O'REILLY®

Wydanie III

Raspberry Pi
Receptury

Helion

Simon Monk

Tytuł oryginału: Raspberry Pi Cookbook: Software and Hardware Problems and Solutions, 3rd Edition

Tłumaczenie: Anna Mizerska z wykorzystaniem fragmentów Raspberry Pi. Receptury w przekładzie Konrada Matuka

ISBN: 978-83-283-6647-3

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

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/raspr3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/raspr3.zip

Printed in Poland.

Spis treści

ROZDZIAŁ 16.

Internet rzeczy

16.0. Wprowadzenie

Internet rzeczy (ang. *Internet of Things* — IoT) to bardzo szybko rozwijająca się sieć urządzeń (rzeczy) podłączonych do Internetu. Internet rzeczy to nie tylko komputery, na których korzystamy z przeglądarek internetowych, ale także wszystkie sprzęty, które mogą łączyć się z globalną siecią, włączając w to różnego rodzaju inteligentne systemy zarządzania domem, takie jak inteligentne sprzęty domowe i oświetlenie, systemy ochrony, a nawet karmniki dla zwierząt sterowane przez Internet lub inne, mniej praktyczne, lecz zabawne projekty.

Po lekturze tego rozdziału będziesz wiedzieć, w jaki sposób Twoje Raspberry Pi może stać się częścią Internetu rzeczy.

16.1. Sterowanie złączem GPIO za pomocą sieci Web

Problem

Chcesz sterować pracą wyjść złącza GPIO za pośrednictwem sieci Web.

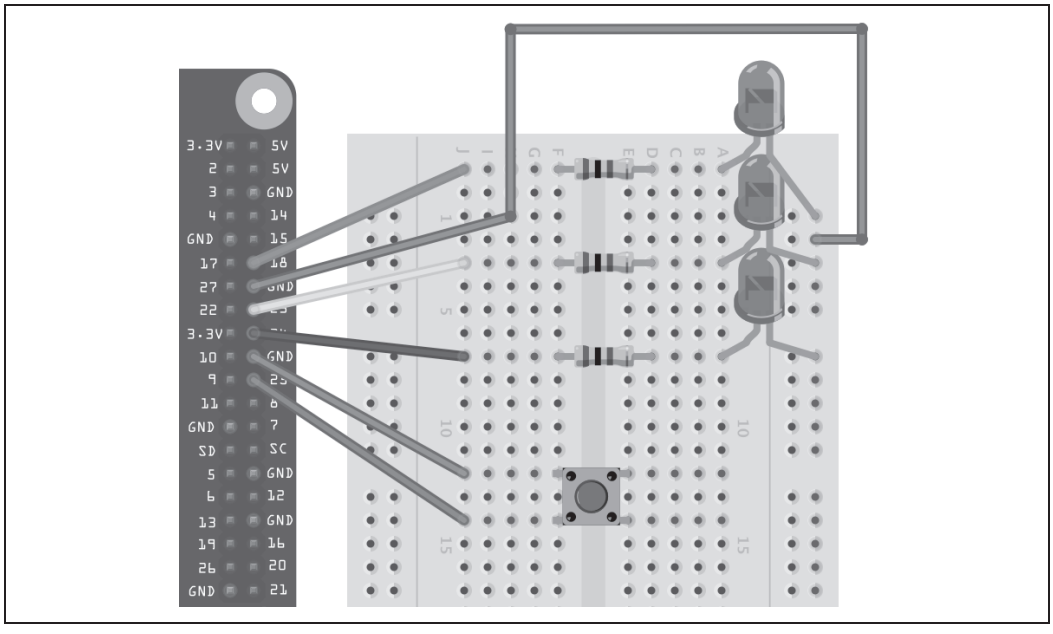
Rozwiązanie

Skorzystaj z biblioteki `bottle` (zobacz receptura 7.17). 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:

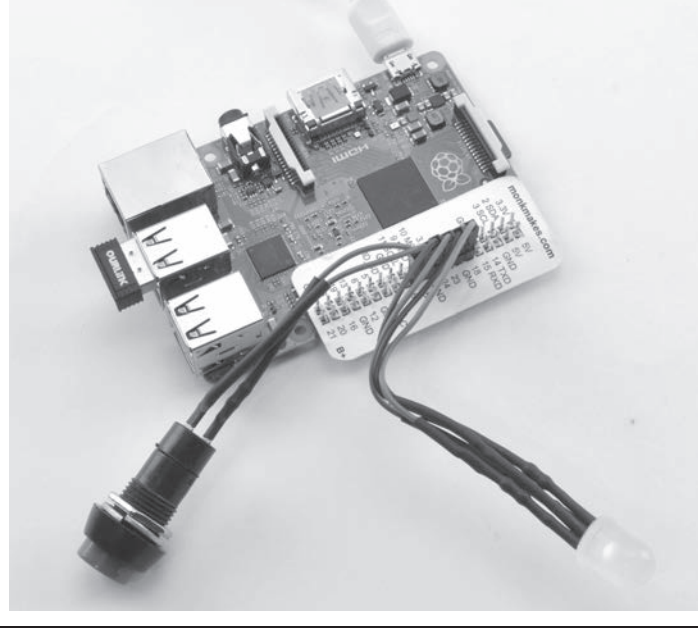
- płytki prototypowej i przewodów połączeniowych (zobacz sekcja „Sprzęt przydatny podczas wykonywania prototypów” w dodatku A),
- trzech rezystorów 1 kΩ (zobacz sekcja „Rezystory i kondensatory” w dodatku A),
- trzech diod LED (zobacz sekcja „Optoelektronika” w dodatku A),
- trzech przełączników chwilowych (zobacz sekcja „Pozostałe” w dodatku A).

Na rysunku 16.1 przedstawiono ułożenie tych elementów na płytce prototypowej.



Rysunek 16.1. Schemat obwodu sterowanego za pośrednictwem strony Web

Zamiast płytki prototypowej możesz użyć przełącznika i diody LED umieszczonych na przewodach i podpinanych bezpośrednio do pinów GPIO (zobacz receptury 9.10 i 9.11), tak jak zostało to pokazane na rysunku 16.2.



Rysunek 16.2. Przełącznik i dioda LED podłączone bezpośrednio do pinów GPIO Raspberry Pi

Proces instalacji biblioteki `bottle` opisano w recepturze 7.17.

Umieść poniższy kod (plik *r16_web_control.py*) w oknie środowiska programistycznego IDLE.

```
from bottle import route, run
from gpiozero import LED, Button

leds = [LED(18), LED(23), LED(24)]
switch = Button(25)

def switch_status():
    if switch.is_pressed:
        return 'Wcisniety
    else:
        return 'Niewcisniety'

def html_for_led(led_number):
    i = str(led_number)
    result = " <input type='button' onClick='changed(" + i + ")' value='LED " + i + "'/>"
    return result

@route('/')
@route('/<led_number>')
def index(led_number="n"):
    if led_number != "n":
        leds[int(led_number)].toggle()
    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='0.0.0.0', port=80)
```

Podobnie jak wszystkie programy pokazane w tej książce, powyższy kod znajdziesz w materiałach do pobrania (zobacz recepturę 3.22).

Program należy uruchomić jako użytkownik posiadający uprawnienia administratora, używając wersji Python 2:

```
$ sudo python r16_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://0.0.0.0:80/
Hit Ctrl-C to quit.
```

Jeżeli otrzymujesz wiadomości o błędach, upewnij się, że uruchamiasz program przy użyciu komendy `sudo` oraz `python`, a *nie* `python3`.

Na dowolnym komputerze, lub na samym Raspberry Pi, pracującym w Twojej sieci uruchom przeglądarkę. W polu adresu wpisz adres IP swojego Raspberry Pi (zobacz recepturę 2.2). Powinien się wyświetlić interfejs widoczny na rysunku 16.3.

Rysunek 16.3. 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 *Wciskasz* (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 wysyłane 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).

Jeżeli żądanie sieciowe odwołuje się do strony głównej (*http://192.168.1.8/*), to zmiennej `led_number` 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_number`.

Przypisanie parametrowi `led_number` 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>"
response += "function changed(led)"
response += "{"
response += "  window.location.href='/' + led"
response += "}"
response += "</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):

   l = str(led)
   result = " <input type='button' onClick='changed(" + l + ")'
   value='LED " + l + "'/>"
   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.

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.

Zobacz również

Dodatkowe informacje na temat biblioteki `bottle` znajdziesz na stronie *https://www.bottlepy.org/docs/dev/*.

16.2. Wyświetlanie odczytów czujników na stronie internetowej

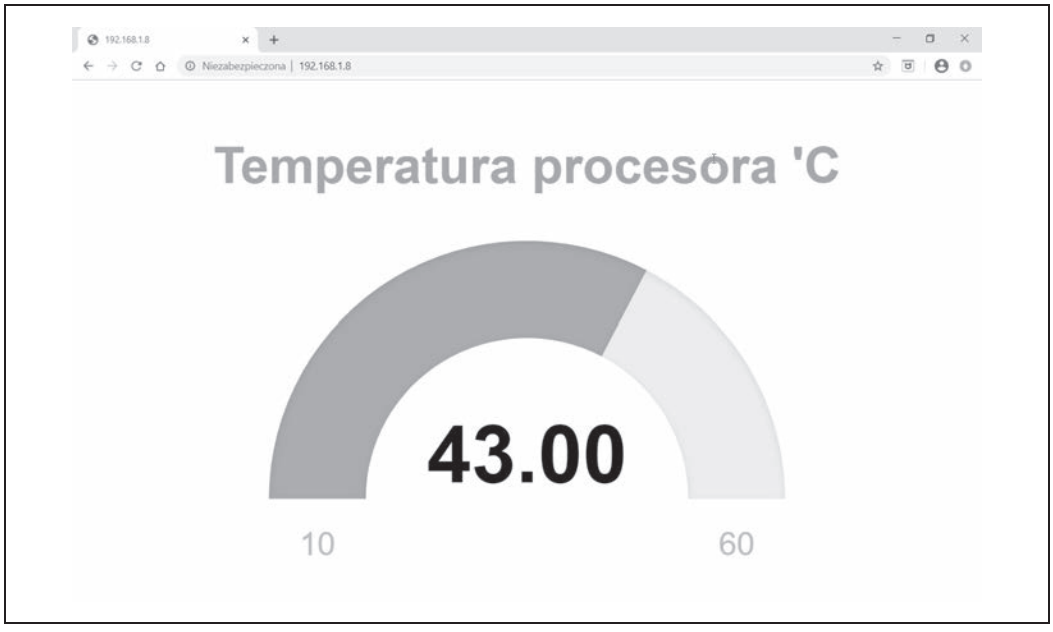
Problem

Chcesz wyświetlać odczyty czujnika podłączonego do Twojego Raspberry Pi na automatycznie aktualizowanej stronie internetowej.

Rozwiązanie

Użyj serwera sieciowego `bottle` i języka JavaScript, aby aktualizować wyświetlane informacje.

Przykładowy projekt, pokazany na rysunku 16.4, polega na wyświetlaniu temperatury procesora Raspberry Pi, mierzonej za pomocą wbudowanego czujnika.



Rysunek 16.4. Wyświetlanie temperatury procesora Raspberry Pi

Aby dowiedzieć się, jak zainstalować bibliotekę `bottle`, zobacz recepturę 7.17.

W folderze *r16_web_sensor*, znajdującym się w materiałach do pobrania, znajdziesz 4 pliki dla tego przykładu.

web_sensor.py
: Zawiera napisany w Pythonie kod dla serwera `bottle`.

main.html
: Zawiera kod HTML, który będzie wyświetlany w Twojej przeglądarce.

justgage.1.0.1.min.js
: Zewnętrzna biblioteka JavaScript, która wyświetla grafikę termometru.

raphael.2.1.0.min.js
: Biblioteka używana przez bibliotekę `justgage`.

W celu uruchomienia programu zmień folder na *r16_web_sensor* i użyj poniższej komendy:

```
$ sudo python web_sensor.py
```

Podobnie jak pozostałe programy omówione w tej książce, powyższy program znajdziesz w materiałach do pobrania (receptura 3.22).

Następnie otwórz przeglądarkę, na tym samym Raspberry Pi lub na innym komputerze podłączonym do tej samej sieci, i wpisz adres IP Raspberry Pi w pasku adresu. Jeżeli wykonałeś wszystko poprawnie, pojawi się strona podobna do tej pokazanej na rysunku 16.4.

Omówienie

Główny program, *web_sensor.py*, jest tak naprawdę bardzo krótki.

```
import os, time
from bottle import route, run, template

def cpu_temp():
    dev = os.popen('/opt/vc/bin/vcgencmd measure_temp')
    cpu_temp = dev.read()[5:-3]
    return cpu_temp

@route('/temp')
def temp():
    return cpu_temp()

@route('/')
def index():
    return template('main.html')

@route('/raphael')
def index():
    return template('raphael.2.1.0.min.js')

@route('/justgage')
def index():
    return template('justgage.1.0.1.min.js')

run(host='0.0.0.0', port=80)
```

Funkcja `cpu_temp` odczytuje temperaturę procesora Raspberry Pi, tak jak zostało to opisane w recepturze 13.10.

W dalszej części programu zostały użyte 4 polecenia `@route`, które łączą podany w nawiasie adres URL z następującą po nich procedurą. Pierwsza ścieżka (`/temp`) zwraca temperaturę procesora Raspberry Pi w stopniach Celsjusza. Wartość ta jest zapisana jako zmienna tekstowa. Główna ścieżka (`/`) zwraca szablon głównej strony HTML, *main.html*. Dwie pozostałe ścieżki umożliwiają dostęp do kopii bibliotek JavaScript: `justgage` i `raphael`.

Plik *main.html* zawiera głównie kod JavaScript do renderowania interfejsu użytkownika.

```
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"
type="text/javascript" charset="utf-8"></script>
<script src="raphael"></script>
<script src="justgage"></script>

<script>
function callback(tempStr, status){
if (status == "success") {
    temp = parseFloat(tempStr).toFixed(2);
    g.refresh(temp);
    setTimeout(getReading, 1000);
}
else {
    alert("Wystapil problem");
    }
}
```

```
function getReading(){
    $.get('/temp', callback);
}
</script>
</head>

<body>
<div id="gauge" class="200x160px"></div>

<script>
var g = new JustGage({
    id: "gauge",
    value: 0,
    min: 10,
    max: 60,
    title: "Temperatura procesora 'C"
});
getReading();
</script>

</body>
</html>
```

Biblioteka `jquery` jest importowana bezpośrednio ze strony internetowej *https://developers.google.com/speed/libraries/#jquery*. Natomiast biblioteki `justgage` i `raphael` importowane są z lokalnych plików.

Pobieranie odczytu z Raspberry Pi do przeglądarki jest procesem dwustopniowym. Najpierw wywoływana jest funkcja `getReading`, która wysyła żądanie przez `/temp` do programu *web_sensor.py*. Gdy nadejdzie odpowiedź, wywołana zostanie funkcja `callback` odpowiedzialna za aktualizację grafiki termometru wyświetlanej na stronie. Po sekundzie funkcja `getReading` zostanie wywołana ponownie, zgodnie z drugim argumentem funkcji `setTimeout`.

Zobacz również

Jeżeli chciałbyś wyświetlać odczyty czujnika w aplikacji napisanej w Pythonie zamiast na stronie internetowej, zobacz recepturę 13.22.

Biblioteka `justgage` ma wiele użytecznych opcji wyświetlania odczytów z czujników. Aby uzyskać dodatkowe informacje na jej temat, odwiedź stronę *https://justgage.com/*.

16.3. Rozpoczęcie pracy z Node-RED

Problem

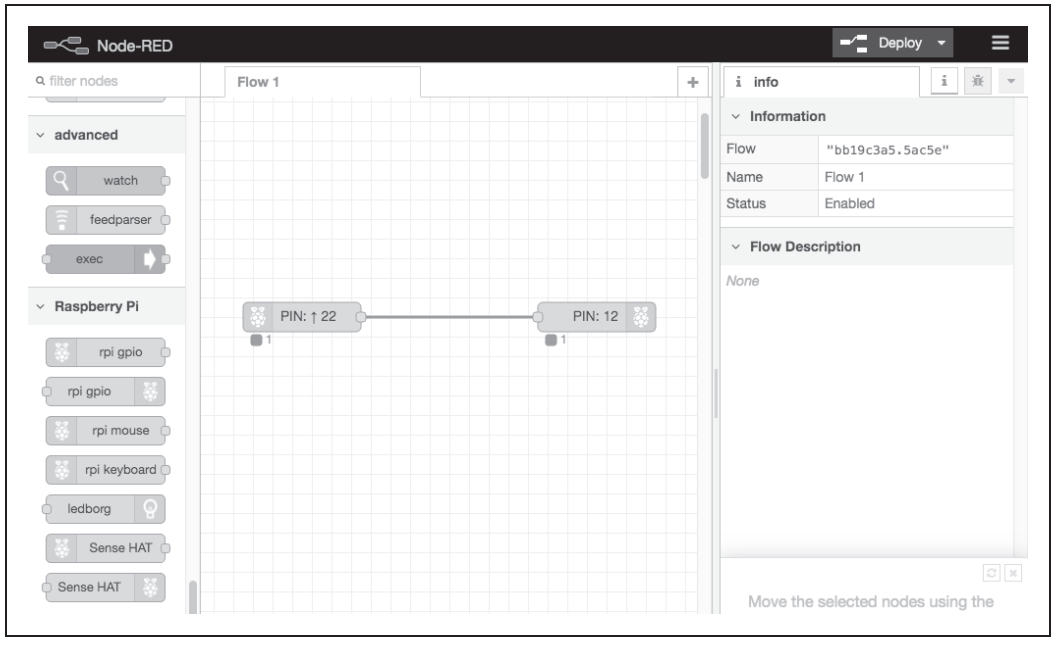
Chcesz stworzyć prosty proces IoT polegający na wysyłaniu tweeta po wciśnięciu przycisku na Raspberry Pi.

Rozwiązanie

Skorzystaj z programu Node-RED, który jest dostępny w Raspbianie. Użyj poniższego polecenia, aby uruchomić serwer Node-RED.

```
$ node-red-pi --max-old-space-size=256
```

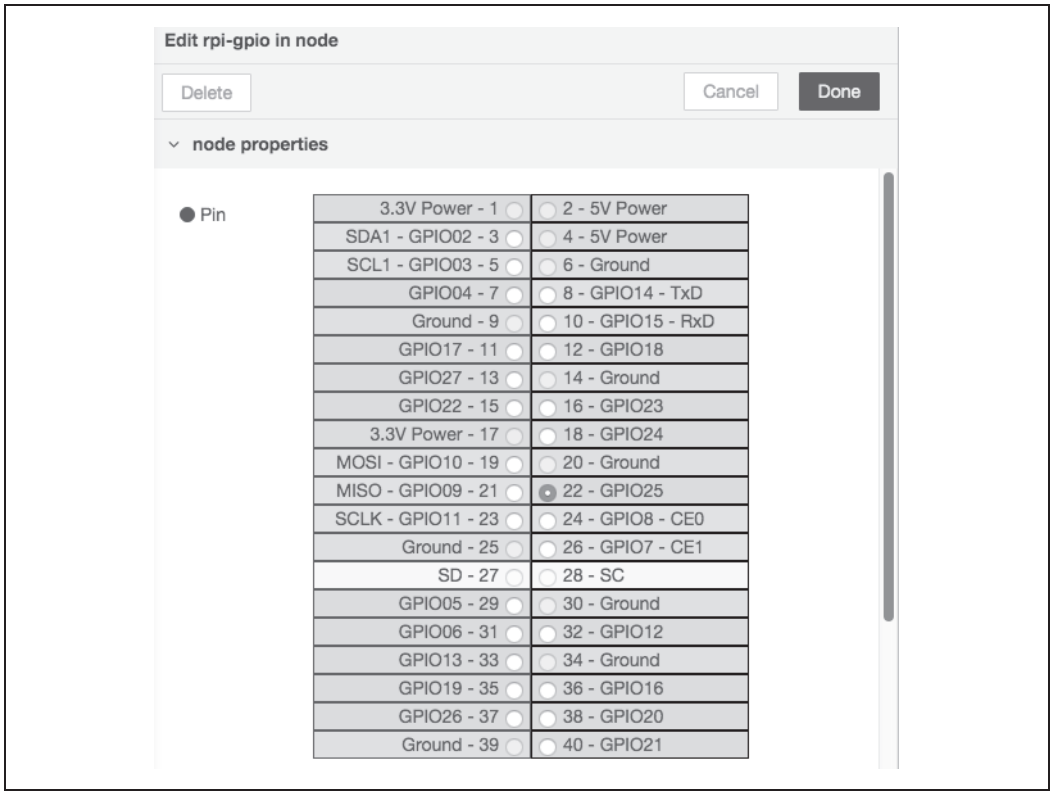
Następnie podłącz się do serwera, używając przeglądarki. Jeżeli łączysz się przez Raspberry Pi, użyj adresu //127.0.0.1:1880/, a jeśli z innego komputera podłączonego do tej samej sieci, zamień 127.0.0.1 na adres IP swojego Raspberry Pi. Po podłączeniu się do serwera Node-RED zobaczysz ekran podobny do tego z rysunku 16.5.



Rysunek 16.5. Interfejs sieciowy Node-RED

W programie Node-RED „rysujesz" swój program (nazywany **schematem**), zamiast pisać kod. Program zbudowany jest z węzłów (ang. *nodes*) i połączeń między nimi. Na rysunku 16.5 widnieje najprostszy przykładowy schemat złożony z dwóch połączonych ze sobą pinów Raspberry Pi. Jeden pin ustawiony jest jako wejście i może być podłączony do przełącznika (wybierzmy styk GPIO 25 jako przykład), a drugi podłączony jest do diody LED (przyjmijmy, że jest to styk GPIO 18). W tym projekcie możesz użyć diody i przycisku z przewodami, podłączanych bezpośrednio do pinów GPIO Raspberry Pi, tak jak zostało to pokazane w recepturze 16.1.

Węzeł po lewej stronie, oznaczony jako PIN 22, działa w charakterze wejścia i jest podłączony do przełącznika podpiętego do złącza GPIO 25. Natomiast węzeł PIN 12 działa jak wyjście i jest podłączony do diody LED podpiętej do złącza GPIO 18. Zamiast nazw złączy GPIO Node-RED używa pozycji pinów. By stworzyć taki sam schemat w swoim edytorze, na liście węzłów znajdującej się po lewej stronie wyszukaj sekcję zatytułowaną Raspberry Pi. Przeciągnij węzeł *rpi gpio* z logo Raspberry Pi po lewej stronie na obszar roboczy programu. Węzeł ten będzie działał jako wejście. Kliknij go dwukrotnie, a gdy otworzy się okno pokazane na rysunku 16.6, wybierz *22 - GPIO25* i wciśnij *Done* (gotowe).



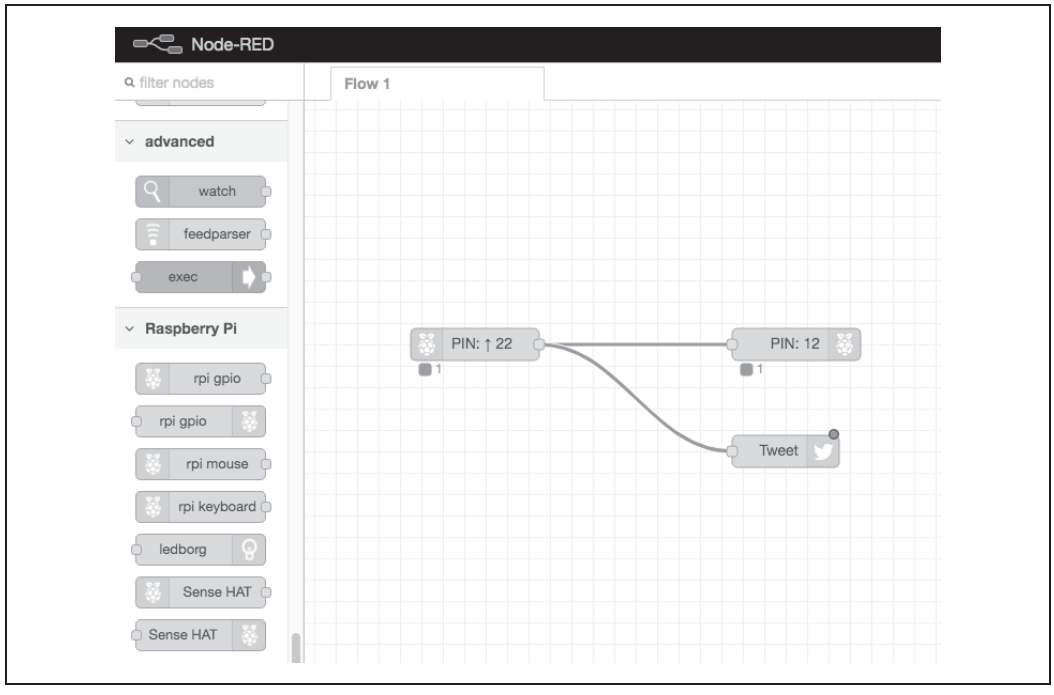
Rysunek 16.6. Wybór GPIO25 w Node-RED

Następnie wybierz kolejny węzeł typu *rpi gpio* (tym razem z logo Raspberry Pi po prawej stronie) i przeciągnij go na obszar roboczy. Podobnie jak w przypadku poprzedniego węzła, kliknij go dwukrotnie, a gdy otworzy się okno pokazane na rysunku 16.6, wybierz *12 - GPIO18* i naciśnij *Done*.

Połącz te dwa węzły ze sobą, przeciągając okrągły łącznik węzła wejściowego do węzła po prawej stronie, aby uzyskać efekt pokazany na rysunku 16.5.

Jeśli podłączyłeś już przełącznik i diodę LED do Raspberry Pi, możesz uruchomić stworzony właśnie schemat, klikając przycisk *Deploy* (uruchom). Po uruchomieniu programu dioda LED powinna się zaświecić, a po naciśnięciu przełącznika powinna zgasnąć. Jak pewnie zauważyłeś, logika tego procesu jest odwrócona, ale zmienimy to dopiero w kolejnym rozdziale, w którym Node-RED zostanie omówiony dużo szerzej.

Na tym etapie stworzony proces nie ma wiele wspólnego z Internetem rzeczy. W celu rozbudowania naszego projektu użyjemy innych typów węzłów Node-RED. Przeglądając listę dostępnych węzłów, pośród wielu innych zauważysz węzeł *Tweet*. Możesz go podłączyć jako drugie wyjście do węzła PIN 22. Twój projekt powinien wyglądać tak jak na rysunku 16.7. Aby skonfigurować węzeł *Tweet*, kliknij go dwukrotnie i podaj swoje dane logowania do Twittera. Teraz, gdy naciśniesz przełącznik, zostanie wysłany tweet.



Rysunek 16.7. Wysyłanie tweetów z Node-RED

Omówienie

Node-RED jest systemem o bardzo dużych możliwościach. Aby oswoić się ze wszystkimi funkcjonalnościami i specyfiką pracy z Node-RED, będziesz potrzebował trochę czasu. Poza tworzeniem projektów z węzłami połączonymi ze sobą bezpośrednio, Node-RED umożliwia tworzenie połączeń warunkowych (takich jak wyrażenie `if`) i funkcji, które przekształcają informację przekazywaną między węzłami.

W tym przykładzie wykorzystaliśmy możliwości Node-RED w minimalnym stopniu. Jeżeli chcesz dowiedzieć się więcej na temat tego systemu, sugeruję zapoznać się z dokumentacją wskazaną w punkcie „Zobacz również". Zachęcam również do przeczytania rozdziału 17., w którym Node-RED zostanie wykorzystany w szerszym zakresie.

Jeżeli spodobał Ci się Node-RED i chcesz, aby uruchamiał się automatycznie po włączeniu Raspberry Pi, wywołaj poniższe komendy:

```
$ sudo systemctl enable nodered.service
$ sudo systemctl start nodered.service
```

Zobacz również

Pełna dokumentacja dotycząca użycia NODE-Red z Raspberry Pi dostępna jest na stronie: *https://oreil.ly/NpEhD*.

Z kolei na stronach: *https://oreil.ly/uxGHZ* i *https://oreil.ly/ZSDNi* opublikowano filmy przedstawiające w bardzo przystępny sposób podstawy Node-RED.

16.4. Wysyłanie powiadomień z użyciem IFTTT

Problem

Chcesz stworzyć dla swojego Raspberry Pi elastyczny proces polegający na wysyłaniu powiadomień e-mailem, na Facebooka, Twittera czy Slacka.

Rozwiązanie

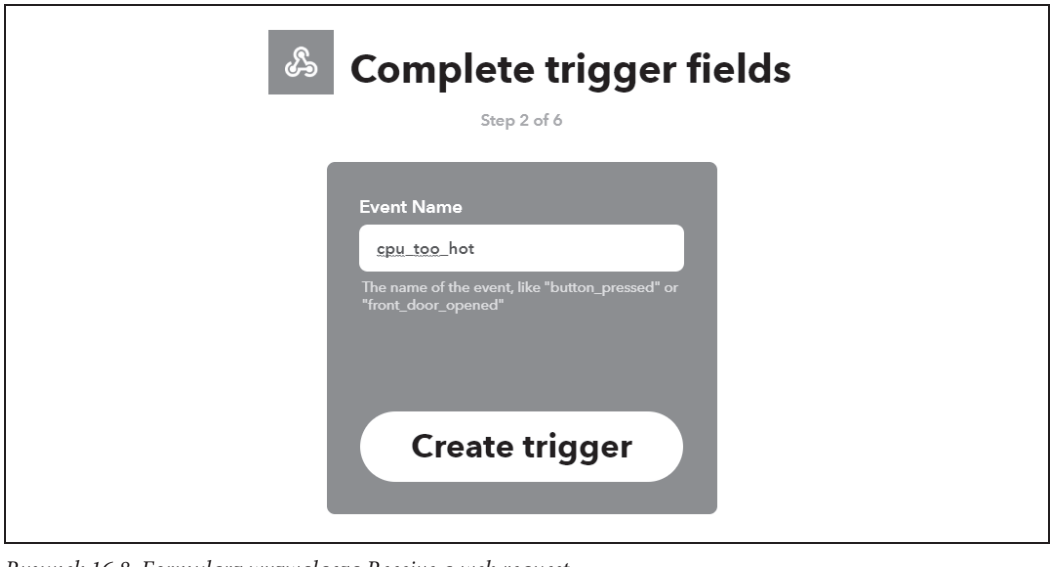
Spraw, by Twój Raspberry Pi wysyłał żądania do darmowej usługi sieciowej *If This Then That* — IFTTT (jeżeli „to", to wtedy „tamto") w celu tworzenia powiadomień, które można dowolnie konfigurować.

Projekt omówiony w tej recepturze polega na wysyłaniu powiadomień e-mailem, gdy temperatura procesora przekroczy wartość graniczną.

Przed rozpoczęciem pracy będziesz musiał stworzyć konto na IFTTT (*www.ifttt.com*).

Po założeniu konta stwórz nowy aplet IFTTT. **Aplet** jest swego rodzaju regułą, taką jak na przykład: *Gdy otrzymam żądanie z Raspberry Pi, wyślij e-mail*. Naciśnij awatar swojego konta i z rozwijalnego menu wybierz opcję *Create* (utwórz). Najpierw otworzy się ekran początkowy, z plusami, za pomocą których możesz konfigurować poszczególne etapy apletu. Należy rozpocząć od części *THIS*, klikając znak plusa, który znajduje się po słowie *If*.

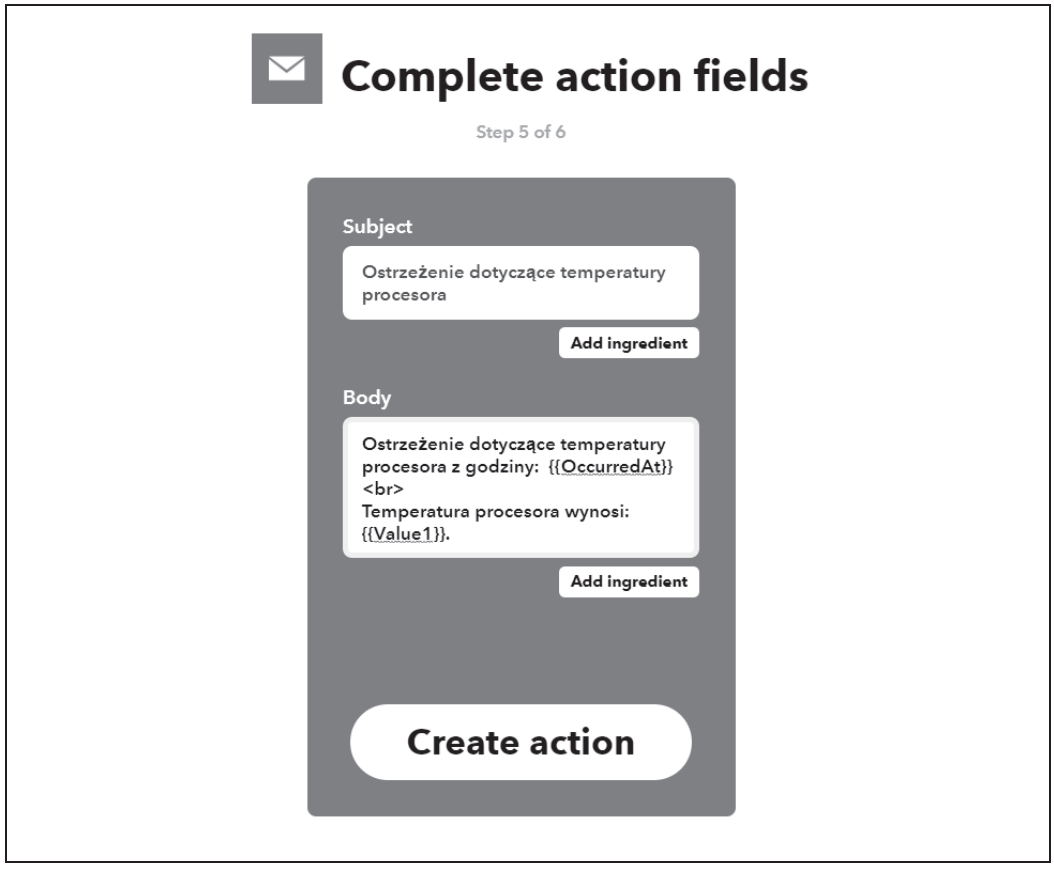
W naszym przypadku część *THIS* (wyzwalacz — ang. *trigger*) będzie tożsama z otrzymaniem żądania sieciowego z Twojego Raspberry Pi. Po kliknięciu ikony ze znakiem plus w polu wyszukiwania wpisz **`Webhooks`**, aby wyszukać odpowiedni kanał. Wybierz *Webhooks*, a na kolejnym ekranie kliknij *Connect* (połącz). W kolejnym kroku pojawi się opcja *Receive a web request* (otrzymaj żądanie sieciowe), którą należy kliknąć, aby pojawił się ekran pokazany na rysunku 16.8.



Rysunek 16.8. Formularz wyzwalacza Receive a web request

W polu *Event Name* (nazwa wydarzenia) wpisz **`cpu_too_hot`** i kliknij *Create trigger* (utwórz wyzwalacz).

Zostaniesz przeniesiony do ekranu początkowego tworzenia apletu. Tym razem kliknij ikonę plusa, która znajduje się po lewej stronie *That* (to). Otworzy się formularz, w którym należy wybrać typ akcji, jaka ma się wykonać. W polu wyszukiwania wpisz **`Email`**, aby wyszukać odpowiedni kanał. Spośród kanałów, które się wyświetlą, wybierz *Email*, a następnie kliknij *Connect*. Otworzy się nowe okno, w którym należy wpisać adres e-mail, jakiego ma używać aplet. Po wpisaniu swojego adresu mailowego naciśnij *Send PIN* (wyślij PIN). Na adres, który podałeś, zostanie wysłany czterocyfrowy PIN, który należy wpisać w polu PIN, a następnie nacisnąć *Connect*. Po wpisaniu poprawnego PIN-u pojawi się opcja *Send me an email* (wyślij do mnie e-mail), którą należy kliknąć, by pojawił się ekran pokazany na rysunku 16.9.



Rysunek 16.9. Wypełniony formularz akcji na platformie IFTTT

Wypełnij formularz danymi z rysunku 16.9. Zauważ, że wartości dynamiczne, *OccuredAt* i *Value1*, są umieszczone w podwójnym nawiasie klamrowym. Wartości te nazywane są składnikami (ang. *ingredients*) i w rzeczywistości są to zmienne, które będą przyjmować wartość wysłaną przez żądanie sieciowe, a następnie będą umieszczane w odpowiednich miejscach w temacie i treści e-maila.

Kliknij *Create action* (utwórz akcję), a następnie *Finish* (zakończ), aby zakończyć tworzenie apletu.

Ostatnią informacją, jakiej potrzebujemy, jest klucz do API (ang. *Application Programming Interface* — interfejs programowania aplikacji) dla kanału *Webhooks*. Klucz pomoże zapobiec otrzymywaniu e-maili dotyczących temperatury procesora Raspberry Pi od innych użytkowników.

Swój klucz możesz odnaleźć w dokumentacji usługi *Webhooks*. Z menu, które wyświetla się po naciśnięciu Twojego awatara, wybierz opcję *My services* (moje usługi), a następnie usługę *Webhooks*. Na kolejnym ekranie, w prawym górnym rogu pojawi się przycisk *Documentation*. Po jego naciśnięciu wyświetli się klucz do API (rysunek 16.10 — klucz został celowo zamazany). Swój klucz do API będziesz musiał wkleić do poniższego programu w linijce: `KEY = 'wklej_swoj_klucz_tutaj'`.

Rysunek 16.10. Klucz do API w dokumentacji Webhooks

Program wysyłający żądania sieciowe, nazywa się *r16_ifttt_cpu_temp.py*.

```
import time, os
import requests

MAX_TEMP = 37.0
MIN_T_BETWEEN_WARNINGS = 60 # Minuty

EVENT = 'cpu_too_hot'
BASE_URL = 'https://maker.ifttt.com/trigger/'
KEY = 'wklej_swoj_klucz_tutaj'

def send_notification(temp):
    data = {'value1' : temp}
    url = BASE_URL + EVENT + '/with/key/' + KEY
    response = requests.post(url, json=data)
    print(response.status_code)

def cpu_temp():
    dev = os.popen('/opt/vc/bin/vcgencmd measure_temp')
```

```
        cpu_temp = dev.read()[5:-3]
        return float(cpu_temp)

    while True:
        temp = cpu_temp()
        print("Temparatura procesora (C): " + str(temp))
        if temp > MAX_TEMP:
            print("PROCESOR MA ZA WYSOKA TEMPERATURE!")
            send_notification(temp)
            print("Brak powiadomien przez: " + str(MIN_T_BETWEEN_WARNINGS) + " minut")
            time.sleep(MIN_T_BETWEEN_WARNINGS * 60)
        time.sleep(1)
```

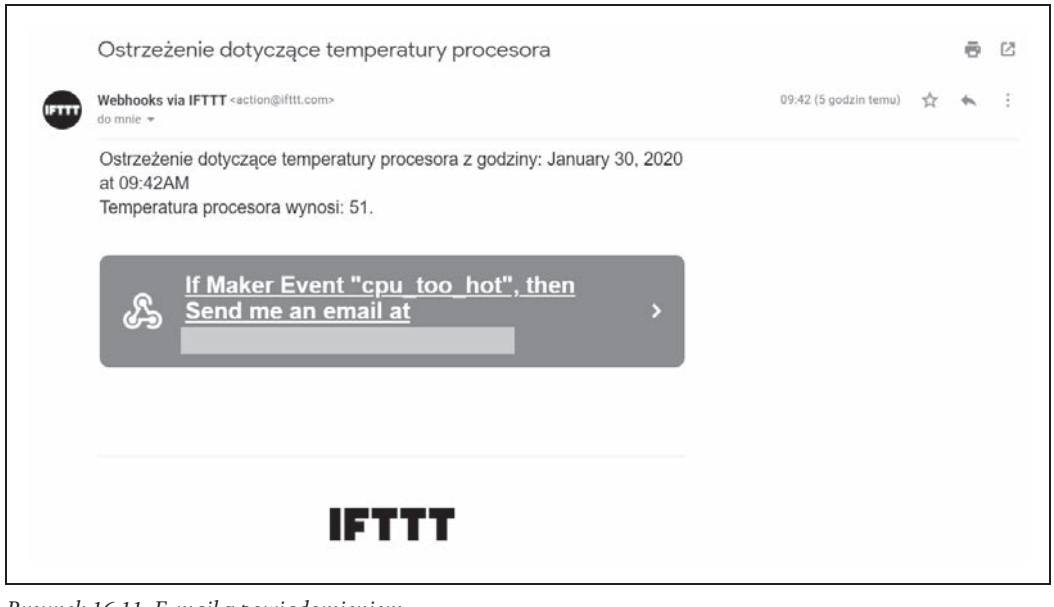
Podobnie jak w przypadku innych programów omawianych w tej książce, powyższy program znajdziesz w materiałach do pobrania, w folderze *python*, pod nazwą: *r16_ifttt_cpu_temp.py* (zobacz recepturę 3.22).

Do celów testowych ustawiłem stosunkowo niską wartość zmiennej `MAX_TEMP`. Jeżeli tam, gdzie aktualnie przebywasz, jest gorąco, możesz podnieść tę wartość do 60 lub 70.

Wklej swój klucz do linijki zaczynającej się od `KEY=` i uruchom program przy użyciu następującego polecenia:

```
$ python3 r16_ifttt_cpu_temp.py
```

Temperaturę procesora możesz podnieść przez odtworzenie filmu lub zawijając *tymczasowo* Raspberry Pi w folię bąbelkową. W momencie, gdy temperatura procesora przekroczy wartość `MAX_TEMP`, powinieneś otrzymać e-mail podobny do tego pokazanego na rysunku 16.11. Zaobserwuj, w jaki sposób dane zostały wstawione do treści wiadomości (adres e-mail został zamazany celowo).



Rysunek 16.11. E-mail z powiadomieniem

Omówienie

Funkcja `send_notification` odpowiada za większość akcji podjętych przez ten program. W pierwszej kolejności tworzy ona adres URL zawierający klucz API oraz pobiera wartość parametru `value1` (który przechowuje temperaturę), a następnie wykorzystuje bibliotekę Pythona — `requests`, by wysłać żądanie sieciowe do IFTTT.

Pętla główna cały czas sprawdza temperaturę procesora i porównuje ją z maksymalną dopuszczalną wartością, określoną jako `MAX_TEMP`. Jeśli temperatura przekroczyła wartość `MAX_TEMP`, zostaje wysłane żądanie sieciowe, a program czeka tyle minut, ile zostało podane w zmiennej `MIN_T_BETWEEN_WARNINGS`, zanim wyśle kolejne powiadomienie. Przerwa między wysyłaniem kolejnych powiadomień ma zapobiec „zasypaniu" Twojej skrzynki e-mailami.

Oczywiście zamiast korzystać z IFTTT, mógłbyś wysłać e-mail bezpośrednio z poziomu programu napisanego w Pythonie, używając receptury 7.16. Jednak IFTTT ma tę przewagę, że nie jesteś ograniczony tylko do wysyłania powiadomień e-mailem, ale możesz użyć wielu innych dostępnych kanałów, bez potrzeby pisania kodu.

Zobacz również

Aby dowiedzieć się, jak wysłać e-mail za pomocą Pythona, zobacz recepturę 7.16.

Kod programu, który mierzy temperaturę procesora, został opisany w recepturze 13.10.

16.5. Wysyłanie tweetów za pomocą ThingSpeak

Problem

Chcesz automatycznie wysyłać z Raspberry Pi tweety dotyczące na przykład temperatury procesora.

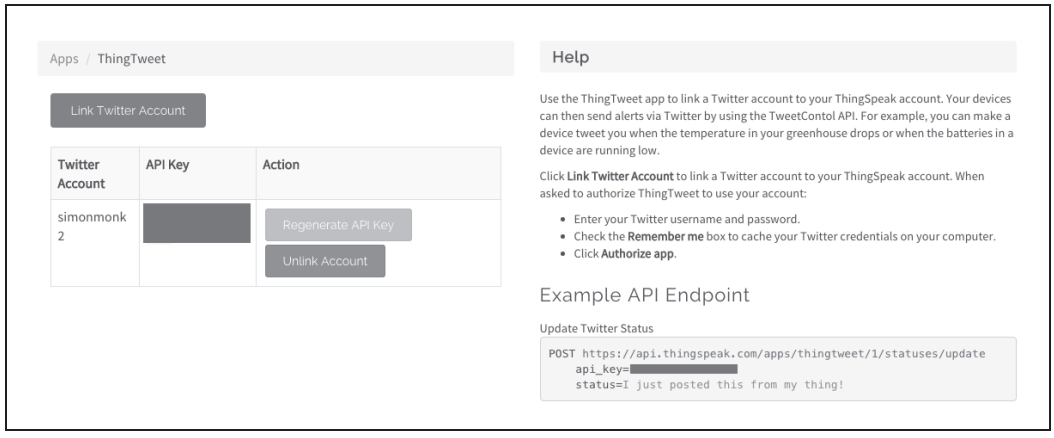
Rozwiązanie

Do tego celu mógłbyś użyć platformy IFTTT, omówionej w recepturze 16.4, zmieniając kanał *Email* na *Twitter*. Jednak w tej recepturze użyjemy platformy ThingSpeak.

Platforma ThingSpeak (*https://thingspeak.com*) jest podobna do IFTTT, z tą różnicą, że jest całkowicie nakierowana na projekty związane z IoT. Umożliwia tworzenie kanałów, które są w stanie przechowywać i pobierać dane, używając żądań sieciowych. Ponadto ThingSpeak ma wiele akcji, między innymi ThingTweet, która zawiera klasy opakowujące dla Twittera. Ten sposób jest łatwiejszy niż korzystanie z API, które wymaga zarejestrowania aplikacji na Twitterze.

Rozpocznij od otwarcia strony *https://thingspeak.com* i założenia konta. Warto zauważyć, że zakładając konto na ThingSpeak, automatycznie tworzy się konto na MATLAB, choć nie ma ku temu konkretnych powodów.

Następnie wybierz akcję *ThingTweet* z menu *Apps*. Zostaniesz poproszony o zalogowanie się do Twittera, po czym Twoja akcja zostanie aktywowana (rysunek 16.12).



Rysunek 16.12. Akcja ThingTweet

Program, który automatycznie wysyła żądania sieciowe skutkujące wysłaniem tweeta, znajduje się w materiałach do pobrania pod nazwą *r16_send_tweet*.

```
import time, os
import requests

MAX_TEMP = 37.0
MIN_T_BETWEEN_WARNINGS = 60 # Minuty

BASE_URL = 'https://api.thingspeak.com/apps/thingtweet/1/statuses/update/'
KEY = ' wklej_swoj_klucz_tutaj'

def send_notification(temp):
    status = 'Thingtweet: Raspberry Pi staje sie goracy. Temperatura procesora=' + str(temp)
    data = {'api_key' : KEY, 'status' : status}
    response = requests.post(BASE_URL, json=data)
    print(response.status_code)

def cpu_temp():
    dev = os.popen('/opt/vc/bin/vcgencmd measure_temp')
    cpu_temp = dev.read()[5:-3]
    return float(cpu_temp)

while True:
    temp = cpu_temp()
    print("Temperatura procesora (C): " + str(temp))
    if temp > MAX_TEMP:
        print("PROCESOR JEST ZBYT GORACY!")
        send_notification(temp)
        print("Brak powiadomien przez: " + str(MIN_T_BETWEEN_WARNINGS) + " minut")
        time.sleep(MIN_T_BETWEEN_WARNINGS * 60)
    time.sleep(1)
```

Aby dowiedzieć się, jak pobrać powyższy program, zobacz recepturę 3.22.

Tak samo jak w przypadku receptury 16.4, przed uruchomieniem programu musisz wkleić swój klucz API, który znajdziesz na ekranie pokazanym na rysunku 16.12. Uruchom program dokładnie w ten sam sposób, jak uruchomiłeś program z poprzedniej receptury 16.4.

Omówienie

Kod programu jest bardzo podobny do tego z receptury 16.4. Jednak w przypadku programu r15_send_tweet.py funkcja `send_notification` tworzy treść tweeta, która jest wysyłana za pomocą żądania sieciowego jako parametr `status`.

Zobacz również

Pełna dokumentacja dotycząca platformy ThingSpeak znajduje się na stronie *https://oreil.ly/pVyhG*.

Receptura 16.6 wykorzystuje popularny projekt CheerLights zaimplementowany w ThingSpeak. Z kolei na przykładzie receptury 16.7 nauczysz się, jak za pomocą ThingSpeak zbierać dane otrzymane z czujnika.

16.6. CheerLights

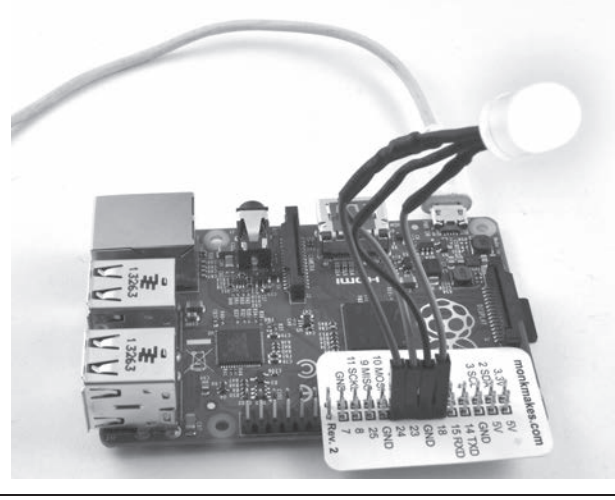
Problem

Chcesz podłączyć swoje Raspberry Pi do diody LED RGB i wziąć udział w popularnym projekcie CheerLights.

CheerLights jest usługą sieciową, która zapisuje nazwę koloru wysłanego jako tweet na @cheerlights. Wielu ludzi na całym świecie posiada własny projekt CheerLights, który korzysta z usługi sieciowej, pobierając ostatni opublikowany kolor, i zmienia światło podłączonej diody na tę właśnie barwę. W rezultacie, jeśli ktoś wyśle tweeta z nazwą koloru, dioda każdego użytkownika zmienia kolor.

Rozwiązanie

Podłącz diodę RGB bezpośrednio do złącza GPIO swojego Raspberry Pi (rysunek 16.13), a następnie uruchom program *r16_cheerlights.py*, którego kod znajdziesz pod rysunkiem.



Rysunek 16.13. „Wyświetlacz" CheerLights

```
from gpiozero import RGBLED
from colorzero import Color
import time, requests

led = RGBLED(18, 23, 24)
cheerlights_url = "http://api.thingspeak.com/channels/1417/field/2/last.txt"

while True:
    try:
        cheerlights = requests.get(cheerlights_url)
        c = cheerlights.content
        print(c)
        led.color = Color(c)
    except Exception as e:
        print(e)
    time.sleep(2)
```

Powyższy program znajdziesz także w materiałach do pobrania, zobacz recepturę 3.22.

Po uruchomieniu programu Twoja dioda zapali się na jakiś kolor. Prawdopodobnie po pewnym czasie zmieni barwę, gdy ktoś wyśle tweeta. Jeśli się tak nie stanie, spróbuj samemu wysłać tweeta „@cheerlights red", a wtedy Twoja dioda, i diody użytkowników z całego świata, zmieni kolor na czerwony. CheerLights akceptuje nazwy kolorów w języku angielskim. Kolory, jakich możesz użyć, to: *red*, *green*, *blue*, *cyan*, *white*, *oldlace, purple*, *magenta*, *yellow*, *orange*, *pink* (czerwony, niebieski, turkusowy, biały, beżowy, purpurowy, magenta, żółty, pomarańczowy, różowy).

Omówienie

Program *r16_cheerlights.py* wysyła żądanie sieciowe, które zwraca kolor zapisany w postaci sześciocyfrowej liczby złożonej z liczb zapisanych w systemie szesnastkowym. Informacja ta następnie jest wykorzystywana do zmiany koloru diody LED.

Dzięki blokowi `try/except` w przypadku chwilowego braku połączenia z siecią program nie przestanie się wykonywać.

Zobacz również

CheerLights wykorzystuje kanał ThingSpeak do przechowywania informacji na temat ostatnio wysłanego koloru. Receptura 16.7 wykorzystuje kanał do zapisywania odczytów z czujnika.

Jeśli nie posiadasz diody z przewodem, podłączanej bezpośrednio do złącza GPIO, możesz wykorzystać płytkę prototypową do stworzenia układu (zobacz recepturę 10.10) albo przerobić recepturę 14.6 tak, by kontrolować wszystkie diody na taśmie LED.

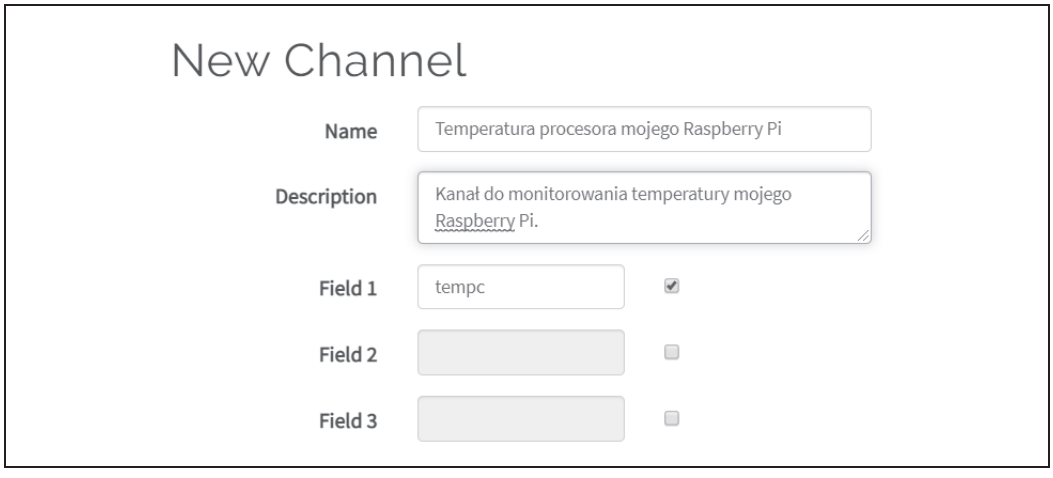
16.7. Wysyłanie odczytów czujnika do ThingSpeak

Problem

Chcesz zapisywać odczyty czujnika na ThingSpeak oraz wyświetlać wykresy danych w ujęciu czasowym.

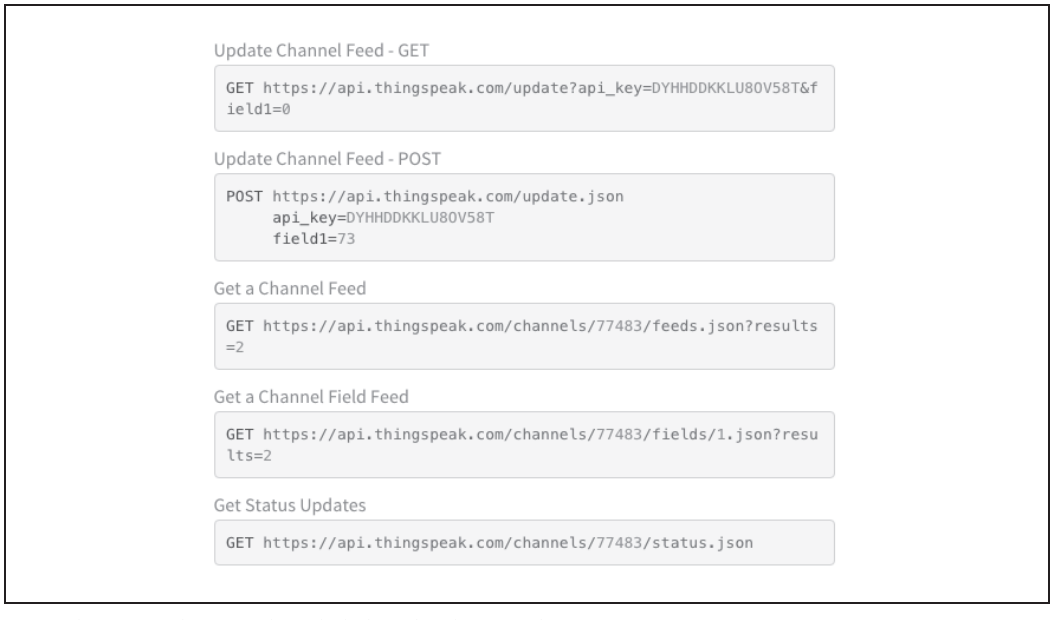
Rozwiązanie

Zaloguj się do ThingSpeak, a następnie z menu *Channels* (kanały), wybierz opcję *My Channels* (moje kanały). Stwórz nowy kanał, wypełniając górną część formularza pokazanego na rysunku 16.14. Pozostałe pola mogą pozostać puste.



Rysunek 16.14. Tworzenie kanału w ThingSpeak

Następnie kliknij przycisk *Save Channel* (zapisz kanał) znajdujący się na dole strony. Na kolejnym ekranie naciśnij zakładkę *API Keys* (klucze API), aby wyświetlić żądania sieciowe, których możesz użyć wraz z kluczem API dla właśnie stworzonego kanału (rysunek 16.15).



Rysunek 16.15. Pobieranie danych do kanału ThingSpeak

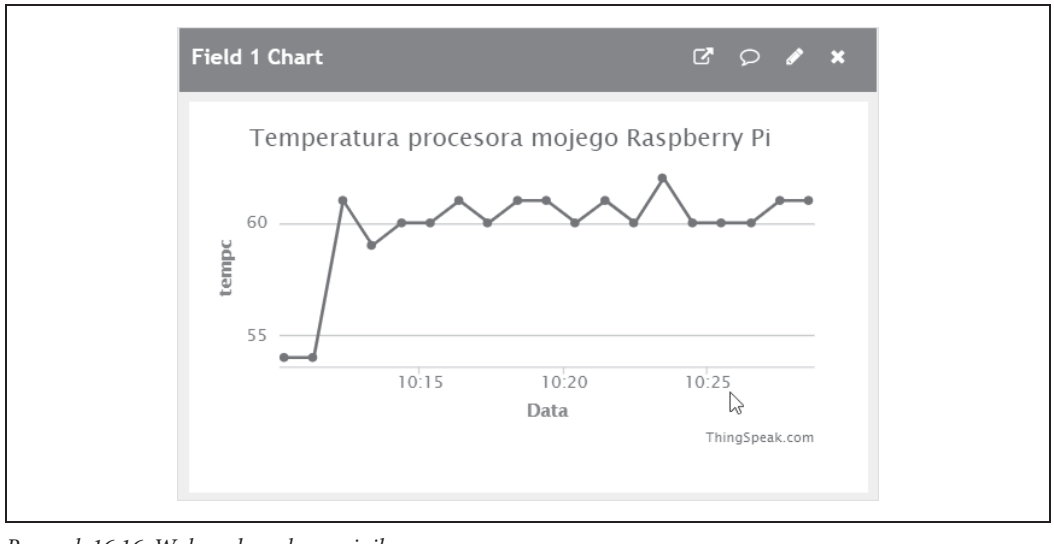
W poniższym programie do wartości zmiennej KEY wklej swój klucz API wykorzystywany do zapisu.

Program *r16_thingspeak_data.py* wysyła żądanie sieciowe w celu wysłania danych do kanału.

```
import time, os
import requests

PERIOD = 60 # sekundy
BASE_URL = 'https://api.thingspeak.com/update.json'
KEY = 'wklej_swoj_klucz_tutaj'

def send_data(temp):
    data = {'api_key' : KEY, 'field1' : temp}
    response = requests.post(BASE_URL, json=data)

def cpu_temp():
    dev = os.popen('/opt/vc/bin/vcgencmd measure_temp')
    cpu_temp = dev.read()[5:-3]
    return float(cpu_temp)

while True:
    temp = cpu_temp()
    print("Temperatura procesora (C): " + str(temp))
    send_data(temp)
    time.sleep(PERIOD)
```

Podobnie jak wszystkie programy umieszczone w tej książce, powyższy program znajdziesz w materiałach do pobrania (zobacz recepturę 3.22).

Uruchom program, a na stronie kanału ThingSpeak, w zakładce *Private View* (widok prywatny) zobaczysz wykres podobny do pokazanego na rysunku 16.16.



Rysunek 16.16. Wykres danych z czujnika

Wykres będzie aktualizowany co minutę, jak tylko zostanie otrzymany nowy odczyt.

Omówienie

Zmienna `PERIOD` określa odstęp czasowy w sekundach pomiędzy kolejnymi przekazami odczytów temperatury.

Funkcja `send_data` tworzy żądanie sieciowe, które przekazuje wartość temperatury do parametru `field1`.

Jeżeli przekazywane przez Ciebie dane mogą być interesujące dla innych, możesz zmienić swój kanał z prywatnego na publiczny.

Zobacz również

Receptura 13.23 pokazuje, jak eksportować dane z czujnika do arkusza kalkulacyjnego.

Objaśnienie programu odczytującego temperaturę procesora znajdziesz w recepturze 13.10.

16.8. Odpowiadanie na tweety przy użyciu Dweet i IFTTT

Problem

Chcesz, aby Twój Raspberry Pi wykonywał pewne akcje w odpowiedzi na określony hasztag (ang. *hashtag*) lub określone słowo w tweecie.

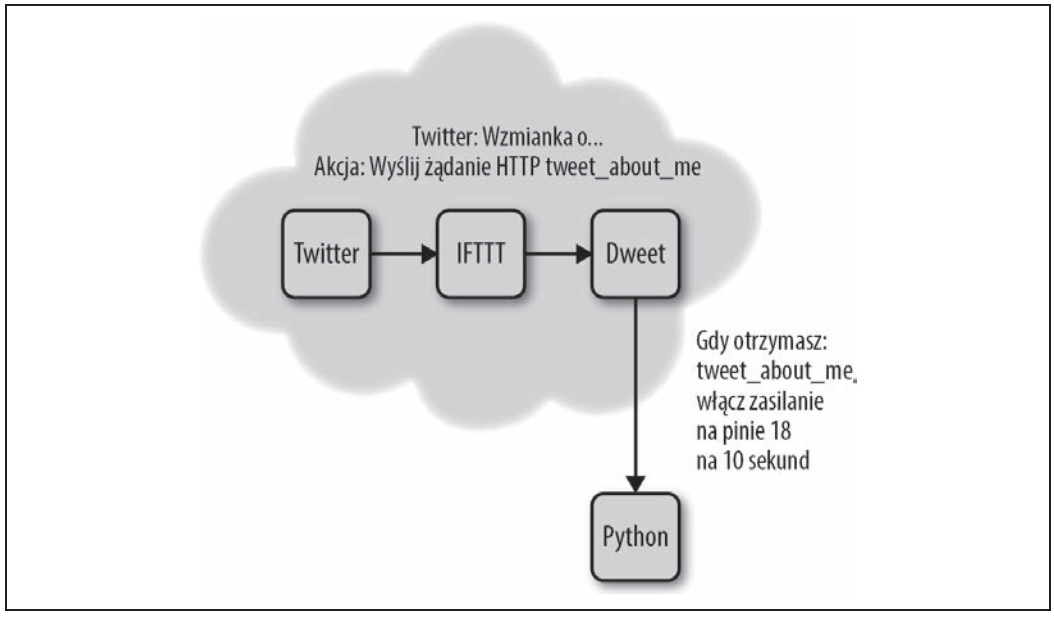
Receptura 16.6 wykonuje podobne zadanie, jednak robi to bardzo nieefektywnie, gdyż w sposób ciągły wysyła żądanie sieciowe, sprawdzając, czy kolor się zmienił.

Rozwiązanie

Wykorzystanie IFTTT (receptura 16.4) do wyszukiwania tylko tych tweetów, które nas interesują, a następnie wysyłanie żądania sieciowego do usługi Dweet jest wydajnym rozwiązaniem, ponieważ nie ma potrzeby ciągłego odpytywania przez żądania sieciowe. Usługa Dweet może wysyłać powiadomienia bezpośrednio do programu napisanego w Pythonie i uruchomionego na Twoim Raspberry Pi (rysunek 16.17).

I tak na przykład mógłbyś sprawić, by za każdym razem, gdy ktoś wymieni Twoją nazwę użytkownika, dioda LED podłączona do Raspberry Pi się zaświeciła.

W tym projekcie może być użyte każde urządzenie elektroniczne, które wykona coś zauważalnego, gdy zostanie włączone zasilanie na pinie 18. Do tego celu można użyć diody LED z przewodami (patrz receptura 9.10), pojedynczej diody LED umieszczonej na płytce prototypowej (patrz receptura 10.1) lub przekaźnika (patrz receptura 10.5), który daje najwięcej możliwości. Użycie tego ostatniego umożliwi realizację takiego projektu jak Bubblino (*http://bubblino.com*) — opartego na Arduino robota puszczającego bańki mydlane.



Rysunek 16.17. Współpraca między IFTTT, usługą Dweet i Pythonem

Najpierw zaloguj się do IFTTT (zobacz receptura 16.4) i stwórz nowy aplet. Z listy dostępnych kanałów akcji wybierz Twittera, kliknij opcję *New mention of you* (nowa wzmianka o Tobie) i zatwierdź, klikając *Create trigger*. W kolejnym kroku wybierz kanał *Webhooks* oraz akcję *Make a web request* (utwórz żądanie sieciowe) i wypełnij formularz danymi, które widnieją na rysunku 16.18.

Adres URL zawiera żądany parametr oraz składnik `text`, który będzie przechowywał treść tweeta. W naszym przypadku treść będzie wyświetlana jedynie w konsoli, jednak w przyszłości możesz chcieć użyć do tego celu ekranu LCD, dlatego warto wiedzieć, jak przekazywać dane z tweeta do programu napisanego w Pythonie.

Po wypełnieniu wszystkich pól naciśnij *Create action* (utwórz akcję).

Usługa dweet.io działa podobnie do Twittera dla kwestii związanych z Internetem rzeczy. Ma interfejs sieciowy, który umożliwia zarówno wysyłanie, jak i „nasłuchiwanie" wiadomości *dweet*. Nie trzeba tworzyć konta ani się logować, żeby korzystać z usługi Dweet. W naszym przykładzie wystarczą 2 rzeczy: IFTTT do wysyłania wiadomości do usługi Dweet i program napisany w Pythonie, który czeka na powiadomienia z usługi, że właśnie wydarzyło się coś, co Ciebie interesuje. Te dwie rzeczy połączone są ze sobą oznaczeniem `tweet_about_me`. Oznaczenie to nie jest unikalne i jeżeli więcej osób czytających tę książkę uruchamia ten przykład, będziesz otrzymywał również ich powiadomienia. Jeśli chcesz tego uniknąć, użyj unikalnego oznaczenia złożonego na przykład z losowych liter i cyfr.

Żeby Twój program mógł korzystać z usługi Dweet, musisz zainstalować bibliotekę `dweet`, uruchamiając poniższą komendę.

```
$ sudo pip3 install dweepy
```

Rysunek 16.18. Wypełnianie formularza akcji „Make a web request” w IFTTT

Program dla tej receptury nazywa się *r16_twitter_trigger.py*.

```
import time
import dweepy
from gpiozero import LED

KEY = 'tweet_about_me'
led = LED(18)

while True:
    try:
        for dweet in dweepy.listen_for_dweets_from(KEY):
            print('Tweet: ' + dweet['content']['text'])
            led.on()
            time.sleep(10)
            led.off()
    except Exception:
        pass
```

Podobnie jak inne programy omówione w tej książce, powyższy program znajdziesz w materiałach do pobrania (receptura 3.22).

Po uruchomieniu tego programu spróbuj wysłać tweeta ze swoją nazwą użytkownika. Dioda LED powinna zaświecić się na 10 sekund.

Omówienie

Powyższy program używa metody `listen_for_dweets_from`, by pozostawić otwarte połączenie z serwerem dweet.io. Program czeka na informacje wysłane z serwera, które są skutkiem wiadomości *dweet* wysłanej z IFTTT jako reakcja na opublikowanego tweeta zawierającego konkretną nazwę użytkownika. Blok `try/except` sprawia, że w momencie braku połączenia z siecią program ponownie rozpocznie proces „nasłuchiwania”.

Zobacz również

W recepturze 16.6 został opisany podobny projekt, lecz z zastosowaniem innego podejścia.

Raspberry Pi: morze możliwości dla inżyniera z pasją!

Raspberry Pi sukcesywnie zdobywa coraz więcej użytkowników. Dla niektórych jest to sposób na realizację życiowych pasji, dla innych — praktyczny, tani komputer, który może pracować pod kontrolą Linuxa i pełnić funkcję platformy obsługującej przeróżne urządzenia elektroniczne. Skupiona wokół Raspberry Pi społeczność nieprzerwanie tworzy nowe oprogramowanie oraz płytki interfejsów. To wszystko sprawia, że możliwości tego minikomputera stale rosną. Pojawiające się technologie przy odrobinie kreatywności mogą łatwo przeobrazić się w praktyczne i niedrogie, a przy tym zdecydowanie innowacyjne i nowoczesne rozwiązania.

To zaktualizowane wydanie znakomitego zbioru receptur ułatwiających wykorzystanie potencjału Raspberry Pi. Uwzględniono tu nowe modele tego komputera, a także zmiany i ulepszenia systemu operacyjnego Raspbian. Dodano rozdziały traktujące o dźwięku i automatyce domowej. Te receptury bez trudu wykorzystasz dla zwiększenia wygody we własnym domu. Dzięki lekturze poznasz podstawowe reguły tej technologii, aby łatwiej zrozumieć zagadnienia dotyczące konkretnej płytki czy kodu. Z tej pozycji możesz korzystać podobnie jak z książki kucharskiej: przeczytać od deski do deski albo skupić się na rozwiązaniu jednego, konkretnego problemu. Być może docenisz, że w recepturach dotyczących sprzętu uwzględniono przede wszystkim rozwiązania niewymagające lutowania obwodów.

W tej książce znajdziesz receptury, dzięki którym:

- rozpoczniesz pracę z Raspberry Pi, również w sieci
- zaprogramujesz Raspberry Pi w języku Python
- wykorzystasz technologię rozpoznawania obrazów
- będziesz sterować pracą silników, czujników i innych elementów elektroniki
- połączysz Raspberry Pi z różnymi urządzeniami wejściowymi
- podłączysz swój dom do internetu rzeczy

Dr Simon Monk jest inżynierem cybernetykiem, informatykiem i uzdolnionym hakerem. Po kilku latach pracy na uczelni zajął się działalnością biznesową jako współzałożyciel firmy Momote Ltd., która tworzy aplikacje mobilne. Obecnie dzieli swój czas pomiędzy pisanie bardzo dobrze przyjmowanych książek i projektowanie produktów dla Monk Makes — firmy, którą prowadzi wraz z żoną Lindą. Mieszka w Preston w Wielkiej Brytanii.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-6647-3
9 788328 366473
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 99,00 zł