypisano funkcję klawisza spustu. Emulator posiada wiele opcji. Z pewnością zechcesz zapoznać się z ustawieniami wideo i włączyć tryb pełnoekranowy (ang. *fullscreen*).

Jeśli chcesz zmienić klawisze używane podczas gry, zajrzyj do zakładki *Emul Events* znajdującej się w menu *Input Settings*.

Omówienie

Emulator korzysta z zadziwiająco dużej ilości zasobów sprzętowych, a więc prawdopodobnie będziesz musiał przetaktować Raspberry Pi w celu uzyskania odpowiedniej wydajności pracy programu (zobacz receptura 1.14).

W internecie można znaleźć projekty, w których ludzie podłączyli do Raspberry Pi dość tanie kontrolery przypominające swym wyglądem klasyczne pady (np. Nintendo Retrolink USB Super SNES Classic Controller), a urządzenie wraz z monitorem zostało umieszczone w dużej obudowie przypominającej automat do gier.

Zobacz również

Istnieje wiele innych emulatorów konsol przeznaczonych dla Raspberry Pi. Programy te są w różnych stadiach rozwoju, a więc ich funkcjonalność może być czasami ograniczona. Programem wartym uwagi jest **Mame** (<http://www.mamedev.org/>) — emuluje on wiele różnych platform.

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia `apt-get`.

4.7. Uruchamianie gry Minecraft

Problem

Chcesz uruchomić popularną grę *Minecraft* na swoim Raspberry Pi.

Rozwiążanie

Firma Mojang — wykonawca gry *Minecraft* — przeniosła ją na platformę Raspberry Pi.

Aby zainstalować tę grę, musisz mieć system operacyjny Raspbian (zobacz receptura 1.4). W celu zainstalowania gry (zobacz rysunek 4.9) wpisz poniższe polecenie.

```
$ wget https://s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz  
$ tar -zxvf minecraft-pi-0.1.1.tar.gz  
$ cd mcpi  
$ ./minecraft-pi
```



Rysunek 4.9. Gra Minecraft uruchomiona na Raspberry Pi

Omówienie

Twórcy gry *Minecraft* w celu przeniesienia jej na platformę Raspberry Pi musieli wnieść pewne ograniczenia do jej kodu graficznego. W grę można grać tylko na Raspberry Pi za pomocą bezpośrednio podłączonych urządzeń periferyjnych, takich jak klawiatura, mysz i monitor. Nie możesz w tym celu korzystać z połączenia sieciowego.

Zobacz również

Więcej informacji na temat gry *Minecraft* przeznaczonej na platformę Raspberry Pi znajdziesz pod adresem <http://pi.minecraft.net/>.

4.8. Uruchamianie gry Open Arena

Problem

Chcesz uruchomić pochodną gry *Quake* — grę *Open Arena*.

Rozwiążanie

Pobierz grę *Open Arena* z serwisu Pi Store.



Rysunek 4.10. Gra Open Arena uruchomiona na Raspberry

Omówienie

Aplikację *Open Arena* znajdziesz w dziale *Games* serwisu Pi Store. Uwaga! Gra jest dość brutalna i krwawa.

Zobacz również

Więcej informacji na temat gry *Open Arena* znajdziesz na stronie <http://www.openarena.ws/smfnnews.php>.

Więcej informacji na temat serwisu Pi Store znajdziesz w recepturze 4.4.

4.9. Raspberry Pi jako nadajnik radiowy

Problem

Chcesz zamienić swoje Raspberry Pi w nadajnik FM dużej mocy. Oczekujesz, że nadawany sygnał będzie odbierany za pomocą zwykłych odbiorników radiowych (zobacz rysunek 4.11).



Rysunek 4.11. Raspberry Pi w roli nadajnika radiowego

Rozwiązanie

Sprytni kolesie z londyńskiego Imperial College stworzyli kod w języku C, który został opakowany kodem Pythona pozwalającym na przekształcenie Raspberry Pi w nadajnik FM. Do stworzonego oprogramowania dołączyli oni nawet próbkę dźwięku w postaci motywu przewodniego z filmu *Gwiezdne wojny*.

Będziesz potrzebował krótkiego kabla podłączonego do styku o numerze 4 złącza GPIO. Dobrze sprawdzi się tutaj kabel zakończony dwoma żeńskimi końcówkami przeznaczonymi do pinów. Tak naprawdę nadawany sygnał jest tak silny, że powinien być odbierany przez radio stojące w pobliżu Raspberry Pi nawet bez zastosowania anteny.

Przekształcanie Raspberry Pi w nadajnik radiowy zaczni od zainstalowania biblioteki `pifm`. W tym celu skorzystaj z poniższego polecenia.

```
$ mkdir pifm  
$ cd pifm  
$ wget http://www.icrobotics.co.uk/wiki/images/c/c3/Pifm.tar.gz  
$ tar -xzf Pifm.tar.gz
```

Następnie przygotuj jakiś radioodbiornik i ustaw go na częstotliwość 103,0 MHz. Jeżeli na tej częstotliwości odbierana jest jakaś stacja radiowa, znajdź jakąś inną wolną częstotliwość i zapisz ją.

Teraz uruchom poniższe polecenie. Jeżeli chcesz nadawać sygnał na innej częstotliwości niż 103,0 MHz, to zmień wartość ostatniego parametru.

```
sudo ./pifm sound.wav 103.0
```

Jeżeli wszystko działa poprawnie, z Twojego radioodbiornika powinien dobiegać motyw przewodni z filmu *Gwiezdne wojny*.

Omówienie

W niektórych państwach używanie radionadajników bez odpowiednich zezwoleń jest niewegalne. Moc nadajnika opartego na Raspberry Pi jest znacznie większa od mocy standardowych nadajników używanych z odtwarzaczami plików MP3.

Możesz odtwarzać również inne pliki *.wav*, ale muszą to być pliki monofoniczne zapisane z rozdzielczością 16 bitów i częstotliwością próbkowania 44,1 kHz.

Do kodu dołączona jest biblioteka, którą możesz stosować w swoich własnych programach napisanych w Pythonie. Możesz więc stworzyć aplikację przeznaczoną do nadawania muzyki, zawierającą graficzny interfejs użytkownika.

Poniższy kod ilustruje zastosowanie interfejsu Pythona.

```
pi@raspberrypi ~$ sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import PiFm
>>> PiFm.play_sound("sound.wav")
```

Gdybyś korzystał z Raspberry Pi w samochodzie, to właśnie w ten sposób mógłbyś odtwarzać dźwięk przez samochodowy system nagłośnieniowy.

Zobacz również

Niniejsza sekcja powstała w oparciu o artykuł napisany przez autorów pomysłu na przekształcenie Raspberry Pi w nadajnik radiowy — http://www.icr robotics.co.uk/wiki/index.php/Turning_the_Raspberry_Pi_Into_an_FM_Transmitter.

4.10. Uruchamianie edytora grafiki GIMP

Problem

Chcesz pracować w edytorze grafiki.

Rozwiążanie

Ściagnij program GIMP (ang. *GNU Image Manipulation Program*) — zobacz rysunek 4.12.

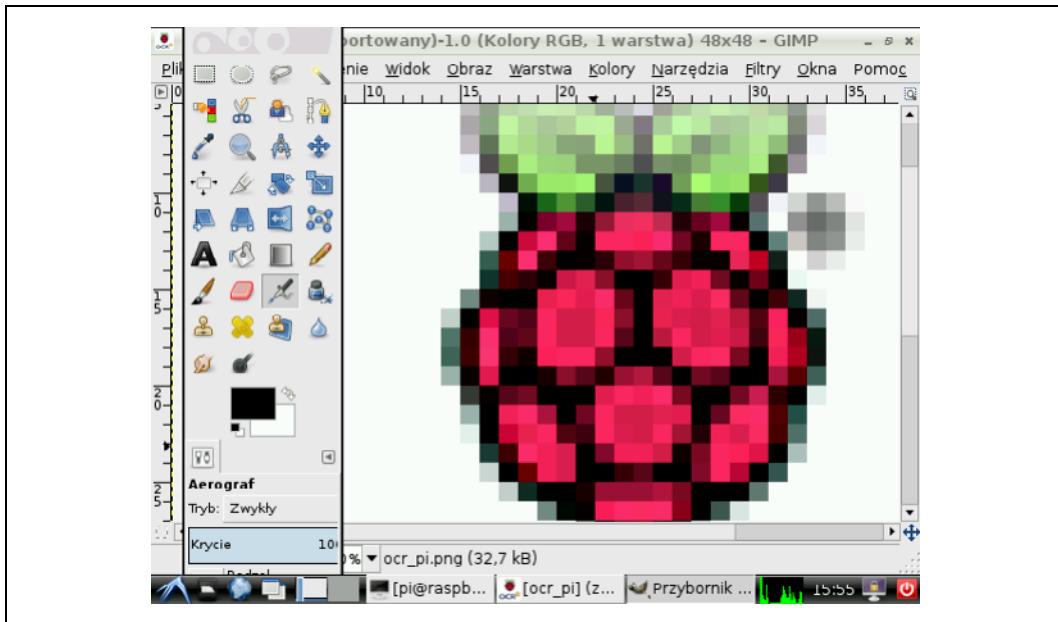
Aby zainstalować program GIMP, otwórz sesję Terminala i wpisz następujące polecenie:

```
$ sudo apt-get install gimp
```

Po zainstalowaniu aplikacji w menu *Start* pojawi się nowa sekcja — *Grafika*. To właśnie tam będzie się znajdował skrót do programu GIMP.

Omówienie

GIMP znacznie obciąża procesor i pamięć. Program ten działa jednak całkiem sprawnie na Raspberry Pi model B.



Rysunek 4.12. GIMP uruchomiony na Raspberry Pi

Zobacz również

Więcej informacji na temat edytora graficznego GIMP znajdziesz na jego witrynie internetowej — <http://www.gimp.org/>.

Edytor ten jest bardzo rozbudowany i posiada wiele funkcji, a więc nauka jego obsługi może się okazać czasochłonna. We wspomnianej wcześniej witrynie w zakładce *Documentation* znajdziesz obszerny podręcznik użytkownika.

Zajrzyj do receptury 3.16, aby uzyskać więcej informacji na temat polecenia apt-get.

4.11. Radio internetowe

Problem

Chcesz słuchać radia internetowego za pomocą swojego Raspberry Pi.

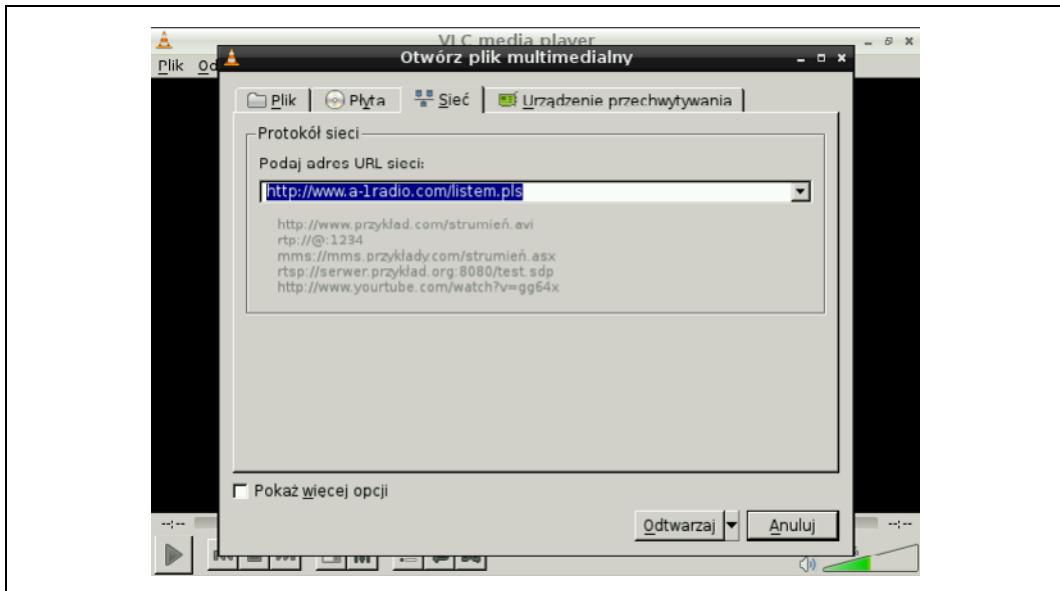
Rozwiążanie

Zastosuj poniższe polecenie i zainstaluj program **VLC media player**.

```
sudo apt-get install vlc
```

Skrót do tego programu zostanie automatycznie umieszczony w menu *Start* w sekcji *Dźwięk i obraz*.

Uruchom program i w menu *Plik* wybierz opcję *Otwórz strumień w sieci*. Otworzyesz w ten sposób okno dialogowe (zobacz rysunek 4.13). Wprowadź w nim adres internetowej radiostacji, której chcesz posłuchać.



Rysunek 4.13. Odtwarzacz multimedialny VLC uruchomiony na Raspberry Pi

Do Raspberry Pi będziesz musiał jeszcze podłączyć słuchawki lub głośniki wyposażone we wzmacniacz.

Omówienie

VLC media player możesz uruchomić również z poziomu wiersza poleceń:

```
$ vlc http://www.a-1radio.com/listen.pls -I dummy
```

Program początkowo wyświetli serię komunikatów informujących Cię o różnych błędach, ale później dźwięk będzie odtwarzany poprawnie.

Zobacz również

Niniejsza sekcja została oparta na projekcie Jana Holsta (<http://www.jan-holst.dk/pi-radio/pi-radio.html>), który poszedł o krok dalej i dodał do Raspberry Pi pokrętła sterujące przypominające swym wyglądem pokrętła umieszczone na radioodbiornikach.

Na forum serwisu *chuby.com* w wątku <http://forum.chubby.com/viewtopic.php?id=7054> znajduje się lista adresów URL rozgłośni radiowych BBC strumieniowanych w sieci.

Podstawy Pythona

5.0. Wprowadzenie

Programy uruchamiane na Raspberry Pi mogą być pisane w wielu językach, jednak najpopularniejszym z nich jest Python. Drugi człon nazwy Raspberry Pi pochodzi zresztą od słowa Python.

W tym rozdziale znajdziesz wiele receptur, które pomogą Ci rozpocząć programowanie na Raspberry Pi.

5.1. Wybór pomiędzy Pythonem 2 a 3

Problem

Chcesz stosować Pythona, ale nie wiesz, którą wersję wybrać.

Rozwiążanie

Stosuj obie wersje. Korzystaj z Pythona 3 do czasu, aż natkniesz się na problem, który łatwiej rozwiązać, stosując starszą wersję.

Omówienie

Python 3 jest najnowszą wersją tego języka, która jest już obecna na rynku od kilku lat. Pomimo to wielu użytkowników wciąż korzysta z Pythona w wersji 2. Systemy operacyjne Raspbian i Occidental zazwyczaj zawierają Pythona zarówno w wersji 2 (nazwanej po prostu *Pythonem*), jak i w wersji 3 (nazwanej *Pythonem 3*). Najnowszą wersję Pythona uruchomisz, stosując polecenie `python3`. Wszystkie przykłady przedstawione w tej książce, o ile nie zaznaczono inaczej, są napisane w Pythonie 3. Większość z nich można uruchomić bez dokonywania dodatkowych modyfikacji za pomocą obydwu wersji Pythona.

Niechęć części społeczności do całkowitego odejścia od Pythona 2 wynika z tego, że wersja 3 wprowadziła pewne zmiany, które sprawiły, że nie jest ona kompatybilna ze starszą wersją. Oznacza to, że wiele bibliotek przeznaczonych dla Pythona 2, które stworzyli użytkownicy, nie może być użytych w programach pisanych w nowszej wersji Pythona.

Obrałem strategię pisania programów w nowszej wersji Pythona, o ile jest to możliwe. Gdy spotykam się z problemami z kompatybilnością, tworzę kod zgodny z Pythonem 2.

Zobacz również

Różnice pomiędzy obiema wersjami Pythona podsumowano w artykule <https://wiki.python.org/moin/Python2orPython3>.

5.2. Pisanie aplikacji Pythona za pomocą IDLE

Problem

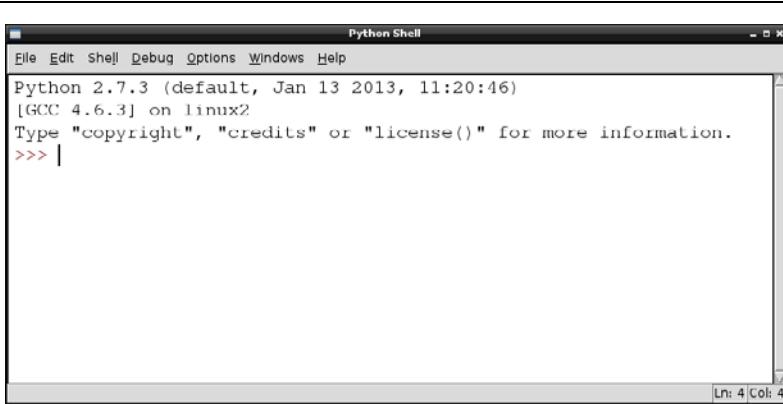
Nie wiesz, jakimi narzędziami należy się posługiwać podczas pisania programów w Pythonie.

Rozwiązanie

Większość dystrybucji systemów operacyjnych przeznaczonych dla Raspberry Pi zawiera narzędzie programistyczne o nazwie IDLE, które pozwala na pisanie programów w obu wersjach języka Python. Jeżeli korzystasz z dystrybucji Raspbian lub Occidentalis, to skrót do obu wersji środowiska IDLE znajdziesz bezpośrednio na pulpicie.

Omówienie

IDLE i IDLE3 wyglądają identycznie. Różnią się jedynie wersją języka programowania, jakiej można w nich używać. Otwórz konsolę IDLE3 (zobacz rysunek 5.1).



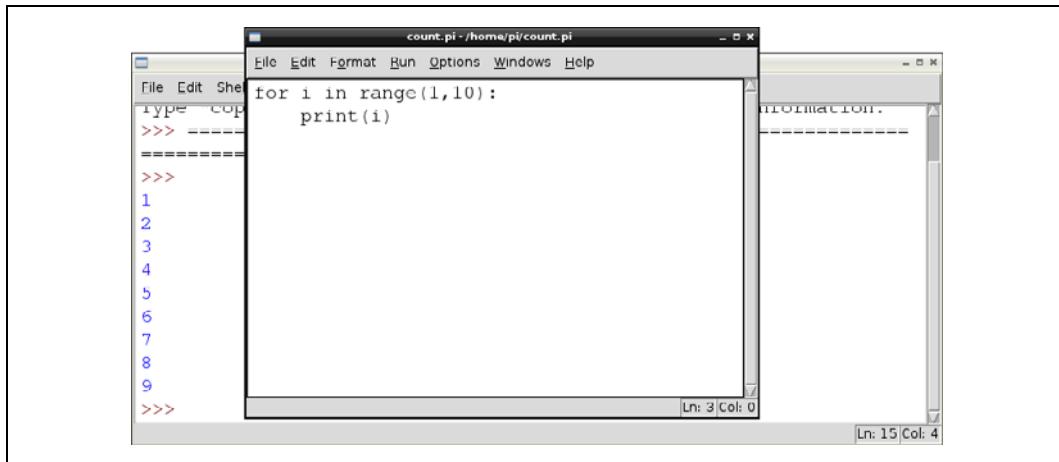
Rysunek 5.1. Konsola IDLE

Otwarte okno nosi nazwę *Python Shell*. Jest to interaktywna przestrzeń, w której możesz korzystać z Pythona i natychmiast widzieć skutki modyfikacji wpisywanego kodu. Wpisz tuż za znakiem zachęty (`>>>`) poniższe działanie:

```
>>> 2 + 2
4
>>>
```

Python rozwiązał działanie (`2 + 2`) i podał jego wynik (4).

Okno *Python Shell* jest doskonałym rozwiązyaniem w przypadku sprawdzania działania różnych rzeczy, ale jeżeli chcesz pisać programy, to powinieneś skorzystać z edytora. Otwórz go, wybierając opcję *New Window* z menu *File* (zobacz rysunek 5.2).



Rysunek 5.2. Edytor IDLE

W oknie edytora możesz wprowadzać kod swojego programu. W celu sprawdzenia działania tego okna umieść w nim poniższy kod i zapisz plik pod nazwą *count.py*. Następnie z menu *Run* wybierz opcję *Run Module*. Efekt działania programu zostanie wyświetlony w konsoli Pythona.

```
for i in range(1, 10):
    print(i)
```

Pythona od większości języków programowania odróżnia to, że wcięcia są istotnym elementem kodu. W wielu językach programowania podobnych do języka C bloki kodu są oznaczane za pomocą znaków `{ i }`. W Pythonie w tym samym celu stosuje się wcięcia. W zaprezentowanym przykładzie Python wie, że polecenie `print` powinno być ciągle wywoływanie w ramach pętli `for`, bo jest wcięte — znajdują się przed nim cztery spacje.

W tej książce zastosowaliśmy konwencję polegającą na oznaczaniu wcięcia za pomocą czterech znaków spacji.



Kiedy rozpoczynasz przygodę z programowaniem w Pythonie, dość często możesz się spotkać z błędem `IndentationError: unexpected indent`, który będzie oznaczał, że któryś z wierszy programu zawiera nieprawidłowe wcięcie. Jeżeli kod wydaje się napisany poprawnie, to sprawdź go jeszcze raz — być może któryś z wierszy zawiera znak tabulacji. Python rozróżnia spacje i znaki tabulacji.

Zwróć uwagę na to, że środowisko IDLE oznacza pewne elementy kodu odpowiednimi kolorami.

Zobacz również

W wielu recepturach zamieszczonych w tym rozdziale IDLE jest stosowane do modyfikacji przykładowych kodów napisanych w Pythonie.

Kod programów możesz również pisać w edytorze nano (receptura 3.6), a następnie uruchamiać go w sesji Terminala (receptura 5.4).

5.3. Korzystanie z konsoli Pythona

Problem

Chcesz wprowadzać pojedyncze polecenia Pythona, nie pisząc całego programu.

Rozwiązanie

Korzystaj z konsoli Pythona za pomocą środowiska IDLE (receptura 5.2) lub sesji Terminala.

Omówienie

Aby uruchomić konsolę Pythona 2 w oknie Terminala, wystarczy wpisać polecenie `python`. Jeśli chcesz uruchomić konsolę Pythona 3, powinieneś zastosować polecenie `python3`.

Znak zachęty `>>>` sygnalizuje możliwość wpisywania poleceń Pythona. Jeżeli musisz wpisać polecenia składające się z wielu wierszy, konsola automatycznie zapewni kontynuację polecenia, oznaczając takie wiersze symbolem trzech kropiek. Musisz samodzielnie zadbać o odpowiednie wcięcie takich wierszy, stosując cztery znaki spacji, jak widać poniżej.

```
>>> for i in range(1, 10):
...     print(i)
...
1
2
3
4
5
6
7
8
9
>>>
```

Na końcu wpisanej komendy wciśnij dwukrotnie klawisz *Enter* — pozwoli to na rozpoznanie końca wcięcia bloku i uruchomienie kodu.

Konsola Pythona pozwala na przeglądanie historii wpisywanych poleceń za pomocą klawiszy strzałek (do góry i do dołu).

Zobacz również

Jeżeli kod, który chcesz wpisać, jest dłuższy niż kilka linii, to prawdopodobnie lepiej by było, gdybyś skorzystał ze środowiska IDLE (zobacz receptura 5.2).

5.4. Uruchamianie programów napisanych w Pythonie za pomocą Terminala

Problem

Programy można uruchamiać bezpośrednio ze środowiska IDLE, ale czasami może zaistnieć konieczność uruchomienia programu w oknie Terminala.

Rozwiązanie

Skorzystaj z poleceń `python` lub `python3`, po czym podaj nazwę pliku zawierającego program, który chcesz uruchomić.

Omówienie

W celu uruchomienia programu napisanego w Pythonie 2 musisz wprowadzić polecenie o następującej składni:

```
$ python mójprogram.py
```

Jeżeli chcesz uruchomić program w nowszej wersji Pythona, zamiast polecenia `python` zastosuj polecenie `python3`. Plik zawierający program, który chcesz uruchomić, powinien mieć rozszerzenie `.py`.

Większość programów Pythona można uruchomić jako zwykły użytkownik. Jednak niektóre aplikacje, zwłaszcza te, które korzystają z portu GPIO, będziesz musiał uruchamiać jako administrator. Polecenie uruchamiające taki program należy poprzedzić prefiksem `sudo`.

```
$ sudo python mójprogram.py
```

Zobacz również

Möżesz zaplanować automatyczne uruchomienie programu napisanego w Pythonie (zobacz receptura 3.21).

Program taki może być również automatycznie uruchamiany podczas startu Raspberry Pi (zobacz receptura 3.20).

5.5. Zmienne

Problem

Chcesz nazwać jakąś wartość.

Rozwiązanie

Nadaj nazwę wartości za pomocą operatora `=`.

Omówienie

Pracując w Pythonie, nie musisz deklarować typu zmiennej. Wystarczy ją po prostu zainicjować, jak pokazano poniżej:

```
a = 123  
b = 12.34  
c = "Witaj"  
d = 'Witaj'  
e = True
```

Łańcuchy znaków można deklarować, korzystając z pojedynczego lub podwójnego cudzysłowu. Wartości logiczne definiujemy jako `True` (prawda) i `False` (fałsz). Program rozróżnia wielkie i małe litery.

Istnieje konwencja mówiąca o tym, że nazwy zmiennych powinny się zaczynać małą literą. Jeżeli składają się z więcej niż jednego słowa, to słowa te powinny być rozdzielone znakiem podkreślenia. Warto nadawać zmiennym nazwy, które będą dla Ciebie zrozumiałe.

Przykładami dobrych nazw dla zmiennych są: `x`, `suma` i `liczba_znaków`.

Zobacz również

Zmiennym można przypisać elementy takie jak listy (zobacz receptura 6.1) oraz słowniki (zobacz receptura 6.12).

Zasady działań arytmetycznych wykonywanych na zmiennych omówiono w recepturze 5.8.

5.6. Wyświetlanie danych generowanych przez program

Problem

Chcesz wyświetlić wartość jakiejś zmiennej.

Rozwiążanie

Skorzystaj z polecenia `print`. Wypróbuj działanie poniższego kodu — wpisz go w konsoli Pythona (receptura 5.3).

```
>>> x = 10  
>>> print(x)  
10  
>>>
```

Omówienie

W Pythonie 2 nie trzeba ujmować w nawias argumentu funkcji `print`. Jednak Python 3 wymaga już zastosowania nawiasów w tym miejscu. Aby kod był kompatybilny z obiema wersjami, wyświetlana wartość wpisaliśmy w nawiasie.

Zobacz również

W recepturze 5.7 opisano wczytywanie do programu danych wprowadzanych przez użytkownika.

5.7. Wczytywanie danych wprowadzonych przez użytkownika

Problem

Chcesz, żeby użytkownik programu mógł wprowadzić do niego liczbę.

Rozwiązanie

Skorzystaj z polecenia `input` (Python 3) lub `raw_input` (Python 2). Wypróbuj działanie powyższego przykładu w konsoli.

```
>>> x = input("Wprowadź wartość")
Wprowadź wartość: 23
>>> print(x)
23
>>>
```

Omówienie

Jeżeli korzystasz z Pythona 2, to polecenie `input` zastąp poleceniem `raw_input`.

Python 2 również posiada funkcję `input`, ale służy ona do sprawdzania danych wejściowych i zamiany ich na wartość odpowiedniego typu. Natomiast funkcja `raw_input` działa dokładnie w ten sam sposób co funkcja `input` w Pythonie 2 — po prostu wczytuje łańcuch.

Zobacz również

Więcej informacji na temat funkcji `input` w Pythonie 2 znajdziesz w internecie — <http://docs.python.org/2/library/functions.html#input>.

5.8. Działania arytmetyczne

Problem

Chcesz wykonywać działania arytmetyczne w Pythonie.

Rozwiązanie

Korzystaj z operatorów `+`, `-`, `*` i `/`.

Omówienie

Najczęściej stosowanymi operatorami są: `+` (operator dodawania), `-` (operator odejmowania), `*` (operator mnożenia) i `/` (operator dzielenia).

Działania można również grupować za pomocą nawiasów — tak jak pokazano w poniższym przykładzie, w którym użytkownik wprowadza temperaturę wyrażoną w stopniach Celsjusza, a program zamienia podaną wartość na skalę Fahrenheita.

```
>>> tempC = input("Podaj temperaturę w stopniach Celsjusza: ")
Podana temperatura w stopniach Celsjusza: 20
>>> tempF = (int(tempC) * 9) / 5 + 32
>>> print(tempF)
68.0
>>>
```

Inne operatory arytmetyczne to: % (dzielenie modulo) i ** (potęgowanie). W celu podniesienia liczby 2 do ósmej potęgi należy zastosować operator ** w następujący sposób:

```
>>> 2 ** 8
256
```

Zobacz również

Słosowanie polecenia `input` omówiono szerzej w recepturze 5.7. W tej samej recepturze opisano również proces konwersji wprowadzonego łańcucha znaków na wartość liczbową.

Pod adresem <https://docs.python.org/3.0/library/math.html> znajdziesz informacje dotyczące biblioteki `Math`, która zawiera wiele przydatnych operatorów matematycznych.

5.9. Tworzenie łańcuchów

Problem

Chcesz stworzyć zmienną będącą łańcuchem.

Rozwiązanie

Aby stworzyć nowy łańcuch, skorzystaj z operatora przypisania oraz stałej będącej łańcuchem. łańcuch możesz umieścić w pojedynczych lub podwójnych cudzysłowach.

Na przykład:

```
>>> s = "abc def"
>>> print(s)
abc def
>>>
```

Omówienie

Jeżeli chcesz umieścić w łańcuchu symbole pojedynczego lub podwójnego cudzysłowu, to początek i koniec łańcucha oznacz cudzysłowem, który nie znajduje się w treści łańcucha. Poniższy przykład ilustruje łańcuch będący zdaniem pytającym w języku angielskim. W łańcuchu tym znajduje się apostrof ('):

```
>>> s = "Isn't it warm?"
>>> print(s)
Isn't it warm?
>>>
```

Czasami będziesz chciał umieścić w łańcuchu **znaki specjalne**, takie jak znak tabulacji lub znak nowego wiersza. Będziesz musiał zastosować ich symbole zastępcze — znak tabulacji należy zapisać jako `\t`, a znak nowego wiersza jako `\n`. Na przykład:

```
>>> s = "name\tage\nMateusz\t14"
>>> print(s)
name      age
Mateusz  14
>>>
```

Zobacz również

Pełną listę znaków specjalnych znajdziesz pod adresem <https://docs.python.org/release/2.5.2/ref/strings.html>.

5.10. Scalanie (łączenie) łańcuchów

Problem

Chcesz połączyć ze sobą kilka łańcuchów.

Rozwiążanie

Zastosuj operator łączenia (+).

Na przykład:

```
>>> s1 = "abc"
>>> s2 = "def"
>>> s = s1 + s2
>>> print(s)
abcdef
>>>
```

Omówienie

W wielu językach programistycznych próba połączenia ze sobą łańcuchów będących znakami i łańcuchów będących wartościami spowoduje automatyczną konwersję wartości liczbowych na łańcuch znaków (ciąg cyfr). Jednak w Pythonie taka konwersja nie ma miejsca. Próba uruchomienia poniższego polecenia zakończy się wyświetleniem komunikatu o błędzie.

```
>>> "abc" + 23
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Każdy element musi być przekształcony na łańcuch znaków przed wykonaniem operacji scalania, tak jak pokazano w poniższym przykładzie.

```
>>> "abc" + str(23)
'abc23'
>>>
```

Zobacz również

Więcej informacji na temat konwersji wartości liczbowych na łańcuchy znaków za pomocą funkcji `str` znajdziesz w receptorze 5.11.

5.11. Konwersja liczb na łańcuchy

Problem

Chcesz dokonać konwersji liczby na łańcuch.

Rozwiązanie

Zastosuj funkcję `str`. Na przykład:

```
>>> str(123)
'123'
>>>
```

Omówienie

Liczبę często zamienia się na łańcuch znaków po to, aby połączyć go z innym łańcuchem (zobacz receptura 5.10).

Zobacz również

Operację odwrotną (konwersję łańcucha na zmienną liczbową) przedstawiono w recepturze 5.12.

5.12. Konwersja łańcuchów na liczby

Problem

Chcesz dokonać konwersji łańcucha na liczbę.

Rozwiązanie

Skorzystaj z funkcji `int` lub `float`.

Na przykład w celu przekształcenia ciągu znaków `-123` na liczbę mógłbyś zastosować poniższe polecenie:

```
>>> int("-123")
-123
>>>
```

Polecenie to będzie działać z dodatnimi i ujemnymi liczbami całkowitymi.

Aby dokonać konwersji liczby zmiennoprzecinkowej, należy skorzystać z polecenia `int`:

```
>>> float("00123.45")
123.45
>>>
```

Omówienie

Polecenia `int` i `float` potrafią poradzić sobie z zerami znajdującymi się na początku liczby, a także spacjami i innymi białymi znakami otaczającymi liczbę.

Polecenie `int` można również stosować do konwersji łańcucha zawierającego liczbę w systemie innym niż dziesiętny. Podstawę systemu należy podać jako drugi argument polecenia. Poniższy przykład przedstawia konwersję liczby 1001, która jest zapisana w systemie binarnym.

```
>>> int("1001", 2)
9
>>>
```

Kolejny przykład przedstawia konwersję liczby AFF0, która jest zapisana w systemie szesnastkowym.

```
>>> int("AFF0", 16)
45040
>>>
```

Zobacz również

Operację odwrotną, polegającą na konwersji zmiennej liczbowej na łańcuch znaków, omówiono w recepturze 5.11.

5.13. Ustalanie długości łańcucha

Problem

Chcesz wiedzieć, z ilu znaków składa się łańcuch.

Rozwiążanie

Skorzystaj z funkcji `len`.

Omówienie

Poniższy przykład przedstawia proces ustalania długości łańcucha `abcdef`.

```
>>> len("abcdef")
6
>>>
```

Zobacz również

Funkcję `len` można stosować również w kontekście tablic (zobacz receptura 6.3).

5.14. Ustalanie pozycji łańcucha w łańcuchu

Problem

Musisz ustalić pozycję łańcucha znajdującego się w innym łańcuchu.

Rozwiążanie

Skorzystaj z funkcji `find`.

W celu znalezienia np. pozycji łańcucha `def`, który jest częścią łańcucha `abcdefghi`, należy zastosować następujące polecenie:

```
>>> s = "abcdefghijkl"  
>>> s.find("def")  
3  
>>>
```

Pamiętaj o tym, że znaki tworzące łańcuch są numerowane od zera, a więc znak znajdujący się na pozycji numer trzy będzie czwartym znakiem tego łańcucha.

Omówienie

Jeżeli łańcuch znaków, którego szukasz, nie znajduje się wewnątrz przeszukiwanego łańcucha, to funkcja `find` zwróci wartość `-1`.

Zobacz również

W celu znalezienia i wymienienia jakiegoś łańcucha powtarzającego się wielokrotnie wewnątrz innego łańcucha możesz skorzystać z funkcji `replace` (zobacz receptura 5.16).

5.15. Wydobywanie fragmentu łańcucha

Problem

Chcesz wyciąć z łańcucha pewien jego fragment znajdujący się pomiędzy określonymi znakami.

Rozwiążanie

Skorzystaj z notacji `[:]`.

Na przykład w celu wycięcia fragmentu łańcucha `abcdefghijkl` znajdującego się pomiędzy drugim a piątym znakiem należy użyć polecenia:

```
>>> s = "abcdefghijkl"  
>>> s[1:5]  
'bcde'  
>>>
```

Zwróć uwagę na to, że pierwszy element łańcucha znajduje się na pozycji o numerze 0, a więc na pozycji o numerze 1 znajduje się drugi element łańcucha, a na pozycji o numerze 5 jego szósty

element. Litera `f` nie została umieszczona w łańcuchu wyjściowym tej operacji, ponieważ znak znajdujący się na pozycji określonej przez większą liczbę definiującą zakres wyodrębnianych znaków nie jest wyodrębniany z łańcucha.

Omówienie

Notacja `[:]` może być stosowana także w inny sposób. Możesz pominąć podanie któregoś z argumentów, co spowoduje automatyczne rozpoczęcie wydobywania elementów z łańcucha od pierwszego elementu lub zakończenie pobierania elementów na ostatnim znaku w łańcuchu. Przypatrz się dwóm poniższym przykładom:

```
>>> s[:5]
'abcde'
>>>
```

oraz

```
>>> s = "abcdefghijkl"
>>> s[3:]
'defghi'
>>>
```

Możesz również korzystać z indeksów o wartościach ujemnych — pozwoli to na odliczanie miejsc od końca łańcucha. Taki zabieg przyda się w sytuacji, gdy chcesz wyodrębnić trzylite-rowe rozszerzenie pliku. Przyjrzyj się poniższemu przykładowi.

```
>>> "mójplik.txt"[-3:]
'txt'
```

Zobacz również

Proces łączenia ze sobą łańcuchów opisuje receptura 5.10.

Zastosowanie notacji `[:]` w kontekście list opisuje receptura 6.10.

5.16. Zastępowanie fragmentu łańcucha innym łańcuchem

Problem

Chcesz zastąpić pewien ciąg znaków występujący w łańcuchu.

Rozwiązanie

Skorzystaj z funkcji `replace`.

W poniższym przykładzie we wszystkich miejscach, w których występuje litera `X`, zostanie ona zastąpiona wyrazem `lata`.

```
>>> s = "To były najlepsze X. To były najgorsze X."
>>> s.replace("X", "lata")
'To były najlepsze lata. To były najgorsze lata.'
>>>
```

Omówienie

Poszukiwany ciąg znaków musi być dokładnie określony. Funkcja `replace` rozróżnia wielkie i małe litery oraz nie pomija spacji.

Zobacz również

Poszukiwanie ciągów znaków wewnętrz łańcucha, ale bez ich zastępowania, opisano w recepturze 5.14.

5.17. Zamiana znaków łańcucha na wielkie lub małe litery

Problem

Chcesz sprawić, że wszystkie znaki znajdujące się w łańcuchu będą wielkimi lub małymi literami.

Rozwiązanie

Funkcja `upper` zamienia wszystkie znaki łańcucha na wielkie litery, a funkcja `lower` — na małe litery.

W poniższym przykładzie łańcuch `aBcDe` zostanie zapisany w całości wielkimi literami.

```
>>> "aBcDe".upper()  
'ABCDE'  
>>>
```

A tym razem łańcuch ten zostanie zapisany w całości małymi literami.

```
>>> "aBcDe".lower()  
'abcde'  
>>>
```

Omówienie

Tak jak większość funkcji operujących na łańcuchach, funkcje `upper` i `lower` nie modyfikują łańcucha, a jedynie zwracają jego zmodyfikowaną kopię.

Przyjrzyj się poniższemu przykładowi. Funkcja zwraca kopię łańcucha `s`, ale oryginalny łańcuch nie zostaje zmodyfikowany.

```
>>> s = "aBcDe"  
>>> s.upper()  
'ABCDE'  
>>> s  
'aBcDe'  
>>>
```

Jeżeli chcesz, aby wszystkie znaki łańcucha `s` były wielkimi literami, to musisz to zrobić tak jak pokazano poniżej.

```
>>> s = "aBcDe"  
>>> s = s.upper()  
>>> s  
'ABCDE'  
>>>
```

Zobacz również

Modyfikację łańcuchów omówiono w recepturze 5.16.

5.18. Uruchamianie poleceń po spełnieniu określonych warunków

Problem

Chcesz, żeby pewne instrukcje były wykonywane tylko po spełnieniu określonych warunków.

Rozwiązanie

Zastosuj polecenie `if` (jeżeli).

Poniższy przykład wyświetli komunikat o treści `x` jest duże tylko wtedy, gdy wartość zmiennej `x` będzie większa od 100.

```
>>> x = 101  
>>> if x > 100:  
...     print("x jest duże")  
...  
x jest duże
```

Omówienie

Po słowie kluczowym `if` należy określić **warunek**. Warunek często polega na porównaniu ze sobą dwóch wartości i zwróceniu informacji, czy spełnienie warunku jest *prawdą*, czy *falszem*. Jeżeli funkcja warunkowa zwróci *prawdę*, to kolejne instrukcje znajdujące się w odpowiednio wciętych wierszach zostaną wykonane.

Dość często będziesz chciał wykonać pewne instrukcje, gdy warunek będzie spełniony, a w przeciwnym wypadku wykonać inne polecenia. W takim przypadku wraz z poleceniem `if` będziesz stosować polecenie `else`, co ilustruje poniższy przykład:

```
x = 101  
if x > 100:  
    print("x jest duże")  
else:  
    print("x jest małe")  
  
print("Ten komunikat będzie zawsze wyświetlany")
```

Możesz połączyć ze sobą w szereg całą listę warunków elif. Jeżeli któryś z warunków będzie spełniony, to wykonany zostanie znajdujący się pod nim blok kodu, a kolejne warunki nie będą sprawdzane. Przyjrzyj się poniższemu przykładowi.

```
x = 90
if x > 100:
    print("x jest duże")
elif x < 10:
    print("x jest małe")
else:
    print("x jest średnie")
```

W wyniku działania tego przykładowego programu na ekranie wyświetli się napis x jest średnie.

Zobacz również

W recepturze 5.19 znajduje się więcej informacji na temat różnych poleceń porównujących zmienne.

5.19. Porównywanie wartości

Problem

Chcesz porównywać wartości.

Rozwiązanie

Korzystaj z operatorów porównujących: <, >, <=, >=, == i !=.

Omówienie

W recepturze 6.15 przedstawiono użycie operatorów < (większy od) i > (mniejszy od). Poniżej znajduje się lista operatorów.

<	mniejszy od
>	większy od
<=	mniejszy lub równy
>=	większy lub równy
==	dokładnie równy
!=	różny od

Niektórzy zamiast operatora `!=` wolą stosować operator `<>`. Oba działają w ten sam sposób.

Możesz sprawdzić działanie tych operatorów przy użyciu konsoli Pythona (zobacz receptura 5.3). Ilustruje to poniższy przykład, w którym operatory zwracają prawdę logiczną (`True`) lub fałsz logiczny (`False`).

```
>>> 1 != 2
True
>>> 1 != 1
False
>>> 10 >= 10
True
>>> 10 >= 11
False
>>> 10 == 10
True
>>>
```

Często popełnianym błędem jest używanie operatora przypisującego wartość (`=`) zamiast operatora porównującego (`==`). Taki błąd trudno zauważyc, ponieważ jednym z porównywanych elementów jest zwykle zmienna. Taki zapis nie jest błędem składniowym, a program ulegnie komplikacji, jednak wynik jego działania nie będzie taki, jakiego byśmy się spodziewali.

Przedstawione operatory możesz również stosować do porównywaniałańcuchów, jak w poniższym przykładzie:

```
>>> 'aa' < 'ab'
True
>>> 'aaa' < 'aa'
False
```

Łańcuchy są porównywane leksykograficznie — według kolejności alfabetycznej.

Nie jest to w pełni poprawne, ponieważ wielkim literom przypisywane są mniejsze wartości niż małym literom.

Zobacz również

Zajrzyj także do receptury 6.15.

5.20. Operatory logiczne

Problem

Chcesz określić złożony warunek w kontekście polecenia `if`.

Rozwiązanie

Zastosuj operatory logiczne `and` (i), `or` (lub) i `not` (nie).

Omówienie

W poniższym przykładzie sprawdzamy, czy zmiennej `x` przypisano wartość znajdująca się w przedziale od 10 do 20. Zastosujemy w tym celu operator `and`.

```
>>> x = 17
>>> if x >= 10 and x <= 20:
...     print('x znajduje się w przedziale')
...
x znajduje się w przedziale
```

Możesz połączyć ze sobą wiele instrukcji warunkowych and i or. Złożone warunki grupuj za pomocą nawiasów.

Zobacz również

Zobacz także receptury 5.18 i 6.15.

5.21. Powtarzanie instrukcji określona liczbę razy

Problem

Chcesz, żeby pewna część programu była wykonywana określona liczbę razy.

Rozwiążanie

W celu dokonania iteracji zastosuj pętlę for.

W poniższym przykładzie polecenie jest wykonywane dziesięć razy.

```
>>> for i in range(1, 11):
...     print(i)
...
1
2
3
4
5
6
7
8
9
10
>>>
```

Omówienie

Drugi z parametrów polecenia range (zakres) jest wartością, która nie jest osiągana, a więc żeby policzyć do 10, jego wartość musi wynosić 11.

Zobacz również

Zajrzyj do receptury 5.22, jeżeli warunkiem przerwania pętli jest coś bardziej skomplikowanego od powtórzenia jej określona liczbę razy.

Jeżeli próbujesz powtarzać polecenia dla kolejnych elementów listy lub słownika, zajrzyj odpowiednio do receptury 6.7 lub 6.15.

5.22. Powtarzanie instrukcji do momentu, w którym zostanie spełniony określony warunek

Problem

Chcesz powtarzać jakąś część programu aż do momentu spełnienia pewnego warunku.

Rozwiązanie

Zastosuj pętlę `while`. Powtarza ona zagnieźdzoną w niej polecenia do momentu, w którym warunek pętli przestanie być prawdziwy. Poniższy przykład przedstawia pętlę, która zostanie przerwana w momencie wprowadzenia litery `X` przez użytkownika.

```
>>> answer = ''  
>>> while answer != 'X':  
...     answer = input('Wpisz polecenie:')  
...  
Wpisz polecenie:A  
Wpisz polecenie:B  
Wpisz polecenie:X  
>>>
```

Omówienie

W zaprezentowanym przykładzie zastosowano polecenie `input`, które działa w Pythonie 3. Aby uruchomić ten przykład w Pythonie 2, zastąp polecenie `input` poleceniem `raw_input`.

Zobacz również

Jeżeli chcesz wykonać jakąś instrukcję określoną liczbę razy, zajrzyj do receptury 5.20.

Jeżeli próbujesz powtarzać polecenia dla kolejnych elementów listy lub słownika, to zajrzyj odpowiednio do receptury 6.7 lub 6.15.

5.23. Przerywanie działania pętli

Problem

Chcesz, żeby zaistnienie pewnych warunków przerywało działanie pętli.

Rozwiązanie

W celu przerwania pętli `while` lub `for` zastosuj funkcję `break`.

Poniższy przykład działa dokładnie w ten sam sposób co kod przedstawiony w recepturze 5.22.

```
>>> while True:  
...     answer = input('Wpisz polecenie:')  
...     if answer == 'X':  
...         break
```

```
...
Wpisz polecenie:A
Wpisz polecenie:B
Wpisz polecenie:X
>>>
```

Omówienie

W zaprezentowanym przykładzie zastosowano polecenie `input`, które działa w Pythonie 3. Aby uruchomić ten przykład w Pythonie 2, zastąp polecenie `input` polecienniem `raw_input`.

Powyższy przykład działa dokładnie w ten sam sposób co przykład przedstawiony w recepturze 5.21. Różni się on tym, że warunek zawarty w pętli `while` jest cały czas prawdziwy (`True`), a więc pętla ta będzie wykonywana w nieskończoność. Jej działanie przerywa polecenie `break` wywołane po wprowadzeniu przez użytkownika znaku `X`.

Zobacz również

Pętla `while` może ulec zakończeniu również wtedy, gdy zadeklarowany w niej warunek przestanie być prawdziwy — zobacz receptura 5.21.

5.24. Definiowanie funkcji

Problem

Chcesz uniknąć kilkakrotnego powtarzania pewnego bloku kodu w programie.

Rozwiążanie

Utwórz funkcję składającą się z kilku linii kodu, który będzie mógł być wywoływany z różnych miejsc programu.

Poniższy przykład ilustruje deklarowanie funkcji, a także późniejsze jej wywołanie.

```
def count_to_10():
    for i in range(1, 11):
        print(i)

count_to_10()
```

W zaprezentowanym przykładzie funkcja została zdefiniowana za pomocą polecenia `def`. Funkcja wyświetlająca liczby od 1 do 10 zostanie wywołana, gdy w kodzie programu zostanie umieszczony zapis:

```
count_to_10()
```

Omówienie

Konwencje nazywania funkcji są takie same jak konwencje nazywania zmiennych (zobacz receptura 5.5). Ich nazwy powinny się zaczynać małą literą. Gdy nazwy składają się z więcej niż jednego wyrazu, to wyrazy te powinny być oddzielone od siebie znakiem podkreślenia.

Funkcja przedstawiona w przykładzie jest mało elastyczna — może liczyć tylko do 10. Gdybyśmy chcieli, żeby była bardziej elastyczna — liczyła do dowolnej wskazanej liczby — musielibyśmy dodać do niej **parametr** określający największą liczbę, do której ma ona liczyć. Poniżej przedstawiono zmodyfikowaną funkcję.

```
def count_to_n(n):
    for i in range(1, n + 1):
        print(i)

count_to_n(5)
```

Parametr o nazwie `n` jest umieszczony w nawiasach, a następnie zostaje użyty w poleceniu `range`, ale nie przed dodaniem do niego 1.

Użycie parametru określającego liczbę, do której chcemy policzyć, oznacza, że będziemy musieli zawsze określać jego wartość, nawet gdy zwykle liczymy do 10. Istnieje jednak sposób na określenie wartości domyślnej parametru. Pozwala to na uelastycznenie funkcji i zwalnia przy tym użytkownika z obowiązku każdorazowego określania wartości parametru:

```
def count_to_n(n=10):
    for i in range(1, n + 1):
        print(i)
count_to_n()
```

Funkcja będzie liczyć do 10, chyba że podczas wywoływania jej zostanie podana inna liczba.

Tworząc funkcję wymagającą więcej niż jednego parametru, np. funkcję, która będzie liczyła pomiędzy podanymi liczbami, parametry należy oddzielić przecinkami:

```
def count(from_num=1, to_num=10):
    for i in range(from_num, to_num + 1):
        print(i)

count()
count(5)
count(5, 10)
```

Wszystkie przytoczone przykłady przedstawiają funkcje, które nie zwracają żadnych wartości. One po prostu wykonują pewne zadania. Jeżeli potrzebujesz funkcji, która zwraca jakąś wartość, musisz zastosować polecenie `return`.

Poniższa funkcja jako argument pobiera łańcuch znaków, a następnie dodaje na jego końcu słowo `proszę`.

```
def make_polite(sentence):
    return sentence + "proszę"

print(make_polite("Podaj sól"))
```

Wartość zwracaną przez funkcję możesz przypisać do zmiennej lub, tak jak w tym przypadku, po prostu wyświetlić na ekranie.

Zobacz również

Jeżeli chcesz, żeby funkcja zwracała więcej niż jedną wartość, zajrzyj do receptury 7.3.

Python — listy i słowniki

6.0. Wprowadzenie

W rozdziale 5. przyjrzyliśmy się podstawom Pythona. Teraz zajmiemy się głównymi strukturami danych, które występują w tym języku — listom i słownikom.

6.1. Tworzenie list

Problem

Chcesz utworzyć zmienną, która będzie zawierała całą serię wartości (więcej niż jedną wartość).

Rozwiązanie

Korzystaj z list. W Pythonie lista jest zbiorem wartości ułożonych w kolejności. Znając kolejność ułożenia elementów listy, możemy uzyskiwać do nich dostęp.

Elementy, które chcesz umieścić na liście, należy objąć nawiasami kwadratowymi [i]:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>>
```

W przeciwieństwie do mniej elastycznych tablic stosowanych w języku C, pracując z listami, nie musisz określać ich rozmiaru w momencie deklaracji. Liczbę elementów znajdujących się na liście możesz zmienić w dowolnej chwili.

Omówienie

Jak wynika z powyższego przykładu, elementy znajdujące się na liście nie muszą być tego samego typu, choć w praktyce często się tak zdarza.

Aby utworzyć pustą listę, do której będziesz później dodawać element, zastosuj zapis:

```
>>> a= []
>>>
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.2. Uzyskiwanie dostępu do elementu znajdującego się na liście

Problem

Chcesz uzyskać dostęp do jednego z elementów listy lub go zmodyfikować.

Rozwiążanie

Korzystaj z notacji zawierającej nawiasy. W celu uzyskania dostępu do elementu należy podać jego pozycję na liście. Na przykład:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a[1]
'Marcin'
```

Omówienie

Numeracja elementów listy zaczyna się od 0.

Notację wykorzystywaną do uzyskiwania dostępu do elementu listy można stosować również do modyfikacji tego elementu:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a[1] = 777
[34, 777, 12, False, 72.3]
```

Jeżeli będziesz próbował uzyskać dostęp do elementu o zbyt dużej wartości indeksu (elementu nieznajdującego się na liście), to zostanie wyświetlony komunikat błędu o treści index out of range.

```
>>> a[50]=777
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
>>>
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.3. Ustalanie długości listy

Problem

Chcesz się dowiedzieć, z ilu elementów składa się dana lista.

Rozwiążanie

Użyj polecenia len. Poniższy przykład ilustruje sposób korzystania z niego:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> len(a)
5
```

Omówienie

Polecenia `len` można używać również w celu określenia długości łańcucha (zobacz receptura 5.13).

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.4. Dodawanie elementów do listy

Problem

Chcesz dodać element do listy.

Rozwiązanie

Zastosuj jedną z funkcji `append`, `insert` lub `extend`.

W celu wstawienia jednego elementu na koniec listy skorzystaj z polecenia `append`:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a.append("nowy")
>>> a
[34, 'Marcin', 12, False, 72.3, 'nowy']
```

Omówienie

Czasami może zaistnieć potrzeba wstawienia nowego elementu na określoną pozycję listy, a nie na jej koniec. W takim przypadku należy skorzystać z polecenia `insert`. Pierwszym argumentem tego polecenia jest numer indeksu, pod którym ma być wstawiony nowy element, a drugim argumentem jest wstawiany element.

```
>>> a.insert(2, "nowy2")
>>> a
[34, 'Marcin', 'nowy2', 12, False, 72.3]
```

Elementy znajdujące się za łańcuchem `nowy2` zostały przesunięte o jedną pozycję.

Funkcji `append` i `insert` można użyć w celu dodania do listy tylko jednego elementu. Funkcja `extend` dodaje wszystkie elementy jednej listy do końca drugiej listy:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> b = [74, 75]
>>> a.extend(b)
>>> a
[34, 'Marcin', 12, False, 72.3, 74, 75]
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.5. Usuwanie elementów z listy

Problem

Chcesz usunąć pewne elementy z listy.

Rozwiązanie

Skorzystaj z funkcji pop.

Polecenie pop użyte bez jakiegokolwiek parametru usuwa ostatni element listy.

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a.pop()
72.3
>>> a
[34, 'Marcin', 12, False]
```

Omówienie

Zwróć uwagę na to, że funkcja pop zwraca element usunięty z listy.

Żeby usunąć z listy określony element, musisz zastosować funkcję pop wraz z parametrem wskazującym pozycję elementu, który chcesz usunąć.

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a.pop(0)
34
```

Próba użycia indeksu, który nie będzie się znajdował na liście, spowoduje wyświetlenie komunikatu o błędzie.

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.6. Tworzenie listy w wyniku przetwarzania łańcucha

Problem

Musisz dokonać konwersji łańcucha zawierającego słowa oddzielone jakimiś znakami na tablicę łańcuchów (każde słowo tworzy odrębny łańcuch).

Rozwiązanie

Skorzystaj z funkcji split.

Polecenie split zastosowane bez parametrów zamienia słowa znajdujące się w łańcuchu na kolejne elementy tablicy:

```
>>> "abc def ghi".split()
['abc', 'def', 'ghi']
```

Parametr funkcji `split` służy do określania elementów rozdzielających wyrazy znajdujące się w łańcuchu, np.:

```
>>> "abc--de--ghi".split('--')
['abc', 'de', 'ghi']
```

Omówienie

Polecenie to może być bardzo przydatne np. podczas importowania danych z pliku. W roli argumentu funkcji `split` można zastosować ciąg znaków, które rozdzielają elementy przetwarzanego łańcucha. Gdyby elementy łańcucha były oddzielone od siebie przecinkami, mógłbyś je rozdzielić w następujący sposób:

```
>>> "abc,def,ghi".split(',')
['abc', 'def', 'ghi']
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.7. Iteracja listy

Problem

Chcesz wykonać jakieś operacje na każdym elemencie listy.

Rozwiązanie

Skorzystaj z polecenia `for`.

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> for x in a:
...     print(x)
...
34
Marcin
12
False
72.3
>>>
```

Omówienie

Bezpośrednio po poleceniu `for` należy podać nazwę zmiennej (w tym przypadku jest to `x`). Polecenie to nosi nazwę pętli i zostanie wykonane dla każdego elementu określonego po słowie kluczowym `in`.

Kod znajdujący się w kolejnej odpowiednio wcietowej linii zostanie wykonany dla każdego elementu listy. Podczas każdego przebiegu pętli zmiennej `x` będzie przypisywany kolejny element listy. Następnie zawartość tej zmiennej zostanie wyświetlona, jak pokazano to w podanym wcześniej przykładzie.

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.8. Numerowanie elementów listy

Problem

Musisz wykonać jakieś operacje na elementach listy i w związku z tym chcesz poznać numer indeksu każdego z elementów.

Rozwiązanie

Skorzystaj z poleceń `for` oraz `enumerate`.

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> a = [34, 'Fred', 12, False, 72.3]
>>> for (i, x) in enumerate(a):
...     print(i, x)
...
(0, 34)
(1, 'Marcin')
(2, 12)
(3, False)
(4, 72.3)
>>>
```

Omówienie

Wiedza o tym, pod jakimi indeksami leżą elementy listy, w praktyce dość często się przydaje. Alternatywny sposób wyświetlenia elementów oraz ich indeksów polega na prostym wyświetlanie kolejnych wartości indeksu za pomocą składni zawierającej znaki `[i]`:

```
>>> a = [34, 'Marcin', 12, False, 72.3]
>>> for i in range(len(a)):
...     print(i, a[i])
...
(0, 34)
(1, 'Marcin')
(2, 12)
(3, False)
(4, 72.3)
>>>
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

Receptura 6.7 opisuje iterację elementów listy bez podawania numerów indeksu.

6.9. Sortowanie listy

Problem

Chcesz posortować elementy listy.

Rozwiązanie

Skorzystaj z polecenia `sort`:

```
>>> a = ["to", "były", "najlepsze", "lata"]
>>> a.sort()
>>> a
['były', 'lata', 'najlepsze', 'to']
```

Omówienie

Sortowanie listy powoduje jej modyfikację. Polecenie `sort` nie zwraca posortowanej kopii listy. Jeżeli potrzebujesz również oryginalnej, nieposortowanej listy, to musisz ją skopiować za pomocą funkcji `copy`, która znajduje się w standardowej bibliotece Pythona. W takim przypadku wykonaj kopię listy przed jej posortowaniem.

```
>>> import copy
>>> a = ["to", "były", "najlepsze", "lata"]
>>> b = copy.copy(a)
>>> b.sort()
>>> a
["to", "były", "najlepsze", "lata"]
>>> b
['były', 'lata', 'najlepsze', 'to']
>>>
```

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.10. Wycinanie fragmentu listy

Problem

Chcesz stworzyć podlistę zawierającą pewien zbiór elementów znajdujących się już na jakiejś liście.

Rozwiązanie

Skorzystaj ze składni `[:]`. Poniższy przykładowy kod zwraca nową listę składającą się z elementów (o indeksach od 1 do 2) przetwarzanej listy. Drugi z indeksów wpisywanych w składni `[:]` określa element, który nie zostanie umieszczony na nowej liście.

```
>>> l = ["a", "b", "c", "d"]
>>> l[1:3]
['b', 'c']
```

Indeksy elementów listy są numerowane od 0, a więc indeks o numerze 1 będzie wskazywał drugi element listy, a indeks o numerze 5 będzie wskazywał szósty element listy. Drugi z indeksów wpisywanych w składni [:] określa element, który nie zostanie umieszczony na nowej liście, a zatem nie znajdzie się tam litera d.

Omówienie

Składnię [:] można stosować na wiele sposobów. Niepodanie pierwszego indeksu spowoduje umieszczenie na nowej liście zbioru elementów zaczynających się od indeksu o numerze 0, a niepodanie drugiego indeksu spowoduje umieszczenie na nowej liście zbioru elementów, który będzie się kończył na ostatnim elemencie starej listy. Przypatrz się ilustrującemu to przykładowi:

```
>>> l = ["a", "b", "c", "d"]
>>> l[:3]
['a', 'b', 'c']
>>> l[3:]
['d']
>>>
```

W celu odliczenia od końca listy możesz stosować indeksy o ujemnych numerach. Funkcja zaprezentowana poniżej zwróci dwa elementy listy:

```
>>> l[-2:]
['c', 'd']
```

Gdybyśmy w poprzednim przykładzie zastosowali kod l[:-2], zostałaby nam zwrócona lista ['a', 'b'].

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

Receptura 5.15 opisuje zastosowanie tej samej składni w kontekście łańcuchów.

6.11. Przetwarzanie elementów listy przez funkcję

Problem

Chcesz, żeby funkcja przetworzyła każdy element listy, a następnie zwróciła wyniki przeprowadzonych operacji.

Rozwiążanie

Skorzystaj z tak zwanego **złożenia**.

W poniższym przykładzie nastąpi przekształcenie każdego z łańcuchów w liście, tak aby był on zapisany wielkimi literami. Zostanie zwrócona lista o takiej samej długości jak lista źródłowa, ale będzie się ona składać z elementów zapisanych tylko wielkimi literami.

```
>>> l = ["abc", "def", "ghi", "ijk"]
>>> [x.upper() for x in l]
['ABC', 'DEF', 'GHI', 'IJK']
```

Może się to wydawać zagmatwane, ale nie ma powodu, dla którego nie można byłoby zagnieżdżać poleceń w ten właśnie sposób.

Omówienie

Jest to bardzo zwięzły sposób wykonywania złożień. Całe wyrażenie musi być umieszczone w nawiasach kwadratowych ([]). Pierwszym elementem złożenia jest kod, który jest wykonywany na każdym z elementów listy. Reszta złożenia przypomina zwyczajne polecenie iteracji listy (zobacz receptura 6.7). Zmienna przetwarzana przez pętlę została umieszczona po komendzie `for`, a po komendzie `in` określono użytą listę.

Zobacz również

Receptury od 6.1 do 6.11 dotyczą zagadnień związanych z listami.

6.12. Tworzenie słownika

Problem

Chcesz stworzyć tabelę zawierającą powiązane ze sobą klucze i wartości.

Rozwiązanie

Stwórz słownik.

Tablice sprawdzają się świetnie, gdy potrzebujesz dostępu do uszeregowanej listy elementów lub gdy zawsze wiesz, którego elementu chcesz użyć. Słowniki to typ danych będący alternatywą dla list. Słowniki są zorganizowane w zupełnie inny sposób niż listy.

Na rysunku 6.1 przedstawiono schemat organizacji słownika.

phone_numbers	
Klucz: Szymon	Wartość: 01234 567899
Klucz: Joanna	Wartość: 01234 666666
Klucz: Piotr	Wartość: 01234 777555
Klucz: Ewa	Wartość: 01234 887788

Rysunek 6.1. Słownik w Pythonie

W słowniku zapisane są pary klucz-wartość. Klucz może być użyty do odczytania wartości. Jest to bardzo wygodne rozwiązanie, pozwalające na uniknięcie konieczności przeszukiwania całej listy.

W celu utworzenia słownika zastosuj notację {}:

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}
```

Omówienie

W przedstawionym przykładzie kluczami słownika były łańcuchy, ale w roli kluczy mogą występować dowolne typy danych. Najczęściej w roli kluczy stosuje się właśnie łańcuchy.

Wartości przechowywane w słowniku mogą być również danymi dowolnego typu — mogą być nawet słownikami lub listami. W poniższym przykładzie stworzono słownik (a), a następnie umieszczono go jako wartość w drugim słowniku (b):

```
>>> a = {'klucz1': 'wartość1', 'klucz2': 2}
>>> a
{'klucz2': 2, 'klucz1': 'wartość1'}
>>> b = {'b_klucz1': a}
>>> b
{'b_klucz1': {'klucz2': 2, 'klucz1': 'wartość1'}}
```

Po wyświetleniu zawartości słownika zauważysz, że znajdujące się w nim elementy mogą być ułożone w innej kolejności niż podczas tworzenia i inicjalizowania słownika.

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}
>>> phone_numbers
{'Joanna': '01234 666666', 'Szymon': '01234 567899'}
```

Elementy znajdujące się w słowniku nie są ułożone w określonej kolejności, tak jak miało to miejsce w przypadku list. Kolejność elementów umieszczonych w słowniku jest losowa. Wynika to z wewnętrznej reprezentacji danych przechowywanych w słowniku.

Kolejność ta wydaje się losowa, ponieważ słownik jest oparty na **tablicy mieszącej**. Wartości przechowywane w tej tablicy są tam umieszczane według **funkcji mieszącej**. Funkcja ta oblicza numeryczny ekwiwalent obiektu dowolnego typu.

Więcej informacji na temat tablic mieszących znajdziesz w Wikipedii — http://pl.wikipedia.org/wiki/Tablica_miesza%C4%85ca.

Zobacz również

Receptury od 6.12 do 6.15 dotyczą zagadnień związanych z zastosowaniem słowników.

6.13. Uzyskiwanie dostępu do elementów znajdujących się w słowniku

Problem

Chcesz uzyskać dostęp do elementów słownika i modyfikować je.

Rozwiązanie

Skorzystaj w tym celu z notacji []. W nawiasach kwadratowych należy umieścić klucz elementu, do którego chcesz uzyskać dostęp.

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}
>>> phone_numbers['Szymon']
'01234 567899'
>>> phone_numbers['Joanna']
'01234 666666'
```

Omówienie

Jeżeli będziesz próbował uzyskać dostęp do elementu, którego klucz nie znajduje się w słowniku, zostanie wyświetlony komunikat `key error`.

```
{'b_klucz1': {'klucz2': 2, 'klucz1': 'wartość1'}}  
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}  
>>> phone_numbers['Jan']  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
KeyError: 'Jan'  
>>>
```

Notacji [] możesz również używać do dodawania nowych wartości do słownika lub do nadpisywania wartości już obecnych w słowniku.

Poniższy kod dodaje nowy element do słownika o kluczu Jan i wartości 01234 777555.

```
>>> phone_numbers['Jan'] = '01234 777555'  
>>> phone_numbers['Jan']  
'01234 777555'
```

Jeżeli dany klucz nie został użyty w słowniku, to automatycznie tworzony jest w nim nowy element. Gdy dany klucz już istnieje w słowniku, to obecna wartość jest nadpisywana.

Reguły te kontrastują z zasadami odczytywania danych ze słownika — wprowadzenie nieoznaczonego klucza podczas próby odczytu danych ze słownika spowoduje wyświetlenie komunikatu o błędzie.

Zobacz również

Receptury od 6.12 do 6.15 dotyczą zagadnień związanych z zastosowaniem słowników.

Informacje na temat obsługi komunikatów o błędach znajdziesz w recepturze 7.10.

6.14. Usuwanie elementów ze słownika

Problem

Chcesz usunąć element ze słownika.

Rozwiązanie

Zastosuj polecenie pop, określając klucz elementu, który chcesz usunąć:

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}  
>>> phone_numbers.pop('Joanna')  
'01234 666666'  
>>> phone_numbers  
{'Szymon': '01234 567899'}
```

Omówienie

Polecenie pop zwraca wartość elementu usuwanego ze słownika.

Zobacz również

Receptury od 6.12 do 6.15 dotyczą zagadnień związanych z zastosowaniem słowników.

6.15. Iteracja słownika

Problem

Chcesz przeprowadzić pewną operację na kolejnych elementach znajdujących się w słowniku.

Rozwiązanie

W celu przeprowadzenia iteracji elementów znajdujących się w słowniku zastosuj polecenie `for`:

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}
>>> for name in phone_numbers:
...     print(name)
...
Joanna
Szymon
```

Omówienie

Istnieją również inne techniki iteracji słownika. Poniższa metoda jest przydatna, jeżeli potrzebujesz dostępu zarówno do wartości, jak i kluczy:

```
>>> phone_numbers = {'Szymon': '01234 567899', 'Joanna': '01234 666666'}
>>> for name, num in phone_numbers.items():
...     print(name + " " + num)
...
Joanna 01234 666666
Szymon 01234 567899
```

Zobacz również

Receptury od 6.12 do 6.15 dotyczą zagadnień związanych z zastosowaniem słowników.

Inne zastosowania polecenia `for` przedstawiono w recepturach 5.3, 5.21, 6.7 i 6.11.

Python — zaawansowane funkcje

7.0. Wprowadzenie

W tym rozdziale zajmiemy się bardziej zaawansowanymi możliwościami oferowanymi przez Pythona — skupimy się szczególnie na programowaniu obiektowym, odczycie i zapisie plików, obsłudze wyjątków, stosowaniu modułów i programowaniu sieciowym.

7.1. Tworzenie multimedialnego centrum rozrywki

Problem

Chcesz sformatować liczby tak, aby miały określoną liczbę cyfr po przecinku.

Rozwiązanie

Zastosuj łańcuch format. Możesz z niego skorzystać w sposób przedstawiony w poniższym przykładzie:

```
>>> x = 1.2345678
>>> "x={:.2f}".format(x)
'x=1.23'
```

Omówienie

Łańcuch formatujący może zawierać zarówno tekst, jak i znaczniki umieszczone w nawiasach klamrowych ({}). Funkcja format może przyjąć dowolną liczbę parametrów. Parametry te określają sposób, w jaki mają być sformatowane przetwarzane dane.

W podanym wyżej przykładzie specyfikatorem formatu było wyrażenie :.2f. Symbolizuje ono liczbę zmiennoprzecinkową (typu *float*), której rozwinięcie dziesiętne składa się z dwóch cyfr.

Jeżeli chciałbyś, żeby liczby były sformatowane tak, aby miały stałą długość siedmiu cyfr (lub spacji dopełniających), trzeba by dodać liczbę przed separatorem dziesiętnym:

```
>>> "x={:7.2f}".format(x)
'x=    1.23'
>>>
```

W związku z tym, że liczba składa się tylko z trzech cyfr, przed cyfrą 1 dodano spacje dopełniające.

Bardziej skomplikowanym zadaniem może być wyświetlenie temperatury w stopniach Celsjusza i w skali Fahrenheita:

```
>>> c = 20.5
>>> "Temperatura {:5.2f} stopni Celsjusza, {:5.2f} Fahrenheita.".format(c, c * 9 / 5 + 32)
Temperatura 20.50 stopni Celsjusza, 68.90 Fahrenheita.'
>>>
```

Zobacz również

Więcej informacji na temat formatowania w Pythonie znajdziesz na stronie internetowej <https://docs.python.org/2/library/string.html#format-specification-mini-language>.

7.2. Formatowanie dat

Problem

Chcesz zamienić datę na łańcuch i sformatować go w określony sposób.

Rozwiązanie

Zastosuj łańcuch `format` na obiekcie zawierającym datę.

Na przykład:

```
>>> from datetime import datetime
>>> d = datetime.now()
>>> "{:%Y-%m-%d %H:%M:%S)".format(d)
'2013-05-02 16:00:45'
>>>
```

Omówienie

W Pythonie istnieją specjalne symbole przeznaczone do formatowania dat: `%Y` (rok), `%m` (miesiąc) i `%d` (dzień).

Zobacz również

Zobacz również recepturę 7.1, w której opisano formatowanie liczb.

Formatowanie w Pythonie to niemal wewnętrzny złożony język. Więcej informacji na ten temat znajdziesz w witrynie <https://docs.python.org/2/library/string.html#format-specification-mini-language>.

7.3. Zwracanie więcej niż jednej wartości

Problem

Chcesz napisać funkcję zwracającą więcej niż jedną wartość.

Rozwiążanie

Skorzystaj z **krotki** i zastosuj składnię przypisującą wiele zmiennych.

Na przykład:

```
>>> def calculate_temperatures (kelvin):
...     celsius = kelvin - 273
...     fahrenheit = celsius * 9 / 5 + 32
...     return celsius, fahrenheit
...
>>> c, f = calculate_temperatures(340)
>>>
>>> print(c)
67
>>> print(f)
152.6
```

Krotka to struktura danych Pythona, która przypomina listę. Jest ona umieszczona w nawiasach zwykłych, a nie kwadratowych. Ponadto rozmiar krotki jest określony.

Omówienie

Taki zabieg przydaje się zwykle, gdy chce się zwrócić jedną lub dwie wartości. Jednak w przypadku zwracania bardziej złożonych danych wygodniej będzie Ci zastosować elementy programowania obiektowego znajdujące się w Pythonie i zdefiniować klasę zawierającą dane. W ten sposób zamiast krotki zwróciś klasę.

Zobacz również

Więcej informacji na temat definiowania klasy znajdziesz w recepturze 7.4.

7.4. Definiowanie klasy

Problem

Chcesz utworzyć klasę zawierającą powiązane ze sobą dane i funkcje.

Rozwiążanie

Zdefiniuj klasę i umieść w niej odpowiednie zmienne.

W poniższym przykładzie definiowana jest klasa reprezentująca pozycję w książce adresowej:

```
class Person:  
    """Ta klasa reprezentuje obiekt person"""  
    def __init__(self, name, tel):  
        self.name = name  
        self.tel = tel
```

Pierwsza linia kodu znajdującej się w definicji klasy zawiera łańcuch dokumentujący ujęty w pojedyncze, podwójne lub potrójne cudzysłowy. łańcuch ten powinien wyjaśniać cel utworzenia klasy. Nie jest to element obowiązkowy, jednak pomaga innym użytkownikom. Zabieg ten przydaje się zwłaszcza wtedy, gdy klasę mogą stosować również inni użytkownicy.

Łańcuchy dokumentujące nie są zwyczajnymi komentarzami. Co prawda nie są to aktywne wiersze kodu, ale są one połączone z klasą, a więc w dowolnym momencie możesz wyświetlić ich treść za pomocą polecenia:

```
Person.__doc__
```

Wewnątrz definicji klasy znajduje się metoda konstruktora, która będzie stosowana do tworzenia nowych egzemplarzy klasy. Klasa pełni rolę formularza. Definiując klasę o nazwie Person, nie tworzymy jeszcze tak naprawdę żadnego obiektu typu Person:

```
def __init__(self, name, tel):  
    self.name = name  
    self.tel = tel
```

Nazwa metody konstruktora musi zawierać na początku i na końcu symbol podkreślenia, tak jak w przytoczonym kodzie (wokół wyrażenia `init`).

Omówienie

Jedną z różnic pomiędzy Pythonem a innymi językami programowania obiektowego jest to, że wszystkie metody definiowane wewnątrz klasy muszą zawierać w roli parametru specjalną zmienną `self`. W tym przypadku jest to odniesienie do nowego egzemplarza. Zmienna `self` w Pythonie pełni podobną rolę do zmiennej `this` występującej między innymi w języku Java.

Kod znajdujący się w tej metodzie przenosi dołączone do niej parametry do zmiennych składowych. Zmienne składowe nie muszą być wcześniej deklarowane, ale muszą być poprzedzone przedrostkiem `self`.

Linia kodu:

```
self.name = name
```

tworzy więc zmienną o nazwie `name`, która jest dostępna dla każdego elementu składowego klasy `Person`. Gdy zmienna ta zostanie zainicjalizowana wartością, zostanie utworzony egzemplarz, który będzie wyglądać mniej więcej tak:

```
p = Person("Szymon", "1234567")
```

Możemy się upewnić, że nowemu obiektowi `Person` przypisano imię „Szymon”, wpisując poniższy kod:

```
>>> p.name  
Szymon
```

W przypadku tworzenia złożonego programu dobrą praktyką jest zapisywanie każdej klasy w oddzielnym pliku. Nazwa pliku powinna być zgodna z nazwą klasy. Taki zabieg ułatwi Ci przekonwertowanie klasy na moduł (zobacz receptura 7.11).

Zobacz również

Informacje na temat definiowania metod znajdziesz w recepturze 7.5.

7.5. Definiowanie metody

Problem

Chcesz dodać metodę do klasy.

Rozwiązanie

Poniższy przykład prezentuje dodawanie metody do definicji klasy.

```
class Person:  
    """Ta klasa reprezentuje obiekt Person"""  
  
    def __init__(self, first_name, surname, tel):  
        self.first_name = first_name  
        self.surname = surname  
        self.tel = tel  
  
    def full_name(self):  
        return self.first_name + " " + self.surname
```

Metoda `full_name` scala atrybuty `first_name` (z ang. imię) i `surname` (z ang. nazwisko) obiektu `Person` i dodaje pomiędzy nimi spację.

Omówienie

Klasy pełnią rolę funkcji przywiązań do określonej klasy, które mogą przetwarzać zmienne składowe klasy, ale nie muszą tego robić. A zatem metoda może zawierać dowolny kod, w tym kod wywołujący inną metodę.

Jeżeli w obrębie tej samej klasy jedna metoda wywołuje drugą, to wywołanie takie musi być poprzedzone prefiksem `self..`.

Zobacz również

Więcej informacji o definiowaniu klasy znajdziesz w recepturze 7.4.

7.6. Dziedziczenie

Problem

Potrzebujesz wyspecjalizowanej wersji stworzonej wcześniej klasy.

Rozwiązanie

Skorzystaj z możliwości **dziedziczenia** i stwórz podklasę utworzonej wcześniej klasy w celu dodania nowych zmiennych składowych i metod.

Domyślnie wszystkie nowe klasy, które tworzysz, są podkласami obiektu `object`. Możesz to zmienić, podając w nawiasach klasę, którą chcesz zastosować w roli klasy nadzędnej, po nazwie klasy umieszczonej w jej definicji. W poniższym przykładzie zdefiniowano klasę `Employee` (z ang. pracownik) jako podklasę `Person` (z ang. osoba) i dodano do niej nową zmienną składową `salary` (z ang. wynagrodzenie) i dodatkową metodę `give_raise` (z ang. daj podwyżkę):

```
class Employee(Person):

    def __init__(self, first_name, surname, tel, salary):
        super().__init__(first_name, surname, tel)
        self.salary = salary

    def give_raise(self, amount):
        self.salary = self.salary + amount
```

Zwróć uwagę na to, że przykład przedstawiony powyżej jest przeznaczony dla Pythona 3. W Pythonie 2 nie możesz używać funkcji `super` w ten sposób. Musiałbyś zastosować poniższy zapis.

```
class Employee(Person):

    def __init__(self, first_name, surname, tel, salary):
        Person.__init__(self, first_name, surname, tel)
        self.salary = salary

    def give_raise(self, amount):
        self.salary = self.salary + amount
```

Omówienie

W obu przedstawionych przykładach inicjator metody podklasy używa najpierw inicjatora metody klasy macierzystej (klasy nadzędnej), a następnie dodaje zmienne składowe. Dzięki temu nie musisz powtarzać kodu inicjującego w nowej podklasie.

Zobacz również

Informacje dotyczące definiowania klasy znajdziesz w recepturze 5.24.

Mechanizm dziedziczenia w Pythonie ma wiele zastosowań. Dopuszczalne jest **wielodziedziczenie** — zabieg, w którym podklasa dziedziczy po więcej niż jednej klasie nadzędnej. Więcej informacji na temat dziedziczenia znajdziesz w oficjalnej dokumentacji Pythona: <https://docs.python.org/release/1.5/tut/node66.html>.

7.7. Zapis danych w pliku

Problem

Chcesz coś zapisać w pliku.

Rozwiążanie

Skorzystaj z funkcji open (otwierającej plik), write (zapisującej dane) i close (zamykającej plik):

```
>>> f = open('test.txt', 'w')
>>> f.write('Tekst zapisywany w pliku')
>>> f.close()
```

Omówienie

Gdy otworzysz plik, przed jego zamknięciem możesz wykonać dowolnie wiele operacji zapisu. Zamykanie pliku za pomocą polecenia close jest bardzo ważne. Każda operacja zapisu powinna się kończyć natychmiastowym uaktualnieniem zawartości pliku, jednak zapisywane dane mogą być buforowane w pamięci, a niezamknięcie pliku może doprowadzić do ich utraty. Uniemożliwi ponadto otwarcie pliku przez inne programy.

Polecenie open przyjmuje dwa parametry. Pierwszy z nich określa ścieżkę do pliku, w którym chcesz dokonywać operacji zapisu. Ścieżka może być względna (od bieżącego katalogu, w którym pracujesz) lub bezwzględna (rozpoczynająca się od symbolu /).

Drugi parametr określa tryb otwierania pliku. Aby nadpisać istniejący plik lub stworzyć nowy plik o określonej nazwie, zastosuj parametr w. W tabeli 7.1 przedstawiono wszystkie tryby otwierania plików. Tryby można łączyć ze sobą za pomocą znaku +. Na przykład w celu otwarcia pliku w trybie do odczytu i w trybie binarnym zastosowałbyś następujące polecenie:

```
>>> f = open('test.txt', 'r+b')
```

Tabela 7.1. Tryby otwierania plików

Tryb	Opis
r	Odczyt
w	Zapis
a	Plik nie zostanie nadpisany; nowe dane zostaną dodane na jego końcu
b	Tryb binarny
f	Tryb tekstowy (domyślny)
+	Skrótowa forma zapisu r+w

Tryb binarny pozwala na zapis binarnych strumieni danych, takich jak np. obrazy.

Zobacz również

Odczytywanie zawartości pliku opisano w recepturze 7.8. Więcej informacji na temat obsługi wyjątków znajdziesz w recepturze 7.10.

7.8. Odczytywanie pliku

Problem

Chcesz odczytać dane z pliku i umieścić je w łańcuchu.

Rozwiążanie

W celu odczytania zawartości pliku skorzystaj z funkcji `open` (otwierającej plik), `read` (odczytującej dane) i `close` (zamykającej plik). Poniższy przykładowy kod wczytuje całą zawartość pliku do zmiennej `s`.

```
f = open('test.txt')
s = f.read()
f.close()
```

Omówienie

Za pomocą polecenia `readline` możesz wczytywać pojedyncze wiersze pliku tekstowego.

Próba odczytu pliku (za pomocą podanego wcześniej kodu), który nie istnieje lub jest z jakiegoś powodu niedostępny, spowoduje wyświetlenie komunikatu błędu. Zamiast komunikatu, który będzie prawdopodobnie niezrozumiały dla użytkownika, możesz wyświetlić komunikat o określonej przez siebie treści. W tym celu skorzystaj z poleceń `try` i `except`.

```
try:
    f = open('test.txt')
    s = f.read()
    f.close()
except IOError:
    print("Nie można otworzyć pliku")
```

Zobacz również

W recepturze 7.7 znajdują się informacje na temat zapisu danych w pliku, a także tabela opisująca tryby otwierania plików.

Więcej informacji na temat obsługi wyjątków znajdziesz w recepturze 7.10.

7.9. Serializacja

Problem

Chcesz zapisać całą zawartość jakiejś struktury danych do pliku, tak aby możliwe było jego wczytanie podczas kolejnego uruchomienia programu.

Rozwiążanie

Serializacja (ang. *pickling*) pozwoli Ci zrzucić całą strukturę danych do pliku w formacie, który umożliwia późniejsze załadowanie do pamięci zapisanej struktury.

Poniższy kod zapisuje w pliku złożoną strukturę listy.

```
>>> import pickle  
>>> mylist = ['przykładowy tekst', 123, [4, 5, True]]  
>>> f = open('mylist.pickle', 'w')  
>>> pickle.dump(mylist, f)  
>>> f.close()
```

W celu dodania do nowej listy zawartości umieszczonej wcześniej w pliku należy zastosować następujący kod:

```
>>> f = open('mylist.pickle')  
>>> other_array = pickle.load(f)  
>>> f.close()  
>>> other_array  
['przykładowy tekst', 123, [4, 5, True]]
```

Omówienie

Serializację można stosować z niemal każdą strukturą danych. Nie musi być to koniecznie lista.

Zapisywane dane mają format tekstu, który może być odczytany przez użytkownika, jednak zwykle nie będziesz musiał przyglądać się temu plikowi ani edytować ręcznie jego zawartości.

Zobacz również

W recepturze 7.7 znajdują się informacje na temat zapisu danych w pliku, a także tabela opisująca tryby otwierania plików.

7.10. Obsługa wyjątków

Problem

Chcesz, żeby zamiast standardowego komunikatu o błędzie, w momencie gdy jakaś operacja wykonywana przez program się nie powiedzie, była wyświetlana informacja zrozumiała dla użytkownika.

Rozwiązanie

Skorzystaj z konstrukcji `try – except`.

Poniższy przykład, przedstawiony wcześniej w recepturze 7.8, powoduje zastąpienie wszystkich komunikatów o błędach związanych z otwarciem pliku jedną zrozumiałą dla użytkownika informacją.

```
try:  
    f = open('test.txt')  
    s = f.read()  
    f.close()  
except IOError:  
    print("Nie można otworzyć pliku")
```

Omówienie

Do powstawania wyjątków podczas działania programu może dochodzić dość często, poza próbami otwarcia pliku np. podczas uzyskiwania dostępu do indeksu, który nie znajduje się na liście. W poniższym przykładzie przedstawiono próbę uzyskania dostępu do czwartego elementu trójelementowej listy:

```
>>> list = [1, 2, 3]
>>> list[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Błędy i wyjątki mają swoją hierarchię. Przechwytyując wyjątki, możesz wyświetlać szczegółowy opis problemu lub podać tylko ogólną informację o tym, co się stało.

Polecenie `exception` znajduje się bardzo wysoko w hierarchii i przechwytuje prawie wszystkie wyjątki. Możesz dodać pod nim dodatkową sekcję `except`, która będzie służyła do przechwytywania innych wyjątków — takich, które mają zostać obsłużone w inny sposób. Jeżeli nie określisz jakieś klasy wyjątków, przechwytywane będą wszystkie wyjątki.

Obsługując komunikaty o błędach w Pythonie, możesz również korzystać z klauzul `else` i `finally`:

```
list = [1, 2, 3]
try:
    list[8]
except:
    print("element poza listą")
else:
    print("element znajduje się na liście")
finally:
    print("ten komunikat będzie zawsze wyświetlany")
```

Klauzula `else` zostanie uruchomiona, w przypadku gdy nie dojdzie do żadnego wyjątku, a klauzula `finally` będzie uruchamiana niezależnie od tego, czy dojdzie do wyjątku, czy nie.

Gdy dojdzie do wyjątku, możesz uzyskać więcej informacji na jego temat, korzystając z obiektu wyjątku, który jest dostępny tylko po zastosowaniu słowa kluczowego `as`. Przyjrzyj się poniższemu kodowi.

```
>>> list = [1, 2, 3]
>>> try:
...     list[8]
... except Exception as e:
...     print("element poza listą")
...     print(e)
...
element poza listą
podany numer indeksu nie znajduje się na liście
>>>
```

Zobacz również

Dokumentacja Pythona zawiera informacje na temat hierarchii klas wyjątków: <https://docs.python.org/2/library/exceptions.html>.

7.11. Stosowanie modułów

Problem

W swoim programie chcesz zastosować moduł.

Rozwiązanie

Skorzystaj z polecenia `import`:

```
import random
```

Omówienie

Istnieje wiele modułów (czasem nazywanych **bibliotekami**) przeznaczonych dla Pythona. Wiele z nich jest dołączanych do niego standardowo. Możesz je również pobierać z internetu, a następnie instalować.

Standardowe biblioteki Pythona zawierają moduły pozwalające na generowanie losowych liczb, uzyskiwanie dostępu do baz danych, obsługę protokołów internetowych, serializację obiektów itd.

Jedną z konsekwencji posiadania tak wielu modułów jest potencjalne zagrożenie konfliktiem. Na przykład dwa moduły mogą mieć różne funkcje o tych samych nazwach. W celu uniknięcia konfliktu możesz określić dostępną część modułu podczas jego importowania.

Jeżeli na przykład zastosujesz następujące polecenie:

```
import random
```

to zaistnienie konfliktu nie będzie możliwe, bo dostęp do funkcji i zmiennych znajdujących się w module jest możliwy tylko po zastosowaniu prefiku `random.`. Tak się składa, że będziemy korzystać z tego modułu w kolejnej recepturze.

Jeżeli jednak zastosujesz poniższe polecenie, to będziesz mógł uzyskać dostęp do wszystkiego, co znajduje się w module, bez potrzeby stosowania jakichkolwiek prefiksów. Jeżeli nie znasz wszystkich funkcji znajdujących się w module, z którego korzystasz, istnieje duże prawdopodobieństwo powstania konfliktu.

```
from random import *
```

Istnieje jeszcze pewna mniej skrajna technika pozwalająca określać komponenty modułu, z których chcesz korzystać, a które będą mogły być używane w sposób wygodny bez potrzeby stosowania prefiksów.

Na przykład:

```
>>> from random import randint
>>> print(randint(1,6))
2
>>>
```

Alternatywnym rozwiązaniem jest zastosowanie słowa kluczowego `as`, które pozwoli Ci na określenie innej nazwy modułu, którą będziesz mógł stosować, odwołując się do niego.

```
>>> import random as R
>>> R.randint(1, 6)
```

Zobacz również

Listę wszystkich modułów dołączonych do środowiska programistycznego Pythona znajdziesz pod adresem <https://docs.python.org/2/library/>.

7.12. Liczby losowe

Problem

Chcesz wygenerować losowo liczbę znajdująca się w określonym przedziale.

Rozwiązanie

Zastosuj bibliotekę `random`:

```
>>> import random  
>>> random.randint(1, 6)  
2  
>>> random.randint(1, 6)  
6  
>>> random.randint(1, 6)  
5
```

Wygenerowana liczba będzie się znajdowała w podanym przedziale (jest on obustronnie domknięty). Podany kod symuluje działanie kostki do gry.

Omówienie

Wygenerowane liczby nie będą tak naprawdę losowe — będą to tak zwane **liczby pseudolosowe**. Jeżeli zbierze się dużą sekwencję takich liczb, to statystycznie będą one **rozłożone losowo** na pewnym zbiorze. Liczby takie dobrze sprawdzają się w grach, ale gdybyś chciał wygenerować liczby na potrzeby loterii, musiałbyś skorzystać z wyspecjalizowanego sprzętowego generatatora liczb losowych. Losowość nie jest czymś naturalnym w świecie komputerów, a więc nie radzą sobie one z tym zbyt dobrze.

Liczby losowe stosuje się często w celu wybrania losowego elementu z listy. Możesz to zrobić, generując losową liczbę, a następnie stosując ją w roli indeksu. Istnieje jednak również przeznaczony do tego moduł `random`. Wypróbuj działanie poniższego przykładowego kodu:

```
>>> import random  
>>> random.choice(['a', 'b', 'c'])  
'a'  
>>> random.choice(['a', 'b', 'c'])  
'b'  
>>> random.choice(['a', 'b', 'c'])  
'a'
```

Zobacz również

Biblioteka `random` ma również inne przydatne funkcje, które mogą się przydać do takich zadań jak serializacja losowych elementów. Więcej informacji o tej bibliotece znajdziesz na stronie <https://docs.python.org/2/library/random.html>.

7.13. Wysyłanie żądań do sieci Web

Problem

Chcesz skorzystać z Pythona w celu wczytania treści strony internetowej do łańcucha.

Rozwiązanie

Python ma wbudowaną obszerną bibliotekę przeznaczoną do tworzenia żądań HTTP.

Poniższy przykładowy kod wczytuje zawartość strony głównej Google do łańcucha `contents`:

```
import urllib2
contents = urllib2.urlopen("https://www.google.com/").read()
print(contents)
```

Omówienie

Po odczytaniu treści dokumentu HTML prawdopodobnie będziesz chciał ją przeszukać, a następnie wyciąć z niej potrzebne elementy. W tym celu przydadzą Ci się funkcje przeprowadzające operacje na łańcuchach (zobacz receptury 5.14 i 5.15).

Zobacz również

Przykład zastosowania żądań sieciowych w celu odebrania wiadomości z poczty Gmail znajdziesz w recepturze 7.15.

7.14. Argumenty Pythona w wierszu poleceń

Problem

Chcesz uruchomić program napisany w Pythonie za pomocą wiersza poleceń i jednocześnie przekazać do tego programu pewne parametry.

Rozwiązanie

Skorzystaj z wyrażeń `sys` i `argv`, tak jak pokazano w poniższym przykładzie. Zostanie zwrócona tablica, której pierwszym elementem będzie nazwa programu. Pozostałe elementy będą parametrami (rozdzielonymi za pomocą spacji), które zostały wpisane w wierszu poleceń po nazwie programu.

```
import sys

for (i, value) in enumerate(sys.argv):
    print("arg: %d %s " % (i, value))
```

Uruchomienie programu z wiersza poleceń i podanie w nim dodatkowych parametrów spowoduje wygenerowanie następujących danych wyjściowych:

```
$ python cmd_line.py a b c
arg: 0 cmd_line.py
arg: 1 a
arg: 2 b
arg: 3 c
```

Omówienie

Umiejętność stosowania argumentów w wierszu poleceń może się przydać podczas automatyzacji uruchamiania programów Pythona — zarówno w trakcie uruchamiania Raspberry Pi (zobacz receptura 3.20), jak i planowania automatycznego włączania aplikacji o określonej porze (zobacz receptura 3.21).

Zobacz również

Podstawowe informacje dotyczące uruchamiania aplikacji Pythona z poziomu wiersza poleceń znajdziesz w recepturze 5.4.

W celu wyświetlenia efektów działania polecenia `argv` ponumerowaliśmy elementy listy (zobacz receptura 6.8).

7.15. Wysyłanie wiadomości pocztą elektroniczną z poziomu aplikacji Pythona

Problem

Chcesz wysyłać e-maile za pomocą programu napisanego w Pythonie.

Rozwiązanie

Python ma wbudowaną bibliotekę obsługującą protokół komunikacyjny SMTP (`smtplib`). Skorzystaj z niej w celu wysyłania e-maili:

```
import smtplib

GMAIL_USER = 'twój_adres@gmail.com'
GMAIL_PASS = 'twoje_hasło'
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587

def send_email(recipient, subject, text):
    smtpserver = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    smtpserver.ehlo()
    smtpserver.starttls()
    smtpserver.ehlo
    smtpserver.login(GMAIL_USER, GMAIL_PASS)
    header = 'Do:' + recipient + '\n' + 'Od:' + GMAIL_USER
    header = header + '\n' + 'Temat:' + subject + '\n'
    msg = header + '\n' + text + '\n\n'
    smtpserver.sendmail(GMAIL_USER, recipient, msg)
    smtpserver.close()

send_email(docelowy_adres_email', 'temat', 'treść')
```

Za pomocą tego przykładowego kodu możesz wysyłać wiadomości na dowolny adres, ale najpierw musisz zmodyfikować zmienne `GMAIL_USER` i `GMAIL_PASS` — podać hasło i nazwę użytkownika swojej skrzynki. Jeżeli nie korzystasz z poczty Gmail, będziesz musiał zmienić również dane dotyczące serwera SMTP (`SMTP_SERVER`) oraz prawdopodobnie numer portu (`SMTP_PORT`).

W ostatniej linii kodu należy podać adres, pod który chcesz wysłać wiadomość.

Omówienie

W podanym przykładzie obsługę biblioteki `smtplib` zamknięto w jednej funkcji `send_email`. Możesz ją wywołać w dowolnym miejscu programu.

Możliwość wysyłania wiadomości za pomocą programów pisanych w Pythonie otwiera przed Tobą wiele nowych możliwości. Możesz np. stworzyć aplikację, która będzie wysyłała wiadomości, gdy czujnik ruchu podłączony do Raspberry Pi wykryje ruch (zobacz receptura 11.9).

Zobacz również

Wysyłanie żądań HTTP omówiono w recepturze 7.13.

Więcej informacji na temat biblioteki `smtplib` znajdziesz na stronie <https://docs.python.org/2/library/smtplib.html>.

7.16. Prosty serwer sieci Web napisany w Pythonie

Problem

Chcesz stworzyć prosty serwer sieci Web w Pythonie, ale nie chcesz uruchamiać całego rozbudowanego środowiska serwerowego.

Rozwiązanie

W celu uruchomienia serwera sieci Web, który będzie odpowiadał na żądania HTTP, skorzystaj z biblioteki `bottle`.

Zainstaluj bibliotekę `bottle` za pomocą następującego polecenia:

```
sudo apt-get install python-bottle
```

Poniższy program (w materiałach dodatkowych znajduje się on w pliku `bottle_test.py`) uruchamia serwer, który wyświetla aktualną datę i godzinę. Na rysunku 7.1 przedstawiono stronę, jaka zostanie wyświetlona po wywołaniu adres IP Raspberry Pi w przeglądarce zainstalowanej na dowolnym komputerze znajdującym się w Twojej sieci domowej.

```
from bottle import route, run, template
from datetime import datetime

@route('/')
def index(name='time'):
    dt = datetime.now()
    time = "{:%Y-%m-%d %H:%M:%S}".format(dt)
    return template('<b>Pi uważa, że mamy datę i godzinę: {{t}}</b>', t=time)

run(host='192.168.1.16', port=80)
```



Rysunek 7.1. Dokument generowany przez serwer sieci Web napisany w Pythonie

Program musisz uruchomić jako administrator:

```
$ sudo python bottle_test.py
```

Zaprezentowany przykład wymaga pewnych wyjaśnień.

Polecenie @route znajdujące się za poleceniem import łączy adres URL / ze znajdującą się dalej procedurą.

Umieszczona dalej procedura przedstawia datę i godzinę w odpowiednim formacie i zwraca łańcuch zawierający kod HTML, który zostanie wyświetlony przez przeglądarkę. W tym przypadku zastosowano szablon, w który można wstawiać różne wartości.

Na koniec polecenie run uruchamia proces serwera. Pamiętaj o tym, że musisz określić nazwę komputera roboczego i port. Domyślnie do obsługi sieci Web używany jest port o numerze 80. Jeżeli chcesz korzystać z innego portu, musisz podać jego numer za znakiem :. Należy go umieścić po adresie IP serwera, z którym chcesz uzyskać połaczenie.

Omówienie

W programie możesz zdefiniować dowolną ilość operacji i procedur.

Biblioteka bottle znakomicie sprawdza się podczas pracy nad prostymi projektami wymagającymi serwera sieci Web. Jest ona napisana w Pythonie, a więc utworzenie funkcji obsługującej reakcje sprzętu na czynności wykonywane przez użytkownika strony jest bardzo łatwe.

Zobacz również

Więcej szczegółowych informacji znajdziesz w dokumentacji biblioteki bottle na stronie <http://bottlepy.org/docs/dev/>.

Dodatkowe informacje dotyczące formatowania daty i godziny znajdziesz w receptorze 7.2.

Podstawowe zagadnienia dotyczące złącza GPIO

8.0. Wprowadzenie

W tym rozdziale omówiono podstawowe informacje dotyczące konfiguracji i zastosowania złącza GPIO. Jest to interfejs wejścia i wyjścia znajdujący się na płytce Raspberry Pi.

8.1. Styki złącza GPIO

Problem

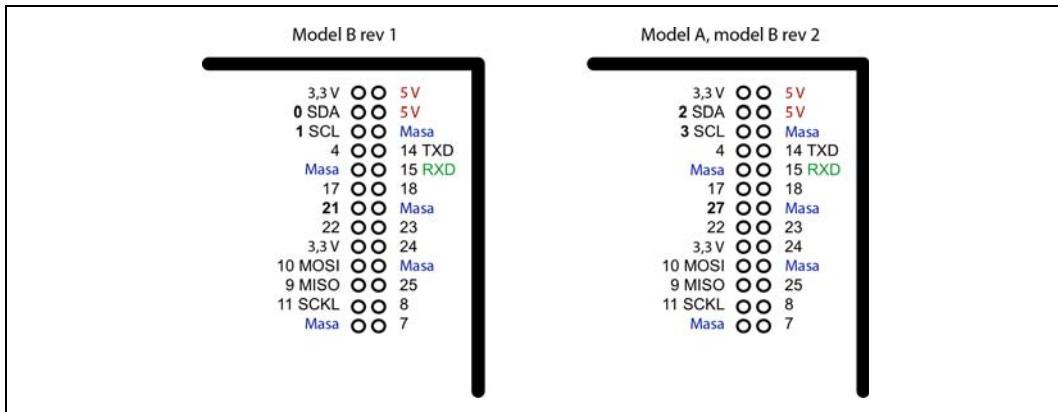
Chcesz podłączać urządzenia elektroniczne do złącza GPIO, ale nie wiesz, do czego służą styki tego złącza.

Rozwiążanie

Na rysunku 8.1 przedstawiono schemat styków złączy GPIO różnych modeli i wersji Raspberry Pi. Wersje modelu B najprościej odróżnić od siebie, patrząc na gniazdo audio. Starsza płytka — rev 1 — ma gniazdo koloru czarnego, a nowsza — rev 2 — ma niebieskie.

Różnice pomiędzy złączami na poszczególnych płytach zaznaczono pogrubioną czcionką (zobacz rysunek 8.1). Zamieniony został port I2C. Styki *SDA* i *SCL* są w tym samym miejscu, ale używają innego wewnętrznego interfejsu I2C. A zatem jeżeli korzystasz z tych styków jako części złącza GPIO, a nie magistrali I2C, to będziesz się do nich odwoływał tak jak do styków 2 i 3 znajdujących się na płytce rev 2. Na płytce rev 2 styl GPIO 21 został zastąpiony stykiem GPIO 27.

W górnej części złącza znajdują się styki zasilające prądem o napięciu 3,3 V i 5 V. Wszystkie wejścia i wyjścia interfejsu GPIO korzystają z napięcia 3,3 V. Każdy ze styków oznaczonych numerem może działać w charakterze złącza GPIO. Styki, które są oznaczone zarówno numerem, jak i skrótowną nazwą, mogą służyć również do określonych innych celów. Piny 14 *TXD* i 15 *RXD* służą do transmisji i odbierania danych za pomocą interfejsu szeregowego. Magistrala I2C korzysta ze styków *SDA* i *SCL*. Interfejsowi SPI przypisane są złącza *MOSI*, *MISO* i *SCKL*.



Rysunek 8.1. Schemat złącza GPIO

Omówienie

Styk interfejsu GPIO może pracować jako cyfrowe wejście oraz jako cyfrowe wyjście (w obu przypadkach piny będą pracowały pod napięciem 3,3 V). Raspberry Pi, w przeciwieństwie do Arduino, nie posiada żadnych wejść analogowych. W celu korzystania z takiego wejścia musiałbyś użyć zewnętrznego przetwornika analogowo-cyfrowego lub podłączyć Raspberry Pi do specjalnej płytki interfejsu, takiej jak np. Gertboard. Możesz też korzystać z płyt Arduino lub aLaMode — problematykę tę opisano szerzej w rozdziale 14.

Zobacz również

Więcej informacji dotyczących podłączania przetwornika analogowo-cyfrowego do Raspberry Pi znajdziesz w receptorze 12.4.

8.2. Bezpieczne korzystanie ze złącza GPIO

Problem

Chcesz podłączyć do Raspberry Pi jakieś inne urządzenie elektroniczne, ale nie chcesz go przypadkowo uszkodzić ani zepsuć.

Rozwiążanie

Poniżej znajdują się zasady. Przestrzeganie ich znacząco obniży ryzyko uszkodzenia Raspberry Pi podczas korzystania ze złącza GPIO.

- Do żadnego ze styków złącza GPIO nie podłączaj prądu o napięciu wyższym niż 3,3 V.
- Nie pobieraj z żadnego ze złączy wyjściowych prądu o natężeniu większym niż 3 mA. Co prawda *uda Ci się* pobrać prąd o większym natężeniu, może to jednak skrócić żywotność Raspberry Pi. Prąd o natężeniu 3 mA wystarczy, by zasilić czerwoną diodę LED wraz z połączonym z nią szeregowo rezystorem 470 Ω.

- Gdy Raspberry Pi jest podłączone do zasilania, nie dotykaj styków złącza GPIO śrubokrętem ani innym metalowym przedmiotem.
- Nie zasilaj Raspberry Pi prądem o napięciu wyższym niż 5 V.
- Dbaj o to, by urządzenia podłączone do styków zasilających prądem o napięciu 3,3 V nie pobierały w sumie prądu o natężeniu większym niż 50 mA.
- Dbaj o to, by urządzenia podłączone do styków zasilających prądem o napięciu 5 V nie pobierały w sumie prądu o natężeniu większym niż 250 mA.

Omówienie

Raspberry Pi może dość łatwo ulec uszkodzeniu na skutek współpracy z zewnętrzną elektroniką. Zachowaj ostrożność. Zawsze przed zasileniem Raspberry Pi sprawdzaj poprawność wykonanych połączeń. Pozwoli to uniknąć uszkodzenia Raspberry Pi.

Zobacz również

Zapoznaj się z bardzo dobrym artykułem dotyczącym zasilania elektroniki za pośrednictwem płytka Raspberry Pi — http://www.thebox.myzen.co.uk/Raspberry/Understanding_Outputs.html.

8.3. Instalacja biblioteki RPi.GPIO

Problem

Chcesz skonfigurować styki wyjściowe złącza GPIO, a także wczytywać dane za pomocą styków wejściowych.

Rozwiążanie

Pobierz i zainstaluj bibliotekę RPi.GPIO.

W oknie Terminala wpisz poniższe polecenia. Biblioteka RPi.GPIO zostanie pobrana i zainstalowana na Twoim Raspberry Pi.

```
$ sudo apt-get install python-dev  
$ sudo apt-get install python-rpi.gpio
```

Wiele najnowszych dystrybucji systemów operacyjnych dysponuje wbudowaną biblioteką RPi.GPIO. W takim przypadku powyższe polecenia spowodują po prostu jej uaktualnienie.

Omówienie

Będziemy używać tej biblioteki także w innych rozdziałach tej książki. Biblioteka korzysta z języka C opakowanego Pythonem — dzięki temu interfejs wejścia-wyjścia będzie działał tak szybko, jak to tylko możliwe. Raspberry Pi nie zostało zaprojektowane do pełnienia roli mikrokontrolera, a więc jego złącza nie będą odpowiadały tak szybko jak złącza Arduino.

Zobacz również

Alternatywną biblioteką pełniąącą te same funkcje co RPi.GPIO jest WiringPi. Więcej informacji na jej temat znajdziesz w witrynie Gordons Projects pod adresem <https://projects.drogon.net/raspberry-pi/wiringpi/>.

W wielu recepturach z tej książki korzystamy z biblioteki RPi.GPIO.

8.4. Konfiguracja magistrali I2C

Problem

Posiadasz urządzenie wyposażone w magistralę I2C i chcesz podłączyć je do swojego Raspberry Pi, a więc chcesz wiedzieć, jak włączyć obsługę tego interfejsu.

Rozwiązanie

Jeżeli korzystasz z systemu Adafruit Occidentalis w wersji 0,2 lub nowszej, to nie musisz niczego robić. System jest przygotowany do obsługi magistrali I2C.

W nowszych wersjach tego systemu prawdopodobnie nie będziesz musiał wykonywać opisanych tutaj czynności, ale jeśli je przeprowadzisz, upewnij się, że posiadasz najnowszą wersję dystrybucji.

Jeżeli korzystasz z systemu Raspbian, musisz odpowiednio skonfigurować tę dystrybucję.

Zacznij od edytowania pliku `/etc/modules` — zastosuj w tym celu polecenie `sudo nano /etc/modules` i dodaj poniższe linie kodu na końcu tego pliku.

```
i2c-bcm2708  
i2c-dev
```

Prawdopodobnie będziesz musiał również edytować plik `/etc/modprobe.d/raspi-blacklist.conf` i oznaczyć poniższą linię kodu jako komentarz.

```
blacklist i2c-bcm2708
```

Dodaj znak `#` na początku tej linii:

```
#blacklist i2c-bcm2708
```

Jeżeli planujesz korzystać z interfejsu SPI (zobacz receptura 8.6), to jako komentarz powinieneś oznaczyć linię dotyczącą wpisania na *czarną listę* (ang. *blacklist*) również tego złącza.

W celu zainstalowania biblioteki I2C skorzystaj z poniższego polecenia.

```
$ sudo apt-get install python-smbus
```

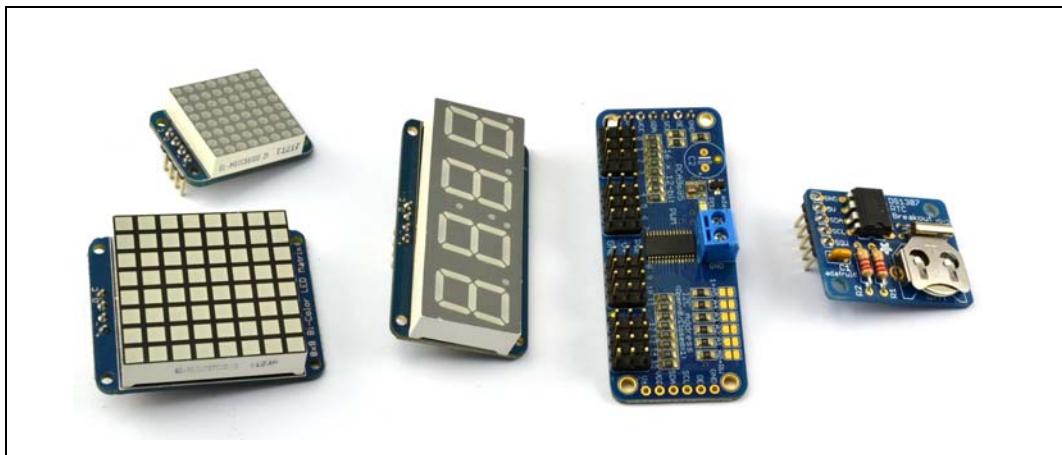
Uruchom ponownie Raspberry Pi. Teraz będziesz mógł rozpocząć pracę z magistralą I2C.

Omówienie

Korzystanie z modułów I2C jest naprawdę dobrym sposobem na komunikowanie się z Raspberry Pi. Ogranicza to liczbę kabli połączeniowych do tylko czterech. Istnieje wiele naprawdę świetnych gotowych modułów I2C.

Nie zapominaj o tym, żeby obliczyć sumę natężenia prądu pobieranego przez moduły I2C i sprawdzić, czy wartość ta nie przekracza dopuszczalnych wielkości podanych w recepturze 8.2.

Na rysunku 8.2 pokazano część modułów sprzedawanych przez firmę Adafruit. Inne firmy, takie jak np. SparkFun, również mają w swojej ofercie urządzenia I2C. Na rysunku widać (od strony lewej do prawej): wyświetlacze matrycowe; czterocyfrowy siedmiosegmentowy wyświetlacz diodowy; oparty na technologii PWM szesnastokanałowy sterownik serwomechanizmów; moduł zegara czasu rzeczywistego.



Rysunek 8.2. Moduły I2C

Inne dostępne moduły I2C to nadajniki radiowe FM, dalmierze ultradźwiękowe, wyświetlacze OLED i różne czujniki.

Zobacz również

Zobacz również inne receptury dotyczące magistrali I2C — 10.2, 13.1, 13.2 i 13.4.

8.5. Korzystanie z narzędzi I2C

Problem

Podłączyleś do Raspberry Pi jakieś urządzenie za pośrednictwem magistrali I2C. Chcesz sprawdzić, czy jest ono prawidłowo podłączone, i ustalić jego adres.

Rozwiążanie

Skorzystaj z narzędzi `i2c-tools`.



Narzędzia te mogą być wbudowane w nowsze dystrybucje systemów operacyjnych.

Pobierz i zainstaluj *i2c-tools*, uruchamiając w oknie Terminala następujące polecenie:

```
$ sudo apt-get install i2c-tools
```

Podłącz urządzenie korzystające z interfejsu I2C do swojego Raspberry Pi, a następnie uruchom polecenie:

```
$ sudo i2cdetect -y 1
```

Jeżeli posiadasz starszą wersję płytka (rev 1), to w powyższym poleceniu musisz zmienić wartość 1 na 0.

Jeżeli magistrala I2C jest dostępna, wyświetli Ci się coś podobnego do danych pokazanych na rysunku 8.3. Komunikat ten informuje o tym, że w użyciu są dwa adresy magistrali I2C: 0x40 i 0x70.

```
pi@raspberrypi: ~ $ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  68  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi: ~ $
```

Rysunek 8.3. Narzędzia *i2c-tools*

Omówienie

Narzędzie diagnostyczne *i2cDetect* jest przydatnym programem diagnostycznym. Warto go uruchamiać za każdym razem, gdy po raz pierwszy korzystasz z jakiegoś urządzenia wyposażonego w magistralę I2C.

Zobacz również

Zobacz również inne receptury dotyczące magistrali I2C — 10.2, 13.1, 13.2 i 13.4.

Więcej informacji dotyczących instalowania oprogramowania za pomocą polecenia *apt-get* znajdziesz w recepturze 3.16.

8.6. Przygotowanie do pracy interfejsu SPI

Problem

Chcesz korzystać z zewnętrznego interfejsu szeregowego — SPI.

Rozwiązanie

Jeżeli korzystasz z systemu Adafruit Occidentalis w wersji 0,2 lub nowszej, to nie musisz niczego robić. System będzie domyślnie skonfigurowany, tak aby obsługiwał interfejs SPI.

Jeżeli korzystasz z systemu Raspbian, musisz odpowiednio skonfigurować tę dystrybucję.

Zacznij od edycji pliku `/etc/modules` — zastosuj w tym celu polecenie `sudo nano /etc/modules` i dodaj poniższą linię kodu na końcu tego pliku.

```
spidev
```

Prawdopodobnie będziesz musiał również edytować plik `/etc/modprobe.d/raspi-blacklist.conf` i oznaczyć poniższą linię kodu jako komentarz.

```
blacklist spi-bcm2708
```

Dodaj znak # na początku tej linii:

```
# blacklist spi-bcm2708
```

Jeżeli planujesz korzystać z magistrali I2C, to jako komentarz powinieneś również oznaczyć linię dotyczącą tego interfejsu, która jest wpisana na czarną listę (ang. *blacklist*).

Interfejs SPI jest obsługiwany przez specjalną bibliotekę Pythona, która zapewnia komunikację pomiędzy urządzeniami podłączonymi do niego a programami napisanymi w Pythonie. Aby pobrać tę bibliotekę, najpierw musisz zainstalować narzędzie Git (zobacz receptura 3.19). Po wykonaniu tej czynności możesz wpisać w oknie Terminala następujące polecenia:

```
$ cd ~  
$ sudo apt-get install python-dev  
$ git clone git://github.com/doceme/py-spidev  
$ cd py-spidev/  
$ sudo python setup.py install
```

Uruchom ponownie Raspberry Pi. Teraz możesz już korzystać z interfejsu SPI.

Omówienie

Interfejs SPI pozwala na szeregowe przesyłanie danych pomiędzy Raspberry Pi a urządzeniami zewnętrznymi, takimi jak przetworniki analogowo-cyfrowe, układy portów rozszerzeń itp.

Mözesz trafić na przykłady korzystania ze styków interfejsu SPI w sposób niestandardowy — bez użycia samego interfejsu. W technice zwanej *bit banging* używa się biblioteki `RPi.GPIO` w celu przesyłania sygnału przez cztery styki złącza GPIO, z których standardowo korzysta interfejs SPI.

Zobacz również

Korzystanie z układu przetwornika analogowo-cyfrowego podłączonego za pośrednictwem interfejsu SPI omówiono w recepturze 12.4.

8.7. Zwalnianie portu szeregowego

Problem

Chcesz korzystać z portu szeregowego (styki oznaczone jako *Rx* i *Tx*) w swoich projektach, ale Linux używa tego portu jako połączenia konsolowego.

Rozwiążanie

Domyślnie port szeregowy działa jako konsola służąca do sterowania pracą Raspberry Pi za pośrednictwem specjalnego kabla szeregowego (zobacz receptura 2.6).

Aby wyłączyć tę funkcję w systemie operacyjnym i móc podłączać do portu szeregowego urządzenia takie jak np. GPS (zobacz receptura 11.10), musisz oznaczyć jako komentarz jedną z linii pliku */etc/inittab*:

```
$ sudo nano /etc/inittab
```

Na końcu pliku odnajdź wiersz:

```
T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Dodaj znak # na początku tej linii (oznacz ją jako komentarz):

```
# T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
```

Zapisz plik — wciśnij *Ctrl+X*, a następnie klawisz *Y*.

Aby zmiany zostały zastosowane, należy ponownie uruchomić Raspberry Pi. W tym celu wpisz polecenie *sudo reboot*.

Omówienie

Efektem ubocznym tej zmiany będzie to, że nie będziesz się mógł połączyć z Raspberry Pi za pomocą kabla konsolowego. Wciąż jednak będziesz mógł korzystać z połączenia za pośrednictwem sieci Wi-Fi lub Ethernet.

Gdybyś w przyszłości chciał korzystać z kabla konsolowego, usuń znak #, który wstawiłeś do pliku, i ponownie uruchom Raspberry Pi.

Zobacz również

Będziesz musiał wykonać przedstawioną tutaj operację, aby móc korzystać z receptur dotyczących podłączania sprzętu do portu szeregowego (np. z receptury 11.10). Wiele receptur przedstawionych w rozdziale 14. dotyczy komunikacji z Arduino za pośrednictwem portu szeregowego.

8.8. Instalowanie biblioteki PySerial pozwalającej na korzystanie z portu szeregowego przez aplikacje Pythona

Problem

Chcesz korzystać z portu szeregowego (styki Rx i Tx) w aplikacjach pisanych w Pythonie.

Rozwiążanie

Zainstaluj bibliotekę *PySerial*:

```
$ sudo apt-get install python-serial
```

Przed zastosowaniem tej biblioteki w praktyce w swoich projektach musisz wyłączyć obsługę połączenia konsolowego (zobacz receptura 8.7).

Omówienie

Stosowanie tej biblioteki jest dość proste. Polecenie utworzenia połączenia ma następującą składnię:

```
ser = serial.Serial(URZĄDZENIE, PRZEPUSTOWOŚĆ)
```

W miejscu hasła *URZĄDZENIE* należy określić adres urządzenia komunikującego się za pośrednictwem portu szeregowego (*/dev/ttyAMA0*), a w miejscu hasła *PRZEPUSTOWOŚĆ* podać przepustowość wyrażoną liczbą. Nie może być ona wyrażona ciągiem, np.:

```
ser = serial.Serial('/dev/ttyAMA0', 9600)
```

Gdy połączenie zostanie nawiązane, możesz przesyłać dane przez port szeregowy:

```
ser.write('przykładowy tekst')
```

Oczekiwanie na sygnał zwrotny wymaga stworzenia pętli, która odczytuje sygnały i wyświetla komunikaty, np.:

```
while True:  
    print(ser.read())
```

Zobacz również

Będziesz musiał wykonać przedstawioną tutaj operację, aby móc korzystać z receptur dotyczących podłączania sprzętu do portu szeregowego (np. z receptury 11.10).

8.9. Testowanie portu szeregowego za pomocą aplikacji Minicom

Problem

Chcesz wysyłać i odbierać polecenia przez port szeregowy za pomocą okna Terminala.

Rozwiążanie

Zainstaluj narzędzie Minicom:

```
$ sudo apt-get install minicom
```

Zanim będziesz mógł korzystać z komunikacji przez port szeregowy za pomocą aplikacji Minicom, musisz wyłączyć obsługę konsoli szeregowej w systemie operacyjnym Raspberry Pi (zobacz receptura 8.7).

Po zainstalowaniu programu Minicom możesz uruchomić komunikację z urządzeniami podłączonymi do portu szeregowego poprzez styki RXD i TXD złącza GPIO. Wystarczy skorzystać z polecenia:

```
$ minicom -b 9600 -o -D /dev/ttyAMA0
```

Po -b należy podać przepustowość, a po -D określić port szeregowy. Musisz zadeklarować taką samą prędkość transmisji danych, jaką ustawiono w urządzeniu, z którym próbujesz się połączyć.

Zostanie otwarta sesja programu Minicom. Pierwszą rzeczą, jaką powinieneś zrobić, jest włączenie funkcji local echo — dzięki niej będziesz widzieć wpisywane przez siebie polecenia. Aby włączyć tę funkcję, wciśnij kombinację klawiszy *Ctrl+A*, a następnie klawisz Z. Zobaczysz menu pokazane na rysunku 8.4. Wciśnij klawisz E, aby włączyć wspomnianą funkcję.



Rysunek 8.4. Polecenia aplikacji Minicom

Teraz wszystko to, co wpiszesz, zostanie wysłane do urządzenia podłączonego do portu szeregowego, a wszystkie komunikaty generowane przez to urządzenie będą wyświetlane za pośrednictwem Raspberry Pi.

Omówienie

Minicom jest narzędziem, które świetnie się sprawdza, w przypadku gdy chcesz wyświetlić komunikaty generowane przez urządzenie podłączone do portu szeregowego lub upewnić się, że działa ono poprawnie. Prawdopodobnie chcesz napisać własny program obsługujący komunikację za pomocą portu szeregowego — w tym celu musisz zainstalować odpowiednią bibliotekę Pythona (zobacz receptura 8.8).

Zobacz również

Zajrzyj do dokumentacji programu Minicom — <http://linux.die.net/man/1/minicom>. W recepturze 13.3 przedstawiono przykład zastosowania aplikacji Minicom.

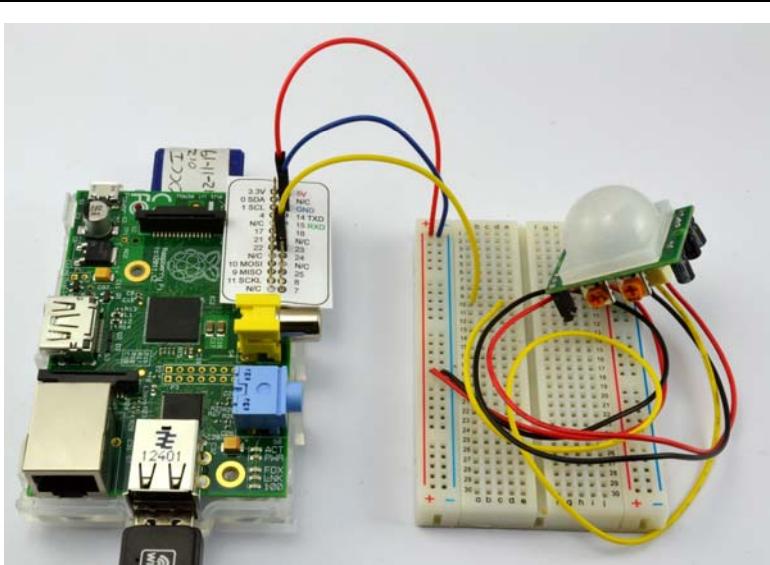
8.10. Łączenie Raspberry Pi z płytą prototypową za pomocą przewodów połączeniowych

Problem

Chcesz wykonać prototyp jakiegoś urządzenia elektrycznego, korzystając z możliwości Raspberry Pi i płytki prototypowej.

Rozwiązanie

Skorzystaj z przewodów połączeniowych zakończonych żeńskimi końcówkami do tzw. gold-pinów. Na złącze GPIO nałożź wydrukowany szablon opisujący styki.



Rysunek 8.5. Raspberry Pi połączone z płytą prototypową za pomocą przewodów połączeniowych

Omówienie

Identyfikacja styków złącza GPIO jest dość trudna. Możesz wydrukować sobie papierową nakładkę na złącze GPIO, która będzie zawierała opisy styków. Szablon takiej nakładki możesz pobrać ze strony internetowej <http://www.doctormonk.com/2013/02/raspberry-pi-and-breadboard-raspberry.html>.

Warto się zaopatrzyć w zapas kabli połączeniowych zakończonych z obu stron końcówkami męskimi. Przydadzą Ci się one do wykonywania połączeń pomiędzy częściami płytka prototypowej.

Przewody połączeniowe zakończone po obu stronach końcówkami żeńskimi przydadzą Ci się do podłączania bezpośrednio do Raspberry Pi modułów posiadających wyjścia w postaci goldpinów (gdy nie musisz korzystać z pośrednictwa płytka prototypowej). Na rysunku 8.5 pokazano przykład połączenia Raspberry Pi z płytka prototypową.

Zobacz również

Podłączenie elementów za pomocą przewodów sprawdza się w przypadku małej liczby kabli. Jeżeli musisz podłączyć do złącza GPIO wiele przewodów, lepiej skorzystaj z modułu Pi Cobbler (zobacz receptura 8.11).

8.11. Łączenie modułu Pi Cobbler z płytka prototypową

Problem

Chcesz wykonać prototyp jakiegoś układu elektronicznego, korzystając z Raspberry Pi i płytka prototypowej.

Rozwiązanie

Zastosuj Pi Cobbler — moduł składający się z małej płytka drukowanej obwodu wyposażonej w piny ułożone w ten sposób, że można ten moduł wpiąć w płytka prototypową, tak jakby był on układem scalonym zamkniętym w obudowie typu DIL (podłużnej dwurzędowej). Styki są odpowiednio oznaczone, a płytka drukowana ma złącze, do którego wpinany jest kabel wstępny służący do podłączania jej do Raspberry Pi (zobacz rysunek 8.6).

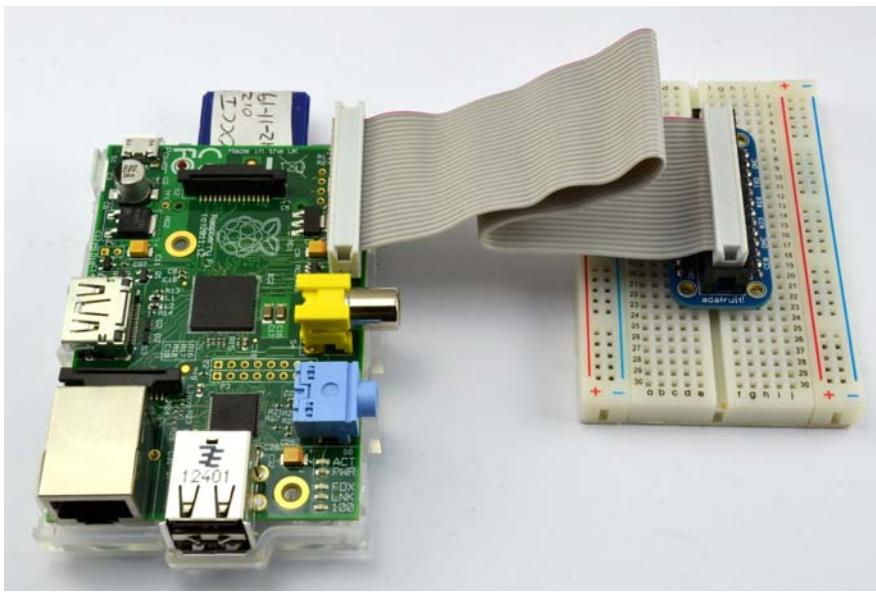
Omówienie

Pi Cobbler pozwala na podłączenie Raspberry Pi do płytka prototypowej po umieszczeniu w niej wszystkich komponentów. To duża zaleta tego modułu.

Wpinając kabel w złącze GPIO, upewnij się, że czerwony koniec kabla jest zwrócony w stronę gniazda karty SD. Kabel można wpiąć w to złącze tylko w jeden sposób.

Zobacz również

Zobacz również recepturę 8.10.



Rysunek 8.6. Podłączanie Raspberry Pi do płytka prototypowej za pośrednictwem modułu Pi Cobbler

8.12. Zmniejszanie napięcia sygnałów z 5 do 3,3 V za pomocą dwóch rezystorów

Problem

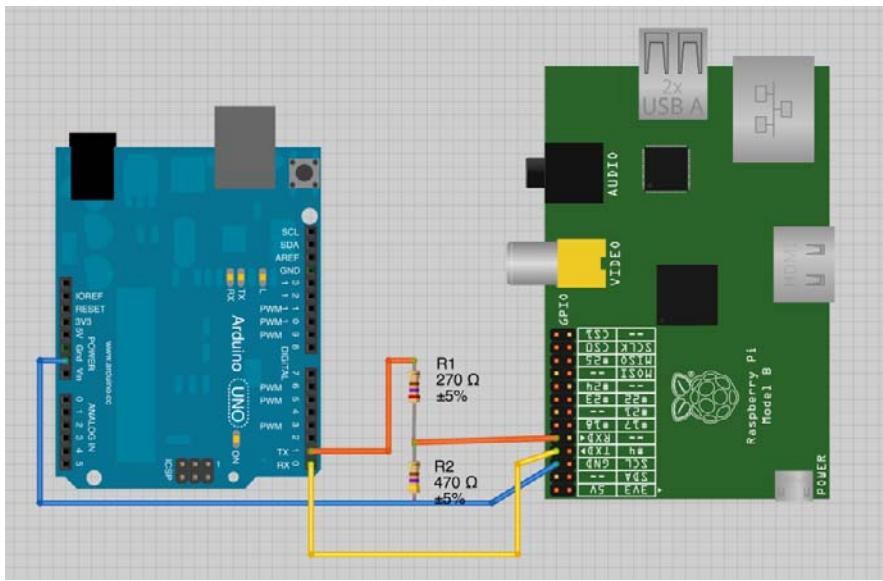
Złącza Raspberry Pi mogą przyjmować sygnał o napięciu 3,3 V, ale Ty chcesz podłączyć do takiego złącza sygnał o napięciu 5 V generowany przez moduł. Nie chcesz przy tym uszkodzić Raspberry Pi.

Rozwiązanie

Obniż napięcie za pomocą pary rezystorów tworzących dzielnicę napięcia. Na rysunku 8.7 pokazano, jak podłączyć do Raspberry Pi sygnał o napięciu 5 V generowany przez wyjście szeregowego Arduino.

W celu wykonania takiego połączenia będziesz potrzebować:

- rezystora charakteryzującego się oporem $270\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- rezystora charakteryzującego się oporem $470\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).



Rysunek 8.7. Obniżanie napięcia sygnału z 5 do 3,3 V za pomocą pary rezystorów

Sygnal wyjściowy generowany przez Raspberry Pi na styku TXD ma napięcie 3,3 V. Można go bezproblemowo podłączyć bezpośrednio do złącza wejściowego Arduino, które może przyjmować sygnały o napięciu 5 V. Złącze to interpretuje jako wysoki każdy sygnał o napięciu wyższym od 2,5 V.

Problem pojawia się, gdy musisz połączyć złącze wyjściowe Arduino (generujące sygnał o napięciu 5 V) ze stykiem RXD Raspberry Pi. Złącze tych nie można łączyć bezpośrednio! Sygnał o napięciu 5 V może doprowadzić do uszkodzenia Raspberry Pi. Zastosuj dwa rezystory, tak jak pokazano na rysunku 8.7.

Omówienie

Zastosowane rezystory pobierają prąd o natężeniu 6 mA. W związku z tym, że Raspberry Pi pobiera prąd o natężeniu 500 mA, wzrost poboru prądu nie będzieauważalny.

Jeżeli chcesz zminimalizować ilość prądu pobieranego przez dzielik napięcia, zastosuj proporcjonalnie rezystory charakteryzujące się wyższym oporem, np. 27 kΩ i 47 kΩ. Dzielik składający się z takich rezystorów będzie pobierał prąd o natężeniu zaledwie 60 µA.

Zobacz również

Więcej informacji na temat łączenia ze sobą płyt Arduino oraz Raspberry Pi znajdziesz w rozdziale 14.

Jeżeli musisz obniżyć napięcie wielu sygnałów, to prawdopodobnie najlepiej będzie, gdy zastosujesz moduł opisany w recepturze 8.13.

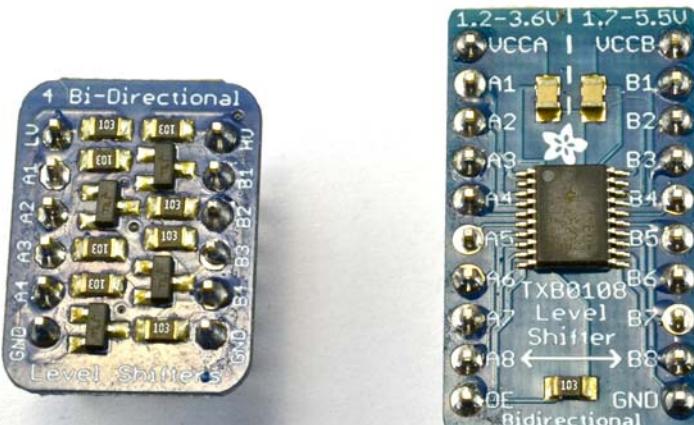
8.13. Korzystanie z modułu przetwornika obniżającego napięcie sygnałów z 5 do 3,3 V

Problem

Złącza Raspberry Pi są przystosowane do odbioru sygnału o napięciu 3,3 V. Chcesz podłączyć do gniazda GPIO wiele cyfrowych linii sygnałowych o napięciu 5 V, nie uszkadzając przy tym Raspberry Pi.

Rozwiążanie

Skorzystaj z dwukierunkowego modułu przetwornika. Przykłady takich modułów pokazano na rysunku 8.8.



Rysunek 8.8. Moduły przetworników obniżających napięcie sygnałów z 5 do 3,3 V

Użytkowanie tych modułów jest bardzo proste. Z jednej strony płytka takiego modułu należy podłączyć sygnały charakteryzujące się takim samym napięciem jak napięcie zasilające podłączone do tej strony modułu. Z drugiej strony modułu należy podłączyć źródło zasilania o innym napięciu. Na złączach znajdujących się po tej stronie modułu pojawią się sygnały o odpowiednich napięciach. Złącza mogą służyć zarówno jako wejścia, jak i wyjścia sygnału.

Omówienie

Dostępne przetworniki dysponują różną liczbą kanałów. Na rysunku 8.8 pokazano przetwornik czterokanałowy, a także moduł ośmiokanałowy.

Dostawców takich przetworników wymieniono w sekcji „Moduły” w dodatku A.

Zobacz również

Jeżeli musisz zmniejszyć napięcie tylko jednej lub dwóch linii sygnałowych, to zajrzyj do receptury 8.12.

8.14. Zasilanie Raspberry Pi za pomocą baterii

Problem

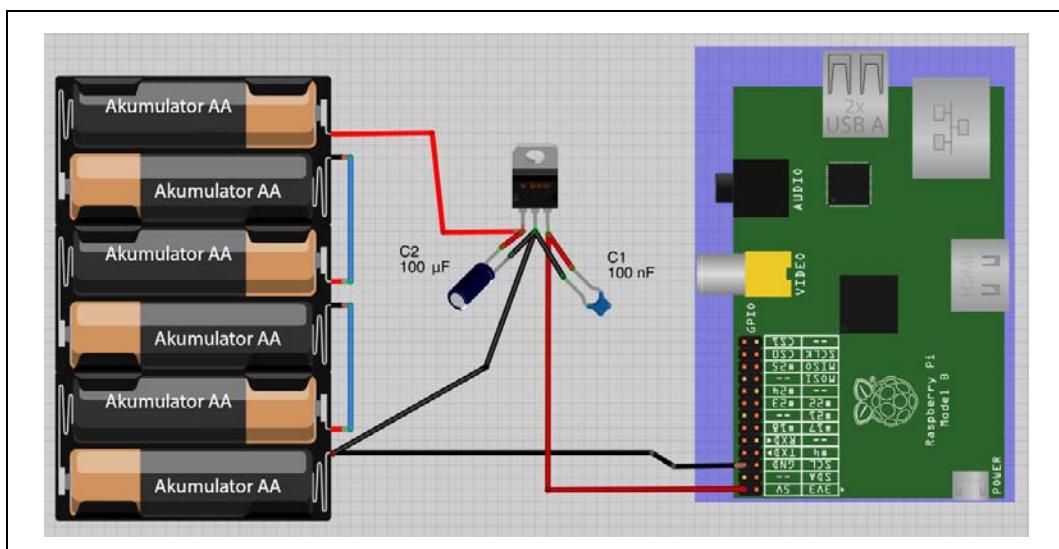
Chcesz zastosować Raspberry Pi w roli elementu składowego robota, a więc musisz zasilać je za pomocą baterii.

Rozwiązanie

Zwykle Raspberry Pi wymaga do pracy prądu o napięciu 5 V i maksymalnym natężeniu około 600 mA (zobacz receptura 1.3). Urządzenie to musi być zasilane prądem o napięciu wynoszącym dokładnie 5 V. Napięcie to nie może być nawet odrobinę mniejsze lub większe od tej wartości. W praktyce oznacza to, że będziesz zasilać Raspberry Pi baterią o napięciu 9 V za pośrednictwem regulatora napięcia obniżającego je do 5 V.

Raspberry Pi pobiera dość dużo prądu, a więc nie można zasilać tego urządzenia małą baterią płaską o napięciu 9 V lub zestawem ogniw AAA. Najsensowniejszym rozwiązaniem jest zbudowanie baterii z sześciu akumulatorów AA i podłączenie do niej regulatora napięcia.

Na rysunku 8.9 pokazano układ zasilający Raspberry Pi składający się z baterii i regulatora napięcia, który jest podłączony do styku 5 V złącza GPIO.



Rysunek 8.9. Zasilanie Raspberry Pi z akumulatorów AA

Omówienie

Regulator napięcia 7805 będzie się prawdopodobnie dość mocno nagrzewał. Jeżeli nagrzeję się zbyt mocno, to zadziała wbudowany w nim układ przeciwdziałający przegrzaniu, który obniży napięcie wyjściowe tego układu, co prawdopodobnie spowoduje wyłączenie Raspberry Pi.

Problem ten można rozwiązać, zakładając radiator na wspomniany układ scalony.

Układ 7805 wymaga, by napięcie wejściowe było przynajmniej o 2 V wyższe od 5 V. Możesz również kupić regulator napięcia typu LDO (charakteryzujący się niskim spadkiem), taki jak np. LM2940. Układ ten charakteryzuje się taką samą konfiguracją złączy jak układ 7805, ale wymaga, żeby napięcie wejściowe było tylko o 0,5 V wyższe od napięcia wyjściowego (5 V). Pamiętaj o tym, że napięcie ogniw AA charakteryzującego się napięciem znamionowym 1,5 V dość szybko spada do poziomu 1,2 V. A zatem bateria składająca się z czterech takich ogniw będzie w stanie zapewnić właściwe napięcie tylko przez bardzo krótki czas. Zastosowanie układu złożonego z sześciu ogniw jest o wiele lepszym rozwiązaniem.

Przedstawiony układ możesz również zastosować w celu zasilania Raspberry Pi w samochodzie lub w przyczepie kempingowej. Dodatkowo będziesz potrzebował małego monitora, który może być zasilany prądem stałym (DC). Takie urządzenia są łatwe do znalezienia, ponieważ są powszechnie stosowane w sieciach telewizji przemysłowej.

Zobacz również

Możesz również poszukać gotowej ładowarki telefonów komórkowych, która może być zasilana z baterii AA. Kabel wyjściowy takiej ładowarki powinien być zakończony wtyczką mini USB. Przed zakupem sprawdź, czy ładowarka jest w stanie dostarczyć prąd o natężeniu przynajmniej 600 mA.

Problematykę dotyczącą zasilania Raspberry Pi z akumulatora LiPo poruszono w recepcie 8.15.

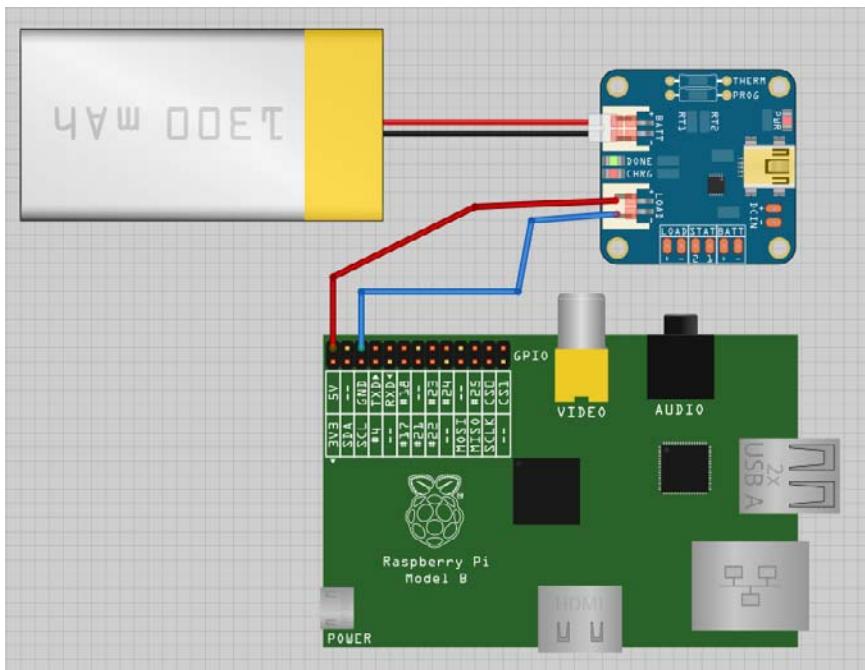
8.15. Zasilanie Raspberry Pi za pomocą akumulatora litowo-polimerowego (LiPo)

Problem

Chcesz zasilać Raspberry Pi, będące częścią robota, za pomocą akumulatora litowo-polimerowego charakteryzującego się napięciem znamionowym 3,7 V.

Rozwiązanie

Zastosuj moduł zawierający układ podnoszący napięcie (zobacz rysunek 8.10). Układ widoczny na rysunku został dostarczony przez firmę SparkFun, jednak na serwisach aukcyjnych typu eBay i Allegro można znaleźć tańsze urządzenia tego typu.



Rysunek 8.10. Zasilanie Raspberry Pi za pomocą akumulatora litowo-polimerowego (LiPo) charakteryzującego się napięciem znamionowym 3,7 V.

Tak jak w przypadku każdego okazyjnego zakupu w serwisie aukcyjnym, powinieneś dokładnie sprawdzić działanie modułu przed podłączeniem go do Raspberry Pi. Nie zawsze działają one poprawnie, a jakość ich produkcji niekiedy nie jest najwyższa.

Zaletą takiego modułu jest to, że działa on jako regulator napięcia 5 V zasilającego Raspberry Pi, a jednocześnie posiada gniazdo USB i obwód ładowania akumulatora. A zatem można do takiego modułu podłączyć standardowy zasilacz Raspberry Pi i, korzystając z niego, jednocześnie ładować baterię. Po odłączeniu przewodu zasilającego od gniazda USB moduł będzie zasilał Raspberry Pi energią zgromadzoną w akumulatorze.

Akumulator o pojemności 1300 mAh może zasilać Raspberry Pi przez około dwie godziny.

Omówienie

Jeżeli chcesz, żeby ładowaniem baterii zajął się odrewny, wyspecjalizowany układ, to rozważ zakup przetwornicy zwiększającej napięcie niewyposażonej w obwód ładowania — zaoszczędzisz trochę pieniędzy.

Zobacz również

Możesz się również pokusić o zakup gotowego modułu akumulatora LiPo wyposażonego w złącze USB. Takie moduły często charakteryzują się dużą pojemnością i mogą zasilać Raspberry Pi przez kilka godzin.

8.16. Rozpoczynanie pracy z płytą PiFace

Problem

Chcesz wiedzieć, jak należy korzystać z płytki PiFace.

Rozwiążanie

Płytkę PiFace (zobacz sekcja „Moduły” w dodatku A) jest modułem nakładanym na wierzch płytki Raspberry Pi. Zapewnia ona przydatny zestaw wejść i wyjść (zobacz rysunek 8.11).



Rysunek 8.11. Płytkę interfejsu PiFace

Do płytki PiFace dołączono oprogramowanie, które musi zostać odpowiednio skonfigurowane.

Zacznij od włączenia obsługi interfejsu SPI. W tym celu edytuj plik `/etc/modprobe.d/raspi-blacklist.conf`. Otwórz go w edytorze nano lub w dowolnym innym preferowanym przez Ciebie edytorze tekstuowym. Oznacz jako komentarz trzy linie w tym pliku. Zawartość pliku powinna wyglądać w następujący sposób:

```
# blacklist spi and i2c by default (many users don't need them)
```

```
#blacklist spi-bcm2708  
#blacklist i2c-bcm2708
```

Linia odnosząca się do magistrali I2C mogła być już wcześniej oznaczona jako komentarz.

Aby zainstalować pakiety niezbędne do obsługi modułu PiFace, w najnowszej wersji systemu Raspbian skorzystaj z następujących poleceń:

```
$ sudo apt-get update  
$ sudo apt-get install python3-pifacedigitalio python-pifacedigitalio
```

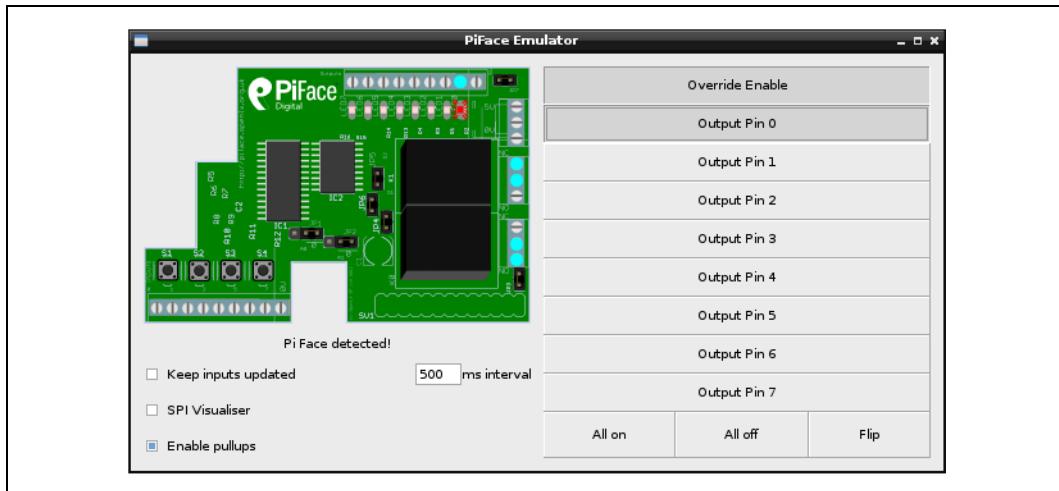
W celu zainstalowania emulatora i późniejszego stosowania go w szkicach uruchom polecenie:

```
$ sudo apt-get install python3-pifacedigital-emulator python3-pifacedigital-scratch-handler'
```

Instalacja może zająć trochę czasu. Gdy już dobiegnie końca, uruchom ponownie Raspberry Pi za pomocą polecenia:

```
$ sudo reboot
```

Najłatwiejszym sposobem testowania działania PiFace jest korzystanie z emulatora dołączonego do oprogramowania (zobacz rysunek 8.12).



Rysunek 8.12. Emulator płytki PiFace

Program ten pozwala na sterowanie tym, co dzieje się na płytce PiFace podłączonej do Raspberry Pi, oraz na monitorowanie znajdujących się na niej przycisków.

Uruchom emulator za pomocą polecenia:

```
$ /usr/bin/pifacedigital-emulator
```

Emulator posiada graficzny interfejs użytkownika, a więc możesz uruchamiać go poprzez sesję SSH (zobacz receptura 2.7). Twój Raspberry Pi potrzebuje klawiatury, myszy i monitora. Możesz również korzystać z VNC (zobacz receptura 2.8).

W otwartym oknie emulatorka kliknij przycisk *Override Enable*, a następnie *Output Pin 0*. Powinieneś usłyszeć kliknięcie — jeden z przekaźników domyka obwód. Zapali się dioda oznaczona napisem *LED0*. Klikając przycisk *Output Pin 0*, będziesz sterował pracą przekaźnika i diody, która będzie na przemian włączana i wyłączana.

Klikanie przycisków odpowiedzialnych za pracę złączy wyjściowych będzie dawało właściwe taki efekt. Klikając przyciski *Output Pin 0* i *Output Pin 1*, będziesz włączać dwie małe niebieskie kropki widoczne po prawej stronie emulatorka. Informują one o tym, który obwód przekaźnika jest aktualnie zowany.

Podobnie, gdy w emulatorku zaznaczyś opcję *Keep inputs updated*, a następnie wcisniesz któryś z przycisków znajdujących się po lewej stronie płytki PiFace, to zobaczysz, że zostanie podświetlona mała niebieska kropka znajdująca się obok zacisków wejściowych.

Emulator jest doskonałym narzędziem pozwalającym na zlokalizowanie wejść i wyjść. Umożliwia on również testowanie elektroniki podłączonej do płytki przed napisaniem programów sterujących jej pracą.

Omówienie

Emulator jest narzędziem bardzo przydatnym podczas pisania kodu obsługującego płytę PiFace — przedżej czy później będziesz chciał samodzielnie pisać kod programów.

PiFace można programować za pomocą języków Python, Scratch i C. My będziemy pracować w Pythonie. Na początku otwórz konsolę Pythona (zobacz receptura 5.3) i wpisz następujące polecenia:

```
>>> import piface.pfio as pfio  
>>> pfio.init()  
>>> pfio.digital_write(0, 1)  
>>> pfio.digital_write(0, 0)
```

Kod ten pozwala na włączanie i wyłączanie diody LED1 (i jej przekaźnika).

PiFace jest wyposażone w:

- osiem wyjść cyfrowych (otwarty kolektor),
- osiem wejść cyfrowych (3,3 V),
- osiem diod LED podłączonych do wyjść,
- cztery przełączniki podłączone do wejść o numerach od 0 do 3,
- dwa przekaźniki podłączone do wyjść o numerach 0 i 1.

W związku z tym, że PiFace posiada tylko cyfrowe wejścia i wyjścia, będą Ci potrzebne tak naprawdę jedynie dwie funkcje:

- funkcja `digital_write`, która przyjmuje dwa parametry — pierwszy z nich określa numer wyjścia, na którym ma być dokonana operacja zapisu (od 0 do 7), a drugi parametr określa sygnał podawany na wyjściu: wysoki lub niski (1 lub 0),
- funkcja `digital_read` służy do określania poziomu sygnału podawanego na wejście — funkcja ta przyjmuje jeden argument określający wejście, które ma być monitorowane, i zwraca logiczną prawdę lub fałsz (1 lub 0).

W celu wykrycia wcisnięcia przełącznika numer 1, znajdującego się po lewej stronie płytki, wypróbuj poniższy kod. Możesz go wprowadzić bezpośrednio do konsoli Pythona lub skorzystać ze środowiska IDLE (zobacz receptura 5.2):

```
>>> import piface.pfio as pfio  
>>> import time  
>>> pfio.init()  
>>> while True:  
...     print(pfio.digital_read(0))  
...     time.sleep(1)  
...  
0  
0  
1  
1  
1  
1  
0
```

Gdy przycisk będzie wciśnięty, wyświetlana będzie cyfra 1; w przeciwnym przypadku wyświetlana będzie cyfra 0.

Zobacz również

Podłączanie przekaźników do Raspberry Pi (bez użycia płytki interfejsu) opisano w recepturze 9.5.

Innym przykładem modułu jest płytka RaspiRobot (zobacz receptura 8.18). Służy ona do sterowania pracą robotów (zobacz receptura 10.8).

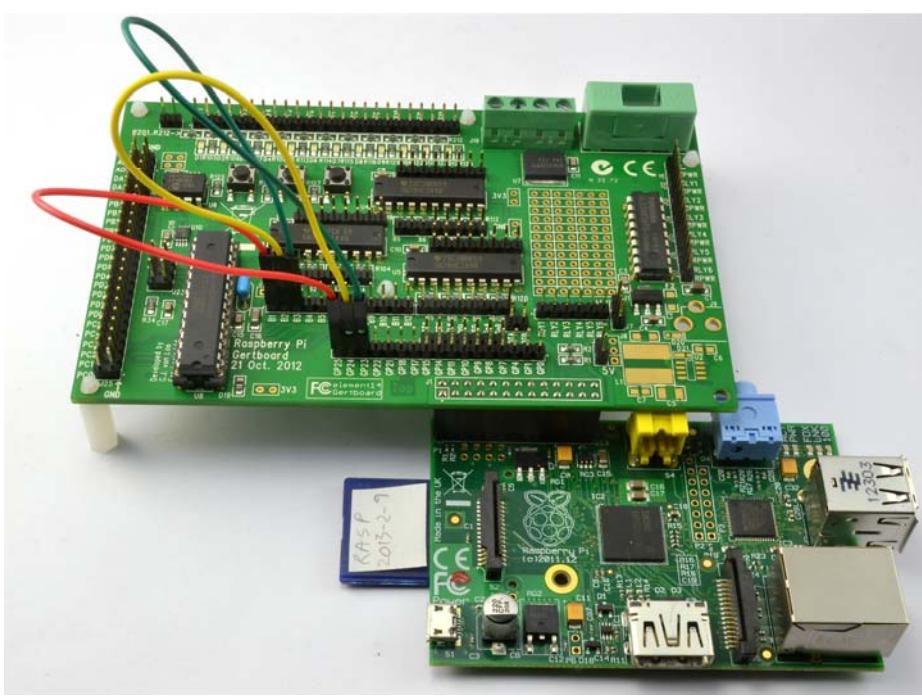
8.17. Rozpoczynanie pracy z płytka Gertboard

Problem

Chcesz rozpocząć pracę z płytka Gertboard.

Rozwiązanie

Na rysunku 8.13 pokazano płytka Gertboard podłączoną do Raspberry Pi.



Rysunek 8.13. Płytki rozszerzeń Gertboard

Płytki ta wygląda na dość skomplikowaną, ale korzystanie z niej jest naprawdę proste. Najpierw musisz zainstalować przykładowe oprogramowanie przygotowane dla tego modułu. W przykładach korzystamy zarówno z biblioteki RPi.GPIO, jak i Wiring Pi. W związku z tym, że jesteśmy bardziej obyci z biblioteką RPi.GPIO, będziemy korzystać z przykładów, w których zastosowano właśnie tę bibliotekę Pythona. Zainstaluj ją, jeżeli jeszcze tego nie zrobiłeś (zobacz receptura 8.3).

Aby zainstalować przykłady programów korzystających z płytki Gertboard, wpisz następujące polecenia w oknie Terminala:

```
$ wget http://raspi.tv/download/GB_Python.zip  
$ unzip GB_Python.zip  
$ cd GB_Python
```

Płytki Gertboard jest podzielona na kilka stref, które należy łączyć za pomocą przewodów połączeniowych. Kolejny przykład odczytuje stan trzech przycisków znajdujących się na płytce Gertboard. Program wyświetla informację za każdym razem, gdy któryś z przycisków zostanie przycisnięty lub zwolniony. Jak się wkrótce dowiesz, oprogramowanie przy pierwszym uruchomieniu poinformuje Cię o tym, które styki znajdujące się na płytce należy zewrzeć, aby przykładowy program działał prawidłowo.

Najpierw wykonaj połączenia zgodnie z wyświetlonymi instrukcjami. Pierwsze trzy połączenia należy wykonać, aby mieć możliwość odczytu stanu przycisków (styk GP25 znajdujący się w sekcji J2 należy połączyć ze stykiem B1 w sekcji J3, styk GP24 znajdujący się w sekcji J2 należy połączyć ze stykiem B2 w sekcji J3, styk GP23 znajdujący się w sekcji J2 należy połączyć ze stykiem B3 w sekcji J3). W dolnej części komunikatu podano informacje na temat ustawienia zworek, tak aby diody LED zapalały się, gdy przyciski będą wciskane. Ostatnia linia komunikatu informuje Cię o tym, że po wykonaniu wszystkich połączeń należy nacisnąć klawisz *Enter*.Więcej na temat działania płytki dowiesz się w kolejnej sekcji.

```
$ sudo python buttons-rg.py  
These are the connections for the Gertboard buttons test:  
GP25 in J2 --- B1 in J3  
GP24 in J2 --- B2 in J3  
GP23 in J2 --- B3 in J3  
 Optionally, if you want the LEDs to reflect button state do the following:  
 jumper on U3-out-B1  
 jumper on U3-out-B2  
 jumper on U3-out-B3  
 When ready hit enter.
```

```
111  
110
```

Wcisnij przyciski znajdujące się na płytce Gertboard. Gdy będziesz już miał dość eksperymentowania, wciśnij kombinację klawiszy *Ctrl+C*.

Omówienie

W przedstawionym przykładzie podłączałeś styki złącza GPIO o numerach 25, 24 i 23 do obszaru płytki Gertboard, na którym zlokalizowany jest układ buforu wraz z przełącznikami podłączonymi do jego trzech wejść. Wyjścia tego układu są połączone ze stykami złącza GPIO, które skonfigurowano tak, aby pracowały jako wejścia. Bufor zastosowany w tym układzie chroni wejścia Raspberry Pi.

Na płytce Gertboard znajduje się wiele elementów:

- 12 buforowanych linii wejścia-wyjścia,
- 3 przyciski,
- 12 diod LED,
- 6 sterowników otwartego kolektora (50 V, 0,5 A),
- sterownik silnika (18 V, 2 A),
- mikrokontroler firmy ATmega zamknięty w dwurzędowej obudowie posiadającej 28 złączy,
- dwukanałowy przetwornik cyfrowo-analogowy mogący pracować z rozdzielcością 8, 10 lub 12 bitów,
- dwukanałowy przetwornik analogowo-cyfrowy pracujący z rozdzielcością 10 bitów.

Ponadto układ ATmega 328 posiada sześć wejść analogowych i sporo własnych linii wejścia-wyjścia. Ten sam układ pełni rolę serca mikrokontrolera Arduino Uno (zobacz rozdział 14.). Możesz zaprogramować ten układ za pomocą zintegrowanego środowiska programistycznego Arduino uruchomionego na Raspberry Pi, do którego podłączono płytę Gertboard (zobacz receptura 14.15).

Zobacz również

Zapoznaj się z podręcznikiem użytkownika płytki Gertboard: http://www.automaticon.pl/novosci2013/doc/Gertboard_Assembled.pdf.

Jeżeli chcesz pracować z jakąś inną, prostszą płytą, to zajrzyj do receptury 8.16, w której opisano płytę PiFace, lub do receptury 8.18, w której opisano płytę RaspiRobot.

Wiele osób woli do obsługi elektroniki zastosować płytę mikrokontrolera Arduino i powierzyć Raspberry Pi wykonywanie czynności, w których się ono lepiej sprawdza — mowa o obsłudze ekranu i sieci. Więcej informacji na ten temat znajdziesz w rozdziale 14.

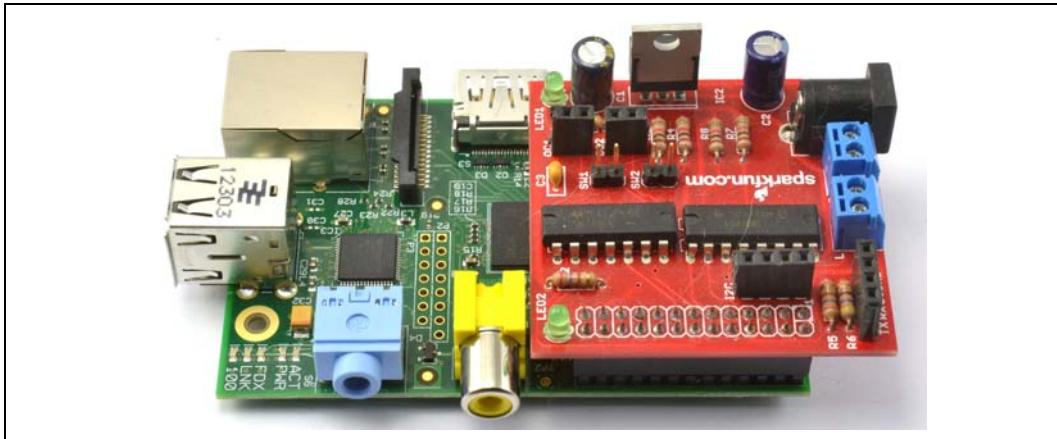
8.18. Rozpoczynanie pracy z płytą RaspiRobot

Problem

Chcesz rozpoczęć pracę z płytą RaspiRobot.

Rozwiązanie

Na rysunku 8.14 pokazano płytę RaspiRobot (wersja 1). Na płytce znajduje się podwójny sterownik silników, którym można sterować pracą dwóch silników zasilanych prądem stałym lub jednego silnika krokowego. Płyta ta, dzięki wbudowanemu regulatorowi napięcia, może również dostarczać prąd o napięciu 5 V, którym można zasilać Raspberry Pi. Moduł RaspiRobot wyposażono w dwa wejścia przełączające i dwa złącza wyjściowe o małej mocy. Moduł ten można w łatwy sposób podłączyć do magistrali I2C i interfejsu szeregowego Raspberry Pi.



Rysunek 8.14. Płytki RaspiRobot

Po podłączeniu płytki zgodnie z dołączoną instrukcją będziesz musiał zainstalować wymagane przez nią biblioteki. W oknie Terminala wpisz następujące polecenia:

```
$ sudo apt-get install python-rpi.gpio  
$ sudo apt-get install python-serial
```

Aby zainstalować dodatkowe biblioteki, uruchom kolejne polecenia w oknie Terminala:

```
$ wget https://github.com/simonmonk/raspirobotboard/archive/master.zip  
$ unzip master.zip  
$ cd raspirobotboard-master  
$ sudo python setup.py install
```

Omówienie

Zainstaluj moduł RaspiRobot na płytce Raspberry Pi. Teraz podłącz zasilanie do Raspberry Pi. Następnie możesz wypróbować wpływ pewnych poleceń wpisywanych w konsoli Pythona na płytę RaspiRobot. Na razie nie musisz podłączać zewnętrznego zasilania do modułu RaspiRobot. Nie będą Ci również potrzebne żadne silniczki. Biblioteka płytki RaspiRobot korzysta z portu GPIO, a więc musisz uruchomić Pythona jako administrator:

```
$ sudo python
```

Gdy podłączysz po raz pierwszy do zasilania płytę RaspiRobot, powinny się na niej palić obie diody LED. Wprowadzenie poniższych poleceń i zainicjowanie biblioteki powinno spowodować zgaśnięcie tych diod.

```
>>> from raspirobotboard import *  
>>> rr = RaspiRobot()
```

Jedną z diod możesz włączać i wyłączać za pomocą następujących poleceń:

```
>>> rr.set_led1(1)  
>>> rr.set_led1(0)
```

Na płytce RaspiRobot znajdują się dwie pary złączy, które są przeznaczone do podłączania przełączników. Posiadają one oznaczenia SW1 i SW2. To, czy przełącznik SW1 domyka obwód, możesz sprawdzić za pomocą polecenia:

```
>>> print(rr.sw1_closed())  
False
```

Zwarcie dwóch styków płytki RaspiRobot posiadających oznaczenie *SW1* za pomocą śrubokręta i ponowne uruchomienie powyższego polecenia spowoduje zwrócenie przez program prawdy logicznej (True).

Polecenia *set_oc1* i *set_oc2* służą do aktywowania złączy wyjściowych otwartego kolektora. Polecenia *forward*, *reverse*, *left*, *right* i *stop* służą do sterowania pracą silnika. Pełną listę poleceń znajdziesz na stronie <http://www.raspirobot.com/>.

Oczywiście z płytki RaspiRobot możesz również korzystać za pomocą biblioteki *RPi.GPIO*. W tabeli 8.1 wymieniono styki, z których korzysta wspomniana płytka.

Tabela 8.1. Styki złącza GPIO używane przez płytke RaspiRobot

Złącze płytki RaspiRobot	Złącze Raspberry Pi
Motor 1A	17
Motor 1B	4
Motor 2A	10
Motor 2B	25
SW1	11
SW2	9
OC1	22
OC2	27

Zobacz również

Więcej informacji na temat płytki RaspiRobot oraz projektów, w których można ją zastosować, znajdziesz na stronie <http://www.raspirobot.com/>.

W recepturze 10.8 przedstawiono zastosowanie tej płytka w robocie poruszającym się za pomocą kółek.

Receptura 10.7 zawiera informacje na temat zastosowania płytki RaspiRobot do sterowania bipolarnym silnikiem krokowym.

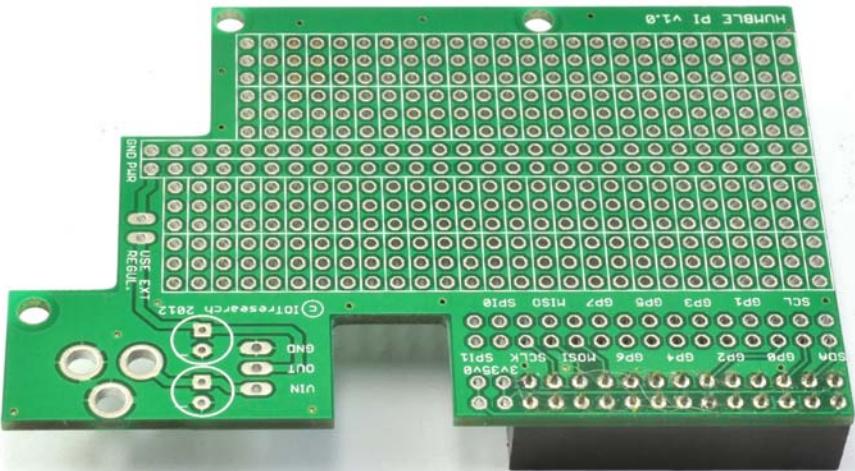
8.19. Używanie płytki prototypowej Humble Pi

Problem

Chcesz rozpoczęć pracę z płytą prototypową Humble Pi.

Rozwiążanie

Humble Pi (zobacz rysunek 8.15) jest płytą prototypową. Nie jest to płytka interfejsowa, tak jak np. płytki PiFace (zobacz receptura 8.16) i RaspiRobot (zobacz receptura 10.8). Innymi słowy, nie ma na niej żadnych podzespołów elektronicznych. Znajdują się na niej styki, do których możesz samodzielnie przylutować takie komponenty.



Rysunek 8.15. Płytki Humble Pi

Zaletą płytka Humble Pi jest to, że ma ona wyznaczone specjalne miejsce, w którym możesz zamocować gniazdo zasilania i regulator napięcia wraz z kondensatorami. Firma Ciseco — producent płytka — sprzedaje dedykowany jej regulator napięcia i zestaw kondensatorów (<http://shop.ciseco.co.uk/>).

Omówienie

Płytki składają się z siatki otworów montażowych oddalonych od siebie o 0,25 cm, co odpowiada rozstawowi nóżek większości komponentów przeznaczonych do instalacji na płytach drukowanych — w tym układów scalonych zamkniętych w obudowach podłużnych dwurzędowych. Elementy instaluje się na płytce, wsadzając je w otwory od górnej strony i przyutowując ich wyprowadzenia po dolnej stronie płytka.

Jeżeli nie wykonywałeś jeszcze nigdy połączeń lutowniczych, zapoznaj się z filmami instruktażowymi znajdującymi się w serwisie YouTube.

Otwory w białym prostokącie są połączone ze sobą ścieżkami znajdującymi się na płytce drukowanej. Na płytce są umieszczone dwie linie zasilające — jedna jest połączona z masą, a druga może zostać mostkowana z napięciem 5 lub 3,3 V generowanym przez Raspberry Pi, a także wyjściem opcjonalnego regulatora napięcia.

Po każdej stronie tych głównych linii zasilających znajdują się rzędy pogrupowanych złączy. W ten sposób po obu stronach linii zasilających stworzono dużo miejsca na układy scalone zamknięte w obudowach dwurzędowych.

Po przyutowaniu komponentów w odpowiednie miejsca będziesz musiał wykonać jeszcze połączenia pomiędzy nimi. Druty łączące mogą zostać umieszczone na górnej stronie płytka, na dolnej jej stronie lub po obu stronach (gdy układ jest złożony).

Tak czy inaczej, warto zaplanować ułożenie podzespołów na płytce jeszcze przed rozpoczęciem prac lutowniczych.

Zobacz również

Więcej informacji na temat tego produktu znajdziesz w witrynie firmy Ciseco — <http://shop.ciseco.co.uk/>.

Podobną płytę ma w swojej ofercie firma Adafruit — nosi ona nazwę Pi Plate. Więcej informacji na jej temat znajdziesz w recepturze 8.20.

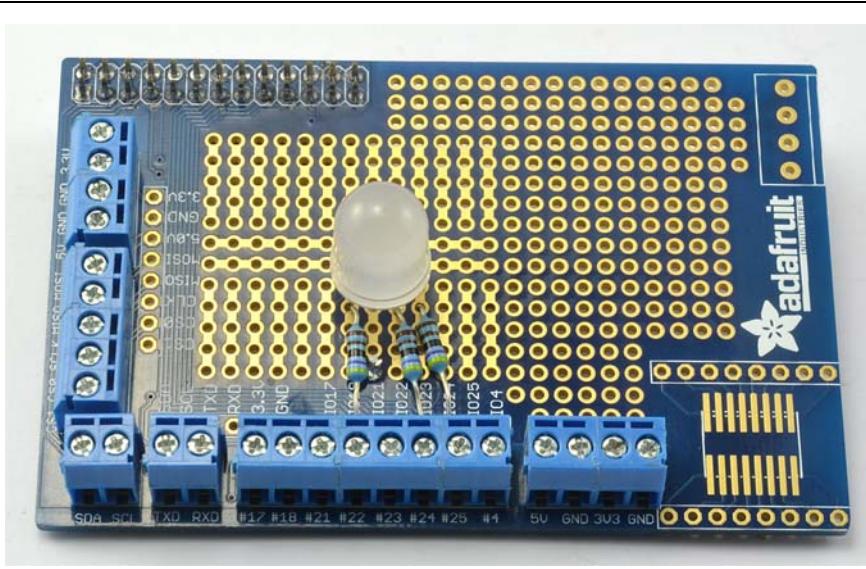
8.20. Używanie płytki prototypowej Pi Plate

Problem

Chcesz rozpoczęć pracę z płytą prototypową Pi Plate.

Rozwiązanie

Pi Plate (zobacz rysunek 8.16) jest płytą prototypową. Nie jest to płytka interfejsowa, tak jak np. płytki PiFace (zobacz receptura 8.16) i RaspiRobot (zobacz receptura 10.8). Innymi słowy, nie ma na niej żadnych podzespołów elektronicznych. Wykonano w niej otwory, w których możesz samodzielnie zainstalować takie komponenty.



Rysunek 8.16. Płyta Pi Plate

W przeciwieństwie do płytki Humble Pi (zobacz receptura 8.19), Pi Plate nie posiada nacięć na wyższe gniazda znajdujące się na Raspberry Pi. Jest ona prostokątna i ma dokładnie takie same wymiary jak płytki Raspberry Pi. Takie rozwiązanie jest możliwe ze względu na zastosowanie bardzo wysokiego gniazda złączowego, dzięki któremu zachowany jest odpowiedni dystans pomiędzy płytami Pi Plate i Raspberry Pi.

Płytki ma miejsce przewidziane do powierzchniowego montażu szesnastostykowego układu scalonego. Na module znajduje się rząd ośmiu otworów oddalonych od siebie tak, że można w nich zainstalować zaciski śrubowe.

Na dwóch bokach płytka znajdują się zaciski podłączone do wszystkich styków złącza GPIO. Możesz zignorować miejsce na płytce przeznaczone do montażu komponentów i podłączyć je bezpośrednio do styków gniazda GPIO bez wykonywania połączeń lutowniczych.

Omówienie

Płytki składa się z siatki otworów montażowych oddalonych od siebie o 0,25 cm, co odpowiada rozstawowi nóżek większości komponentów przeznaczonych do instalacji na płytach drukowanych — w tym układów scalonych zamkniętych w obudowach podłużnych dwurzędowych. Elementy instaluje się na płytce, wsadzając je w otwory od górnej strony i przyutowując ich wyprowadzenia po dolnej stronie płytki.

Ścieżki łączące otwory są wyraźnie widoczne na górnej stronie płytki, która jest podzielona na kilka stref: miejsce z głównymi liniami zasilającymi przeznaczone do montażu układów scalonych; miejsce przeznaczone na większość elementów prototypowego obwodu; miejsce na układ montowany powierzchniowo; przestrzeń na dodatkowe zaciski.

Po przylutowaniu komponentów do płytka będziesz musiał jeszcze wykonać połączenia pomiędzy nimi. Możesz je wykonać po dowolnej stronie płytka, a w przypadku bardziej złożonych obwodów druty połączeniowe znajdą się po obu stronach płytka.

Warto przemyśleć ułożenie podzespołów na płytce jeszcze przed ich przylutowaniem.

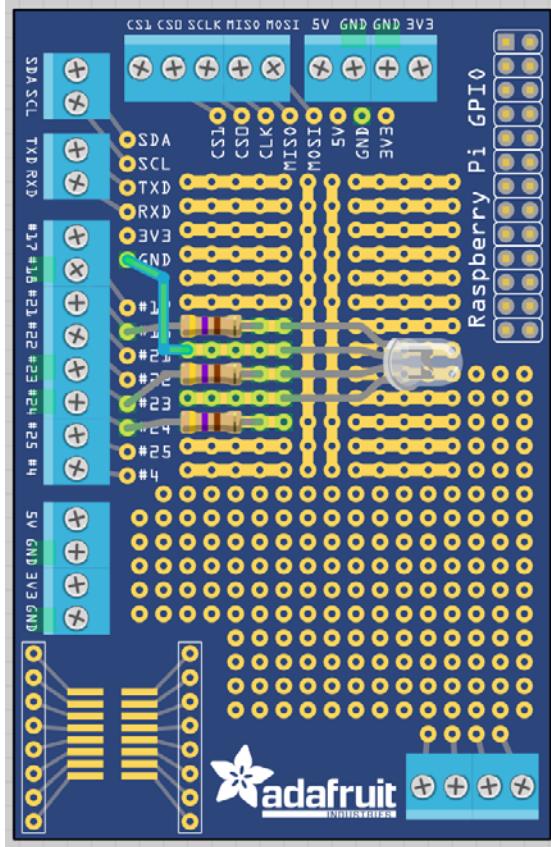
Teraz zajmiemy się przykładem przedstawiającym proces podłączania diody LED RGB do płytka Pi Plate. Obwód ten przedstawiono na rysunku 8.17. Podobny obwód zaprezentowano w recepturze 9.9, z tą różnicą, że obwód omawiany tutaj nie jest umieszczony na płytce prototypowej i wymaga wykonania połączeń lutowniczych.

Pracę zacznij od przylutowania rezystorów. Zegnij wystające z rezystora druciki i wepchnij je we właściwe otwory płytki. Teraz odwróć płytkę i przez około sekundę dotykaj grotem lutownicy miejsce, w którym druciki wystają z otworów. Dopiero po chwili rozprowadź w tym miejscu cynę, która powinna się rozplynąć wokół drucika (zobacz rysunek 8.18).

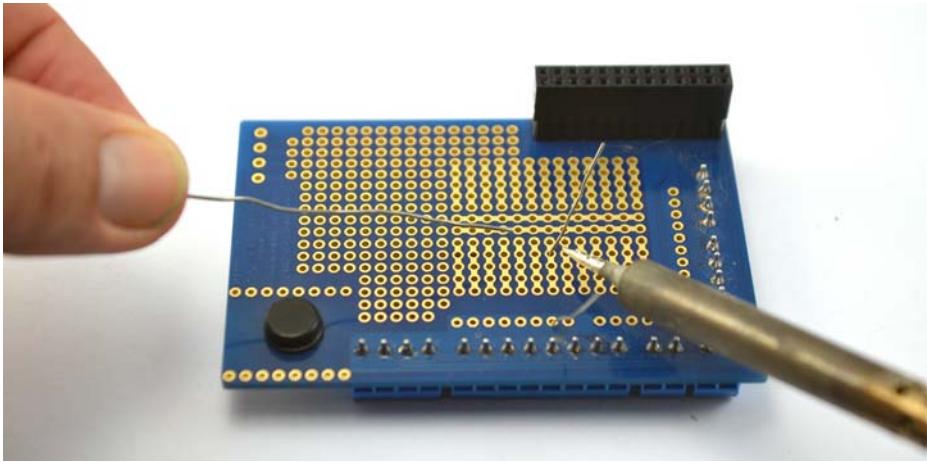
Po przylutowaniu obu końcówek rezystora odetnij zbędne druciki wystające z płytka i powtórz opisane wcześniej czynności dla dwóch kolejnych rezystorów (zobacz rysunek 8.19).

Teraz przylutuj w odpowiednie miejsce diodę LED. Uważaj, żeby nie przylutować jej odwrotnie. Najdłuższa nóżka diody jest wspólną katodą — powinna być ona jedynym wyprowadzeniem diody podłączonym bezpośrednio do ścieżki, do której nie podłączono żadnego rezystora. Bardzo rzadko spotyka się diody, których dłuższe wyprowadzenie jest elektrodą dodatnią (w ten sposób wykonywane są diody LED generujące promieniowanie podczerwone). Jeżeli nie jesteś pewien polaryzacji swojej diody, zajrzyj do jej dokumentacji lub przejdź na witrynę internetową jej producenta.

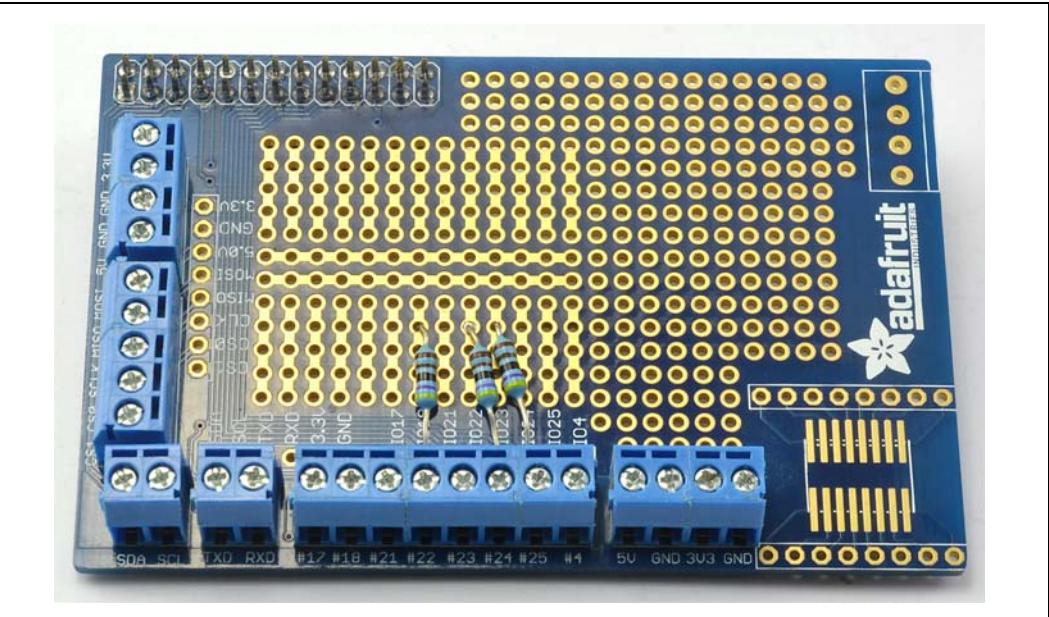
Wspomniany wcześniej punkt będziesz musiał połączyć z masą za pomocą krótkiego kawałka drutu (zobacz rysunek 8.20).



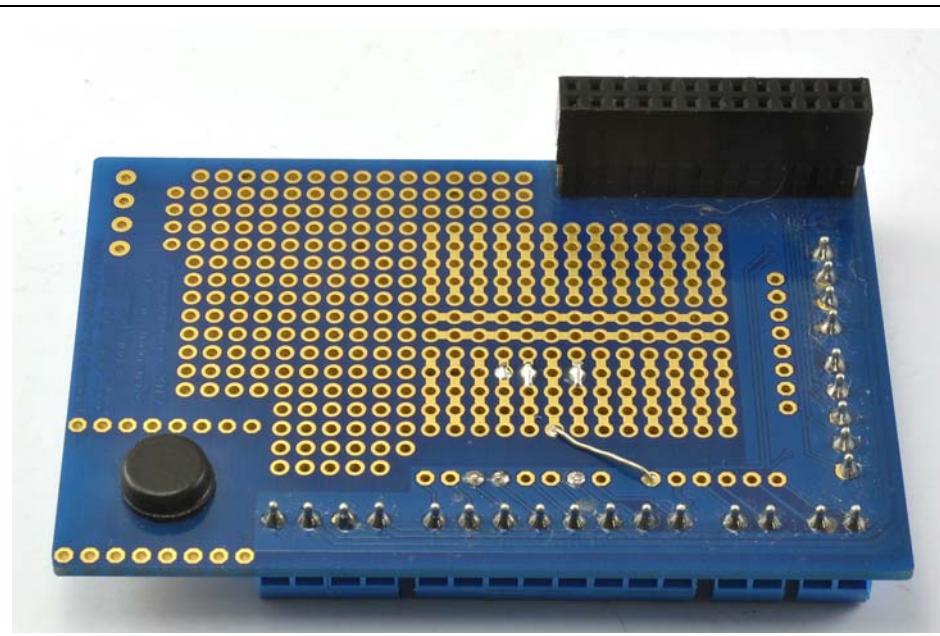
Rysunek 8.17. Schemat wykonawczy obwodu diody LED RGB



Rysunek 8.18. Przyłutowywanie rezystora do płytki Pi Plate

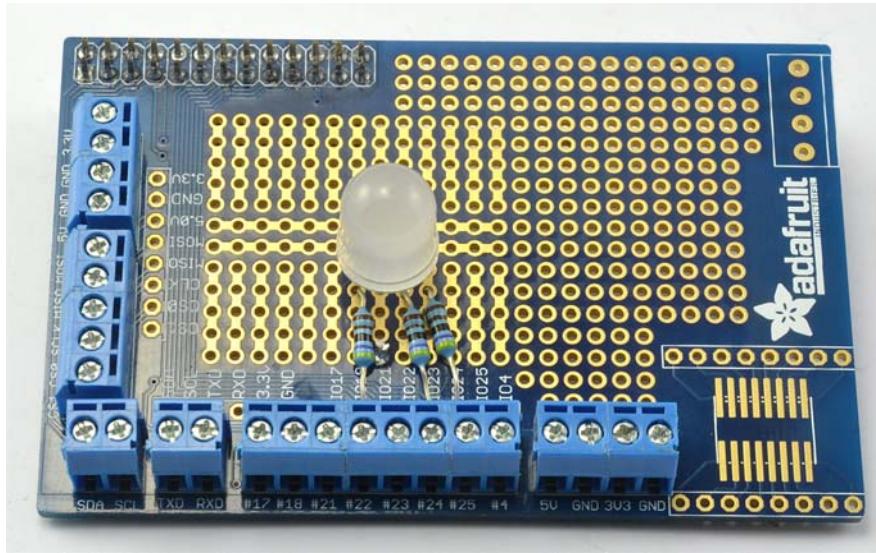


Rysunek 8.19. Rezystory przylutowane do płytki Pi Plate



Rysunek 8.20. Lutowanie przewodu łączącego katodę diody z masq

Gotową płytke pokazano na rysunku 8.21.



Rysunek 8.21. Płytki Pi Plate z zainstalowanym na niej obwodem diody LED RGB

Działanie diody możesz sprawdzić za pomocą aplikacji Pythona omówionej w recepturze 9.9.

Zobacz również

Więcej informacji na temat tego produktu znajdziesz w witrynie internetowej firmy Adafruit — <http://www.adafruit.com/products/801>.

Płytki Pi Plate firmy Adafruit jest bardzo podobna do płytki Humble Pi przedstawionej w recepturze 8.19. Montaż elementów elektronicznych na obu płytach przebiega w ten sam sposób.

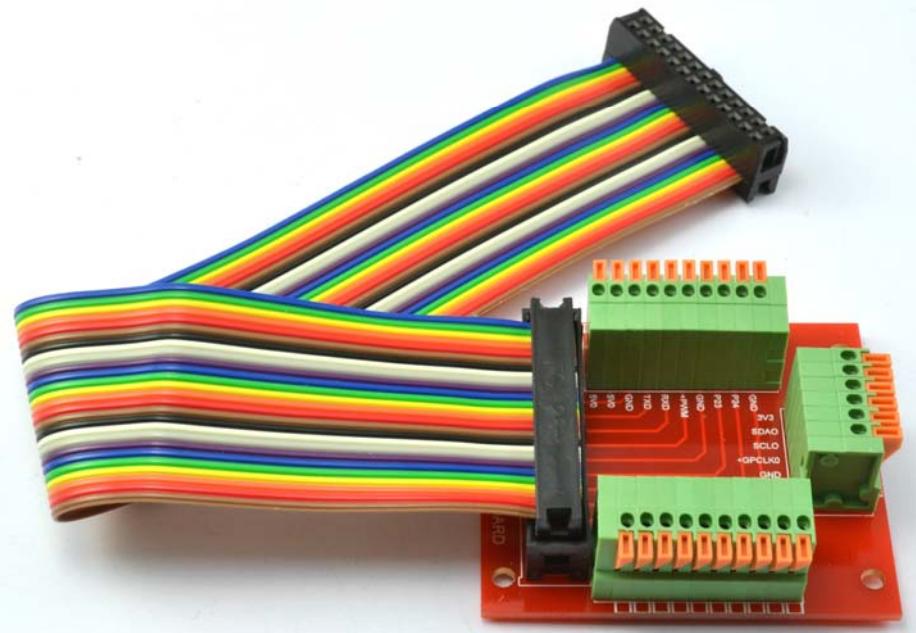
8.21. Podłączanie płytki drukowanej z zaciskami sprężynowymi

Problem

Chcesz podłączyć do złącza GPIO płytę drukowaną z zaciskami.

Rozwiążanie

Możesz skorzystać z płytki z zaciskami pokazanej na rysunku 8.22. Pozwala ona na szybkie podłączanie komponentów oraz przewodów do Raspberry Pi. Korzystając z takiej płytki, nie musisz wykonywać połączeń lutowniczych. Wystarczy, że odciagneszt zacisk w dół, wsadzisz w otwór przewód lub wyprowadzenie komponentu, a następnie puścisz zacisk, unieruchamiając przewodnik wsadzony w otwór.



Rysunek 8.22. Płytki drukowane z zaciskami sprężynowymi

Omówienie

Tego typu płytki przydają się od czasu do czasu w celu szybkiego połączenia układu.

Zobacz również

Możesz również skorzystać z możliwości oferowanych przez zaciski śrubowe znajdujące się na płytce Pi Plate (zobacz receptura 8.20).

Sterowanie sprzętem elektronicznym

9.0. Wprowadzenie

W tym rozdziale zostanie poruszona problematyka sterowania elektroniką za pośrednictwem złącza GPIO znajdującego się na płytce Raspberry Pi.

W większości receptur korzystam z płytki prototypowej niewymagającej wykonywania połączeń lutowniczych oraz z dwóch rodzajów przewodów połączeniowych. W przypadku pierwszych jedne końce przewodów zostały wyposażone w końcówkę męską, a drugie w żeńską, zaś drugie są obustronnie wyposażone w końcówki męskie (zobacz receptura 8.10).

9.1. Podłączanie diody LED

Problem

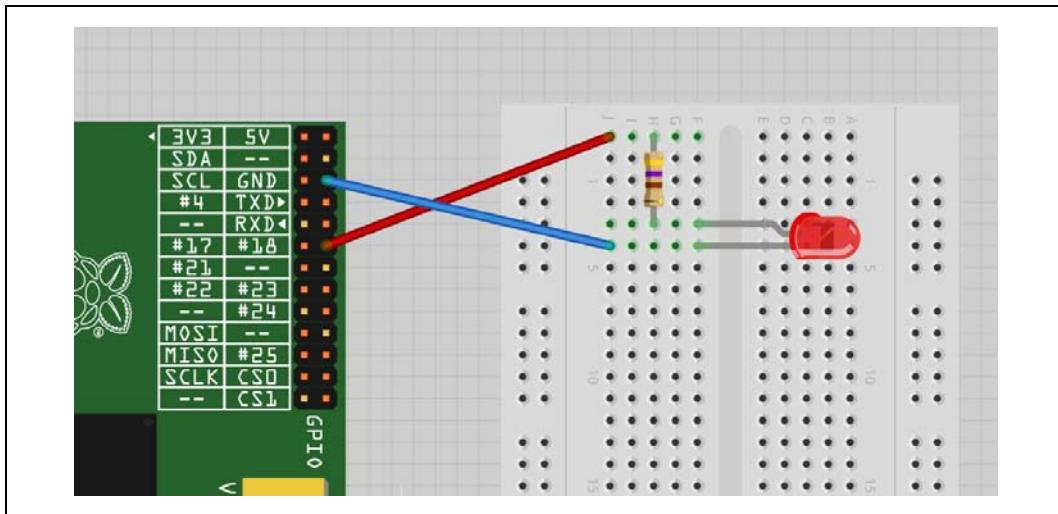
Chcesz się dowiedzieć, w jaki sposób możesz podłączyć diodę LED do swojego Raspberry Pi.

Rozwiążanie

Do jednego ze styków gniazda GPIO podłącz szeregowo diodę LED (zobacz sekcja „Optoelektronika” w dodatku A) z rezystorem charakteryzującym się oporem $470\ \Omega$ lub $1\ k\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A). Rezystor ma ograniczać prąd płynący przez diodę. Będą Ci potrzebne:

- płytka prototypowa i przewody połączeniowe (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezistor charakteryzujący się oporem $1\ k\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- dioda LED (zobacz sekcja „Optoelektronika” w dodatku A).

Na rysunku 9.1 przedstawiono diodę podłączoną do płytka Raspberry Pi za pośrednictwem płytka prototypowej niewymagającej wykonywania połączeń lutowniczych oraz przy użyciu dwóch przewodów połączeniowych.



Rysunek 9.1. Dioda LED podłączona do płytki Raspberry Pi

Po podłączeniu diody można ją włączać i wyłączać za pomocą poleceń Pythona. W tym celu musisz zainstalować bibliotekę RPi.GPIO (zobacz receptura 8.3).

Korzystając z sesji Terminala, uruchom jako administrator konsolę Pythona (zobacz receptura 5.3) i wprowadź następujące polecenia:

```
$ sudo python
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(18, GPIO.OUT)
>>> GPIO.output(18, True)
>>> GPIO.output(18, False)
```

Powyższy kod będzie włączał i wyłączał diodę LED.

Omówienie

Diody LED są tanim i wydajnym źródłem światła, mającym szerokie spektrum zastosowań, jednak korzystając z nich, musisz zachować ostrożność. Diody te po podłączeniu bezpośrednio do źródła napięcia większego od około 1,7 V (takiego jak złącze GPIO) będą pobierać prąd o bardzo dużym natężeniu. Natężenie to jest na tyle duże, że uszkodzeniu może ulec dioda LED lub układ ją zasilający (w naszym przypadku jest to Raspberry Pi).

Diodę LED należy zawsze łączyć szeregowo z rezystorem umieszczonym pomiędzy diodą a źródłem napięcia. Ogranicza on natężenie prądu płynącego przez diodę LED do poziomu, który jest bezpieczny zarówno dla diody, jak i złącza GPIO.

Styki złącza GPIO mogą dostarczyć prąd o natężeniu 3 mA. Diody LED będą świecić, gdy do staczy się im prąd o natężeniu większym od 1 mA. Im większe natężenie prądu płynącego przez diodę, tym jaśniej będzie ona świecić. Dobierając rezystor do diody, kieruj się wytycznymi znajdującymi się w tabeli 9.1. W tabeli tej dodatkowo podano natężenie prądu pobieranego ze złącza GPIO.

Tabela 9.1. Dobieranie właściwego rezystora dla diody LED podłączonej do złącza GPIO dostarczającego prąd o napięciu 3,3 V

Typ diody LED	Rezystor	Natężenie prądu (mA)
czerwona	470 Ω	3,5
czerwona	1 kΩ	1,5
pomarańczowa, żółta, zielona	470 Ω	2
pomarańczowa, żółta, zielona	1 kΩ	1
niebieska, biała	100 Ω	3
niebieska, biała	270 Ω	1

Jak widzisz, we wszystkich przypadkach bezpiecznie można stosować rezystory charakteryzujące się oporem 470 Ω. W przypadku niebieskich i białych diod LED możesz korzystać z rezystorów o niższym oporze. Tak czy inaczej, jeżeli chcesz zachować ostrożność, stosuj rezystory 1 kΩ.

Jeżeli po przeprowadzeniu eksperymentu w konsoli Pythona chcesz napisać program na przemiennie włączający i wyłączający diodę LED, wpisz poniższy kod w okno IDLE (zobacz receptura 5.2) lub do edytora nano (zobacz receptura 3.6) i zapisz plik pod nazwą *led_blink.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

while (True):
    GPIO.output(18, True)
    time.sleep(0.5)
    GPIO.output(18, False)
    time.sleep(0.5)
```

Pamiętaj, że program ten musisz uruchamiać jako użytkownik posiadający uprawnienia administratora (wymaga tego biblioteka *RPi.GPIO*). Zapisany plik uruchom za pomocą polecenia:

```
$ sudo python led_blink.py
```

Zobacz również

Na stronie internetowej <http://led.linear1.org/1led.wiz> znajduje się kalkulator przeznaczony do obliczania wartości oporu rezystora połączonego szeregowo z diodą LED.

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

9.2. Regulacja jasności diody LED

Problem

Chcesz zmieniać jasność, z jaką świeci dioda, za pomocą programu napisanego w Pythonie.

Rozwiążanie

Biblioteka RPi.GPIO obsługuje modulację czasu trwania impulsu — układ PWM, który pozwala na sterowanie mocą dostarczaną do diody LED, a więc jej jasnością.

Wypróbuj działanie układu PWM. Podłącz diodę LED tak, jak zostało to opisane w recepturze 9.1, i uruchom poniższy program.

```
import RPi.GPIO as GPIO

led_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(led_pin, GPIO.OUT)

pwm_led = GPIO.PWM(led_pin, 500)
pwm_led.start(100)

while True:
    duty_s = raw_input("Podaj jasność (liczba od 0 do 100):")
    duty = int(duty_s)
    pwm_led.ChangeDutyCycle(duty)
```

Jeżeli korzystasz z Pythona 3, a nie 2, zmień polecenie `raw_input` na `input`.

Uruchom ten program i steruj jasnością diody LED, wprowadzając liczby od 0 do 100:

```
pi@raspberrypi ~ $ sudo python led_brightness.py
Podaj jasność (liczba od 0 do 100):0
Podaj jasność (liczba od 0 do 100):20
Podaj jasność (liczba od 0 do 100):10
Podaj jasność (liczba od 0 do 100):5
Podaj jasność (liczba od 0 do 100):1
Podaj jasność (liczba od 0 do 100):90
```

Aby zakończyć działanie programu, wciśnij kombinację klawiszy `Ctrl+C`.

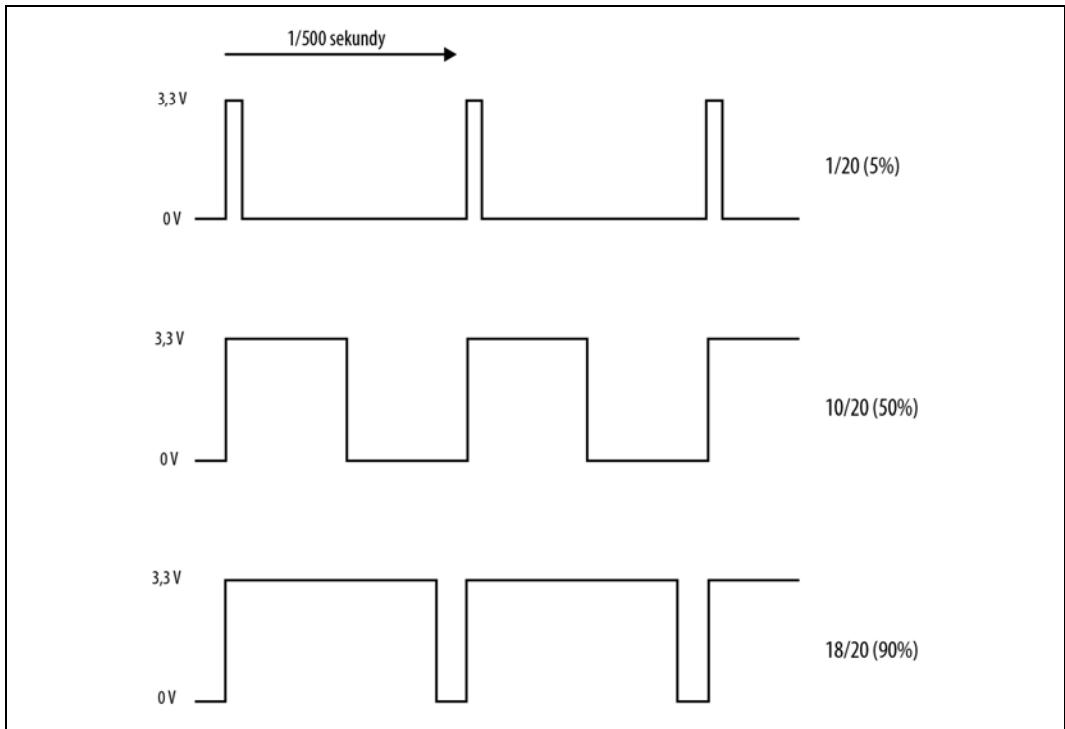
Omówienie

Modulacja czasu trwania impulsu jest sprytną techniką, w której zmianie ulega tylko czas trwania impulsu, a liczba impulsów w jednostce czasu (częstotliwość) już nie. Zasadę działania układu PWM pokazano na rysunku 9.2.

Przy wysokich częstotliwościach zmierzona częstotliwość modulacji może odbiegać od wartości zadanej. Być może ulegnie to zmianie w nowszych wersjach biblioteki RPi.GPIO obsługującej tę technologię.

Częstotliwość modulacji czasu trwania impulsu możesz zmienić, modyfikując poniższe polecenie:

```
pwm_led = GPIO.PWM(led_pin, 500)
```



Rysunek 9.2. Modulacja czasu trwania impulsu

W roli argumentu funkcji należy podać częstotliwość wyrażoną w hercach. W podanym przykładzie częstotliwość wynosi 500 Hz.

W tabeli 9.2 zestawiono ze sobą częstotliwości podawane jako drugi parametr polecenia GPIO.PWM i te uzyskane w wyniku pomiaru oscyloskopem sygnału generowanego na złączu GPIO.

Tabela 9.2. Zestawienie częstotliwości zadanych i częstotliwości faktycznie generowanych przez urządzenie

Częstotliwość zadana	Częstotliwość zmierzona
50 Hz	50 Hz
100 Hz	98,7 Hz
200 Hz	195 Hz
500 Hz	470 Hz
1 kHz	890 Hz
10 kHz	4,4 kHz

Odkryłem również, że wraz ze wzrostem częstotliwości spada jej stabilność. Oznacza to, że układ PWM nie sprawdzi się w technice audio, ale świetnie nadaje się on do sterowania jasnością diody LED lub prędkością obrotową silnika.

Zobacz również

Więcej informacji na temat modulacji czasu trwania impulsu znajdziesz w Wikipedii — https://pl.wikipedia.org/wiki/Modulacja_szeroko%C5%9Bci_impu%C5%82u.

W recepturze 9.9 układ modulacji czasu trwania impulsu zastosowano do zmiany koloru diody RGB LED, a w recepturze 10.3 — do sterowania prędkością obrotową silnika zasilanego prądem stałym.

Więcej informacji na temat korzystania z płyt prototypowych i przewodów połączeniowych znajdziesz w recepturze 8.10. Jasność diody możesz regulować suwakiem — zobacz recepturę 9.8.

9.3. Generowanie brzęczącego dźwięku

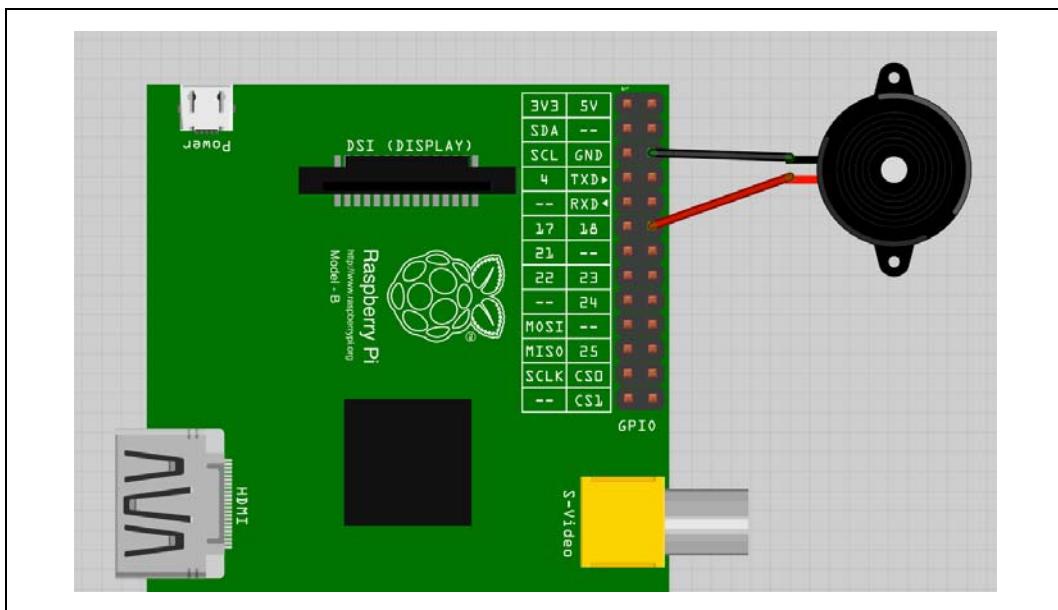
Problem

Chcesz sterować pracą brzęczyka za pomocą Raspberry Pi.

Rozwiązanie

Do złącza GPIO podłącz piezoelektryczny brzęczyk.

Większość małych brzęczyków piezoelektrycznych działa poprawnie po podłączeniu do płytki Raspberry Pi w sposób pokazany na rysunku 9.3. Zastosowałem brzęczyk firmy Adafruit (zobacz sekcja „Pozostałe” w dodatku A). Brzęczyk możesz połączyć bezpośrednio ze stykami Raspberry Pi za pomocą przewodów połączeniowych zakończonych obustronnie końcówkami męskimi (zobacz sekcję „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A).



Rysunek 9.3. Brzęczyk piezoelektryczny podłączony do Raspberry Pi

Brzęczyki pobierają prąd o bardzo małym natężeniu. Jeśli jednak podłączasz dość duży brzęczyk lub po prostu chcesz zachować ostrożność, to pomiędzy stykiem złącza GPIO a przewodem biegnącym do brzęczyka podłącz szeregowo rezystor 470 Ω.

Umieść poniższy kod w oknie środowiska programistycznego IDLE (receptura 5.2) lub w edytorze tekstowym nano (receptura 3.6) i zapisz utworzony plik pod nazwą *buzzer.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

buzzer_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

def buzz(pitch, duration):
    period = 1.0 / pitch
    delay = period / 2
    cycles = int(duration * pitch)
    for i in range(cycles):
        GPIO.output(buzzer_pin, True)
        time.sleep(delay)
        GPIO.output(buzzer_pin, False)
        time.sleep(delay)

while True:
    pitch_s = raw_input("Podaj częstotliwość (od 200 do 2000): ")
    pitch = float(pitch_s)
    duration_s = raw_input("Podaj czas trwania (w sekundach): ")
    duration = float(duration_s)
    buzz(pitch, duration)
```

Gdy uruchomisz ten program, najpierw zostaniesz poproszony o podanie częstotliwości wyrażonej w hercach, a następnie o czas generowania dźwięku w sekundach:

```
$ sudo python buzzer.py
Podaj częstotliwość (od 200 do 2000): 2000
Podaj czas trwania (w sekundach): 20
```

Omówienie

Brzęczyki piezoelektryczne charakteryzują się wąskim zakresem częstotliwości generowanego dźwięku, a także jego niską jakością. Pozwalają jednak na generowanie tonów w pewnym zakresie częstotliwości. Częstotliwość generowana przez program będzie miała wartość zblizzoną do podanej przez użytkownika.

Działanie programu w praktyce polega na włączaniu i wyłączaniu osiemnastego pinu złącza GPIO. Pomiędzy tymi czynnościami upływa czas zależny od częstotliwości generowanego dźwięku. Im wyższy dźwięk (wyższa częstotliwość), tym krótszy może być ten czas.

Zobacz również

Specyfikację brzęczyka piezoelektrycznego znajdziesz w tym dokumencie: http://www.tdk.co.jp/tefe02/ef532_ps.pdf.

9.4. Sterowanie pracą urządzenia o dużej mocy zasilanego prądem stałym za pośrednictwem tranzystora

Problem

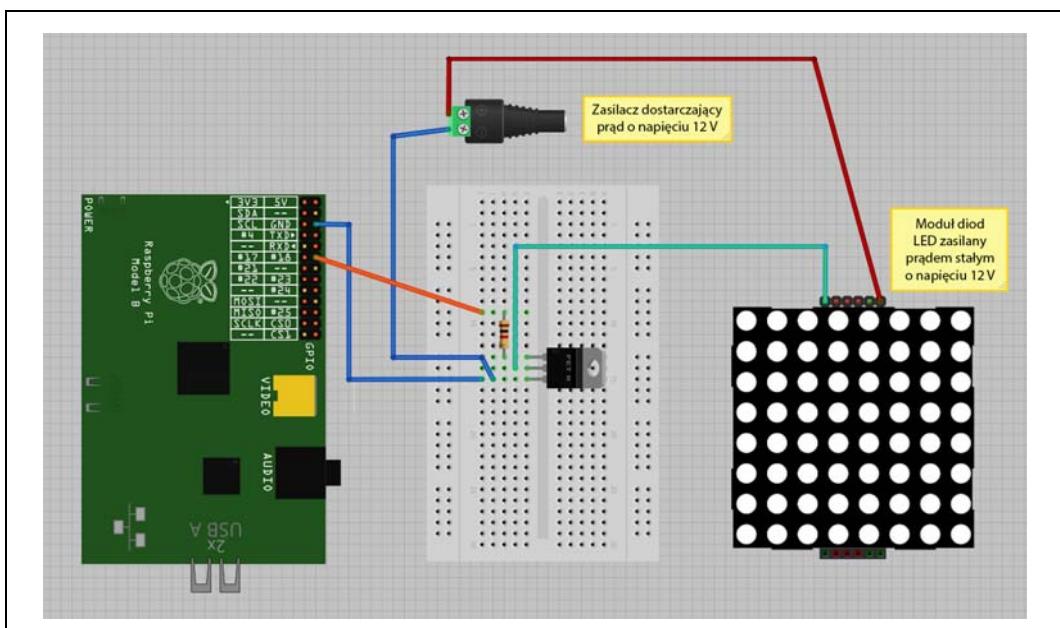
Chcesz sterować mocą urządzeniami o dużej mocy zasilanego prądem stałym o niskim napięciu, np. modułem diod LED zasilanym prądem o napięciu 12 V.

Rozwiązanie

Diody LED o dużej mocy pobierają prąd o natężeniu przekraczającym możliwości złącza GPIO. Ponadto zwykle są one zasilane prądem o napięciu 12 V, a Raspberry Pi jest w stanie dostarczyć prąd o napięciu zaledwie 3,3 V. Sterowanie mocą takiej diody musi przebiegać za pośrednictwem tranzystora.

Tym razem będziemy korzystali z tranzystora wysokiej mocy typu MOSFET (tranzystora polowego typu metal-tlenek-półprzewodnik). Kosztuje on mniej niż 5 zł, a jest w stanie sterować pracą komponentu zasilanego prądem o natężeniu 30 A, co wielokrotnie przekracza natężenie prądu pobieranego przez diody LED. Korzystamy z tranzystora FQP30N06 (zobacz sekcję „Tranzystory i diody” w dodatku A).

Na rysunku 9.4 pokazano sposób montażu tranzystora na płytce prototypowej. Upewnij się, że właściwie zidentyfikowałeś anodę oraz katodę modułu diody.



Rysunek 9.4. Sterowanie przepływem prądu o dużym natężeniu za pośrednictwem tranzystora typu MOSFET

Aby zbudować ten obwód, będziesz potrzebować:

- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- układu FQP30N06 — MOSFET z kanałem typu n (zobacz sekcja „Tranzystory i diody” w dodatku A),
- zasilacza dostarczającego prąd o napięciu 12 V,
- modułu diod LED zasilanego prądem stałym o napięciu 12 V.

Do włączania i wyłączania zestawu diod LED zastosuj dokładnie ten sam kod, który został wykorzystany w recepturze 9.2 do sterowania pracą pojedynczej diody LED bez tranzystora.

Jasnością modułu zasilanego za pośrednictwem tranzystora typu MOSFET możesz sterować, modulując czas trwania impulsu (zobacz receptura 9.2).

Omówienie

Urządzenia o większej mocy podłączane do złącza GPIO należy zasilać bateriami lub zewnętrznym zasilaczem. Złącze GPIO jest w stanie dostarczyć prąd o niskim natężeniu (zobacz receptura 8.2). W omawianym projekcie panel z diodami LED będziesz zasilać zasilaczem dostarczającym prąd stały o napięciu 12 V. Wybierz zasilacz, który jest w stanie dostarczyć wystarczająco dużo mocy. Jeżeli moduł diod ma moc 5 W, to Twój zasilacz powinien się charakteryzować przynajmniej taką mocą wyjściową (zasilacz o mocy 6 W byłby jeszcze lepszy). Jeżeli na zasilaczu nie określono jego mocy, a podano maksymalne natężenie prądu, jakie jest on w stanie dostarczyć, to moc zasilacza możesz obliczyć, mnożąc maksymalne natężenie prądu przez napięcie. A zatem zasilacz dostarczający prąd o napięciu 12 V i natężeniu 500 mA może zasilać urządzenia o mocy 6 W.

Zastosowanie w obwodzie rezystora jest konieczne w celu zapobiegnięcia uszkodzeniu złącza GPIO przez prądy szczytowe pojawiające się podczas włączania i wyłączania tranzystora typu MOSFET. Tranzystor steruje ujemnym biegunem panelu z diodami. Do dodatniego biegunu panelu podłączono bezpośrednio dodatni przewód zasilacza, a dodatni biegun panelu podłączono do *drenu* tranzystora. Źródło tranzystora połączono z masą, a jego *bramkę* ze stykiem sterującym przepływem prądu z drenu do źródła. Jeżeli napięcie na bramce przekroczy 2 V, to MOSFET się włączy — prąd popłynie zarówno przez tranzystor, jak i przez moduł diod LED.

Obwód ten można stosować również do sterowania mocą innych urządzeń zasilanych prądem stałym o niskim napięciu. Wyjątkiem są silniki i przekaźniki, którymi steruje się w inny sposób (zobacz receptura 10.3).

Zobacz również

Zapoznaj się z dokumentacją tranzystora typu MOSFET — <http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General/FQP30N06L.pdf>.

Jeżeli chcesz sterować modułem z diodami LED za pomocą interfejsu graficznego, zajrzyj do receptury 9.7 — zaprezentowano tam program włączający i wyłączający diody LED, a w recepturze 9.8 przedstawiono program regulujący jasność modułu za pomocą suwaka.

9.5. Włączanie urządzeń o dużej mocy za pomocą przekaźnika

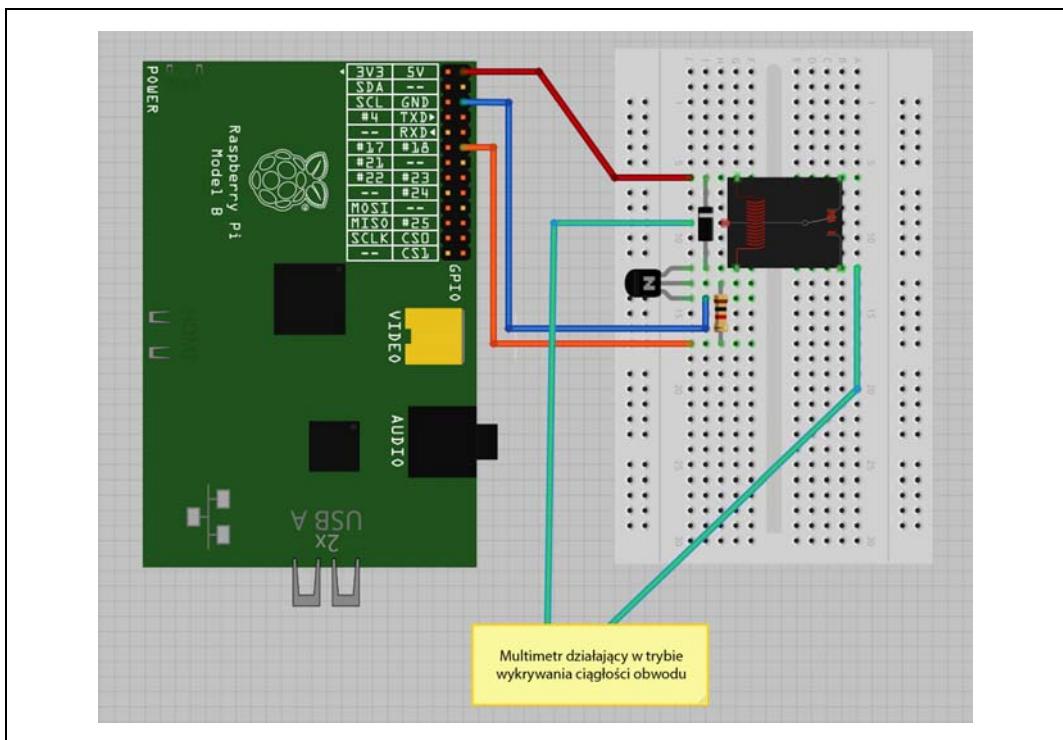
Problem

Chcesz włączać i wyłączać urządzenia, które nie mogą być sterowane za pośrednictwem tranzystora typu MOSFET.

Rozwiązanie

Zastosuj przekaźnik i mały tranzystor.

Na rysunku 9.5 pokazano obwód składający się z przekaźnika i tranzystora, który został umieszczony na płytce prototypowej. Upewnij się, że tranzystor i dioda nie są podłączone odwrotnie. Jeden z końców diody został oznaczony paskiem. W zastosowanym tutaj tranzystorze jedno wyprowadzenie jest proste, a drugie zagięte.



Rysunek 9.5. Przekaźnik podłączony do Raspberry Pi

Aby zbudować ten obwód, będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),

- tranzystora 2N3904 (zobacz sekcja „Tranzystory i diody” w dodatku A),
- diody 1N4001 (zobacz sekcja „Tranzystory i diody” w dodatku A),
- przekaźnika 5 V (zobacz sekcja „Pozostałe” w dodatku A),
- multimetru.

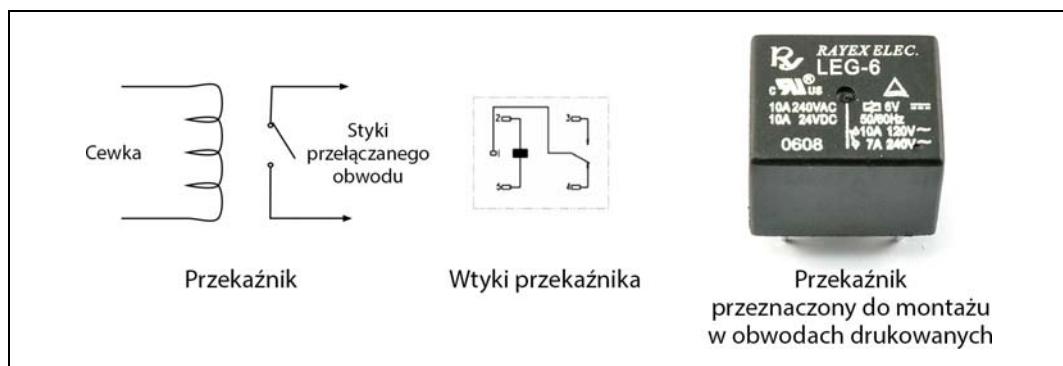
Możesz skorzystać z programu włączającego i wyłączającego diodę, omówionego w recepturze 9.1. Jeżeli wszystko działa poprawnie, powinieneś słyszeć charakterystyczne klikanie przekaźnika domykającego obwód. Przekaźniki są urządzeniami mechanicznymi działającymi dość wolno, a więc nie próbuj sterować ich pracą za pomocą układu PWM (modulacji czasu trwania impulsu). Mogłoby to doprowadzić do uszkodzenia przekaźnika.

Omówienie

Przekaźniki są obecne w elektronice niemalże od jej początków. Są one łatwe w użyciu i można je stosować wszędzie tam, gdzie można użyć włącznika. Z przekaźników można korzystać do włączania obwodów prądu przemiennego lub urządzeń, których budowy w pełni nie znamy.

Jeżeli przekaźnik zastosujesz w obwodzie, który przekracza jego specyfikację, to pomiędzy stykami przekaźnika będzie dochodziło do iskrzenia, wskutek czego mogą one ulec zespawaniu. Przekaźnik może się również przegrzać. Jeżeli masz wątpliwości dotyczące wyboru przekaźnika, decyduj się zawsze na taki model, którego specyfikacja przewyższa potrzeby danego obwodu.

Na rysunku 9.6 przedstawiono budowę przekaźnika.



Rysunek 9.6. Budowa przekaźnika

Przekaźnik to tak naprawdę przełącznik, którego styki zwierają się, gdy przyciągnie je do siebie elektromagnes. Elektromagnes i przełącznik nie są ze sobą połączone elektrycznie, a więc obwód sterujący pracą przekaźnika jest chroniony przed wysokimi napięciami występującymi w załączanym obwodzie.

Wadą przekaźników jest to, że pracują stosunkowo wolno i ulegają zużyciu po setkach tysięcy wykonanych operacji domknięcia obwodu. Można je więc stosować tylko do włączania i wyłączania obwodu. Nie można za ich pośrednictwem sterować mocą obwodu przy użyciu układu PWM (modulacji czasu trwania impulsu).

Cewka przekaźnika, domykając obwód, pobiera prąd o natężeniu 50 mA. Złącze GPIO jest w stanie dostarczyć prąd o natężeniu zaledwie 3 mA, a więc musisz skorzystać z małego tranzystora pełniącego rolę włącznika. Nie musisz tutaj stosować MOSFET-u o dużej mocy, jak to było w recepturze 9.4. W zupełności wystarczy mały tranzystor. Element ten posiada trzy złącza: *bazę* (złącze środkowe) podłączoną za pośrednictwem rezystora 1 k Ω ograniczającego prąd do złącza GPIO, *emiter* podłączony do masy oraz *kolektor* podłączony do przekaźnika. Druga strona przekaźnika jest podłączona do szyny 5 V złącza GPIO. Diodę zastosowano w celu tłumienia impulsów wysokiego napięcia powstających w wyniku załączania cewki przekaźnika przez tranzystor.



Przekaźniki mogą być używane do załączania obwodów prądu przemiennego zasilanych z sieci energetycznej o napięciu 230 V. Prąd o takim napięciu jest niebezpieczny. Obwodu o takim napięciu nie należy wykonywać na płytce prototypowej. Jeżeli chceszłączyć obwody wysokiego napięcia, zapoznaj się z treścią receptury 9.6.

Zobacz również

W recepturze 9.4 opisano układ przełączający zbudowany w oparciu o tranzystor mocy typu MOSFET.

Pracą przekaźnika można sterować również za pośrednictwem płytki PiFace (zobacz recepturę 8.16).

9.6. Sterowanie urządzeniami zasilanymi prądem przemiennym o wysokim napięciu

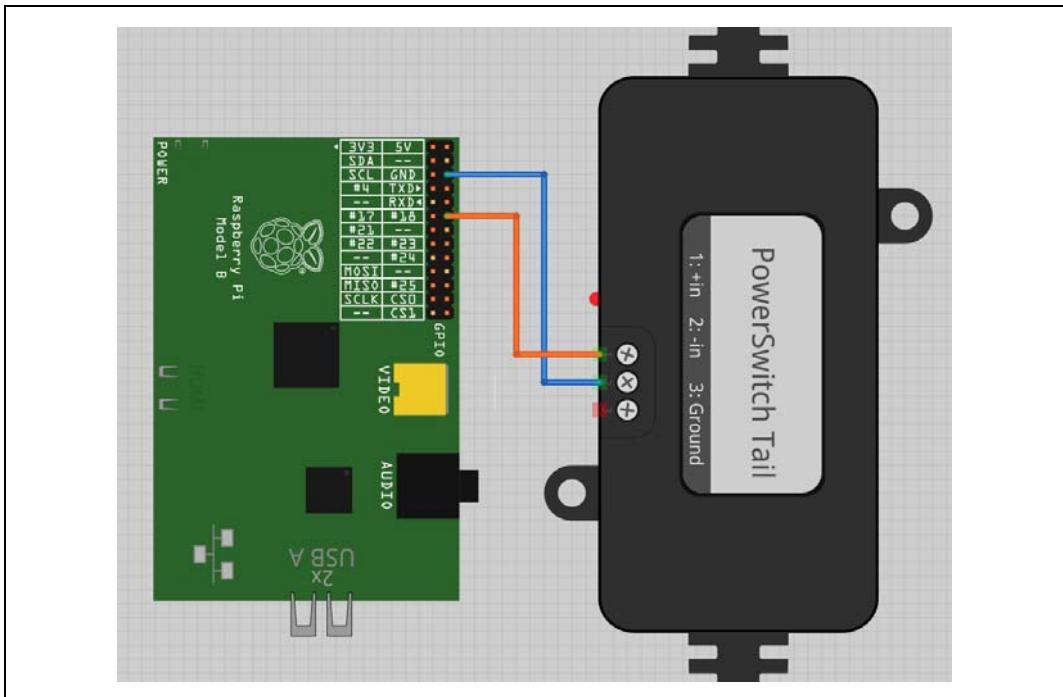
Problem

Chcesz sterować pracą urządzeń zasilanych prądem przemiennym o napięciu 230 lub 400 V.

Rozwiązanie

Zastosuj moduł PowerSwitch Tail II (zobacz sekcja „Moduły” w dodatku A). Będziesz musiał dokonać w nim pewnych modyfikacji — przystosować go do europejskich wtyczek. Urządzenie to pozwala w prosty sposób włączać i wyłączać urządzenia zasilane prądem przemiennym. Po jednej stronie posiada ono gniazdo sieciowe, a po drugiej wtyczkę. Urządzenie to przypomina przedłużacz, ale wyposażono je w trzy dodatkowe zaciski śrubowe. Złącze o numerze 2 należy podłączyć do masy, a złącze o numerze 1 do portu GPIO znajdującego się na Twoim Raspberry Pi. Moduł ten działa jak przełącznik włączający i wyłączający podłączone do niego urządzenie.

Do obsługi modułu możesz zastosować kod przedstawiony w recepturze 9.1. Na rysunku 9.7 zaprezentowano moduł PowerSwitch Tail podłączony do Raspberry Pi.



Rysunek 9.7. Moduł PowerSwitch Tail podłączony do Raspberry Pi

Omówienie

W omawianym module zastosowano przekaźnik, jest on jednak załączany za pośrednictwem tak zwanego **optoizolatora** — diody LED oświetlającej optotriak (włącznik wysokiego napięcia wrażliwy na światło). Zapalenie diody LED powoduje rozpoczęcie przepływu prądu przez optotriak i podłączoną do niego cewkę przekaźnika.

Dioda LED znajdująca się wewnętrznie optoizolatora posiada rezystor ograniczający prąd, a więc po podłączeniu tego modułu do złącza GPIO o napięciu 3,3 V pobierany będzie prąd o natężeniu zaledwie 3 mA.

Zobacz również

W recepturze 9.4 opisano układ przełączający zbudowany w oparciu o tranzystor mocy typu MOSFET, a w recepturze 9.5 omówiono zagadnienia dotyczące korzystania z przekaźników.

Istnieje również wersja modułu PowerSwitch Tail włączająca urządzenie trifazowe. Taki moduł możesz kupić w formie zestawu do samodzielnego montażu — <http://www.powerswitchtail.com/Pages/PSTKKit.aspx>.

9.7. Tworzenie graficznego interfejsu pozwalającego na włączanie i wyłączanie elektroniki podłączonej do Raspberry Pi

Problem

Chcesz napisać aplikację wyposażoną w graficzny interfejs użytkownika, pozwalającą na włączanie i wyłączanie elektroniki podłączonej do Raspberry Pi.

Rozwiązanie

Skorzystaj ze struktury Tkinter. Jest to struktura interfejsu użytkownika, którą możesz zastosować w celu napisania aplikacji Pythona posiadającej pole wyboru umożliwiające włączenie lub wyłączenie danego styku złącza GPIO (zobacz rysunek 9.8).



Rysunek 9.8. Graficzny interfejs pozwalający na włączanie i wyłączanie elektroniki podłączonej do Raspberry Pi

Do osiemnastego styku gniazda GPIO podłącz diodę LED lub jakieś inne urządzenie. Skorzystanie z diody LED (zobacz receptura 9.1) będzie najprostsze.

Otwórz edytor (nano lub IDLE) i umieść w jego oknie poniższy kod. Następnie zapisz plik pod nazwą *gui_switch.py*. Plik ten, podobnie jak pozostałe pliki zawierające aplikacje omówione w tej książce, można pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        self.check_var = BooleanVar()
        check = Checkbutton(frame, text='Pin 18',
                            command=self.update,
                            variable=self.check_var, onvalue=True, offvalue=False)
        check.grid(row=1)

    def update(self):
        GPIO.output(18, self.check_var.get())

root = Tk()
root.wm_title('Przelacznik')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Przedstawiony program należy uruchamiać za pomocą polecenia z przedrostkiem sudo. W przeciwnym wypadku biblioteka RPi.GPIO nie uzyska dostępu do zasobów sprzętowych bez uprawnień administratora.

```
$ sudo python gui_switch.py
```



Jeżeli korzystasz z Pythona w wersji 3, to musisz zmienić nazwę biblioteki Tkinter na tkinter (nazwa biblioteki powinna się rozpoczynać małą literą „t”).

Omówienie

W przedstawionym programie zdefiniowano klasę o nazwie App, w której umieszczono większość kodu programu. Funkcja inicjująca ją tworzy zmienną składową o nazwie check_var, która zawiera egzemplarz BooleanVar. Współgra on ze zmienną variable, której wartość ulega zmianie w zależności od tego, czy pole wyboru zostało zaznaczone przez użytkownika, czy nie. Opcja command uruchamia funkcję update po każdej zmianie wartości tej zmiennej.

Funkcja update kieruje wartość check-var do złącza wyjściowego interfejsu GPIO.

Zobacz również

Program ten może być użyty do sterowania pracą: diody LED (receptura 9.1), urządzenia o dużej mocy zasilanego prądem stałym (receptura 9.4), przekaźnika (receptura 9.5) i urządzenia zasilanego prądem przemiennym o wysokim napięciu (receptura 9.6).

9.8. Tworzenie graficznego interfejsu użytkownika pozwalającego na sterowanie mocą diod i silników za pomocą modulacji czasu trwania impulsu

Problem

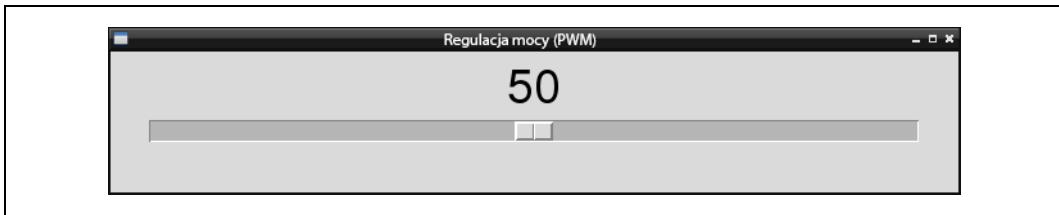
Chcesz napisać aplikację wyposażoną w graficzny interfejs użytkownika, pozwalającą na sterowanie mocą elektroniki podłączonej do Raspberry Pi za pomocą suwaka regulującego działanie układu PWM.

Rozwiązanie

Skorzystaj ze struktury Tkinter. To struktura interfejsu użytkownika, którą możesz zastosować w celu napisania aplikacji Pythona posiadającej suwak sterujący pracą układu PWM od 0 do 100 procent mocy (zobacz rysunek 9.9).

Do styku o numerze 18 podłącz diodę LED lub inne urządzenie elektroniczne, które może być sterowane za pomocą układu PWM. Skorzystanie z diody LED (zobacz receptura 9.1) będzie najprostsze.

Otwórz edytor (nano lub IDLE) i umieść w jego oknie poniższy kod. Następnie zapisz plik pod nazwą *gui_slider.py*. Plik ten, podobnie jak pozostałe pliki zawierające aplikacje omówione w tej książce, można pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.



Rysunek 9.9. Graficzny interfejs pozwalający na sterowanie mocą elektroniki podłączonej do układu PWM Raspberry Pi

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 500)
pwm.start(100)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=100,
                      orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

    def update(self, duty):
        pwm.ChangeDutyCycle(float(duty))

root = Tk()
root.wm_title('Regulacja mocy (PWM)')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Przedstawiony program należy uruchamiać z przedrostkiem sudo. W przeciwnym wypadku biblioteka RPi.GPIO nie uzyska dostępu do zasobów sprzętowych bez uprawnień administratora.

```
$ sudo python gui_slider.py
```



Jeżeli korzystasz z Pythona w wersji 3, to musisz zmienić nazwę biblioteki Tkinter na tkinter (nazwa biblioteki powinna się rozpoczynać małą literą „t”).

Omówienie

W przedstawionym programie zdefiniowano klasę o nazwie App, w której umieszczono większość kodu programu. Opcja command uruchamia funkcję update za każdym razem, gdy użytkownik zmieni pozycję suwaka. Funkcja ta zmienia cykl roboczy złącza wyjściowego.

Zobacz również

Program ten może być użyty do sterowania pracą: diody LED (receptura 9.1), silnika zasilanego prądem stałym (receptura 10.3) i urządzenia o dużej mocy zasilanego prądem stałym (receptura 9.4).

9.9. Zmiana koloru diody RGB LED

Problem

Chcesz wpływać na kolor światła generowanego przez diodę RGB LED.

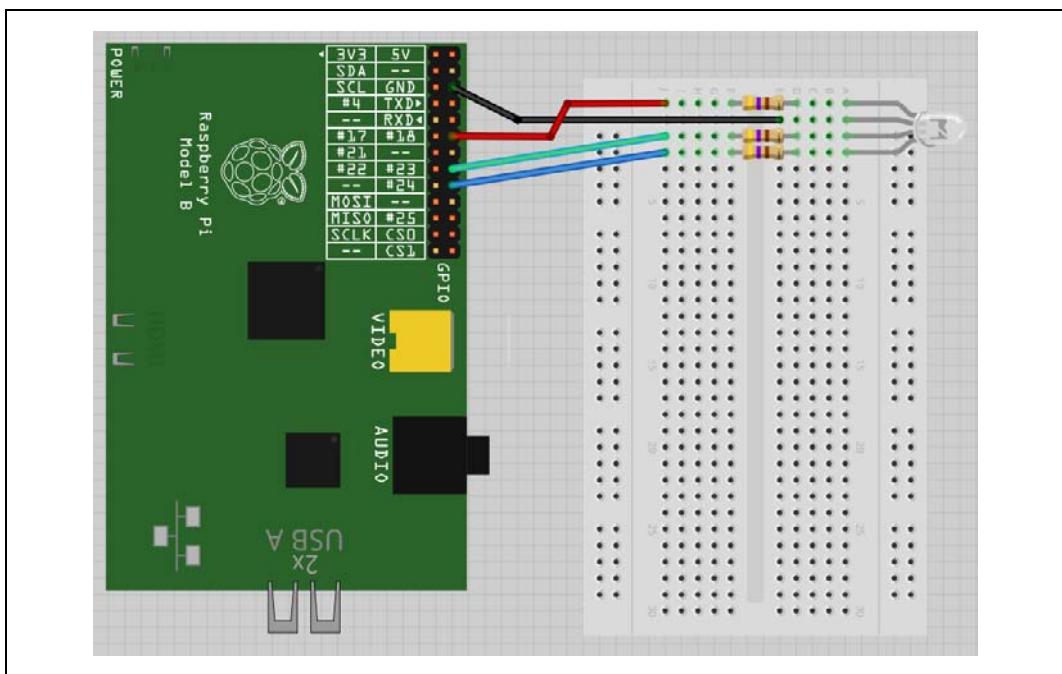
Rozwiązanie

Zastosuj układ PWM do sterowania mocą każdego z kanałów diody RGB LED.

Aby zbudować taki obwód, będziesz potrzebować:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- trzech rezystorów 1 kΩ (zobacz sekcję „Rezystory i kondensatory” w dodatku A),
- diody RGB o wspólnej katodzie (zobacz sekcję „Optoelektronika” w dodatku A),
- dodatkowej płytki Pi Plate (zobacz receptura 8.20) lub Humble Pi (zobacz receptura 8.19), w przypadku gdybyś chciał, żeby Twój projekt nie był tylko tymczasową konstrukcją.

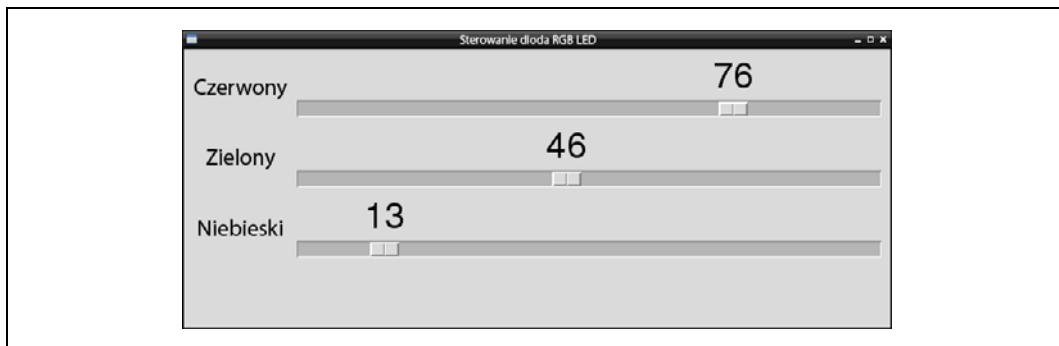
Na rysunku 9.10 przedstawiono schemat połączeń wykonanych pomiędzy diodą LED umieszczoną na płytce prototypowej a Raspberry Pi. Upewnij się, że dioda nie została podłączona odwrotnie. Jej najdłuższa nóżka powinna być ułożona tak, by była drugim wyprowadzeniem zamontowanej diody, licząc od góry płytki. Złącze to nosi nazwę **wspólnej katody** — ujemne złącza (katody) diody zielonej, diody czerwonej i diody niebieskiej połączono ze sobą. Zastosowanie jednej wspólnej katody zmniejsza liczbę złączy wyprowadzonych ze wspólnej obudowy tych diod.



Rysunek 9.10. Dioda RGB LED podłączona do Raspberry Pi

Gdybyś chciał, żeby Twój projekt nie był tylko tymczasową konstrukcją, to komponenty możesz przylutować do płytki Pi Plate lub Humble Pi. Proces ten opisano w recepturze 8.19.

Program, z którego będziemy korzystać, wyświetla na ekranie trzy suwaki służące do sterowania mocą trzech kanałów diody LED: czerwonego, zielonego i niebieskiego (zobacz rysunek 9.11).



Rysunek 9.11. Graficzny interfejs pozwalający na sterowanie diodą RGB LED

Otwórz edytor (nano lub IDLE) i umieść w jego oknie poniższy kod. Następnie zapisz plik pod nazwą `gui_sliderRGB.py`. Plik ten, podobnie jak pozostałe pliki zawierające aplikacje omówione w tej książce, można pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

pwmRed = GPIO.PWM(18, 500)
pwmRed.start(100)

pwmGreen = GPIO.PWM(23, 500)
pwmGreen.start(100)

pwmBlue = GPIO.PWM(24, 500)
pwmBlue.start(100)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        Label(frame, text='Czerwony').grid(row=0, column=0)
        Label(frame, text='Zielony').grid(row=1, column=0)
        Label(frame, text='Niebieski').grid(row=2, column=0)
        scaleRed = Scale(frame, from_=0, to=100,
                         orient=HORIZONTAL, command=self.updateRed)
        scaleRed.grid(row=0, column=1)
        scaleGreen = Scale(frame, from_=0, to=100,
                           orient=HORIZONTAL, command=self.updateGreen)
        scaleGreen.grid(row=1, column=1)
        scaleBlue = Scale(frame, from_=0, to=100,
                          orient=HORIZONTAL, command=self.updateBlue)
        scaleBlue.grid(row=2, column=1)

    def updateRed(self):
        print("Red: " + str(scaleRed.get()))

    def updateGreen(self):
        print("Green: " + str(scaleGreen.get()))

    def updateBlue(self):
        print("Blue: " + str(scaleBlue.get()))
```

```
scaleBlue.grid(row=2, column=1)

def updateRed(self, duty):
    pwmRed.ChangeDutyCycle(float(duty))

def updateGreen(self, duty):
    pwmGreen.ChangeDutyCycle(float(duty))

def updateBlue(self, duty):
    pwmBlue.ChangeDutyCycle(float(duty))

root = Tk()
root.wm_title('Sterowanie dioda RGB LED')
app = App(root)
root.geometry("200x150+0+0")
root.mainloop()
```

Omówienie

Przedstawiony program działa podobnie do aplikacji sterującej jednym kanałem PWM omówionej w recepturze 9.8. Teraz mamy jednak do czynienia z programem obsługującym trzy kanały PWM — każdy kanał odpowiada za jedną barwę składową diody. Każdej barwie przypisano odrębny suwak.

Korzystałem z diody RGB LED o wspólnej katodzie. Jeżeli posiadasz diodę o wspólnej anodzie, to możesz korzystać z tego programu, ale anodę takiej diody należy podłączyć do styku pochodzącego napięcie 3,3 V na złączu GPIO. Suwaki w uruchomionym programie będą działały odwrotnie — ustawienie suwaka w pozycji 100 sprawi, że dany kanał diody będzie wyłączony, a ustawienie go w pozycji 0 sprawi, że dany kanał diody będzie pracował z pełną mocą.

Wybierając diodę, postaraj się znaleźć diodę rozpraszającą (posiadającą oznaczenie *diffused*). Kolory takiej diody będą ze sobą lepiej wymieszane.

Zobacz również

Jeżeli chcesz sterować tylko jednym kanałem PWM, zajrzyj do receptury 9.8.

Więcej informacji na temat korzystania z płytki prototypowej i przewodów połączeniowych znajdziesz w recepturze 8.10.

9.10. Tworzenie multimedialnego centrum rozrywki

Problem

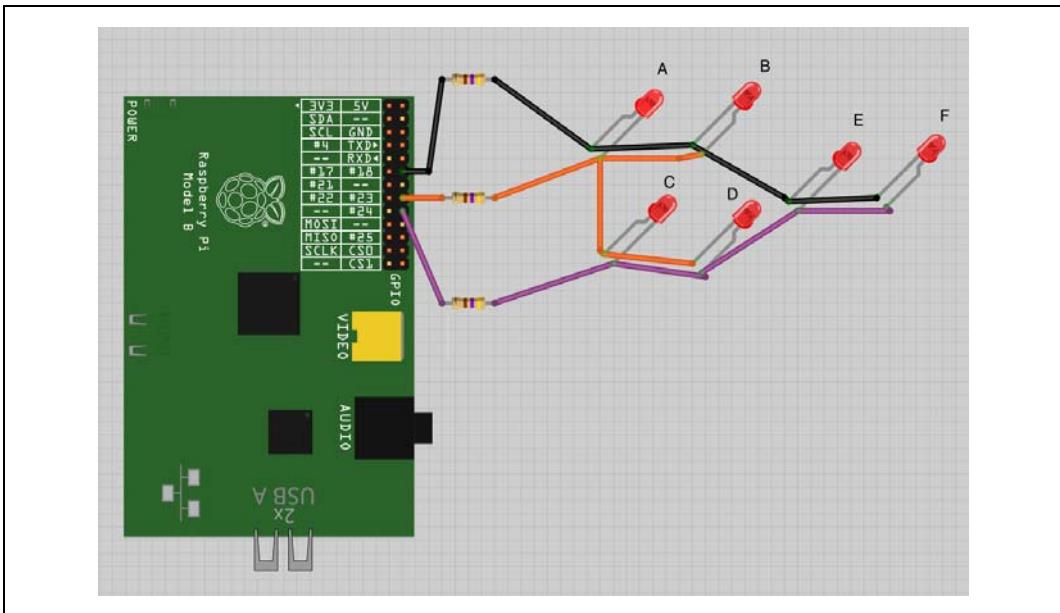
Chcesz sterować wieloma diodami LED za pomocą jak najmniejszej liczby pinów złącza GPIO.

Rozwiązanie

Skorzystaj z techniki o nazwie *Charlieplexing*. Nazwa tej techniki pochodzi od imienia jej wynalazcy — Charliego Allena — pracownika firmy Maxim. Technika ta wykorzystuje to, że pin złącza GPIO może pracować jako wejście lub wyjście. Tryby pracy mogą być zmieniane

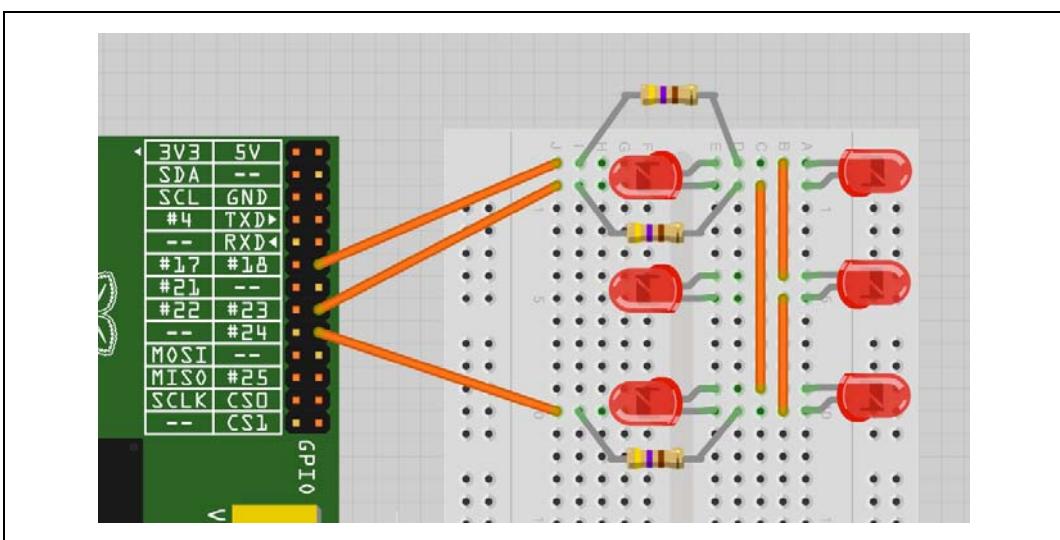
w trakcie działania programu. Gdy dany pin pracuje jako wejście, to nie wypływa z niego prąd o natężeniu wystarczającym do zapalenia diody LED. Zmiana trybu pracy jednego ze styków złącza GPIO nie wpływa na pracę pozostałych pinów.

Na rysunku 9.12 pokazano układ sterujący pracą sześciu diod za pomocą trzech styków złącza GPIO.



Rysunek 9.12. Technika Charlieplexing

Na rysunku 9.13 zaprezentowano ułożenie elementów tego układu na płytce prototypowej.



Rysunek 9.13. Technika Charlieplexing — elementy zamontowane na płytce prototypowej

Aby zbudować ten obwód, będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- sześciu diod LED (zobacz sekcja „Optoelektronika” w dodatku A).

Umieść poniższy kod w oknie środowiska programistycznego IDLE (receptura 5.2) lub w edytorze tekstowym nano (receptura 3.6) i zapisz utworzony plik pod nazwą *charlieplexing.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Program ten prosi użytkownika o podanie liczby z zakresu od 0 do 5, a następnie zapala jedną z sześciu diod LED:

```
import RPi.GPIO as GPIO

pins = [18, 23, 24]

pin_led_states = [
    [1, 0, -1], #A
    [0, 1, -1], #B
    [-1, 1, 0], #C
    [-1, 0, 1], #D
    [1, -1, 0], #E
    [0, -1, 1] #F
]

GPIO.setmode(GPIO.BCM)

def set_pin(pin_index, pin_state):
    if pin_state == -1:
        GPIO.setup(pins[pin_index], GPIO.IN)
    else:
        GPIO.setup(pins[pin_index], GPIO.OUT)
        GPIO.output(pins[pin_index], pin_state)

def light_led(led_number):
    for pin_index, pin_state in enumerate(pin_led_states[led_number]):
        set_pin(pin_index, pin_state)

set_pin(0, -1)
set_pin(1, -1)
set_pin(2, -1)

while True:
    x = int(raw_input("Podaj numer pinu (od 0 do 5):"))
    light_led(x)
```

Omówienie

Przeanalizujmy działanie techniki Charlieplexing. Założymy, że chcesz zapalić diodę A (zobacz rysunek 9.12). Dioda LED zapali się tylko wtedy, gdy na złączu, do którego podłączone jest jej dodatnie wyprowadzenie, pojawi się sygnał wysoki, a na złączu, do którego podłączone jest jej ujemne wyprowadzenie, pojawi się sygnał niski. Jeżeli różnica potencjałów będzie odwrotna, dioda się nie zapali. Aby zapalić diodę A, musisz połączyć jedno jej wyprowadzenie ze stykiem złącza GPIO o numerze 18 (za pośrednictwem rezystora). Na styku tym powinien być podany sygnał wysoki. Drugie wyprowadzenie omawianej diody należy połączyć

ze stykiem złącza GPIO o numerze 23 (za pośrednictwem rezystora). Na styku tym powiniennabyć podany sygnał niski. Musisz jednak skonfigurować również styk o numerze 24, tak aby działał on jako wejście. W przeciwnym wypadku zapali się również dioda C lub D (w zależności od sygnału podawanego na styku o numerze 24).

Tablica `pin_led_states` przechowuje konfigurację złącz dla każdej z sześciu diod LED. Jeżeli w tablicy umieszczono wartość 0, to oznacza to, że na danym styku podawany jest sygnał niski, 1 oznacza podanie sygnału wysokiego, a -1 powoduje, że dany styk będzie działał jako wejście.

Liczبę diod LED, które można obsługiwać przez daną liczbę styków złącza GPIO, można obliczyć za pomocą wzoru:

$$\text{liczba diod} = n^2 - n$$

Do czterech styków można podłączyć 4, 12 lub 16 diod. Do dziesięciu styków można podłączyć aż 90 diod.

W podanym przykładowym programie diody są zapalane pojedynczo. Jeżeli chcesz zapalać jednocześnie więcej diod, musisz stworzyć pętlę odświeżającą, która będzie modyfikować kolejne elementy tablicy zawierającej informacje o diodach. Pętla ta będzie zapalać kolejno diody. Operacja ta musi przebiegać tak szybko, aby obserwator odniósł wrażenie, że kilka diod świeci równocześnie światłem ciągłym.

Im więcej diod będzie zapalanych w ten sposób, tym krótszy będzie czas, przez jaki pojedyncza dioda będzie włączona. Diody będą świecić ciemniej.

Zobacz również

Więcej informacji na temat techniki Charlieplexing znajdziesz w Wikipedii — <http://en.wikipedia.org/wiki/Charlieplexing>. Jeżeli chcesz sterować pracą tylko jednej diody LED, zajrzyj do receptury 9.1.

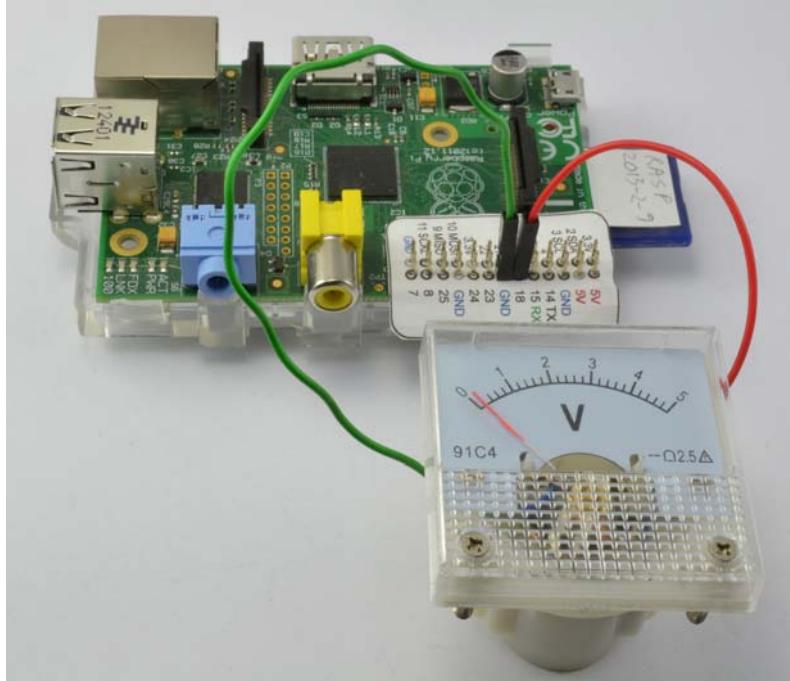
9.11. Stosowanie analogowego woltomierza w charakterze wyświetlacza wskazówkowego

Problem

Chcesz podłączyć analogowy woltomierz do Raspberry Pi.

Rozwiązanie

Jeżeli dysponujesz woltomierzem pracującym w zakresie do 5 V, to możesz go podłączyć bezpośrednio do układu PWM. W takim przypadku dodatni biegum miernika należy podłączyć do styku złącza GPIO, a ujemny do masy (zobacz rysunek 9.14). Jeżeli jednak dysponujesz miernikiem pracującym w zakresie do 5 V o wspólnej masie zasilania, to będziesz mógł nim mierzyć napięcia tylko do 3,3 V.



Rysunek 9.14. Sterowanie woltomierzem podłączonym bezpośrednio do złącza GPIO

Jeżeli chcesz korzystać z pełnej skali woltomierza o zakresie pomiarowym do 5 V, będziesz musiał zastosować tranzystor pełniący funkcję przełącznika sygnału PWM, a także rezystor 1 kΩ ograniczający prąd płynący przez bazę tranzystora.

Aby zbudować ten obwód, będziesz potrzebować:

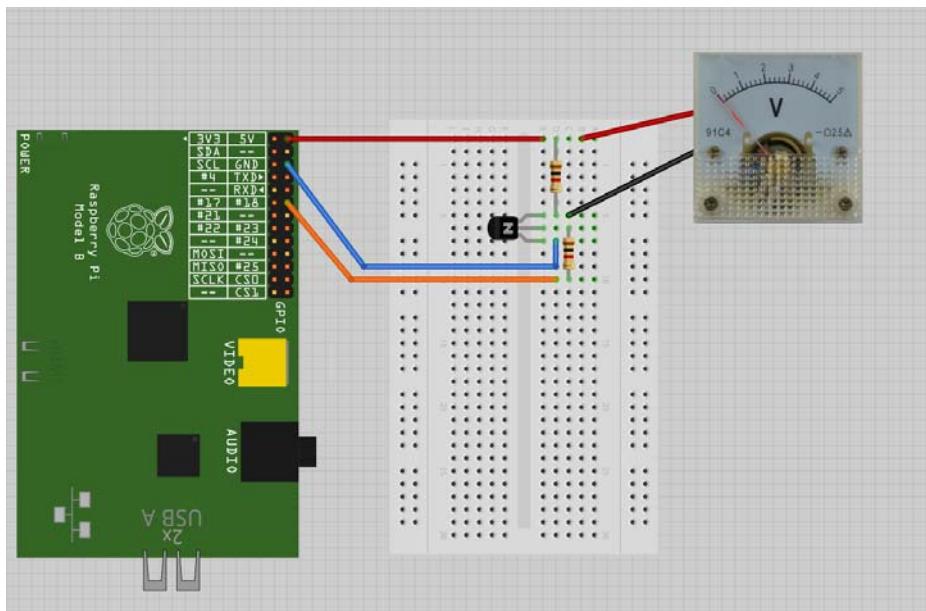
- woltomierza o zakresie pomiarowym do 5 V (zobacz sekcja „Pozostałe” w dodatku A),
- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- dwóch rezystorów 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- tranzystora 2N3904 (zobacz sekcja „Tranzystory i diody” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej pokazano na rysunku 9.15.

Omówienie

W celu sprawdzenia woltomierza skorzystaj z programu, którym sterowałaś jasnością diody LED (zobacz receptura 9.2).

Najprawdopodobniej wskazówka będzie się zachowywała stabilnie tylko na krańcach skali. W pozostałych punktach będzie lekko drgała. Jest to efekt uboczny wynikający z charakterystyki sygnału generowanego przez układ PWM. W celu zwiększenia stabilności pracy wskazówki miernika możesz skorzystać z zewnętrznego modułu PWM. Przykładowy sześciokanałowy moduł zastosowano w recepturze 10.2.



Rysunek 9.15. Podłączanie woltomierza o zakresie pomiarowym do 5 V do styku złącza GPIO, na którym podawane jest napięcie 3,3 V

Zobacz również

Więcej informacji na temat działania analogowego woltomierza znajdziesz w Wikipedii — <https://pl.wikipedia.org/wiki/Woltomierz>.

Więcej informacji dotyczących korzystania z płytka prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

9.12. Tworzenie programów korzystających z przerwań

Problem

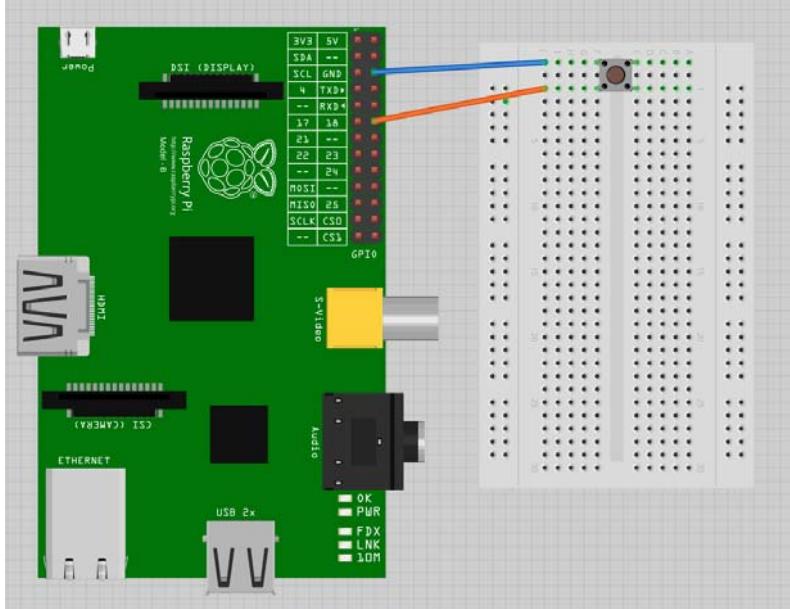
Chcesz wywołać odpowiedź na jakieś zdarzenie — np. na wcisnięcie przycisku — bez konieczności ciągłego monitorowania złącza wejściowego.

Rozwiążanie

Zastosuj funkcję `add_event_detect` znajdującą się w bibliotece `RPi.GPIO`.

Poniższy przykład pokazuje Ci sposób, w jaki procedura przerwania może zostać wywołana na skutek wcisnięcia przycisku.

Na płytce prototypowej zainstaluj przycisk, tak jak pokazano na rysunku 9.16.



Rysunek 9.16. Przycisk podłączony do gniazda GPIO w celu pokazania działania przerwań

Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz utworzony plik pod nazwą *interrupts.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Poniższy program co sekundę wyświetla kolejną liczbę. Wciśnięcie przycisku spowoduje wyświetlenie komunikatu *Wciszasz przycisk*.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

def my_callback(channel):
    print('Wciszasz przycisk')

GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.add_event_detect(18, GPIO.FALLING, callback=my_callback)

i = 0
while True:
    i = i + 1
    print(i)
    time.sleep(1)
```

Program uruchomiony z uprawnieniami administratora powinien wyświetlać wspomniany wcześniej komunikat po wciśnięciu przycisku.

```
$ sudo python interrupts.py
1
2
3
Wciszasz przycisk
4
```

Wcisñasz przycisk
5
Wcisñasz przycisk
Wcisñasz przycisk
6

Omówienie

Wciśnięcie przycisku — zmianę stanu wejścia GPIO — możesz sprawdzać za pomocą pętli:

```
while True:  
    if GPIO.input(18) == False:  
        # umieść tu uruchamiany kod  
    time.sleep(0.1)
```

Wadą takiego rozwiązania jest to, że nie pozwala ono na wykonywanie innych operacji poza sprawdzaniem tego, czy przycisk nie został wciśnięty. Kolejną wadą jest to, że szybkie i krótkie przyciśnięcie guzika może nie zostać zarejestrowane przez funkcję `GPIO.input`. Ta technika programistyczna nosi nazwę **zapytywania**.

Przerwania działają w inny sposób. Pozwalają na skojarzenie funkcji z jednym z pinów złącza GPIO, tak że zmiana sygnału odbieranego na danym złączu spowoduje uruchomienie określonej funkcji.

Działanie przerwań dobrze ilustruje zamieszczony wcześniej przykładowy program. Najpierw zdefiniowano w nim funkcję o nazwie `my_callback`. Przyjmuje ona jeden argument, który określa wejście wyzwalające przerwanie. Dzięki temu możliwe jest zastosowanie tej samej funkcji obsługującej kilka przerwań.

```
def my_callback(channel):  
    print('Wcisñasz przycisk')
```

W tym przykładzie funkcja `my_callback` po prostu wyświetla komunikat.

Poniższa linia kodu odpowiada za właściwe powiązanie:

```
GPIO.add_event_detect(18, GPIO.FALLING, callback=my_callback)
```

Pierwszy parametr określa pin złącza GPIO (18). Drugim argumentem może być `GPIO.FALLING` lub `GPIO.RISING`. Argument `GPIO.FALLING` spowoduje, że funkcja będzie wywoływana tylko wówczas, gdy sygnał podawany na złączce GPIO przejdzie ze stanu wysokiego w niski. Taką sytuację mamy w analizowanym przykładzie — wciśnięcie przycisku powoduje opadnięcie sygnału podwyższanego przez wewnętrzny rezystor podciągający. Natomiast jeżeli w roli argumentu zastosujemy `GPIO.RISING`, to funkcja będzie wywoywana tylko wtedy, gdy sygnał odbierany przez złącze wejściowe przejdzie ze stanu niskiego w wysoki (gdy przycisk zostanie zwolniony).

Funkcja obsługująca zdarzenie nie zatrzymuje działania głównej pętli liczącej. Jest ona uruchamiana w odrębnym wątku wykonywania programu.

Przełączniki często *podskakują* podczas wciskania — nie przechodzą od razu ze stanu otwartego w zamknięty. Przez chwilę balansują pomiędzy tymi dwoma stanami. Pomimo że przycisk został wciśnięty tylko raz, program analizujący sygnał generowany przez taki włącznik może dojść do wniosku, że włącznik został kilkakrotnie bardzo szybko wciśnięty.

Jeżeli będziesz wciskał przycisk, z pewnością zauważysz, że program czasami wyświetla komunikat dwukrotnie, mimo że przycisk został wciśnięty tylko jeden raz.

W bibliotekę wbudowano funkcję rozwiązującą ten problem. Przeciwdziała ono ponownemu uruchomieniu przerwania w ciągu określonego czasu. Aby skorzystać z tej funkcji, wystarczy, że do wywołania funkcji `add_event_detect` dodasz parametr `bouncetime`. Wartość parametru `bouncetime` należy podawać w milisekundach.

```
GPIO.add_event_detect(18, GPIO.FALLING, callback=my_callback, bouncetime=100)
```

Zobacz również

Więcej informacji na temat podłączania przełączników do Raspberry Pi znajdziesz w recepturze 11.1.

9.13. Sterowanie złączem GPIO za pomocą sieci Web

Problem

Chcesz sterować pracą wyjść złącza GPIO za pośrednictwem sieci Web.

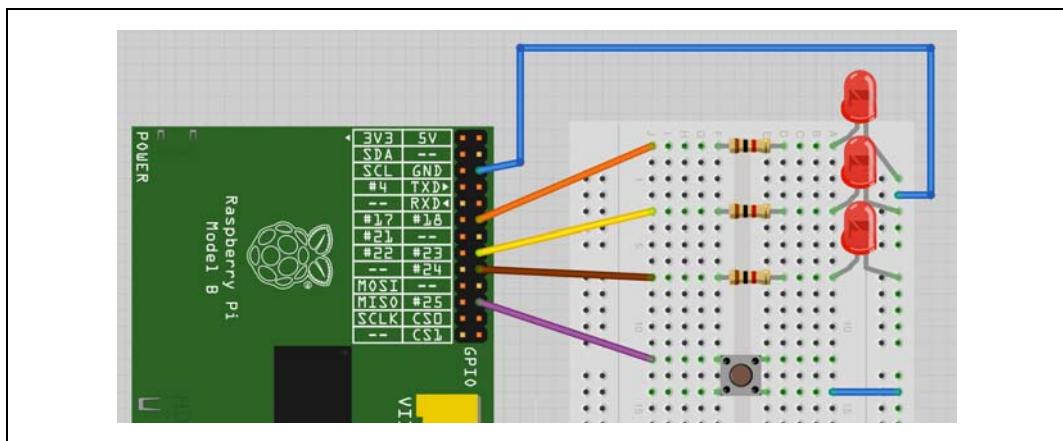
Rozwiązanie

Skorzystaj z biblioteki `bottle` (zobacz receptura 7.16). Pozwoli ona na stworzenie interfejsu sieciowego umożliwiającego sterowanie złączem GPIO za pośrednictwem dokumentu HTML.

Aby skorzystać z tej receptury, musisz zbudować obwód składający się z:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- trzech rezystorów 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- trzech diod LED (zobacz sekcja „Optoelektronika” w dodatku A),
- trzech przełączników chwilowych (zobacz sekcja „Pozostałe” w dodatku A).

Na rysunku 9.17 przedstawiono ułożenie tych elementów na płytce prototypowej.



Rysunek 9.17. Schemat obwodu sterowanego za pośrednictwem witryny sieci Web

Proces instalacji biblioteki bottle opisano w recepturze 7.16.

Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz utworzony plik pod nazwą *web_control.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Na stronie tej znajdziesz także program *web_control_test*. Można nim sprawdzić działanie obwodu. Program ten zapala kolejno diody i wyświetla informację o tym, czy przycisk jest wcisnięty. Nie posiada on funkcji serwera sieciowego.

Oto kod programu *web_control.py*:

```
from bottle import route, run
import RPi.GPIO as GPIO

host = '192.168.1.8'

GPIO.setmode(GPIO.BCM)
led_pins = [18, 23, 24]
led_states = [0, 0, 0]
switch_pin = 25

GPIO.setup(led_pins[0], GPIO.OUT)
GPIO.setup(led_pins[1], GPIO.OUT)
GPIO.setup(led_pins[2], GPIO.OUT)
GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def switch_status():
    state = GPIO.input(switch_pin)
    if state:
        return 'Nie wciskasz'
    else:
        return 'Wciskasz'

def html_for_led(led):
    l = str(led)
    result = " <input type='button' onClick='changed(" + l + ")" + value='LED " + l + "'/>"
    return result

def update_leds():
    for i, value in enumerate(led_states):
        GPIO.output(led_pins[i], value)

@route('/')
@route('/<led>')
def index(led="n"):
    print(led)
    if led != "n":
        led_num = int(led)
        led_states[led_num] = not led_states[led_num]
        update_leds()
    response = "<script>"
    response += "function changed(led)"
    response += "{"
    response += "    window.location.href='/"+ led"
    response += "}"
    response += "</script>

    response += '<h1>Sterowanie gniazdem GPIO</h1>'
    response += '<h2>Przycisk=' + switch_status() + '</h2>'
    response += '<h2>Diody</h2>'
    response += html_for_led(0)
    response += html_for_led(1)
    response += html_for_led(2)
    return response

run(host=host, port=80)
```

Przed uruchomieniem programu zmodyfikuj jedną z początkowych linii kodu — określ adres IP swojego Raspberry Pi:

```
host = '192.168.1.8'
```

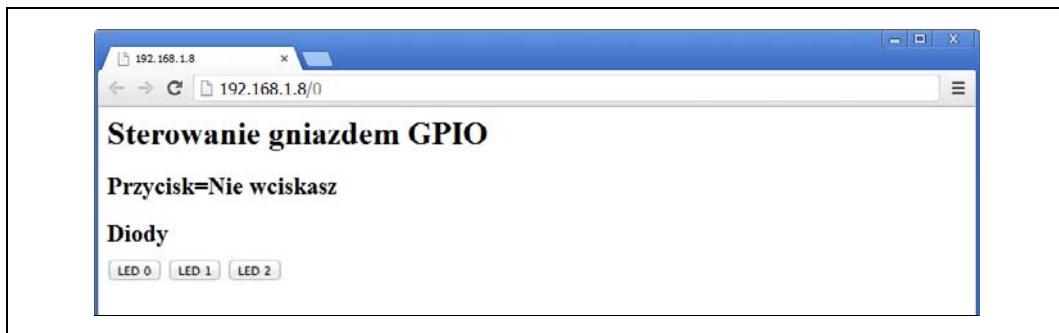
Program należy uruchomić jako użytkownik posiadający uprawnienia administratora:

```
sudo python web_control.py
```

Jeżeli program zostanie uruchomiony poprawnie, powinien się wyświetlić komunikat o treści podobnej do poniższej:

```
Bottle server starting up (using WSGIRefServer())...
Listening on http://192.168.1.8:80/
Hit Ctrl-C to quit.
```

Na dowolnym komputerze pracującym w Twojej sieci uruchom przeglądarkę. W polu adresu wpisz adres IP Twojego Raspberry Pi. Powinien się wyświetlić interfejs widoczny na rysunku 9.18.



Rysunek 9.18. Sieciowy interfejs sterujący pracą złącza GPIO

Klikając któryś przycisk LED, zobaczysz, jak skorelowana z nim dioda zapala się i gaśnie.

Jeżeli wciśniesz przycisk podłączony do Raspberry Pi i przeładujesz stronę w przeglądarce, to zobaczysz, że obok słowa *Przycisk* widnieje napis *Wcisasz* (zamiast *Nie wciskasz*).

Omówienie

Żeby przedstawić działanie programu, najpierw przyjrzymy się interfejsowi sieci Web. Działanie wszystkich interfejsów sieciowych opiera się na serwerze odpowiadającym na żądania wysypane przez przeglądarkę. W naszym przypadku funkcję serwera pełni program uruchomiony na Raspberry Pi.

Serwer po odebraniu żądania analizuje zawarte w nim informacje i generuje w odpowiedzi dokument HTML (dokument sformatowany w języku hipertekstowego znakowania informacji). W przypadku tego programu informacje pobierane przez serwer sieciowy sprowadzają się do jednej linii kodu:

```
def index(led="n"):
```

Jeżeli żądanie sieciowe odwołuje się do strony głównej (<http://192.168.1.8/>), to zmiennej led przypisana zostanie domyślna wartość parametru n. Jeżeli w przeglądarce wpisalibyśmy adres <http://192.168.1.8/2>, to liczba 2 znajdująca się na końcu adresu URL zostałaby przypisana do parametru led.

Przypisanie parametrowi led liczby 2 spowoduje przełączenie diody LED 2.

Aby uzyskać dostęp do adresu URL przełączającego diodę, musimy spowodować, że po kliknięciu przycisku *LED 2* strona zostanie przeładowana, a na końcu adresu zostanie dodany dodatkowy parametr. Cała sztuczka polega na umieszczeniu w dokumencie HTML funkcji JavaScript wpływającej na pracę przeglądarki. Gdy funkcja ta zostanie uruchomiona przez przeglądarkę, strona zostanie przeładowana, a na końcu adresu zostanie dodany odpowiedni parametr.

Wszystko to sprawia, że mamy do czynienia ze złożoną sytuacją, w której program napisany w Pythonie generuje kod JavaScript uruchamiany później w przeglądarce internetowej. Poniższe linie kodu generują wspomnianą funkcję JavaScript.

```
response = "<script>\nresponse += "function changed(led) {\nresponse += '\n    window.location.href='/' + led\nresponse += '}'\nresponse += "</script>"
```

Kod HTML nie jest powtarzany dla każdego z przycisków z osobna. Jest on generowany przez funkcję `html_for_led`:

```
def html_for_led(led):\n    l = str(led)\n    result = " <input type='button' onClick='changed(" + l + ")" value='LED ' + l + "'/>"\n    return result
```

Kod ten jest używany trzykrotnie — po jednym razie dla każdego przycisku. Przekazuje on informacje na temat kliknięcia przycisku do funkcji `changed`. Do funkcji w roli parametru przekazywany jest również numer diody LED.

Kod odpowiadający za pracę styków złącza GPIO znajduje się w funkcji `update_leds`. Jest ona wywoływana za każdym razem, gdy serwer otrzyma żądanie zawierające numer diody, którą trzeba przełączyć:

```
def update_leds():\n    for i, value in enumerate(led_states):\n        GPIO.output(led_pins[i], value)
```

Funkcja iteruje tablicę stanów, przełączając kolejne wyjścia we właściwy tryb pracy.

Poniższa linia kodu zawarta w funkcji `index` przełączca dla diody LED określonej parametrem `led` wartość znajdująca się w tablicy stanów.

```
led_states[led_num] = not led_states[led_num]
```

Sposób określania, czy przycisk jest wciśnięty, jest o wiele prostszy. Polega on na odczytaniu stanu wejścia i wygenerowaniu kodu HTML informującego odbiorcę o tym, czy przycisk jest wciśnięty. Zadania te są wykonywane przez funkcję `switch_status`:

```
def switch_status():\n    state = GPIO.input(switch_pin)\n    if state:\n        return 'Nie wciskasz'\n    else:\n        return 'Wciszasz'
```

Zobacz również

Więcej informacji na temat biblioteki `bottle` znajdziesz w recepturze 7.16 i w jej dokumentacji znajdującej się pod adresem <http://www.bottlepy.org/docs/dev/>.

10.0. Wprowadzenie

W tym rozdziale poruszono zagadnienia związane z podłączaniem różnych rodzajów silników do Raspberry Pi.

10.1. Sterowanie pracą serwomotoru

Problem

Chcesz sterować położeniem serwomotoru za pomocą Raspberry Pi.

Rozwiążanie

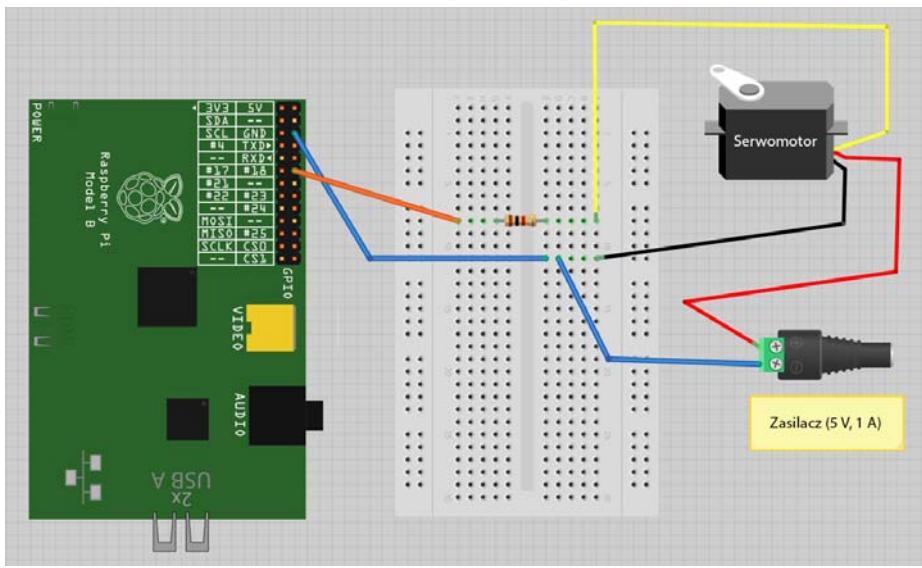
Kąt położenia serwomotoru możesz zmieniać za pomocą wbudowanego układu PWM. Co prawda można nim sterować pracą silnika, jednak praca tego układu nie jest w pełni stabilna, a więc ramię sterowanego silnika może drgać.

Serwomotor powinien być zasilany z zewnętrznego zasilacza dostarczającego prąd o napięciu 5 V. Wysokie prądy szczytowe pobierane przez silnik podłączony bezpośrednio do Raspberry Pi mogłyby spowodować jego niestabilną pracę, a nawet uszkodzenie.

Aby skorzystać z tej receptury, musisz zbudować obwód składający się z:

- serwomotoru zasilanego prądem o napięciu 5 V (zobacz sekcja „Pozostałe” w dodatku A),
- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- baterii o napięciu znamionowym 4,8 V lub zasilacza dostarczającego prąd o napięciu 5 V i natężeniu 1 A (zobacz sekcja „Pozostałe” w dodatku A).

Na rysunku 10.1 przedstawiono schemat wykonawczy obwodu.



Rysunek 10.1. Sterowanie pracą serwomotoru

Zastosowanie rezystora $1\text{ k}\Omega$ jest zabiegiem opcjonalnym zabezpieczającym złącze GPIO przed niespodziewanie wysokimi prądami sygnału sterującego, które mogą się pojawić na skutek uszkodzenia serwomotoru.

Przewody Twojego serwomotoru mogą mieć inne kolory niż przewody widoczne na rysunku 10.1. Zwykle przewód zasilający serwomotor prądem o napięciu 5 V ma kolor czerwony, przewód masowy — kolor brązowy, a przewód sygnału sterującego — pomarańczowy.

Jeżeli nie chcesz korzystać z dodatkowego zasilacza, możesz zasilać serwomotor baterią składającą się z czterech akumulatorów AA dających sumaryczne napięcie 4,8 V. Wiele serwomotorów może być zasilanych z czterech ogniw alkalicznych, które po połączeniu szeregowym dają napięcie 6 V. Zanim podłączysz prąd o napięciu 6 V do swojego serwomotoru, sprawdź w dokumentacji, czy nie spowoduje to jego uszkodzenia.

Interfejs programu sterującego położeniem ramienia serwomotoru oparto na programie `gui_slider.py`, który w recepturze 9.2 służył do sterowania jasnością diody LED. Możesz zmodyfikować ten program tak, aby służył on do regulacji kąta położenia ramienia serwomotoru od 0 do 180 stopni (zobacz rysunek 10.2).



Rysunek 10.2. Graficzny interfejs służący do sterowania serwomotorem

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz utworzony plik pod nazwą *servo.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Program ten posiada graficzny interfejs użytkownika, a więc nie możesz z niego korzystać za pośrednictwem protokołu SSH.

Musisz go uruchomić bezpośrednio na Raspberry Pi w oknie środowiska graficznego lub zdalnie za pomocą oprogramowania VNC (zobacz receptura 2.8). Aby program działał prawnie, musisz go uruchomić jako użytkownik posiadający uprawnienia administratora, a więc zastosuj w tym celu polecenie `sudo python servo.py`.

```
from Tkinter import *
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 100)
pwm.start(5)

class App:

    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        scale = Scale(frame, from_=0, to=180,
                      orient=HORIZONTAL, command=self.update)
        scale.grid(row=0)

    def update(self, angle):
        duty = float(angle) / 10.0 + 2.5
        pwm.ChangeDutyCycle(duty)

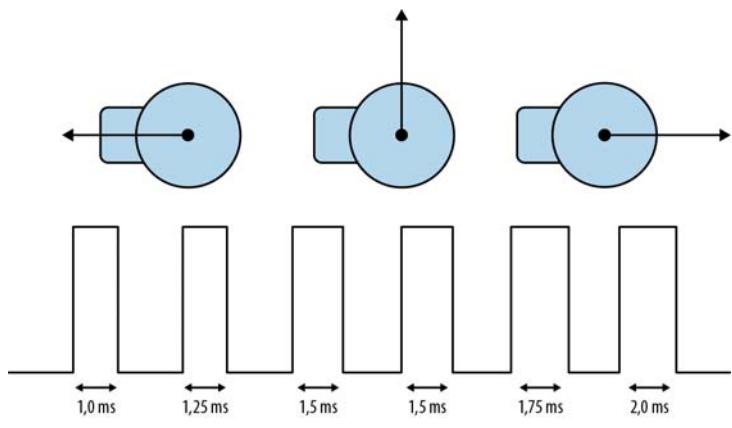
root = Tk()
root.wm_title('Sterowanie serwomotorem')
app = App(root)
root.geometry("200x50+0+0")
root.mainloop()
```

Omówienie

Serwomotory są stosowane do zdalnego sterowania pojazdami i robotami. Większość serwomotorów nie jest w stanie wykonać obrotu wokół własnej osi. Kąt obrotu wynosi zwykle około 180 stopni.

Pozycja, w jakiej ustawione jest ramię serwomotoru, zależy od długości impulsu. Serwomotor oczekuje impulsu co 20 milisekund. Jeżeli sygnał będzie wysoki przez 1 milisekundę, to serwomotor ustawi się w pozycji początkowej. Jeżeli sygnał będzie wysoki przez 1,5 milisekundy, to ustawi się on w połowie zakresu ruchu, a gdy wysoki sygnał będzie trwał przez 2 milisekundy, to ramię serwomotoru zostanie odchylone o 180 stopni w stosunku do pozycji początkowej (zobacz rysunek 10.3).

W zaprezentowanym programie układ PWM pracuje przy częstotliwości 100 Hz — impuls jest wysyłany do serwomotoru co 10 milisekund. Zakres ruchu ramienia serwomotoru jest podzielony na 100 pozycji. Z tego względu impulsy powodujące ustawienie serwomotoru w pozycji początkowej powinny być krótsze od 1 milisekundy, a impulsy powodujące maksymalne wychylenie serwomotoru powinny być dłuższe od 2 milisekund.



Rysunek 10.3. Działanie serwomotoru

Zobacz również

Jeżeli musisz sterować wieloma serwomotorami lub wymagasz od tych silników większej stabilności i precyzyji, to rozważ użycie dedykowanego modułu sterującego przedstawionego w recepturze 10.2.

Firma Adafruit opracowała inny sposób sterowania pracą serwomotorów. Informacje na jego temat znajdziesz na stronie <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-8-using-a-servo-motor>.

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

10.2. Sterowanie pracą wielu serwomotorów

Problem

Chcesz sterować pracą wielu serwomotorów. Ponadto wymagasz dużej dokładności i stabilności pracy tych silników.

Rozwiążanie

Zastosuj moduł sterujący serwomotorami. Taki moduł produkuje np. firma Adafruit: <http://www.adafruit.com/products/815>.

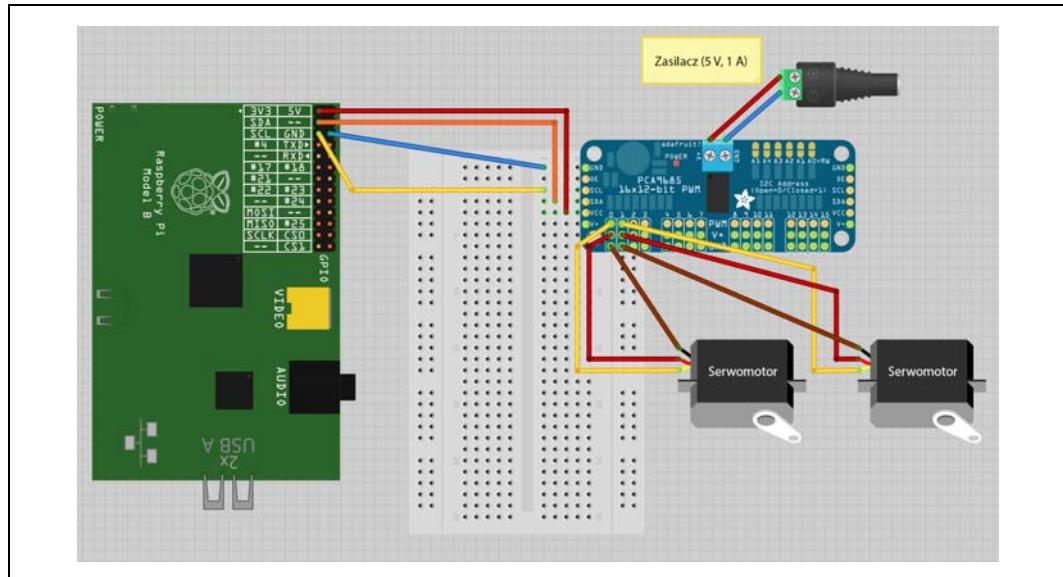
Moduł ten pozwala na sterowanie 16 serwomotorami lub kanałami PWM za pośrednictwem magistrali I2C znajdującej się na płytce Raspberry Pi.

Aby skorzystać z tej receptury, musisz zbudować obwód składający się z:

- przynajmniej jednego serwomotoru zasilanego prądem o napięciu 5 V (zobacz sekcja „Pozostałe” w dodatku A),
- dwunastobitowego sterownika kanałów PWM i serwomotorów wyprodukowanego przez firmę Adafruit (zobacz sekcja „Moduły” w dodatku A),

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- baterii o napięciu znamionowym 4,8 V lub zasilacza dostarczającego prąd o napięciu 5 V i natężeniu 1 A (zobacz sekcję „Pozostałe” w dodatku A).

Na rysunku 10.4 przedstawiono moduł podłączony do Raspberry Pi za pośrednictwem płytki prototypowej.



Rysunek 10.4. Moduł pozwalający na sterowanie serwomotorami lub kanałami PWM

Układ logiczny jest zasilany prądem o napięciu 3,3 V dostarczonym przez złącze Raspberry Pi. Serwomotory nie są zasilane z tego obwodu. Silniki są zasilane prądem o napięciu 5 V pochodząącym z zewnętrznego zasilacza.

Jeżeli nie chcesz korzystać z dodatkowego zasilacza, możesz zasilać serwomotor baterią składającą się z czterech akumulatorów AA dających w sumie napięcie 4,8 V. Wiele serwomotorów może być zasilanych z czterech ogniw alkalicznych, które po połączeniu szeregowym dają napięcie 6 V. Zanim podłączysz prąd o napięciu 6 V do swojego serwomotoru, sprawdź w dokumentacji, czy nie spowoduje to jego uszkodzenia.

Złącza serwomotorów są zorganizowane tak, że przewody silników można podłączać bezpośrednio do płytki. Uważaj, by nie podłączyć ich odwrotnie.

Przed zainstalowaniem oprogramowania omawianego modułu musisz zainstalować program Git (zobacz receptura 3.19), a także skonfigurować magistralę I2C (zobacz receptura 8.4). Pracę rozpocznij od wykonania instrukcji zawartych w tych dwóch recepturach.

Biblioteka Adafruit nie jest właściwie biblioteką — nie ma bowiem skryptu instalacyjnego. Jest to po prostu folder zawierający pliki. Korzystając z niej, musisz pracować w jej folderze. W przeciwnym wypadku program nie będzie w stanie odnaleźć potrzebnych plików.

W celu pobrania biblioteki Adafruit wpisz w oknie Terminala następujące polecenia:

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git  
$ cd Adafruit-Raspberry-Pi-Python-Code  
$ cd Adafruit_PWM_Servo_Driver
```

Ostatnie dwie linie zmieniają bieżący folder na folder, w którym znajdują się pliki obsługujące układ PWM, a także przykładowy program stworzony przez firmę Adafruit. Program ten możesz uruchomić za pomocą polecenia:

```
$ sudo python Servo_Example.py
```

Poniższy alternatywny przykładowy program jest zmodyfikowaną wersją programu omówionego wcześniej w recepturze 10.2. Położeniem serwomotoru można sterować za pomocą suwaka. Plik z programem należy zapisać w katalogu *Adafruit_PWM_Servo_Driver*. Suwak steruje jednocześnie pracą serwomotorów podłączonych do kanałów 0 i 1.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz utworzony plik pod nazwą *servo_module.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Program ten posiada graficzny interfejs użytkownika, a więc nie możesz z niego korzystać za pośrednictwem protokołu SSH. Musisz go uruchomić bezpośrednio na Raspberry Pi w oknie środowiska graficznego lub zdalnie za pomocą oprogramowania VNC (zobacz receptura 2.8).

```
from Tkinter import *  
from Adafruit_PWM_Servo_Driver import PWM  
import time  
  
pwm = PWM(0x40)  
pwm.setPWMFreq(50)  
  
class App:  
  
    def __init__(self, master):  
        frame = Frame(master)  
        frame.pack()  
        scale = Scale(frame, from_=0, to=180,  
                     orient=HORIZONTAL, command=self.update)  
        scale.grid(row=0)  
  
    def update(self, angle):  
        pulse_len = int(float(angle) * 500.0 / 180.0) + 110  
        pwm.setPWM(0, 0, pulse_len)  
        pwm.setPWM(1, 0, pulse_len)  
  
root = Tk()  
root.wm_title('Sterowanie serwomotorem')  
app = App(root)  
root.geometry("200x50+0+0")  
root.mainloop()
```

Omówienie

Pierwsza linia kodu po poleceniu `import` tworzy nowy egzemplarz PWM, korzystając z adresu magistrali I2C określonej przez argument (w tym przypadku jest to adres 0x40). Moduł posiada złącza, które pozwalają na zmianę adresu magistrali I2C w przypadku pojawienia się konfliktu z innym urządzeniem korzystającym z tej samej magistrali. Zmiana adresu magistrali będzie konieczna w przypadku korzystania z więcej niż jednego modułu serwomotorów.

Kolejna linia kodu określa częstotliwość działania układu PWM (50 Hz). Impuls będzie dostarczany do serwomotoru co 20 milisekund.

Dopiero poniższa linia kodu kieruje sygnał PWM do określonego kanału:

```
pwm.setPWM(0, 0, pulse_len)
```

Pierwszym argumentem podanym w tej linii jest numer kanału PWM, którego cykl pracy ma zostać zmodyfikowany. Każdy cykl pracy układu PWM jest podzielony na 4096 cykli zegara. Drugi z podanych argumentów określa cykl zegara, w którym rozpocznie się generowanie impulsu. W naszym przypadku jest to 0. Trzeci argument określa, przy którym cyklu zegara generowanie impulsu powinno zostać zakończone.

Stale 500.0 i 110 (znajdujące się w kolejnej linii kodu) zostały określone metodą prób i błędów — pozwalają one na ruch ramienia standardowego serwomotoru w zakresie 180 stopni:

```
pulse_len = int(float(angle) * 500.0 / 180.0) + 110
```

Wybierając zasilacz do tego modułu, pamiętaj o tym, że standardowy zdalnie sterowany serwomotor podczas wykonywania ruchu może pobierać prąd o natężeniu 400 mA. Gdy jego ramię będzie czymś obciążone, to pobór prądu może być jeszcze większy. Jeżeli planujesz jednocześnie poruszać wieloma dużymi serwomotorami, to musisz zakupić porządky zasilacz.

Zobacz również

Jeżeli chcesz sterować serwomotorem bezpośrednio (bez użycia modułu), to zajrzyj do receptury 10.1. Jeżeli chcesz sterować jednocześnie wieloma serwomotorami za pośrednictwem Arduino, przejdź do receptury 14.9.

Dokumentację biblioteki firmy Adafruit znajdziesz pod adresem <https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/library-reference>.

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

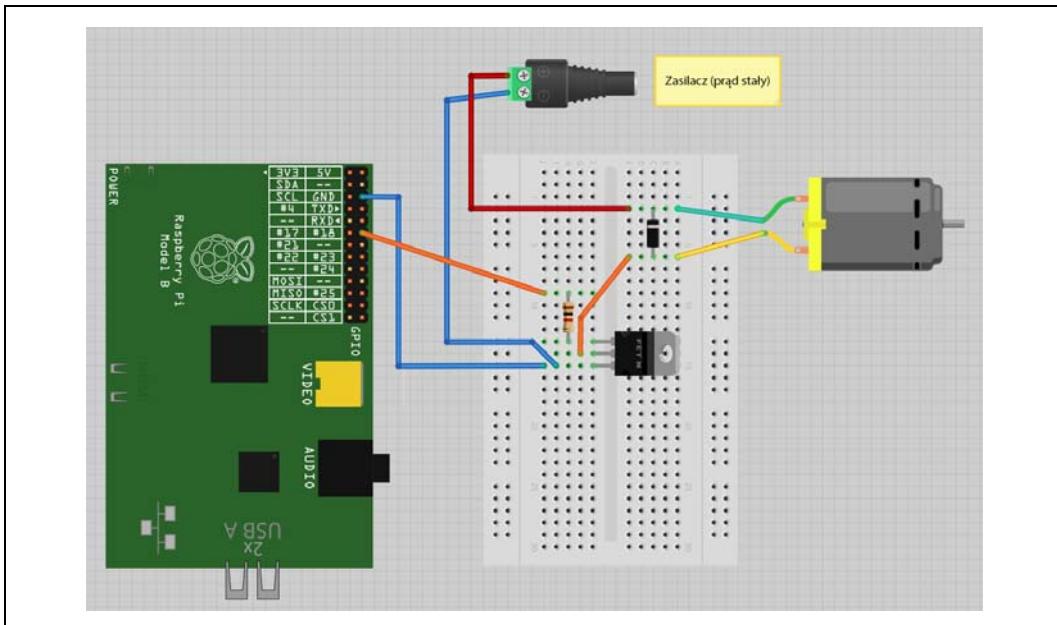
10.3. Sterowanie prędkością obrotową silnika zasilanego prądem stałym

Problem

Chcesz sterować za pośrednictwem Raspberry Pi prędkością obrotową silnika zasilanego prądem stałym.

Rozwiązanie

Możesz skorzystać z obwodu przedstawionego w recepturze 9.4, jednak do obwodu silnika warto dodać diodę (zobacz rysunek 10.5). Będzie ona zapobiegała uszkodzeniu tranzystora i Raspberry Pi przez wysokie napięcia szczytowe. Diodą, która dobrze się sprawdzi w tej roli, będzie 1N4001 (zobacz sekcję „Tranzystory i diody” w dodatku A). Dioda posiada oznaczenie w postaci paska — uważaj, żeby nie zamontować jej odwrotnie.



Rysunek 10.5. Sterowanie silnikiem o dużej mocy

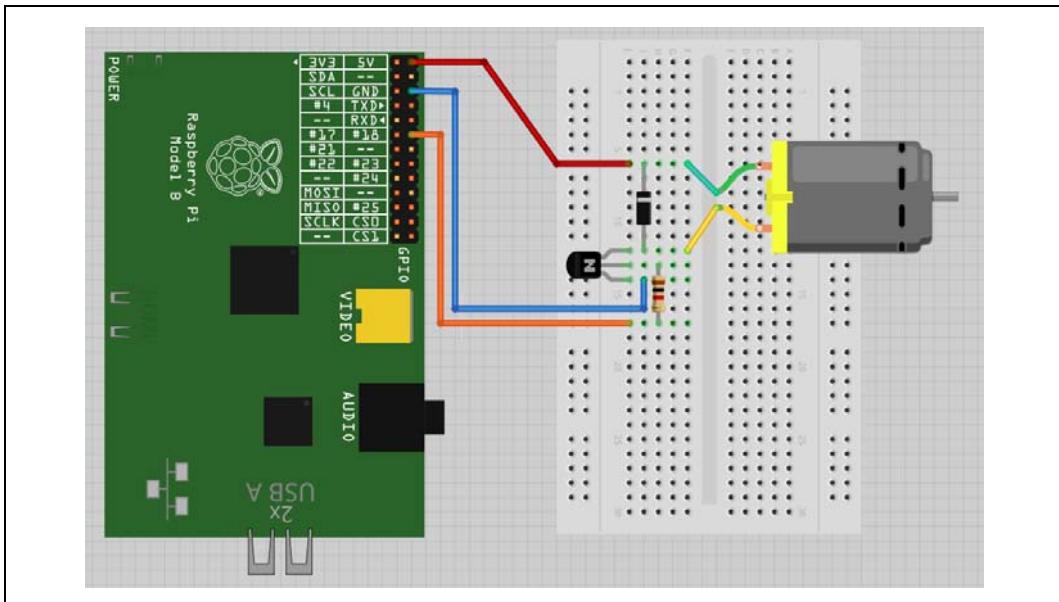
Jeżeli używasz silników o małej mocy (pobierających prąd o natężeniu mniejszym niż 200 mA), to możesz skorzystać z obwodu podobnego do obwodu przekaźnika przedstawionego w recepturze 9.5. Będziesz potrzebować:

- silnika zasilanego prądem stałym o napięciu od 3 do 12 V,
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora $1\text{ k}\Omega$ (zobacz sekcję „Rezystory i kondensatory” w dodatku A),
- tranzystora 2N3904 (zobacz sekcję „Tranzystory i diody” w dodatku A),
- diody 1N4001 (zobacz sekcję „Tranzystory i diody” w dodatku A),
- zasilacza dostarczającego prąd o napięciu odpowiednim dla posiadanej silnika.

Jeżeli używasz silnika o małej mocy (pobierającego prąd o natężeniu mniejszym niż 200 mA), to możesz zastosować mniejszy (i tańszy) tranzystor (zobacz rysunek 10.6).

Prawdopodobnie będziesz mógł zasilić mały silnik prądem o napięciu 5 V bezpośrednio z gniazda GPIO. Jeżeli Twoje Raspberry Pi zacznie pracować niestabilnie, podłącz silnik do zewnętrznego zasilania, jak pokazano na rysunku 10.5.

Prędkością obrotową silnika możesz sterować za pomocą programu *gui_slider.py* omówionego w recepturze 9.8. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Program ten posiada graficzny interfejs użytkownika, a więc nie możesz z niego korzystać za pośrednictwem protokołu SSH. Musisz go uruchomić bezpośrednio na Raspberry Pi w oknie środowiska graficznego lub zdalnie za pomocą oprogramowania VNC (zobacz receptura 2.8).



Rysunek 10.6. Sterowanie silnikiem o małej mocy

Omówienie

Dokładnie tego samego obwodu używaliśmy do sterowania pracą przekaźnika w recepturze 9.5. Tym razem w miejscu cewki przekaźnika umieściliśmy silnik. Opis działania tego obwodu znajdziesz w recepturze 9.4.

Zobacz również

Przedstawiony obwód może służyć do sterowania prędkością obrotową silnika. Nie można za jego pomocą wybierać kierunku obrotów wrzeciona silnika. Jeżeli chcesz zmieniać kierunek obrotów silnika, zajrzyj do receptury 10.4.

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

10.4. Zmienianie kierunku obrotów silnika zasilanego prądem stałym

Problem

Chcesz sterować zarówno prędkością obrotową, jak i kierunkiem obrotów wrzeciona małego silnika zasilanego prądem stałym.

Rozwiążanie

Zastosuj mostek H w postaci modułu lub układu scalonego. Informacje na temat funkcjonowania tego mostka znajdziesz w sekcji „Omówienie”.

Obwód sterujący silnikiem możesz zbudować na dwa sposoby. Pierwszy sposób polega na samodzielnym zbudowaniu obwodu przy użyciu układu scalonego L293D i płytki prototypowej niewymagającej wykonywania połączeń lutowniczych. Drugi sposób wymaga bezpośredniego podłączenia do Raspberry Pi gotowego modułu mostka H sprzedawanego przez firmę SparkFun.

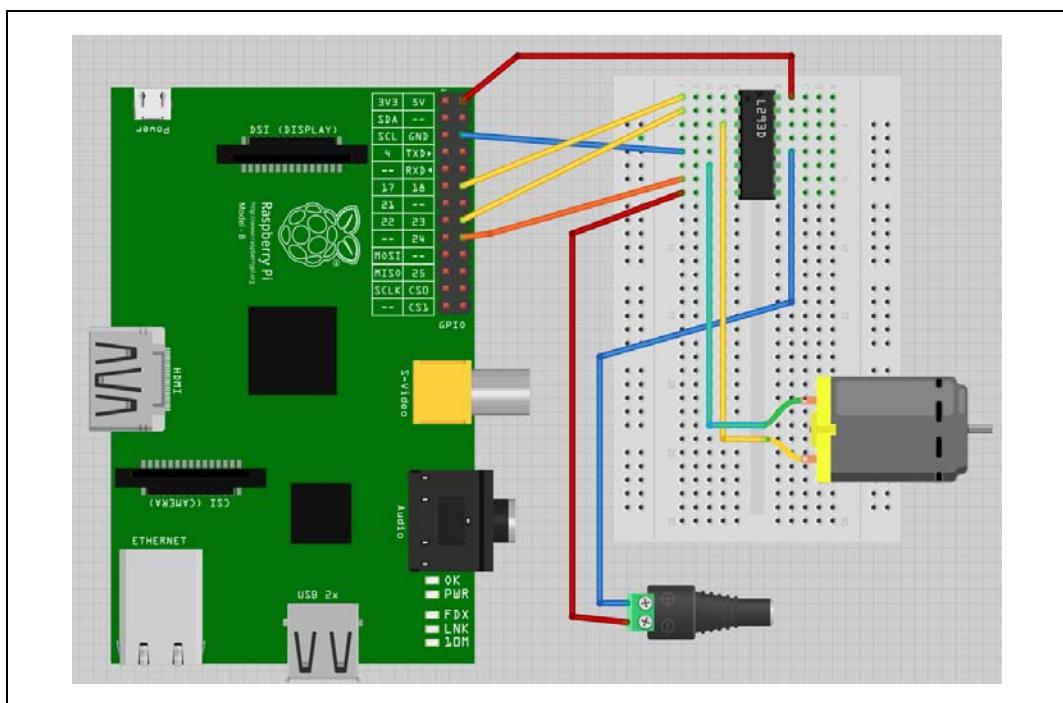
Zarówno gotowy moduł dostarczany przez firmę SparkFun, jak i układ L293D są w stanie samodzielnie sterować pracą dwóch silników.

Opcja 1. Układ L293D i płytka prototypowa

Do zbudowania obwodu opartego na układzie L293D będziesz potrzebować:

- silnika zasilanego prądem stałym o napięciu od 3 do 12 V,
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu L293D (zobacz sekcja „Układy scalone” w dodatku A),
- zasilacza generującego prąd o napięciu odpowiednim dla posiadanej silnika.

Schemat wykonawczy obwodu pokazano na rysunku 10.7.



Rysunek 10.7. Sterowanie silnikiem za pomocą układu scalonego L293D

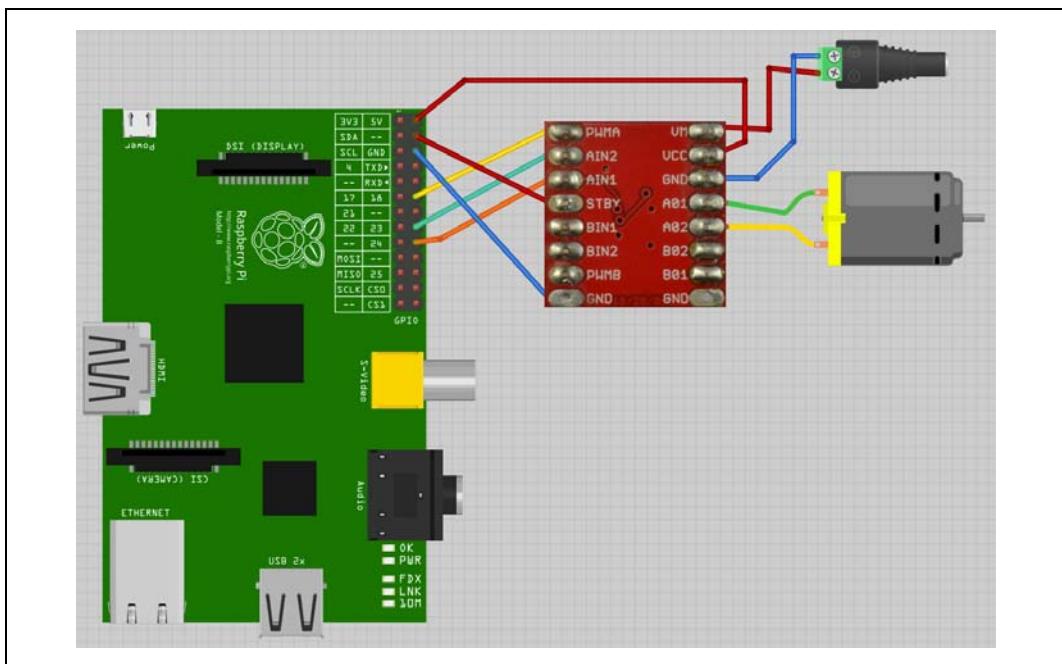
Upewnij się, że nie zainstalowałeś odwrotnie układu scalonego. Na górze obudowy układu znajduje się nacięcie, które powinno być zwrócone w kierunku zewnętrznej krawędzi płytki.

Opcja 2. Gotowy moduł sterownika silnika

Do zbudowania obwodu opartego na gotowym module firmy SparkFun będziesz potrzebować:

- silnika zasilanego prądem stałym o napięciu od 3 do 12 V,
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- listwy goldpinów (zobacz sekcję „Pozostałe” w dodatku A),
- podwójnego sterownika silników (1 A) firmy SparkFun (zobacz sekcję „Moduły” w dodatku A),
- zasilacza generującego prąd o napięciu odpowiednim dla posiadanej silniki.

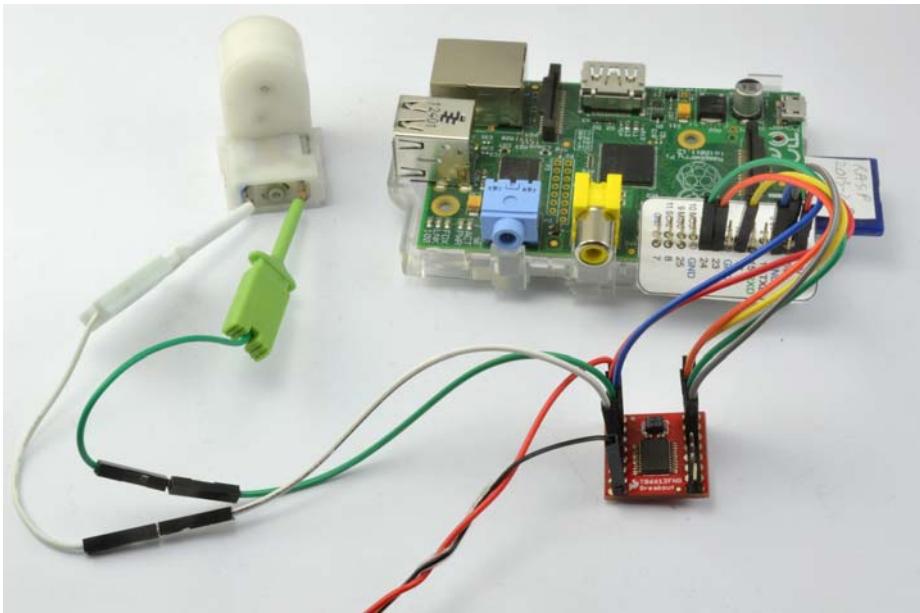
Schemat wykonawczy obwodu pokazano na rysunku 10.8. Na rysunku widoczny jest sam silnik bez przekładni. Zwykle do napędu kół stosuje się **silniki przekładniowe** — silniki elektryczne z wbudowanym reduktorem obniżającym prędkość obrotową wrzeciona i zwiększającym moment obrotowy.



Rysunek 10.8. Podłączanie silnika za pośrednictwem sterownika firmy SparkFun

Moduł kontrolera silników nie jest wyposażony w goldpiny. Przed podłączeniem silnika do kontrolera przylutuj goldpiny do płytki modułu. Połączenia będą wykonywane przewodami połączeniowymi obustronnie zakończonymi końcówkami żeńskimi.

Na rysunku 10.9 pokazano moduł podłączony do małego silnika przekładniowego według schematu znajdującego się na rysunku 10.8.



Rysunek 10.9. Silnik przekładniowy podłączony do sterownika firmy SparkFun

Oprogramowanie

Niezależnie od wybranej opcji działanie silnika możesz sprawdzić za pomocą tego samego programu. Program obsługuje się, wpisując literę *f* lub *r* i pojedynczą cyfrę z zakresu od 0 do 9. Silnik będzie się obracał do przodu (*f*) lub do tyłu (*r*) z prędkością obrotową określoną przez podaną cyfrę (0 oznacza zatrzymanie silnika, a 9 — maksymalną prędkość).

```
$ sudo python motor_control.py  
Polecenie, f/r 0-9, np. f5 :f5  
Polecenie, f/r 0-9, np. f5 :f1  
Polecenie, f/r 0-9, np. f5 :f2  
Polecenie, f/r 0-9, np. f5 :r2
```

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz utworzony plik pod nazwą *motor_control.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Program ten nie posiada graficznego interfejsu użytkownika, a więc możesz z niego korzystać za pośrednictwem protokołu SSH.

Jeżeli korzystasz z Pythona 3, zmień polecenie *raw_input* na *input*.

```
import RPi.GPIO as GPIO  
import time  
  
enable_pin = 18  
in1_pin = 23  
in2_pin = 24  
  
GPIO.setmode(GPIO.BCM)  
GPIO.setup(enable_pin, GPIO.OUT)  
GPIO.setup(in1_pin, GPIO.OUT)
```

```

GPIO.setup(in2_pin, GPIO.OUT)
pwm = GPIO.PWM(enable_pin, 500)
pwm.start(0)

def clockwise():
    GPIO.output(in1_pin, True)
    GPIO.output(in2_pin, False)

def counter_clockwise():
    GPIO.output(in1_pin, False)
    GPIO.output(in2_pin, True)

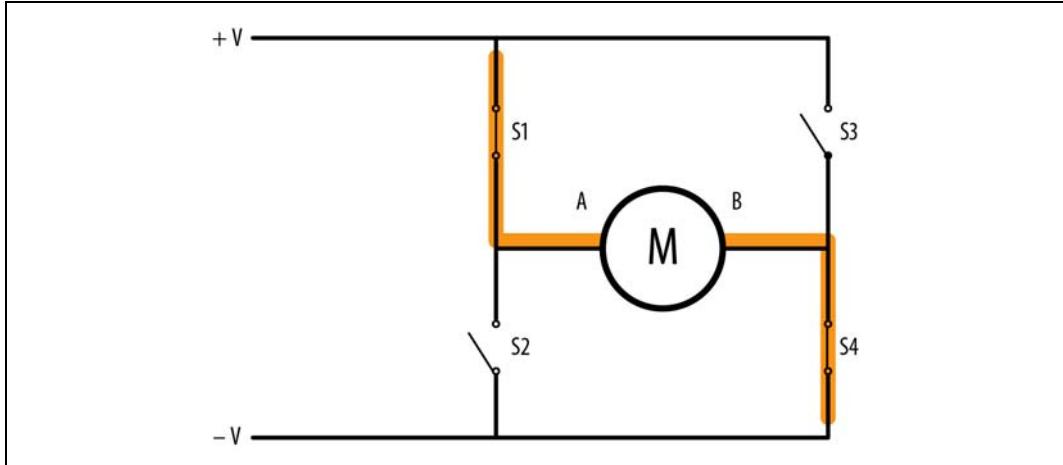
while True:
    cmd = raw_input("Polecenie, f/r 0-9, np. f5 :")
    direction = cmd[0]
    if direction == "f":
        clockwise()
    else:
        counter_clockwise()
    speed = int(cmd[1]) * 10
    pwm.ChangeDutyCycle(speed)

```

Omówienie

Zanim przyjrzymy się działaniu programu, chciałbym przybliżyć Ci funkcjonowanie mostka H.

Na rysunku 10.10 przedstawiono działanie mostka H składającego się z przełączników, a nie tranzystorów lub układów scalonych. Zmiana polaryzacji prądu zasilającego silnik za pomocą mostka H powoduje zmianę kierunku obrotów silnika.



Rysunek 10.10. Mostek H

Na rysunku 10.10 przełączniki S1 i S4 domykają obwód, a przełączniki S2 i S3 są otwarte. Pozwala to na przepływ prądu przez silnik — zacisk A jest wtedy zaciskiem dodatnim, a zacisk B — ujemnym.

Gdybyśmy zwarły przełączniki S2 i S3, a otworzyli S1 i S4, to zacisk B byłby zaciskiem dodatnim, a A — ujemnym. Silnik obracałby się w przeciwnym kierunku.

Mogłeś już zauważyc pewne niebezpieczeństwo związane z działaniem tego układu. Gdyby przełączniki S1 i S2 jednocześnie domykały obwód, to doszłoby do zwarcia. Taka sama sytuacja miałaby miejsce, gdyby przełączniki S3 i S4 jednocześnie domykały obwód.

Możesz zbudować mostek H z pojedynczych tranzystorów, jednak łatwiej zastosować gotowy mostek w postaci układu scalonego takiego jak L293D. Układ ten tak naprawdę składa się z dwóch mostków H, a więc można nim sterować dwoma silnikami.

Układ L293D posiada trzy złącza sterujące dla każdego z dwóch silników. Styk *Enable* służy do włączenia całego kanału. W zaprezentowanym przykładzie jest on podłączony do wyjścia układu PWM sterującego prędkością obrotową silnika. Pozostałe dwa styki (*IN1* i *IN2*) odpowiadają za kierunek obrotów silnika. Funkcje *clockwise* i *counter_clockwise* korzystają właśnie z tych styków:

```
def clockwise():
    GPIO.output(in1_pin, True)
    GPIO.output(in2_pin, False)

def counter_clockwise():
    GPIO.output(in1_pin, False)
    GPIO.output(in2_pin, True)
```

Jeżeli na złączu *IN1* podawany jest sygnał wysoki, a na złączu *IN2* niski, to silnik będzie się obracał w pewnym kierunku. Jeżeli te dwa sygnały zostaną odwrócone, silnik będzie się obracał w przeciwnym kierunku.

Zobacz również

W recepturze 10.8 przedstawiono zastosowanie układu L293D zamontowanego na płytce RaspiRobot.

Jeżeli chcesz sterować tylko prędkością jednego silnika, to nie musisz korzystać z układu L293D. Wystarczy, że zbudujesz układ oparty na jednym tranzystorze (zobacz receptura 10.3).

Zajrzyj do dokumentacji układu L293D znajdującej się pod adresem <http://www.ti.com/lit/ds/symlink/l293d.pdf>. Odwiedź stronę modułu sterownika silników firmy SparkFun: <https://www.sparkfun.com/products/9457>.

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

10.5. Używanie unipolarnych silników krokowych

Problem

Chcesz sterować pracą unipolarnego silnika krokowego posiadającego pięciostykowe wyprowadzenie.

Rozwiązanie

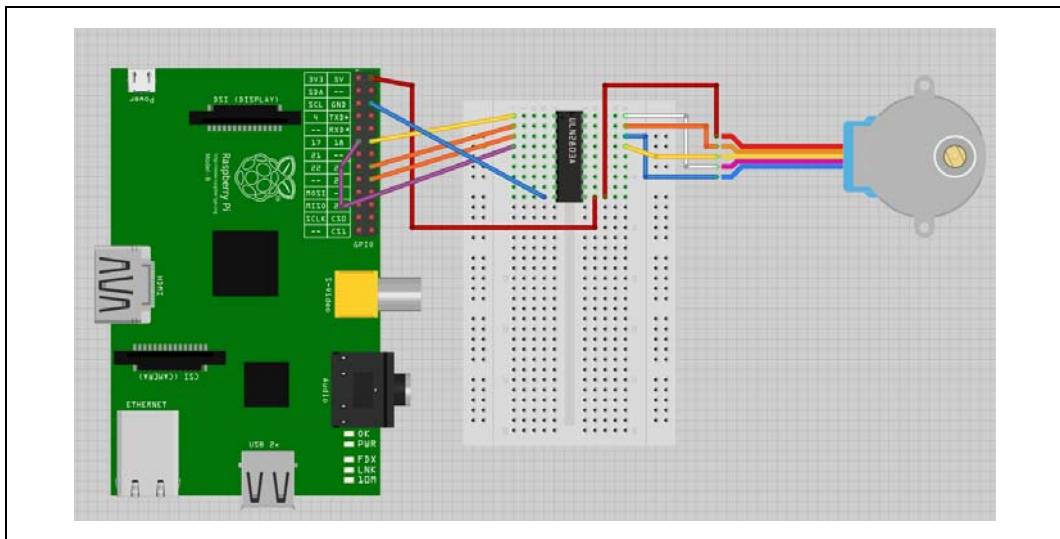
Skorzystaj z układu sterownika Darlingtona ULN2803.

Silnik krokowy to z technologicznego punktu widzenia coś pomiędzy zwyczajnym silnikiem zasilanym prądem stałym a serwomotorem. Mogą się one obracać w kółko, tak jak zwykłe silniki, ale pozycję, w której są ustawione, można precyzyjnie określić — takimi silnikami można obracać w dowolnym kierunku o określoną liczbę kroków w danej jednostce czasu.

W celu zbudowania obwodu sterującego pracą takiego silnika będziesz potrzebować:

- unipolarnego silnika krokowego mającego pięć złączy, zasilanego prądem o napięciu 5 V (zobacz sekcja „Pozostałe” w dodatku A),
- sterownika Darlingtona ULN2803 (zobacz sekcja „Układy scalone” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A).

Na rysunku 10.11 przedstawiono schemat wykonawczy tego obwodu. Układ ULN2803 może być użyty do sterowania dwoma silnikami krokowymi. Aby sterować drugim silnikiem krokowym, musiałbyś podłączyć cztery kolejne styki sterujące złącza GPIO do układu ULN2803 (pinów od 5. do 8.). Do pinów 11. – 14. tego układu scalonego należałyby wtedy podłączyć cztery przewody biegnące od drugiego silnika.



Rysunek 10.11. Układ ULN2803 sterujący pracą silnika krokowego

Małe silniki krokowe można zasilać prądem o napięciu 5 V bezpośrednio ze złącza GPIO. Jeżeli Raspberry Pi podczas używania silnika działa niestabilnie lub Twój silnik jest duży, to podłącz zewnętrzny zasilacz do układu ULN2803 do nóżki oznaczonej numerem 10.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *stepper.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Program ten nie posiada graficznego interfejsu użytkownika, a więc możesz z niego korzystać za pośrednictwem protokołu SSH.

Jeżeli korzystasz z Pythona 3, zmień polecenie `raw_input` na `input`:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

coil_A_1_pin = 18
coil_A_2_pin = 23
coil_B_1_pin = 24
coil_B_2_pin = 17

GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)

forward_seq = ['1010', '0110', '0101', '1001']
reverse_seq = list(forward_seq) # w celu skopiowania listy
reverse_seq.reverse()

def forward(delay, steps):
    for i in range(steps):
        for step in forward_seq:
            set_step(step)
            time.sleep(delay)

def backwards(delay, steps):
    for i in range(steps):
        for step in reverse_seq:
            set_step(step)
            time.sleep(delay)

def set_step(step):
    GPIO.output(coil_A_1_pin, step[0] == '1')
    GPIO.output(coil_A_2_pin, step[1] == '1')
    GPIO.output(coil_B_1_pin, step[2] == '1')
    GPIO.output(coil_B_2_pin, step[3] == '1')

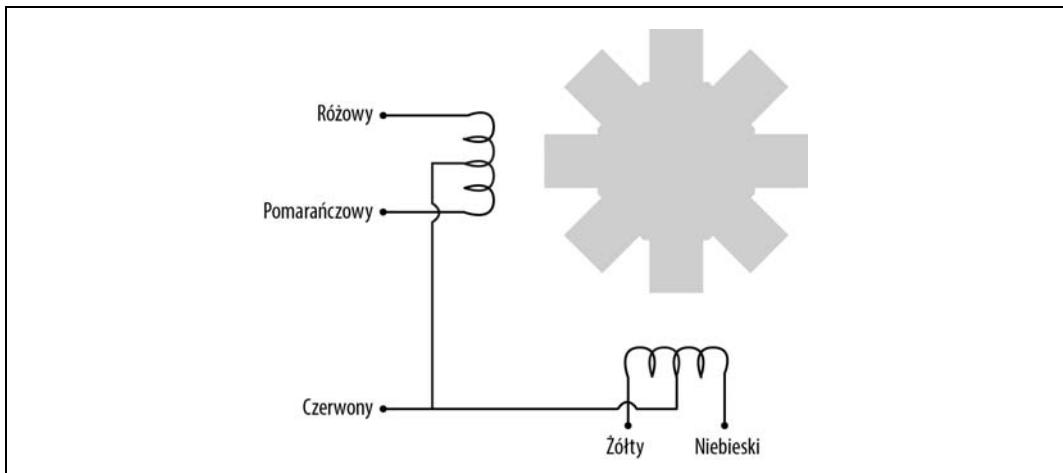
while True:
    set_step('0000')
    delay = raw_input("Jaki odstęp pomiędzy krokami(milisekundy)? ")
    steps = raw_input("Ile kroków do przodu? ")
    forward(int(delay) / 1000.0, int(steps))
    set_step('0000')
    steps = raw_input("Ile kroków do tyłu? ")
    backwards(int(delay) / 1000.0, int(steps))
```

Po uruchomieniu programu zostaniesz poproszony o podanie odstępu czasu pomiędzy krokami silnika. Odstęp ten powinien wynosić co najmniej dwie milisekundy. Później zostaniesz poproszony o podanie liczby kroków, jakie ma wykonać silnik w każdym z kierunków:

```
$ sudo python stepper.py
Jaki odstęp pomiędzy krokami(milisekundy)? 2
Ile kroków do przodu? 100
Ile kroków do tyłu? 100
Jaki odstęp pomiędzy krokami(milisekundy)? 10
Ile kroków do przodu? 50
Ile kroków do tyłu? 50
Jaki odstęp pomiędzy krokami(milisekundy)? ?
```

Omówienie

W silniku krokowym znajduje się wirnik z zębatką i poruszający go elektromagnes. Silnik jednorazowo może wykonać ruch o jeden krok (rysunek 10.12). Kolory przewodów Twojego silnika mogą się różnić od kolorów przewodów silnika zastosowanego przeze mnie.



Rysunek 10.12. Silnik krokowy

Cewki zasilane w odpowiedniej kolejności obracają wirnikiem silnika. Liczba kroków, jakie silnik wykonuje podczas obrotu wokół własnej osi, zależy od liczby zębów znajdujących się na wirniku.

W przytoczonym programie każdy z czterech etapów zasilania składających się na krok silnika jest reprezentowanyłańcuchem:

```
forward_seq = ['1010', '0110', '0101', '1001']
```

Odwracając sekwencję wywołującą ruch silnika w przód, otrzymamy sekwencję wywołującą ruch silnika w tył.

W przedstawionym programie funkcja `forward` służy do wykonania kroku silnika w przód, a funkcja `backward` w tył. Pierwszym argumentem przekazywanym do tych funkcji jest odstęp czasu (podany w milisekundach) pomiędzy kolejnymi etapami sekwencji wykonywania kroku. Minimalna ilość czasu zależy od silnika, z którego korzystasz. Podanie zbyt krótkiego czasu sprawi, że silnik nie wykona kroku. Omawiany odstęp powinien (dla większości silników) wynosić przynajmniej 2 milisekundy. Drugim parametrem przekazywanym do tych funkcji jest liczba kroków, które chcemy, aby silnik wykonał.

```
def forward(delay, steps):
    for i in range(steps):
        for step in forward_seq:
            set_step(step)
            time.sleep(delay)
```

Funkcja `forward` ma dwie zagnieżdżone pętle `for`. Zewnętrzna pętla jest wykonywana tyle razy, ile kroków ma wykonać silnik. Wewnętrzna pętla iteruje sekwencję ruchu silnika, wywołując funkcję `set_step`.

```
def set_step(step):
    GPIO.output(coil_A_1_pin, step[0] == '1')
    GPIO.output(coil_A_2_pin, step[1] == '1')
    GPIO.output(coil_B_1_pin, step[2] == '1')
    GPIO.output(coil_B_2_pin, step[3] == '1')
```

Funkcja `set_step` ustawia sygnał generowany na pinach sterujących silnikiem zgodnie z wzorem przekazanym do funkcji w postaci argumentu.

Główna funkcja programu pomiędzy wykonywaniem ruchów silnikiem w przód i w tył ustawia parametr kroku na 0000. Ma to miejsce, gdy silnik nie wykonuje żadnego ruchu. Przeciwdziała to pozostawieniu którejś z cewek silnika pod napięciem, a w efekcie zbędнемu pobieraniu prądu przez silnik.

Zobacz również

Jeżeli posiadasz bipolarny silnik krokowy z czterema przewodami wyjściowymi, zajrzyj do receptury 10.6.

Więcej informacji o silnikach krokowych — ich typach i sposobie działania — znajdziesz w Wikipedii: http://pl.wikipedia.org/wiki/Silnik_krokowy. Umieszczono tam animację ilustrującą działanie silników tego typu.

Więcej informacji dotyczących użytkowania silników krokowych znajdziesz w recepturze 10.1. Informacje na temat zwykłych silników zasilanych prądem stałym znajdziesz w recepturach 10.3 i 10.4.

Więcej informacji na temat korzystania z płytka prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

10.6. Korzystanie z bipolarnych silników krokowych

Problem

Chcesz sterować pracą bipolarnego silnika krokowego o czterech przewodach wyjściowych.

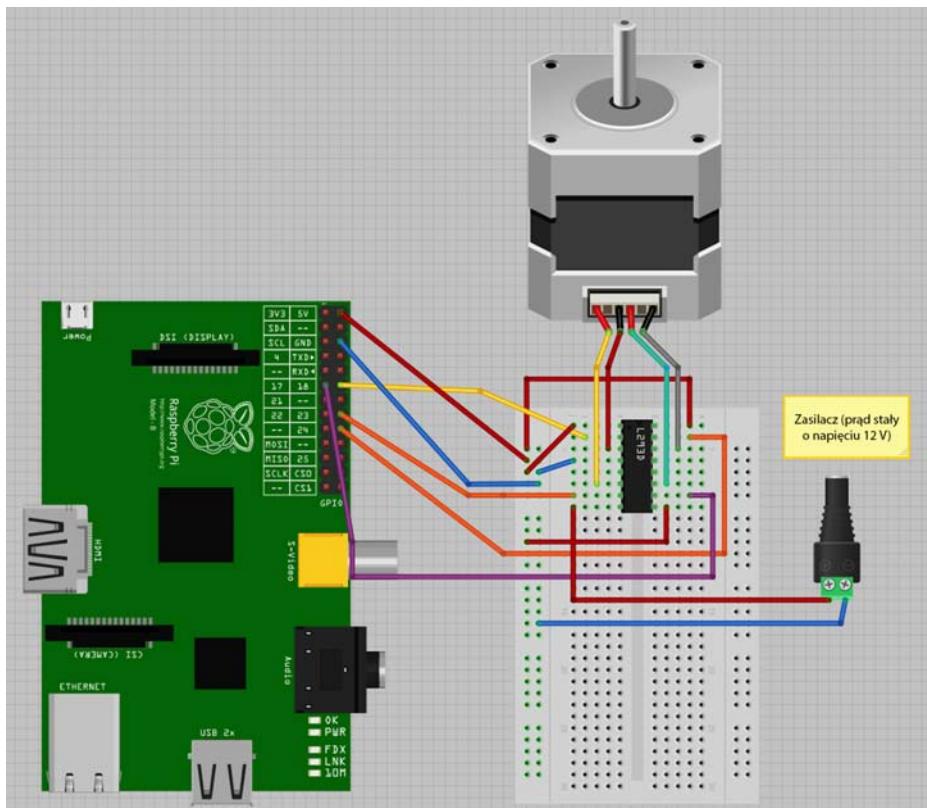
Rozwiążanie

Zastosuj sterujący mostek H w postaci układu scalonego L293D. Aby sterować pracą bipolarnego silnika krokowego, niezbędny jest mostek H, ponieważ — jak sugeruje słowo **bipolarne** — kierunek przepływu prądu przez uzwojenia silnika musi być odwracany. Z podobną sytuacją mieliśmy do czynienia w recepturze 10.4, omawiając zagadnienia związane ze sterowaniem zwyczajnymi silnikami zasilanymi prądem stałym.

W celu zbudowania obwodu bipolarnego silnika krokowego będziesz potrzebować:

- bipolarnego silnika krokowego mającego cztery złącza, zasilanego prądem o napięciu 12 V (zobacz sekcja „Pozostałe” w dodatku A),
- układu scalonego L293D (zobacz sekcja „Układy scalone” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny” podczas wykonywania prototypów” w dodatku A).

Zastosowany tutaj silnik zasilany prądem o napięciu 12 V jest nieco większy od omówionego wcześniej unipolarnego silnika krokowego. Nie należy go zasilać za pośrednictwem płytki Raspberry Pi. W obwodzie przedstawionym na rysunku 10.13 zastosowano zewnętrzny zasilacz.



Rysunek 10.13. Układ L293D sterujący pracę bipolarnego silnika krokowego

Omówienie

W celu sterowania bipolarnym silnikiem krokowym możesz zastosować program *stepper.py* omówiony wcześniej w recepturze 10.5. Obwód korzysta z obu mostków H znajdujących się w układzie L293D. Sterowanie dwoma silnikami wymaga zastosowania dwóch układów scalonych tego typu.

Zobacz również

Jeżeli posiadasz unipolarny silnik krokowy z pięcioma przewodami wyjściowymi, zajrzyj do receptury 10.5.

Więcej informacji o silnikach krokowych — ich typach i sposobie działania — znajdziesz w Wikipedii: http://pl.wikipedia.org/wiki/Silnik_krokowy. Umieszczono tam animację ilustrującą działanie silników tego typu.

Więcej informacji dotyczących użytkowania silników krokowych znajdziesz w recepturze 10.1. Informacje na temat zwykłych silników zasilanych prądem stałym znajdziesz w recepturach 10.3 i 10.4.

Pracę silnika krokowego można również sterować za pośrednictwem płytka RaspiRobot (zobacz receptura 10.7).

Więcej informacji na temat korzystania z płytki prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

10.7. Sterowanie pracą bipolarnego silnika krokowego za pośrednictwem płytki RaspiRobot

Problem

Chcesz sterować pracą bipolarnego silnika krokowego za pośrednictwem płytki RaspiRobot.

Rozwiązanie

Płytki RaspiRobot korzysta z tego samego układu scalonego L293D składającego się z dwóch mostków H, który był stosowany w recepturze 10.6.

Płytki RaspiRobot dostarcza prąd zarówno do silnika krokowego, jak i Raspberry Pi. Wbudowany regulator obniża napięcie do 5 V, zasilając obwody Raspberry Pi. A zatem Raspberry Pi oraz silnik krokowy można zasilać za pomocą tego samego zasilacza zapewniającego prąd o napięciu 12 V.



Nie zasilaj Raspberry Pi za pośrednictwem gniazda USB, gdy do płytka RaspiRobot podłączony jest zasilacz. Niewielkie różnice potencjałów pomiędzy napięciem dostarczonym przez płytka RaspiRobot a napięciem dostarczonym do gniazda USB mogą doprowadzić do przepływu zbyt dużych prądów, co prawdopodobnie spowoduje uszkodzenie Raspberry Pi lub płytka RaspiRobot. Nie podłączaj jednocześnie zasilania do obu płytka.

Do płytka RaspiRobot podłącz silnik krokowy i zasilacz, tak jak pokazano na rysunku 10.14. Jeżeli korzystasz z silnika krokowego firmy Adafruit zasilanego prądem o napięciu 12 V, to zaczynając od strony gniazda zasilania, podłącz kolejno przewody: żółty, czerwony, szary i zielony.

Możesz ponownie zastosować program opisany w recepturze 10.6. Wystarczy, że zmodyfikujesz sekwencję, według której wykonywany jest krok, oraz numery używanych pinów gniazda GPIO.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `stepper_rrb.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspire.htm>. Program ten nie posiada graficznego interfejsu użytkownika, a więc możesz z niego korzystać za pośrednictwem protokołu SSH.



Rysunek 10.14. Sterowanie bipolarnym silnikiem krokowym za pośrednictwem płytki RaspiRobot

Jeżeli korzystasz z Pythona 3, zmień polecenie raw_input na input:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

coil_A_1_pin = 17
coil_A_2_pin = 4
coil_B_1_pin = 10
coil_B_2_pin = 25

GPIO.setup(coil_A_1_pin, GPIO.OUT)
GPIO.setup(coil_A_2_pin, GPIO.OUT)
GPIO.setup(coil_B_1_pin, GPIO.OUT)
GPIO.setup(coil_B_2_pin, GPIO.OUT)

forward_seq = ['1011', '1111', '1110', '1010']
reverse_seq = list(forward_seq) # w celu skopiowania listy
reverse_seq.reverse()

def forward(delay, steps):
    for i in range(steps):
        for step in forward_seq:
            set_step(step)
            time.sleep(delay)

def backwards(delay, steps):
    for i in range(steps):
        for step in reverse_seq:
            set_step(step)
            time.sleep(delay)

def set_step(step):
    GPIO.output(coil_A_1_pin, step[0] == '1')
    GPIO.output(coil_A_2_pin, step[1] == '1')
    GPIO.output(coil_B_1_pin, step[2] == '1')
    GPIO.output(coil_B_2_pin, step[3] == '1')
```

```
while True:  
    set_step('0000')  
    delay = raw_input("Jaki odstęp pomiędzy krokami(milisekundy)?")  
    steps = raw_input("Ile kroków do przodu? ")  
    forward(int(delay) / 1000.0, int(steps))  
    set_step('0000')  
    steps = raw_input("Ile kroków do tyłu? ")  
    backwards(int(delay) / 1000.0, int(steps))
```

Omówienie

Moduł RaspiRobot korzysta z układu L293D w inny sposób niż układ przedstawiony w recepturze 10.4. Korzysta on z pinów o numerach 17 i 10 w celu uruchomienia kanału i z pinów o numerach 4 i 25 w celu określenia kierunku obrotu każdego z silników. W związku z tym musisz zmodyfikować alokację styków gniazda GPIO oraz sekwencję wywołującą obrót silnika o krok:

```
forward_seq = ['1011', '1111', '1110', '1010']
```

Pierwsza i trzecia cyfra każdej sekwencji są zawsze cyfrą 1 — oba silniki są aktywowane. Polaryzacja każdego z dwóch uzożeń jest teraz określana przez drugi i czwarty bit.

Zobacz również

Więcej informacji o płytce RaspiRobot, a także innych projektach z niej korzystających znajdziesz na stronie <https://github.com/simonmonk/raspirobotboard2>.

Jeżeli chcesz sterować silnikiem krokom na pomocą układu L293D zamontowanego na płytce prototypowej, to zajrzyj do receptury 10.6.

10.8. Budowa prostego jeżdżącego robota

Problem

Chcesz sterować za pomocą Raspberry Pi prostym poruszającym się robotem.

Rozwiązanie

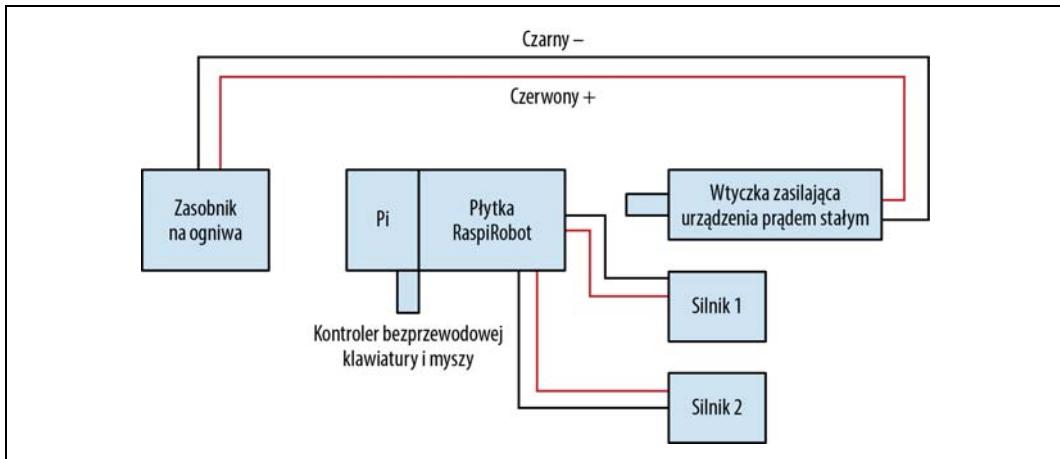
Zastosuj płytę RaspiRobot podłączoną do Raspberry Pi w roli interfejsu sterującego pracą dwóch silników zamontowanych na podstawie przeznaczonej do konstrukcji robotów (np. na podstawie Magician Chassis).

W celu zbudowania robota będziesz potrzebować:

- płytki RaspiRobot (zobacz sekcja „Moduły” w dodatku A),
- płyty podwoziowej przeznaczonej do konstrukcji robotów (Magician Chassis) oraz dwóch silników przekładniowych (zobacz sekcja „Pozostałe” w dodatku A),
- bezprzewodowej klawiatury i myszy.

Pracę nad robotem zacznij od przygotowania płyty podwoziowej Magician Chassis. Płyta jest wyposażona w zasobnik na cztery ogniwka AA. Aby zasilić Raspberry Pi, będziesz potrzebować zasobnika na sześć ogniw. A zatem wykonując polecenia zawarte w instrukcji płyty podwoziowej, pomiń punkt dotyczący montażu zasobnika baterii.

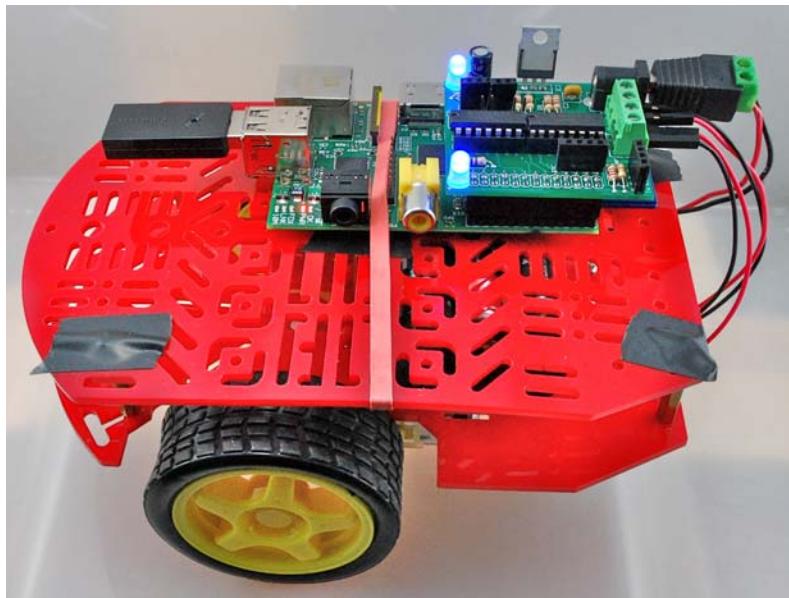
Możesz się też spotkać z płytą RaspiRobot sprzedawaną jako zestaw do samodzielnego montażu. Prace lutownicze przeprowadź zgodnie z dołączoną do niej instrukcją, a następnie wykonaj obwód pokazany na rysunku 10.15.



Rysunek 10.15. Schemat połączeń pomiędzy głównymi elementami robota

Bateria jest podłączona do płytki RaspiRobot, która dostarcza prąd o napięciu 5 V do Raspberry Pi, a więc wystarczy nam tylko jedno źródło zasilania.

Ukończony robot został pokazany na rysunku 10.16.



Rysunek 10.16. Ukończony robot

Robot będzie kontrolowany za pośrednictwem programu pozwalającego na sterowanie za pomocą klawiszy strzałek (w górę i w dół) klawiatury bezprzewodowej połączonej z Raspberry Pi. Program ten wymaga zainstalowania biblioteki dedykowanej płytce RaspiRobot.

Biblioteka RaspiRobot wymaga wcześniejszego zainstalowania dwóch innych bibliotek: RPi.GPIO (zobacz receptura 8.3) i PySerial (zobacz receptura 8.8). W oknie Terminala uruchom następujące polecenia:

```
$ sudo apt-get install python-rpi.gpio  
$ sudo apt-get install python-serial
```

Po zainstalowaniu wspomnianych dwóch bibliotek w oknie Terminala uruchom kolejne polecenia:

```
$ wget https://github.com/simonmonk/raspirobotboard/archive/master.zip  
$ unzip master.zip  
$ cd raspirobotboard-master  
$ sudo python setup.py install
```

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *rover.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Program przechwytuje wciśnięcia klawiszy za pomocą biblioteki PyGame. Wymaga ona zastosowania graficznego interfejsu użytkownika nawet wtedy, gdy nie będziemy mieli ekranu zamontowanego na robocie, a więc nie będziemy widzieli tego interfejsu. W związku z tym nie możesz uruchomić tego programu za pośrednictwem protokołu SSH. Musisz go uruchomić w sesji VNC (zobacz receptura 2.8) lub skonfigurować Raspberry Pi tak, aby program był uruchamiany automatycznie podczas startu systemu operacyjnego (zobacz receptura 3.20).

```
from raspirobotboard import *  
import pygame  
from pygame.locals import *  
  
rr = RaspiRobot()  
  
pygame.init()  
screen = pygame.display.set_mode((640, 480))  
  
pygame.display.set_caption('RaspiRobot')  
pygame.mouse.set_visible(0)  
  
...  
  
while True:  
    for event in pygame.event.get():  
  
        if event.type == QUIT:  
            sys.exit()  
        if event.type == KEYDOWN:  
            if event.key == K_UP:  
                rr.forward()  
                rr.set_led1(True)  
                rr.set_led2(True)  
            elif event.key == K_DOWN:  
                rr.set_led1(True)  
                rr.set_led2(True)  
                rr.reverse()  
            elif event.key == K_RIGHT:  
                rr.set_led1(False)  
                rr.set_led2(True)
```

```
    rr.right()
elif event.key == K_LEFT:
    rr.set_led1(True)
    rr.set_led2(False)
    rr.left()
elif event.key == K_SPACE:
    rr.stop()
    rr.set_led1(False)
    rr.set_led2(False)
```

Omówienie

Możesz rozbudować swojego robota, instalując na nim urządzenia peryferyjne, takie jak np. kamera internetowa strumieniująca obraz do sieci lokalnej lub internetu (zobacz receptura 4.5). W ten sposób uzyskasz ciekawy gadżet szpiegowski.

Możesz sterować swoim robotem również na inne sposoby. Wystarczy, że podłączysz do Raspberry Pi kontroler sieci bezprzewodowej Wi-Fi i uruchomisz serwer sieci Web, który pozwoli na sterowanie robotem za pośrednictwem przycisków wyświetlanym w przeglądarce internetowej Twojego komputera (zobacz receptura 9.13). Robotem możesz sterować przy użyciu łącza bezprzewodowego zegarkiem firmy Chronos.Więcej informacji na ten temat znajdziesz na stronie <https://github.com/simonmonk/raspirobotboard/wiki/Tutorial-03-Chronos-Watch-Controlled-Rover>.

Zobacz również

Więcej informacji o płytce RaspiRobot, a także o innych projektach, w których ją zastosowano, znajdziesz na stronie <http://www.raspirobot.com/>.

Cyfrowe wejścia

11.0. Wprowadzenie

W tym rozdziale znajdziesz receptury dotyczące podłączania do cyfrowych wejść Raspberry Pi elementów takich jak przełączniki, bloki klawiszy, a także moduły generujące cyfrowy sygnał wyjściowy.

Obwody przedstawione w większości receptur są zbudowane na bazie płytka prototypowej (niewymagającej wykonywania połączeń lutowniczych) oraz przewodów połączeniowych obustronnie zakończonych końcówkami żeńskimi (zobacz receptura 8.10).

11.1. Podłączanie przełącznika chwilowego

Problem

Chcesz podłączyć do Raspberry Pi przycisk, a następnie za jego pomocą uruchamiać kod programu napisanego w Pythonie.

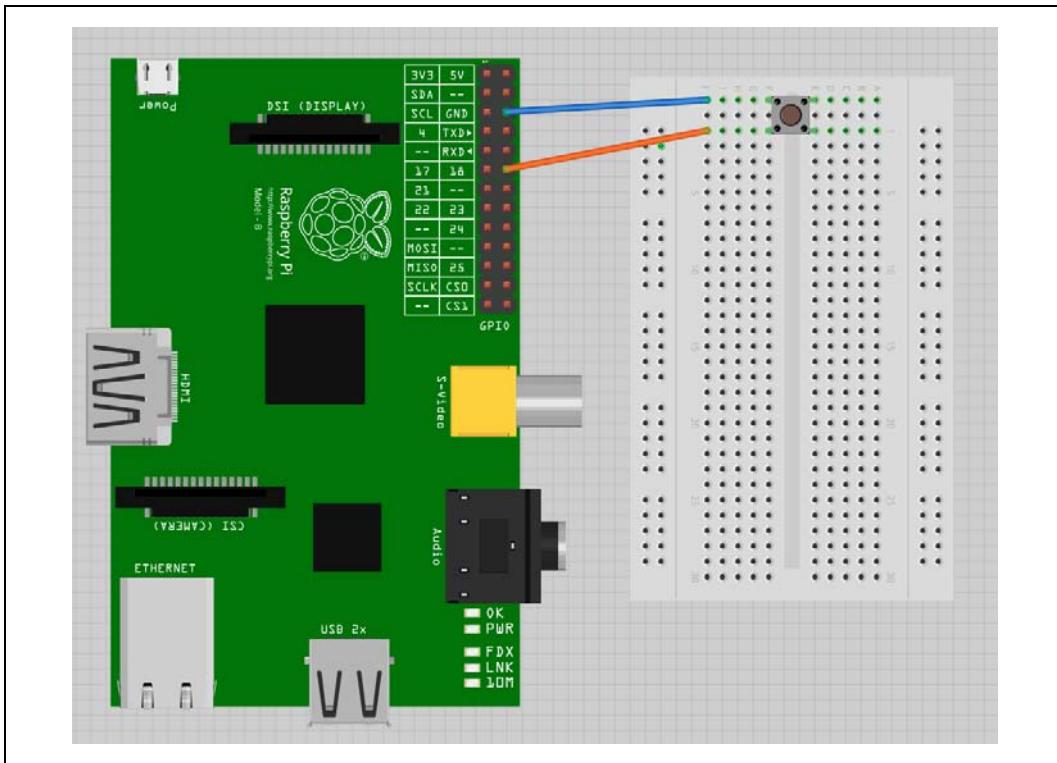
Rozwiązanie

Podłącz przycisk do styków złącza GPIO. W celu wykrycia przez program wciśnięcia przycisku zastosuj bibliotekę `RPi.GPIO`.

Do podłączenia przełącznika do Raspberry Pi będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- przełącznika chwilowego (zobacz sekcja „Pozostałe” w dodatku A).

Na rysunku 11.1 przedstawiono schemat wykonawczy obwodu składającego się z przełącznika, płytka prototypowej i przewodów połączeniowych.



Rysunek 11.1. Przelącznik podłączony do Raspberry Pi

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *switch.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Program wyświetla komunikat, gdy przycisk zostanie wciśnięty.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

GPIO.setup(18, GPIO.IN, pull_up_down=GPIO.PUD_UP)

while True:
    input_state = GPIO.input(18)
    if input_state == False:
        print('Wciszasz przycisk')
        time.sleep(0.2)
```

Program należy uruchomić jako użytkownik posiadający uprawnienia administratora:

```
pi@raspberrypi ~ $ sudo python switch.py
Wciszasz przycisk
Wciszasz przycisk
Wciszasz przycisk
```

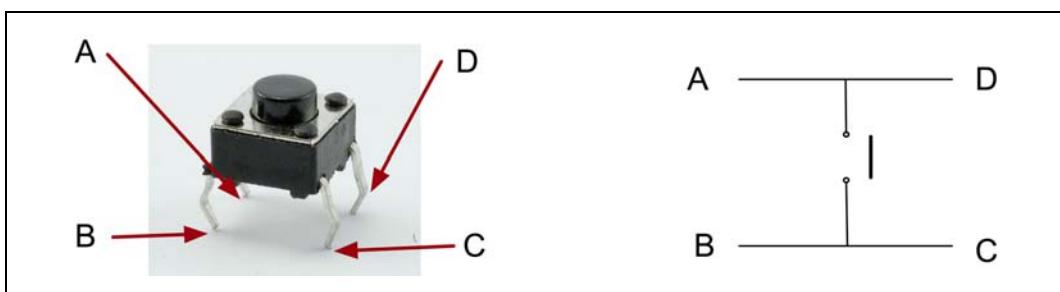
Omówienie

Przycisk jest podłączony tak, że po jego wcisnięciu 18. pin złącza GPIO działający jako wejście jest zwierany z masą. Napięcie na styku wejściowym jest zwykle podwyższane do 3,3 V za pomocą dodatkowego argumentu `pull_up_down=GPIO.PUD_UP` funkcji `GPIO.setup`. Oznacza to, że przy odczycie wartości wejścia funkcja `GPIO.input` zwróci fałsz (`False`), gdy przycisk będzie wcisnięty, co może się wydawać nieintuicyjne.

Każdy styk złącza GPIO wyposażono w programowe rezystory: podciągający i ściągający. Gdy pin złącza GPIO działa w charakterze wejścia, można je aktywować za pomocą dodatkowego parametru `pull_up_down` przekazanego do funkcji `GPIO.setup`. Jeżeli parametr ten zostanie pominięty, to żaden z rezystorów nie zostanie uruchomiony, a wejście będzie miało charakter **bezpotencjałowy** — sygnał odbierany przez takie złącze będzie oscylował pomiędzy stanem wysokim i niskim, w zależności od zakłóceń.

Polecenie `GPIO.PUD_UP` uruchamia rezystor podciągający, a polecenie `GPIO.PUD_DOWN` — rezystor ściągający.

Wydawałoby się, że przełącznik chwilowy powinien posiadać dwa złącza. Niektóre takie przełączniki wyposażono w dwa złącza, jednak większość z nich ma ich cztery. Budowę takich przełączników pokazano na rysunku 11.2.



Rysunek 11.2. Budowa przełącznika chwilowego

Tak naprawdę z elektrycznego punktu widzenia taki przełącznik ma tylko dwa złącza. Jak widać na rysunku, wyprowadzenia B i C są ze sobą zwarte, tak samo jak wyprowadzenia A i D.

Zobacz również

Więcej informacji na temat korzystania z płytka prototypowej oraz przewodów połączeniowych znajdziesz w recepturze 8.10.

W recepturze 9.12 opisano wywoływanie przerwania za pomocą przełącznika.

Usuwanie stuków przełącznika omówiono w recepturze 11.5.

Jeżeli chcesz korzystać z zewnętrznych rezystorów podciągających lub ściągających, zajrzyj do receptury 11.6.

11.2. Korzystanie z przełącznika chwilowego

Problem

Chcesz, żeby każde wcisnięcie przełącznika powodowało naprzemiennie włączenie jakiejś funkcji i jej wyłączenie.

Rozwiązanie

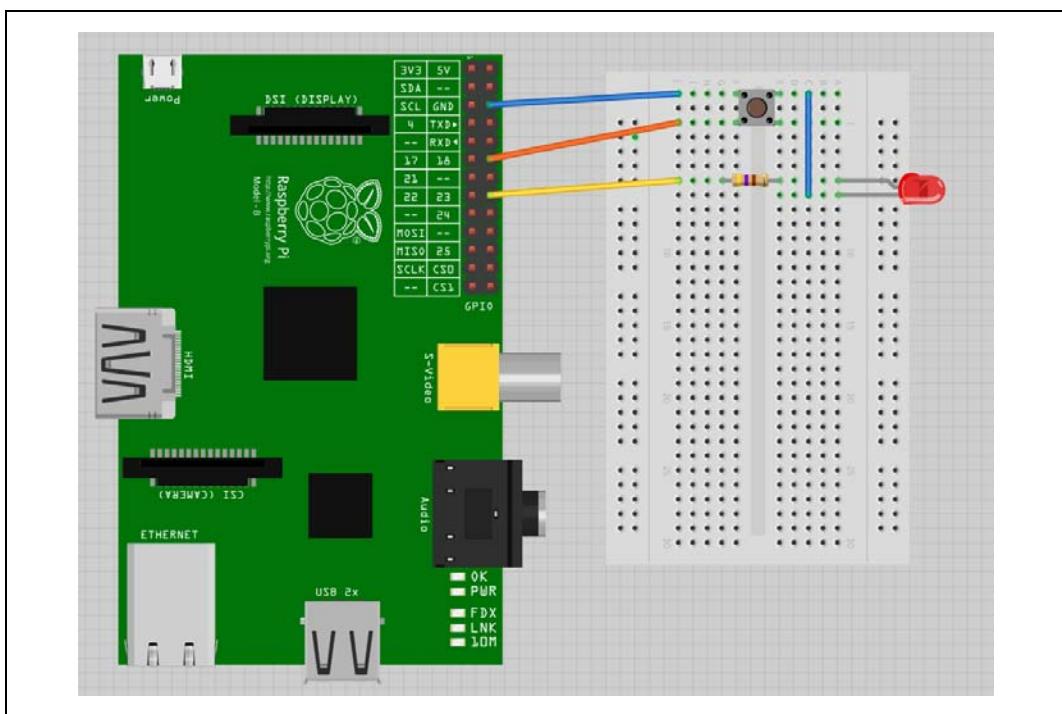
Po każdym wcisnięciu przycisku zapisz jego *stan*, a następnie go odwróć.

W tej recepturze zajmiemy się programem, w którym wciskanie przycisku będzie powodowało włączenie lub wyłączenie diody LED.

Do zbudowania obwodu, którym będzie sterował program, będziesz potrzebować:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- przełącznika chwilowego (zobacz sekcję „Pozostałe” w dodatku A),
- diody LED (zobacz sekcję „Optoelektronika” w dodatku A),
- rezystora $470\ \Omega$ (zobacz sekcję „Rezystory i kondensatory” w dodatku A).

Schemat podłączenia przełącznika i diody LED do Raspberry Pi za pomocą płytka prototypowej i przewodów połączeniowych pokazano na rysunku 11.3.



Rysunek 11.3. Przełącznik oraz dioda LED podłączone do Raspberry Pi

Poza przewodami połączeniowymi zakończonymi obustronnie końówkami żeńskimi (służącymi do połączenia płytki prototypowej i Raspberry Pi) będziesz potrzebować jednego przewodu połączeniowego zakońzonego obustronnie końówkami męskimi lub po prostu drutu.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *switch_on_off.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

switch_pin = 18
led_pin = 23

GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led_pin, GPIO.OUT)

led_state = False
old_input_state = True #podciagnięty

while True:
    new_input_state = GPIO.input(switch_pin)
    if new_input_state == False and old_input_state == True:
        led_state = not led_state
    old_input_state = new_input_state
    GPIO.output(led_pin, led_state)
```

Omówienie

Informacja dotycząca tego, czy dioda LED jest włączona, jest przechowywana przez zmienną `led_state` (wartość `True` oznacza włączoną diodę, a wartość `False` — wyłączoną). Każdorazowe wcisnięcie przycisku powoduje uruchomienie następującej linii kodu:

```
led_state = not led_state
```

Polecenie `not` zmienia wartość zmiennej `led_state` na odwrotną, a więc jeżeli zmienna ta ma wartość `True`, to zostanie ona zmieniona na `False` — i na odwrót.

Zmienna `old_input_state` jest używana do przechowywania pozycji przycisku — wcisnięcie przycisku jest zdefiniowane jako zmiana danych wejściowych z `True` (przełącznik nie jest przyciskany) na `False` (przełącznik jest przyciskany).

Zobacz również

Zapewne czasami spotkasz się z sytuacją, w której wcisnięcie przycisku nie zapali diody LED. Przyczyną tego są stuki generowane przez przełącznik. Techniki niwelowania stuków przełącznika opisano w recepturze 11.5.

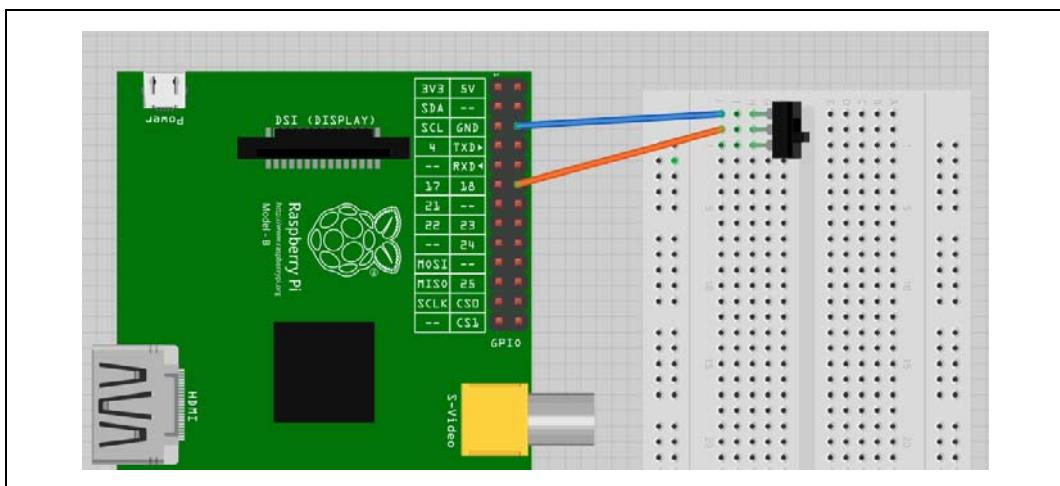
11.3. Korzystanie z dwupozycyjnego przełącznika dwustabilnego lub suwakowego

Problem

Chcesz podłączyć do swojego Raspberry Pi dwupozycyjny przełącznik dwustabilny lub suwakowy, a następnie określić pozycję przełącznika za pomocą aplikacji Pythona.

Rozwiązanie

Przełącznik dwustabilny podłącza się podobnie do przełącznika chwilowego przedstawionego w recepturze 11.1. Tym razem do obwodu podłącz złącze środkowe oraz jedno ze złączy bocznych przełącznika dwustabilnego (zobacz rysunek 11.4).



Rysunek 11.4. Dwustabilny przełącznik suwakowy podłączony do Raspberry Pi

Do zbudowania tego obwodu będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- miniaturowego przełącznika dwustabilnego lub dwupozycyjnego przełącznika suwakowego (zobacz sekcja „Pozostałe” w dodatku A).

Z przedstawionym układem współpracuje kod programu z receptury 11.1.

Omówienie

Przełączniki suwakowe są wygodne, ponieważ to, czy zwierają obwód, można określić, po prostu patrząc na nie. Obwód nie musi zawierać dodatkowych diod LED. Przełączniki dwustabilne łatwiej jednak uszkodzić, a ponadto są nieco droższe od przełączników chwilowych, które są coraz popularniejsze w elektronice, ponieważ można je umieścić za ładnym plastikowym przyciskiem.

Zobacz również

W recepturze 11.4 opisano użycie przełącznika trójpozycyjnego, którego środkowe położenie przerywa obwód.

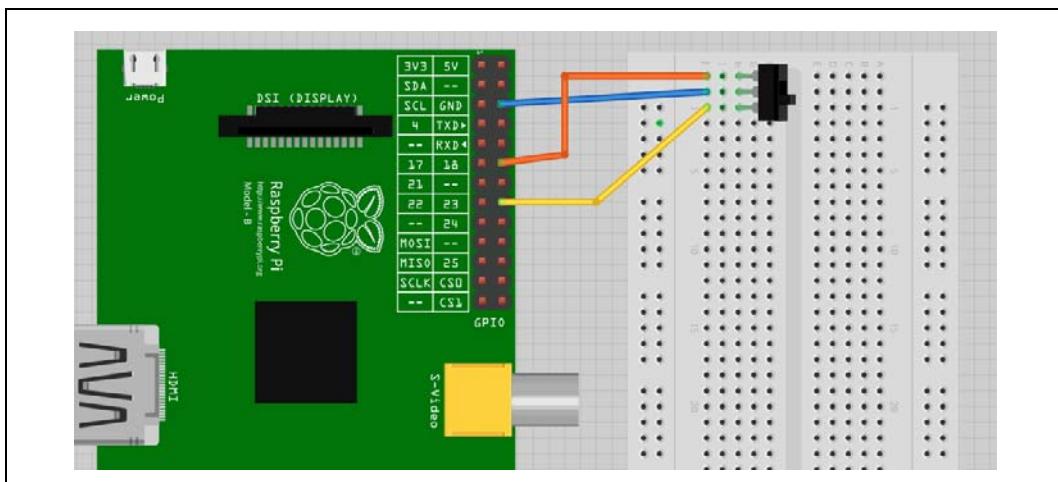
11.4. Korzystanie z przełącznika trójpozycyjnego

Problem

Chcesz podłączyć do swojego Raspberry Pi przełącznik trójpozycyjny, którego środkowe położenie przerywa obwód. Przełącznik ten ma sterować pracą programów napisanych w Pythonie.

Rozwiązanie

Podłącz przełącznik do dwóch styków złącza GPIO, tak jak pokazano na rysunku 11.5. W celu wykrycia położenia przełącznika zastosuj w swoim programie bibliotekę RPi.GPIO.



Rysunek 11.5. Przełącznik trójpozycyjny podłączony do Raspberry Pi

Do zbudowania obwodu będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- miniaturowego przełącznika trójpozycyjnego, którego środkowe położenie przerywa obwód (zobacz sekcję „Pozostałe” w dodatku A).

Wspólne środkowe złącze przełącznika jest połączone z masą. Złącza boczne są podłączone do pinów gniazda GPIO. Aby obwód działał prawidłowo, należy włączyć wewnętrzny rezystor podciągający tych pinów.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *switch_3_pos.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

top_input = 18
bottom_input = 23

GPIO.setup(top_input, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(bottom_input, GPIO.IN, pull_up_down=GPIO.PUD_UP)

switch_position = "nieznana"

while True:
    top_state = GPIO.input(top_input)
    bottom_state = GPIO.input(bottom_input)
    new_switch_position = "nieznana"
    if top_state == False:
        new_switch_position = "gorna"
    elif bottom_state == False:
        new_switch_position = "dolna"
    else:
        new_switch_position = "srodkowa"
    if new_switch_position != switch_position:
        switch_position = new_switch_position
        print(switch_position)

```

Uruchom program i zobacz, że każda zmiana położenia przełącznika spowoduje wyświetlenie odpowiedniego komunikatu:

```

$ sudo python switch_3_pos.py
gorna
srodkowa
dolna

```

Omówienie

Program korzysta z dwóch pinów złącza GPIO, na których włączono rezystory podciągające. Zmienna `switch_position` przechowuje informacje na temat bieżącego położenia przełącznika.

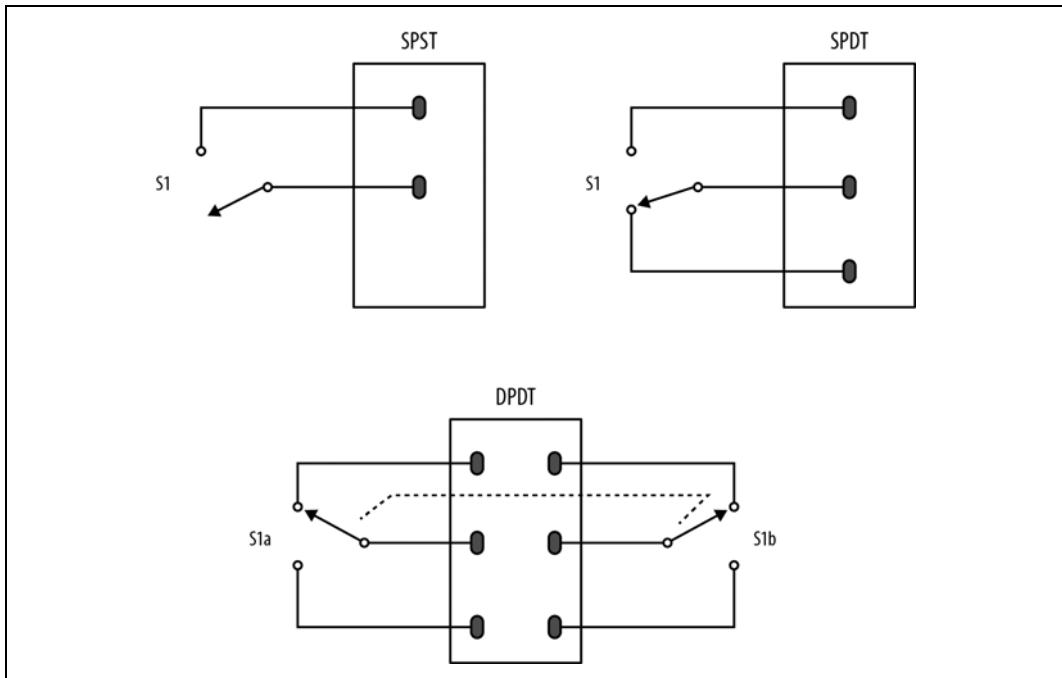
Stan dwóch złączy wejściowych jest odczytywany wewnątrz pętli, w której znajdują się struktury `if`, `elif` i `else`, określające położenie przełącznika i przypisujące je do zmiennej o nazwie `new_switch_position`. Jeżeli tej zmiennej przypisano inną wartość niż zmiennej `switch_position`, to wyświetlany jest komunikat określający położenie przełącznika.

Istnieje wiele różnych typów przełączników. Mogą być chwilowe, mogą też posiadać oznaczenia *DPDT*, *SPDT*, *SPST* itp. Litery te oznaczają:

- *D* — podwójny,
- *S* — pojedynczy,
- *P* — biegun,
- *T* — położenie.

Przełącznik *DPDT* jest przełącznikiem dwubiegunowym dwupołożeniowym. Liczba *biegunów* określa liczbę niezależnych obwodów zwieranych lub przerywanych za pomocą wspólnej dźwigni. Przełącznik typu *DPDT* może włączać i wyłączać dwa niezależne obwody. Przełącznik *SPST* zwiera lub przerywa dwa styki, a przełącznik *SPDT* zwiera środkowy styk z jednym z dwóch skrajnych styków. Przełącznik *DPDT* to tak naprawdę dwa niezależne przełączniki *SPDT* operowane za pomocą jednej dźwigni, zamknięte we wspólnej obudowie.

Budowę najpopularniejszych przełączników pokazano na rysunku 11.6.



Rysunek 11.6. Rodzaje przełączników

Zobacz również

Więcej informacji na temat polecenia if znajdziesz w recepturze 5.18.

Wiadomości na temat podłączania prostszego przełącznika znajdziesz w recepturze 11.1.

11.5. Redukcja stuków powstających podczas wciskania przycisku

Problem

Czasami wcisnięcie przycisku powoduje, że operacja uruchamiana przez przycisk zostanie wykonana więcej niż raz. Dzieje się tak, ponieważ przyciski generują *stuki*. Chcesz napisać kod *usuwający stuki* powstające podczas wciskania przycisku.

Rozwiązanie

Problem stuków można rozwiązać na wiele sposobów. Aby je przeanalizować, zbuduj układ przedstawiony w recepturze 11.2.

W poniższym przykładowym kodzie nie zastosowano żadnej techniki rozwiązującej problem stuków:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

switch_pin = 18
led_pin = 23

GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led_pin, GPIO.OUT)

led_state = False
old_input_state = True #podciagnięty

while True:
    new_input_state = GPIO.input(switch_pin)
    if new_input_state == False and old_input_state == True:
        led_state = not led_state
    old_input_state = new_input_state
    GPIO.output(led_pin, led_state)
```

Przedstawiony program zacznie sprawiać kłopoty, gdy wciskany przycisk zacznie generować stuki — program będzie działał tak, jakby przycisk był wciskany kilkakrotnie w bardzo krótkim odstępie czasu. Jeżeli stuki wygenerują nieparzystą liczbę impulsów, użytkownik będzie mieć wrażenie, że wszystko działa poprawnie. Jeśli jednak stuki wygenerują parzystą liczbę impulsów odebranych przez złącze wejściowe, dioda LED zostanie włączona i niemal natychmiast wyłączona.

Aby stuki generowane przez przełącznik nie wpływały na pracę programu, powinien on ignorować sygnały odbierane przez złącze wejściowe przez jakiś czas po odebraniu pierwszego sygnału — w tym czasie przełącznik się ustabilizuje.

Prostym i łatwym sposobem, aby to zrobić, jest zastosowanie polecenia `time.sleep`, które po wykryciu wcisnięcia przycisku zatrzymywałoby odbieranie sygnałów przez złącze portu GPIO przez, powiedzmy, 0,2 sekundy. Taki odstęp czasu jest bardzo długi. Najprawdopodobniej możesz go znacznie skrócić.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *switch_on_off_no_bounce.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

switch_pin = 18
led_pin = 23

GPIO.setup(switch_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led_pin, GPIO.OUT)

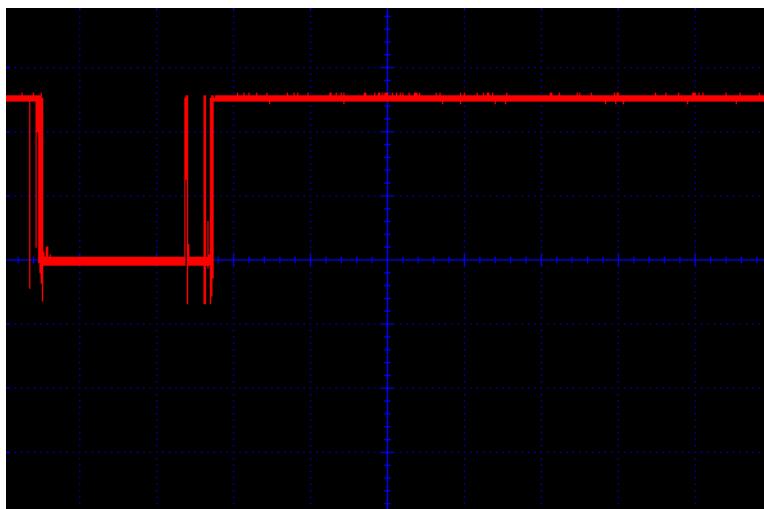
led_state = False
old_input_state = True #podciagnięty

while True:
    new_input_state = GPIO.input(switch_pin)
    if new_input_state == False and old_input_state == True:
        led_state = not led_state
        time.sleep(0.2)
    old_input_state = new_input_state
    GPIO.output(led_pin, led_state)
```

Omówienie

Takie rozwiązanie sprawdza się w większości przypadków. Możesz jednak również w kontekście obsługi przełącznika zastosować przerwanie (zobacz receptura 9.12).

Stuki są problemem większości przełączników. Na rysunku 11.7 przedstawiono ślad (widoczny na ekranie oscyloskopu) powstały na skutek wciśnięcia przycisku wyraźnie generującego stuki.



Rysunek 11.7. Ślad wciśnięcia przycisku widoczny na ekranie oscyloskopu

Stuki powstają podczas zwierania obwodu oraz podczas jego przerywania. Większość przełączników nie jest jednak aż tak marnej jakości.

Zobacz również

Podstawowe informacje dotyczące podłączania przycisku znajdziesz w recepturze 11.1.

11.6. Korzystanie z zewnętrznego rezystora podciągającego

Problem

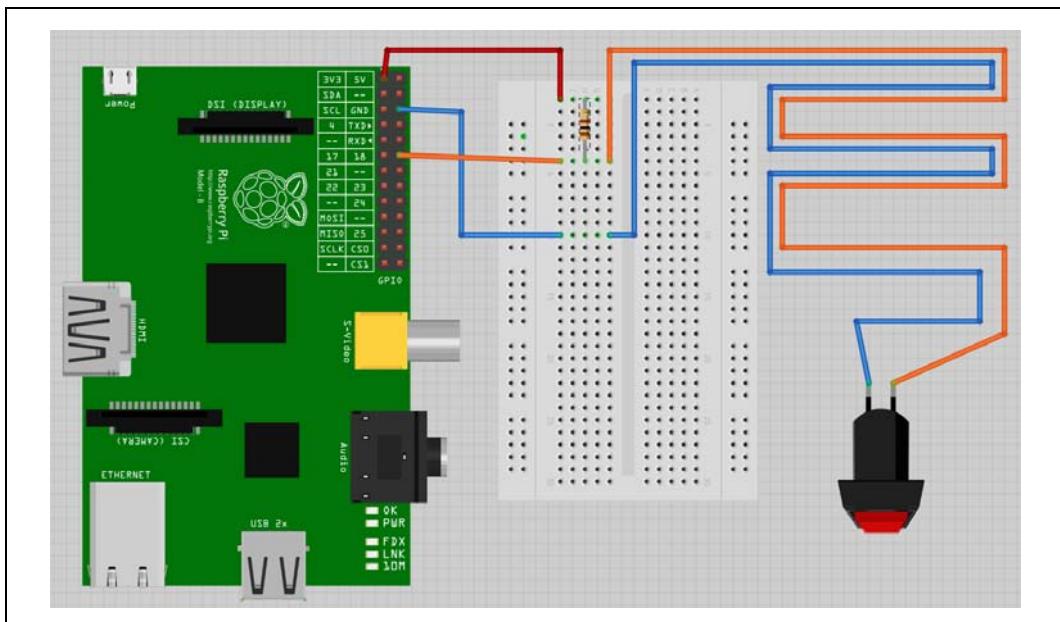
Chcesz podłączyć przełącznik do Raspberry Pi za pomocą dłuższego kabla. Niestety wskutek takiego rozwiązania otrzymujesz nieprawidłowe odczyty na wejściu gniazda GPIO.

Rozwiążanie

Wewnętrzne rezystory podciągające są dość słabe (około $40\text{ k}\Omega$). A zatem gdy korzystasz z długiego kabla w środowisku, w którym występują zakłócenia elektromagnetyczne, możesz uzyskiwać nieprawidłowe odczyty stanu cyfrowego wejścia Raspberry Pi. Problem ten możesz

rozwiązać, wyłączając wewnętrzne rezystory (podciągający i ściągający), a następnie włączając do układu zewnętrzny rezistor podciągający.

Na rysunku 11.8 przedstawiono obwód, w którym zastosowano zewnętrzny rezistor podciągający.



Rysunek 11.8. Korzystanie z zewnętrznego rezystora podciągającego

Działanie tego obwodu możesz sprawdzić za pomocą programu *switch.py* omówionego w recepturze 11.1.

Omówienie

Im mniejszy opór stawiany przez rezistor, tym dłuższy kabel możesz zastosować. Pamiętaj jednak o tym, że wcisnięcie przycisku powoduje przepływ prądu o napięciu 3,3 V przez rezistor. Rezystor stawiający opór 100 Ω będzie pobierał prąd o natężeniu $3,3 \text{ V} / 100 \Omega = 33 \text{ mA}$. Wartość ta znajduje się w bezpiecznej granicy natężenia prądu (50 mA) dostarczanego przez złącze 3,3 V. Nie stosuj rezystorów charakteryzujących się mniejszym oporem.

W większości przypadków bez żadnych problemów możesz stosować rezistor charakteryzujący się oporem 1 k Ω .

Zobacz również

Podstawowe informacje dotyczące podłączania przycisku znajdziesz w recepturze 11.1.

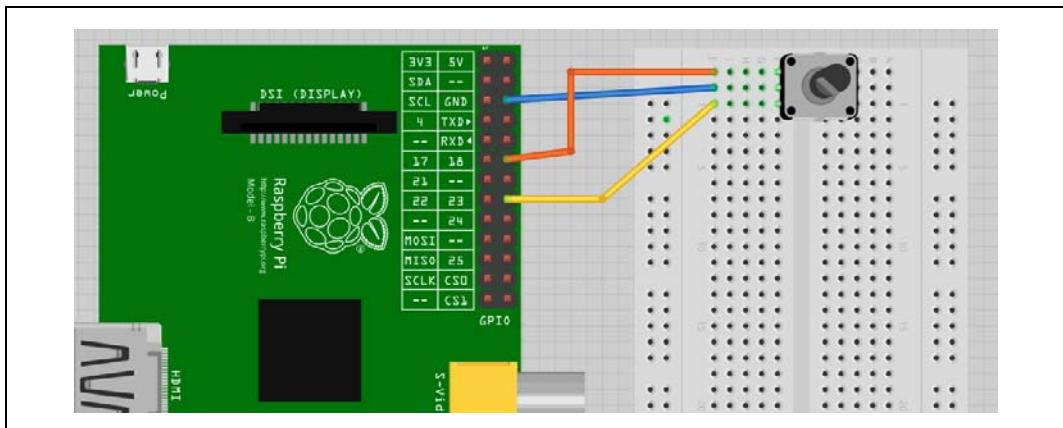
11.7. Korzystanie z (kwadratowego) kodera obrotowego

Problem

Chcesz wykrywać ruch obrotowy za pomocą kodera obrotowego.

Rozwiązanie

Do dwóch styków gniazda GPIO podłącz kwadratowy koder obrotowy, tak jak pokazano na rysunku 11.9.



Rysunek 11.9. Koder obrotowy podłączony do Raspberry Pi

Do zbudowania powyższego obwodu będziesz potrzebować:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- kwadratowego kodera obrotowego (zobacz sekcję „Pozostałe” w dodatku A).

Koder obrotowy, którego używamy, nazywany jest **kwadratowym**. Jego działanie można porównać do pary przełączników. Kolejność, w jakiej przełączniki te zwierają i otwierają swoje obwody, określa kierunek obrotu wałka kodera.

Koder ten ma trzy wyprowadzenia: *wspólne* złącze środkowe i złącza *A* oraz *B* znajdujące się po bokach. Nie wszystkie kodery mają tak ułożone wyprowadzenia. Opis złączy posiadaneego modelu znajdziesz w jego dokumentacji. Kolejną komplikacją jest to, że wiele koderów obrotowych może również pełnić rolę przycisku. W takim przypadku wbudowany przełącznik będzie posiadał oddzielną parę złączy.

Umieść poniższy kod programu w oknie środowisk programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *rotary_encoder.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
```

```

input_A = 18
input_B = 23

GPIO.setup(input_A, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(input_B, GPIO.IN, pull_up_down=GPIO.PUD_UP)

old_a = True
old_b = True

def get_encoder_turn():
    #zwraca -1, 0 lub +1
    global old_a, old_b
    result = 0
    new_a = GPIO.input(input_A)
    new_b = GPIO.input(input_B)
    if new_a != old_a or new_b != old_b :
        if old_a == 0 and new_a == 1 :
            result = (old_b * 2 - 1)
        elif old_b == 0 and new_b == 1 :
            result = -(old_a * 2 - 1)
    old_a, old_b = new_a, new_b
    time.sleep(0.001)
    return result

x = 0

while True:
    change = get_encoder_turn()
    if change != 0 :
        x = x + change
        print(x)

```

Program przeznaczony do testowania kodera wyświetla kolejne liczby, gdy jest on obracany w kierunku zgodnym z ruchem wskazówek zegara. Obracanie wałka kodera w kierunku przeciwnym do ruchu wskazówek zegara spowoduje odliczanie:

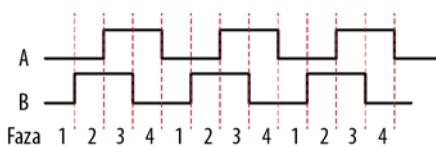
```

pi@raspberrypi ~ $ sudo python rotary_encoder.py
1
2
3
4
5
6
7
8
9
10
9
8
7
6
5
4

```

Omówienie

Na rysunku 11.10 przedstawiono sekwencję impulsów generowaną na złączach A i B. Jak widzisz, wzór powtarza się co cztery kroki. Stąd właśnie wzięła się nazwa kodera *kwadratowego*.



Rysunek 11.10. Działanie kodera kwadratowego

Obracając wałkiem kodera w kierunku zgodnym z ruchem wskazówek zegara (na rysunku 11.10 od strony lewej do prawej), otrzymamy następującą sekwencję:

Faza	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Obracając wałkiem kodera w przeciwnym kierunku, otrzymamy poniższą sekwencję:

Faza	A	B
4	1	0
3	1	1
2	0	1
1	0	0

W przedstawionym wcześniej programie Pythona algorytm określający kierunek obrotu umieszczono w funkcji `get_encoder_turn`. Funkcja ta zwraca 0 (gdy nie wykryto żadnego ruchu), 1 (gdy koder został obrócony w kierunku zgodnym z ruchem wskazówek zegara) lub -1 (gdy koder został obrócony w kierunku przeciwnym do ruchu wskazówek zegara). Funkcja ta przechowuje poprzednie stany przełączników *A* i *B* w zmiennych globalnych o nazwach `old_a` i `old_b`. Program porównując (wykonując operacje logiczne na bitach) wartości przechowywane w tych zmiennych z aktualnie odczytanymi wartościami, określa kierunek obrotu kodera.

Uśpienie trwające 1 milisekundę przeciwdziała uzyskiwaniu nieprawidłowych odczytów wywołanych przez zbyt szybkie zmiany stanu wejścia.

Program testowy powinien działać prawidłowo niezależnie od tego, jak szybko będziesz kręcić pokrętłem kodera. Unikaj wykonywania czasochłonnych operacji w pętli — może to doprowadzić do pominięcia przez program kilku kroków, o jakie został obrócony koder.

Zobacz również

Pozycję, w jakiej znajduje się obracana gałka, możesz również określić za pomocą rezystora nastawnego (zobacz receptura 12.1) lub przy użyciu przetwornika analogowo-cyfrowego (zobacz receptura 12.4).

11.8. Korzystanie z bloku klawiszy

Problem

Chcesz podłączyć blok klawiszy do Raspberry Pi.

Rozwiązanie

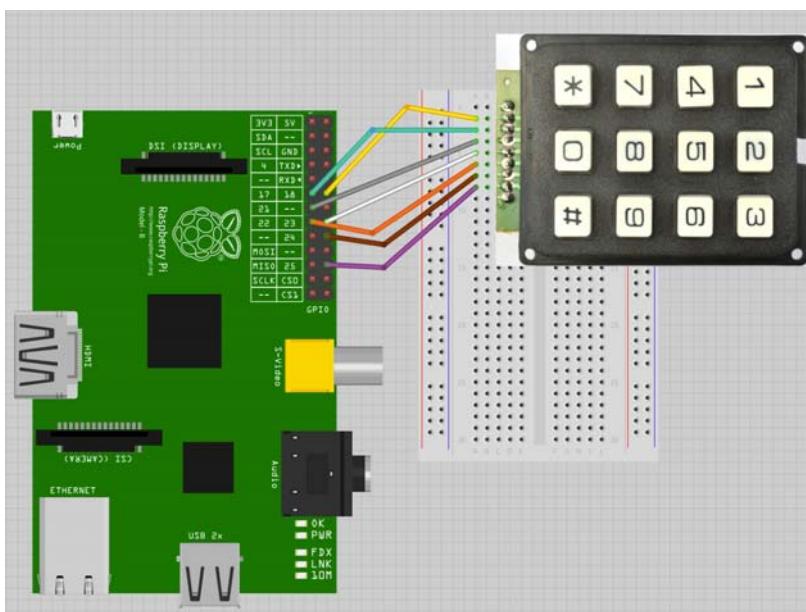
Bloki klawiszy składają się z klawiszy umieszczonych w rzędach i kolumnach. Każdy wiersz i każda kolumna zawierają przełącznik. Aby Raspberry Pi potrafiło określić to, który przycisk został wciśnięty, musisz podłączyć wszystkie rzędy i kolumny bloku do styków złącza GPIO. Dla bloku klawiszy 4x3 będziesz musiał podłączyć przewody do 4+3, czyli 7 pinów. Raspberry Pi określa, który przycisk został wciśnięty, skanując kolejno wszystkie złącza, do których podłączono kolumny bloku klawiszy, a następnie odczytuje stany wejść, do których podłączone są wiersze.

Nie ma jednego standardu określającego ułożenie złączy wyjściowych bloku klawiszy.

Do zbudowania obwodu przedstawionego w dalszej części niniejszej receptury będziesz potrzebować:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- bloku klawiszy 4x3 (zobacz sekcja „Pozostałe” w dodatku A),
- siedmiostykowej listwy złączy goldpin (zobacz sekcja „Pozostałe” w dodatku A).

Na rysunku 11.11 przedstawiono schemat wykonawczy obwodu zawierającego blok klawiszy firmy SparkFun wymieniony w sekcji „Pozostałe” w dodatku A. Blok nie został wyposażony fabrycznie w złącza typu goldpin. Należy je samodzielnie przylutować do płytki modulu.



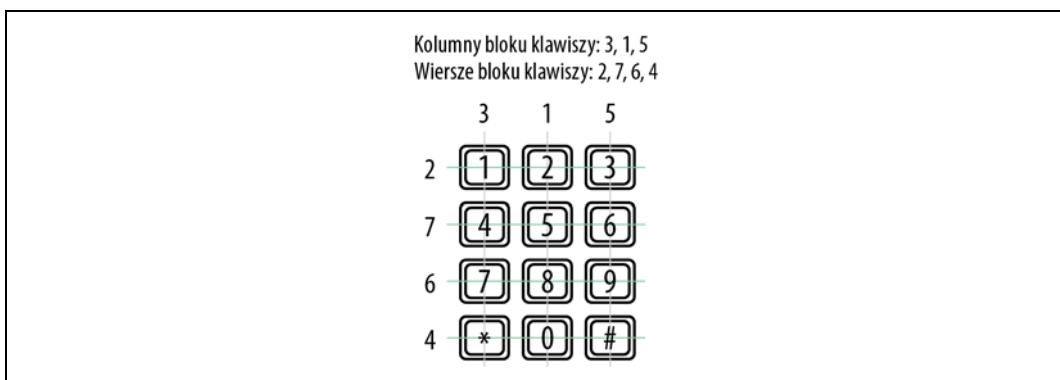
Rysunek 11.11. Blok klawiszy podłączony do Raspberry Pi

Kod programu znajdujący się w tym podrozdziale umieść w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *keypad.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.



Upewnij się, że prawidłowo zidentyfikowałeś piny wierszy i kolumn posiadanego bloku klawiszy. Jeżeli zajdzie taka potrzeba, zmodyfikuj zmienne *rows* i *cols*. Jeżeli tego nie zrobisz, wcisnięcie przycisku może spowodować zwarcie ze sobą dwóch złącz wyjściowych gniazda GPIO. Zwarcie gniazda podającego sygnał wysoki z gniazdem podającym sygnał niski może doprowadzić do uszkodzenia Twojego Raspberry Pi.

W programie zdefiniowano wiersze i kolumny bloku klawiszy produkowanego przez firmę SparkFun, wymienionego w sekcji „Pozostałe” w dodatku A. Pierwszy wiersz jest połączony do pinu gniazda GPIO o numerze 17, drugi do pinu o numerze 25 i tak dalej. Numery złączy znajdujących się na bloku klawiszy, odpowiadających za pracę wierszy i kolumn, przedstawiono na rysunku 11.12.



Rysunek 11.12. Złącza bloku klawiszy

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

rows = [17, 25, 24, 23]
cols = [27, 18, 22]
keys = [
    ['1', '2', '3'],
    ['4', '5', '6'],
    ['7', '8', '9'],
    ['*', '0', '#']]

for row_pin in rows:
    GPIO.setup(row_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

for col_pin in cols:
    GPIO.setup(col_pin, GPIO.OUT)

def get_key():
    key = 0
    for col_num, col_pin in enumerate(cols):
```

```

GPIO.output(col_pin, 1)
for row_num, row_pin in enumerate(rows):
    if GPIO.input(row_pin):
        key = keys[row_num][col_num]
GPIO.output(col_pin, 0)
return key

while True:
    key = get_key()
    if key :
        print(key)
    time.sleep(0.3)

```

Program korzysta ze złącza GPIO, a więc musisz go uruchomić jako użytkownik posiadający uprawnienia administratora. Program wyświetla informacje o kolejno wciskanych klawiszach.

```

pi@raspberrypi ~ $ sudo python keypad.py
1
2
3
4
5
6
7
8
9
*
0
#

```

Omówienie

Zmienna `keys` zawiera mapę z nazwami wszystkich klawiszy znajdujących się w bloku. Możesz dostosować te nazwy do posiadanej modułu klawiszy.

Złącza odpowiadające za obsługę wierszy i kolumn są inicjowane w pętlach, ponieważ program musi zainicjować wiele wejść i wyjść.

Sercem programu jest funkcja `get_key`. Uruchamia ona kolejno każdą kolumnę, podając wysoki sygnał na odpowiednie złącza. Wewnętrzna pętla sprawdza kolejno wszystkie wiersze. Jeżeli na złączu, do którego podłączony jest któryś wiersz, odbierany jest sygnał wysoki, to program szuka nazwy klawisza w tablicy `keys`. Jeżeli funkcja nie wykryje wcisnięcia żadnego z klawiszy, zwraca domyślną wartość zmiennej `key` (0).

Do głównej pętli `while` przekazywana jest wartość zmiennej `key`. Pętla ta wyświetla nazwę wcisniętego klawisza. Polecenie `sleep` spowalnia generowanie danych wyjściowych przez program.

Zobacz również

Zamiast używać bloku klawiszy, możesz podłączyć do swojego Raspberry Pi standardową klawiaturę wyposażoną w złącze USB.

11.9. Wykrywanie ruchu

Problem

Chcesz, żeby program Pythona wykonywał określone operacje po wykryciu ruchu przez specjalny czujnik.

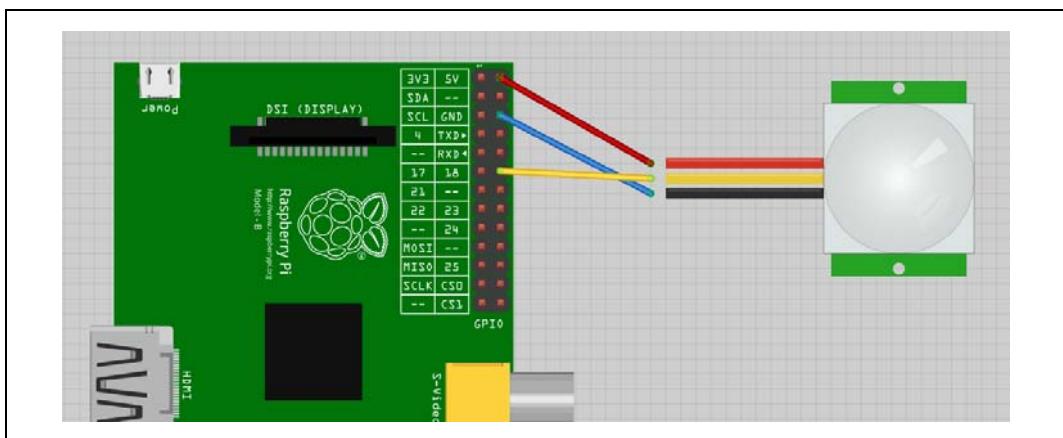
Rozwiązanie

Skorzystaj z modułu zawierającego pasywny czujnik podczerwieni.

Do zbudowania obwodu przedstawionego w dalszej części niniejszej receptury będziesz potrzebować:

- przewodów połączeniowych obustronnie zakończonych końcówkami żeńskimi (zobacz sekcję „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- modułu detektora ruchu działającego w paśmie podczerwieni (zobacz sekcję „Moduły” w dodatku A).

Na rysunku 11.13 przedstawiono schemat wykonawczy obwodu zawierającego moduł detektora ruchu. Zaprezentowany moduł powinien być zasilany prądem o napięciu 5 V. Generuje on sygnał wyjściowy o napięciu 3,3 V, a więc jest idealny do użycia z Raspberry Pi.



Rysunek 11.13. Detektor ruchu działający w paśmie podczerwieni podłączony do Raspberry Pi



Upewnij się, że czujnik podczerwieni, który chcesz zastosować, generuje sygnał wyjściowy o napięciu 3,3 V. Jeżeli generowany przez niego sygnał charakteryzuje się napięciem 5 V, musisz je obniżyć do 3,3 V za pomocą pary rezystorów (zobacz receptura 8.12).

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `pir.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.IN)

while True:
    input_state = GPIO.input(18)
    if input_state == True:
        print('Wykryto ruch')
    time.sleep(1)
```

Program monitoruje stan złącza GPIO (pin wejścia o numerze 18) i wyświetla komunikat informujący o wykryciu ruchu przez czujnik.

```
$ sudo python pir.py
Wykryto ruch
Wykryto ruch
```

Omówienie

Czujnik po wykryciu ruchu generuje przez jakiś czas wysoki sygnał wyjściowy. Czas ten możesz wyregulować za pomocą jednego z potencjometrów dostrojczych znajdujących się na płytce obwodu czujnika.

Zobacz również

Möżesz połączyć tę recepturę z recepturą 7.15. W ten sposób zbudujesz urządzenie przeciw-włamaniowe wysyłające wiadomość informującą o wykryciu intruza za pośrednictwem poczty elektronicznej.

11.10. Raspberry Pi i moduł GPS

Problem

Chcesz podłączyć do Raspberry Pi moduł GPS, a następnie uzyskać dostęp do danych za pomocą aplikacji Pythona.

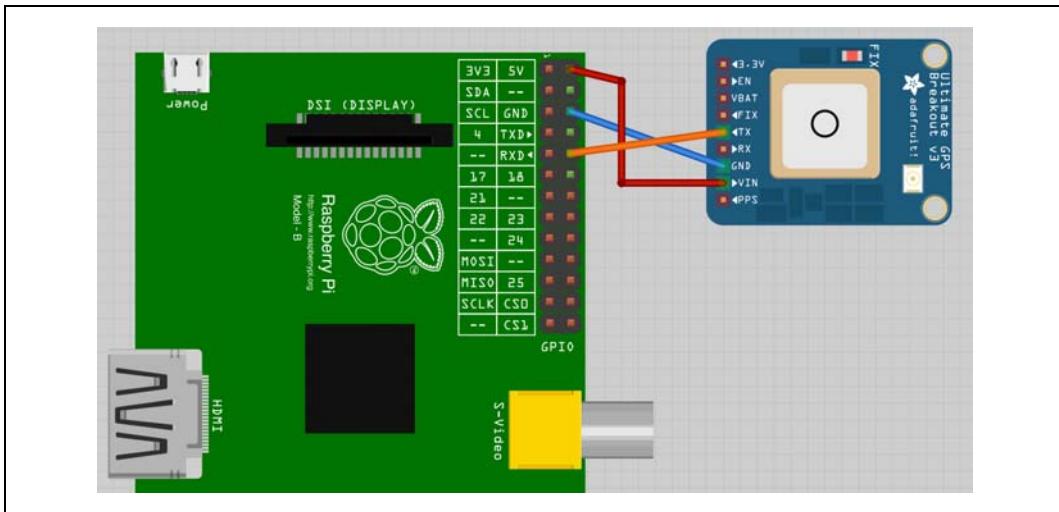
Rozwiążanie

Szeregowy moduł GPS zasilany prądem o napięciu 3,3 V może być podłączony bezpośrednio do złącza RXD znajdującego się na płytce Raspberry Pi. Aby korzystać z portu szeregowego, musisz najpierw wyłączyć funkcję logowania konsoli, a więc pracę zacznij od wykonania czynności opisanych w recepturze 8.7.

Na rysunku 11.14 pokazano sposób podłączania modułu. Złącze RXD znajdujące się na płytce Raspberry Pi należy połączyć ze złączem Tx na module GPS. Pozostałe dwa przewody to masa i prąd zasilający o napięciu 5 V. Połączenia te można wykonać za pomocą przewodów połączeniowych obustronnie zakończonych wtykami żeńskimi.

Dane GPS muszą być przetworzone. Na szczęście stworzono w tym celu dobry zestaw narzędzi. Zainstaluj następujące pakiety:

```
$ sudo apt-get install gpsd
$ sudo apt-get install gpsd-clients
```



Rysunek 11.14. Moduł GPS podłączony do Raspberry Pi

Najważniejszym składnikiem tych pakietów jest narzędzie `gpsd`. Pozwala ono na odczytanie danych GPS za pośrednictwem portu szeregowego, interfejsu USB, a także innych źródeł. Narzędzie to udostępnia przetworzone dane dla aplikacji klienckiej za pośrednictwem lokalnej usługi sieciowej uruchomionej na porcie numer 2748.

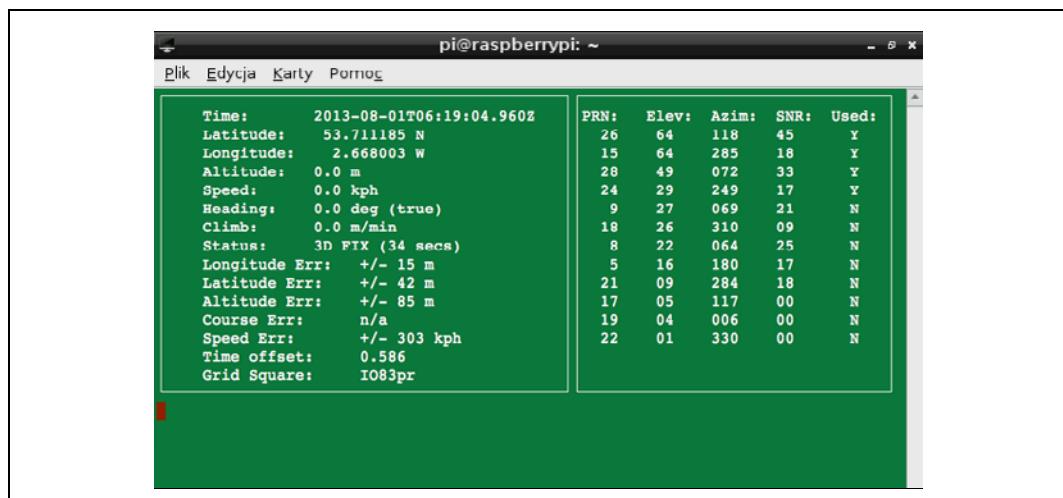
Uruchom usługę `gpsd` za pomocą następującego polecenia:

```
$ sudo gpsd /dev/ttyAMA0
```

Działanie usługi można sprawdzić przy użyciu polecenia:

```
$ cgps -s
```

Parametr `-s` jest opcjonalny. Sprawia on, że nieprzetworzone dane nie zostaną wyświetcone (zobacz rysunek 11.15).



Rysunek 11.15. Sprawdzanie działania modułu GPS za pomocą narzędzia `cgps`

Trzeci z zainstalowanych pakietów (`python-gps`) jest biblioteką umożliwiającą programom napisanym w Pythonie na łatwe uzyskiwanie dostępu do danych GPS. Bibliotekę `python_gps` zastosujemy w prostym programie wyświetlającym szerokość geograficzną (`lat`), długość geograficzną (`lon`) oraz czas (`time`).

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `gps_test.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
from gps import *
session = gps()
session.stream(WATCH_ENABLE|WATCH_NEWSTYLE)

while True:
    report = session.next()
    if report.keys()[0] == 'epx' :
        lat = float(report['lat'])
        lon = float(report['lon'])
        print("lat=%f\lon=%f\ctime=%s" % (lat, lon, report['time']))
    time.sleep(0.5)
```

Po uruchomieniu tego programu na ekranie powinny się wyświetlić dane:

```
$ python gps_test.py
lat=53.710257  lon=-2.664245  time=2013-08-01T08:06:24.960Z
lat=53.710258  lon=-2.664252  time=2013-08-01T08:06:25.960Z
lat=53.710258  lon=-2.664252  time=2013-08-01T08:06:25.960Z
lat=53.710248  lon=-2.664243  time=2013-08-01T08:06:26.960Z
lat=53.710248  lon=-2.664243  time=2013-08-01T08:06:26.960Z
lat=53.710248  lon=-2.664250  time=2013-08-01T08:06:27.960Z
```

Omówienie

Program tworzy *sesję*, a następnie strumień danych do odczytu. Moduł GPS stale dostarcza komunikaty w różnych formatach. Polecenie `if` ogranicza ilość danych przechwytywanych przez program do informacji dotyczących położenia. Fragmenty komunikatów modułu GPS są przechowywane w słowniku, co pozwala na uzyskanie do nich dostępu oraz ich wyświetlenie.

Dane GPS możesz przetwarzać za pomocą aplikacji Pythona. Możesz je również wyświetlić przy użyciu narzędzia `xgps` (zobacz rysunek 11.16). Aby uruchomić to narzędzie, wpisz polecenie:

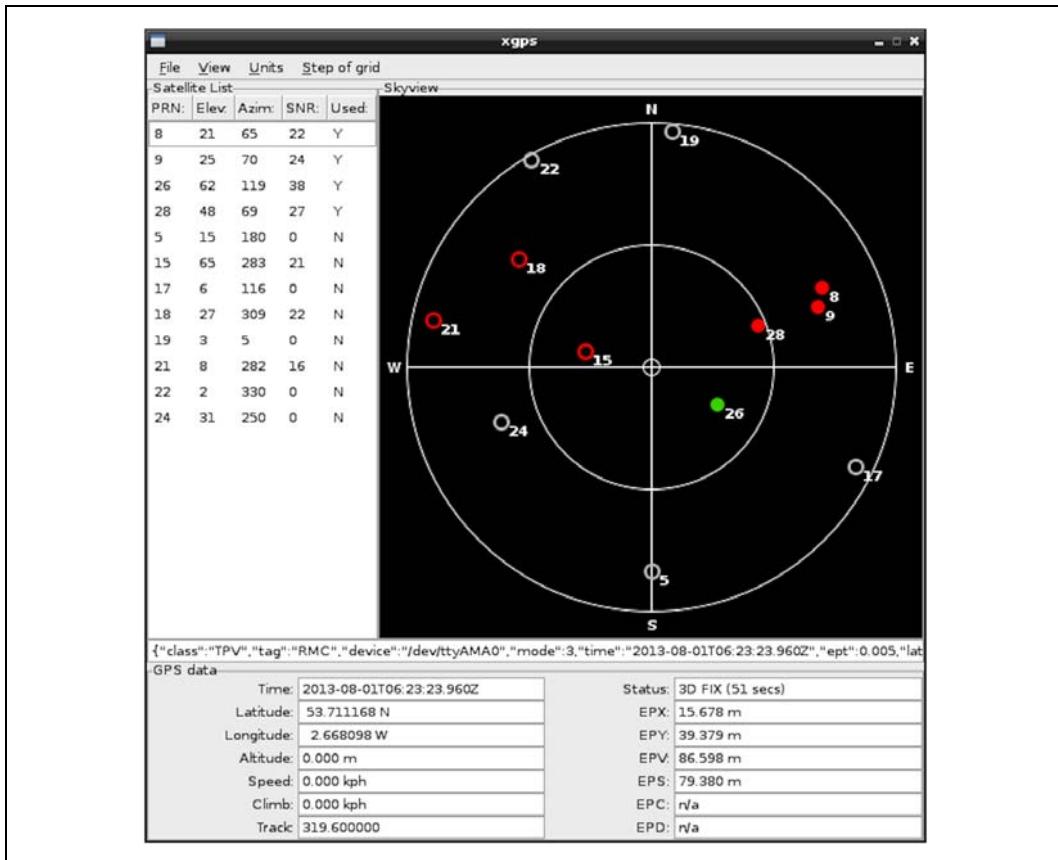
```
$ xgps
```

Jest to narzędzie posiadające interfejs graficzny, a więc musisz je uruchomić bezpośrednio na Raspberry Pi podłączonym do ekranu lub zastosować w tym celu VNC (zobacz receptura 2.8).

Zobacz również

Zaprezentowane programy możesz również zastosować do pracy z modułem GPS wyposażonym w interfejs USB — <http://blog.retep.org/2012/06/18/getting-gps-to-work-on-a-raspberry-pi/>.

Więcej informacji na temat narzędzia `gpsd` znajdziesz w internecie pod adresem: <https://savannah.nongnu.org/projects/gpsd>.



Rysunek 11.16. Dane GPS wyświetcone w programie xgps

11.11. Wprowadzanie danych z klawiatury

Problem

Chcesz, aby program przechwytywał dane dotyczące klawiszy wciskanych przez użytkownika na klawiaturze podłączonej do Raspberry Pi za pośrednictwem gniazda USB.

Rozwiążanie

Z zagadnieniem tym można się zmierzyć przynajmniej na dwa sposoby. Najprostszym rozwiązaniem jest zastosowanie funkcji `sys.stdin.read`. Metoda ta wydaje się najlepsza, ponieważ nie wymaga graficznego interfejsu użytkownika, a więc program korzystający z tej funkcji można uruchomić za pomocą sesji protokołu SSH.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `keys_sys.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```

import sys, tty, termios

def read_ch():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch

while True:
    ch = read_ch()
    if ch == 'x':
        break
    print("wciskasz klawisz: " + ch)

```

Alternatywnym sposobem na rozwiązywanie opisywanego problemu jest zastosowanie biblioteki Pygame. Biblioteka ta jest przeznaczona do tworzenia gier, można jej jednak również użyć do przechwytywania informacji o wciskanych klawiszach. Przechwycone dane można wykorzystać w celu wykonania pewnych operacji w programie lub do sterowania pracą robota (zobacz receptura 10.8).

W poniższym programie zastosowano bibliotekę Pygame w celu wyświetlania komunikatów po naciśnięciu klawisza. Program ten działa tylko w środowisku graficznym, a więc musisz uruchomić go bezpośrednio na Raspberry Pi lub skorzystać z VNC (zobacz receptura 2.8). Kod tego programu zapisz w pliku o nazwie *keys_pygame.py*.

```

import pygame
import sys
from pygame.locals import *

pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.mouse.set_visible(0)

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            sys.exit()
        if event.type == KEYDOWN:
            print("Kod: " + str(event.key) + " Znak: " + chr(event.key))

```

Po uruchomieniu programu zobaczysz na swoim ekranie puste okno środowiska Pygame. Program będzie przechwytywał klawisze tylko wtedy, gdy jego okno będzie aktywne. Aplikacja wyświetla dane wyjściowe w oknie Terminala, w którym została uruchomiona.

Wciśnięcie któregoś z klawiszy kierunkowych lub klawisza *Shift* spowoduje wyświetlenie komunikatu o błędzie. Klawisze te nie mają swojego odzwierciedlenia w tablicy znaków ASCII.

```
$python keys_pygame.py
Kod: 97 Znak: a
Kod: 98 Znak: b
Kod: 99 Znak: c
Kod: 120 Znak: x
Kod: 13 Znak:
```

Zaprezentowanego programu nie da się zamknąć kombinacją klawiszy *Ctrl+C*. Program można wyłączyć, klikając kursem myszy znak X umieszczony w górnym prawym rogu okna Pygame.

Omówienie

Korzystając z biblioteki Pygame, przypisujesz klawiszom stałe, zdefiniowane wartości. Dzięki temu możesz korzystać z klawiszy strzałek i innych klawiszy, które nie reprezentują znaków ASCII (np. z klawisza *Home*). Inne sposoby przechwytywania klawiszy na to nie pozwalają.

Zobacz również

Czasami wygodniej jest przechwytywać informacje o wciśnięciu klawiszy na klawiaturze, niż korzystać z dodatkowego bloku klawiszy (zobacz receptura 11.8).

11.12. Przechwytywanie ruchów myszy

Problem

Chcesz wykrywać ruchy myszy za pomocą aplikacji Pythona.

Rozwiążanie

Możesz tutaj skorzystać z techniki podobnej do tej, której używałeś, przechwytyując dane dotyczące wciśnięcia klawiszy na klawiaturze (zobacz receptura 11.11).

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *mouse_pygame.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pygame
import sys
from pygame.locals import *

pygame.init()
screen = pygame.display.set_mode((640, 480))
pygame.mouse.set_visible(0)

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            sys.exit()
        if event.type == MOUSEMOTION:
            print("Mysz: (%d, %d)" % event.pos)
```

Zdarzenie **MOUSEMOTION** jest uruchamiane podczas każdego ruchu myszy w oknie Pygame. Współrzędne kurSORA są przechowywane w zmiennej **pos**. Są to współrzędne względne. Punktem ich odniesienia jest lewy górnY róg okna.

```
Mysz: (262, 285)
Mysz: (262, 283)
Mysz: (262, 281)
Mysz: (262, 280)
Mysz: (262, 278)
Mysz: (262, 274)
Mysz: (262, 270)
Mysz: (260, 261)
Mysz: (258, 252)
Mysz: (256, 241)
Mysz: (254, 232)
```

Omówienie

Z pomocą zdarzeń MOUSEBUTTONDOWN i MOUSEBUTTONUP możesz przechwytywać wciśnięcie i zwolnienie lewego klawisza myszy.

Zobacz również

Więcej informacji na temat obsługi myszy przez bibliotekę Pygame znajdziesz na stronie <http://www.pygame.org/docs/ref/mouse.html>.

Jeżeli szukasz informacji na temat przechwytywania danych dotyczących wciskania klawiszy klawiatury, zajrzyj do receptury 11.11.

11.13. Korzystanie z modułu zegara czasu rzeczywistego

Problem

Chcesz, żeby Raspberry Pi pamiętało czas, nawet gdy nie posiada ono dostępu do sieci.

Rozwiązanie

Skorzystaj z modułu zegara czasu rzeczywistego.

Popularnym układem zegara czasu rzeczywistego jest chip DS1307. Wyposażono go w interfejs I2C. Ponadto można go kupić w formie gotowego modułu zawierającego poza wspomnianym układem scalonym również rezonator kwarcowy (zapewniający dokładność pomiaru czasu), a także gniazdo baterii litowej dostarczającej do układu prąd o napięciu 3 V.

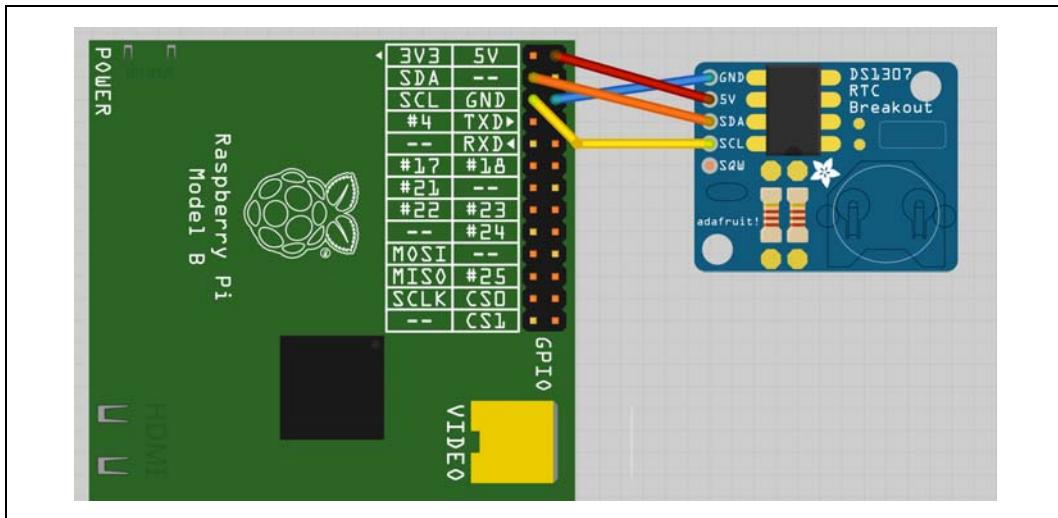
Aby skorzystać z niniejszej receptury, potrzebujesz:

- modułu zegara czasu rzeczywistego (moduł powinien być oparty na układzie DS1307 lub innym z nim kompatybilnym) (zobacz sekcja „Moduły” w dodatku A), możesz również skorzystać z płytka aLaMode (zobacz receptura 14.13),
- przewodów połączeniowych obustronnie zakończonych końcówkami żeńskimi (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A).



Używany przez Ciebie moduł zegara czasu rzeczywistego musi generować sygnał wyjściowy o napięciu 3,3 V. Oznacza to, że interfejs I2C nie powinien zawierać żadnych rezystorów podciągających lub powinien mieć rezystory podciągające napięcie do 3,3 V. Nie stosuj układu zawierającego rezystory podciągające napięcie do 5 V. Jeżeli korzystasz z zegara firmy Adafruit, to po prostu nie przylutowuj do modułu dwóch rezystorów. W przypadku gdy zakupiłeś gotowy prefabrykowany moduł, ostrożnie zdemontuj z jego płytka wszelkie rezystory podciągające.

Jeżeli kupiony przez Ciebie zegar jest zestawem do samodzielnego montażu, zbuduj dostarczony układ, pomijając rezystory podciągające. Następnie podłącz go do Raspberry Pi, tak jak pokazano na rysunku 11.17.



Rysunek 11.17. Moduł zegara czasu rzeczywistego podłączony do Raspberry Pi

Układ DS1307 korzysta z magistrali I2C, a więc musisz odpowiednio skonfigurować swoje Raspberry Pi. W tym celu skorzystaj ze wskazówek umieszczonych w recepturze 8.4. Prawidłowość połączenia możesz sprawdzić za pomocą narzędzi omówionych w recepturze 8.5.

```
$ sudo i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  - - - - - - - - - - - - - - - - - - - - - -
10:  --  - - - - - - - - - - - - - - - - - - - - - -
20:  --  - - - - - - - - - - - - - - - - - - - - - -
30:  --  - - - - - - - - - - - - - - - - - - - - - -
40:  --  - - - - - - - - - - - - - - - - - - - - - -
50:  --  - - - - - - - - - - - - - - - - - - - - - -
60:  --  - - - - - - - - - 68  - - - - - - - - - -
70:  --  - - - - - - - - - - - - - - - - - - - - - -
```

Jak wynika z powyższej tabeli moduł zegara czasu rzeczywistego podłączono do magistrali I2C pod adresem o numerze 68.

Jeżeli korzystasz ze starszej wersji Raspberry Pi (model B, rev1), to za parametrem `y` umieść wartość 0 (w poleceniu użytym do wywołania tabeli). Modele rev1 charakteryzują się tym, że mają czarne gniazda słuchawkowe.

Uruchom poniższe polecenia. Pozwolą one na korzystanie z zegara przez program o nazwie `hwclock`.

```
$ sudo modprobe rtc-ds1307
$ sudo bash
$ echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Jeżeli korzystasz ze starszej wersji Raspberry Pi (model B, rev1), zamień parametr `i2c-1` na `i2c-0`.

Teraz możesz uzyskać dostęp do zegara czasu rzeczywistego za pomocą polecenia:

```
$ sudo hwclock -r
sob, 01 sty 2000, 00:08:08 CEST -0.293998 seconds
```

Jak widzisz, zegar nie jest jeszcze ustawiony. Najpierw sprawdź, czy Raspberry Pi dysponuje prawidłowym czasem. Jeżeli ma ono połaczenie z internetem, to czas powinien być aktualizowany automatycznie. Sprawdź to za pomocą polecenia date.

```
$ date  
sob, 17 maj 2014, 21:59:12 CEST
```

Jeżeli wyświetlony czas jest nieprawidłowy, to ustawi go ręcznie za pomocą polecenia date (zobacz receptura 3.33). Aby przenieść dane dotyczące czasu systemowego Raspberry Pi do modułu zegara czasu rzeczywistego, użyj polecenia:

```
$ sudo hwclock -w
```

Jeżeli chcesz odczytać czas, zastosuj w powyższym poleceniu parametr -r:

```
$ sudo hwclock -r  
sob, 17 maj 2014, 21:59:12 CEST -0.179786 seconds
```

Zegar czasu rzeczywistego jest stosowany do określenia przez Raspberry Pi właściwego czasu systemowego podczas uruchamiania systemu Linux. Aby czas był wczytywany przez system automatycznie, należy go odpowiednio skonfigurować.

Edytuj plik /etc/modules (zastosuj w tym celu polecenie sudo nano /etc/modules) i na końcu listy modułów dodaj zapis rtc-ds1307. Jeżeli już wcześniej edytowałeś ten plik podczas konfiguracji magistrali I2C lub interfejsu SPI, to Twój plik może wyglądać następująco:

```
# /etc/modules: kernel modules to load at boot time.  
#  
# This file contains the names of kernel modules that should be loaded  
# at boot time, one per line. Lines beginning with "#" are ignored.  
# Parameters can be specified after the module name.  
snd-bcm2835  
i2c-bcm2708  
i2c-dev  
spidev  
rtc-ds1307
```

Teraz musisz jeszcze utworzyć dwa polecenia, które będą uruchamiane automatycznie podczas startu systemu. Dopiero po tych modyfikacjach czas systemowy będzie automatycznie ustawiany według zegara czasu rzeczywistego. Edytuj plik /etc/rc.local (zastosuj w tym celu polecenie sudo nano /etc/rc.local) i na końcu listy modułów dodaj zapis rtc-ds1307. Przed ostatnią linią o treści exit 0 umieść poniższe dwie linie kodu. Jeżeli korzystasz ze starej wersji Raspberry Pi (model B, rev1), to zamień parametr i2c-1 na i2c-0.

```
$ echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device  
$ sudo hwclock -s
```

Zmodyfikowany plik powinien wyglądać tak:

```
#  
# In order to enable or disable this script just change the execution  
# bits.  
#  
# By default this script does nothing.  
  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
printf "My IP address is %s\n" "$_IP"  
fi  
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device  
sudo hwclock -s  
exit 0
```

Po ponownym uruchomieniu Raspberry Pi powinno automatycznie ustawić datę i godzinę wskazaną przez zegar czasu rzeczywistego. Jeżeli Raspberry Pi będzie podłączone do internetu, dane dotyczące daty i godziny zostaną samoczynnie pobrane z sieci.

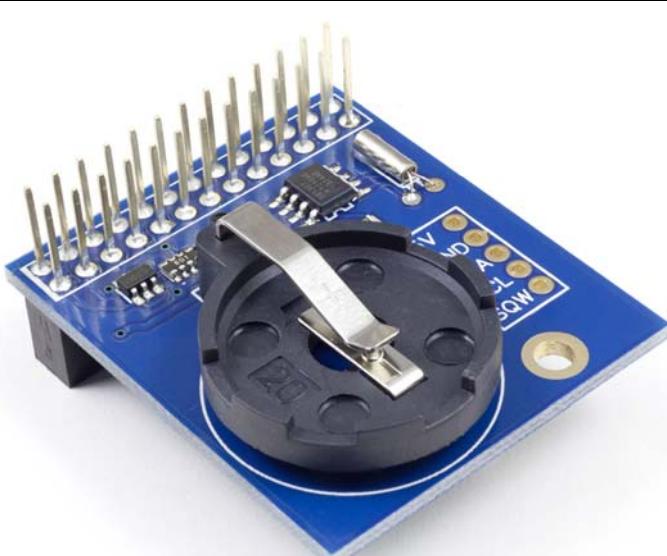
Omówienie

Raspberry Pi nie musi być wyposażone w zegar czasu rzeczywistego. System automatycznie pobiera dane dotyczące daty i godziny z internetu z serwera czasu, a następnie odpowiednio ustawia czas systemowy. Nie zawsze jednak możliwe jest uzyskanie połączenia z internetem. W takich przypadkach warto rozważyć zastosowanie zegara czasu rzeczywistego.

Płytkę aLaMode (zobacz receptura 14.13), będącą płytka interfejsową opartą na platformie Arduino, jest wyposażona w zegar czasu rzeczywistego. Jeśli masz tę płytke, możesz również korzystać ze wskazówek zawartych w tej recepturze.

Zobacz również

Firma AB Electronics posiada w swojej ofercie zgrabny moduł zegara czasu rzeczywistego wpinany bezpośrednio w gniazdo GPIO — <http://www.abelectronics.co.uk/products/3/Raspberry-Pi/15/RTC-Pi-Real-time-Clock-Module>. Moduł ten pokazano na rysunku 11.18.



Rysunek 11.18. Moduł zegara czasu rzeczywistego firmy AB Electronics

Niniejsza receptura została oparta na poradniku firmy Adafruit znajdującym się pod adresem <https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/>.

12.0. Wprowadzenie

W niniejszym rozdziale znajdziesz receptury dotyczące czujników pozwalających Raspberry Pi na pomiar między innymi takich wielkości jak temperatura i natężenie światła.

Raspberry Pi w przeciwieństwie do Arduino nie posiada wejść analogowych. Wiele czujników będziesz musiał podłączać za pośrednictwem dodatkowego przetwornika analogowo-cyfrowego. Na szczęście jest to dość prosty zabieg. Ponadto do obwodu zawierającego niektóre czujniki będziesz musiał włączać takie elementy jak rezystory i kondensatory.

Większość zaprezentowanych obwodów wymaga zastosowania płytka prototypowej oraz przewodów połączeniowych obustronnie zakończonych wtykami męskimi (zobacz receptura 8.10).

12.1. Korzystanie z czujników rezystancyjnych

Problem

Chcesz połączyć swoje Raspberry Pi z rezystorem i mierzyć pozycję, w jakiej jest on ustawiony (chcesz zmierzyć stawiany przez niego opór elektryczny).

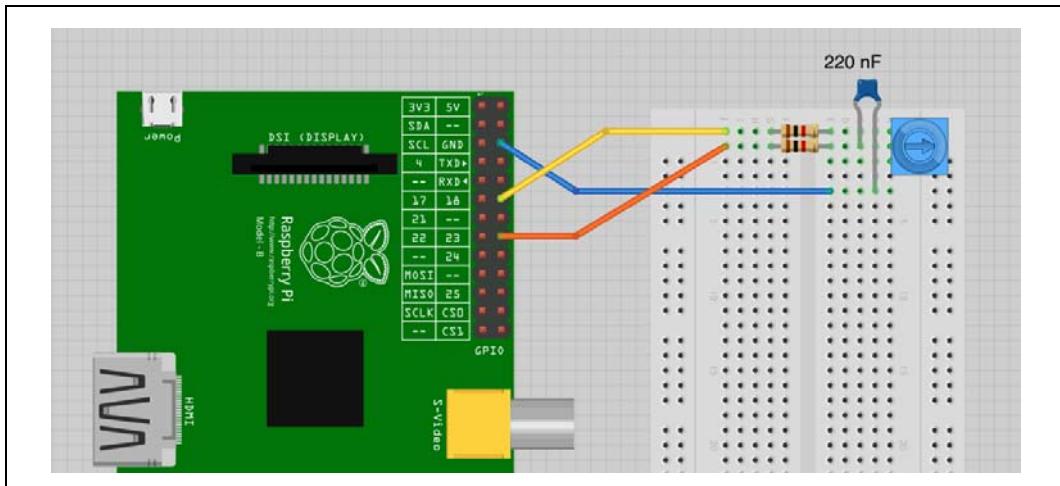
Rozwiążanie

Wykonanie pomiaru oporu wymaga zastosowania dwóch złączy gniazda GPIO Twojego Raspberry Pi, a także kondensatora i kilku rezystorów. W zaprezentowanym przykładzie będziesz my dokonywać pomiaru oporu elektrycznego generowanego przez mały potencjometr dostrojczy.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- potencjometru dostrojczego $10\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- dwóch rezystorów $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- kondensatora o pojemności 220 nF (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej przedstawiono na rysunku 12.1.



Rysunek 12.1. Pomiar rezystancji za pomocą Raspberry Pi

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *pot_step.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

a_pin = 18
b_pin = 23

def discharge():
    GPIO.setup(a_pin, GPIO.IN)
    GPIO.setup(b_pin, GPIO.OUT)
    GPIO.output(b_pin, False)
    time.sleep(0.005)

def charge_time():
    GPIO.setup(b_pin, GPIO.IN)
    GPIO.setup(a_pin, GPIO.OUT)
    count = 0
    GPIO.output(a_pin, True)
    while not GPIO.input(b_pin):
        count = count + 1
    return count

def analog_read():
    discharge()
    return charge_time()

while True:
    print(analog_read())
    time.sleep(1)
```

Uruchomienie programu powinno spowodować wyświetlenie na ekranie następujących elementów:

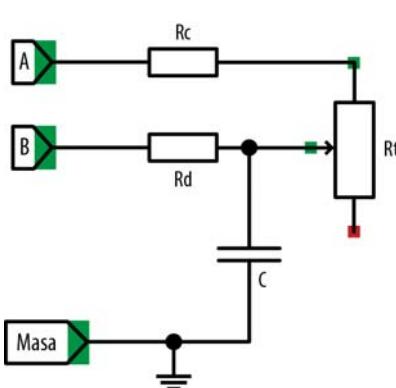
```
$ sudo python pot_step.py  
10  
12  
10  
10  
16  
23  
43  
53  
67  
72  
86  
105  
123  
143  
170
```

Poruszanie gałką potencjometru spowoduje wyświetlenie liczb w zakresie od około 10 do około 170.

Omówienie

Zanim wyjaśnię Ci działanie tego programu, muszę opisać technikę **odpowiedzi skokowej** zastosowaną do pomiaru rezystancji potencjometru.

Na rysunku 12.2 znajduje się schemat ideowy układu pomiarowego.



Rysunek 12.2. Pomiar rezystancji techniką odpowiedzi skokowej

Odpowiedź skokowa polega na badaniu odpowiedzi układu na zmianę sygnału generowanego na złączu wejściowym z niskiego na wysoki.

Kondensator jest zbiornikiem na prąd. Wraz ze wzrostem napięcia w obwodzie gromadzi się w nim coraz większy ładunek. Raspberry Pi nie posiada przetwornika analogowo-cyfrowego, a więc pomiar napięcia nie może być dokonany w sposób bezpośredni. Istnieje jednak możliwość pomiaru czasu ładowania kondensatora do poziomu 1,65 V — odbioru sygnału wysokiego na złączu wejściowym. Czas ładowania kondensatora zależy od oporu stawianego przez rezystor nastawny (R_t). Im mniejszy opór, tym szybciej wzrasta napięcie w obwodzie.

W celu przeprowadzenia prawidłowego pomiaru musisz rozładowywać kondensator przed każdym odczytem mierzonej wartości. Na rysunku 12.2 kondensator jest ładowany prądem płynącym z zacisku A poprzez kondensatory R_c i R_t . Zacisk B służy do rozładowywania kondensatora za pośrednictwem rezystora R_d . Rezystory R_c i R_d ograniczają prąd płynący przez kondensator podczas jego ładowania oraz rozładowywania.

W celu odczytania rezystancji należy najpierw rozładować kondensator za pośrednictwem rezystora R_d , a następnie naładować go za pośrednictwem rezystorów R_c i R_t .

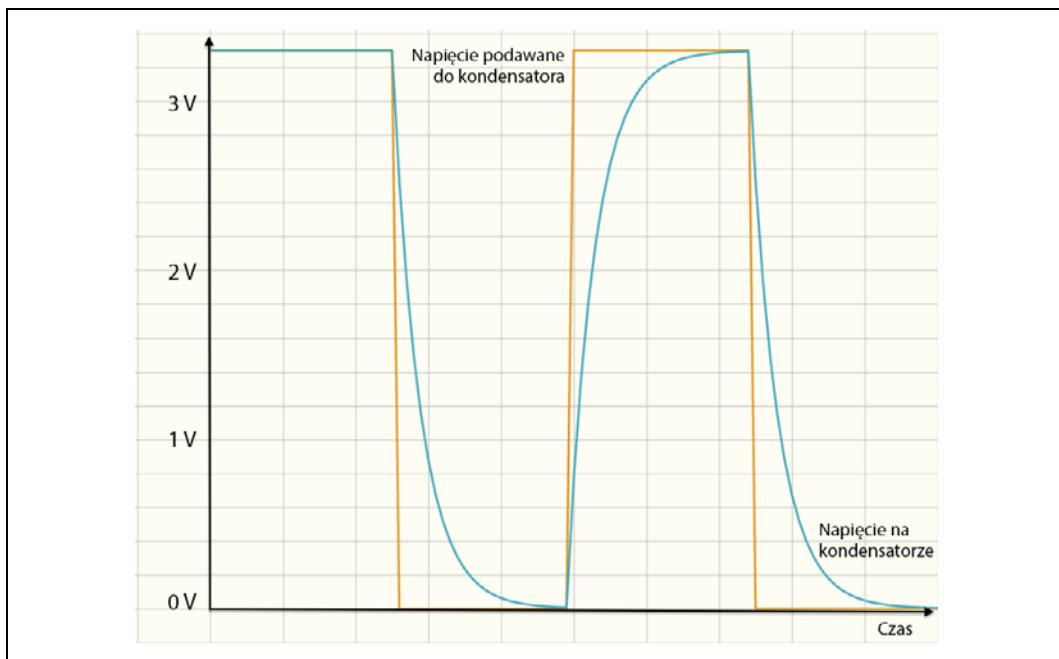
Podczas rozładowywania zacisk A (złącze o numerze 18 portu GPIO) działa w charakterze wejścia — wyłącza rezystory R_c i R_t z obwodu. Zacisk B (złącze o numerze 23 portu GPIO) jest następnie konfigurowany tak, aby działał w charakterze wyjścia podającego sygnał niski. Proces rozładowywania kondensatora trwa 5 milisekund.

Po rozładowaniu kondensatora można przystąpić do ładowania go — zacisk B działa wtedy jako wejście (zostaje on odłączony od obwodu), a zacisk A działa w charakterze wyjścia podającego sygnał wysoki (o napięciu 3,3 V). Kondensator C będzie w takich warunkach ładowany za pośrednictwem rezystorów R_c i R_t .

Funkcja `while` będzie zliczała czas do momentu podania na wejście (zacisk B) sygnału wysokiego (napięcie przekroczy poziom 1,65 V).

Po przekroczeniu określonego napięcia liczenie zostanie przerwane, a wartość wyświetlona na ekranie.

Na rysunku 12.3 pokazano wykres napięcia pomiędzy okładzinami kondensatora podczas przełączania sygnału wyjściowego generowanego przez Raspberry Pi pomiędzy stanem wysokim a niskim.



Rysunek 12.3. Ładowanie i rozładowywanie kondensatora

Widzisz, że na początku napięcie na kondensatorze wzrasta gwałtownie, ale gdy kondensator jest coraz bardziej naładowany, wzrost napięcia nie jest już aż tak gwałtowny. Na szczęście interesuje nas pomiar napięcia do poziomu 1,65 V. Ten fragment wykresu jest niemal linią prostą. Oznacza to, że czas potrzebny na uzyskanie takiego napięcia przez kondensator jest prawie proporcjonalny do rezystancji Rt , a tym samym — zależny od położenia gałki potencjometru.

Technika ta nie jest zbyt dokładna, ale jest łatwa do zastosowania, a komponenty niezbędne do zbudowania układu są bardzo tanie.

Zobacz również

Odpowiedź skokową można stosować do wszystkich rezystancyjnych światłomierzy (zobacz receptura 12.2), a nawet detektora gazu (zobacz receptura 12.3).

Jeżeli interesuje Cię dokładniejszy pomiar rezystancji potencjometru dinstrojczego, zajrzyj do receptury 12.4. Zastosowano tam przetwornik analogowo-cyfrowy.

12.2. Pomiar jasności światła

Problem

Chcesz mierzyć jasność światła padającego na fotorezystor podłączony do Raspberry Pi.

Rozwiążanie

Skorzystaj z wiadomości umieszczonych w recepturze 12.1. Zastosuj ten sam program, a w obwodzie w miejsce potencjometru dinstrojczego wstaw fotorezystor.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- fotorezystora (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- dwóch rezystorów 1 k Ω (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- kondensatora o pojemności 220 nF (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

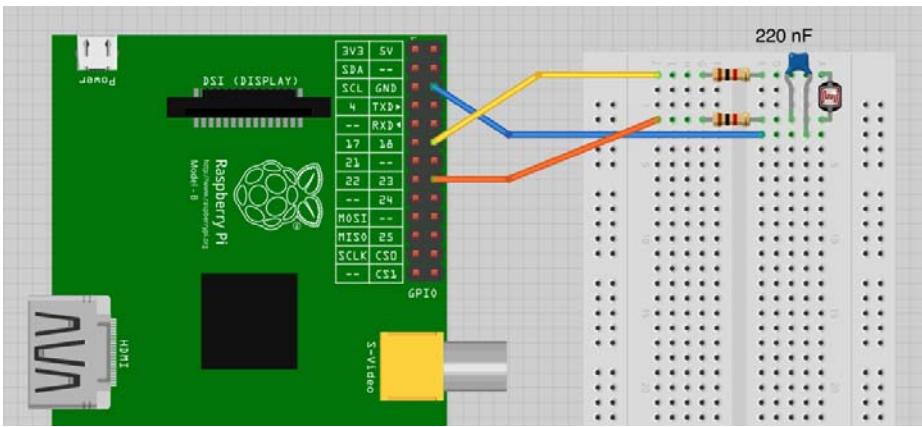
Ułożenie elementów obwodu na płytce prototypowej przedstawiono na rysunku 12.4.

Uruchom program *pot_step.py* omówiony w recepturze 12.1. Wartości generowane przez program będą zależeć od ilości światła padającego na fotorezystor.

Omówienie

Fotorezystor jest rezystorem, który stawia opór zależny od ilości światła wpadającego przez przezroczyste okienko wykonane w jego obudowie. Im więcej światła, tym mniejszy opór stawiany przez ten element obwodu. Zwykle opór wahana się od 1 k Ω (w świetle o dużej jasności) do 100 k Ω (w całkowitej ciemności).

Opisany czujnik niestety może dostarczyć tylko niezbyt dokładne informacje o jasności światła.



Rysunek 12.4. Pomiar jasności światła za pomocą Raspberry Pi

Zobacz również

Fotorezystor może również zostać podłączony do przetwornika analogowo-cyfrowego przedstawionego w recepturze 12.4.

12.3. Wykrywanie metanu

Problem

Chcesz mierzyć stężenie gazu za pomocą czujnika metanu.

Rozwiążanie

Rezystywne czujniki gazu takiego jak np. metan są tanie i można je bezproblemowo podłączyć do Raspberry Pi. Możesz odczytywać stężenie gazu za pomocą techniki odpowiedzi skokowej opisanej w recepturze 12.1.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- czujnika metanu (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- dwóch rezystorów 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- kondensatora o pojemności 220 nF (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

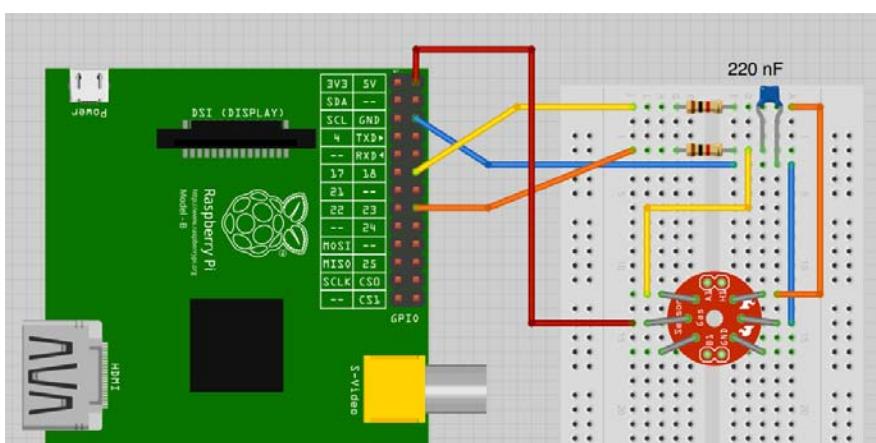
Czujnik posiada element grzejny wymagający zasilania prądem o napięciu 5 V i natężeniu 150 mA. Czujnik można podłączyć bezpośrednio do Raspberry Pi, pod warunkiem że jego zasilacz potrafi dostarczyć dodatkowe 150 mA.

Czujnik ma dość grube wyprowadzenia. Nie pozwalają one na jego montaż bezpośrednio w płytce prototypowej. Możesz rozwiązać ten problem, przylutowując cieńszy drucik do każdego wyprowadzenia detektora (zobacz rysunek 12.5). Innym rozwiązaniem tego problemu jest zakup specjalnej płytki przeznaczonej do testowania czujników.Więcej informacji na temat tego produktu znajdziesz na stronie <https://www.sparkfun.com/products/8891>.

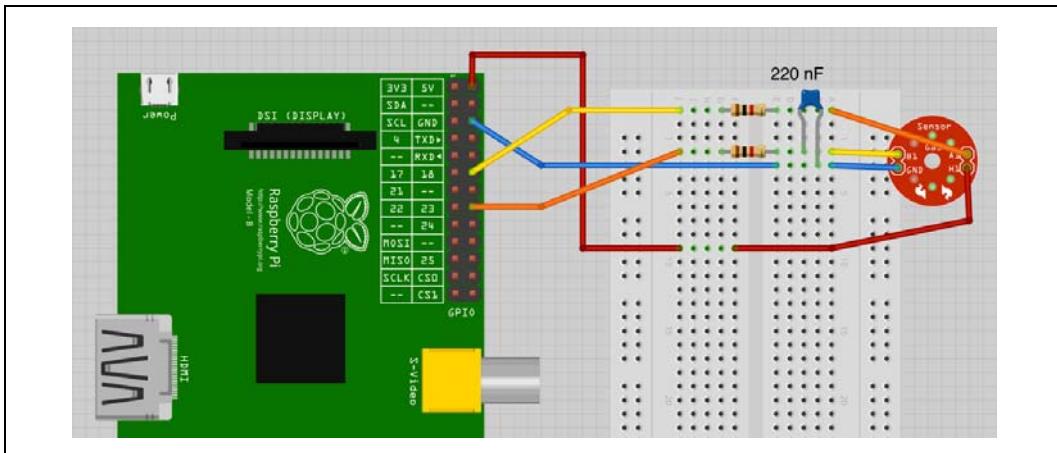


Rysunek 12.5. Dodatkowe druciki przylutowane do czujnika gazu

Połącz ze sobą wszystkie elementy obwodu, korzystając z rysunku 12.6 (jeżeli zakupiłeś płytke testową firmy SparkFun) lub rysunku 12.7 (jeżeli przylutowałeś cieńsze druciki do wyprowadzeń czujnika).



Rysunek 12.6. Podłączanie czujnika metanu do Raspberry Pi za pośrednictwem płytki testowej



Rysunek 12.7. Czujnik metanu podłączony bezpośrednio do Raspberry Pi

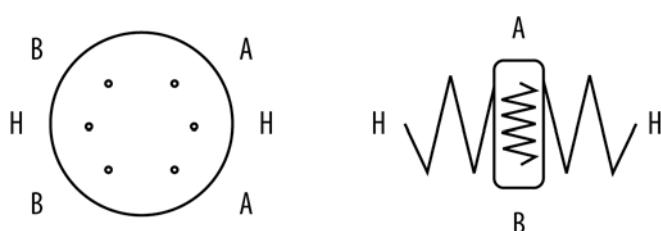
Na rysunku 12.7 czujnik oznaczono tym samym symbolem co płytkę testową. Jeżeli przyjrzysz się uważnie wykonanym połączeniom, zauważysz, że wykonano je z sześcioma wyrowadzeniami, a nie z czterema, tak jak miało to miejsce w przypadku płytki testowej.

Do obsługi czujnika możesz zastosować program przedstawiony w recepturze 12.1. Jeżeli zaczniesz robić wydechy w kierunku czujnika metanu, to odczytywane przez Raspberry Pi wartości powinny maleć.

Omówienie

Oczywiście czujnik metanu można zastosować w celu zbudowania wykrywacza pierdnięć. Poważniejszym projektem wykorzystującym ten moduł jest obwód wykrywający ulatnianie się gazu ziemnego. Wyobraź sobie Raspberry Pi monitorujące budynek mieszkalny za pomocą wielu czujników. Będąc na wakacjach, mógłbyś zostać powiadomiony pocztą elektroniczną o zaistnieniu jakiegoś niebezpieczeństwa.

Czujnik przedstawiony na rysunku 12.8 jest wyposażony, tak jak wszystkie czujniki tego typu, w element podgrzewający powierzchnię pokrytą substancją katalityczną odpowiednią dla wykrywanego gazu. Gaz reaguje z tą substancją, w wyniku czego zmienia się oporność elektryczna warstwy katalitycznej.



Rysunek 12.8. Czujnik metanu

Zarówno element grzejny, jak i warstwa detekcyjna z punktu widzenia elektroniki są po prostu rezystorami. Nie są więc one spolaryzowane.

Omawiany czujnik jest najbardziej wrażliwy na metan, jednak będzie on również, choć w mniejszym stopniu, wykrywał obecność innych gazów. Właśnie dlatego poprawność działania czujnika sprawdzaliśmy, dmuchając na niego (zdrowe płuca człowieka nie wydają z siebie metanu). Dmuchając na czujnik, obniżamy jego temperaturę, co też może wpływać na jego pracę.

Zobacz również

Dokumentację techniczną użytego przeze mnie czujnika możesz znaleźć pod adresem <https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-4.pdf>. Znajdziesz tam wiele przydatnych informacji dotyczących wykrywalności różnych gazów przez to urządzenie.

Istnieje wiele różnych tanich czujników przeznaczonych do wykrywania obecności różnych gazów. Listę czujników oferowanych przez firmę SparkFun znajdziesz na stronie <https://www.sparkfun.com/categories/146>.

12.4. Pomiar napięcia

Problem

Chcesz korzystać z analogowego woltomierza.

Rozwiążanie

Złącze GPIO jest wyposażone tylko w cyfrowe wejścia. Aby połączyć Raspberry Pi z analogowym woltomierzem, musisz skorzystać z przetwornika analogowo-cyfrowego.

W swoim projekcie zastosuj układ MCP3008, będący ośmiokanałowym przetwornikiem analogowo-cyfrowym. Układ ten posiada osiem wejść analogowych, a więc można podłączyć do niego nawet osiem przyrządów pomiarowych. Komunikację układu z Raspberry Pi zapewnia interfejs SPI.

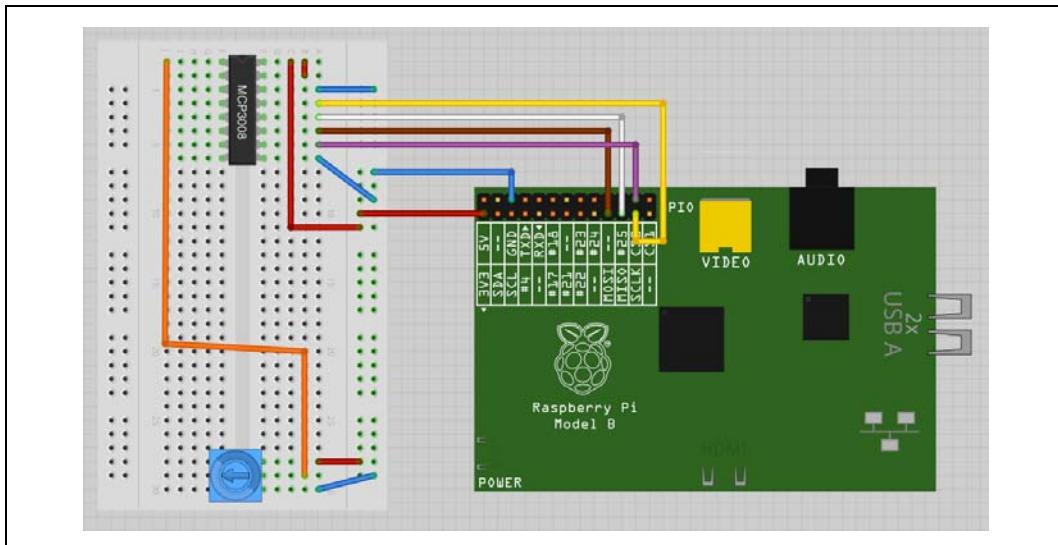
Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu scalonego MCP3008 składającego się z ośmiu przetworników A/C (zobacz sekcja „Układy scalone” w dodatku A),
- potencjometru dostrojczego $10\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Na rysunku 12.9 przedstawiono układ podłączony do Raspberry Pi za pośrednictwem płytki prototypowej. Upewnij się, że nie wstawiłeś odwrotnie chipu. Małe wycięcie znajdujące się na wierzchu obudowy tego układu powinno być zwrócone ku górze płytki prototypowej.

Jedno ze skrajnych wyprowadzeń potencjometru jest podłączone do napięcia 3,3 V, a drugie do masy — w ten sposób na środkowym złączu tego elementu możliwe jest uzyskanie dowolnego napięcia w zakresie od 0 do 3,3 V.

Zanim wypróbowujesz program, upewnij się, że uruchomłeś obsługę interfejsu SPI oraz zainstalowałeś bibliotekę pozwalającą na jego użycie (zobacz receptura 8.6).



Rysunek 12.9. Układ MCP3008 podłączony do Raspberry Pi

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *adc_test.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import spidev, time
spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

while True:
    reading = analog_read(0)
    voltage = reading * 3.3 / 1024
    print("Odczyt=%d\tU=%f" % (reading, voltage))
    time.sleep(1)
```

Najbardziej interesującą częścią tego programu jest funkcja *analog_read*. Do funkcji tej przekazywany jest parametr o wartości od 0 do 7 — określa on, stan którego z ośmiu analogowych wejść znajdujących się po lewej stronie układu powinien zostać odczytany.

W wyniku operowania bitami do układu MCP3008 wysyłane jest żądanie dotyczące właściwego kanału, a następnie odczytywane są dane wynikowe:

```
$ sudo python adc_test.py
Odczyt=0      U=0.000000
Odczyt=126    U=0.406055
Odczyt=221    U=0.712207
Odczyt=305    U=0.982910
Odczyt=431    U=1.388965
Odczyt=527    U=1.698340
Odczyt=724    U=2.333203
Odczyt=927    U=2.987402
Odczyt=10     U=3.296777
Odczyt=1020   U=3.287109
Odczyt=1022   U=3.293555
```

Omówienie

Układ MCP3008 wyposażono w dziesięciobitowe przetworniki A/C, a więc wielkości odczytywane za ich pomocą przybierają wartości w zakresie od 0 do 1023. Przedstawiony program zamienia te wartości na napięcie, mnożąc je przez górną wartość zakresu pomiarowego (3,3 V), a następnie dzieląc przez 1024.

Odczytując dane z wszystkich ośmiu czujników układu MCP3008, możesz korzystać z technik przedstawionych w kolejnych recepturach.

Z układem MCP3008 możesz również łączyć czujniki rezystancyjne. Po połączeniu z rezystorem charakteryzującym się stałą wartością oporu będą one tworzyć dzielniki napięcia (zobacz receptura 12.6 i rysunek 12.11).

Zobacz również

Jeżeli interesuje Cię tylko wykrywanie obrotu gałką, to zamiast potencjometru możesz zastosować koder obrotowy (zobacz receptura 11.7).

Położenie potencjometru może być również określone bez układu przetwornika A/C. Wystarczy zastosować technikę odpowiedzi skokowej opisaną w recepturze 12.1.

Zajrzyj również do dokumentacji układu MCP3008 znajdującej się pod adresem <http://www.microchip.com/downloads/en/DeviceDoc/21295d.pdf>.

12.5. Stosowanie dzielnika napięcia

Problem

Chcesz dokonać pomiaru napięcia większego od 3,3 V, co jest maksymalną dopuszczalną wartością napięcia mierzonego przez układ MCP3008 (zobacz receptura 12.4).

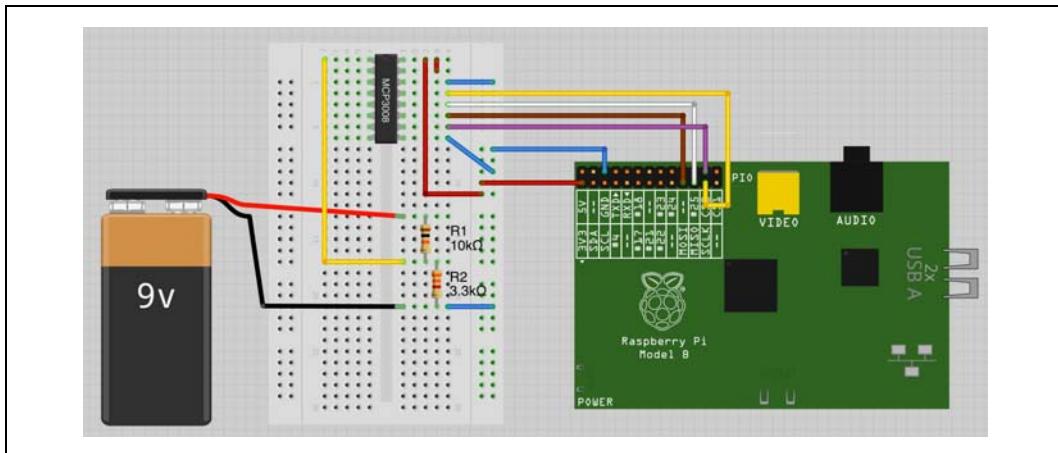
Rozwiążanie

Obniż napięcie, dostosowując je do zakresu układu pomiarowego za pomocą pary rezystorów tworzącej dzielnik napięcia.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu scalonego MCP3008 składającego się z ośmiu przetworników A/C (zobacz sekcja „Układy scalone” w dodatku A),
- rezystora 10 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- rezystora 3,3 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- przewodu zakończonego klipsem do baterii 9 V.

Na rysunku 12.10 przedstawiono schemat układu pomiarowego podłączonego do Raspberry Pi za pośrednictwem płytki prototypowej. Układ ten jest przeznaczony do pomiaru napięcia podłączonej do niego baterii.



Rysunek 12.10. Obniżanie napięcia na wejściu analogowym



Nigdy nie korzystaj z tego obwodu do pomiaru prądu przemiennego, a zwłaszcza prądu przemiennego o wysokim napięciu. Układ MCP3008 służy tylko do pomiaru prądu stałego o niskim napięciu.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *adc_scaled.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import spidev

R1 = 10000.0
R2 = 3300.0

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

reading = analog_read(0)
voltage_adc = reading * 3.3 / 1024
voltage_actual = voltage_adc / (R2 / (R1 + R2))
print("U baterii=" + str(voltage_actual))
```

Program ten jest bardzo podobny do tego, który został omówiony w recepturze 12.4. Różni się on jedynie algorytmem skalującym uzależnionym od oporu stawianego przez dwa rezystory. Wartości tych rezystancji są przechowywane przez zmienne *R1* i *R2*.

Po uruchomieniu programu zostanie wyświetlony komunikat informujący Cię o napięciu pomiędzy zaciskami baterii:

```
$ sudo python adc_scaled.py
U baterii =8.62421875
```

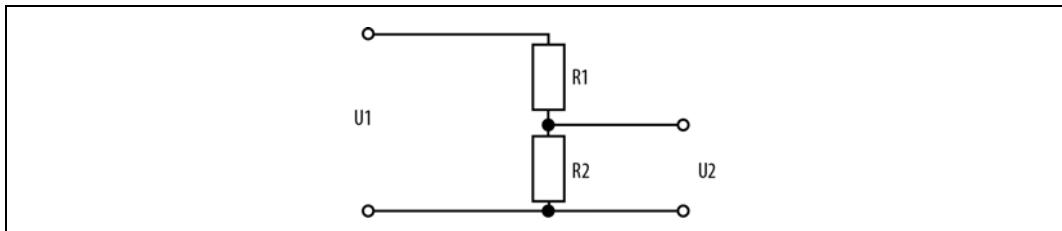


Przed podłączeniem do układu pomiarowego zacisków, pomiędzy którymi istnieje różnica potencjałów większa niż 9 V, zapoznaj się z poniższą sekcją „Omówienie”. W przeciwnym wypadku możesz uszkodzić układ MCP3008.

Omówienie

Układ pokazany na rysunku 12.11 jest nazywany **dzielnikiem napięcia**. Napięcie wyjściowe (U_2) można obliczyć na podstawie napięcia wejściowego (U_1) oraz oporu stawianego przez rezystory (R_1 i R_2) ze wzoru:

$$U_2 = U_1 \times R_2 / (R_1 + R_2)$$



Rysunek 12.11. Dzielnik napięcia

Z podanego wzoru wynika, że jeżeli rezystory R_1 i R_2 charakteryzowałyby się takim samym oporem (np. po $1\text{ k}\Omega$ każdy), to U_2 byłoby o połowę mniejsze od U_1 .

Wybierając rezystory R_1 i R_2 , musisz brać pod uwagę natężenie płynącego przez nie prądu. Natężenie to wyliczasz ze wzoru $I = U_1 / (R_1 + R_2)$. W zaprezentowanym wcześniej przykładzie R_1 wynosiło $10\text{ k}\Omega$, a $R_2 = 3,3\text{ k}\Omega$. Natężenie prądu będzie więc wynosić $9\text{ V} / 13,3\text{ k}\Omega = 0,68\text{ mA}$. Jest to dość niska wartość, ale pobieranie nawet tak niewielkiego prądu w końcu doprowadzi do rozładowania baterii. Gdy nie używasz obwodu pomiarowego, odłącz od niego baterię.

Zobacz również

Jeżeli nie chcesz wykonywać obliczeń, skorzystaj z internetowego kalkulatora znajdującego się pod adresem: <http://www.electronics2000.co.uk/calc/potential-divider-calculator.php>.

Dzielnik napięcia jest również stosowany w obwodzie zamieniającym wahania oporu elektrycznego na wahania napięcia. Zabieg taki pozwala na podłączenie czujnika rezystancyjnego do przetwornika A/C (zobacz receptura 12.6).

12.6. Podłączanie rezystancyjnego czujnika do przetwornika analogowo-cyfrowego

Problem

Dysponujesz czujnikiem rezystancyjnym i chcesz podłączyć go do układu zawierającego przetwornik A/C MCP3008.

Rozwiążanie

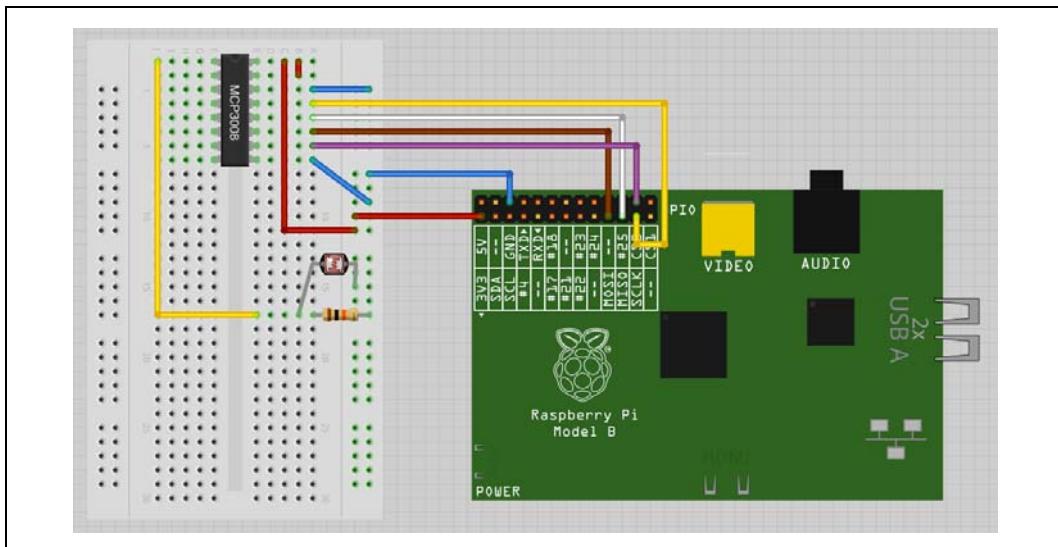
Skorzystaj z układu dzielnika napięcia — jeden z jego rezistorów zastąp rezystancyjnym czujnikiem. W ten sposób wahania oporu czujnika zostaną zamienione na wahania napięcia prądu, które będą odbierane przez przetwornik A/C.

W celu zademonstrowania takiego układu zajmijmy się światłomierzem omówionym w recepturze 12.2. Zamiast techniki odpowiedzi krokowej zastosujemy układ MCP3008.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu scalonego MCP3008 składającego się z ośmiu przetworników A/C (zobacz sekcja „Układy scalone” w dodatku A),
- rezystora $10\text{ k}\Omega$ (zobacz sekcję „Rezystory i kondensatory” w dodatku A),
- fotorezystora (zobacz sekcję „Rezystory i kondensatory” w dodatku A).

Na rysunku 12.12 przedstawiono schemat układu pomiarowego podłączonego do Raspberry Pi za pośrednictwem płytki prototypowej.



Rysunek 12.12. Fotorezyistor podłączony do przetwornika A/C

Możesz skorzystać z programu przedstawionego w recepturze 12.4 (*adc_test.py*). Poruszanie dlonią pomiędzy fotorezystorem a źródłem światła spowoduje zmiany wyświetlanych przez program wartości. Przed uruchomieniem aplikacji należy skonfigurować interfejs SPI. Jeżeli jeszcze tego nie zrobiłeś, to wykonaj czynności opisane w recepturze 8.6.

```
$ sudo python adc_test.py
Odczyt=341      U=1.098926
Odczyt=342      U=1.102148
Odczyt=227      U=0.731543
Odczyt=81       U=0.261035
Odczyt=86       U=0.277148
```

Wartości uzyskane w wyniku działania programu zależą od zastosowanego fotorezystora, powinny jednak ulegać zmianie zależnie od natężenia światła padającego na ten element układu.

Omówienie

Wybór wartości rezystora zastosowanego w obwodzie nie jest tutaj szczególnie krytyczny. Wybór niewłaściwego rezystora ograniczy zakres pomiarowy czujnika. Zdecyduj się na rezistor o wartości znajdującej się pomiędzy minimalną i maksymalną rezystancją czujnika. Element ten warto dobierać na drodze eksperymentów — wykonywania pomiarów w interesującym Cię zakresie. Jeżeli masz wątpliwości, zacznij od zastosowania rezystora charakteryzującego się oporem 10 kΩ.

Do zaprezentowanego układu w miejsce fotorezystora możesz włączyć praktycznie dowolny czujnik rezystancyjny — np. czujnik gazu wspomniany w recepturze 12.3.

Zobacz również

Jeżeli chcesz mierzyć jasność światła bez pośrednictwa przetwornika A/C, zajrzyj do receptury 12.2. W recepturze 12.8 przedstawiono obwód korzystający z więcej niż jednego kanału układu MCP3008.

12.7. Pomiar temperatury za pomocą przetwornika analogowo-cyfrowego

Problem

Chcesz mierzyć temperaturę za pomocą czujnika TMP36 podłączonego do przetwornika A/C.

Rozwiązanie

Zastosuj układ scalony MCP3008.

Jeżeli nie używasz więcej niż jednego analogowego kanału, rozważ zastosowanie cyfrowego termometru w postaci układu DS18B20. Jest on dokładniejszy i nie wymaga stosowania zewnętrznego przetwornika analogowo-cyfrowego (zobacz receptura 12.9).

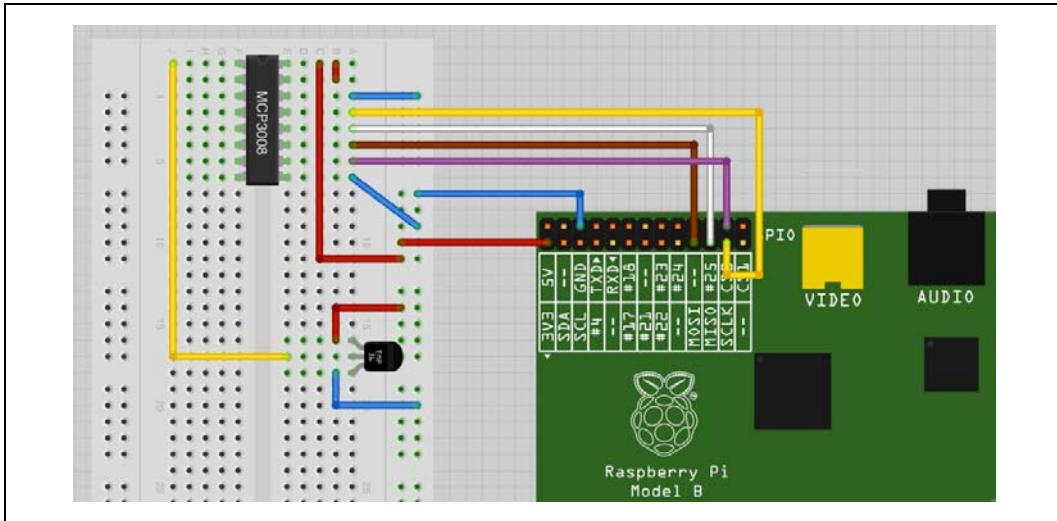
Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu scalonego MCP3008 składającego się z ośmiu przetworników A/C (zobacz sekcja „Układy scalone” w dodatku A),
- czujnika temperatury TMP36 (zobacz sekcja „Układy scalone” w dodatku A).

Na rysunku 12.13 przedstawiono schemat układu pomiarowego podłączonego do Raspberry Pi za pośrednictwem płytki prototypowej.

Upewnij się, że układ TMP36 nie jest podłączony odwrotnie. Jedna nóżka tego układu jest prosta, a druga zagięta.

Musisz skonfigurować interfejs SPI swojego Raspberry Pi. Jeżeli jeszcze tego nie zrobiłeś, wykonaj polecenia opisane w recepturze 8.6.



Rysunek 12.13. Układ TMP36 podłączony do przetwornika A/C

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `adc_tmp36.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import spidev, time

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

while True:
    reading = analog_read(0)
    voltage = reading * 3.3 / 1024
    temp_c = voltage * 100 - 50
    temp_f = temp_c * 9.0 / 5.0 + 32
    print("Temp. C=%f\tTemp. f=%f" % (temp_c, temp_f))
    time.sleep(1)
```

Powyższy program napisano w oparciu o program przedstawiony w recepturze 12.4. Dodano w nim sekcję przeliczającą mierzoną temperaturę na stopnie Celsjusza i skalę Fahrenheita:

\$ sudo python adc_tmp36.py	
Temp. C=19.287109	Temp. f=66.716797
Temp. C=18.642578	Temp. f=65.556641
Temp. C=18.964844	Temp. f=66.136719
Temp. C=20.253906	Temp. f=68.457031
Temp. C=20.898438	Temp. f=69.617188
Temp. C=20.576172	Temp. f=69.037109
Temp. C=21.865234	Temp. f=71.357422
Temp. C=23.154297	Temp. f=73.677734
Temp. C=23.476562	Temp. f=74.257812
Temp. C=23.476562	Temp. f=74.257812
Temp. C=24.121094	Temp. f=75.417969
Temp. C=24.443359	Temp. f=75.998047
Temp. C=25.087891	Temp. f=77.158203

Omówienie

Układ TMP36 generuje napięcie wyjściowe proporcjonalne do temperatury. Zgodnie z dokumentacją układu TMP36 temperaturę wyrażoną w stopniach Celsjusza można obliczyć, mnożąc napięcie w woltach przez 100, a następnie od uzyskanej wartości odejmując 50.

Układ TMP36 świetnie sprawdza się do mierzenia przybliżonej temperatury. Błąd pomiaru może sięgać 2%. Parametr ten ulegnie pogorszeniu, jeżeli zastosujesz długie kable połączeniowe. Możesz oczywiście wykalibrować posiadany egzemplarz tego układu, jednak w celu uzyskania większej dokładności pomiarów zastosuj układ DS18B20 (zobacz receptura 12.9). Charakteryzuje się on niepewnością pomiarową 0,5% w zakresie temperatur od -10 do +85 stopni Celsjusza. Jest to urządzenie cyfrowe, a więc zastosowanie długich kabli połączeniowych nie powinno wpływać na dokładność pomiaru.

Zobacz również

Zajrzyj do dokumentacji układu TMP36: http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/Tmp/TMP35_36_37.pdf.

12.8. Pomiar przyspieszenia

Problem

Do swojego Raspberry Pi chcesz podłączyć trójosiowy przyspieszeniomierz.

Rozwiążanie

Podłącz analogowy przyspieszeniomierz za pomocą przetwornika A/C MCP3008. Pozwoli to na odczyt sygnałów generowanych przez przyspieszeniomierz dla osi X, Y i Z.

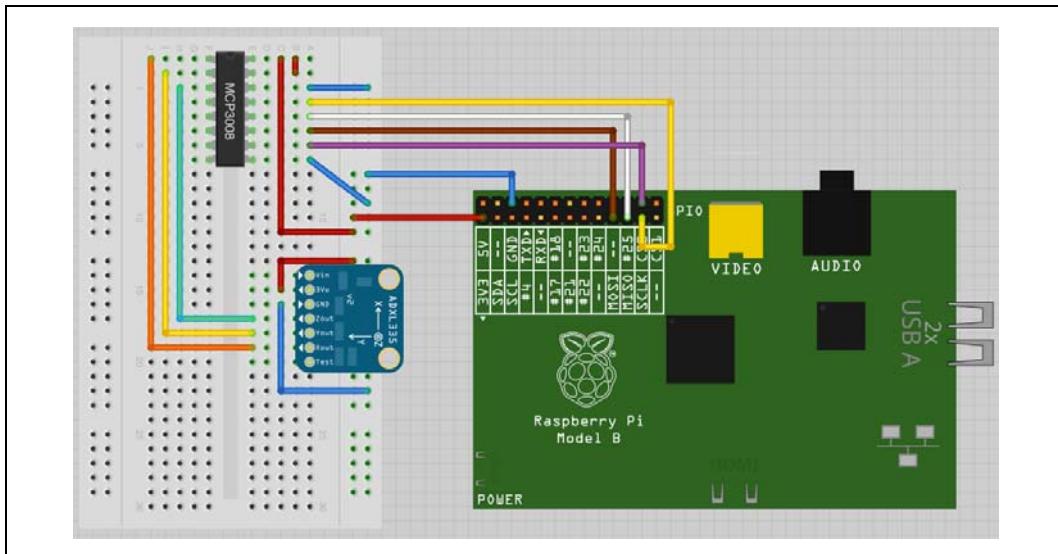
Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- układu scalonego MCP3008 składającego się z ośmiu przetworników A/C (zobacz sekcję „Układy scalone” w dodatku A),
- trójosiowego przyspieszeniomierza ADXL335 (zobacz sekcja „Moduły” w dodatku A).

Na rysunku 12.14 przedstawiono przyspieszeniomierz podłączony do Raspberry Pi za pośrednictwem płytki prototypowej. Element ten korzysta z trzech kanałów przetwornika A/C w celu przekazywania do Raspberry Pi informacji dotyczących przyspieszenia w płaszczyznach osi X, Y i Z.

Musisz skonfigurować interfejs SPI swojego Raspberry Pi. Jeżeli jeszczе tego nie zrobiłeś, wykonaj polecenia opisane w recepturze 8.6.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *adc_accelerometer.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.



Rysunek 12.14. Trójosiowy przyspieszeniomierz podłączony do przetwornika A/C

```
import spidev, time

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

while True:
    x = analog_read(0)
    y = analog_read(1)
    z = analog_read(2)
    print("X=%d\Y=%d\Z=%d" % (x, y, z))
    time.sleep(1)
```

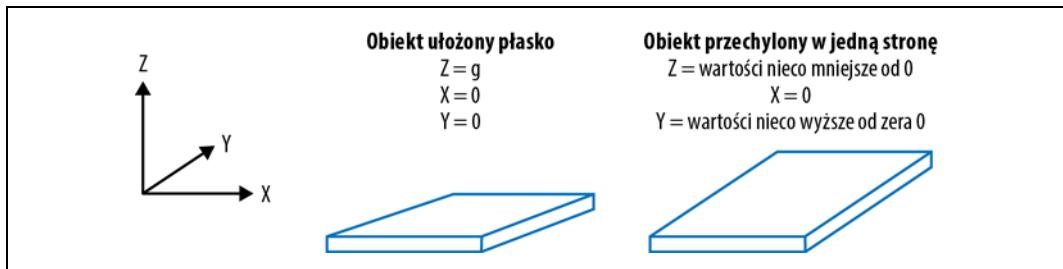
Program odczytuje dane dotyczące sił działających w trzech płaszczyznach i wyświetla je na ekranie:

```
$ sudo python adc_accelerometer.py
X=508  Y=503  Z=626
X=508  Y=504  Z=624
X=506  Y=505  Z=627
X=423  Y=517  Z=579
X=411  Y=513  Z=548
X=532  Y=510  Z=623
X=609  Y=518  Z=495
X=607  Y=521  Z=496
X=610  Y=513  Z=499
```

Pierwsze trzy odczyty wartości miały miejsce, gdy płytka leżała nieruchomo. Następnie płytka została poruszona. Widzisz, że wartości X spadły. Poruszenie płytka w kierunku przeciwnym spowodowało wzrost wartości X.

Omówienie

Przyspieszeniomierz najczęściej jest stosowany do wykrycia przechylenia. Urządzenie to można zastosować w tym celu, ponieważ na osi Z wpływa w dużym stopniu siła grawitacji (zobacz rysunek 12.15).



Rysunek 12.15. Wykrywanie przechylenia za pomocą przyspieszeniomierza

Po przechyleniu przyspieszeniomierza na którąś ze stron siła grawitacji zaczyna wpływać na pozostałe osie tego przyrządu pomiarowego.

Mogliśmy zastosować to zjawisko do wykrycia przechylenia przyspieszeniomierza. Poniższy program (*tilt.py*) korzysta z omawianej zależności.

```
import spidev, time

spi = spidev.SpiDev()
spi.open(0,0)

def analog_read(channel):
    r = spi.xfer2([1, (8 + channel) << 4, 0])
    adc_out = ((r[1]&3) << 8) + r[2]
    return adc_out

while True:
    x = analog_read(0)
    y = analog_read(1)
    z = analog_read(2)
    if x < 450:
        print("W lewo")
    elif x > 550:
        print("W prawo")
    elif y < 450:
        print("Do tylu")
    elif y > 550:
        print("Do przodu")
    time.sleep(0.2)
```

Program po uruchomieniu będzie wyświetlał komunikaty informujące o kierunku przechylenia czujnika. Można go zastosować w celu właściwego wypoziomowania kamery zainstalowanej na ruchomym robocie.

```
$ sudo python tilt.py
W lewo
W lewo
W prawo
Do przodu
Do przodu
Do tylu
Do tylu
```

Zobacz również

Dokumentację zastosowanego przyspieszeniomierza znajdziesz pod adresem: http://www.analog.com/static/imported-files/data_sheets/ADXL335.pdf.

Istnieje wiele różnych modułów przyspieszeniomierza. Mogą one generować różne wartości odczytywanych danych. Upewnij się, że napięcie sygnału generowanego przez zastosowany moduł nie przekracza 3,3 V.

12.9. Pomiar temperatury za pomocą cyfrowego czujnika

Problem

Chcesz mierzyć temperaturę za pomocą dokładnego, cyfrowego czujnika.

Rozwiązanie

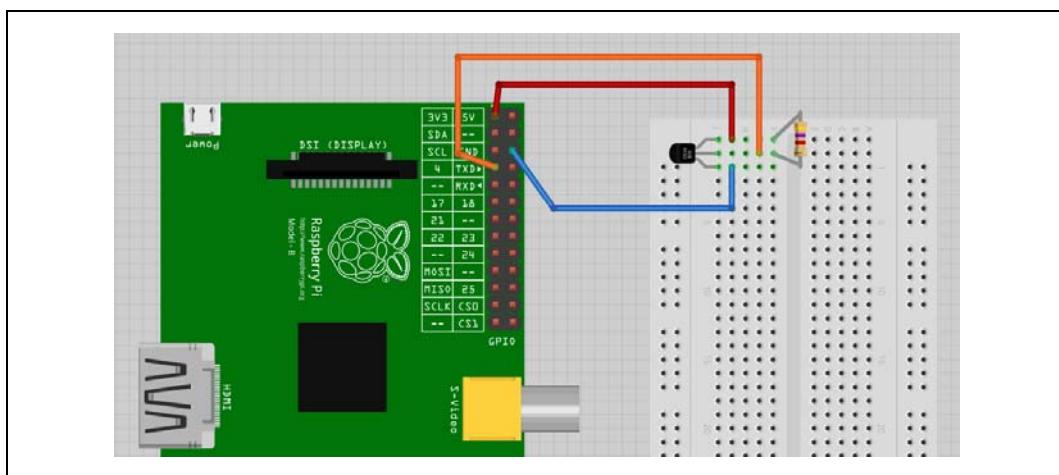
Zastosuj cyfrowy czujnik temperatury DS18B20. Układ ten charakteryzuje się większą dokładnością pomiarową od przedstawionego w recepturze 12.7 układu TMP36. Ponadto jest on urządzeniem cyfrowym, a więc nie wymaga przetwornika A/C.

Dane są wysyłane przez urządzenie za pomocą jednego przewodu, jednak jego podłączenie do Raspberry Pi wymaga zastosowania większej liczby kabli.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- czujnika temperatury DS18B20 (zobacz sekcja „Układy scalone” w dodatku A),
- rezystora 4,7 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Połącz elementy obwodu, korzystając z rysunku 12.16. Upewnij się, że nie zamontowałeś układu DS18B20 odwrotnie.



Rysunek 12.16. Układ DS18B20 podłączony do Raspberry Pi

System operacyjny Occidentalis oraz nowsze wersje systemu Raspbian mają domyślnie włączoną obsługę jednoprzewodowego interfejsu, w jaki wyposażony jest układ DS18B20. Jeżeli poniższy program nie będzie działał, uruchom polecenie:

```
$ sudo apt-get upgrade
```

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *temp_DS18B20.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import os, glob, time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
    return temp_c, temp_f

while True:
    print("temp. C=%f\ntemp. F=%f" % read_temp())
    time.sleep(1)
```

Uruchomiony program wyświetla temperatury w stopniach Celsjusza oraz w skali Fahrenheita:

```
$ python temp_DS18B20.py
temp. C=25.187000  temp. F=77.336600
temp. C=25.125000  temp. F=77.225000
temp. C=25.062000  temp. F=77.111600
temp. C=26.312000  temp. F=79.361600
temp. C=27.875000  temp. F=82.175000
temp. C=28.875000  temp. F=83.975000
```

Omówienie

Na pierwszy rzut oka program może się wydawać nieco dziwny. Odczytując dane z modułu DS18B20, korzystamy z techniki podobnej do interfejsu stosowanego do odczytu plików. Pliki, z których korzysta interfejs, będą się zawsze znajdować w folderze */sys/bus/w1/devices/*, a ścieżka plików będzie się zaczynać liczbą 28. Kolejne czujniki będą się różnić pozostałą częścią ścieżki dostępu. Kod zakłada, że mamy tylko jeden czujnik, i odnajduje pierwszy folder o nazwie rozpoczynającej się liczbą 28. W tym folderze będzie się znajdował plik o nazwie *w1_slave*. Program go otworzy i odczyta dane o temperaturze.

W rzeczywistości czujnik zwraca tekst umieszczony w łańcuchu takim jak ten:

```
81 01 4b 46 7f ff 0f 10 71 : crc=71 YES
81 01 4b 46 7f ff 0f 10 71 t=24062
```

Dalsza część programu odczytuje z tego łańcucha dane o temperaturze, która zostanie wyświetlona z dokładnością do jednej tysięcznej stopnia Celsjusza.

Funkcja `read_temp` przelicza dane dotyczące temperatury, tak aby wyrazić ją w skali Fahrenheita, a następnie zwraca obie wartości.

Moduł DS18B20 występuje w postaci układu scalonego, ale możesz również znaleźć specjalną wersję zamkniętą w obudowie gwarantującej wstrząsoodporność i wodoodporność tego urządzenia.

Zobacz również

Jeżeli chcesz rejestrować dane generowane przez układ DS18B20, zajrzyj do receptury 12.12.

Niniejsza receptura została oparta na poradniku znajdującym się na stronie firmy Adafruit — <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing/>.

Pomiar temperatury za pomocą mniej dokładnego, analogowego czujnika TMP36 opisano w recepturze 12.7.

Dokumentację układu DS18B20 znajdziesz pod adresem: <http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>.

12.10. Pomiar odległości

Problem

Chcesz mierzyć odległość za pomocą ultradźwiękowego dalmierza.

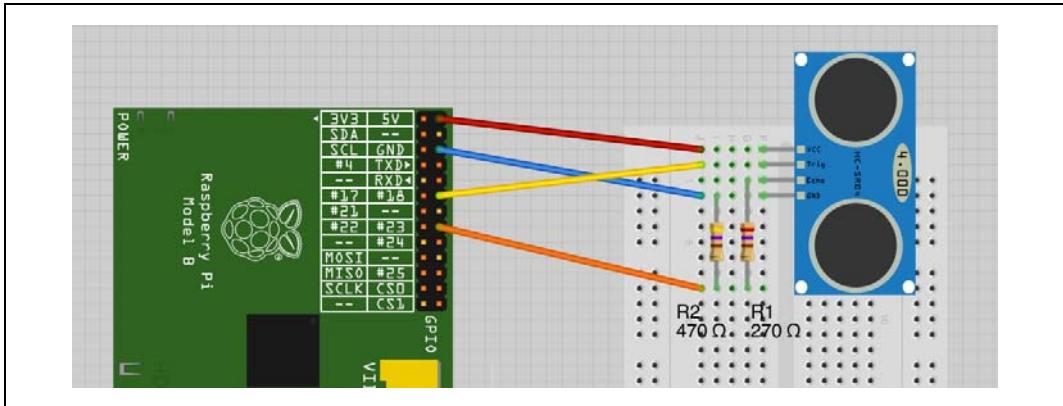
Rozwiążanie

Skorzystaj z taniego dalmierza SR-04. Dalmierz ten używa dwóch styków złącza GPIO — za pomocą jednego styku wywoływany jest impuls ultradźwiękowy, a za pomocą drugiego monitorowany jest czas powrotu odbitej fali dźwiękowej do urządzenia.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- dalmierza SR-04 (Allegro, eBay),
- rezystora $470\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- rezystora $270\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Połącz ze sobą elementy obwodu, korzystając z rysunku 12.17. Rezystory zastosowano w celu obniżenia napięcia sygnału wyjściowego dalmierza z 5 do 3,3 V (zobacz receptura 8.12).



Rysunek 12.17. Dalmierz SR-04 podłączony do Raspberry Pi

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ranger.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import RPi.GPIO as GPIO
import time

trigger_pin = 18
echo_pin = 23

GPIO.setmode(GPIO.BCM)
GPIO.setup(trigger_pin, GPIO.OUT)
GPIO.setup(echo_pin, GPIO.IN)

def send_trigger_pulse():
    GPIO.output(trigger_pin, True)
    time.sleep(0.0001)
    GPIO.output(trigger_pin, False)

def wait_for_echo(value, timeout):
    count = timeout
    while GPIO.input(echo_pin) != value and count > 0:
        count = count - 1

def get_distance():
    send_trigger_pulse()
    wait_for_echo(True, 10000)
    start = time.time()
    wait_for_echo(False, 10000)
    finish = time.time()
    pulse_len = finish - start
    distance_cm = pulse_len / 0.000058
    distance_in = distance_cm / 2.5
    return (distance_cm, distance_in)

while True:
    print("cm=%f\tafe=%f" % get_distance())
    time.sleep(1)
```

Działanie programu zostanie opisane w kolejnej sekcji tego podrozdziału. Uruchomiony program podaje raz na sekundę odległość wyrażoną w centymetrach oraz calach. Przed dalmierzem położ dłoń lub jakąś przeszkodę — wartości wyświetlane przez program ulegną zmianie.

```
sudo python ranger.py
cm=154.741879  cale=61.896752
cm=155.670889  cale=62.268356
cm=154.865199  cale=61.946080
cm=12.948595   cale=5.179438
cm=14.087249   cale=5.634900
cm=13.741954   cale=5.496781
cm=20.775302   cale=8.310121
cm=20.224473   cale=8.089789
```

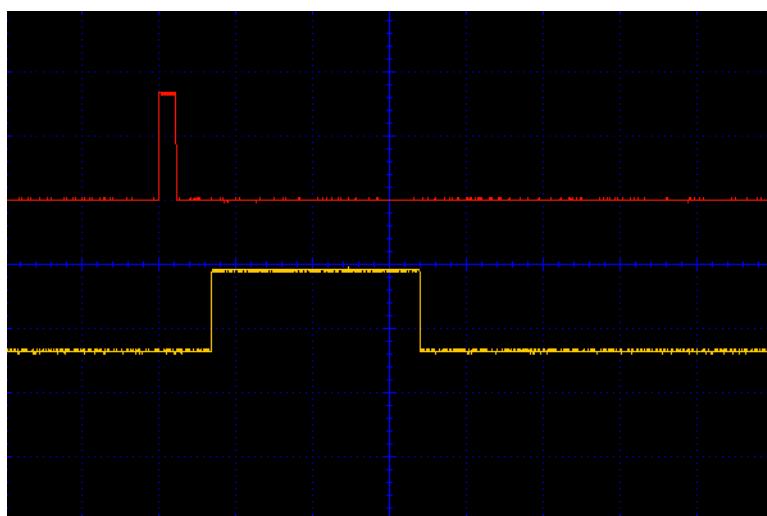
Omówienie

Istnieje wiele różnych dalmierzy ultradźwiękowych. Zastosowaliśmy model tani i prosty w obsłudze. Urządzenie to wysyła impuls ultradźwiękowy, a następnie mierzy czas potrzebny wysłanej fali na odbicie się od przeszkody i powrót. Na urządzeniu znajdują się dwa przetworniki ultradźwiękowe: nadajnik i odbiornik.

Proces ten jest sterowany przez Raspberry Pi. I właśnie tutaj kryje się różnica pomiędzy tańszymi a droższymi dalmierzami. Droższe modele są wyposażone we własny mikrokontroler, który samodzielnie wykonuje obliczenia i przekazuje dane do Raspberry Pi za pośrednictwem magistrali I2C lub interfejsu szeregowego.

Podłączając czujnik do Raspberry Pi, złącze wejściowe *trig* (wyzwalające) dalmierza należy połączyć ze złączem wyjściowym gniazda GPIO. Złącze wyjściowe (*echo*) dalmierza należy połączyć ze złączem wejściowym Raspberry Pi za pomocą rezystorów obniżających napięcie z 5 do 3,3 V.

Na rysunku 12.18 przedstawiono ślady sygnałów generowanych przez czujnik. Górnny (czarny) ślad ilustruje sygnał podawany na zacisk *trig*, a dolny (żółty) ślad ilustruje sygnał podawany na zacisk *echo*. Jak widzisz, najpierw krótki impuls jest podawany na złącze *trig*, a po krótkiej chwili na zacisku *echo* jest odbierany sygnał zwrotny. Czas trwania tego impulsu jest proporcjonalny do zmierzonej odległości.



Rysunek 12.18. Ślady sygnałów na ekranie oscyloskopu

Program obsługujący czujnik najpierw generuje impuls wyzwalający (za pomocą funkcji `send_trigger_pulse`). Program następnie oczekuje na podanie sygnału na wyjściu dalmierza i liczy czas trwania tego sygnału.

Mnożąc czas trwania impulsu przez prędkość dźwięku, możemy obliczyć odległość.

Zależnie od pierwszego argumentu funkcja `wait_for_echo` czeka na początek lub koniec sygnału podawanego na złączu `echo`. Drugi argument tej funkcji określa czas, po jakim funkcja przestanie działać, gdyby z jakiegoś powodu stan sygnału przekazywanego przez złącze `echo` nie ulegał zmianie. Zapobiega to zapętleniu się programu w nieskończoność.

Zobacz również

Zajrzyj do dokumentacji dalmierza ultradźwiękowego znajdującej się pod adresem: <http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf>.

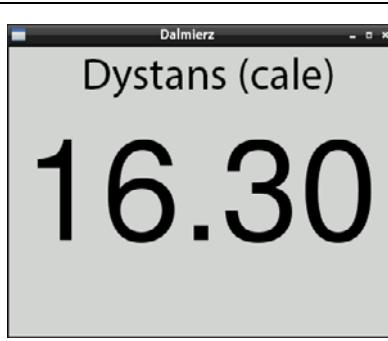
12.11. Wyświetlanie mierzonych wielkości

Problem

Podłączyleś czujnik do swojego Raspberry Pi i chcesz wyświetlić mierzone wartości w formie graficznej.

Rozwiążanie

Wyświetl dane dużą czcionką w oknie wygenerowanym przy użyciu biblioteki Tkinter (zobacz rysunek 12.19).



Rysunek 12.19. Wyświetlanie mierzonej wielkości za pomocą biblioteki Tkinter

W poniższym przykładzie będziemy wyświetlać odległość zmierzona za pomocą dalmierza. Aby wypróbować działanie programu, musisz najpierw wykonać instrukcje znajdujące się w recepturze 12.10.

Aby sprawdzić działanie dalmierza, umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą `gui_sensor_reading.py`. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

Pracą dalmierza powinien sterować kod przedstawiony wcześniej w recepturze 12.10. Poniższy listing zawiera tylko fragment kodu programu, który odpowiada za wyświetlanie na ekranie danych dotyczących odległości.

```
class App:  
    def __init__(self, master):  
        self.master = master  
        frame = Frame(master)  
        frame.pack()  
        label = Label(frame, text='Dystans (cale)', font=("Helvetica", 32))  
        label.grid(row=0)  
        self.reading_label = Label(frame, text='12.34', font=("Helvetica", 110))  
        self.reading_label.grid(row=1)  
        self.update_reading()  
  
    def update_reading(self):  
        cm, inch = get_distance()  
        reading_str = "{:.2f}".format(inch)  
        self.reading_label.configure(text=reading_str)  
        self.master.after(500, self.update_reading)  
  
root = Tk()  
root.wm_title('Dalmierz')  
app = App(root)  
root.geometry("400x300+0+0")  
root.mainloop()
```

Omówienie

Zastosowana technika wyświetlania danych może być również użyta w celu wyświetlenia wielkości mierzonych przez inne czujniki. Wystarczy zmienić wyświetlane etykiety, a także metodę pozyskiwania danych z czujnika.

Zobacz również

Informacje na temat formatowania liczb, tak aby były wyświetlane z określona liczbą cyfr za separatorem dziesiętnym, znajdziesz w recepturze 7.1.

12.12. Zapisywanie danych do dziennika utworzonego w pamięci USB

Problem

Chcesz zapisać wyniki pomiarów w pamięci USB.

Rozwiążanie

Napisz aplikację Pythona zapisującą dane do pliku znajdującego się w pamięci USB. Dane zapisane w formacie CVS (wartości rozdzielone przecinkami) można następnie importować do arkusza kalkulacyjnego, takiego jak np. Gnumeric (zobacz receptura 4.2).

Przykładowy program przedstawiony w dalszej części tego podrozdziału zapisuje dane generowane w wyniku pomiaru temperatury za pomocą układu DS18B20. Jeżeli chcesz go uruchomić, musisz wcześniej wykonać instrukcje zawarte w recepturze 12.9.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *temp_log.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import os, glob, time, datetime

log_period = 600 # sekundy

logging_folder = glob.glob('/media/*')[0]
dt = datetime.datetime.now()
file_name = "temp_log_{:%Y_%m_%d}.csv".format(dt)
logging_file = logging_folder + '/' + file_name

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
    return temp_c, temp_f

def log_temp():
    temp_c, temp_f = read_temp()
    dt = datetime.datetime.now()
    f = open(logging_file, 'a')
    f.write('\n"{:%H:%M:%S}"'.format(dt))
    f.write(str(temp_c))
    f.close()

print("Zapisywanie danych do: " + logging_file)
while True:
    log_temp()
    time.sleep(log_period)
```

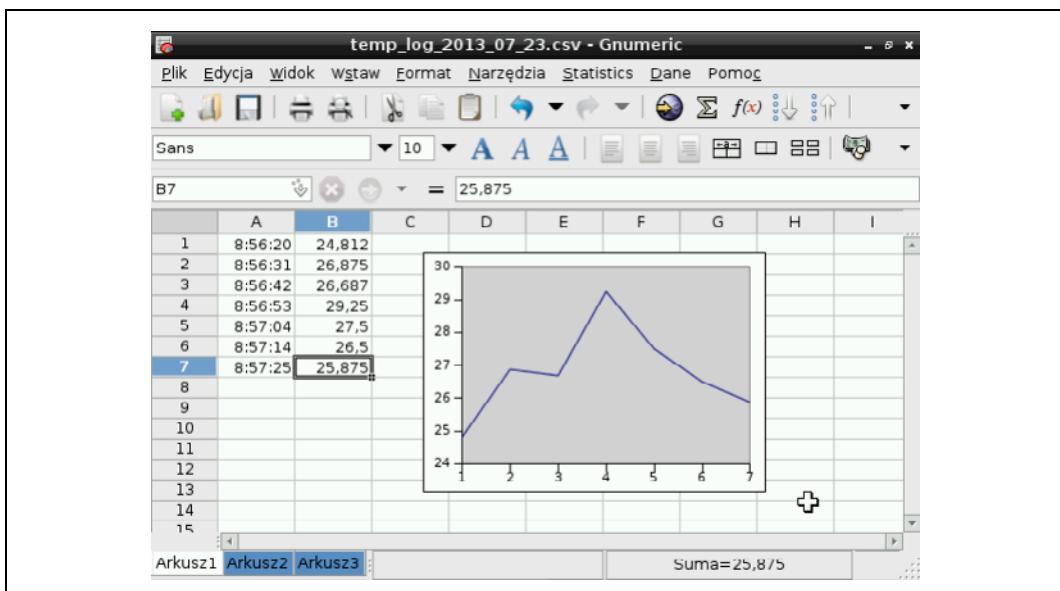
Program zapisuje temperaturę co 10 minut (600 sekund). Możesz to zmienić, modyfikując wartość przypisana zmiennej *log_period*.

Omówienie

Pamięć USB po podłączeniu do Raspberry Pi jest automatycznie instalowana w folderze `/media`. Jeżeli w systemie zainstalowano więcej niż jeden napęd, to program będzie zapisywał dane do pierwszego katalogu znajdującego się w folderze `/media`. Nazwa pliku, w którym zapisywane są dane, jest tworzona na podstawie bieżącej daty.

Jeżeli otworzysz wygenerowany plik w arkuszu kalkulacyjnym będącym składnikiem np. pakietu OpenOffice, będziesz mógł edytować bezpośrednio jego zawartość. Arkusz kalkulacyjny może Cię zapytać o separator danych. W tym przypadku będzie to przecinek (separatorem dziesiętnym zapisanych danych jest kropka).

Na rysunku 12.20 pokazano zestaw danych zapisanych za pomocą powyższego programu, otwarty w arkuszu kalkulacyjnym Gnumeric działającym na Raspberry Pi.



Rysunek 12.20. Wykres utworzony na podstawie danych

Zobacz również

Program ten można łatwo zmodyfikować, tak aby działał z innymi czujnikami, takimi jak czujnik światła (zobacz receptura 12.2) i czujnik przyspieszenia (zobacz receptura 12.8).

Wyświetlacz

13.0. Wprowadzenie

Raspberry Pi może być podłączone do telewizora lub monitora, jednak czasem warto podłączyć je do bardziej wyspecjalizowanego wyświetlacza. W tym rozdziale poznasz różne rodzaje wyświetlaczy oraz sposoby podłączania ich do Raspberry Pi.

Obwody przedstawione w większości receptur znajdujących się w tym rozdziale można wykonać za pomocą przewodów połączeniowych (zakończonych z jednej strony wtykiem męskim, a z drugiej żeńskim) oraz płytka prototypowej niewymagającej wykonywania połączeń lutowniczych (zobacz receptura 8.10).

13.1. Korzystanie z czterocyfrowego wyświetlacza LED

Problem

Chcesz wyświetlić czterocyfrową liczbę na siedmiosegmentowym wyświetlaczu LED.

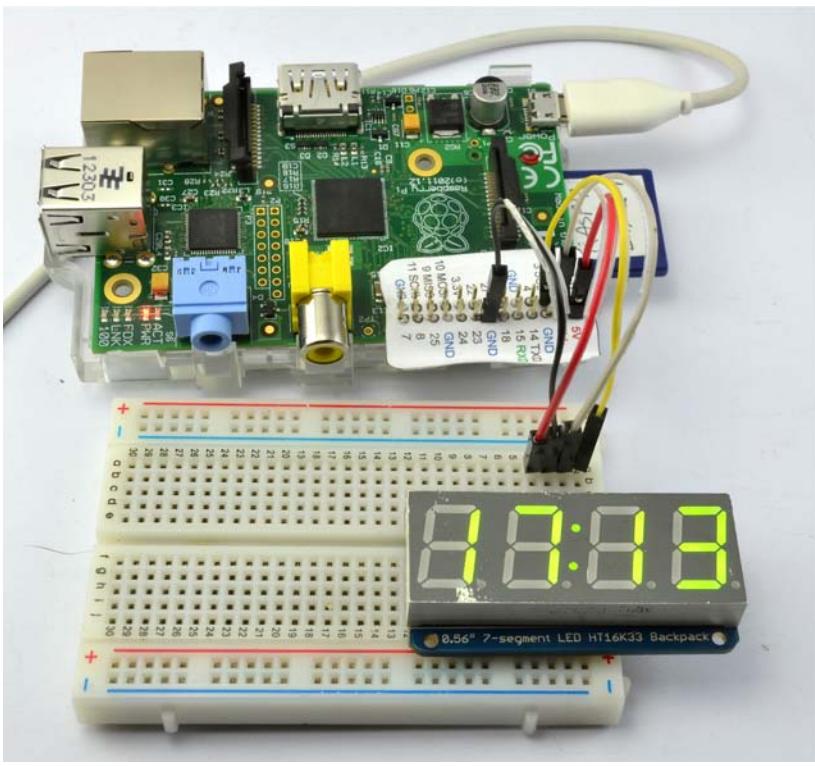
Rozwiązanie

Zastosuj moduł wyświetlacza obsługujący magistralę I2C. Na rysunku 13.1 przedstawiono moduł podłączony do Raspberry Pi za pośrednictwem płytki prototypowej.

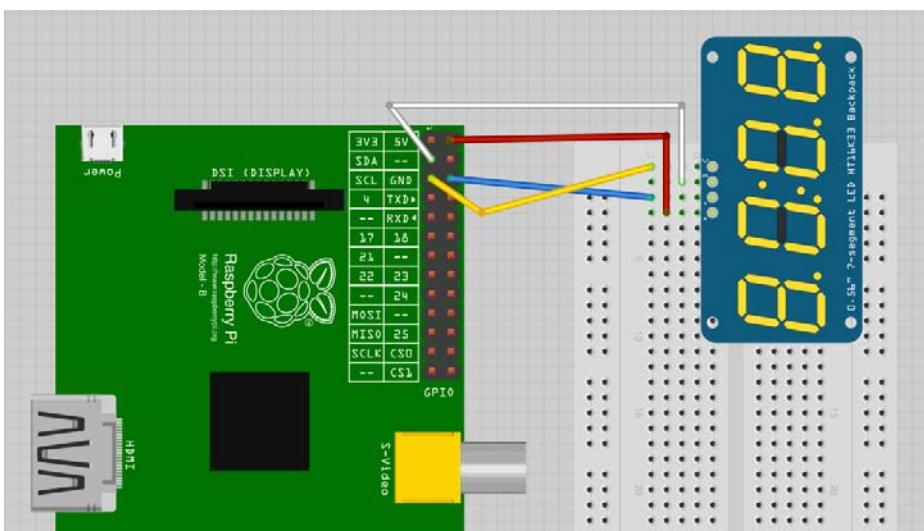
Aby skorzystać z tej receptury, potrzebujesz:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- modułu firmy Adafruit składającego się z czterech siedmiosegmentowych wyświetlaczy LED, wyposażonego w interfejs I2C (zobacz sekcja „Moduły” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej przedstawiono na rysunku 13.2.



Rysunek 13.1. Siedmiosegmentowy wyświetlacz LED podłączony do Raspberry Pi



Rysunek 13.2. Wyświetlacz LED podłączony do Raspberry Pi za pośrednictwem płytka prototypowej

Zanim skorzystasz z tej receptury, uruchom obsługę magistrali I2C. Jeżeli nie zrobiłeś tego wcześniej, to pracę zacznij od wykonania poleceń zawartych w recepturze 8.4.

Firma Adafruit stworzyła specjalną bibliotekę służącą do obsługi swojego wyświetlacza. Nie można jej jednak zainstalować w sposób właściwy dla *standardowych* bibliotek. Zanim zaczniesz z niej korzystać, musisz pobrać jej folder. Jeżeli jeszcze nie zainstalowałeś aplikacji Git (zobacz receptura 3.19), to zrób to za pomocą następującego polecenia:

```
$ sudo apt-get install git
```

Teraz możesz przystąpić do pobrania folderu z biblioteką z serwisu GitHub:

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

Zmień katalog, w którym pracujesz, na folder zawierający bibliotekę:

```
$ cd Adafruit-Raspberry-Pi-Python-Code  
$ cd Adafruit_LEDBackpack
```

W folderze tym znajdziesz program testujący, który wyświetla aktualny czas. Uruchom go za pomocą polecenia:

```
$ sudo python ex_7segment_clock.py
```

Omówienie

Gdy otworzysz plik *ex_7segment_clock.py* w edytorze nano, zobaczyś, że głównym poleceniem programu jest:

```
from Adafruit_7Segment import SevenSegment
```

Importuje ono do programu kod biblioteki. Kolejna linia kodu tworzy egzemplarz *SevenSegment*. W roli argumentu dostarczany jest adres magistrali I2C (zobacz receptura 8.4).

Adres nadawany jest każdemu urządzeniu podłączonemu do magistrali. Płytką modułu ma na spodzie trzy pary punktów, które po połączeniu ze sobą spowodują zmianę adresu. Możliwość taka przyda Ci się w przypadku podłączania kilku takich samych wyświetlaczy do jednego Raspberry Pi.

```
segment = SevenSegment(address=0x70)
```

Żeby wyświetlić coś za pomocą omawianego modułu, program musi uruchomić następującą linię kodu:

```
segment.writeDigit(0, int(hour / 10))
```

Pierwszy argument (0) jest pozycją cyfry. Możemy korzystać z pozycji o numerach 0, 1, 3 i 4. Pozycja o numerze 2 jest zarezerwowana dla dwóch kropek znajdujących się w środkowej części wyświetlacza.

Cyfrę, którą chcemy wyświetlić, podajemy jako drugi argument.

Zobacz również

Więcej informacji na temat biblioteki Adafruit znajdziesz pod adresem <https://learn.adafruit.com/matrix-7-segment-led-backpack-with-the-raspberry-pi/using-the-adafruit-library>.

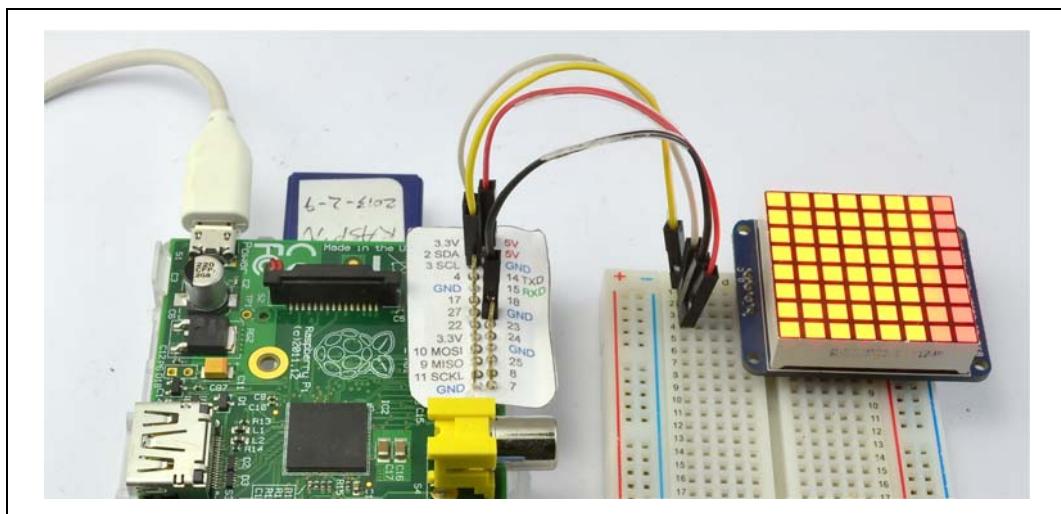
13.2. Wyświetlanie komunikatów za pomocą wyposażonego w interfejs I2C wyświetlacza składającego się z matrycy diod LED

Problem

Chcesz sterować pikselami, z jakich składa się kolorowy wyświetlacz będący matrycą diod LED.

Rozwiązanie

Podłącz do Raspberry Pi za pośrednictwem płytka prototypowej moduł wyświetlacza matrycowego (zobacz rysunek 13.3). Układ widoczny na tym zdjęciu może wyglądać trochę inaczej, niż sugerowałby to schemat umieszczony w dalszej części rozdziału, ale wykonane połączenia są identyczne.

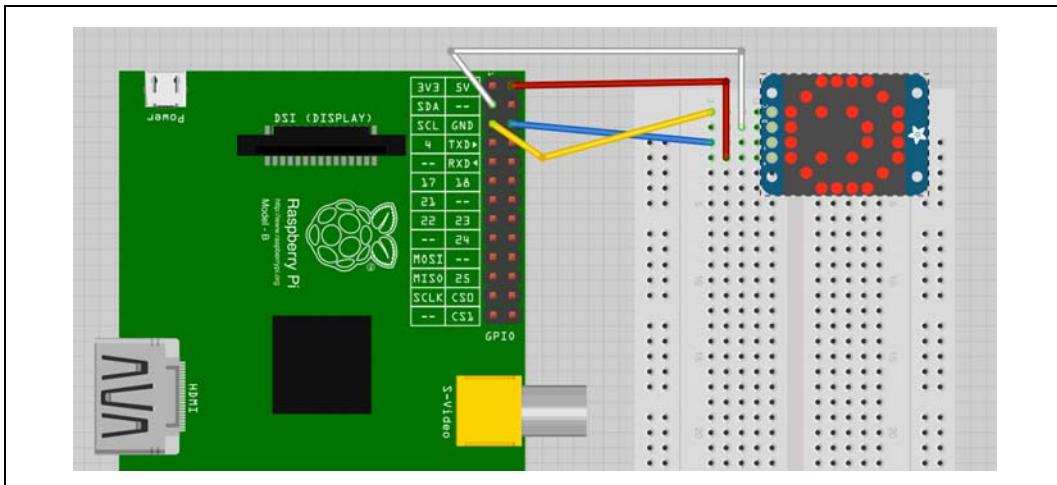


Rysunek 13.3. Matryca diod LED podłączona do Raspberry Pi

Aby skorzystać z tej receptury, potrzebujesz:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- kolorowej matrycy diod firmy Adafruit wyposażonej w interfejs I2C (zobacz sekcja „Moduły” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej przedstawiono na rysunku 13.4. Tak naprawdę, jeżeli nie planujesz dodawać do tego układu przycisków ani innych komponentów, to możesz podłączyć wyświetlacz bezpośrednio do Raspberry Pi za pomocą przewodów połączeniowych zakończonych obustronnie wtykami żeńskimi.



Rysunek 13.4. Matryca diod LED podłączona do Raspberry Pi za pośrednictwem płytki prototypowej

Zanim skorzystasz z tej receptury, uruchom obsługę magistrali I2C. Jeżeli nie zrobileś tego wcześniej, to pracę zacznij od wykonania poleceń zawartych w recepturze 8.4.

Firma Adafruit stworzyła specjalną bibliotekę służącą do obsługi swojego wyświetlacza. Nie można jej jednak zainstalować w sposób właściwy dla *standardowych* bibliotek. Zanim zaczniesz z niej korzystać, musisz pobrać jej folder. Jeżeli jeszcze nie zainstalowałeś aplikacji Git (zobacz receptura 3.19), to zrób to za pomocą następującego polecenia:

```
$ sudo apt-get install git
```

Teraz możesz przystąpić do pobrania folderu z biblioteką z serwisu GitHub:

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```

Zmień katalog, w którym pracujesz, na folder zawierający bibliotekę:

```
$ cd Adafruit-Raspberry-Pi-Python-Code  
$ cd Adafruit_LEDBackpack
```

W folderze tym znajduje się program testujący działanie wyświetlacza — wyświetla aktualny czas za pomocą cyfr przesuwających się po matrycy. Uruchom go za pomocą polecenia:

```
$ sudo python ex_8x8_color_pixels.py
```

Omówienie

Program kolejno wyświetla wszystkie kolory pikseli. Kod programu po usunięciu nieinteresujących nas elementów ma następującą postać:

```
import time  
from Adafruit_8x8 import ColorEightByEight  
  
grid = ColorEightByEight(address=0x70)  
  
iter = 0  
  
# Stale odświeża kolejne piksele wyświetlacza 8×8  
while(True):
```

```
iter += 1

for x in range(0, 8):
    for y in range(0, 8):
        grid.setPixel(x, y, iter % 4)
        time.sleep(0.02)
```

Po zimportowaniu biblioteki tworzony jest egzemplarz `ColorEightByEight`.

Wartość podana jako argument w poniższej linii kodu jest adresem magistrali I2C (zobacz receptura 8.4).

```
grid = ColorEightByEight(address=0x70)
```

Adres nadawany jest każdemu urządzeniu podłączonemu do magistrali. Płytką modułu ma na spodzie trzy pary punktów, które po połączeniu ze sobą spowodują zmianę adresu. Taka możliwość przyda Ci się w przypadku podłączania kilku takich samych wyświetlaczy do jednego Raspberry Pi.

Po każdym wykonaniu poleceń zawartych w pętli do wartość przechowywana przez zmieniąną `iter` jest zwiększana o 1. Dwa pierwsze parametry funkcji `grid.setPixel` to współrzędne piksela (względem osi `x` i `y`). Trzeci parametr określa kolor piksela. Liczba 0 oznacza niepodświetlenie piksela, liczba 1 spowoduje podświetlenie piksela na kolor zielony, 2 na kolor czerwony, a 3 na kolor pomarańczowy.

Zmienną `iter` stosuje się do generowania liczb z zakresu od 0 do 3 za pomocą operatora dzielenia modulo (%). Zwraca on resztę z dzielenia wartości zmiennej `iter` przez 4.

Zobacz również

Więcej informacji na temat zaprezentowanej matrycy diod znajdziesz pod adresem <http://www.adafruit.com/products/902>.

13.3. Korzystanie z płytki Pi-Lite

Problem

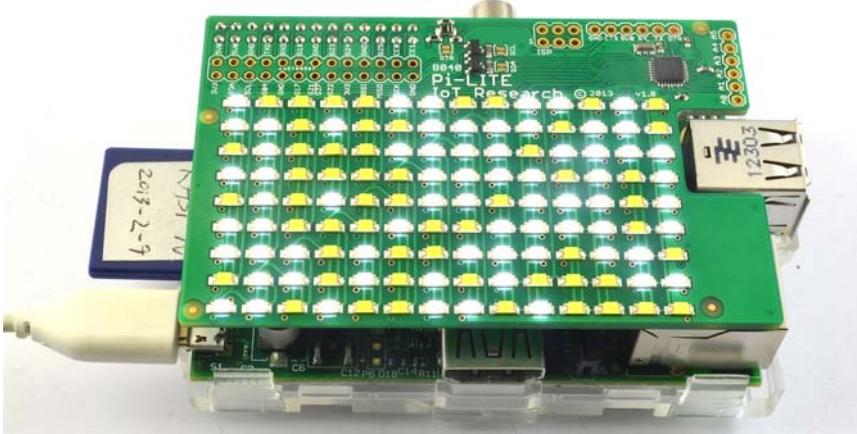
Chcesz wyświetlać komunikaty tekstowe za pomocą matrycy 9×14 diod LED.

Rozwiążanie

Podłącz płytke Pi-Lite do portu GPIO swojego Raspberry Pi za pośrednictwem portu szeregowego, a następnie napisz aplikację Pythona wyświetlającą na niej komunikaty.

Płytkę Pi-Lite komunikuje się z Raspberry Pi za pośrednictwem portu szeregowego. Pracę z płytka zacznij od wyłączenia konsoli szeregowej — wykonaj polecenia znajdujące się w recepturze 8.7, a następnie zainstaluj bibliotekę `PySerial`, korzystając z receptury 8.8.

Pi-Lite jest płytą dającą wiele możliwości. Jest tak duża, że przesyłania niemal całą powierzchnię Raspberry Pi (zobacz rysunek 13.5).



Rysunek 13.5. Płytkę Pi-Lite założona na Raspberry Pi

Płytkę Pi-Lite (zobacz sekcję „Moduły” w dodatku A) posiada własny procesor sterujący matrycą diod LED. Polecenia do tego modułu można wysyłać za pomocą połączenia szeregowego. Domyslnie urządzenie wyświetla przesyłany tekst w postaci przesuwających się w poziomie napisów.

Wyłącz Raspberry Pi i podłącz do niego płytę Pi-Lite. Po podłączeniu Raspberry Pi do zasilania płytki wykona procedurę testową polegającą na zapalaniu diod.

Działanie płytki można sprawdzić za pomocą programu Minicom (zobacz receptura 8.9). Możesz go zainstalować przy użyciu następującego polecenia:

```
$ sudo apt-get install minicom
```

Otwórz sesję programu Minicom, uruchamiając polecenie:

```
minicom -b 9600 -o -D /dev/ttyAMA0
```

Tekst wpisywany w programie powinien zostać wyświetlony za pomocą matrycy składającej się z diod LED.

Omówienie

Wyświetlanie komunikatów za pomocą aplikacji Pythona jest również proste. Operacja taka wymaga zainstalowania biblioteki odpowiedzialnej za obsługę portu szeregowego (zobacz receptura 8.8). Zainstaluj ją, uruchamiając polecenie:

```
$ sudo apt-get install python-serial
```

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *pi_lite_message.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import serial  
  
ser = serial.Serial('/dev/ttyAMA0', 9600)  
  
while True:  
    message = raw_input("Wpisz tekst: ")  
    ser.write(message)
```

Program poprosi Cię o wpisanie tekstu, który zostanie następnie przesłany do płytki Pi-Lite:

```
$ sudo python pi_lite_message.py  
Wpisz tekst: Witaj  
Wpisz tekst:
```

Poza wyświetlaniem tekstu możesz również sterować pracą pojedynczych pikseli znajdujących się na płytce Pi-Lite albo tworzyć wykresy słupkowe (adres, pod jakim znajdziesz oficjalny poradnik użytkownika, podano w sekcji „Zobacz również” na końcu tego podrozdziału).

Poniższy kod programu (*pi_lite_rain.py*) włącza i wyłącza pojedynczo diody płytka Pi-Lite:

```
import serial  
import random  
  
ser = serial.Serial('/dev/ttyAMA0', 9600)  
  
while True:  
    col = random.randint(1, 14)  
    row = random.randint(1, 9)  
    ser.write("$$P%d,%d,TOGGLE\r" % (col, row))
```

Zobacz również

Oficjalny poradnik poczatkujacego uzytkownika płytki Pi-Lite znajdziesz pod adresem <http://www.openmicr0s.org/index.php/articles/94-ciseco-product-documentation/raspberry-pi/280-b040-pi-lite-beginners-guide>.

Na płytce Pi-Lite można wyświetlić Grę o życie stworzoną przez Johna Conwaya — <https://www.youtube.com/watch?v=bVavjoeHuak>.

13.4. Wyświetlanie komunikatów na alfanumerycznym wyświetlaczu LCD

Problem

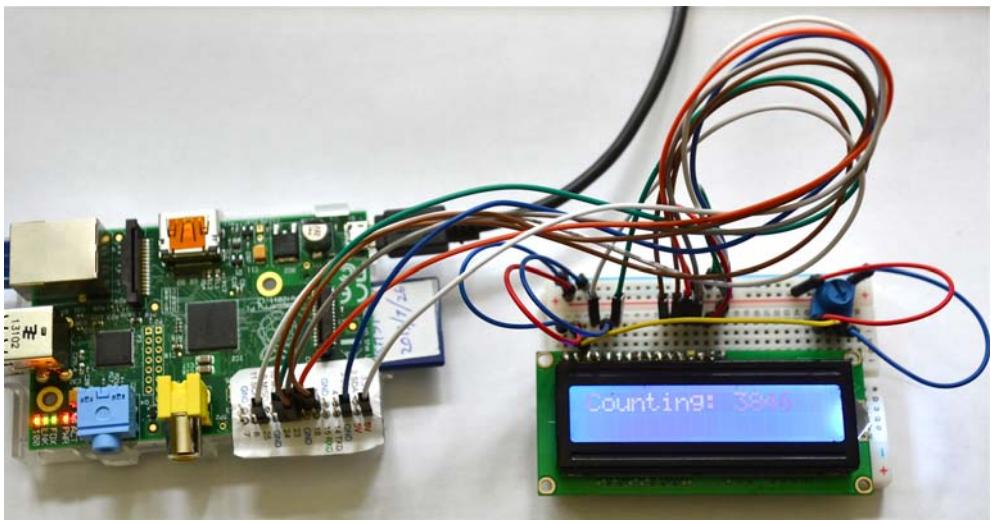
Chcesz wyświetlać komunikaty tekstowe za pomocą alfanumerycznego wyświetlacza LCD.

Rozwiążanie

Zastosuj moduł ekranu ciekłokrystalicznego kompatybilnego z układem HD44780. Podłącz go do złącza GPIO.

Znajdziesz wiele tanich modułów wyświetlaczy ciekłokrystalicznych. Przykładowy moduł takiego ekranu pokazano na rysunku 13.6. Wypożyczyłeś go w 16 styków. Na szczęście nie musisz podłączać ich wszystkich do gniazda GPIO.

Moduły tego typu mają różne rozmiary. Różnią się one liczbą kolumn i wierszy, w których mogą być wyświetlane znaki. W tej recepturze korzystamy z ekranu 16×2 — może on wyświetlać tekst w dwóch wierszach, a w każdym z nich mieści się 16 znaków. Inne popularne rozmiary ekranów to 8×1, 16×1, 16×2, 20×2 i 20×4.



Rysunek 13.6. Wyświetlacz ciekłokrystaliczny (16×2) podłączony do Raspberry Pi

Aby skorzystać z tej receptury, potrzebujesz:

- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- modułu ekranu LCD 16×2 kompatybilnego z HD44780 (zobacz sekcja „Moduły” w dodatku A),
- listwy składającej się z 16 goldpinów (zobacz sekcja „Pozostałe” w dodatku A),
- potencjometru dostrojczego $10\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej przedstawiono na rysunku 13.7. Moduły wyświetlaczy nie są zwykle wyposażone w goldpiny. Będziesz musiał je samodzielnie przymontować.

Potencjometr dostrojczy reguluje kontrast wyświetlacza. Jeżeli na ekranie nie widać żadnych znaków, to poruszaj pokrętłem potencjometru. Sprawdź funkcjonowanie ekranu w całym zakresie działania potencjometru. Zbyt niski kontrast może spowodować, że będziesz miał wrażenie, iż na wyświetlaczu nie są wyświetlane żadne treści.

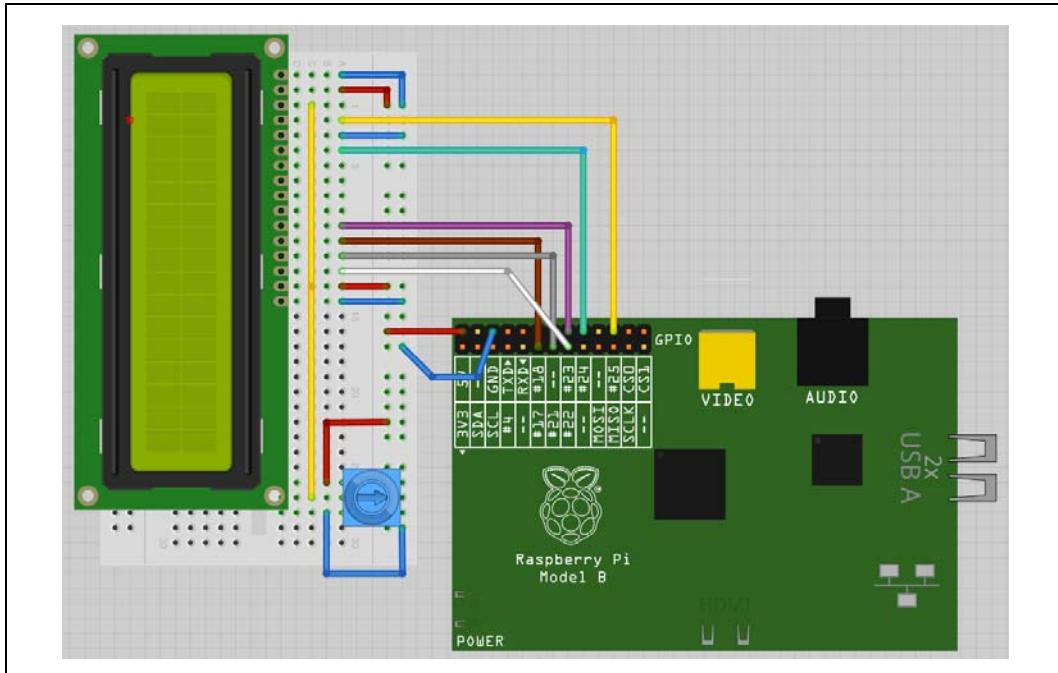
Przykładowe kody stworzone przez firmę Adafruit możesz pobrać z serwisu GitHub. Do kodów dołączona jest biblioteka sterująca wyświetlaczami ciekłokrystalicznymi opartymi na sterowniku HD44780. Zanim przystąpisz do jej instalacji, zainstaluj bibliotekę RPi.GPIO (wykonaj polecenia znajdujące się w recepturze 8.3).

Biblioteki firmy Adafruit nie można zainstalować w sposób właściwy dla *zwyczajnych* bibliotek. Zanim zaczniesz z niej korzystać, musisz pobrać jej folder. Jeżeli jeszcze nie zainstalowałeś aplikacji Git (zobacz receptura 3.19), to zrób to za pomocą następującego polecenia:

```
$ sudo apt-get install git
```

Teraz możesz przystąpić do pobrania folderu z biblioteką z serwisu GitHub:

```
$ git clone https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
```



Rysunek 13.7. Wyświetlacz ciekłokrystaliczny podłączony do Raspberry Pi

Zmień katalog, w którym pracujesz, na folder zawierający bibliotekę:

```
$ cd Adafruit-Raspberry-Pi-Python-Code
$ cd Adafruit_LEDBackpack
```

W folderze tym znajduje się program testujący działanie wyświetlacza — wyświetla on aktualny czas systemowy oraz adres IP przypisany do Raspberry Pi. Jeżeli korzystasz z nowszej płytki Raspberry Pi (model B rev2), to musisz zmodyfikować plik *Adafruit_CharLCD.py*:

```
$ nano Adafruit_CharLCD.py
```

W otwartym pliku znajdź wiersz:

```
ef __init__(self, pin_rs=25, pin_e=24, pins_db=[23, 17, 21, 22], GPIO = None):
```

Zamień liczbę 21 na 27, tak aby linia ta wyglądała następująco:

```
def __init__(self, pin_rs=25, pin_e=24, pins_db=[23, 17, 27, 22], GPIO = None):
```

Zapisz zmiany w pliku i zamknij edytor.

Program możesz uruchomić za pomocą polecenia:

```
$ sudo python Adafruit_CharLCD_IPclock_example.py
```

Omówienie

Wyświetlacze tego typu mogą współpracować z cztero- oraz ośmiobitową szyną danych. Dodatkowo mają trzy złącza sterujące. W tabeli 13.1 przedstawiono połączenia pomiędzy modułem wyświetlacza a Raspberry Pi.

Tabela 13.1. Połączenia pomiędzy wyświetlaczem a Raspberry Pi

Pin modułu wyświetlacza	Pin gniazda GPIO	Uwagi
1	GND	0 V
2	+5V	zasilanie układu logicznego prądem o napięciu 5 V
3	brak połączenia	napięcie sterujące kontrastem
4	25	<i>RS</i> — wybór rejestru
5	GND	<i>RW</i> — wybór pomiędzy zapisem a odczytem (w praktyce tylko tryb zapisu)
6	24	<i>EN</i> — zezwól
7–10	brak połączenia	stosowane tylko w trybie ośmiobitowym
11	23	<i>D4</i> — linia danych 4
12	17	<i>D5</i> — linia danych 5
13	21	<i>D6</i> — linia danych 6
14	22	<i>D7</i> — linia danych 7
15	+5V	podświetlenie diodowe
16	GND	podświetlenie diodowe

Plik *Adafruit_CharLCD.py* (będący składnikiem biblioteki) odpowiada za ustalanie sygnałów podawanych na pinach przesyłających dane. W pliku tym znajdują się definicje następujących funkcji, które możesz zastosować we własnych programach:

- *home()* — ustawia kursor w lewym górnym brzegu ekranu,
- *clear()* — czyści ekran,
- *setCursor(wiersz, kolumna)* — ustawia kursor w określonym punkcie ekranu,
- *cursor()* — wyświetla kursor,
- *noCursor()* — wyłącza wyświetlanie kursora (ustawienie domyślne),
- *message(komunikat)* — wyświetla komunikat w miejscu ustawienia kursora.

Poniższy minimalistyczny przykładowy program ilustruje, jak łatwo za pomocą wspomnianej biblioteki wyświetlać proste komunikaty:

```
from Adafruit_CharLCD import Adafruit_CharLCD
from time import sleep

lcd = Adafruit_CharLCD()
lcd.begin(16,2)

i = 0

while True:
    lcd.clear()
    lcd.message('Liczenie: ' + str(i))
    sleep(1)
    i = i + 1
```

Zobacz również

Niniejsza receptura została napisana w oparciu o poradnik umieszczony w witrynie firmy Adafruit pod adresem <https://learn.adafruit.com/drive-a-16x2-lcd-directly-with-a-raspberry-pi/overview>.

Firma Adafruit sprzedaje również moduły wyświetlaczy nakładane bezpośrednio na płytę Raspberry Pi. Są one kompatybilne z rozwiązaniami przedstawionymi w tej recepturze.

Raspberry Pi i Arduino

14.0. Wprowadzenie

Płytki Arduino (zobacz rysunek 14.1) jest popularną alternatywą dla płyt interfejsowych, takich jak PiFace (zobacz receptura 8.16).



Rysunek 14.1. Płytki Arduino Uno

Płytki Arduino przypominają na pierwszy rzut oka Raspberry Pi — są one małe i w gruncie rzeczy pełnią funkcję komputera. Istnieje jednak pomiędzy nimi wiele różnic. Płytki Arduino:

- nie mają żadnego interfejsu pozwalającego na podłączenie do nich klawiatury, myszy i ekranu,
- posiadają zaledwie 2 kB pamięci RAM i 32 kB pamięci flash służącej do przechowywania programów,
- ich procesory są taktowane zegarem o częstotliwości 16 MHz (procesor Raspberry Pi pracuje z częstotliwością 700 MHz).

Płytki Arduino charakteryzuje się gorszymi parametrami technicznymi. Dlaczego więc korzystamy z jej pośrednictwa, zamiast podłączać rozmaite moduły bezpośrednio do Raspberry Pi?

Płytki Arduino (najpopularniejszą z nich jest płytka Arduino Uno) sprawdzają się w komunikacji z urządzeniami elektronicznymi lepiej od Raspberry Pi. Płytki Arduino są wyposażone w:

- 14 cyfrowych złącz mogących pracować w charakterze wejścia i wyjścia; złącza te działają podobnie do styków gniazda GPIO, ale każde z nich może dostarczyć prąd o natężeniu 40 mA (w przypadku Raspberry Pi było to zaledwie 3 mA) — pozwala to na zasilanie pewnych urządzeń bez konieczności stosowania zewnętrznych układów zasilających;
- 6 wejść analogowych — podłączanie analogowych czujników jest dużo prostsze (zobacz rozdział 12.);
- 6 wyjść wyposażonych w układy PWM — złącza te posiadają sprzętowe zegary i generują o wiele dokładniejszy sygnał od złącz znających się na płytce Raspberry Pi, pozwalając na dokładniejsze sterowanie pracą servomotorów.

Ponadto istnieje wiele płyt (tzw. *shieldów*), które możesz podłączyć do Arduino. Płytki te mogą pełnić różne funkcje — sterownika silników, wyświetlacza ciekłokrystalicznego itd.

Zastosowanie Arduino do wykonywania poleceń generowanych przez Raspberry Pi jest bardzo często wygodnym rozwiązaniem — korzystamy w ten sposób z zalet obu płyt. Najwięcej korzyści odniesiemy, używając płytek interfejsu, które są wyposażone w złącze GPIO, ale jednocześnie współpracują z Arduino. Przykładem takiej płytka jest aLaMode (zobacz receptura 14.13).

14.1. Programowanie Arduino za pośrednictwem Raspberry Pi

Problem

Chcesz uruchomić zintegrowane środowisko programistyczne Arduino na swoim Raspberry Pi, a następnie pisać w nim programy i ładować je do pamięci Arduino.

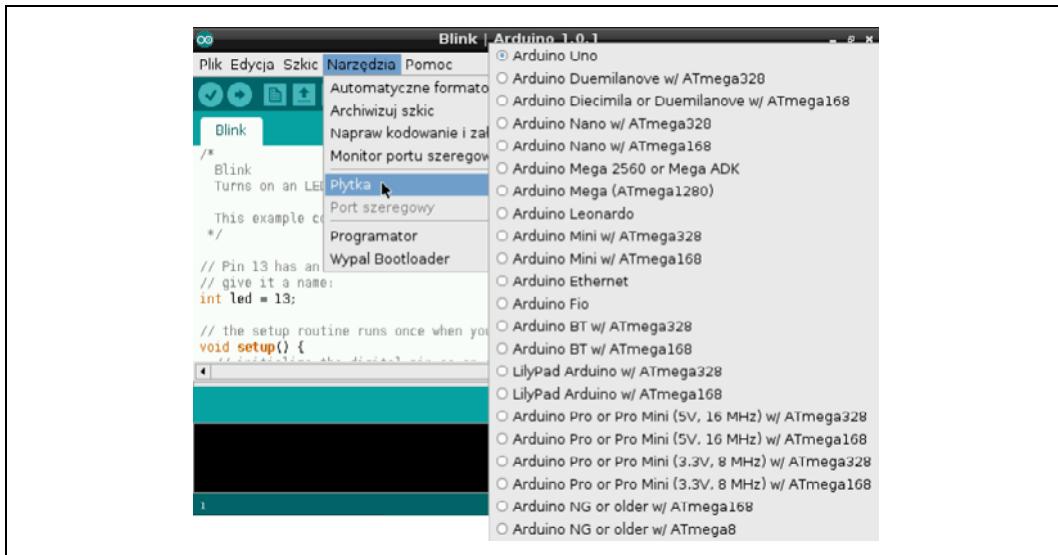
Rozwiązanie

Möesz zainstalować środowisko programistyczne Arduino na swoim Raspberry Pi. Działa ono dość wolno, ale jest przydatnym narzędziem. Aby je zainstalować, uruchom poniższe polecenia:

```
$ sudo apt-get update  
$ sudo apt-get install arduino
```

W chwili wydania tej książki polecenia te spowodują zainstalowanie aplikacji w wersji 1.0.1. Nie jest to jej najnowsza wersja, ale sprawdzi się w przypadku płytki Arduino Uno. Niestety nie obsługuje ona nowszych płyt, takich jak Leonardo czy Due. Möesz podłączać te płytka do Raspberry Pi, ale musisz je wcześniej zaprogramować na innym komputerze.

Po zainstalowaniu środowiska Arduino w Twoim menu z programami zostanie utworzona nowa sekcja o nazwie *Elektronika*. Znajdziesz w niej zainstalowany program (zobacz rysunek 14.2).



Rysunek 14.2. Zintegrowane środowisko programistyczne Arduino uruchomione na Raspberry Pi

Płytkę Arduino programuje się za pośrednictwem gniazda USB znajdującego się na Raspberry Pi. Aby nawiązać połączenie z Arduino, musisz wyłączyć konsolę szeregową. W tym celu możesz wykonać polecenia znajdujące się w recepturze 8.7. Warto jednak, żebyś uruchomił skrypt stworzony przez Kevina Osborna. Wyłącza on konsolę szeregową i jednocześnie konfiguruje porty szeregowe, a także niezbędne profile Arduino. Takie rozwiążanie równocześnie przygotowuje nasze Raspberry Pi do pracy z płytą aLaMode (zobacz receptura 14.13).

Aby pobrać i uruchomić wspomniany skrypt, skorzystaj z następujących poleceń:

```
$ wget https://github.com/wyolum/alamode/blob/master/bundles  
/alamode-setup.tar.gz?raw=true -O alamode-setup.tar.gz  
$ tar -xvzf alamode-setup.tar.gz  
$ cd alamode-setup  
$ sudo ./setup
```

Jeżeli nie wyłączyłeś wcześniej konsoli szeregowej (chciałeś, żeby wyłączył ją wspomniany skrypt), to przed przystąpieniem do dalszej pracy musisz ponownie uruchomić Raspberry Pi.

```
$ sudo reboot
```

Teraz możesz już podłączyć Arduino do swojego Raspberry Pi. W środowisku programistycznym w menu *Narzędzia* wybierz podmenu *Płytki*, w którym znajdziesz opcję *Arduino Uno*. Następnie w menu *Port szeregowy* wybierz opcję */dev/ttyACM0*. Żeby załadować pierwszy program na płytę Arduino (program sprawi, że dioda znajdująca się na płytce zacznie migać), z menu *Plik* przejdź do menu *Przykłady*, a następnie *Basics* i kliknij pozycję *Blink*. Kliknij ikonę strzałki znajdująca się w pasku narzędzi — rozpoczęcie się proces komplikacji programu, który zostanie następnie załadowany do pamięci Arduino. Jeżeli wszystko poszło dobrze, powinieneś zobaczyć w dolnej części okna programu komunikat *Ładowanie zakończone pomyślnie*.



Jeżeli na liście urządzeń podłączonych do portu szeregowego nie znajdziesz *ttyACM0*, a Twoje Arduino jest podłączone do Raspberry Pi, to uruchom ponownie środowisko programistyczne. Jeżeli to nie pomoże, uruchom ponownie swoje Raspberry Pi. Nie odłączaj płytki Arduino podczas ponownego uruchamiania aplikacji lub systemu operacyjnego.

Omówienie

Żeby wykorzystać pełnię możliwości oferowanych przez Arduino podłączone do Raspberry Pi, musisz poznać podstawy programowania Arduino. Lektura książki mojego autorstwa *Arduino dla początkujących. Podstawy i szkice* (Helion, Gliwice 2014) z pewnością Ci w tym pomoże.

Możesz korzystać z płytki Arduino bez konieczności pisania szkiców specjalnie dla niej przeznaczonych — wystarczy zastosować aplikację PyFirmata. Informacje dotyczące tego programu znajdziesz w recepturze 14.3.

Zobacz również

Skrypt konfigurujący środowisko programistyczne Arduino pochodzi z bloga Balda Wisdoma: <http://baldwisdom.com/simplified-setup-for-arduino-on-raspberry-pi/>.

14.2. Komunikacja z Arduino za pośrednictwem monitora portu szeregowego

Problem

Chcesz wyświetlać komunikaty wysyłane przez Arduino.

Rozwiążanie

Środowisko programistyczne Arduino zawiera narzędzie — **monitor portu szeregowego**, które pozwala na wysyłanie poleceń do Arduino (za pośrednictwem interfejsu USB) oraz odbieranie komunikatów generowanych przez tę płytke.

Aby wypróbować działanie tego narzędzia, najpierw musisz napisać program. W społeczności Arduino programy nazywane są **szkicami**. Nasz pierwszy szkic będzie po prostu wysyłał co sekundę komunikat, który będzie wyświetlany w oknie monitora portu szeregowego.

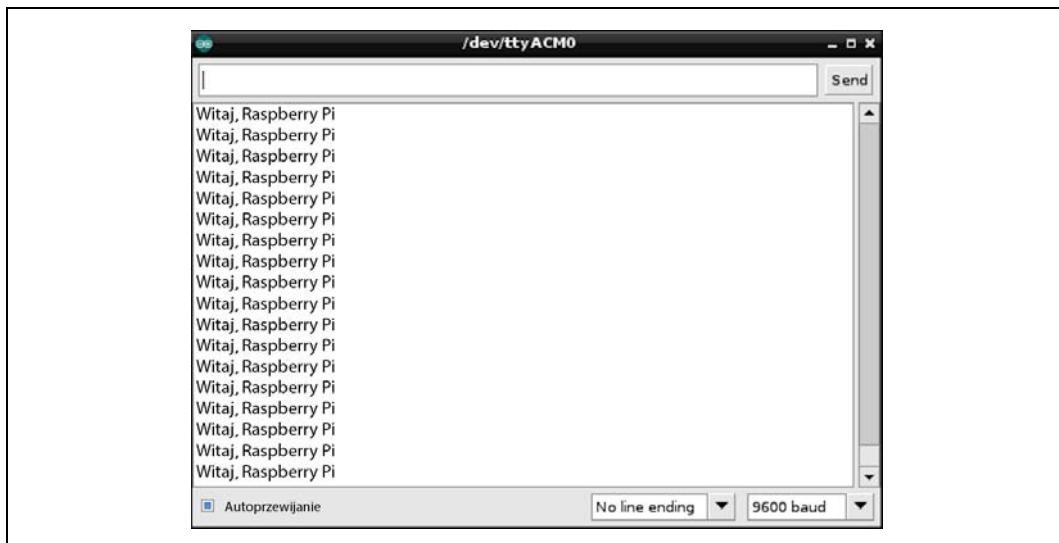
Poniżej znajduje się kod szkicu. Możesz go pobrać, tak jak kody innych programów umieszczone w tej książce, ze strony <http://www.helion.pl/ksiazki/raspre.htm>. Plik z kodem znajdziesz w folderze o nazwie *ArduinoHello*.

W menu *Plik* wybierz polecenie *Nowy* — w ten sposób utworzysz nowy szkic. Następnie wklej w otwarte okno poniższy kod i załaduj go do pamięci Arduino.

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
    Serial.println("Witaj, Raspberry Pi");
    delay(1000);
}
```

Szkic po załadowaniu do pamięci Arduino zostanie automatycznie uruchomiony. Program będzie wysyłał komunikat o treści *Witaj, Raspberry Pi*. Aby go odczytać, otwórz monitor portu szeregowego (zobacz rysunek 14.3). W tym celu kliknij ikonę lupy znajdująą się w pasku narzędzi.



Rysunek 14.3. Komunikaty wyświetlane w oknie monitora portu szeregowego

W prawym dolnym rogu okna znajduje się menu zawierające listę prędkości transmisji danych. Jeżeli domyślnie wybrana jest wartość inna niż 9600, to zmień ją.

Omówienie

W recepturach 14.10 i 14.11 zajmiemy się tworzeniem kodu własnego programu pozwalającego na komunikację z Arduino bez potrzeby uruchamiania środowiska programistycznego.

Sposobem na uniknięcie tworzenia kodu programu jest skorzystanie z aplikacji PyFirmata. Więcej informacji na ten temat znajdziesz w recepturze 14.3.

Zobacz również

Arduino jest dość proste do opanowania. Poniżej znajduje się lista książek oraz zasobów sieciowych, od których możesz zacząć swoją przygodę z Arduino.

- Simon Monk, *Arduino dla początkujących. Podstawy i szkice*, Helion, Gliwice 2014.
- Oficjalny poradnik dla osób rozpoczynających pracę z Arduino: <http://arduino.cc/en/Guide/HomePage>.
- Seria lekcji dotyczących Arduino znajdująca się na stronie internetowej firmy Adafruit: <https://learn.adafruit.com/category/learn-arduino>.
- Michael Margolis, *Arduino Cookbook*, O'Reilly, 2011 (<http://shop.oreilly.com/product/9780596802486.do>).

14.3. Sterowanie Arduino za pomocą biblioteki PyFirmata zainstalowanej na Raspberry Pi

Problem

Chcesz używać Arduino w charakterze płytka interfejsu podłączonej do Raspberry Pi.

Rozwiążanie

Podłącz Arduino do gniazda USB Raspberry Pi — pozwoli to na komunikację pomiędzy płytami, a także dostarczy prąd zasilający do Arduino.

Następnie do pamięci Arduino wgraj szkic *Firmata*, a na Raspberry Pi zainstaluj bibliotekę PyFirmata. Wymaga to wcześniejszego zainstalowania środowiska programistycznego Arduino. Jeżeli jeszcze tego nie zrobileś, wykonaj polecenia znajdujące się w recepturze 14.1.

Szkic *Firmata* jest dołączony do środowiska programistycznego Arduino. Aby go zainstalować, z menu *Plik* wybierz podmenu *Przykłady*, następnie *Firmata* i kliknij szkic *StandardFirmata*.

Po zainstalowaniu szkicu *Firmata* Arduino będzie oczekiwany na polecenia wysypane przez Raspberry Pi.

Teraz musisz zainstalować bibliotekę PyFirmata. Wymaga ona wcześniejszego zainstalowania biblioteki PySerial (w tym celu wykonaj polecenia znajdujące się w recepturze 8.8). Biblioteka PyFirmata znajduje się w serwisie GitHub, a więc w celu jej pobrania musisz zainstalować aplikację Git za pomocą polecenia sudo apt-get install git (zobacz receptura 3.19).

Teraz możesz przystąpić do instalacji PyFirmata. Uruchom następujące polecenia:

```
$ git clone https://github.com/tino/pyFirmata.git  
$ cd pyFirmata  
$ sudo python setup.py install
```

Działanie zainstalowanej biblioteki możesz sprawdzić za pomocą konsoli Pythona. Wprowadź poniższe polecenia w celu włączenia, a następnie wyłączenia diody LED oznaczonej literą *L*, znajdującej się na płytce Arduino (podłączonej do pinu o numerze 13).

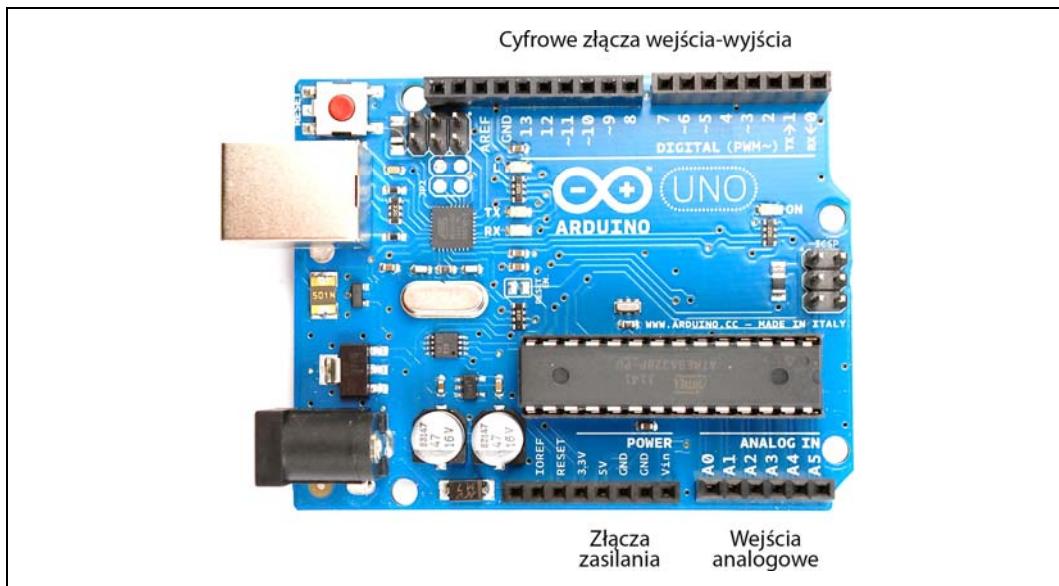
```
$ sudo python  
Python 2.7.3 (default, Jan 13 2013, 11:20:46)  
[GCC 4.6.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import pyfirmata  
>>> board = pyfirmata.Arduino('/dev/ttyACM0')  
>>> pin13 = board.get_pin('d:13:o')  
>>> pin13.write(1)  
>>> pin13.write(0)  
>>> board.exit()
```

Omówienie

Przedstawiony kod najpierw importuje bibliotekę PyFirmata, a następnie tworzy egzemplarz Arduino o nazwie *board*. Jako parametr przekazywany jest adres interfejsu USB (*/dev/ttyACM0*). Następnie aplikacja uzyskuje dostęp do jednego ze złączy znajdujących się na płytce Arduino (w tym przypadku do złącza o numerze 13) i konfiguruje je tak, aby działało ono jako cyfrowe wyjście. Parametr *d* oznacza wyjście cyfrowe, 13 — numer pinu, a parametr *o* określa, że złącze będzie pracować jako wyjście.

Aby podać na złączu wysoki sygnał, należy zastosować polecenie `write(1)`. Polecenie `write(0)` spowoduje podanie niskiego sygnału. W miejscu parametrów 1 i 0 możesz również podać parametry `True` i `False`.

Na rysunku 14.4 możesz zobaczyć rzędy złączy znajdujące się po obu stronach płytki Arduino.



Rysunek 14.4. Złącza wejścia i wyjścia znajdujące się na płytce Arduino

Złącza znajdujące się u góry płytki widocznej na rysunku 14.4 (oznaczone numerami od 0 do 13) mogą działać jako cyfrowe wejścia lub wyjścia. Niektóre z nich są używane również w innych celach. Pin 0 i 1 służą do komunikacji szeregowej. Są one zajęte, gdy korzystamy z portu USB. Do pinu numer 13 podłączona jest dioda LED oznaczona literą L. Złącza o numerach 3, 5, 6, 9, 10 i 11 są wyposażone w układ PWM — modulacji czasu trwania impulsu (zobacz oznaczenia na rysunku 14.4).

Po drugiej stronie płytki znajduje się zestaw złączy zasilających dostarczających prąd o napięciu 5 i 3,3 V, a także sześć wejść analogowych o oznaczeniach od A0 do A5.

Arduino pobiera prąd o natężeniu około 50 mA. Biorąc pod uwagę to, że Raspberry Pi pobiera dziesięć razy więcej prądu, Arduino można zasilać za pośrednictwem złącza USB znajdującego się na płytce Raspberry Pi. Jednak podłączenie dużej liczby zewnętrznych podzespołów elektronicznych może spowodować, że Arduino zacznie pobierać znacznie więcej prądu. W takim przypadku warto zasilić Arduino oddzielnym zasilaczem dostarczającym prąd stały o napięciu znajdującym się w zakresie od 7 do 12 V.

Jedyną wadą wspominanej wcześniej biblioteki jest to, że instrukcje muszą być stale dostarczane z Raspberry Pi. Biblioteka ta nie umożliwia samodzielnego pracy Arduino. Podczas pracy nad złożonymi projektami i tak prawdopodobnie będziesz musiał pisać własne programy uruchamiane na Arduino, które będą się komunikować z Raspberry Pi i wykonywać odbierane polecenia.

Zobacz również

Zagadnienia dotyczące korzystania ze wszystkich złączy Arduino za pomocą biblioteki PyFirmata są poruszane w recepturach 14.4, 14.6, 14.7, 14.8 i 14.9.

Oficjalną dokumentację aplikacji PyFirmata znajdziesz pod adresem <https://github.com/tino/pyFirmata/blob/master/README.rst>.

14.4. Sterowanie pracą cyfrowych wyjść Arduino za pomocą Raspberry Pi

Problem

Chcesz sterować cyfrowymi wyjściami Arduino z aplikacji Pythona uruchomionej na Raspberry Pi.

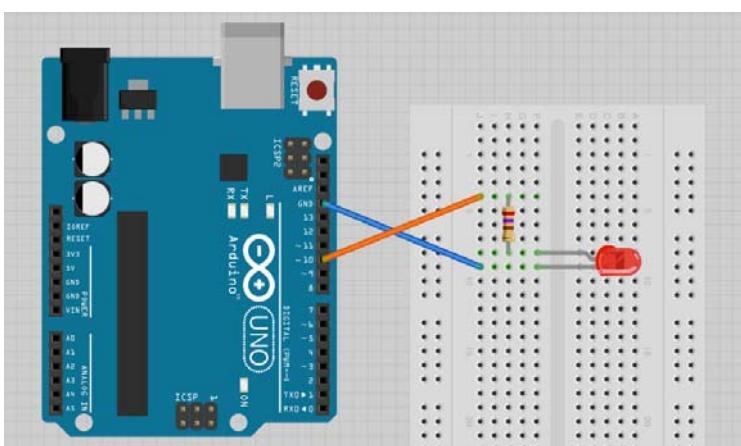
Rozwiązanie

W recepturze 14.3 zaprezentowałem program zapalający diodę LED (oznaczoną literą *L*) znajdująca się na płytce Arduino. Teraz podłączymy do Arduino zewnętrzną diodę LED, a następnie będziemy nią migać.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora 270 Ω (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- diody LED (zobacz sekcja „Optoelektronika” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej podłączonej do Arduino przedstawiono na rysunku 14.5. Wykonaj obwód, korzystając z tego rysunku.



Rysunek 14.5. Dioda LED podłączona do Arduino

Jeżeli jeszcze tego nie zrobiłeś, zainstaluj bibliotekę PyFirmata, korzystając z receptury 14.3.

Poniższy skrypt Pythona sprawi, że dioda LED będzie migać z częstotliwością około 1 Hz. Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstem nano i zapisz plik pod nazwą *ardu_flash.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
import time

board = pyfirmata.Arduino('/dev/ttyACM0')
led_pin = board.get_pin('d:10:o')

while True:
    led_pin.write(1)
    time.sleep(0.5)
    led_pin.write(0)
    time.sleep(0.5)
```

Omówienie

Opisywana czynność przypomina podłączanie diody LED bezpośrednio do Raspberry Pi (zobacz receptura 9.1). W związku z tym, że Arduino potrafi dostarczyć diodzie więcej prądu niż Raspberry Pi, możesz korzystać z rezystorów charakteryzujących się mniejszym oporem — dioda będzie świecić trochę jaśniej. Złącza Arduino podają prąd o napięciu 5 V (Raspberry Pi dostarczało prąd o napięciu 3,3 V). Dioda w tym układzie będzie pobierała prąd o cztery razy większym natężeniu od diody w recepturze 9.1.

Jeżeli chcesz sterować diodą LED za pomocą graficznego interfejsu przedstawionego w recepturze 9.7 (zobacz rysunek 14.6), to możesz zmodyfikować podany tam kod. Zmodyfikowany program zapisaliśmy w pliku *ardu_gui_switch.py*. Pamiętaj o tym, że programu tego nie uruchomisz za pośrednictwem protokołu SSH. Musisz to zrobić w graficznym interfejsie Raspberry Pi.



Rysunek 14.6. Graficzny interfejs pozwalający na włączanie i wyłączanie elektroniki

Zobacz również

Sterowanie diodą LED podłączoną bezpośrednio do Raspberry Pi opisano w recepturze 9.1.

14.5. Sterowanie Arduino za pomocą biblioteki PyFirmata za pośrednictwem portu szeregowego

Problem

Za pomocą biblioteki PyFirmata chcesz sterować pracą Arduino podłączonego do złączy RXD i TXD gniazda GPIO.

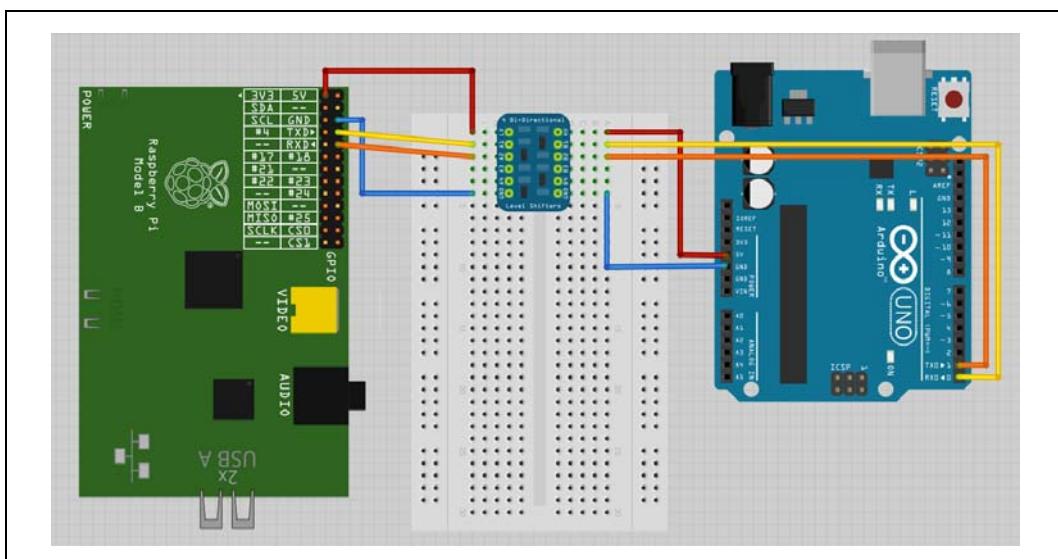
Rozwiązanie

Korzystając z układu ściągającego napięcie, połącz pin *RXD* złącza GPIO z pinem *Tx* znajdującym się na płytce Arduino oraz pin *TXD* złącza GPIO z pinem *Rx* płytki Arduino.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystorów $270\ \Omega$ i $470\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A) lub czterokanałowego dwukierunkowego modułu dokonującego konwersji sygnału (zobacz sekcję „Moduły” w dodatku A).

Jeżeli korzystasz z modułu dokonującego konwersji sygnału, to elementy układu połącz tak jak pokazano na rysunku 14.7.

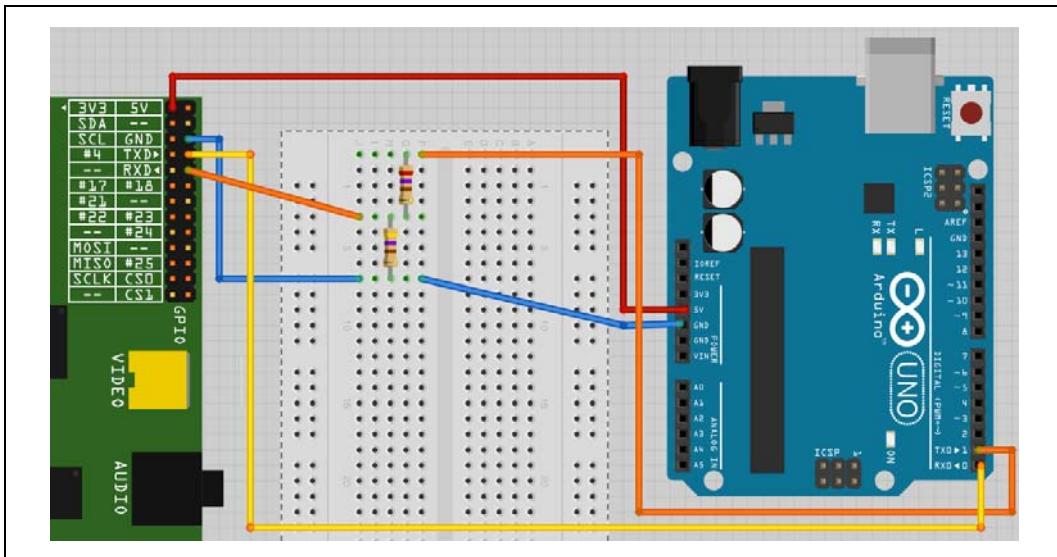


Rysunek 14.7. Arduino podłączone do portu szeregowego Raspberry Pi za pośrednictwem modułu dokonującego konwersji sygnału

Jeżeli zamiast modułu stosujesz parę rezystorów, połącz elementy układu, korzystając z rysunku 14.8.

Na wejście *Rx* Arduino można podawać sygnał o napięciu 3,3 V generowany przez złącze *TXD* Raspberry Pi. Jednak napięcie (5 V) sygnału generowanego na złączu *Tx* Arduino musi być obniżone do poziomu 3 V.

Musisz skonfigurować bibliotekę PyFirmata (zobacz receptura 14.3). W tym projekcie nie dokonujemy żadnych zmian w konfiguracji Arduino. Pozostaje ona taka sama jak w recepturze 14.4, w której zamiast portu szeregowego korzystaliśmy ze złącza USB. Musisz dokonać jednej zmiany w uruchamianej aplikacji Pythona. Zmień nazwę urządzenia z `/dev/ttyACM0` na `/dev/ttyAMA0` — port szeregowy ma inny adres od interfejsu USB.



Rysunek 14.8. Arduino podłączone do portu szeregowego Raspberry Pi za pośrednictwem dwóch rezystorów

Poniższy skrypt Pythona sprawi, że dioda LED będzie migać z częstotliwością około 1 Hz. Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_flash_ser.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
import time

board = pyfirmata.Arduino('/dev/ttyAMA0')
led_pin = board.get_pin('d:13:o')

while True:
    led_pin.write(1)
    time.sleep(0.5)
    led_pin.write(0)
    time.sleep(0.5)
```

Omówienie

Konwersja poziomu sygnału jest konieczna, ponieważ złącza portu szeregowego Raspberry Pi (RXD i TXD) działają pod napięciem 3,3 V, a Arduino Uno działa pod napięciem 5 V. Arduino potrafi odbierać sygnał o napięciu 3 V, a podłączenie sygnału o napięciu 5 V do gniazda RXD Raspberry Pi mogłoby doprowadzić do uszkodzenia tego urządzenia.

Zobacz również

W równie łatwy sposób możesz zaadaptować do pracy z interfejsem szeregowym inne programy korzystające z biblioteki PyFirmata (receptury od 14.6 do 14.9). Wystarczy zmienić nazwę urządzenia.

14.6. Odczytywanie danych z cyfrowych wejść Arduino za pomocą biblioteki PyFirmata

Problem

Chcesz odczytywać sygnały podawane do cyfrowych wejść Arduino za pomocą aplikacji Pythona uruchomionej na Raspberry Pi.

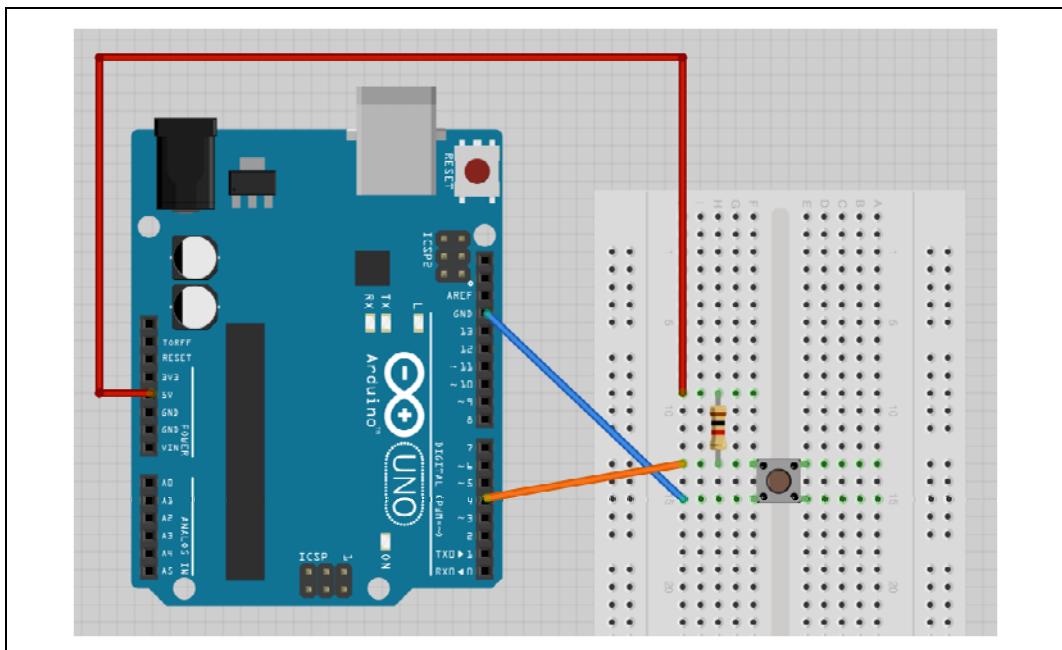
Rozwiązanie

W celu odczytania sygnałów podawanych do cyfrowych złączy Arduino zastosuj bibliotekę PyFirmata.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytka prototypowa i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- przełącznika chwilowego (zobacz sekcja „Pozostałe” w dodatku A).

Elementy układu połącz ze sobą, korzystając ze schematu znajdującego się na rysunku 14.9.



Rysunek 14.9. Przełącznik chwilowy podłączony do Arduino

Jeżeli jeszcze tego nie zrobiłeś, skonfiguruj bibliotekę PyFirmata, korzystając z receptury 14.3.

Poniższy skrypt wyświetla komunikat po każdorazowym wciśnięciu przycisku. Jest on bardzo podobny do programu *switch.py* omówionego w recepturze 11.1. Otwórz edytor (nano lub IDLE) i umieść w jego oknie poniższy kod. Następnie zapisz plik pod nazwą *ardu_switch.py*. Plik ten, jak również pliki zawierające inne aplikacje omówione w tej książce, można pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
import time
import sys

board = pyfirmata.Arduino('/dev/ttyACM0')
switch_pin = board.get_pin('d:4:i')
it = pyfirmata.util.Iterator(board)
it.start()
switch_pin.enable_reporting()

while True:
    input_state = switch_pin.read()
    if input_state == False:
        print('Wcisñasz przycisk')
        time.sleep(0.2)
```

Po uruchomieniu programu należy odczekać jedną lub dwie sekundy, aż szkic *Firmata* nawiążę połączenie z Raspberry Pi. Po tym czasie każdorazowe wciśnięcie przycisku spowoduje wyświetlenie przez program komunikatu.

```
$ sudo python ardu_switch.py
Wcisñasz przycisk
Wcisñasz przycisk
Wcisñasz przycisk
```

Omówienie

PyFirmata korzysta z wątku *Iterator* w celu monitorowania złącza wejściowego. Zabieg taki wynika z implementacji programu *Firmata*. Nie możesz tak po prostu odczytać sygnału podawanego na wejście. W tym celu musisz stworzyć oddzielny wątek *Iterator*, który będzie odczytywał stan przełącznika:

```
it = pyfirmata.util.Iterator(board)
it.start()
```

Następnie musisz uruchomić rapportowanie stanu interesującego Cię złącza za pomocą polecenia:

```
switch_pin.enable_reporting()
```

Ubocznym efektem tego mechanizmu jest to, że po wciśnięciu kombinacji klawiszy *Ctrl+C* program nie zostanie prawidłowo zamknięty. Wątek *Iterator* można wyłączyć tylko poprzez otwarcie innego okna Terminala lub sesji SSH (zobacz receptura 3.24).

Jeżeli ten program jest jedyną uruchomioną aplikacją Pythona, możesz zakończyć jego działanie za pomocą polecenia:

```
$ sudo killall python
```

Odlączenie płytka Arduino od Raspberry Pi (przerwanie przesyłu danych) również spowoduje zamknięcie aplikacji Pythona.

Zobacz również

Podłączanie przycisku za pośrednictwem Arduino przypomina podłączanie go bezpośrednio do Raspberry Pi (zobacz receptura 11.1). O ile nie podłączasz większej liczby przycisków, to korzystanie z Arduino w ten sposób nie przyniesie Ci żadnych korzyści.

14.7. Odczytywanie danych z analogowych wejść Arduino za pomocą biblioteki PyFirmata

Problem

Chcesz odczytywać sygnały podawane do analogowych wejść Arduino za pomocą aplikacji Pythona uruchomionej na Raspberry Pi.

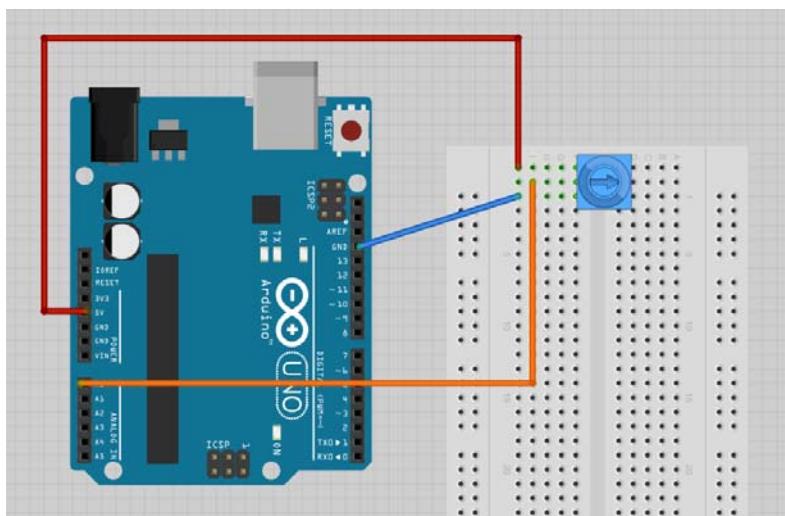
Rozwiązanie

W celu odczytania sygnałów podawanych do analogowych wejść Arduino zastosuj bibliotekę PyFirmata.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora dostrojczego $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Elementy układu połącz ze sobą przy użyciu płytki prototypowej, posługując się schematem znajdującym się na rysunku 14.10.



Rysunek 14.10. Potencjometr dostrojczy podłączony do Raspberry Pi

Jeżeli jeszcze tego nie zrobiłeś, skonfiguruj bibliotekę PyFirmata, korzystając z receptury 14.3.

Poniższy skrypt Pythona (*ardu_adc.py*) wyświetla nieprzetworzony stan odczytywany ze złącza oraz podaje obliczoną na jego podstawie wartość napięcia (U). Program ten jest podobny do programu *adc_test.py* przedstawionego w recepturze 12.4.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_adc.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
import time

board = pyfirmata.Arduino('/dev/ttyACM0')
analog_pin = board.get_pin('a:0:i')
it = pyfirmata.util.Iterator(board)
it.start()
analog_pin.enable_reporting()

while True:
    reading = analog_pin.read()
    if reading != None:
        voltage = reading * 5.0
        print("Odczyt=%f\tU=%f" % (reading, voltage))
    time.sleep(1)
```

Wartość odczytywana ze złącza analogowego będzie się znajdowała w przedziale od 0,0 do 1,0.

```
$ sudo python ardu_adc.py
Odczyt=0.000000    U=0.000000
Odczyt=0.165200    U=0.826000
Odczyt=0.784000    U=3.920000
Odczyt=1.000000    U=5.000000
```

Omówienie

Program ten jest bardzo podobny do programu przedstawionego w recepturze 14.6. Występuje w nim Iterator i dotyczą go takie same problemy związane z zamknięciem okna.

W programie niezbędne było zastosowanie polecenia warunkowego if. Gdyby polecenie read zostało wykonane przed właściwym odczytaniem wartości z analogowego wejścia, to funkcja ta zamiast wartości zwróciłaby None. Polecenie if zapobiega błędom powstały w wyniku nieodczytania wartości.

Zobacz również

Informacje na temat korzystania z wejść cyfrowych znajdziesz w recepturze 14.6.

14.8. Obsługa wyjść analogowych (PWM) za pomocą biblioteki PyFirmata

Problem

Chcesz sterować jasnością diody LED za pomocą układu PWM wbudowanego w Arduino.

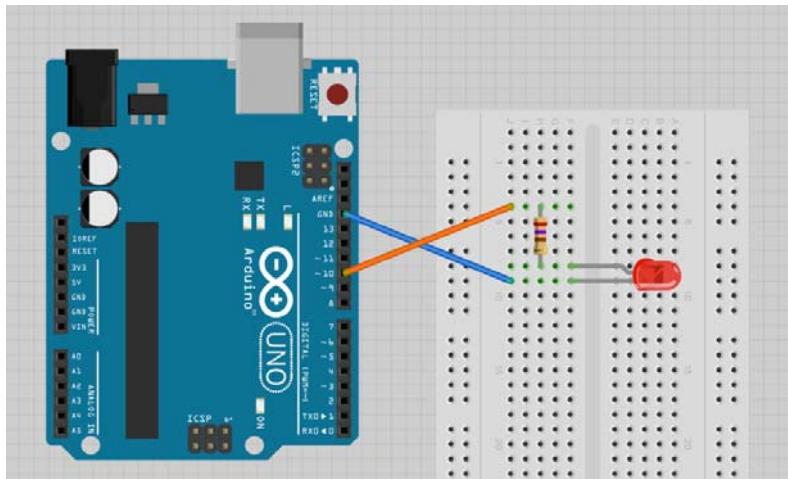
Rozwiązanie

Zastosuj bibliotekę PyFirmata w celu wysyłania polecenia generowania sygnału PWM na jednym z wyjść Arduino.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora 270 Ω (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- diody LED (zobacz sekcja „Optoelektronika” w dodatku A).

Ułożenie elementów obwodu na płytce prototypowej podłączonej do Arduino przedstawiono na rysunku 14.11. Wykonaj obwód, korzystając z tego rysunku. Identyczny obwód przedstawiono wcześniej w recepturze 14.4.



Rysunek 14.11. Sterowanie diodą LED za pomocą modulacji czasu trwania impulsu

Jeżeli jeszcze tego nie zrobiłeś, skonfiguruj bibliotekę PyFirmata, korzystając z receptury 14.3.

Poniższy skrypt Pythona (*ardu_pwm.py*) poprosi Cię o podanie jasności, z jaką ma świecić dioda LED, a następnie spowoduje wygenerowanie przez Arduino odpowiedniego impulsu PWM. Program ten jest podobny do programu omówionego w recepturze 9.2.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_pwm.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
board = pyfirmata.Arduino('/dev/ttyACM0')
led_pin = board.get_pin('d:10:p')

while True:
    duty_s = raw_input("Podaj jasność (od 0 do 100):")
    duty = int(duty_s)
    led_pin.write(duty / 100.0)
```

Po wprowadzeniu liczby 100 dioda LED powinna świecić z maksymalną jasnością. Im mniejsza wartość zostanie podana, tym ciemniejsze będzie światło generowane przez diodę.

```
$ sudo python ardu_pwm.py  
Podaj jasnosc (od 0 do 100):100  
Podaj jasnosc (od 0 do 100):50  
Podaj jasnosc (od 0 do 100):10  
Podaj jasnosc (od 0 do 100):5  
Podaj jasnosc (od 0 do 100):
```

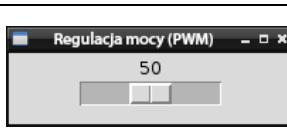
Omówienie

Program ten jest bardzo prosty. Poniższe polecenie aktywuje działanie układu PWM na danym wyjściu:

```
led_pin = board.get_pin('d:10:p')
```

Parametr p aktywuje układ PWM. Pamiętaj o tym, że tylko odpowiednio oznaczone złącza mogą pracować w trybie modulacji czasu trwania impulsu.

Mozesz również zmodyfikować graficzny program sterujący mocą przedstawiany w recepturze 9.8, tak aby regulacja korzystała z biblioteki PiFirmata (zobacz rysunek 14.12). Gotowy program znajdziesz na stronie <http://www.helion.pl/ksiazki/raspre.htm> pod nazwą *ardu_gui_slider.py*.



Rysunek 14.12. Program sterujący jasnością diody wyposażony w graficzny interfejs użytkownika

Zobacz również

Arduino potrafi dostarczyć na złączu wyjściowym prąd o natężeniu 40 mA. Jest to dziesięć razy więcej niż Raspberry Pi, ale i tak jest to zbyt mało, żeby bezpośrednio zasilić silnik lub moduł diody LED o dużej mocy. Takie urządzenia należy zasilać za pomocą obwodu przedstawionego w recepturze 9.4. Wystarczy go podłączyć do wyjścia Arduino, a nie do gniazda GPIO.

14.9. Sterowanie pracą serwomotoru za pomocą biblioteki PyFirmata

Problem

Chcesz sterować pracą serwomotoru za pośrednictwem Arduino.

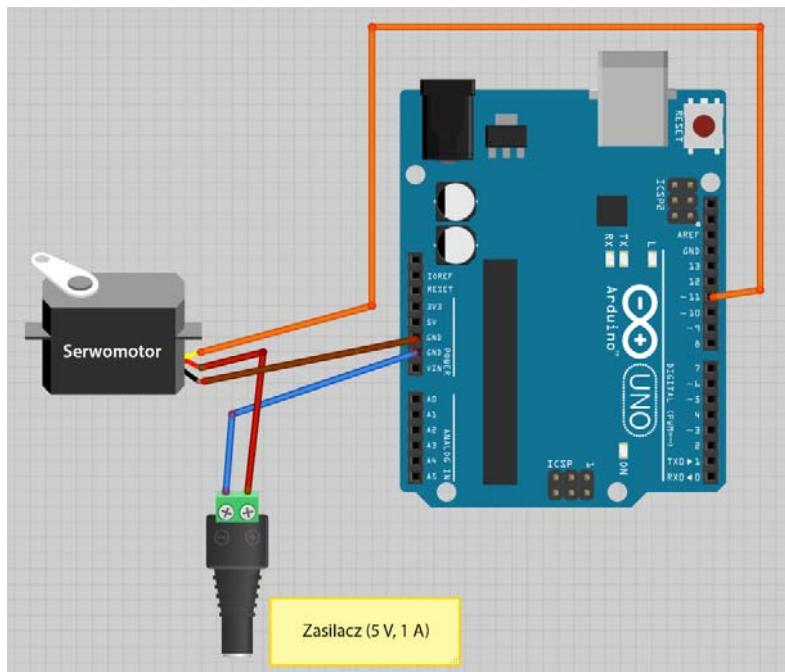
Rozwiązanie

Zastosujemy bibliotekę PyFirmata w celu wysyłania do Arduino polecień niezbędnych do generowania impulsów sterujących położeniem ramienia serwomotoru.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytka prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- rezystora $1\text{ k}\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- diody LED (zobacz sekcja „Optoelektronika” w dodatku A).

Korzystając z rysunku 14.13, połącz ze sobą elementy obwodu.



Rysunek 14.13. Schemat wykonawczy obwodu sterowania pracą serwomotoru za pomocą Arduino

Jeżeli jeszcze tego nie zrobiłeś, skonfiguruj bibliotekę PyFirmata, korzystając z receptury 14.3.

Poniższy skrypt Pythona (*ardu_servo.py*) poprosi Cię o podanie kąta, pod jakim ma zostać ustalone ramię serwomotoru, a następnie wykona określony ruch podłączonym silnikiem.

Umieść poniższy kod programu w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_servo.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import pyfirmata
board = pyfirmata.Arduino('/dev/ttyACM0')
servo_pin = board.get_pin('d:11:s')

while True:
    angle_s = raw_input("Podaj kat (od 0 do 180 stopni):")
    angle = int(angle_s)
    servo_pin.write(angle)
```

Wpisanie kątów 0 i 180 spowoduje ustawienie ramienia serwomotoru w skrajnych położeniach, a podanie wartości 90 spowoduje ustawienie ramienia serwomotoru na środku, pomiędzy skrajnymi położeniami.

```
$ sudo python ardu_servo.py  
Podaj kat (od 0 do 180 stopni):0  
Podaj kat (od 0 do 180 stopni):180  
Podaj kat (od 0 do 180 stopni):90
```

Omówienie

Przedstawiony program jest dość prosty. Wyjście, do którego podłączony jest serwomotor, jest definiowane przez polecenie:

```
led_pin = board.get_pin('d:11:s')
```

Parametr s określa, że do złącza podłączono serwomotor. Można go używać ze wszystkimi cyfrowymi wyjściami znajdującymi się na płytce Arduino.

Jeżeli wcześniej wykonałeś obwód przedstawiony w recepturze 10.1, to teraz możesz porównać pracę serwomotoru w obu obwodach. Ramię serwomotoru podłączonego do Arduino nie drga.

Zobacz również

W recepturze 10.1 znajdziesz instrukcję podłączenia serwomotoru bezpośrednio do płytki Raspberry Pi.

Jeżeli chcesz sterować pracą wielu serwomotorów, to zamiast płytki Arduino możesz skorzystać z modułu sterownika serwomotorów opisanego w recepturze 10.2.

14.10. Komunikacja pomiędzy Raspberry Pi a Arduino za pośrednictwem interfejsu szeregowego bez użycia biblioteki PyFirmata

Problem

Chcesz przesyłać dane pomiędzy Arduino a Raspberry Pi za pośrednictwem interfejsu szeregowego bez użycia biblioteki PyFirmata.

Rozwiążanie

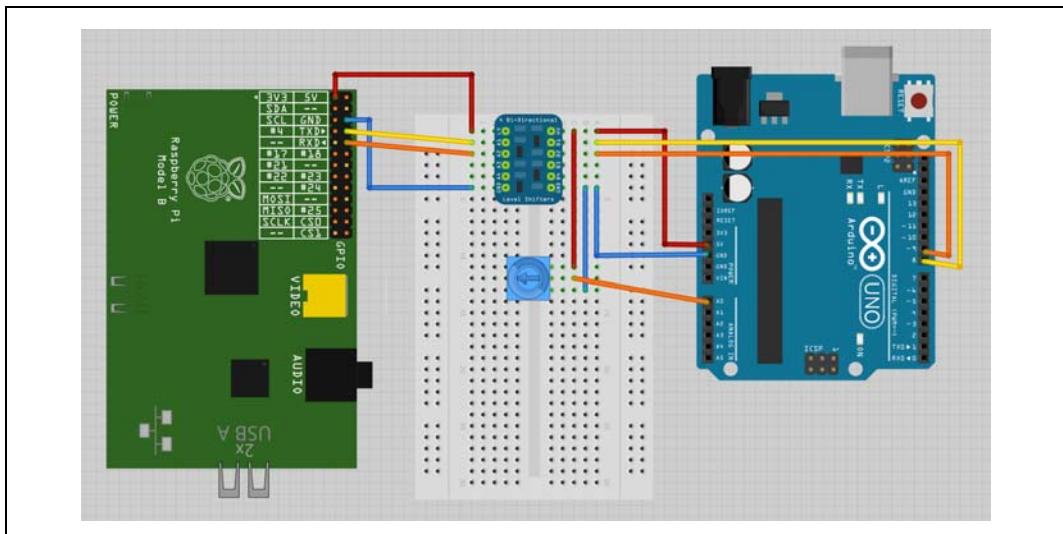
Skorzystaj z modułu dokonującego konwersji napięcia sygnałów lub z pary rezystorów w celu połączenia złącza RXD znajdującego się w gnieździe GPIO ze złączem Tx Arduino. Połącz również złącze TXD będące częścią gniazda GPIO ze złączem Rx Arduino.

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),

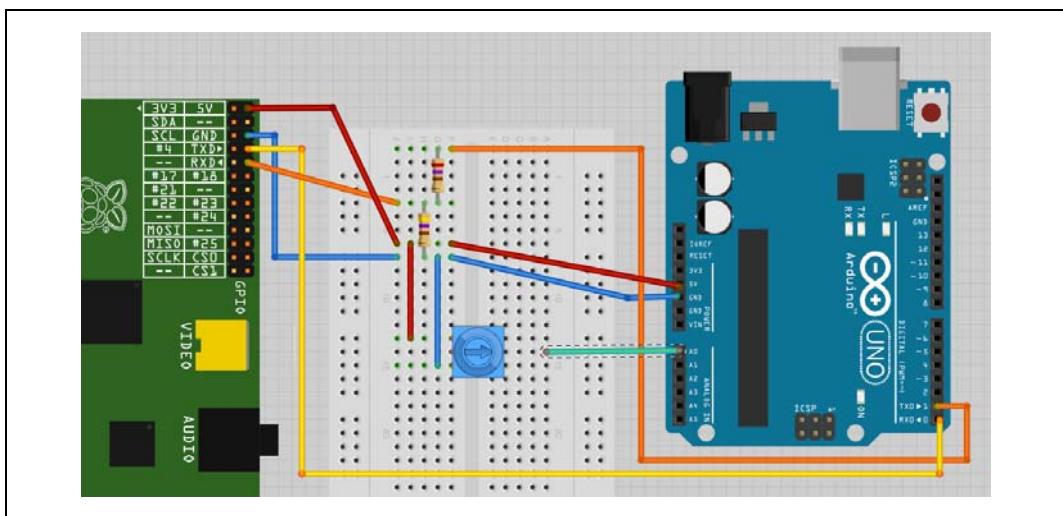
- rezystorów $270\ \Omega$ i $470\ \Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A) lub czterokanałowego dwukierunkowego modułu dokonującego konwersji sygnału (zobacz sekcję „Moduły” w dodatku A),
- potencjometru dostrojczego $10\ k\Omega$ (zobacz sekcja „Rezystory i kondensatory” w dodatku A).

Jeżeli korzystasz z modułu dokonującego konwersji sygnału, to elementy układu połącz tak jak pokazano na rysunku 14.14.



Rysunek 14.14. Arduino podłączone do portu szeregowego Raspberry Pi za pośrednictwem modułu dokonującego konwersji sygnału

Jeżeli zamiast modułu stosujesz parę rezystorów, to połącz elementy układu, korzystając z rysunku 14.15.



Rysunek 14.15. Arduino podłączone do portu szeregowego Raspberry Pi za pośrednictwem dwóch rezystorów

Na wejście Rx Arduino można podawać sygnał o napięciu 3,3 V generowany przez złącze TXD Raspberry Pi. Jednak napięcie (5 V) sygnału generowanego na złączu Tx Arduino musi być obniżone do poziomu 3 V.

Musisz wyłączyć rejestrację danych na porcie szeregowym i zainstalować bibliotekę PySerial (wykonaj receptury 8.7 i 8.8).

Poniższy skrypt Pythona (*ardu_pi_serial.py*) poprosi Cię o wpisanie polecenia. Polecenie l włącza i wyłącza diodę LED znajdująca się na płytce Arduino. Polecenie r spowoduje odczytanie stanu wejścia analogowego A0 i przesłanie do Raspberry Pi odczytanej wartości znajdującej się w przedziale od 0 do 1023.

Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_pi_serial.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
import serial

ser = serial.Serial('/dev/ttyAMA0', 9600)

while True:
    command = raw_input("Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0")
    if command == 'l' :
        ser.write('l')
    elif command == 'r' :
        ser.write('r')
        print(ser.readline())
```

Musisz również załadować poniższy szkic do pamięci Arduino. Wpisz go w nowym oknie środowiska programistycznego Arduino lub pobierz w postaci pliku *ArduinoSerial.ino* ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```
#include "SoftwareSerial.h"

int ledPin = 13;
int analogPin = A0;

SoftwareSerial ser(8, 9); // RX, TX

boolean ledOn = false;

void setup()
{
    ser.begin(9600);
    pinMode(ledPin, OUTPUT);
}

void loop()
{
    if (ser.available())
    {
        char ch = ser.read();
        if (ch == 'l')
        {
            toggleLED();
        }
        if (ch == 'r')
        {
            sendAnalogReading();
        }
    }
}
```

```

}

void toggleLED()
{
    ledOn = ! ledOn;
    digitalWrite(ledPin, ledOn);
}

void sendAnalogReading()
{
    int reading = analogRead(analogPin);
    ser.println(reading);
}

```

Uruchom program i sprawdź, czy wpisanie polecenia l sprawi, że dioda znajdująca się na płytce Arduino będzie zapalana i gaszona. Następnie za pomocą polecenia r odczytuj wartości odbierane przez wejście A0. Przed dokonaniem kolejnych odczytów zmień ustawienie potencjometru dostrojczego.

```

$ sudo python ardu_pi_serial.py
Wpisz polecenie: l - przelacz diode LED, r - odczytaj AO l
Wpisz polecenie: l - przelacz diode LED, r - odczytaj AO l
Wpisz polecenie: l - przelacz diode LED, r - odczytaj AO r
0
Wpisz polecenie: l - przelacz diode LED, r - odczytaj AO r
540
Wpisz polecenie: l - przelacz diode LED, r - odczytaj AO r
1023

```

Omówienie

W przytoczonym programie połączenie z portem szeregowym jest nawiązywane za pośrednictwem biblioteki PySerial. Główna pętla programu powtarza prośbę o wpisanie polecenia przez użytkownika, a następnie przetwarza wprowadzone dane. W obu przypadkach polecenie składa się z jednego znaku przekazywanego za pośrednictwem portu szeregowego.

Program po uruchomieniu polecenia r odczytuje dane przychodzące do Raspberry Pi za pośrednictwem połączenia szeregowego, a następnie je wyświetla.

Zwróć uwagę na to, że zmienna `serial.readline` jestłańcuchem. Jeżeli chcesz dokonać konwersji tego łańcucha na liczbę, możesz zastosować funkcję `int`, np.:

```

line = ser.readline()
value = int(line)

```

Szkic uruchamiany przez Arduino jest nieco bardziej złożony. Może on zostać łatwo zmodyfikowany w celu dodania większej liczby wejść lub wyjść. Może on również obsługiwać więcej poleceń.

Szkic nie korzysta ze sprzętowego portu szeregowego, który jest zwykle używany przez połączenie USB. Zamiast niego zastosowano bibliotekę o nazwie SoftwareSerial, która pozwala na używanie pinów o numerach 8 i 9 w charakterze odpowiednio złączą odbiorczego (*Rx*) i nadawczego (*Tx*).

Wszystkie szkice Arduino muszą posiadać funkcję `setup`, która jest wywoływana jednorazowo po uruchomieniu Arduino, a także funkcję `loop`, która jest wywoywana w nieskończoność.

Funkcja `setup` uruchamia komunikację szeregową i inicjuje pin o numerze 13, do którego podłączona jest wbudowana dioda LED, tak aby działał on w charakterze wyjścia.

Funkcja `loop` wywołuje funkcję `ser.available`, która sprawdza, czy poprzez interfejs szeregowy nadchodzą dane. Odebrane dane są przetwarzane — odczytywany jest znak wprowadzony przez użytkownika, a instrukcja warunkowa `if` uruchamia dalsze polecenia w zależności od wprowadzonego znaku.

Zobacz również

Więcej informacji na temat biblioteki `SoftwareSerial` znajdziesz na stronie <http://arduino.cc/en/Reference/SoftwareSerial>.

Zamiast tworzyć własny kod służący do komunikacji, możesz skorzystać z gotowej biblioteki `PyFirmata` (zobacz receptura 14.3).

Poza komunikacją za pośrednictwem interfejsu szeregowego z Arduino możesz się również komunikować poprzez magistralę I2C (zobacz receptura 14.11).

14.11. Tworzenie programu komunikującego się z Arduino za pośrednictwem magistrali I2C

Problem

Chcesz przesyłać dane pomiędzy Arduino i Raspberry Pi za pośrednictwem magistrali I2C.

Rozwiązanie

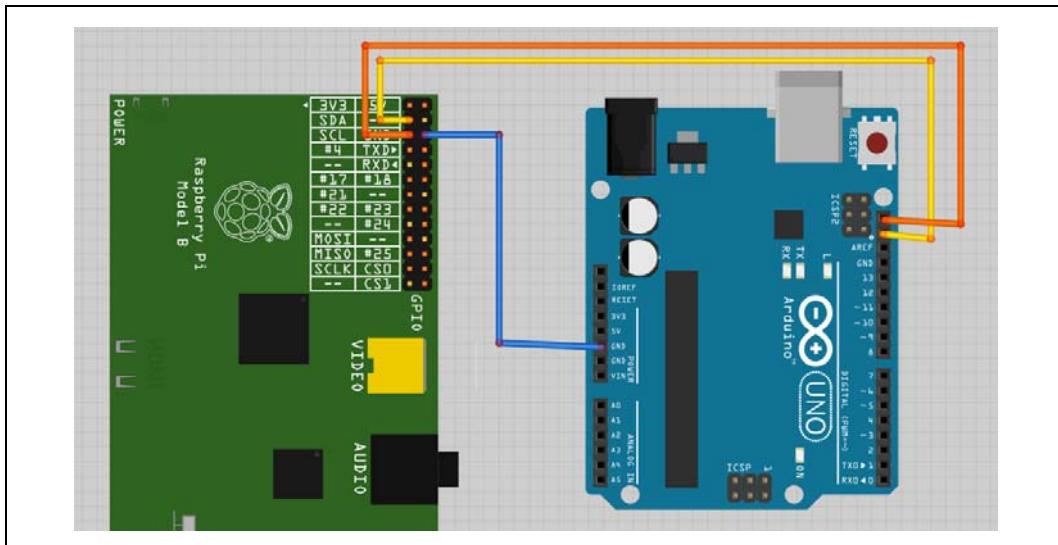
W przypadku magistrali I2C mamy do czynienia z urządzeniami nadrzędnymi i podrzędnymi. Urządzenie nadrzędne steruje pracą magistrali i wieloma urządzeniami podrzędnymi, które mogą zostać do niej podłączone. Każde urządzenie podrzędne posiada swój własny adres.

Na Arduino należy zainstalować szkic, który sprawi, że będzie ono pracowało jako urządzenie podrzędne. Musisz również napisać program dla Raspberry Pi, który będzie korzystać z biblioteki I2C. Urządzenia należy połączyć za pomocą złączy *SDA*, *SCL* i masy. Arduino należy ponadto podłączyć do zewnętrznego zasilacza dostarczającego prąd o napięciu 5 V (w tej roli można wykorzystać urządzenie zasilające Raspberry Pi).

Aby skorzystać z niniejszej receptury, potrzebujesz:

- Arduino Uno (zobacz sekcja „Moduły” w dodatku A),
- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A).

Połącz ze sobą płytki, korzystając ze schematu znajdującego się na rysunku 14.16.



Rysunek 14.16. Arduino i Raspberry Pi połączone ze sobą za pomocą magistrali I2C

Korzystam z najnowszej płytki Arduino Uno R3. Jeżeli masz starszą płytkę, która nie ma dedykowanych złącz SCL i SDA, to złącza SDA i SCL znajdujące się na płytce Raspberry Pi połącz ze złączami A4 i A5 na płytce Arduino.

Wgraj do pamięci Arduino poniższy szkic. Możesz go również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm> w postaci pliku o nazwie *Arduinol2C.ino*.

```
#include <Wire.h>

int SLAVE_ADDRESS = 0x04;
int ledPin = 13;
int analogPin = A0;

boolean ledOn = false;

void setup()
{
    pinMode(ledPin, OUTPUT);
    Wire.begin(SLAVE_ADDRESS);
    Wire.onReceive(processMessage);
    Wire.onRequest(sendAnalogReading);
    Serial.begin(9600);
}

void loop()
{
}

void processMessage(int n)
{
    Serial.println("Wykonuje processMessage");
    char ch = Wire.read();
    if (ch == '1')
    {
        toggleLED();
    }
}
```

```

}

void toggleLED()
{
    ledOn = ! ledOn;
    digitalWrite(ledPin, ledOn);
}

void sendAnalogReading()
{
    Serial.println("Wykonuje sendAnalogReading");
    int reading = analogRead(analogPin);
    Wire.write(reading >> 2);
}

```

Musisz skonfigurować obsługę magistrali I2C — wykonaj polecenia zawarte w recepturze 8.4.

Umieść poniższy kod w oknie środowiska programistycznego IDLE lub w edytorze tekstowym nano i zapisz plik pod nazwą *ardu_pi_i2c.py*. Gotowy plik z kodem możesz również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm>.

```

import smbus
import time

# w przypadku Raspberry Pi w wersji 1 zastosuj "bus = smbus.SMBus(0)"
bus = smbus.SMBus(1)

# Musi być taki sam jak w szkicu Arduino
SLAVE_ADDRESS = 0x04

def request_reading():
    reading = int(bus.read_byte(SLAVE_ADDRESS))
    print(reading)

while True:
    command = raw_input("Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0 ")
    if command == 'l' :
        bus.write_byte(SLAVE_ADDRESS, ord('l'))
    elif command == 'r' :
        request_reading()

```

Program testowy jest bardzo podobny do programu demonstrującego komunikację za pośrednictwem portu szeregowego przedstawionego w recepturze 14.10. Aplikacja została zaprojektowana tak, aby pokazać przepływ danych w obu kierunkach. Polecenie **l** włącza i wyłącza diodę LED znajdująca się na płytce Arduino, a polecenie **r** wyświetla stan analogowego wejścia *A0*. W celu uzyskania różnych odczytów możesz połączyć to wejście za pomocą przewodów połączeniowych obustronnie zakończonych wtykami żeńskimi ze złączami zasilającymi znajdującymi się na płytce Arduino (3,3 V, 5 V, masa).

```

$ sudo python ardu_pi_i2c.py
Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0 l
Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0 l
Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0 r
184
Wpisz polecenie: l - przełącz diode LED, r - odczytaj A0

```

Omówienie

W zastosowanym układzie Arduino działa w charakterze urządzenia podlegającego do magistrali I2C. Do Arduino musi zostać przypisany *adres* — pozwoli to na identyfikację Arduino przez Raspberry Pi (urządzenie nadziedzne), gdy do magistrali zostanie podłączonych więcej urządzeń. W przedstawionym przykładzie Arduino przypisano adres 0x04, który jest przechowywany w zmiennej `SLAVE_ADDRESS`.

W funkcji `setup` znajdują się dwie funkcje wywołania zwrotnego, które zostaną użyte w dalszej części programu. Funkcja `processMessage` jest wywoływaną za każdym razem, gdy dojdzie do zdarzenia `onReceive` — gdy użytkownik wyśle polecenie za pośrednictwem Raspberry Pi. Druga funkcja wywołania zwrotnego — `sendAnalogReading` — jest skojarzona ze zdarzeniem `onRequest`. Do zdarzenia tego dochodzi, gdy Raspberry Pi potrzebuje danych (wartości odbieranej przez analogowe wejście). Wartość ta zostanie następnie podzielona przez cztery (aby mogła być zapisana w postaci pojedynczego bajta), a następnie przesłana do Raspberry Pi.

Współdziałażąca z Arduino aplikacja Pythona tworzy nowy egzemplarz `SMBus` o nazwie `bus`. Przyjmuje on jeden argument — `1` — adres portu I2C używanego przez Raspberry Pi. Jeżeli posiadasz jedną z pierwszych wersji Raspberry Pi (rev 1), powinieneś zastosować `0` w roli tego argumentu. Starsze płytki Raspberry Pi były wyposażone w czarne złącze audio, a nowsze w niebieskie. W nowszym modelu Raspberry Pi zmieniono numery złączy portu GPIO, które obsługują port I2C.

Program prosi użytkownika o wprowadzenie polecenia (`l` lub `r`). W przypadku polecenia `l` program przekazuje do Arduino znak `1`, a Arduino uruchamia procedurę obsługi `onReceive` (`processMessage`). Spowoduje to uruchomienie funkcji `toggleLED`.

Jeżeli użytkownik wprowadzi polecenie `r`, zostanie wywołana funkcja `request_reading`. Uruchomi ona polecenie `read_byte` znajdujące się w bibliotece `SMbus`. W ten sposób zostanie wywołane wydarzenie `onRequest` w szkicu Arduino.

Może Cię zainteresować, dlaczego tym razem podłączamy Arduino pracujące pod napięciem 5 V bezpośrednio do złączy magistrali I2C znajdujących się na płytce Raspberry Pi i nie korzystamy z modułu dokonującego konwersji napięć (stosowaliśmy go przy połączeniu szeregowym w recepturze 14.5). Magistrala I2C działa przy dowolnym napięciu generowanym przez rezystory podciągające podłączone do złączy `SDA` i `SCL`. W tym przypadku płytki Arduino Uno nie mają żadnych rezystorów podciągających podłączonych do złączy interfejsu I2C. Rezystory podciągające znajdują się na płytce Raspberry Pi — podwyższają napięcie do 3,3 V.



Arduino nie ma rezystorów podciągających podłączonych do linii magistrali I2C, ale nie jest to ogólna zasada dotycząca urządzeń wyposażonych w interfejs I2C. Jeżeli posiadane przez Ciebie urządzenie działa pod napięciem 5 V, to upewnij się, że nie zostało ono wyposażone w rezystory podciągające. Rezystory takie są łatwe do zdementowania.

Zobacz również

Problematykę połączenia Raspberry Pi i Arduino za pośrednictwem interfejsu szeregowego omówiono w recepturze 14.10.

14.12. Podłączanie do Raspberry Pi mniejszych płyt Arduino

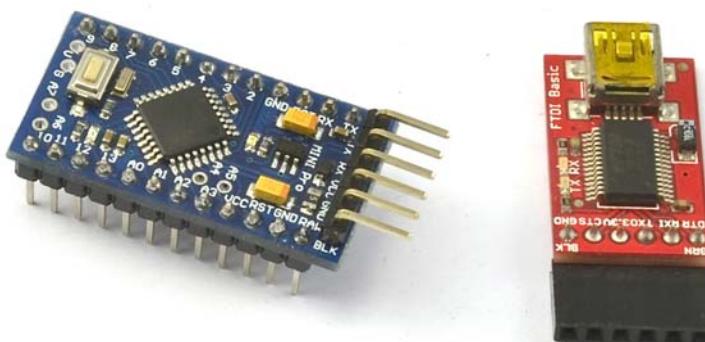
Problem

Chcesz korzystać z płytki Arduino, ale wolalbyś zastosować płytę o mniejszych wymiarach.

Rozwiązanie

Zastosuj jedną z mniejszych płyt kompatybilnych z Arduino.

Na rysunku 14.17 przedstawiono płytę Arduino Pro Mini. Moduły Arduino tego typu można zamontować na płytce prototypowej wraz z innymi komponentami niezbędnymi do wykonania projektu. Płytkę Pro Mini jest również dostępna w wersji pracującej pod napięciem 3,3 V — pozwala ona na uniknięcie potrzeby konwersji napięcia podczas komunikacji z Raspberry Pi.



Rysunek 14.17. Arduino Pro Mini i interfejs służący do jej programowania

Omówienie

Małe moduły Arduino, takie jak płytka Pro Mini widoczna na rysunku 14.17, wymagają dodatkowych interfejsów służących do programowania ich za pośrednictwem portu USB. Dzięki takim interfejsom można je programować przy użyciu Raspberry Pi lub (gdyby to było problematyczne) za pomocą innego komputera.

Poza oficjalnymi płytami Arduino istnieje wiele tańszych klonów tej platformy.

Zobacz również

Możesz również rozważyć zastosowanie innych płyt:

- Teensy (<http://www.pjrc.com/teensy/>),
- Arduino Micro (<http://arduino.cc/en/Main/ArduinoBoardMicro>),
- Arduino Nano (<http://arduino.cc/en/Main/ArduinoBoardNano>).

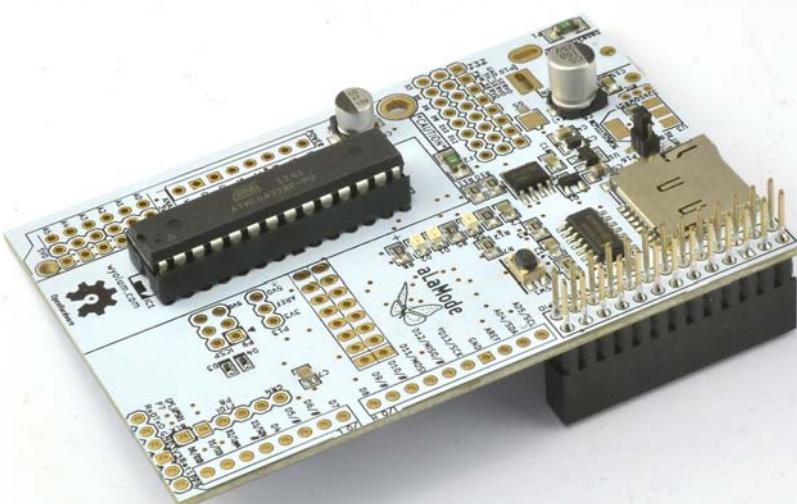
14.13. Podłączanie płytki aLaMode do Raspberry Pi

Problem

Chcesz sterować pracą zewnętrznej elektroniki za pomocą płytki aLaMode podłączonej do Raspberry Pi.

Rozwiązanie

Płytkę aLaMode (zobacz sekcja „Moduły” w dodatku A) przedstawiono na rysunku 14.18. To wspaniały moduł, który jest tak naprawdę Arduino Uno wyposażonym w złącze GPIO. Płytkę ta idealnie pasuje do płytki Raspberry Pi — jest na nią nakładana. Pozwala na korzystanie z dodatkowych płyt, tzw. *shieldów*, przeznaczonych do współpracy z Arduino bez potrzeby stosowania dodatkowych przewodów połączeniowych.

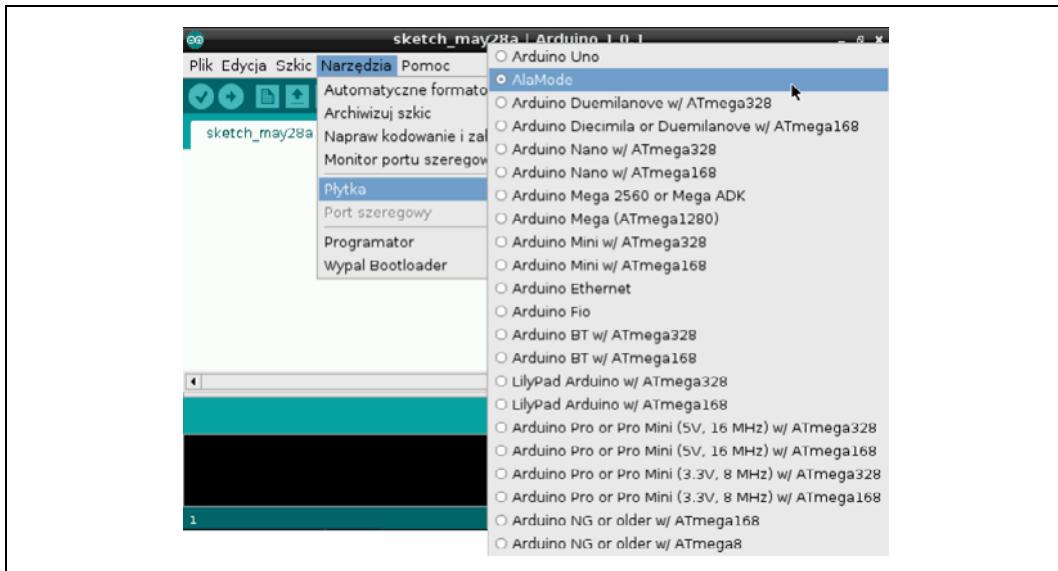


Rysunek 14.18. Płytki aLaMode

Płytkę pokazaną na rysunku 14.18 nie została wyposażona w goldpiny pozwalające na podłączanie *shieldów*.

Środowisko programistyczne, którego instalacja została opisana w recepturze 14.1, pozwala na korzystanie z płytki aLaMode podłączonej do Raspberry Pi. W menu *Narzędzia* wybierz podmenu *Płytki*, a następnie kliknij *AlaMode* (zobacz rysunek 14.19).

Zanim załadujesz program do modułu, zmień adres połączenia szeregowego na `/dev/ttyS0`. Musisz również wyłączyć konsolowe logowanie danych (zobacz receptura 8.7).



Rysunek 14.19. Wybór płytki aLaMode w środowisku programistycznym Arduino

Omówienie

Wszystkie receptury korzystające z aplikacji *Firmata* będą działać również z płytą aLaMode. Dotyczy to także techniki komunikacji szeregowej opisanej w recepturze 14.10. Jedyną modyfikacją, jakiej należy dokonać w programach Pythona, jest zmiana portu z USB na połaczenie szeregowe.

Płyta aLaMode komunikuje się tak jak Arduino — za pomocą pinów *Tx* i *Rx*, tak jak opisano w recepturze 14.5. Płyta ta ma jednak wbudowany układ dokonujący konwersji napięć sygnałów.

Korzystając z aplikacji *Firmata*, musisz zmienić port z */dev/ttyACM0* na */dev/ttyAMA0*:

```
board = pyfirmata.Arduino('/dev/ttyAMA0')
```

To samo dotyczy własnych aplikacji. Będziesz musiał zmodyfikować linię kodu otwierającą połaczenie szeregowe, aby wyglądała tak jak ta przykładowa linia:

```
ser = serial.Serial('/dev/ttyAMA0', 9600)
```

Aby sprawdzić działanie płytki aLaMode, załaduj do jej pamięci szkic *Standard Firmata*, a następnie uruchom aplikację Pythona o nazwie *ardu_flash_ser.py* (zobacz receptura 14.5).

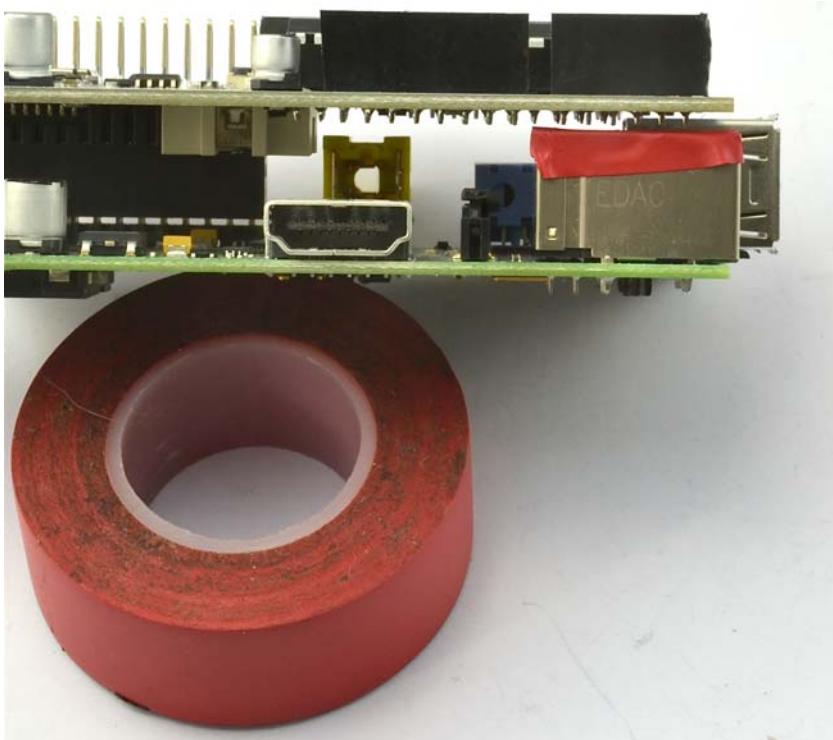
Na poniższej liście wymieniono najbardziej interesujące cechy płytki aLaMode.

- Płyta ta może być zasilana z linii o napięciu 5 V wchodzącej w skład złącza GPIO lub z oddzielnego zasilacza dostarczającego prąd o napięciu 5 V, podłączonego do gniazda mikro USB.
- Zegar czasu rzeczywistego jest podłączony bezpośrednio do płytki Raspberry Pi, a nie Arduino.
- Moduł ten jest wysoce kompatybilny z płytami rozszerzeń (*shieldami*) Arduino (zobacz receptura 14.14).

- Płytkę aLaMode może pracować w charakterze urządzenia podrzędnego magistrali I2C. Ponadto moduł ten jest jednocześnie połączony do Raspberry Pi za pomocą portu szeregowego (zobacz receptura 14.11).
- Płytkę aLaMode jest wyposażona w czytnik kart SD.
- Moduł ten posiada złącza przeznaczone do bezpośredniego podłączania serwomotorów (zobacz receptura 14.9).



Płytkę aLaMode ma jeden niewielki błąd konstrukcyjny. Przylutowane goldpiny będące wyprowadzeniami złączy analogowych ($A0 - A5$) mogą z łatwością zostać zwarte z metalową obudową gniazda RJ45 znajdującego się na płytce Raspberry Pi. Aby temu zapobiec, zanim założysz płytkę aLaMode na swoje Raspberry Pi, pokryj górną powierzchnię obudowy wspomnianego gniazda kilkoma warstwami taśmy izolacyjnej (zobacz rysunek 14.20).



Rysunek 14.20. Gniazdo RJ45 pokryte taśmą izolacyjną

Zobacz również

Oficjalną instrukcję użytkowania płytki aLaMode znajdziesz pod adresem https://docs.google.com/document/d/1HBvd3KNmcs63ZgO6t_u37B-qwV6P9o9FQe62lGkumM/edit.

14.14. Korzystanie z shieldów Arduino i płytki aLaMode podłączonej do Raspberry Pi

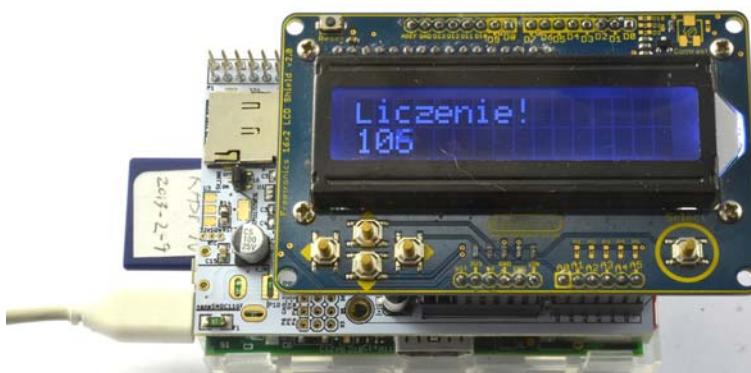
Problem

Chcesz korzystać z płytka rozszerzających funkcjonalność Arduino — tak zwanych *shieldów*.

Rozwiązanie

Zastosuj płytę aLaMode. Prawdopodobnie będziesz musiał dolutować do niej goldpiny pozwalające na komunikację z płytka rozszerzeń.

W niniejszej recepturze moduł wyświetlacza LCD podłączymy do płytki aLaMode (zobacz rysunek 14.21).



Rysunek 14.21. Płytki wyświetlacza LCD przeznaczonego do Arduino podłączona do modułu aLaMode

Aby skorzystać z niniejszej receptury, potrzebujesz:

- płytki aLaMode (zobacz sekcja „Moduły” w dodatku A),
- modułu (*shieldu*) wyświetlacza LCD (zobacz sekcja „Moduły” w dodatku A).

Wgraj do pamięci Arduino poniższy szkic. Możesz go również pobrać ze strony <http://www.helion.pl/ksiazki/raspre.htm> w postaci pliku o nazwie *AlaModeShield.ino*.

```
#include <LiquidCrystal.h>

// piny płytka wyświetlacza LCD firmy Freetronics

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

void setup()
{
    lcd.begin(16, 2);
    lcd.print("Liczenie!");
}
```

```
void loop()
{
    lcd.setCursor(0, 1);
    lcd.print(millis() / 1000);
    delay(1000);
}
```

Załaduj ten szkic do pamięci aLaMode. W górnej linii wyświetlacza powinien się wyświetlić napis *Liczenie!*, a w dolnej będą wyświetlane kolejne liczby (co sekundę).

Omówienie

W zaprezentowanym przykładzie korzystaliśmy z płytki wyświetlacza LCD firmy Freetronics. W swoich projektach możesz stosować wyświetlacze innych producentów. Pamiętaj o tym, że inne wyświetlacze mogą mieć inną konfigurację złączy, a więc być może będziesz musiał zmodyfikować poniższą linię szkicu:

```
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
```

Wymienione kolejne złącza modułu wyświetlacza komunikują się z następującymi stykami płytki aLaMode: *rs*, *uruchom*, *d4*, *d5*, *d6* i *d7*.

Zobacz również

Więcej informacji na temat biblioteki Arduino o nazwie `LiquidCrystal` znajdziesz na stronie <http://arduino.cc/en/Reference/LiquidCrystal>.

14.15. Stosowanie płytki Gertboard w roli interfejsu Arduino

Problem

Chcesz zaprogramować układ scalony ATmega zamontowany na płytce Gertboard, tak jakby była to płytnka Arduino.

Rozwiązanie

Skonfiguruj środowisko programistyczne Arduino, Raspberry Pi oraz płytę Gertboard, korzystając z poradnika znajdującego się na stronie <https://projects.drogon.net/raspberry-pi/gertboard/>.

Omówienie

Procesor ATmega znajdujący się na płytce Gertboard działa pod napięciem 3,3 V, a więc nie musisz stosować żadnych zabiegów mających na celu konwersję napięć sygnałów.

Zobacz również

Podstawowe wiadomości na temat płytki Gertboard znajdziesz w recepturze 8.17.

Poradnik użytkownika płytki Gertboard znajdziesz pod adresem http://www.automaticon.pl/novosci2013/doc/Gertboard_Assembled.pdf.

Komponenty i dystrybutorzy

Komponenty

Poniższe tabele pomogą Ci odnaleźć i zakupić podzespoły użyte w omówionych projektach. Podałem numery katalogowe produktów dostępnych u różnych dystrybutorów.

Jest wielu dostawców komponentów elektronicznych dostarczających je na potrzeby zarówno profesjonalistów, jak i amatorów. Część firm zajmujących się sprzedażą podzespołów została wymieniona w tabeli A.1 i tabeli A.2.

Tabela A.1. Dystrybutorzy z USA

Dystrybutor	Strona internetowa	Uwagi
Adafruit	http://www.adafruit.com/	dobre źródło modułów
Digikey	http://www.digikey.com/	szeroki wachlarz oferowanych produktów
MakerShed	http://www.makershed.com/	dobre źródło modułów, zestawów do samodzielnego montażu, a także narzędzi
MCM Electronics	http://www.mcmelectronics.com/	szeroki wachlarz oferowanych produktów
Mouser	http://pl.mouser.com/	szeroki wachlarz oferowanych produktów
RadioShack	http://www.radioshack.com/	sieć sklepów
SeeedStudio	http://www.seeedstudio.com/	tanie i interesujące moduły
SparkFun	https://www.sparkfun.com/	dobre źródło modułów

Tabela A.2. Inni dystrybutorzy

Dystrybutor	Strona internetowa	Uwagi
CPC	http://cpc.farnell.com/	dystrybutor z Wielkiej Brytanii, szeroki wachlarz oferowanych produktów
Ciseco	http://shop.ciseco.co.uk	dystrybutor płytEK takich jak Pi-Lite, Humble Pi itp.
Farnell	http://pl.farnell.com/	firma międzynarodowa
Maplin	http://www.maplin.co.uk/	sieć sklepów w Wielkiej Brytanii
Proto-PIC	http://proto-pic.co.uk/	dystrybutor z Wielkiej Brytanii, sprzedaje moduły dostarczane przez firmy SparkFun i Adafruit
SK Pang	http://skpang.co.uk/	dystrybutor z Wielkiej Brytanii

Wielu amerykańskich dystrybutorów ma oddziały w innych krajach lub oferuje wysyłkę sprzedawanych towarów za granicę.

Innym dobrym źródłem podzespołów są serwisy aukcyjne, np. eBay i Allegro.

Znalezienie potrzebnego komponentu może być trudne i zająć wiele czasu. Pomocnym narzędziem może się okazać internetowa wyszukiwarka podzespołów — <http://octopart.com/>.

Sprzęt przydatny podczas wykonywania prototypów

W wielu opisanych projektach połączenia są wykonywane za pomocą różnych kabli. Szczególnie przydadzą Ci się przewody połączeniowe z jednej strony zakończone wtykiem męskim, a z drugiej żeńskim (w celu połączenia styków gniazda GPIO z płytą prototypową), a także przewody połączeniowe obustronnie zakończone wtykami męskimi (do wykonywania połączeń na płycie prototypowej). Przewody połączeniowe obustronnie zakończone wtykami żeńskimi przydadzą Ci się czasem do bezpośredniego podłączania modułów do styków gniazda GPIO. Bardzo rzadko będziesz potrzebował przewodów dłuższych niż 75 mm. W tabeli A.3 wymieniono komponenty wraz z ich numerami katalogowymi.

Tabela A.3. Sprzęt przydatny podczas wykonywania prototypów

Opis	Dystrybutor, numer katalogowy
Przewód połączeniowy obustronnie zakończony wtykami męskimi	SparkFun: PRT-08431, Adafruit: 759
Przewód połączeniowy zakończony z jednej strony wtykiem męskim, a z drugiej żeńskim	SparkFun: PRT-09140, Adafruit: 825
Przewód połączeniowy obustronnie zakończony wtykami żeńskimi	SparkFun: PRT-08430, Adafruit: 794
Płytki prototypowa (83,5×54,5×8,5 mm)	SparkFun: PRT-09567, Adafruit: 794
Pi Cobbler	Adafruit: 1105

Rezystory i kondensatory

W tabeli A.4 wymieniono rezystory i kondensatory użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.4. Rezystory i kondensatory

Podzespół	Dystrybutor, numer katalogowy
Rezystor 270 Ω , 0,25 W	Mouse: 293-270-RC
Rezystor 470 Ω , 0,25 W	Mouse: 293-470-RC
Rezystor 1 k Ω , 0,25 W	Mouse: 293-1k-RC
Rezystor 3,3 k Ω , 0,25 W	Mouse: 293-3.3k-RC
Rezystor 4,7 k Ω , 0,25 W	Mouse: 293-4.7k-RC
Potencjometr dostrojczy 10 k Ω	Adafruit: 356, SparkFun: COM-09806, Mouse: 652-3362F-1-103LF
Fotorezystor	Adafruit: 161, SparkFun: SEN-09088
Kondensator 220 nF	MCM: 31-0610, Mouse: 80-C322C224M5U5HA

Tranzystory i diody

W tabeli A.5 wymieniono tranzystory i diody użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.5. Tranzystory i diody

Podzespoł	Dystrybutor, numer katalogowy
Tranzystor MOSFET z kanałem typu n FQP30N06	SparkFun: COM-10213, Adafruit: 355
Bipolarny tranzystor NPN 2N3904	SparkFun: COM-00521, Adafruit: 756
Dioda 1N4001	SparkFun: COM-08589, Adafruit: 755

Układy scalone

W tabeli A.6 wymieniono układy scalone użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.6. Układy scalone

Podzespoł	Dystrybutor, numer katalogowy
Regulator napięcia 7805	SparkFun: COM-00107
Sterownik silnika L293D	SparkFun: COM-00315, Adafruit: 807
Scalony układ Darlingtona ULN2803	SparkFun: COM-00312, Adafruit: 970
Czujnik temperatury DS18B20	SparkFun: SEN-00245, Adafruit: 374
Ośmiokanałowy przetwornik A/C MCP3008	Adafruit: 856
Czujnik temperatury TMP36	SparkFun: SEN-10988, Adafruit: 165

Optoelektronika

W tabeli A.7 wymieniono komponenty optoelektroniczne użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.7. Optoelektronika

Podzespoł	Dystrybutor, numer katalogowy
Czerwona dioda LED o średnicy 5 mm	SparkFun: COM-09590, Adafruit: 299
Dioda LED RGB o wspólnej katodzie	SparkFun: COM-11120
Czujnik podczerwieni TSOP38238	SparkFun: SEN-10266, Adafruit: 157

Moduły

W tabeli A.8 wymieniono moduły użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.8. Moduły

Podzespoł	Dystrybutor, numer katalogowy
Moduł kamery przeznaczony do pracy z Raspberry Pi	Adafruit: 1367, MCM: 28-17733, CPC: SC13023
Arduino Uno	SparkFun: DEV-11021, Adafruit: 50, CPC: A000066
Czterodrożny układ dokonujący konwersji napięcia sygnałów	SparkFun: BOB-11978, Adafruit: 757
Ośmiodrożny układ dokonujący konwersji napięcia sygnałów	Adafruit: 395
Przewtornica/ładowarka baterii LiPo	SparkFun: PRT-11231
PowerSwitch Tail	Adafruit: 268
Szesnastokanałowy sterownik servomotorów	Adafruit: 815
Podwójny sterownik silnika 1A	SparkFun: ROB-09457
Płytki RaspiRobot	SparkFun: KIT-11561, http://www.raspirobot.com/
Płytki cyfrowego interfejsu PiFace	MCM: 83-14472, CPC: SC12827
Humble Pi	MCM: 87-14637, CPC: SC12871
Pi Plate	Adafruit: 801
Gertboard	MCM: 83-14460, CPC: SC12828
Płytki testowa wyposażona w zaciski sprężynowe	MCM: 83-14876, CPC: SC12885
Czujnik ruchu działający w paśmie podczerwieni	Adafruit: 189
Moduł Venus GPS	SparkFun: GPS-11058
Czujnik metanu	SparkFun: SEN-09404
Płytki testowa czujnika gazu	SparkFun: BOB-08891
Trójosiowy przyspieszeniomierz ADXL335	Adafruit: 163
Moduł czterech siedmiosegmentowych wyświetlacz LED wyposażony w interfejs I2C	Adafruit: 878
Kolorowa matryca diod LED wyposażona w interfejs I2C	Adafruit: 902
Płytki interfejsu Pi-Lite	Ciseco, CPC: SC13018
Płytki interfejsu aLaMode	Makershed: MKWY1, Seeedstudio: ARD10251P
Płytki wyświetlacza LCD firmy Freetronics przeznaczona do współpracy z Arduino	http://www.freetronics.com/
Moduł zegara czasu rzeczywistego	Adafruit: 264
Moduł wyświetlacza LCD 16×2 kompatybilnego z HD44780	SparkFun: LCD-00255, Adafruit: 181

Pozostałe

W tabeli A.9 wymieniono pozostałe komponenty użyte w obwodach omówionych w tej książce; podano też ich numery katalogowe.

Tabela A.9. Pozostałe komponenty

Podzespoł	Dystrybutor, numer katalogowy
Bateria LiPo 1200 mAh	Adafruit: 258
Przekaźnik 5 V	SparkFun: COM-00100
Woltomierz przeznaczony do montażu w obudowie (zakres pomiarowy do 5 V)	SparkFun: TOL-10285
Serwomotor	SparkFun: ROB-09065, Adafruit: 1449
Zasilacz 5 V, 1A	Adafruit: 276
Silnik małej mocy zasilany prądem stałym o napięciu 6 V	Adafruit: 711
Goldpiny 2,54 mm	SparkFun: PRT-00116, Adafruit: 392
Unipolarny silnik krokowy, 5 V, 5 złączy	Adafruit: 858
Bipolarny silnik krokowy, 12 V, 4 złącza	Adafruit: 324
Podwozie robota wyposażone w silniki przekładniowe	SparkFun: ROB-10825
Przełącznik chwilowy	SparkFun: COM-00097, Adafruit: 504
Miniaturowy przełącznik ślimakowy	SparkFun: COM-09609, Adafruit: 805
Koder obrotowy	Adafruit: 377
Blok klawiszy 4×3	SparkFun: COM-08653
Brzęczek piezoelektryczny	SparkFun: COM-07950, Adafruit: 160

Skorowidz

A

adapter Pi-View, 27
administrator, 73
adres IP, 43, 45
akumulator LiPo, 179
aliasy poleceń, 93
archiwa, 88, 90
Arduino, 323–354
Arduino Pro Mini, 349
arkusze kalkulacyjne, 98
ASCII, 276
atrybuty pliku, 74
automatyczne uruchamianie programu, 81, 83

B

baterie, 178
biblioteka, 157
Adafruit, 232, 313
bottle, 161, 223, 226
PyFirmata, 328, 331–339
Pygame, 277
PySerial, 171, 316
random, 158
RPi.GPIO, 165, 169, 185
smtplib, 161
SoftwareSerial, 345
tkinter, 211
Tkinter, 307
Wiring Pi, 185
bit banging, 169
blok klawiszy, 268
błąd
key error, 144
unexpected indent, 115
brzęczyk, 202

C

Charlieplexing, 215
cyfrowe wejścia, 253
czarna lista, blacklist, 169
czas trwania impulsu, 201, 338
czujnik
cyfrowy, 302
metanu, 288, 290
rezystancyjny, 283, 295
ruchu, 161, 271
światła, 287
temperatury, 297, 302
TMP36, 297

D

dalmierz, 304
dane GPS, 275
data i godziny, 93, 148
definiowanie
funkcji, 132
klasy, 149
metody, 151
detektor ruchu, 271
DHCP, 42
diody, 357
LED, 197–200
podłączanie, 197
regulacja jasności, 200
wyświetlanie
komunikatów, 314
zmiana koloru, 213
RGB LED, 213
rozpraszające, 215
długość łańcucha, 123
dodawanie elementów do listy,
137

dostęp do

adresu URL, 226

elementów słownika, 144

elementów listy, 136

dostosowywanie funkcji

overscan, 31

drukowanie sieciowe, 59

dwupozycyjny przełącznik

dwustabilny, 258

suwakowy, 258

dystrybucje systemu, 19

dystrybutory, 355

działania arytmetyczne, 119

działanie

kodera kwadratowego, 267

modułu GPS, 273

plików aLaMode, 351

serwomotoru, 230

dziedziczenie, 152

dielnik napięcia, 293, 295

E

edycja pliku, 68, 69

edytor

AbiWord, 98

grafiki, 109

IDLE, 115

nano, 68, 116

tekstowy, 69

vim, 70

efekty przetaktowania, 33

ekran powitalny NOOBS, 21

emulator

Atari 2600, 104

plików PiFace, 182

Stella, 105

F

Finder, 53
 formatowanie dat, 148
 fotorezystor, 287, 296
 funkcja, 132
 add_event_detect, 220
 digital_read, 183
 digital_write, 183
 forward, 243
 get_encoder_turn, 267
 index, 226
 int, 122
 lower, 126
 open, 154
 overscan, 30
 pop, 138
 replace, 124, 125
 send_email, 161
 split, 138
 str, 122
 switch_status, 226
 sys.stdin.read, 275
 update_leds, 226
 upper, 126
 funkcje zaawansowane Pythona, 147

G

generowanie
 brzęczącego dźwięku, 202
 liczb, 158
 sygnału PWM, 338
 gniazdo RJ45, 352
 goldpiny, 173
 GPIO, 163
 GPS, 272, 273
 gra
 Asteroids, 104
 Minecraft, 105, 106
 Open Arena, 106, 107
 graficzny interfejs użytkownika, 35, 210, 211

H

harmonogram uruchamianych programów, 83
 hasło, 34
 historia wiersza poleceń, 85

I

identyfikacja styków, 174
 IDLE, 114, 183
 instalowanie
 biblioteki PySerial, 171
 biblioteki RPi.GPIO, 165
 gier, 101
 modułu kamery, 37
 oprogramowania, 78
 oprogramowania biurowego, 98
 przeglądarki, 99
 interfejs
 Arduino, 354
 graficzny, 51
 SPI, 169
 szeregowy, 341
 iteracja
 listy, 139
 słownika, 146

J

język
 Python, 113
 Scratch, 183

K

kabel konsolowy, 48
 kamera, 37
 kamera internetowa, 102
 kanał PWM, 215
 karta SD, 20, 29, 94
 kasowanie plików, 72
 katalog, 71
 domowy, 64
 init.d, 81
 klasy, 149
 klawiatura, 275
 kod programu, Patrz plik
 koder obrotowy, 265
 komenda sudo, 73
 komponenty, 355, 359
 komputery Macintosh, 52, 54, 57
 komunikacja z Arduino, 326, 341, 345
 komunikaty, 89
 komunikaty o błędach, 156
 koncentrator sieciowy, 42
 kondensatory, 356
 konfiguracja, 15

konfiguracja

magistrali I2C, 166
 systemu Raspbmc, 97

konsola

IDLE, 114
 Pythona, 116

konstrukcja try – except, 155

kontroler Wi-Fi, 251

konwersja

liczb, 122
 łańcuchów, 122

kopiowanie plików, 66

L

liczby losowe, 158

lista

modułów, 158
 urządzeń, 89

listy

dodawanie elementów, 137
 dostęp do elementu, 136
 iteracja, 139
 numerowanie elementów, 140
 przetwarzanie elementów, 142
 sortowanie, 141
 ustalanie długości, 136
 usuwanie elementów, 138
 wycinanie fragmentu, 141
 logowanie, 49, 50, 52

LXTerminal, 63

Ł

ładowanie

kondensatora, 286
 pulpitu, 35

łańcuchy

konwersja liczb, 122
 łączenie, 121
 tworzenie, 120
 ustalanie długości, 123
 ustalanie pozycji, 124
 wydobywanie fragmentu, 124
 zamiana znaków, 126
 zastępowanie fragmentu, 125

łączenie

z serwerem, 57, 58
 z siecią
 bezprzewodową, 46
 przewodową, 41, 44

M

magazyn NAS, 56
magistrala I2C, 166, 345
maksymalizacja wydajności, 32
matryca diod LED, 314, 315
Menedżer
 pakietów, 78
 plików, 61, 62
 zadań, 86
metody, 151
modele Raspberry Pi, 16
modulacja czasu trwania
 impulsu, 202, 338
moduł, 157, 358
 Gertboard, 185
 GPS, 272, 273
 kamery, 38
 konwertujący sygnał, 342
 Pi Cobbler, 174
 PiFace, 181
 PowerSwitch Tail II, 208
 RaspiRobot, 187, 248
 sterownika silnika, 237
 zegara czasu rzeczywistego, 278, 281
moduły
 I2C, 166, 167
 przetworników, 177
modyfikacje łańcuchów, 121
monitor, 27
 CCTV, 28
 portu szeregowego, 326
monitorowanie aktywności
 procesora, 86
mostek H, 239
multimedialne centrum
 rozrywki, 95, 147, 215
mysza, 277

N

nadajnik radiowy, 107
napięcie, 291
napięcie sygnałów, 175, 177
narzędzia
 I2C, 167
 i2c-tools, 168
narzędzie, *Patrz także program apt-get*, 78
 crontab, 84
 Git, 169
 gpsd, 274

IDLE, 114
ImageWriter, 25
Minicom, 172
raspi-config, 29–35, 39, 50
Samba, 57
SD Association, 22
WiFi Config, 46, 47
NAS, Network Attached Storage, 54, 56
nawiasy kwadratowe, 135
nazwy zmiennych, 118
NOOBS, 20
notacja [:], 124, 141
numer identyfikacyjny procesu, 87

O

obiekt object, 152
obniżanie napięcia, 175–177, 294
obsługa
 archiwów, 88, 90
 DHCP, 42
 konsoli szeregowej, 172
 modułu PiFace, 181
 sieci Web, 162
 SSH, 50
 urządzeń drukujących, 59
 wyjątków, 155
 wyjść analogowych, 337
obudowa, 17
obudowa typu DIL, 174
odczytywanie
 danych, 334, 336
 pliku, 154
odpowiedź skokowa, 285
odtwarzacz multimedialny, 111
ogładanie zawartości pliku, 70
okno
 LXTerminal, 63
 Python Shell, 115
opcja
 boot_behaviour, 35
 expand_rootfs, 29
 overclock, 32
 overscan, 30
opcje przetaktowywania, 32
operator
 =, 117
 dodawania, 119
 dzielenia, 119
 mnożenia, 119
 odejmowania, 119

operatorzy
 logiczne, 129
 porównujące, 128
oprogramowanie, 95
 biurowe, 98
 NOOBS, 20
optoelektronika, 357
optoizolator, 209
oscyloskop, 306

P

pakiet LibreOffice, 99
parametr, 133
partyjka główna, 29
petla
 for, 115, 130
 while, 131
Pi Cobbler, 174
Pi Store, 101
piny modułu wyświetlacza, 321
pliki
 atrybuty, 74
 edykcja, 68
 kasowanie, 72
 kopianie, 66
 modyfikacja atrybutów, 75
 przeglądanie, 64, 70
 przenoszenie, 61
 tworzenie, 71
 wyszukiwanie, 84
 zmienna nazwy, 67
 zmienna właściciela, 76

pliki

.wav, 109
NOOBS, 21

płytnka
 aLaMode, 350–353
 Arduino, 323–354
 Arduino Uno, 323
 Gertboard, 184, 186, 354
 Humble Pi, 188
 Pi Plate, 190–194
 PiFace, 181
 Pi-Lite, 316
 RaspiRobot, 186, 246, 251
płytki prototypowe, 173, 188, 190
pobieranie
 kodu źródłowego, 80
 plików, 79
poczta elektroniczna, 160

podłączanie
 Arduino, 325
 czujnika rezystancyjnego, 295
 diody LED, 197
 do internetu, 41
 do płytki prototypowej, 174
 modułu kamery, 38
 monitora, 27
 płytek Arduino, 349
 płytki aLaMode, 350
 płytki drukowanej, 194
 przełącznika chwilowego, 253
 silnika, 237
 telewizora, 28
 urządzeń, 26
 woltomierza, 220
podłączone urządzenia USB, 89
polecenia
 apt-gee, 78
 &, 92
 @route, 162
 >, 71, 89, 91
 append, 137
 apt-get, 79
 apt-get search, 78
 apt-get update, 73
 cat, 70, 71, 90
 cd, 65
 close, 153
 cp, 68
 crontab, 83
 date, 280
 df, 94
 enumerate, 140
 exception, 156
 find, 84, 92
 float, 123
 for, 139, 140
 git, 80
 GPIO.PUD_UP, 255
 grep, 85
 gunzip, 88
 halt, 36
 history, 85
 if, 127
 ifconfig, 44
 input, 119, 131
 insert, 137
 int, 122
 kill, 87
 killall, 88
 ls, 65
 man, 88

mkdir, 71
more, 70
mv, 68
open, 153
passwd, 34
pipe, 91
print, 115, 118
ps, 88
python, 117
range, 130
raspi-config, 35
raspistill, 40
raspivid, 40
readline, 154
rm, 72
scrot, 77
sleep, 270
sort, 141
tar, 88
time.sleep, 262
top, 87
wget, 79

połaczenie szeregowe, 49
położenie przełącznika, 259
pomiar
 jasności światła, 287, 288
 napięcia, 291, 293
 natężenia prądu, 18
 odległości, 304
 przyspieszenia, 299
 rezystancji, 284, 285
 temperatury, 297, 302
porównywanie wartości, 128
port szeregowy, 170, 172, 331, 333
potencjometr dinstrojczy, 283,
 319, 336
potok, 90
powtarzanie instrukcji, 130
prędkość obrotowa, 233
program, Patrz także narzędzie
 AbiWord, 60
 Fedora ARM Installer, 24
 Finder, 53
 GIMP, 109
 Gnumeric, 98
 Minicom, 172
 motion, 102, 104
 Putty, 49
 PyFirmata, 327
 scrot, 77
 vim, 70
 VLC media player, 110
 VNC, 51

XBMC, 95, 97
xgps, 275
programowanie Arduino, 324
protokół
 SMTP, 160
 SSH, 43, 50
przechwytywanie ruchów
 myszy, 277
przeglądanie plików, 64
przeglądarka internetowa
 Chromium, 99
 Iceweasel, 59, 100
przekaźnik, 206
przełącznik
 chwilowy, 253, 256, 334
 dwupozycyjny, 258
 SPDT, 260
 trójpozycyjny, 259
przenoszenie plików, 61
przerwania, 220
przerywanie działania pętli, 131
przetaktowywanie, 32
przetwornik A/C, 293, 295
przyspieszeniomierz, 299–302
pulpit zdalny, 55
Python, 113

R

radio internetowe, 110
redukacja stuków, 261
regulacja jasności, 200
rezystor podwyższający, 263
rezistory, 356
robot, 248, 249
rodzaje przełączników, 261
rozmiar partycji głównej, 29
ruch obrotowy, 265

S

scalanie łańcuchów, 121
schemat
 połączeń robota, 249
 złącza GPIO, 164
serializacja, 154
serwer, 43
 kamery internetowej, 102
 NAS, 54, 58
 sieci Web, 161
 SMTP, 161
 VNC, 43, 51

serwomotor, 227, 230
shieldy Arduino, 353
sieć
 bezprzewodowa, 46
 przewodowa, 41
 radiowa, 97
silnik krokowy
 bipolarny, 244
 unipolarny, 240
silniki, 227
 krokowe, 240, 244
 przekładniowe, 237
składnia [:], 141
skrypt uruchamiający serwer, 81
słownik, 135
 iteracja, 146
 tworzenie, 143
usuwanie elementów, 145
uzyskiwanie dostępu, 144
słowo kluczowe as, 157
sortowanie listy, 141
spełnianie warunków, 127
sterowanie
 Arduino, 328, 331
 diodą LED, 338
 diodą RGB LED, 214
 jasnością diody, 339
 kierunkiem obrotów, 235
 mocą diod, 211
 pracą serwomotoru, 227, 339
 pracą urządzenia, 204
 pracą wielu serwomotorów, 230
 prędkością obrotową, 233
 przepływem prądu, 204
 robotem, 251
 serwomotorem, 228
 silnikiem, 188
 o dużej mocy, 234
 o małej mocy, 235
 silnikiem krokowym, 241, 245, 246
sprzętem elektronicznym, 197
urządzeniami, 208
woltomierzem, 219
wyjściami Arduino, 330
 złączem GPIO, 223
sterownik Darlingtona ULN2803, 240
stężenie gazu, 288
stosowanie modułów, 157

struktura Tkinter, 210
styki złącza GPIO, 163
system
 Adafruit Occidentalis, 166
 CUPS, 59, 60
 operacyjny, 61
 plików FAT, 22
 Raspberry Pi, 26
 Raspbian, 166
szkice, 326

S

ścieżka, 153
ślad wciśnięcia przycisku, 263
ślady sygnałów, 306
środowisko
 IDLE, 114, 183
 programistyczne Arduino, 351
światło, 287

T

tablica mieszająca, 144
technika
 bit banging, 169
 Charlieplexing, 216, 217, 218
 odpowiedzi skokowej, 285
 zapytywania, 222
telewizor, 28
temperatura, 297, 302
Terminal, 63, 64
testowanie portu szeregowego, 172
tranzystor, 204, 357
tranzystor typu MOSFET, 205
tworzenie

 aliasów poleceń, 93
 egzemplarzy klasy, 150
 graficznego interfejsu użytkownika, 210
 katalogów, 71
 list, 135
 łańcuchów, 120
 multimedialnego centrum rozrywki, 95, 215
 plików, 71
 słownika, 143
 żądań, 159

U

udostępnianie
 ekranu, 54
 plików, 52
układ
 ATmega, 354
 DS1307, 279
 DS18B20, 302, 304
 HD44780, 318
 L293D, 236, 240, 245
 MCP3008, 291, 294, 296
 PWM, 211, 229, 337
 TMP36, 298, 299
 ULN2803, 241
układy scalone, 357
ukrywanie danych wyjściowych, 91
uprawnienia administratora, 73
uruchamianie
 programów, 81, 83, 117
 programów w tle, 92
 serwera, 51
 sesji Terminala, 63
urządzenia
 o dużej mocy, 204, 206
 zasilane prądem przemiennym, 208
 zewnętrzne, 26
usuwanie
 elementów z listy, 138
 elementów ze słownika, 145
 oprogramowania, 79

V

VNC, Virtual Network Connection, 51

W

wartości logiczne, 118
warunek, 127
wątek Iterator, 335
wczytywanie danych, 119
wejścia Arduino
 analogowe, 336
 cyfrowe, 334
wieloznacznik, 66
wiersz poleceń
 argumenty Pythona, 159
 historia, 85
 pobieranie plików, 79

włączanie
elektroniki, 210, 331
urządzeń, 206
woltomierz analogowy, 218, 291
wprowadzanie danych, 275
wybór
dystrybucji systemu, 19
modelu, 15
zasilacza, 18
wydajność, 32, 86
wyjątki, 155
wyjścia analogowe, 337
wykonywanie prototypów, 356
wykorzystanie
karty SD, 29
narzędzi I2C, 167
schowka, 70
wykres, 310
wykrywanie
metanu, 288
przechylenia, 301
ruchu, 271
wyłączanie
procesu, 87
Raspberry Pi, 36
wysyłanie wiadomości, 160
wyszukiwanie plików, 84
wyświetlacz, 311
ciekłokryształniczy, 319
LCD, 318, 353
LED, 312
wskazówkowy, 218

wyświetlanie
danych, 118
komunikatów, 314, 317, 321
listy urządzeń, 89
mierzonych wielkości, 307

Z

zaciski sprężynowe, 194
zamiana znaków, 126
zapis
do pliku, 153
karty SD, 20–25
komunikatów, 89
plików, 65
wyników pomiarów, 308
zapytywanie, 222
zarządzanie plikami, 61
zasilacz, 18
zasilanie, 178–180
zdalne sterowanie, 50
zegar czasu rzeczywistego, 278, 281
zintegrowane środowisko
programistyczne, 325
złącza
Arduino, 329
bloku klawiszy, 269
złącze
composite video, 28
DVI, 27
GPIO, 163

HDMI, 27
Rx Arduino, 341
VGA, 27
złożenie, 142
zmiana
hasła, 34
kierunku obrotów, 235, 239
nazwy, 45
nazwy pliku, 67
rozmiaru obrazu, 30
rozmiaru partycji, 29
właściciela pliku, 76
zmienna, 117
self, 150
this, 150
znak
#, 166
/, 153
:, 162
apostrofu, 120
zachęty, 115
znaki
ASCII, 276
specjalne, 121
zrzut ekranu, 77
zwracanie wartości, 149

ż

żądania HTTP, 159

O autorze

Dr Simon Monk (Preston, Wielka Brytania) jest inżynierem cybernetykiem i informatykiem. Posiada doktorat z zakresu inżynierii oprogramowania. Po kilku latach pracy na uczelni został współzałożycielem firmy Momote Ltd., zajmującej się produkcją aplikacji mobilnych. Obecnie Simon Monk zajmuje się głównie pisaniem książek dotyczących otwartego sprzętu, takiego jak Raspberry Pi i Arduino. Jest również autorem wielu książek poruszających zagadnienia ogólnie związane z elektroniką. Więcej informacji na temat jego książek znajdziesz na stronie <http://www.simonmonk.org/>. Autora możesz również śledzić na Twitterze: @simonmonk2.

Kolofon

Ptak przedstawiony na okładce książki *Raspberry Pi. Receptury* to krogulec zwyczajny (łac. *accipiter nisus*), który jest również nazywany jastrzębiem wróblarzem lub po prostu krogulcem. Ten mały drapieżnik występuje w Europie, Afryce i Azji. Dorosłe samce mają niebieskawo-czarną górną część ciała i pomarańczowy brzuch. Samice mogą być o 25% większe od samców.

Krogulec specjalizuje się w polowaniu na ptaki leśne, ale można go również spotkać w innych środowiskach. Może także polować na ptaki ogrodowe występujące w miastach. Samce wolą polować na mniejsze ptaki — sikorki, ziarnojady i wróble. Samice częściej polują na drozdy i szpaki. Są one w stanie zabić ptaki o masie przekraczającej pół kilograma.

Krogulce rozmnażają się w gniazdach mających średnicę nawet 60 cm, zbudowanych z gałęzi. Ptaki te składają cztery lub pięć jaj koloru jasnoniebieskiego (z brązowymi kropkami). Życie młodych ptaków zależy od tego, czy samica utrzyma dużą masę ciała. Samiec dostarcza jedzenie swojej partnerce. Młode wykluwają się po 33 dniach i wylatują z gniazda po kolejnych 24 – 28 dniach.

Prawdopodobieństwo, że młody krogulec przeżyje pierwszy rok swojego życia, wynosi 34%. Szansa przeżycia kolejnego roku podwaja się i prawdopodobieństwo przeżycia roku przez osobniki dorosłe wynosi 69%. Ptaki te żyją zwykle 4 lata. Młode samce umierają częściej od młodych samic. Pomimo spadku populacji po drugiej wojnie światowej krogulec jest najczęściej spotykanym drapieżnikiem w Europie. Stosowanie insektycydów z grupy węglowodorów chlorowanych w celu zaprawienia nasion przed wysiewem szkodziło krogulcom, a nawet je zabijało. Ptaki w wyniku kontaktu z tymi substancjami składały jajka o zbyt delikatnej skorupie, która pękała podczas wysiadywania jaj przez samice. Gdy zakazano stosowania tych substancji chemicznych, populacja krogulca odbudowała się, a gatunek ten jest obecnie uważany przez organizację BirdLife International za niezagrożony.

Obraz umieszczony na okładce pochodzi z monografii *Cassell's Natural History*.

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄZKI
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**