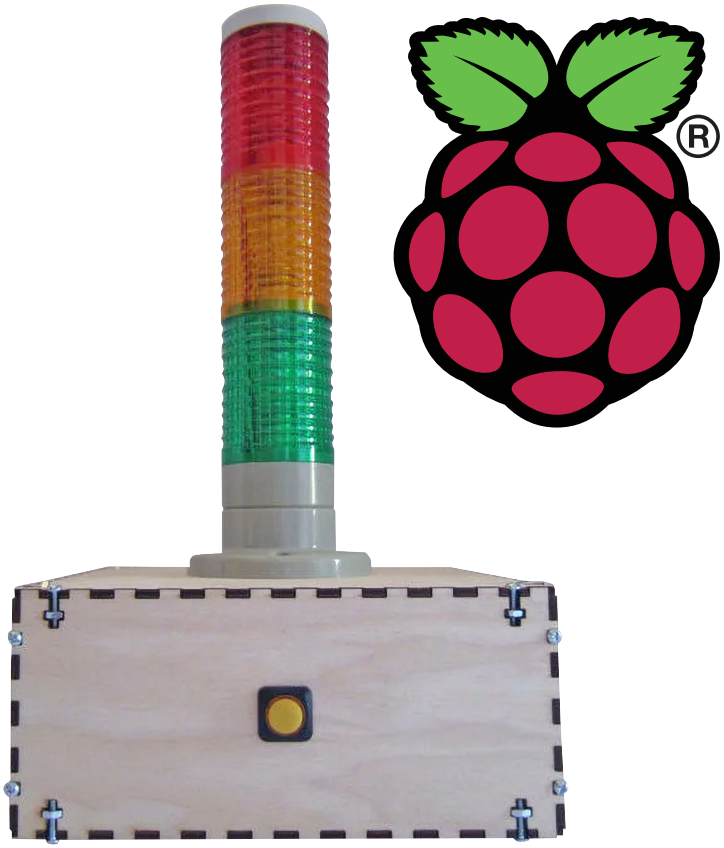
Monitor połączenia z Internetem zbudowany na Raspberry Pi

Coraz trudniej jest nam wyobrazić sobie sytuację, w której nie mamy połączenia z Internetem. Gdy nie możemy połączyć się z jakąś stroną w Internecie zazwyczaj obwiniamy (nie bezzasadnie) zdalny serwer, ale co w momencie gdy to nasze łącze płata nam figla? Jeszcze większy problem mamy w przypadku zastosowania licznych urządzeń Internetu Rzeczy, czy systemów automatyki domowej, które uzależnione są od serwerów znajdujących się w Chmurze.



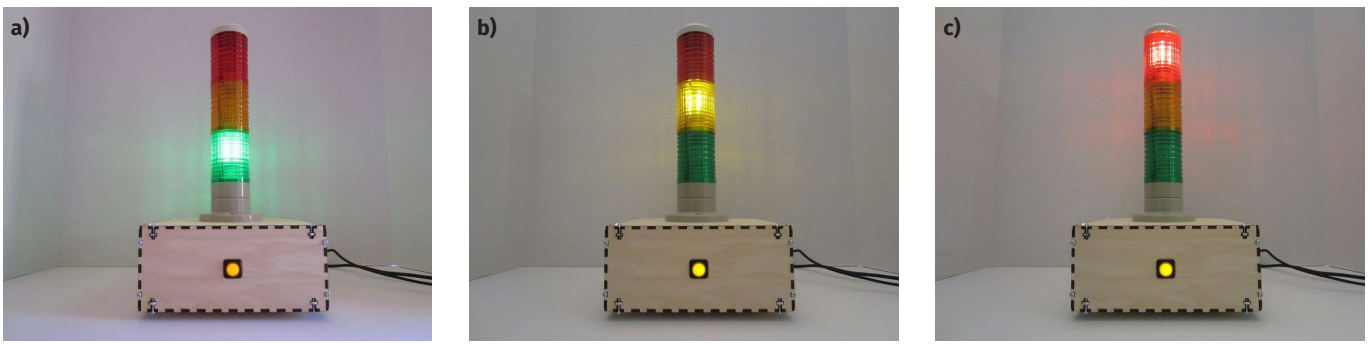
W poniższym artykule opisujemy prosty monitor połączenia z Internetem wykonany z użyciem komputera jednopłytkowego Raspberry Pi. System taki będzie przydatny wszystkim, którzy mają problem ze stabilnością łącza. Urządzenie zapewnia wizualny podgląd kondycji połączenia sieciowego w czasie rzeczywistym. Dzięki niemu, bez pomocy komputera, można sprawdzić stan połączenia z Internetem.

Zasada działania

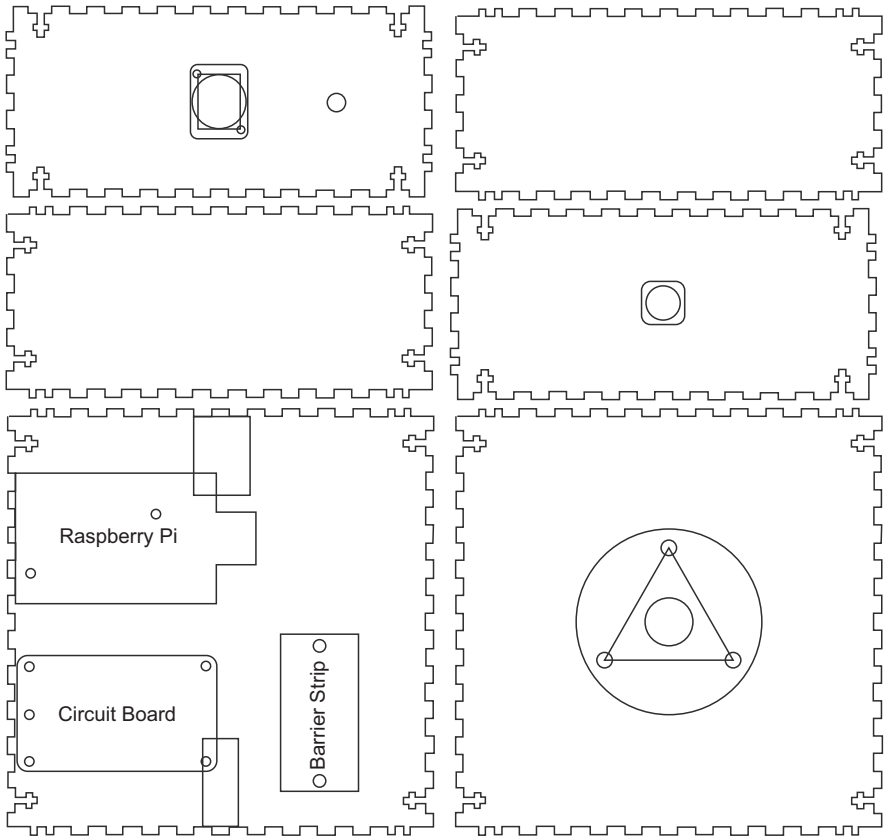
W urządzeniu zastosowano przemysłowy sygnalizator w postaci trójkolorowej kolumny świetlnej. Pracą kolumny steruje komputer jednopłytkowy Raspberry Pi. Skrypt zainstalowany na minikomputerze pracującym pod kontrolą systemu operacyjnyego Linuks, regularnie mierzy czas odpowiedzi szeregu stron internetowych z wykorzystaniem komendy *ping*. W momencie, gdy ponad 50% zapytań wysyłanych poprzez ping dochodzi do adresatów, system zapala zielone światło (**rysunek 1a**), w momencie, gdy odsetek poprawnych pingów mieści się pomiędzy 1% a 50%, zapalony zostaje żółty sygnalizator (**rysunek 1b**), a jeżeli wszystkie pingi nie dochodzą do celu, to zapalone zostaje światło czerwone (**rysunek 1c**).

Potrzebne elementy

Do budowy opisanego urządzenia potrzebnych będzie kilka elementów. Przede wszystkim potrzebny jest komputer jednopłytkowy Raspberry Pi – w dowolnej wersji, autor wykorzystał pierwszą wersję tego minikomputera, model B (z 512 MB pamięci RAM). Dodatkowo do komputera potrzebna jest



Rysunek 1. Trzy kolory sygnalizatora, dla różnych odsetków udanych pingów stron w Internecie: a) >50%, zielony, b) 1...50%, żółty, c) 0%, czerwony



Rysunek 2. Rozkład cięcia arkusza sklejki do wykonania elementów obudowy

karta SD – rekomendowana jest karta o pojemności co najmniej 8 GB oraz karta Wi-Fi w postaci modułu podłączanego na USB. Jeżeli komputer podłączany jest do sieci z pomocą połączenia kablowego, lub korzystamy z modelu Raspberry Pi, które wyposażone jest w bezprzewodowy interfejs sieciowy, zewnętrzna karta Wi-Fi jest nam zbędna.

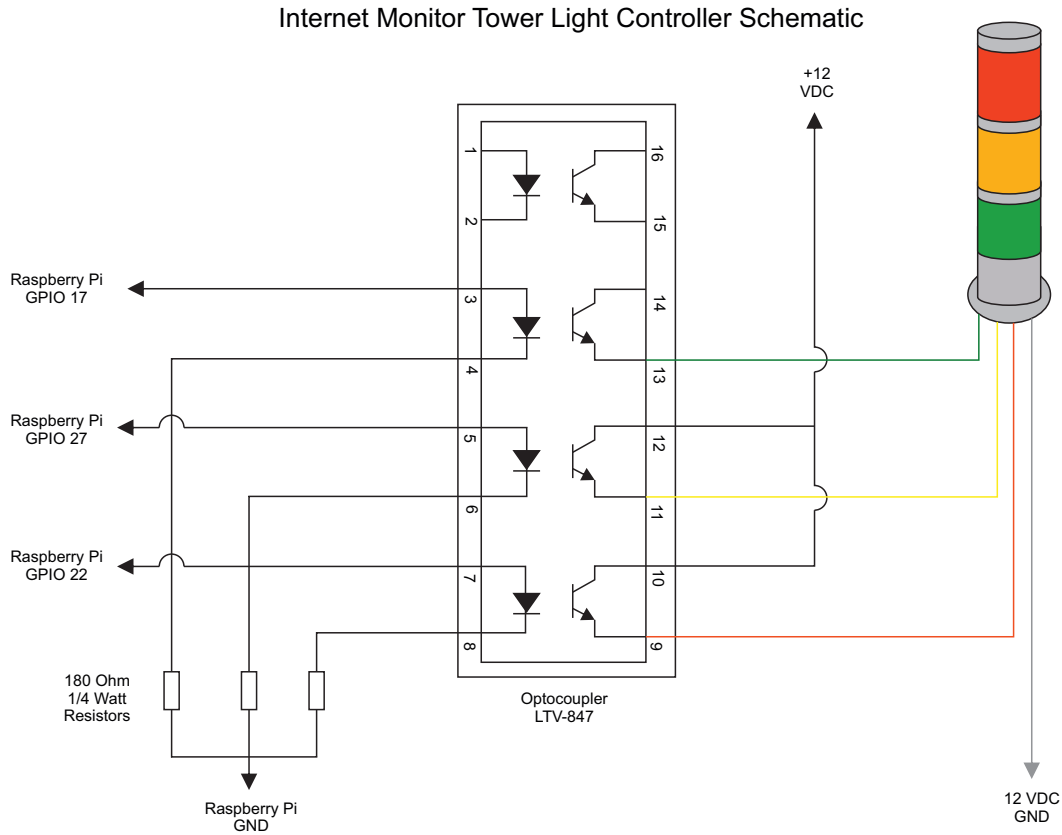
Potrzebny jest również sygnalizator. W tym projekcie najlepiej sprawdzi się sygnalizator kolumnowy z diodami LED lub żarówkami na 12 V. Do jego sterowania potrzebny będzie również transoptor LTV847 lub inny, kompatybilny, umożliwiający sterowanie napięciem 12 V oraz trzy oporniki 18 Ω i jeden opornik 470 Ω. Potrzebny będzie też przycisk chwilowy, który będzie służył do sterowania pracą komputera jednopłytkowego, aby możliwe było jego poprawne wyłączanie.

Oprócz powyższych elementów, do zestawienia opisywanego sygnalizatora potrzebne będą kable do połączeń wszystkich elementów, płytka uniwersalna oraz dwa zasilacze – 5 V z wtyczką microUSB, do zasilania Raspberry Pi oraz 12 V zakończony zwykłą wtyczką zasilaczową, do zasilania sygnalizatora kolumnowego.

Obudowa

Autor projektu włożył wiele wysiłku w zbudowanie estetycznej obudowy dla tego sygnalizatora. Obudowa wycięta została ze sklejki liściastej o grubości 3 mm. Na **rysunku 2** zaprezentowano rozkład cięcia na arkuszu sklejki o wymiarach ok. 40×40 cm. Dokładny projekt cięcia, dedykowany do wykorzystania z ploterem laserowym, pobrać można na stronie źródłowej.

Poszczególne elementy systemu przykręcone są do jego podstawy i osłonięte ściankami bocznymi obudowy, które łączą się z podstawą za pomocą ząbków. Sygnalizator kolumnowy zainstalowany jest na górnej płycie obudowy.



Rysunek 3. Schemat połączeń pomiędzy Raspberry Pi, a sterownikiem kolumny świetlnej

Schemat połączeń

Na **rysunku 3** zaprezentowano schemat połączeń pomiędzy Raspberry Pi, a sterownikiem kolumny świetlnej. Sygnalizator zasilany 12 V i Raspberry Pi są od siebie oddzielone galwanicznie poprzez transoptor. Pozwala to na zabezpieczenie komputera jednopłytkowego przed podwyższonym napięciem, a także umożliwia ewentualnie zasilanie oświetlenia w sygnalizatorze wyższym napięciem, np. 24 V.

Do komputera podłączony jest też pojedynczy przycisk, który służy do sterowania Raspberry Pi. Jego naciśnięcie powoduje zamknięcie systemu operacyjnego i wyłączenie komputera. Przykładowy przycisk, jaki wykorzystano w układzie, zaprezentowano na **rysunku 4**. Przycisk jest podświetlany, w związku z czym wykorzystuje dwie linie GPIO minikomputera. Jedna linia steruje diodą LED wbudowaną w przycisk zasilania – jest podłączona poprzez opornik 470 Ω do jednego z portów GPIO i do masy. Należy pamiętać o podłączeniu diody w kierunku przewodzenia. Przycisk podłączony jest do innej linii GPIO, naciśnięcie przycisku powoduje zwarcie linii do masy.

Oprogramowanie – skrypt do monitorowania połączenia

Za monitorowanie połączenia Internetowego i obsługę sygnalizatora kolumnowego odpowiedzialny jest skrypt *rpi-internet-monitor.py*, który znajdować będzie się w folderze */home/pi/python_programs/*. Aby pobrać folder z repozytorium w internecie i umieścić go w tym folderze w terminalu wpisujemy:

```
wget https://s3-us-west-1.amazonaws.com/talk2bruce/instructables/rpi-internet-monitor/rpi-internet-monitor.py
mv  rpi-internet-monitor.py /home/pi/python_programs/rpi-internet-monitor.py
```

Treść skryptu znajduje się w **listingu 1**. Działanie programu jest bardzo proste. W nieskończonej pętli *while* cyklicznie uruchamiana jest funkcja do realizacji pingów. Na podstawie jej wartości zwrotnej system zapala światło odpowiedniego koloru. Pingi realizowane są z wykorzystaniem systemowego narzędzia *ping*. Program konstruuje polecenie:

```
/bin/ping  -c 1 nazwa_strony
```

Polecenie to zostaje uruchomione, jako osobny proces z wykorzystaniem biblioteki *subprocess*, która została wcześniej zaimportowana. Skrypt wychwytuje zwracaną informację z polecenia – w przypadku błędu, spowodowanego nieudanym pingiem, wychwytuje on błąd, po czym zwraca zero. W przypadku, gdy ping zakończy się poprawnie, funkcja ta zwraca jeden. Pozwala to na proste obliczanie odsetka udanych pingów w dalszej części programu.

Rysunek 4. Przycisk zastosowany w urządzeniu

Listing 1. Skrypt do monitorowania połączenia

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

import subprocess
import sys
import time
import RPi.GPIO as GPIO

GPIO_GREEN_LIGHT = 17 # linia IO podłączona do zielonego światła
GPIO_AMBER_LIGHT = 27 # linia IO podłączona do żółtego światła
GPIO_RED_LIGHT = 22 # linia IO podłączona do czerwonego światła

DELAY_BETWEEN_PINGS = 1 # opóźnienie (w sekundach) pomiędzy pingami
DELAY_BETWEEN_TESTS = 120 # opóźnienie (w sekundach) pomiędzy testami

SITES = [„google.com”, „comcast.com”] # lista domen do testowania

# drukowanie na ekranie wiadomości debugowych, jeżeli opcja jest włączona
def debug_message(debug_indicator, output_message):
          if debug_indicator:
                    print output_message

# wykorzystanie komendy ping z linii poleceń systemu operacyjnego
def ping(site):
          cmd = „/bin/ping -c 1 „ + site
          try:
                    output = subprocess.check_output(cmd,
                    stderr=subprocess.STDOUT, shell=True)
          except subprocess.CalledProcessError, e:
                    debug_message(debug, site + „: nieosiągalny”)
                    return 0
          else:
                    debug_message(debug, site + „: osiągalny”)
          return 1

# funkcja realizująca ping do listy stron i
# obliczająca odsetek udanych pingów
def ping_sites(site_list, wait_time, times):
          successful_pings = 0
          attempted_pings = times * len(site_list)
          for t in range(0, times):
                    for s in site_list:
                              successful_pings += ping(s)
                              time.sleep(wait_time)
          debug_message(debug, „Odsetek udanych pingów: „ +
          str(int(100 * (successful_pings / float(attempted_pings)))) + „%”)
          # funkcja zwraca odsetek udanych pingów
          return successful_pings / float(attempted_pings)

# włączenie żółtego światła
def lamp_amber_on():
          debug_message(debug, „>>> Czerwone OFF; Żółte ON; Zielone OFF”)
          GPIO.output(GPIO_RED_LIGHT, False)
          GPIO.output(GPIO_AMBER_LIGHT, True)
          GPIO.output(GPIO_GREEN_LIGHT, False)

# włączenie zielonego światła
def lamp_green_on():
          debug_message(debug, „>>> Czerwone OFF; Żółte OFF; Zielone ON”)
          GPIO.output(GPIO_RED_LIGHT, False)
          GPIO.output(GPIO_AMBER_LIGHT, False)
          GPIO.output(GPIO_GREEN_LIGHT, True)

# włączenie czerwonego światła
def lamp_red_on():
          debug_message(debug, „>>> Czerwone ON; Żółte OFF; Zielone OFF”)
          GPIO.output(GPIO_RED_LIGHT, True)
          GPIO.output(GPIO_AMBER_LIGHT, False)
          GPIO.output(GPIO_GREEN_LIGHT, False)

# wyłączenie wszystkich świateł
def lamp_all_off():
          debug_message(debug, „>>> Czerwone OFF; Żółte OFF; Zielone OFF”)
```

Oprogramowanie – skrypt do obsługi przycisku zasilania

Drugim istotnym skryptem, jest moduł obsługujący przycisk zasilania. Ten prosty skrypt przedstawiono na **listingu 2**. Aby pobrać i przenieść skrypt do tego samego folderu, co w przypadku pierwszego prezentowanego tutaj modułu, w linii komend wpisujemy:

```
wget https://s3-us-west-1.
amazonaws.com/talk2bruce/
instructables/rpi-internet-
monitor/rpi-halt-btn.py
mv  rpi-halt-btn.py /home/pi/
python_programs/ rpi-halt-btn.py
```

Skrypt wysyła do systemu operacyjnego polecenie *halt*, które rozpoczyna zamykanie systemu operacyjnego. Jest poprzedzone poleceniem *sudo*, które oznacza, że komenda zatrzymania systemu wydawana jest z prawami roota (administratora systemu operacyjnego). Jednakże, aby do tego doszło, układ musi wykryć na linii GPIO podłączonej do przycisku, zbocze opadające. Oczekiwanie można przerwać, nie wyłączając komputera, poprzez naciśnięcie dowolnego przycisku na klawiaturze.

Konfiguracja systemu operacyjnego do pracy automatycznej

Ten krok polega na skonfigurowaniu systemu do automatycznego uruchamiania skryptów, do monitorowania połączenia Internetowego i obsługi przycisku zasilania. Skrypty muszą zostać uruchomione automatycznie, po uruchomieniu systemu operacyjnego.

Aby dodać skrypt do obsługi przycisku zasilania do automatycznego uruchamiania przy starcie systemu, musimy edytować plik *rc.local*. W tym celu w terminalu wpisujemy:

```
sudo nano /etc/rc.local
```

i dodajemy poniższe polecenie na samym dole pliku przed wierszem, w którym znajduje się wpis *„exit 0”*.

```
python /home/pi/python_programs/
rpi-halt-btn.py&
```

Aby skonfigurować system do uruchamiania programu do monitorowania sieci, musimy uruchomić skrypt Pythona, gdy połączenie Wi-Fi jest uruchomione na Raspberry Pi. W tym celu musimy dodać wpis w pliku */etc/network/interfaces*. Plik edytujemy za pomocą *nano*, wpisując w terminalu:

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

i na końcu pliku dodajemy polecenie:

```
post-up python /home/pi/python_
programs/rpi-internet-monitor.py
```

Spowoduje to uruchomienie się skryptu po połączeniu komputera z siecią Wi-Fi. Te dwa wpisy kończą konfiguracje komputera jednopłytkowego do pracy jako monitor połączenia sieciowego.

Teraz możemy wyłączyć Raspberry Pi za pomocą polecenia:

```
sudo halt
```

Listing 1. cd.

```
          GPIO.output(GPIO_RED_LIGHT, False)
          GPIO.output(GPIO_AMBER_LIGHT, False)
          GPIO.output(GPIO_GREEN_LIGHT, False)

# test lamp - mruganie lampami w sekwencji, pięciokrotnie
def lamp_test():
          debug_message(debug, „Testowanie lamp”)
          # jedna dziesiąta sekundy opóźnienia w testach
          TEST_DELAY = 0.1
          for i in range(0, 5):
                    time.sleep(TEST_DELAY)
                    lamp_red_on()
                    time.sleep(TEST_DELAY)
                    lamp_amber_on()
                    time.sleep(TEST_DELAY)
                    lamp_green_on()
          lamp_all_off()
          debug_message(debug, „Testowanie lamp zakończone”)

# Tutaj zaczyna się główna część programu

# sprawdzamy czy użytkownik chce wypisywać informacje debugowe
debug = False
if len(sys.argv) > 1:
          if sys.argv[1] == „-debug”:
                    debug = True
          else:
                    print „nieznane polecenie: „ + sys.argv[1]
                    sys.exit(1)

# konfiguracja pinów GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_GREEN_LIGHT, GPIO.OUT)
GPIO.setup(GPIO_AMBER_LIGHT, GPIO.OUT)
GPIO.setup(GPIO_RED_LIGHT, GPIO.OUT)

# mrugnięcie lampami, sygnalizujące, że program uruchamia się
lamp_all_off()
lamp_test()
time.sleep(0.5)
# po zakończeniu testów, zapala się żółta lampa
lamp_amber_on()

# główna pętla programu: uruchamianie pingów,
# zapalanie lampki, opóźnienie i powtarzanie w pętli
test = 0
while True:
          test+=1
          debug_message(debug, „----- Test „ + str(test) + „ -----”)
          success = ping_sites(SITES, DELAY_BETWEEN_PINGS, 2)
          if success == 0:
                    lamp_red_on()
          elif success <= .50:
                    lamp_amber_on()
          else:
                    lamp_green_on()

          debug_message(debug, „Oczekiwanie „ +
          str(DELAY_BETWEEN_TESTS) + „ sekund do kolejnego testu.”)

          time.sleep(DELAY_BETWEEN_TESTS)
```

Po wyłączeniu minikomputera należy odłączyć wszystkie kable, które dodaliśmy (monitor, klawiatura, mysz etc) zostawiając w środki kartę SD. Możemy teraz zamontować Raspberry Pi w urządzeniu i podłączyć do niego sygnalizator kolumnowy i inne elementy.

Nikodem Czechowski

http://bit.ly/2kIdg0L

Listing 2. Skrypt do obsługi przycisku zasilania

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

import RPi.GPIO as GPIO
import subprocess, time

# linia GPIO podłączona do przycisku
HALT_SWITCH_GPIO_PIN = 24
# linia GPIO podłączona do diody sygnalizacyjnej w przycisku
HALT_SWITCH_LED_GPIO_PIN = 23

# konfiguracja linii GPIO
GPIO.setmode(GPIO.BCM)
# ustawienie linii GPIO diody LED jako wyjścia
GPIO.setup(HALT_SWITCH_LED_GPIO_PIN, GPIO.OUT)
# ustawienie linii GPIO przycisku jako wejścia z podciągnięciem do góry
GPIO.setup(HALT_SWITCH_GPIO_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
# ustawienie linii GPIO diody LED w stanie wysokim - zapalenie diody
GPIO.output(HALT_SWITCH_LED_GPIO_PIN, True)

print(„rpi-halt-btn: uruchomiono, trwa oczekiwanie na naciśnięcie przycisku”)

# oczekiwanie na naciśnięcie przycisku
# gdy zostanie on naciśnięty, skrypt wysyła
# do systemu operacyjnego polecenie zatrzymania
try:
          # oczekiwanie na zbocze opadające na linii GPIO przycisku
          GPIO.wait_for_edge(HALT_SWITCH_GPIO_PIN, GPIO.FALLING)
except KeyboardInterrupt:
          print(„Zatrzymano przez użytkownika”)
          GPIO.cleanup()
print(„rpi-internet-monitor: Naciśnięto przycisk - trwa wyłączanie systemu”)
GPIO.output(HALT_SWITCH_LED_GPIO_PIN, False)
time.sleep(0.5)
GPIO.output(HALT_SWITCH_LED_GPIO_PIN, True)
# wysłanie do systemu komendy sudo, a następnie halt
subprocess.call([„sudo”,”halt”])
```