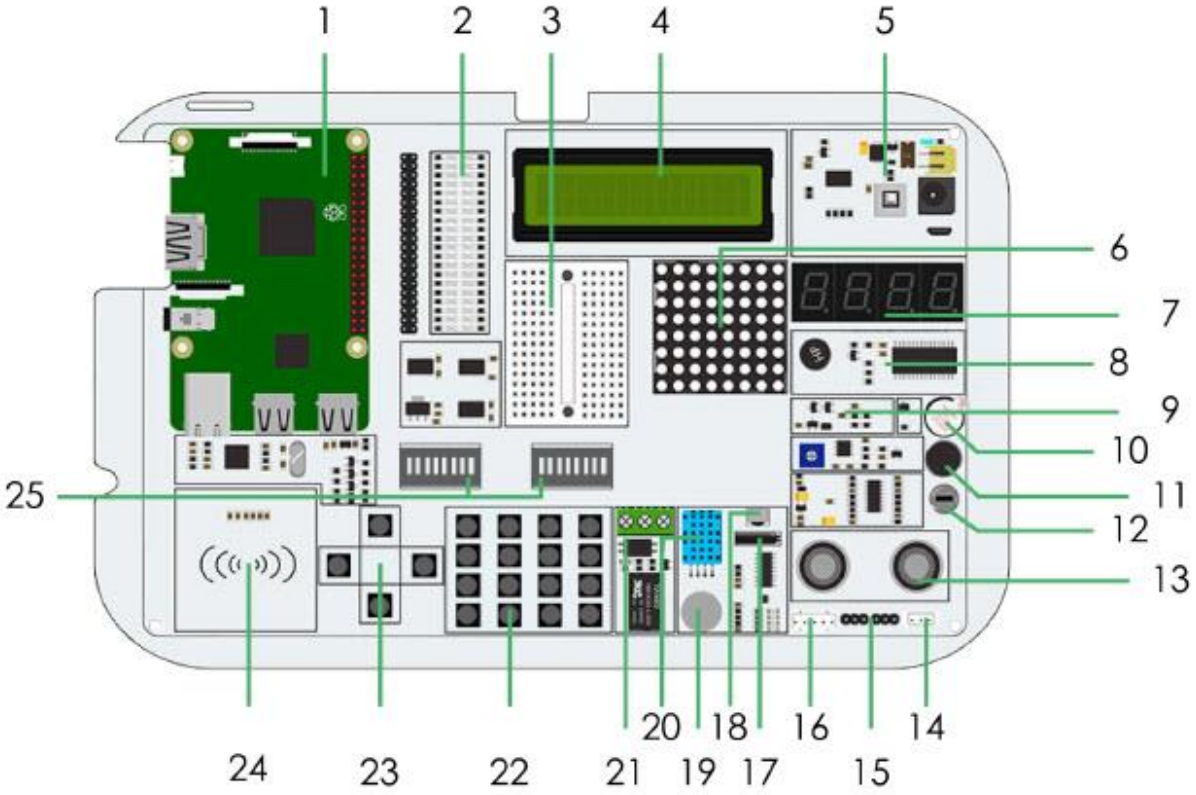
INSTRUKCJA OBSŁUGI

Nr produktu 001762929

Zestaw eksperymentalny Raspberry Pi® Joy-it RB-JOYPI

Dane do logowania to:
Nazwa użytkownika: pi
Hasło: 12345

2. Przegląd urządzenia



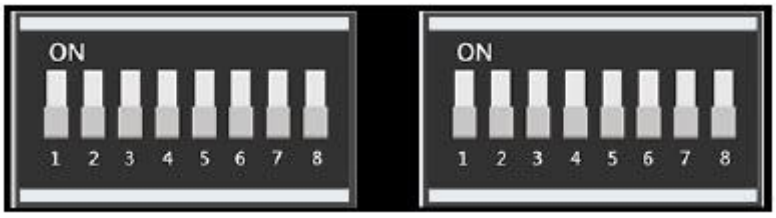
1	Raspberry Pi
2	Wyświetlacz LED GPIO
3	Breadboard - do tworzenia niestandardowych curciuts z zewnętrznymi modułami
4	Moduł LCD 16x2 (MCP23008)
5	Zasilanie
6	Matryca LED 8x8 (MAX7219)
7	7-segmentowy wyświetlacz LED (HT16K33)
8	Moduł wibracyjny

9	Czujnik światła - do pomiaru natężenia światła (BH1750)
10	Brzęczyk - do generowania dźwięków alarmowych
11	Czujnik dźwięku
12	Czujnik ruchu (LH1778)
13	Czujnik ultradźwiękowy - używany do pomiaru odległości
14/15	Interfejsy serwo - do podłączenia silników serwo
16	Interfejs silnika krokowego
17	Czujnik pochylenia (SW-200D)

18	Czujnik podczerwieni
19	Czujnik dotyku
20	Czujnik DHT11 - do pomiaru wilgotności i temperatury
21	Przekaźnik - do otwierania i zamykania obwodów elektronicznych
22	Macierz kluczowa
23	Niezależne przyciski
24	Moduł RFID - do odczytu i zapisu danych przez RFID / NFC (MFRC522)
25	Przełącznik - do przełączania między czujnikami i modułami

3. ZMIANA MODUŁÓW I KORZYSTANIE Z GPIOS

Zmiana modułów



Płytka Joy-Pi zawiera 2 jednostki przełączające. Każda jednostka zawiera 8 przełączników. Przełączniki umożliwiają przełączanie między użyciem czujników i modułów. Ponieważ Raspberry Pi ma tylko ograniczoną liczbę pinów GPIO, potrzebujemy przełączników, aby móc korzystać z większej liczby czujników.

Czujniki / moduły	Jednostka przełączająca	Przełączniki
Tablica przycisków	Lewa	1 – 8
Macierz przycisków	Lewa	1 – 8
Moduł wibracyjny	Prawa	1
Czujnik pochylenia	Prawa	2
Silnik krokowy	Prawa	3,4,5,6
Siłownik	Prawa	7,8

KORZYSTANIE Z GPIOS

Poniżej wyjaśnimy bardziej szczegółowo, czym są GPIO, jak działają i jak są kontrolowane.

GPIO oznacza: „Wejście / wyjście ogólnego przeznaczenia” (uniwersalne wejście / wyjście). Piny GPIO nie mają określonego celu. Mogą być konfigurowane jako wejście lub wyjście i mają ogólne przeznaczenie. To zależy od tego, co chcesz osiągnąć.

Przykład pinu wejściowego: Przycisk
Jeśli przycisk jest wciśnięty, sygnał zostanie przesłany przez pin wejściowy do RaspberryPi

Przykładowy pin wyjściowy: Brzęczyk
Wyślij sygnał przez pin wyjściowy, aby sterować brzęczykiem. Piny GPIO znajdują się po prawej stronie płytki Raspberry Pi, jeśli zaczynasz z perspektywy Joy-Pi.

Istnieją 2 możliwe schematy GPIO Raspberry Pi: GPIO-BOARD i GPIO-BCM

Opcja GPIO-BOARD wskazuje, że odnosisz się do pinów za pomocą numeru pinu. Oznacza to, że będą używane numery pinów wymienione poniżej.

Opcja GPIO.BCM oznacza, że odwołujesz się do pinów „Broadcom SOC Channel”. To są liczby po „GPIO”.

Numer płyty GPIO:			Numer płyty GPIO:	
1	3.3V DC		2	5V DC
3	GPIO 2 (SDA1, I2C)		4	5V DC
5	GPIO 3 (SCL1, I2C)		6	Ground
7	GPIO 4		8	GPIO 14 (TXD0)
9	Ground		10	GPIO 15 (RXD0)
11	GPIO 17		12	GPIO 18
13	GPIO 27		14	Ground
15	GPIO 22		16	GPIO 23
17	3.3V		18	GPIO 24
19	GPIO 10 (SPI, MOSI)		20	Ground
21	GPIO 9 (SPI, MISO)		22	GPIO 25
23	GPIO 11 (SPI, CLK)		24	GPIO 8 (SPI)
25	Ground		26	GPIO 7 (SPI)
27	ID_SD (I2C, EEPROM)		28	ID_SC
29	GPIO 5		30	Ground
31	GPIO 6		32	GPIO 12
33	GPIO 13		34	Ground
35	GPIO 19		36	GPIO 16
37	GPIO 26		38	GPIO 20
39	Ground		40	GPIO 21

PRZYDZIAŁ PINÓW GPIO WEDŁUG SCHEMATU GPIO.BOARD

Numer płyty GPIO:	Zastosowane czujniki i moduły:
1	3,3 V
2	5,0 V
3	I2C, SDA1 (czujnik światła, wyświetlacz LCD, 7-segmentowy wyświetlacz)
4	5,0 V
5	I2C, SCL1 (czujnik światła, wyświetlacz LCD, 7-segmentowy wyświetlacz)
6	Uziemienie
7	Czujnik DHT11
8	TXD0
9	Uziemienie
10	RXD0
11	Czujnik dotykowy
12	Brzęczyk
13	Matryca przycisków (ROW1), silnik wibracyjny
14	Uziemienie

15	Matryca przycisków (ROW2), czujnik pochylenia
16	Czujnik ruchu
17	3,3 V
18	Czujnik dźwiękowy
19	SPI
20	Uziemienie
21	SPI
22	Serwo2, matryca przycisków (COL1), przycisk w lewo

23	SPI
24	Moduł RFID
25	Uziemienie
26	Matryca LED
27	ID_SD (I2C, EEPROM (elektrycznie kasowalna programowalna pamięć tylko do odczytu))
28	ID_SC
29	Silnik krokowy (STEP 1), matryca przycisków (ROW 3)
30	Uziemienie

31	Silnik krokowy (STEP 2), matryca przycisków (ROW 4)
32	Czujnik ultradźwiękowy (echo)
33	Silnik krokowy (STEP 3), matryca przycisków (COL 4), przycisk w dół
34	Uziemienie
35	Silnik krokowy (STEP 4), matryca przycisków (COL3), przycisk w prawo
36	Czujnik ultradźwiękowy (TRIG)
37	Serwo1, matryca przycisków (COL2), przycisk w górę
38	Czujnik podczerwieni
39	Uziemienie
40	Przekaźnik

W naszych przykładach używamy języka Python do sterowania pinami GPIO. W Pythonie znajduje się biblioteka o nazwie „RPi.GPIO”. Jest to biblioteka, która pomaga programowo sterować pinami za pomocą Pythona.

Spójrz na poniższy przykład i komentarze w kodzie, aby lepiej zrozumieć, jak to działa.

Pierwszym krokiem będzie zaimportowanie biblioteki przez wpisanie polecenia „RPi.GPIO as GPIO”, a następnie biblioteka „time” zawiera polecenie „import time”.

Następnie ustawiamy tryb GPIO na GPIO.BOARD. W naszym przykładzie deklarujemy pin wejściowy jako numer 11, a pin wyjściowy jako pin 12 (wejście to czujnik dotyku, a wyjście to brzęczyk). Wysyłamy sygnał do pinu wyjściowego, czekamy 1 sekundę, a następnie wyłączamy. Następnie, aby potwierdzić dane wejściowe, przechodzimy przez pętlę do momentu odebrania sygnału wejściowego GPIO.input. Drukujemy „Input Given”, aby upewnić się, że kliknięcie zostało potwierdzone, wyczyść GPIO za pomocą GPIO.cleanup () i dokończ skrypt.

```
import RPi.GPIO as GPIO
import time                                          #import lybraries
import signal

TOUCH = 11                                           #Declaring variables
BUZZER = 12                                        #and connecting pins

def setup_gpio():                                    #Definition of in and outputs
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TOUCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(BUZZER, GPIO.OUT)

def do_smt(channel):                                 #class for buzzer output and
    print("Touch detected")                        #detected touch
    GPIO.output(BUZZER, GPIO.HIGH)                   #Signal output
    time.sleep (1)                                   #Wait 1 second
    GPIO.output(BUZZER, GPIO.LOW)                    #Stop signal Output

def main():
    setup_gpio()
    try:                                             #Checking if touch is detected
        GPIO.add_event_detect(TOUCH, GPIO.FALLING, callback=do_smt, bouncetime=200)
        signal.pause()
    except KeyboardInterrupt:                        #CTRL+C is closing the script
        pass
    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

Aby dowiedzieć się więcej o celu i zastosowaniu GPIO, zalecamy przeczytanie oficjalnej dokumentacji na temat pinów GPIO napisanej przez fundację Raspberry Pi.
https://www.raspberrypi.org/documentation/usage/gpio/

POBIERANIE WSTĘPNIE ZAINSTALOWANEGO SYSTEMU OPERACYJNEGO

W pierwszym kroku musisz pobrać plik obrazu z systemem operacyjnym Joy-Pi.
Plik można znaleźć na naszej stronie internetowej pod adresem https://joy-pi.net/downloads/.

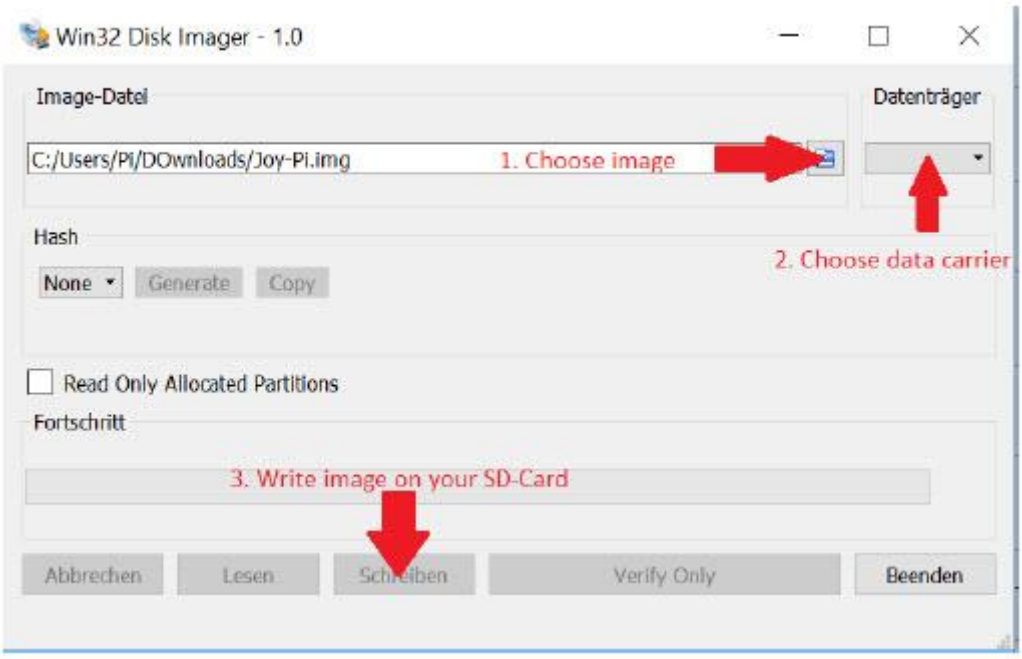
1. Załaduj plik .Zip na swój komputer i rozpakuj go do dowolnego folderu. Powinieneś otrzymać plik .ISO

2. Podłącz kartę MicroSD do komputera za pomocą dołączonego czytnika kart MicroSD.

3. Sformatuj teraz kartę MicroSD za pomocą programu „SD Formatter”

4. Uruchom program „Win32DiskImager” i wybierz rozpakowany plik .Iso, a następnie kliknij przycisk „Zapisz”, aby skopiować obraz na kartę MicroSD.

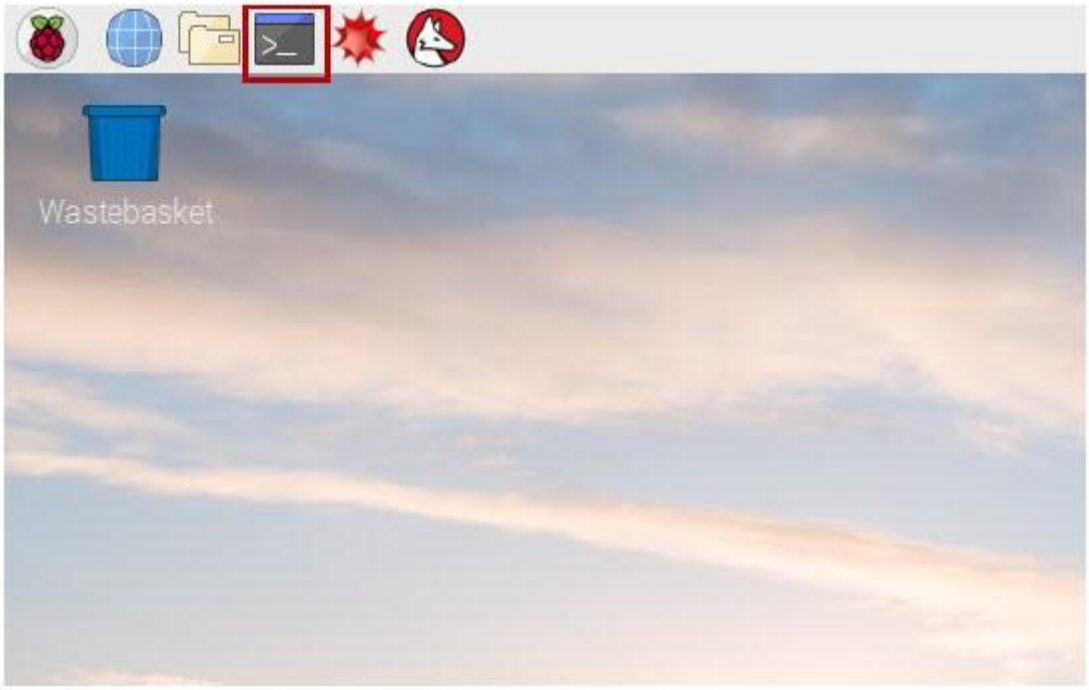
5. Teraz karta MicroSD jest gotowa do użycia, możesz ją teraz włożyć do swojego Raspberry Pi.



4. KORZYSTANIE Z PYTHON I LINUX

Ten krok jest opcjonalny, ale ułatwia wykonywanie skryptów bez konieczności ich indywidualnego tworzenia.
Skrypty użyte w tym przewodniku można pobrać bezpośrednio z pakietu. Wystarczy postępować zgodnie z poniższymi instrukcjami:

1. Otwórz „Terminal”. Używamy tego do uruchamiania większości naszych skryptów Pythona oraz pobierania rozszerzeń i skryptów.



2. Po pomyślnym otwarciu terminala musimy pobrać archiwum skryptów na pulpit za pomocą następujących poleceń:

cd Desktop/
wget http://anleitung.joy-it.net/wp-content/uploads/2019/01/Joy-Pi.zip

3. Naciśnij „Enter” na klawiaturze. Teraz wystarczy tylko rozpakować archiwum:

unzip JoyPi.zip

4. Naciśnij "Enter" i poczekaj, aż proces się zakończy.

5. Poleceniem "cd" przechodzimy do odpowiedniego katalogu, abyśmy mogli skorzystać ze skryptów, które się w nim znajdują:

cd Joy-Pi

Uwaga! Za każdym razem, gdy restartujesz terminal, musisz powtórzyć kroki zmiany katalogu.

WYKONYWANIE SKRYPTÓW PYTHONOWYCH

Po pomyślnym pobraniu naszego skryptu chcielibyśmy go wykonać. Otwórz terminal ponownie i postępuj zgodnie z poniższymi instrukcjami, aby uruchomić skrypt:

Napisz polecenie „sudo python <nazwa skryptu>”, aby wykonać skrypt w języku Python.
Na przykład:

sudo python buzzer.py

Polecenie sudo daje nam uprawnienia roota (uprawnienia administratora), które są później wymagane przez bibliotekę GPIO. Piszemy „python”, aby powiedzieć systemowi, że chcemy wykonać polecenie w Pythonie. Na koniec piszemy nazwę skryptu, tak jak umieściliśmy go na pulpicie. Upewnij się, że zawsze znajdujesz się we właściwym folderze podczas wykonywania polecenia.

5. LEKCJE

5.1 LEKCJA 1: UŻYWANIE BRZĘCZYKA DO DŹWIĘKÓW OSTRZEGAWCZYCH

W poprzednim wyjaśnieniu nauczyliśmy się, jak używać pinu GPIO zarówno jako wyjścia, jak i wejścia. Aby to teraz przetestować, posłużymy się prawdziwym przykładem i zastosujemy naszą wiedzę z poprzedniej lekcji. Moduł, którego będziemy używać to „Brzęczyk”.

Użyjemy wyjścia GPIO, aby wysłać sygnał do brzęczyka i zamknąć obwód, aby wygenerować głośny brzęczenie. Następnie wyślemy kolejny sygnał, aby go wyłączyć.

Brzęczyk znajduje się po prawej stronie płytki Joy-Pi i można go łatwo rozpoznać po głośnym dźwięku, który wydaje po aktywacji. Kiedy używasz Raspberry Pi po raz pierwszy, brzęczyk może mieć naklejkę ochronną. Upewnij się, że ta naklejka została usunięta przed użyciem brzęczyka.

Podobnie jak w poprzednim przykładzie przygotowaliśmy specjalny skrypt ze szczegółowymi komentarzami, który wyjaśni, jak działa cały proces buzzera i jak możemy sterować buzzerem za pomocą GPIO.

Najpierw importujemy bibliotekę RPi.GPIO i bibliotekę czasu. Następnie konfigurujemy brzęczyk. Na pinie 12 ustawiamy tryb GPIO na GPIO BOARD, a pin na OUTPUT.

Wysyłamy sygnał przez 0,5 sekundy, a następnie go wyłączamy.

Przykład kodu brzęczyka:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO     #import librarys
import time

buzzer_pin = 12             #define buzzer pin

GPIO.setmode(GPIO.BOARD)
GPIO.setup(buzzer_pin, GPIO.OUT)

# Make buzzer sound
GPIO.output(buzzer_pin, GPIO.HIGH)
#wait 0.5 seconds
time.sleep(0.5)
# Stop buzzer sound
GPIO.output(buzzer_pin, GPIO.LOW)

GPIO.cleanup()
```

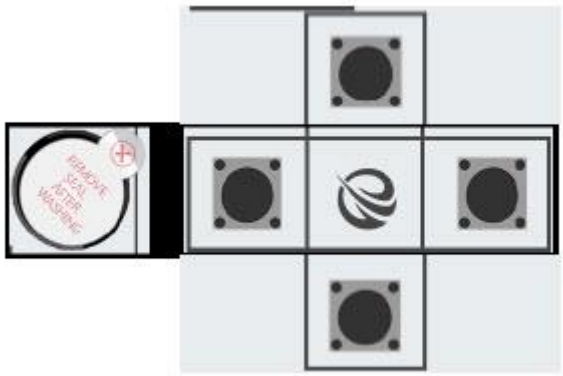
Wykonaj następujące polecenia i spróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python buzzer.py

5.2 LEKCJA 2: STEROWANIE BRZĘCZYKIEM ZA POMOCĄ PRZYCISKÓW

Po pomyślnym zademonstrowaniu, jak włączać i wyłączać brzęczyk, nadszedł czas, aby uczynić rzeczy bardziej ekscytującymi. W tej lekcji połączymy przycisk z brzęczykiem, tak aby brzęczyk był włączany tylko przez naciśnięcie przycisku.

Tym razem użyjemy 2 konfiguracji GPIO. Jednym z nich będzie GPIO.INPUT, które przyjmuje przycisk jako wejście, a innym będzie GPIO.OUTPUT, które wysyła sygnał do brzęczyka, aby wyprowadzić dźwięk.



Uwaga! W tym przykładzie musisz przełączać się między modułami. Włącz przełącznik nr 5, 6, 7 i 8 na lewym zespole przełączającym. Wszystkie pozostałe przełączniki powinny być wyłączone.

W naszym przykładzie używamy górnego z 4 klawiszy w lewym dolnym rogu. Teoretycznie jednak można użyć dowolnego z 4 kluczy. Jeśli nadal chcesz użyć innego klucza, musisz odpowiednio zmienić przypisanie pinów.

GPIO37	Górny przycisk
GPIO27	Dolny przycisk
GPIO22	Lewy przycisk
GPIO35	Prawy przycisk

W tej części naszego samouczka musimy użyć 2 ustawień GPIO. Jedno wejście i jedno wyjście. Wejście GPIO służy do określenia, kiedy klawisz został naciśnięty, a wyjście GPIO służy do aktywacji brzęczyka po naciśnięciu klawisza.

Jak widać w poniższym przykładzie, zdefiniowaliśmy 2 piny o nazwach buzzer_pin i button_pin. Program działa do momentu naciśnięcia klawiszy CTRL + C.

Kiedy naciskasz klawisz na Joy-Pi, brzęczyk wydaje dźwięk! Zwolnij klawisz, a brzęczyk się zatrzyma. Przykładowy kod:

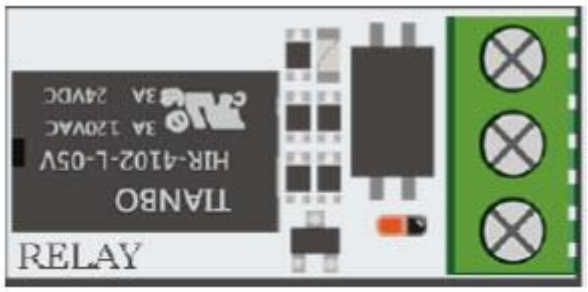
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time

# configure both button and buzzer pins
button_pin = 37
buzzer_pin = 12

# set board mode to GPIO.BOARD
GPIO.setmode(GPIO.BOARD)

# setup button pin as input and buzzer pin as output
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(buzzer_pin, GPIO.OUT)

try:
    while True:
        # check if button pressed
        if(GPIO.input(button_pin) == 0):
            # set buzzer on
            GPIO.output(buzzer_pin, GPIO.HIGH)
        else:
            # it's not pressed, set button off
            GPIO.output(buzzer_pin, GPIO.LOW)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Wykonaj następujące polecenia i spróbuj sam:
cd /home/pi/Desktop/Joy-Pi/
sudo python button_buzzer.py

5.3 LEKCJA 3: JAK DZIAŁA PRZEKAŹNIK I JAK GO KONTROLOWAĆ

Teraz, gdy wiemy już wszystko, co powinniśmy wiedzieć o brzęczyku, czas na następną lekcję. Teraz dowiemy się, jak korzystać z przekaźnika, jaka jest funkcja przekaźnika i jak nim sterować.

Przekaźniki są używane do sterowania obwodem za pomocą oddzielnego sygnału małej mocy lub gdy kilka obwodów musi być sterowanych jednym sygnałem. Jeśli podłączysz przewody do „NC” i „COM” i wyślesz sygnał GPIO.HIGH, przekaźnik zamknie się i wyłączy niestandardowy obwód. Jeśli zatrzymasz sygnał, przekaźnik otworzy się i aktywuje twój obwód niestandardowy.



Przekaźnik znajduje się w środkowej, dolnej części płytki, obok matrycy klawiszy. Ma 3 wejścia, z których 2 użyjemy w tym przykładzie.

„NC” oznacza „normalnie zamknięty”, „NO” oznacza „normalnie otwarty”, a „COM” oznacza „wspólny”.
Wspólny w tym przypadku oznacza wspólną płaszczyznę.
Gdy obwód wspólny nie jest zasilany (GPIO.LOW), obwód „NC” jest zamknięty.
Kiedy wspólny obwód zostanie zasilony (GPIO.HIGH), przekaźnik zamknie obwód dla „NO”.
Używając „NO” i „COM” wszystko jest odwrócone.
Gdy „COM” jest wyłączony (GPIO.LOW), obwód przekaźnika jest otwarty.
Gdy „COM” jest włączony (GPIO.High), obwód przekaźnika jest zamknięty.

Uwaga! Bardzo ważne jest, aby nie próbować podłączać do przekaźnika urządzeń wysokiego napięcia (np. Lampy stołowej, ekspresu do kawy itp.). Może to spowodować porażenie prądem i poważne obrażenia.

Teraz, gdy już zrozumieliśmy, czym jest przekaźnik i jak działa, przyjrzyjmy się kodowi:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time

# define relay pin
relay_pin = 40

# set GPIO mode as GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
# setup relay pin as OUTPUT
GPIO.setup(relay_pin, GPIO.OUT)

# Open Relay
GPIO.output(relay_pin, GPIO.LOW)
# Wait half a second
time.sleep(0.5)
# Close Relay
GPIO.output(relay_pin, GPIO.HIGH)
GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python relay.py

5.4 LEKCJA 4: WYSYŁANIE SYGNAŁU WIBRACYJNEGO

Czy kiedykolwiek zastanawiałeś się, jak wibruje Twój telefon, gdy ktoś do Ciebie dzwoni lub kiedy otrzymujesz wiadomość?
Zbudowaliśmy dokładnie ten sam moduł w naszym Joy-Pi i teraz nauczymy się go używać.

Moduł wibracji znajduje się po prawej stronie matrycy LED i poniżej segmentowej diody LED. Kiedy jest włączony, trudno jest stwierdzić, skąd pochodzą wibracje, ponieważ wydaje się, że cała płyta Joy-Pi wibruje.

Moduł wibracyjny wykorzystuje sygnał GPIO.OUTPUT, podobnie jak Brzęczyk i inne używane wcześniej moduły. Po wysłaniu sygnału wyjściowego moduł zacznie wibrować. Kiedy zatrzymasz sygnał za pomocą GPIO.LOW, wibracje ustaną.

Możesz dostosować długość wibracji w różnych odstępach czasu.

W tym przykładzie musisz przełączać się między modułami. Ustaw przełącznik numer 1 prawej jednostki przełączającej w pozycji włączonej. Wszystkie pozostałe przełączniki powinny być wyłączone.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time

# define vibration pin
vibration_pin = 13

# Set board mode to GPIO.BOARD
GPIO.setmode(GPIO.BOARD)

# Setup vibration pin to OUTPUT
GPIO.setup(vibration_pin, GPIO.OUT)

# turn on vibration
GPIO.output(vibration_pin, GPIO.HIGH)
# wait 4 seconds
time.sleep(4)
# turn off vibration
GPIO.output(vibration_pin, GPIO.LOW)
# cleanup GPIO
GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python vibration.py

5.5 LEKCJA 5: WYKRYWANIE HAŁASU ZA POMOCĄ CZUJNIKA DŹWIĘKU

Podczas tej lekcji nauczymy się, jak używać czujnika dźwięku do wprowadzania danych, wykrywania głośnych dźwięków i odpowiedniego reagowania. Możesz więc zbudować własny system alarmowy, który wykrywa głośne dźwięki lub włączać diodę LED, klaszcząc!

Czujnik dźwięku składa się z dwóch części: niebieskiego potencjometru, który reguluje czułość, oraz samego czujnika, który wykrywa wejście dźwięków. Czujnik dźwięku można łatwo rozpoznać po niebieskim potencjometrze, a sam czujnik znajduje się po prawej stronie pod brzęczykiem.

Za pomocą potencjometru możemy regulować czułość czujnika. Aby nasz skrypt zadziałał, musimy najpierw nauczyć się kontrolować wrażliwość. Aby wyregulować czułość, musisz przekręcić małą śrubkę na potencjometrze za pomocą śrubokręta w lewo lub w prawo. Najlepszym sposobem sprawdzenia czułości jest uruchomienie skryptu. Klaszcz w dłonie i sprawdź, czy urządzenie odbiera sygnał. Brak sygnału oznacza, że czułość czujnika nie jest wystarczająco wysoka. Można to łatwo skorygować, obracając potencjometr.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time

# define sound pin
sound_pin = 18
# set GPIO mode to GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
# setup pin as INPUT
GPIO.setup(sound_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
    while True:
        # check if sound detected or not
        if(GPIO.input(sound_pin)==GPIO.LOW):
            print('Sound Detected')
            time.sleep(0.1)
except KeyboardInterrupt:
    # CTRL+C detected, cleaning and quitting the script
    GPIO.cleanup()
```

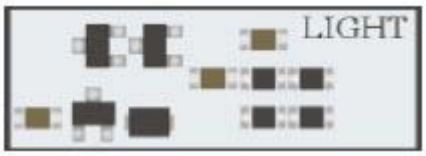
Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python sound.py

Najpierw definiujemy nasz pin, GPIO18. Następnie ustawiliśmy pętlę while, aby trwale uruchomić ten skrypt. Sprawdzamy, czy otrzymaliśmy sygnał wejściowy z czujnika dźwięku wskazujący, że zostały wykryte głośne dźwięki, a następnie drukujemy „Wykryto dźwięk". Jeśli naciśnięto Ctrl + C, program kończy pracę.

5.6 LEKCJA 6: WYKRYWANIE JASNOŚCI CZUJNIKIEM ŚWIATŁA

Czujnik światła jest jednym z naszych ulubionych. Jest niezwykle przydatny w wielu projektach i sytuacjach, np. z lampami, które włączają się automatycznie, gdy tylko się ściemni. Dzięki czujnikowi światła możemy zobaczyć, jak jasna jest powierzchnia modułu.



Czujnik światła jest trudny do wykrycia, ponieważ składa się z bardzo małych części. Czujnik znajduje się po lewej stronie brzęczyka. Jeśli zakryjesz go palcem, wyjście czujnika światła powinno być bliskie zeru, ponieważ żadne światło nie może do niego dotrzeć.

Czas przetestować to w czasie rzeczywistym i zobaczyć, jak to działa. Jednak czujnik światła różni się trochę od innych czujników, ponieważ działa z I2C, a nie z normalnymi GPIO, czego nauczyliśmy się na wcześniejszych lekcjach.

W tym skrypcie używamy funkcji do komunikacji z czujnikiem, w ten sposób możemy uzyskać jasność. Im wyższa liczba, tym wyższe otoczenie.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author: Matt Hawkins
# Author's Git: https://bitbucket.org/MattHawkinsUK/
# Author's website: https://www.raspberrypi-spy.co.uk

import RPi.GPIO as GPIO
import smbus
import time

 # Find the right revision for bus driver
if(GPIO.RPI_REVISION == 1):
    bus = smbus.SMBus(0)
else:
    bus = smbus.SMBus(1)

class LightSensor():

    def __init__(self):

        # Define some constants from the datasheet

        self.DEVICE = 0x5c # Default device I2C address

        self.POWER_DOWN = 0x00 # No active state
        self.POWER_ON = 0x01 # Power on
        self.RESET = 0x07 # Reset data register value

        # Start measurement at 4lx resolution. Time typically 16ms.
        self.CONTINUOUS_LOW_RES_MODE = 0x13
        # Start measurement at 1lx resolution. Time typically 120ms
        self.CONTINUOUS_HIGH_RES_MODE_1 = 0x10
        # Start measurement at 0.5lx resolution. Time typically 120ms
        self.CONTINUOUS_HIGH_RES_MODE_2 = 0x11
        # Start measurement at 1lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_HIGH_RES_MODE_1 = 0x20
        # Start measurement at 0.5lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_HIGH_RES_MODE_2 = 0x21
```

```
        # Start measurement at 1lx resolution. Time typically 120ms
        # Device is automatically set to Power Down after measurement.
        self.ONE_TIME_LOW_RES_MODE = 0x23

    def convertToNumber(self, data):

        # Simple function to convert 2 bytes of data
        # into a decimal number
        return ((data[1] + (256 * data[0])) / 1.2)

    def readLight(self):

        data = bus.read_i2c_block_data
(self.DEVICE,self.ONE_TIME_HIGH_RES_MODE_1)
        return self.convertToNumber(data)

def main():

    sensor = LightSensor()
    try:
        while True:
            print "Light Level : " + str(sensor.readLight()) + " lx"
            time.sleep(0.5)
    except KeyboardInterrupt:
        pass

if __name__ == "__main__":
            main()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python light_sensor.py

5.7 LEKCJA 7: WYKRYWANIE TEMPERATURY I WILGOTNOŚCI

DHT11 to bardzo ciekawy czujnik, ponieważ ma nie tylko jedną funkcję, ale dwie! Zawiera zarówno czujnik wilgotności, jak i czujnik temperatury, z których oba są bardzo dokładne. Idealny do każdego projektu stacji pogodowej lub jeśli chcesz sprawdzić temperaturę i wilgotność w pomieszczeniu!

Czujnik DHT11 jest bardzo łatwy do rozpoznania. Mały niebieski czujnik z wieloma małymi otworami. Znajduje się po prawej stronie przekaźnika i nad czujnikiem dotykowym. Praca z czujnikiem DHT11 jest bardzo łatwa dzięki bibliotece Adafruit_DHT. Biblioteka służy do wyświetlania temperatury i wilgotności jako wartości bez konieczności wykonywania skomplikowanych obliczeń matematycznych.

```
#!/usr/bin/python
# Copyright (c) 2014 Adafruit Industries
# Author: Tony DiCola

import sys
import Adafruit_DHT

# set type of the sensor
sensor = 11
# set pin number
pin = 4

# Try to grab a sensor reading.  Use the read_retry method which will retry up
# to 15 times to get a sensor reading (waiting 2 seconds between each retry).
humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)

# Un-comment the line below to convert the temperature to Fahrenheit.
# temperature = temperature * 9/5.0 + 32

# Note that sometimes you won't get a reading and
# the results will be null (because Linux can't
# guarantee the timing of calls to read the sensor).
# If this happens try again!
if humidity is not None and temperature is not None:
    print('Temp={0:0.1f}*  Humidity={1:0.1f}%'.format(temperature, humidity))
else:
    print('Failed to get reading. Try again!')
sys.exit(1)
```

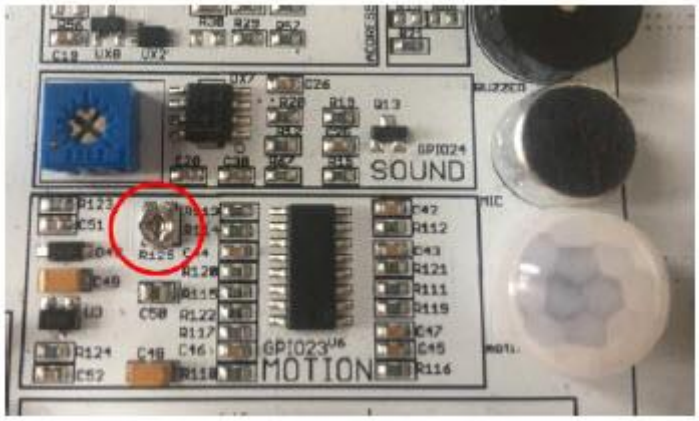
Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python dht11.py

5.8 LEKCJA 8: WYKRYWANIE RUCHÓW

Czujnik ruchu jest jednym z najbardziej przydatnych i najczęściej używanych czujników. Można go wykorzystać np. Do budowy systemu alarmowego. Gdy czujnik wykryje ruch, może wysłać sygnał do brzęczyka, który następnie uruchomi głośny alarm.

Czujnik ruchu znajduje się bezpośrednio pod czujnikiem dźwięku i jest zakryty małą, przezroczystą nasadką. Nasadka pomaga czujnikowi wykryć więcej ruchów poprzez załamanie światła podczerwonego otoczenia. Czułość czujnika ruchu, podobnie jak czujnika dźwięku, regulowana jest potencjometrem. Znajduje się on poniżej potencjometru czujnika dźwięku, ale jest znacznie mniejszy. Za pomocą śrubokręta można ustawić odległości, na jakie ma reagować czujnik ruchu.



Czujnik ruchu sterowany jest pinami GPIO. Po wykryciu ruchu czujnik ruchu wyśle sygnał. To zatrzyma się na jakiś czas, a następnie zatrzyma się ponownie, aż czujnik wykryje następny ruch.

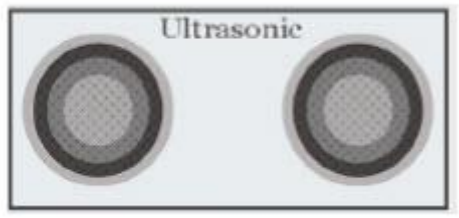
```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time

# define motion pin
motion_pin = 16

# set GPIO as GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
# set pin mode as INPUT
GPIO.setup(motion_pin, GPIO.IN)

try:
    while True:
       if(GPIO.input(motion_pin) == 0):
             print "Nothing moves ..."
       elif(GPIO.input(motion_pin) == 1):
             print "Motion detected!"
       time.sleep(0.1)
except KeyboardInterrupt:
    GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python motion.py

5.9 LEKCJA 9: POMIAR ODLEGŁOŚCI CZUJNIKIEM ULTRADŹWIĘKOWYM

Teraz nauczymy się, jak używać czujnika ultradźwiękowego do mierzenia odległości i wyświetlania ich na ekranie Joy-Pi. Nawiasem mówiąc, samochody używają tej samej metody do pomiaru odległości.



Czujnik ultradźwiękowy znajduje się w prawym dolnym rogu płytki Joy-Pi, bezpośrednio nad silnikiem krokowym i interfejsami serwomechanizmu. Można go łatwo rozpoznać po dwóch dużych kołach. Przesuniemy dłonie po czujniku odległości, aby zmierzyć odległość między naszymi rękami a Joy-Pi.

Czujnik odległości współpracuje z GPIO INPUT, ale różni się nieco od tego, którego używaliśmy na poprzednich lekcjach. Czujnik potrzebuje określonego odstępu czasu, aby móc dokładnie wykryć odległość. Wysyła sygnał ultradźwiękowy, a dzięki wbudowanemu czujnikowi odbiera echo odbite od przeszkody. Na podstawie różnicy czasu między wysłaniem sygnału a odebraniem echa obliczana jest odległość.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : www.modmypi.com
# Link: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)

TRIG = 36
ECHO = 32   #Declare variables

print "Distance Measurement In Progress"  #Console output

GPIO.setup(TRIG,GPIO.OUT) #Using TRIG as output
GPIO.setup(ECHO,GPIO.IN)  #Using ECHO as Input

GPIO.output(TRIG, False)
print "Waiting For Sensor To Settle" #Console output
time.sleep(2)  #Wait 2 seconds

GPIO.output(TRIG, True)  #Start sending a signal
time.sleep(0.00001)      #Wait for 0.00001 seconds
GPIO.output(TRIG, False) #Stop sending a Signal

while GPIO.input(ECHO)==0:
  pulse_start = time.time()

while GPIO.input(ECHO)==1:
  pulse_end = time.time()

pulse_duration = pulse_end - pulse_start #measurement for distance

distance = pulse_duration * 17150 #Calculation for distance

distance = round(distance, 2) #rounded to 2 decimal places

print "Distance:",distance,"cm" #Output distance in console

GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

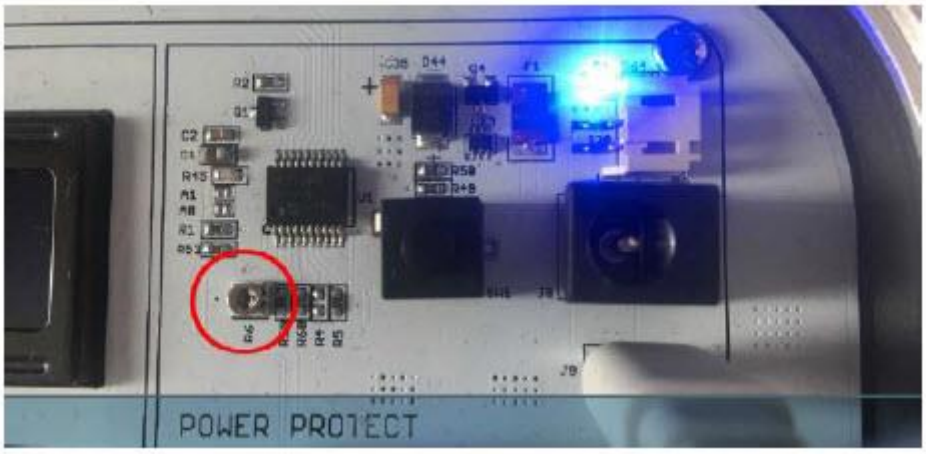
cd /home/pi/Desktop/Joy-Pi/
sudo python distance.py

5.10 LEKCJA 10: STEROWANIE WYŚWIETLACZEM LCD

Dzięki Joy-Pi możesz wyświetlać dane LCD, które zbierasz za pomocą czujników i aktualizować je w czasie rzeczywistym w zależności od zmian, przez które przechodzą moduły. Na przykład w połączeniu z czujnikiem temperatury - zawsze wyświetlaj aktualną temperaturę i wilgotność na wyświetlaczu LCD.

Ekran LCD zajmuje dużą część płyty Joy-Pi - znajduje się w górnej środkowej części Joy-Pi, po prawej stronie wyświetlacza GPIO LED. Gdy tylko skrypt demonstracyjny i przykłady zostaną wykonane, wyświetlacz włączy się. Dzięki zintegrowanemu podświetleniu można odczytać dane na wyświetlaczu nawet w całkowitej ciemności.

Podobnie jak czujniki dźwięku i ruchu, wyświetlacz LCD ma również powiązany potencjometr. Za pomocą tego potencjometru można regulować jasność podświetlenia wyświetlacza. Jeśli obrócisz go w kierunku przeciwnym do ruchu wskazówek zegara, jasność wzrośnie, a jeśli obrócisz go zgodnie z ruchem wskazówek zegara, zmniejszy się. Obróć potencjometr przeciwnie do ruchu wskazówek zegara, aby zwiększyć kontrast, obróć go w prawo, aby zmniejszyć kontrast.



Wyświetlacz LCD i niektóre inne czujniki nie współpracują z technologią GPIO. Dlatego używamy „I2C”. Używamy adresu 21 dla wyświetlacza LCD, ustanawiając połączenie z tym adresem I2C. Możemy więc wysyłać polecenia takie jak pisanie tekstu, włączanie podświetlenia LCD, aktywowanie kursora itp. Do sterowania wyświetlaniem używamy biblioteki Adafruit_CharLCDBackpack.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Example using a character LCD backpack.
import time
import Adafruit_CharLCD as LCD

# Define LCD column and row size for 16x2 LCD.
lcd_columns = 16
lcd_rows    = 2

# Initialize the LCD using the pins
lcd = LCD.Adafruit_CharLCDBackpack(address=0x21)

try:
    # Turn backlight on
    lcd.set_backlight(0)

    # Print a two line message
    lcd.message('Hello\nworld!')

    # Wait 5 seconds
    time.sleep(5.0)

    # Demo showing the cursor.
    lcd.clear()
    lcd.show_cursor(True)
    lcd.message('Show cursor')

    time.sleep(5.0)

    # Demo showing the blinking cursor.
    lcd.clear()
    lcd.blink(True)
    lcd.message('Blink cursor')

    time.sleep(5.0)

    # Stop blinking and showing cursor.
    lcd.show_cursor(False)
    lcd.blink(False)

    # Demo scrolling message right/left.
    lcd.clear()
```

```
    message = 'Scroll'
    lcd.message(message)
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_right()
    for i in range(lcd_columns-len(message)):
        time.sleep(0.5)
        lcd.move_left()

    # Demo turning backlight off and on.
    lcd.clear()
    lcd.message('Flash backlight\nin 5 seconds...')
    time.sleep(5.0)
    # Turn backlight off.
    lcd.set_backlight(1)
    time.sleep(2.0)
    # Change message.
    lcd.clear()
    lcd.message('Goodbye!')
    # Turn backlight on.
    lcd.set_backlight(0)
    # Turn backlight off.
    time.sleep(2.0)
    lcd.clear()
    lcd.set_backlight(1)
except KeyboardInterrupt:
    # Turn the screen off
    lcd.clear()
            lcd.set_backlight(1)
```

Do sterowania LCD używamy biblioteki Adafruit_CharLCDBackpack.

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python lcd.py

5.11 LEKCJA 11: CZYTANIE I PISANIE KART RFID

W tej lekcji dowiesz się, jak sterować modułem RFID. Moduł RFID to bardzo ciekawy i przydatny moduł. Znajduje zastosowanie na całym świecie w różnych rozwiązaniach, takich jak: inteligentne zamki do drzwi, identyfikatory pracowników, wizytówki, a nawet obroże dla psów.

Moduł RFID znajduje się bezpośrednio pod Raspberry Pi i wygląda jak mały symbol Wifi. Ten symbol oznacza łączność bezprzewodową. Aby z niego skorzystać, musimy wziąć chip lub kartę, która jest dostarczana z Joy-Pi i przytrzymać ją nad obszarem chipa RFID Joy-Pi. Musi być wystarczająco blisko, aby nasz skrypt został rozpoznany. 2-4 cm powinno być wystarczająco blisko. Po prostu wypróbuj!

Aby korzystać z osłony RFID RC522, potrzebujemy magistrali SPI. Musimy zmodyfikować plik konfiguracyjny, w przeciwnym razie jądro nie mogłoby się uruchomić, aby uzyskać dostęp do pliku konfiguracyjnego, używamy następującego polecenia:

sudo nano /boot/config.txt

Na końcu pliku należy dołączyć następujące wiersze:

device_tree_param=spi=on
dtoverlay=spi-bcm2708

Zapisujemy plik CTRL + O i wciskamy Enter, po pliku możemy zamknąć edytor CTRL + X. Na koniec musimy aktywować SPI, więc używamy następującego polecenia, aby zmodyfikować ustawienia:

sudo raspi-config

Teraz przechodzimy do „Interfacing options”, następnie aktywujemy „SPI” i klikamy „OK”, restartujemy Raspberry pi i część konfiguracyjna modułu RFID jest zakończona.

Aby przejść do folderu ze skryptami RFID, musisz użyć następującego polecenia:

cd /home/pi/Desktop/Joy-Pi/MFRC522-python

Jeśli chcesz pisać na chipie lub karcie, możesz użyć następującego polecenia:

sudo python Write.py

Możesz zmienić dane, które są zapisywane na chipie lub karcie RFID, zmieniając kod programu:

```
# Select the scanned tag
MIFAREReader.MFRC522_SelectTag(uid)

# Authenticate
status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
print "\n"

# Check if authenticated
if status == MIFAREReader.MI_OK:

    # Variable for the data to write
    data = [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166, 188]

    # Fill the data with 0xFF
    for x in range(0,16):
        data.append(0xFF)
```

Aby zmodyfikować dane, musisz zmienić kolejność liczb w nawiasach kwadratowych, ale liczby nie mogą być mniejsze od 0 ani powyżej 255.
Jeśli chcesz odczytać sekwencję liczb, musisz użyć następującego polecenia:

sudo python Read.py

Teraz możesz położyć chip lub kartę na czytniku RFID, a wyświetli się coś takiego:

```
Card detected
Card read UID: 107,144,78,115
Size: 8
Sector 8 [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166,
188]
```

Sekwencja liczb obok sektora 8 to ta, którą zapisaliśmy teraz na chipie lub karcie.

Przykładowy kod RFID-Read:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
import RPi.GPIO as GPIO
import MFRC522
import signal
continue_reading = True
# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print "Ctrl+C captured, ending read."
    continue_reading = False
    GPIO.cleanup()
# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)
# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()
# Welcome message
print "Welcome to the MFRC522 data read example"
print "Press Ctrl-C to stop."
# This loop keeps checking for chips.
# If one is near it will get the UID and authenticate
while continue_reading:
    # Scan for cards
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)
    # If a card is found
    if status == MIFAREReader.MI_OK:
        print "Card detected"
    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()
    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:
        # Print UID
        print "Card read UID: %s,%s,%s,%s" % (uid[0], uid[1], uid[2], uid[3])
        # This is the default key for authentication
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]
        # Select the scanned tag
        MIFAREReader.MFRC522_SelectTag(uid)
        # Authenticate
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
        # Check if authenticated
        if status == MIFAREReader.MI_OK:
            MIFAREReader.MFRC522_Read(8)
            MIFAREReader.MFRC522_StopCrypto1()
        else:
            print "Authentication error"
```

Przykładowy kod RFID-Write:

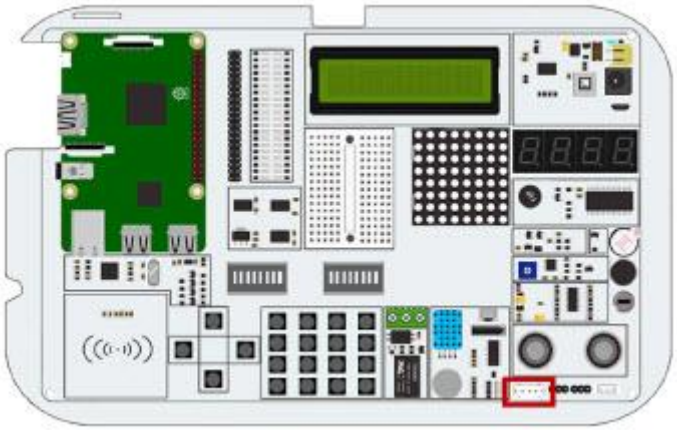
```
#!/usr/bin/env python
# -*- coding: utf8 -*-
import RPi.GPIO as GPIO
import MFRC522
import signal

continue_reading = True

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print "Ctrl+C captured, ending read."
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# This loop keeps checking for chips. If one is near it will get the UID and au-
thenticate
while continue_reading:

    # Scan for cards
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        print "Card detected"

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:

        # Print UID
        print "Card read UID: %s,%s,%s,%s" % (uid[0], uid[1], uid[2], uid[3])

        # This is the default key for authentication
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

        # Select the scanned tag
        MIFAREReader.MFRC522_SelectTag(uid)
```

Kontynuacja kodu RFID-Write:

```
# Authenticate
status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
print "\n"
# Check if authenticated
if status == MIFAREReader.MI_OK:

    # Variable for the data to write
    data = [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166, 188]

    # Fill the data with 0xFF
    for x in range(0,16):
        data.append(0xFF)

    print "Sector 8 looked like this:"
    # Read block 8
    MIFAREReader.MFRC522_Read(8)
    print "\n"

    print "Sector 8 will now be filled with 0xFF:"
    # Write the data
    MIFAREReader.MFRC522_Write(8, data)
    print "\n"

    print "It now looks like this:"
    # Check to see if it was written
    MIFAREReader.MFRC522_Read(8)
    print "\n"

    # Stop
    MIFAREReader.MFRC522_StopCrypto1()

    # Make sure to stop reading for cards
    continue_reading = False
else:
    print "Authentication error"
```

5.12 LEKCJA 12: UŻYWANIE SILNIKÓW KROKOWYCH

Silnik krokowy to niezależny moduł, który będziesz musiał podłączyć do płytki. Musimy wziąć silnik krokowy dostarczony z zestawem i podłączyć go do naszego Joy-Pi. Wystarczy podłączyć silnik krokowy do następującego złącza na płycie Joy-Pi:



Moduł może się nagrzewać podczas użytkowania. Wynika to z przyczyn technicznych i nie jest niczym niezwykłym.

W tym przykładzie musisz przełączać się między modułami. Ustawić przełączniki 3, 4, 5 i 6 na prawej jednostce przełączającej w położenie włączone. Wszystkie pozostałe przełączniki powinny być wyłączone.

Silnik krokowy podłączony jest do 4 pinów GPIO, które włączają się szybko jeden po drugim. Powoduje to, że silnik krokowy „popycha” do przodu i robi jeden krok. Za pomocą funkcji turnSteps można wykonać dowolną liczbę kroków. Funkcja turnDegrees obraca silnik o określony kąt.

Przykładowy kod można znaleźć na następnej stronie.

Przykładowy kod silnika krokowego:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : Original author ludwigschuster
# Original Author Github: https://github.com/ludwigschuster/RasPi-GPIO-
Stepmotor
import time
import RPi.GPIO as GPIO
import math
class Stepmotor:
      def __init__(self):
            # set GPIO mode
            GPIO.setmode(GPIO.BOARD)
            # These are the pins which will be used on the Raspberry Pi
            self.pin_A = 29
            self.pin_B = 31
            self.pin_C = 33
            self.pin_D = 35
            self.interval = 0.010
            # Declare pins as output
            GPIO.setup(self.pin_A,GPIO.OUT)
            GPIO.setup(self.pin_B,GPIO.OUT)
            GPIO.setup(self.pin_C,GPIO.OUT)
            GPIO.setup(self.pin_D,GPIO.OUT)
            GPIO.output(self.pin_A, False)
            GPIO.output(self.pin_B, False)
            GPIO.output(self.pin_C, False)
            GPIO.output(self.pin_D, False)
      def Step1(self):
            GPIO.output(self.pin_D, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_D, False)
      def Step2(self):
            GPIO.output(self.pin_D, True)
            GPIO.output(self.pin_C, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_D, False)
            GPIO.output(self.pin_C, False)
      def Step3(self):
            GPIO.output(self.pin_C, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_C, False)
      def Step4(self):
            GPIO.output(self.pin_B, True)
            GPIO.output(self.pin_C, True)
```

```
            time.sleep(self.interval)
            GPIO.output(self.pin_B, False)
            GPIO.output(self.pin_C, False)

      def Step5(self):

            GPIO.output(self.pin_B, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_B, False)

      def Step6(self):

            GPIO.output(self.pin_A, True)
            GPIO.output(self.pin_B, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_A, False)
            GPIO.output(self.pin_B, False)

      def Step7(self):

            GPIO.output(self.pin_A, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_A, False)

      def Step8(self):

            GPIO.output(self.pin_D, True)
            GPIO.output(self.pin_A, True)
            time.sleep(self.interval)
            GPIO.output(self.pin_D, False)
            GPIO.output(self.pin_A, False)

      def turn(self,count):
            for i in range (int(count)):
                  self.Step1()
                  self.Step2()
                  self.Step3()
                  self.Step4()
                  self.Step5()
                  self.Step6()
                  self.Step7()
                  self.Step8()

      def close(self):
            # cleanup the GPIO pin use
            GPIO.cleanup()

      def turnSteps(self, count):
            # Turn n steps
            # (supply with number of steps to turn)
            for i in range (count):
                  self.turn(1)
```

```
    def turnDegrees(self, count):
        # Turn n degrees (small values can lead to inaccuracy)
        # (supply with degrees to turn)
        self.turn(round(count*512/360,0))

    def turnDistance(self, dist, rad):
        # Turn for translation of wheels or coil (inaccuracies involved
        # e.g. due to thickness of rope)
        # (supply with distance to move and radius in same metric)
        self.turn(round(512*dist/(2*math.pi*rad),0))

def main():

    print("moving started")
    motor = Stepmotor()
    print("One Step")
    motor.turnSteps(1)
    time.sleep(0.5)
    print("20 Steps")
    motor.turnSteps(20)
    time.sleep(0.5)
    print("quarter turn")
    motor.turnDegrees(90)
    print("moving stopped")
    motor.close()

if __name__ == "__main__":
            main()
```

Wykonaj następujące polecenia i wypróbuj sam:

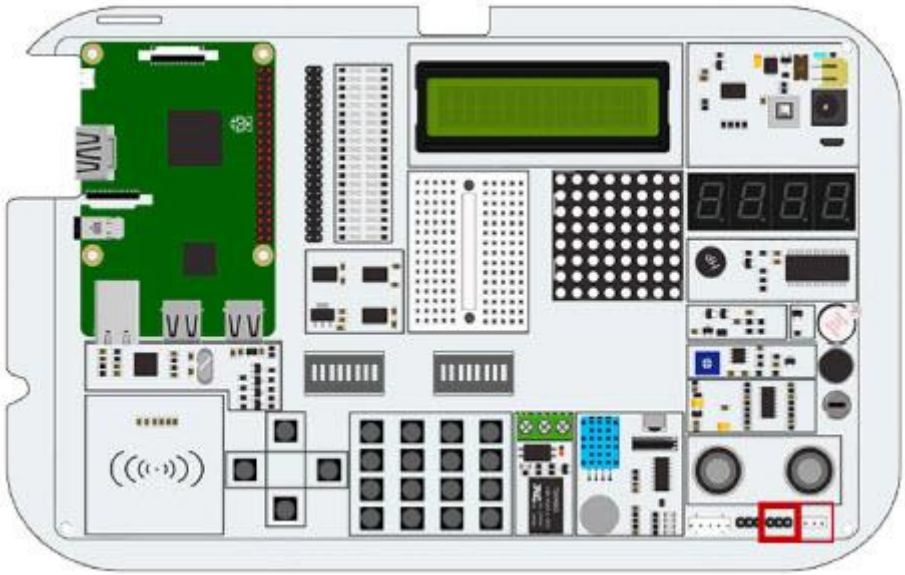
cd /home/pi/Desktop/Joy-Pi/
sudo python stepmotor.py

5.13 LEKCJA 13: STEROWANIE SIŁOWNIKAMI

Za pomocą serwomotoru można mechanicznie sterować urządzeniami i przemieszczać części. Na przykład można stworzyć inteligentne kosze na śmieci, skrzynię z inteligentnym otwieraniem i zamykaniem drzwi oraz wiele innych projektów wypoczynkowych.

Joy-Pi ma dwa interfejsy serwo, z których oba mogą być używane do sterowania silnikami serwo. W tym samouczku użyjemy drugiego interfejsu, który jest oznaczony jako „Servo2”. Oczywiście możesz również użyć innego interfejsu serwomechanizmu, ale musisz w tym celu dostosować skrypt do odpowiednich GPIO.

Serwomotor potrzebuje trzech pinów: dodatniego, ujemnego i pinowego danych. Dodatni pin to czerwony kabel, ujemny pin to czarny kabel (zwany także uziemieniem), a kabel danych jest żółty.



W tym przykładzie musisz przełączać się między modułami. Ustaw przełączniki nr 7 i 8 na prawej jednostce przełączającej w położenie włączone. Wszystkie pozostałe przełączniki powinny być wyłączone.

Kabel	PIN
Czerwony	Środkowy pin Servo2
Czarny	Prawy pin Servo2
Kolorowy	Lewy pin Servo2

Przyjrzyjmy się naszemu przykładowemu kodowi, aby lepiej go zrozumieć: Serwo używa styku GPIO.board numer 22. Za każdym razem skrypt ustawi kierunek obrotów serwomotoru. Możemy użyć dodatnich stopni, aby obrócić w lewo i ujemnych stopni, aby obrócić się w prawo. Po prostu zmień stopnie i zobacz, jak zmienia się obrót silnika.

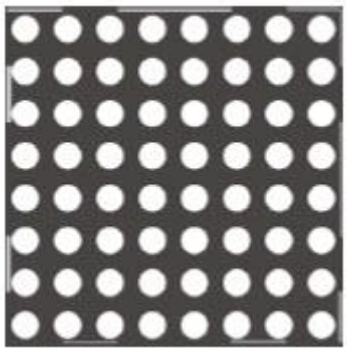
```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : Original author WindVoiceVox
# Original Author Github: https://github.com/WindVoiceVox/Raspi_SG90

import RPi.GPIO as GPIO
import time
import sys                  #Import librarys

class sg90:

  def __init__( self, pin, direction ):

    GPIO.setmode( GPIO.BOARD )        #set pinlayout to GPIO.BOARD
    GPIO.setup( pin, GPIO.OUT )       #declare output
    self.pin = int( pin )
    self.direction = int( direction )
    self.servo = GPIO.PWM( self.pin, 50 )
    self.servo.start(0.0)

  def cleanup( self ):

    self.servo.ChangeDutyCycle(self._henkan(0))
    time.sleep(0.3)
    self.servo.stop()                 # stop servomotor
    GPIO.cleanup()                    #Clean GPIOs for other uses

  def currentdirection( self ):

    return self.direction

  def _henkan( self, value ):

    return 0.05 * value + 7.0
```

```
  def setdirection( self, direction, speed ):

    for d in range( self.direction, direction, int(speed) ):
      self.servo.ChangeDutyCycle( self._henkan( d ) )
      self.direction = d
      time.sleep(0.1)
    self.servo.ChangeDutyCycle( self._henkan( direction ) )
    self.direction = direction

def main():

    servo_pin = 22              #give servo_pin GPIO.BOARD pin 22
    s = sg90(servo_pin,0)

    try:
        while True:
            print "Turn left ..."          #console output
            s.setdirection( 100, 10 )
            time.sleep(0.5)                #wait 0.5 seconds
            print "Turn right ..."
            s.setdirection( -100, -10 )
            time.sleep(0.5)                #wait 0.5 seconds
    except KeyboardInterrupt:
        s.cleanup()

if __name__ == "__main__":
              main()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python servo.py

5.14 LEKCJA 14: STEROWANIE MATRYCĄ LED 8X8



Matryca LED odgrywa ważną rolę w wielu projektach z migającymi diodami LED. Nawet jeśli nie widać tego na pierwszy rzut oka, matryca LED potrafi znacznie więcej niż tylko mruganie na czerwono. Może być używany do wyświetlania informacji, tekstu, emotikonów, a nawet chińskich znaków. Idealny do wyświetlania informacji w zabawny i wyjątkowy sposób, a może nawet do gry takiej jak Snake lub minutnik!

Moduł matrycy LED to duży kwadratowy moduł umieszczony po lewej stronie segmentowej diody LED i tuż pod wyświetlaczem LCD. Można go łatwo rozpoznać po małych białych kropkach, które są diodami LED. Nie daj się zwieść niewielkiemu rozmiarowi diod LED. Ta matryca LED z łatwością oświetli ciemne miejsce!

W tym przykładzie wyświetlamy krótki tekst. W skrypcie tworzymy ciąg z komunikatem i używamy funkcji show_message () do wyświetlenia komunikatu na wyświetlaczu macierzy.

Możemy kontrolować właściwości, takie jak opóźnienia, które sprawiają, że wiadomość jest szybsza lub wolniejsza. Na przykład scroll_delay 0 będzie dość szybki, a opóźnienie 0.1 spowoduje, że przepływ komunikatów nieco się spowolni. Matrix LED, w przeciwieństwie do innych modułów, wykorzystuje interfejs SPI, z którego można nią sterować. Wypróbuj kilka przykładów i zmień kod, aby zobaczyć, co się stanie.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Copyright (c) 2017-18 Richard Hull and contributors
# License: https://github.com/rm-hull/luma.led_matrix/blob/master/LICENSE.rst
# Github link: https://github.com/rm-hull/luma.led_matrix/

# Import all the modules
import re
import time
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT, SIN-
CLAIR_FONT, LCD_FONT
```

```
def main(cascaded, block_orientation, rotate):

    # create matrix device
    serial = spi(port=0, device=1, gpio=noop())
    device = max7219(serial, cascaded=cascaded or 1,
block_orientation=block_orientation, rotate=rotate or 0)
    # debugging purpose
    print("[-] Matrix initialized")

    # print hello world on the matrix display
    msg = "HELLO WORLD"
    # debugging purpose
    print("[-] Printing: %s" % msg)
    show_message(device, msg, fill="white", font=proportional(CP437_FONT),
scroll_delay=0.1)


if __name__ == "__main__":

    # cascaded = Number of cascaded MAX7219 LED matrices, default=1
    # block_orientation = choices 0, 90, -90, Corrects block orientation when
wired vertically, default=0
    # rotate = choices 0, 1, 2, 3, Rotate display 0=0°, 1=90°, 2=180°, 3=270°,
default=0

    try:
        main(cascaded=1, block_orientation=90
, rotate=0)
    except KeyboardInterrupt:
                    pass
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python matrix_demo.py

5.15 LEKCJA 15: STEROWANIE 7-SEGMENTOWYM WYŚWIETLACZEM

Segmentowa dioda LED jest bardzo użytecznym wyświetlaczem, jeśli chodzi o liczby i dane. Może nam pokazać czas, policzyć, ile razy zrobiliśmy pewne rzeczy. Wyświetlacz segmentowy znajduje również zastosowanie w wielu rozwiązaniach przemysłowych, np. W windach.

Wyświetlacz segmentowy znajduje się bezpośrednio nad czujnikiem drgań i obok matrycy LED. Gdy jest wyłączony, widoczne są 4 ósemki. Gdy tylko aktywujesz moduł wyświetlacza segmentowego, ciemny kolor zmieni się w błyszczący, jasnoczerwony.

W naszym przykładzie pokazujemy zegar. Użyjemy modułów czasu i daty, aby uzyskać czas systemowy Raspberry Pi, który wyświetlamy za pomocą funkcji segment.write_display (). Funkcja set_digit () w połączeniu z liczbami 0, 1, 2 i 3, ustawia pozycję na wyświetlaczu, na której ma się pojawić liczba.

Ponieważ w tym przykładzie pobierany jest bieżący czas systemowy, konieczne jest najpierw skonfigurowanie Raspberry Pi na odpowiednią strefę czasową. Otwórz okno terminala i wprowadź następujące polecenie:

sudo dpkg-reconfigure tzdata

Otworzy się okno, w którym możesz wybrać aktualną strefę czasową. Po wybraniu właściwej strefy czasowej potwierdź przyciskiem OK i ponownie naciśnij Enter, aby potwierdzić.

```
#!/usr/bin/python

import time
import datetime
from Adafruit_LED_Backpack import SevenSegment

# ===========================================================================
# Clock Example
# ===========================================================================
segment = SevenSegment.SevenSegment(address=0x70)

# Initialize the display. Must be called once before using the display.
segment.begin()
print "Press CTRL+C to exit"
# Continually update the time on a 4 char, 7-segment display
try:
  while(True):
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second

    segment.clear()
    # Set hours
    segment.set_digit(0, int(hour / 10))     # Tens
    segment.set_digit(1, hour % 10)          # Ones
    # Set minutes
    segment.set_digit(2, int(minute / 10))   # Tens
    segment.set_digit(3, minute % 10)        # Ones
    # Toggle colon
    segment.set_colon(second % 2)              # Toggle colon at 1Hz

    # Write the display buffer to the hardware.  This must be called to
    # update the actual display LEDs.
    segment.write_display()

    # Wait a second.
    time.sleep(1)
except KeyboardInterrupt:
    segment.clear()
            segment.write_display()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python segment.py

5.16 LEKCJA 16: ROZPOZNAWANIE ODCISKÓW PALCÓW

Czujnik dotykowy jest bardzo przydatny, jeśli chodzi o kluczowe funkcje. Wiele produktów na rynku wykorzystuje dotyk zamiast naciskania przycisku, na przykład smartfony i tablety. Czujnik dotykowy znajduje się bezpośrednio pod czujnikiem DHT11 i po prawej stronie przekaźnika. Łatwo dostępne pozycjonowanie Joy-Pi umożliwia łatwą obsługę.

Czujnik dotykowy działa jak każdy inny kluczowy moduł. Jedyną różnicą jest to, że wystarczy go dotknąć, a nie nacisnąć. Poprzez dotknięcie czujnika dotykowego moduł zamyka obwód, ponieważ komputer wykrywa dotknięcie czujnika. Czujnik dotykowy wykorzystuje pin 11 płytki GPIO.

```
from RPi import GPIO
import signal

TOUCH = 11
def setup_gpio():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TOUCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def do_smt(channel):
    print("Touch wurde erkannt")

def main():
    setup_gpio()
    try:
        GPIO.add_event_detect(TOUCH, GPIO.FALLING, callback=do_smt, bounceti-
me=200)
        signal.pause()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()

if __name__ == '__main__':
            main()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python touch.py

5.17 LEKCJA 17: WYKRYWANIE POCHYLENIA ZA POMOCĄ CZUJNIKA POCHYLENIA

Czujnik nachylenia pozwala nam wykryć przechylenie w prawo lub w lewo. Jest używany w robotyce i innych gałęziach przemysłu, aby zapewnić prostą obsługę. To mały, wydłużony, czarny czujnik, który znajduje się między czujnikiem DHT11 a czujnikiem ultradźwiękowym i można go łatwo wykryć po dźwięku, jaki wydaje, gdy lekko przechylamy deskę.

Możesz łatwo pomyśleć, że coś wewnątrz płytki Joy-Pi jest uszkodzone, gdy usłyszysz ten dźwięk, ale ten hałas jest całkowicie normalny. Kiedy czujnik przechyłu jest przechylony w lewo, obwód jest aktywowany i wysyłany jest sygnał GPIO HIGH. Jeśli czujnik przechyłu jest przechylony w prawo, obwód jest dezaktywowany i wysyłany jest sygnał GPIO LOW.

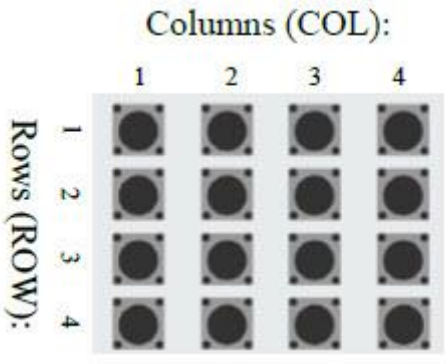
W tym przykładzie musisz przełączać się między modułami. Ustaw przełącznik nr 2 prawej jednostki przełączającej w pozycji włączonej. Wszystkie pozostałe przełączniki powinny być wyłączone.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import RPi.GPIO as GPIO

# define tilt pin
tilt_pin = 15

# set GPIO mode to GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
# set pin as input
GPIO.setup(tilt_pin, GPIO.IN)

try:
    while True:
        # positive is tilt to left negative is tilt to right
        if GPIO.input(tilt_pin):
            print "[-] Left Tilt"
        else:
            print "[-] Right Tilt"
        time.sleep(1)
except KeyboardInterrupt:
    # CTRL+C detected, cleaning and quitting the script
            GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python tilt.py

5.18 LEKCJA 18: UŻYWANIE MATRYCY PRZYCISKÓW



Kolumny (COL) Rzędy (ROW)

Matryca przycisków to moduł z 16 niezależnymi przyciskami, który można wykorzystać do wielu projektów, takich jak klawiatura czy gra pamięciowa. Ogromne możliwości klawiszy pozwalają zrobić prawie wszystko. Matryca przycisków znajduje się w dolnej środkowej części płytki Joy-Pi, po prawej stronie przekaźnika. Można go łatwo rozpoznać po 16 pojedynczych przyciskach. Doskonałe umiejscowienie na tablicy umożliwia łatwą obsługę klawiszy, zapewniając jednocześnie dobry przegląd wszystkich innych czujników.

Macierz przycisków składa się z czterech kolumn i wierszy. Konfigurujemy wiersze i kolumny macierzy za pomocą ich pinów GPIO i inicjalizujemy obiekt ButtonMatrix () jako zmienną przycisku. Następnie możemy nacisnąć dowolny przycisk matrycy i zobaczyć, który z nich został naciśnięty.

W naszym przykładzie po rozpoznaniu naciśnięcia klawisza aktywujemy funkcję activButton (), która wyświetla numer wciśniętego przycisku. Możesz oczywiście edytować ten moduł, aby zrobić wszystko, co możesz sobie wyobrazić.

Przykładowy kod znajduje się w następnych 2 stronach.

W tym przykładzie musisz przełączać się między modułami. Ustaw WSZYSTKIE przełączniki lewej jednostki przełączającej na włączone. Wszystkie pozostałe przełączniki powinny być wyłączone.

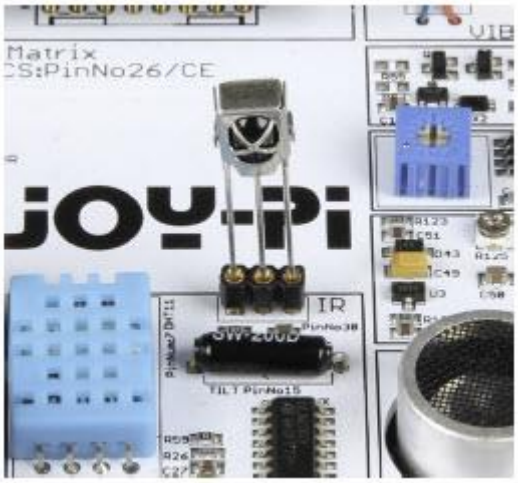

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : original author stenobot
# Original Author Github: https://github.com/stenobot/SoundMatrixPi

import RPi.GPIO as GPIO
import time

class ButtonMatrix():

    def __init__(self):

        GPIO.setmode(GPIO.BOARD)

        # matrix button ids
        self.buttonIDs = [[4,3,2,1],[8,7,6,5],[12,11,10,9],[16,15,14,13]]
        # gpio inputs for rows
        self.rowPins = [13,15,29,31]
        # gpio outputs for columns
        self.columnPins = [33,35,37,22]

        # define four inputs with pull up resistor
        for i in range(len(self.rowPins)):
            GPIO.setup(self.rowPins[i], GPIO.IN, pull_up_down = GPIO.PUD_UP)

        # define four outputs and set to high
        for j in range(len(self.columnPins)):
            GPIO.setup(self.columnPins[j], GPIO.OUT)
            GPIO.output(self.columnPins[j], 1)

    def activateButton(self, rowPin, colPin):
        # get the button index
        btnIndex = self.buttonIDs[rowPin][colPin] - 1
        print("button " + str(btnIndex + 1) + " pressed")
        # prevent button presses too close together
        time.sleep(.3)

    def buttonHeldDown(self,pin):
        if(GPIO.input(self.rowPins[pin]) == 0):
            return True
        return False

def main():

    # initial the button matrix
    buttons = ButtonMatrix()
    try:
```

```
        while(True):
            for j in range(len(buttons.columnPins)):
                # set each output pin to low
                GPIO.output(buttons.columnPins[j],0)
                for i in range(len(buttons.rowPins)):
                    if GPIO.input(buttons.rowPins[i]) == 0:
                        # button pressed, activate it
                        buttons.activateButton(i,j)
                        # do nothing while button is being held down
                        while(buttons.buttonHeldDown(i)):
                            pass
                # return each output pin to high
                GPIO.output(buttons.columnPins[j],1)
    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == "__main__":
    main()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python button_matrix.py

5.19 LEKCJA 19: STEROWANIE I UŻYWANIE CZUJNIKA PODCZERWIENI



W tej lekcji dowiemy się, jak korzystać z odbiornika podczerwieni i jak odbierać kody IR z pilota. Użycie tej metody jest niezwykle przydatne, ponieważ możemy użyć różnych akcji definiowania dla różnych przycisków. Za pomocą pilota możemy włączać różne diody LED lub sterować serwomotorem za każdym naciśnięciem przycisku.

Czujnik podczerwieni jest dostarczany z Joy-Pi, ale nie jest wstępnie zainstalowany. Musisz go podłączyć do gniazda, jak pokazano na powyższym obrazku. Czujnik podczerwieni znajduje się po prawej stronie czujnika DHT11 i nad czujnikiem przechyłu. Wygląda jak mała dioda LED z 3 pinami. Potrzebujemy również pilota na podczerwień, który znajduje się w zestawie Joy-Pi-Kit.

Odbiornik podczerwieni wykorzystuje bibliotekę LIRC i Python-LIRC do odbierania i rozumienia kodów, które wysyłamy pilotem na podczerwień. Zmienna Out zawiera klawisz, który wcisnęliśmy. Za pomocą zapytań if możemy sprawdzić, czy zostały naciśnięte określone klawisze. Informacje te pozwalają nam wykonywać odpowiednie polecenia.

Przykładowy kod znajduje się w następnej stronie.

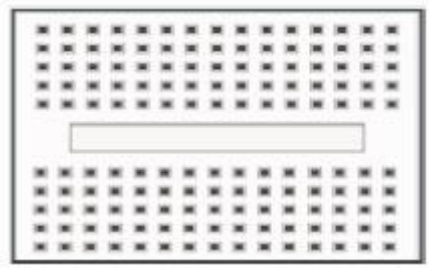
Ważne!!! Usuń czujnik podczerwieni przed zamknięciem obudowy Joy-Pi

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import socket, signal
import lirc, time, sys
import RPi.GPIO as GPIO
from  array import array

GPIO.setmode(11)
GPIO.setup(17, 0)
GPIO.setup(18, 0)
PORT = 42001
HOST = "localhost"
Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Lirc = lirc.init("keys")
#lirc.set_blocking(False, Lirc)       # Un-Comment to stop nextcode() from
# waiting for a signal ( will return empty array when no key is pressed )

def handler(signal, frame):
      Socket.close()
      GPIO.cleanup()
      exit(0)

signal.signal(signal.SIGTSTP, handler)

def sendCmd(cmd):
    n = len(cmd)
    a = array('c')
    a.append(chr((n >> 24) & 0xFF))
    a.append(chr((n >> 16) & 0xFF))
    a.append(chr((n >> 8) & 0xFF))
    a.append(chr(n & 0xFF))
    Socket.send(a.tostring() + cmd)

while True:

      Out = lirc.nextcode()
      print Out[0]
```

Wykonaj następujące polecenia i wypróbuj sam:

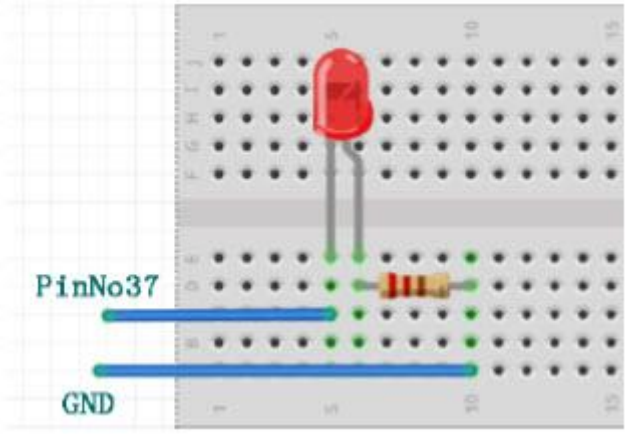
cd /home/pi/Desktop/Joy-Pi/
sudo python IR.py

5.20 LEKCJA 20: własne obwody z płytką stykową



Płytka prototypowa jest niezwykle użyteczną częścią Joy-Pi, która pozwala nam tworzyć własne obwody i funkcje. Teraz, gdy nauczyliśmy się korzystać ze wszystkich czujników, czas stworzyć własny. W tej lekcji utworzysz swój pierwszy niestandardowy obwód na przykładzie migającej diody LED. Płytka prototypowa znajduje się na środku płytki Joy Pi. Jest to mała, biała deska z wieloma małymi otworami.

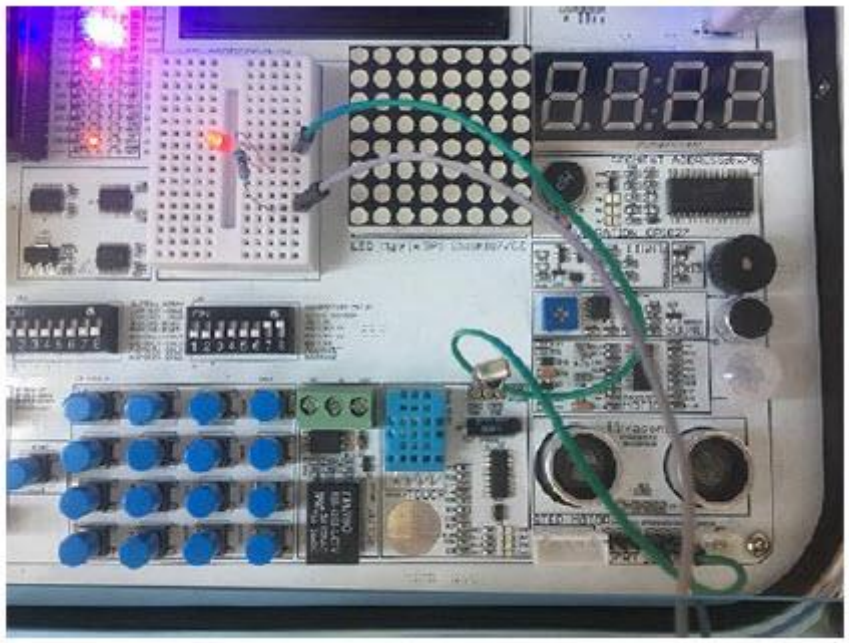
Stworzymy niestandardowy obwód z funkcją migania diody LED. Aby to zrobić, musimy użyć GPIO jako wyjścia i GND, tak jak to zrobiliśmy we wcześniejszych lekcjach. Podłączymy interfejs serwa (interfejs SERVO1) do GPIO 37.



Możesz użyć tego zdjęcia jako przewodnika podczas tworzenia obwodu na płytce z wtyczkami. Pamiętaj, że pin nr 37 znajduje się na porcie GPIO, a GND na porcie GND interfejsu SERVO1.

W tym przykładzie musisz przełączać się między modułami, ponieważ używane są piny serwomechanizmu. Ustawić przełączniki 7 i 8 prawej jednostki przełączającej w położeniu włączenia. Wszystkie pozostałe przełączniki powinny być wyłączone.

Musimy użyć rezystora i podłączyć go do ujemnej strony diody (ujemna strona diody to ta z krótszą nóżką). Drugą stronę rezystora podłączymy przewodem bezpośrednio do pinu GND interfejsu SERVO1. Podłącz dodatnią stronę diody LED do pinu GPIO37 interfejsu SERVO1.



Po zbudowaniu obwodu jest czas na napisanie kodu, który będzie sterował diodą LED. Plan jest taki, aby wysłać GPIO.HIGH do pinu GPIO37, a następnie odczekać 0,2 sekundy i odciąć sygnał za pomocą GPIO.LOW. Zostanie to zapętlone, a dioda LED zacznie migać.

Możesz zatrzymać program, klikając CTRL + C.
Przykładowy kod znajduje się na następnej stronie. Ważne: rezystor, dioda LED i kable nie są dołączone.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import RPi.GPIO as GPIO

# define LED pin
led_pin = 37

# set GPIO mode to GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
# set pin as input
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        # turn on LED
        GPIO.output(led_pin, GPIO.HIGH)
        # Wait half a second
        time.sleep(0.2)
        # turn off LED
        GPIO.output(led_pin, GPIO.LOW)
        # Wait half a second
        time.sleep(0.2)
except KeyboardInterrupt:
    # CTRL+C detected, cleaning and quitting the script
            GPIO.cleanup()
```

Wykonaj następujące polecenia i wypróbuj sam:

cd /home/pi/Desktop/Joy-Pi/
sudo python blinking_led.py

5.21 LEKCJA 21: FOTOGRAFOWANIE APARATEM RASPBERRY PI

Kamera Raspberry Pi jest niezwykle przydatna i może być używana do różnych projektów. Na przykład do kamer bezpieczeństwa, rozpoznawania twarzy i wielu innych. W następnej lekcji wprowadzimy Cię w podstawy korzystania z kamery Raspberry Pi. To nauczy Cię, jak zrobić zdjęcie.

Kamera znajduje się centralnie nad ekranem Joy-Pi i jest podłączona bezpośrednio do Raspberry Pi za pomocą kabla USB.

Najpierw zainstaluj pakiet fswebcam:
(nie musisz go instalować, jeśli używasz naszego gotowego do użycia obrazu)

sudo apt-get install fswebcam

Wpisz polecenie fswebcam, a następnie nazwę pliku, a zdjęcie zostanie zrobione za pomocą kamery internetowej i zapisane pod określoną nazwą:

fswebcam image.jpg

Nasza kamera internetowa ma rozdzielczość 1280x1024, więc aby określić żądaną rozdzielczość, użyjemy flagi –r:

fswebcam -r 1280x1024 image2.jpg

Jeśli chcemy usunąć znacznik czasu, musimy użyć flagi --no-banner:

fswebcam -r 1280x1024 --no-banner image3.jpg

Aby przechwycić wideo, używamy następującego polecenia:

ffmpeg -f v4l2 -r 25 -s 780x480 -i /dev/video0 example.avi

Jeśli chcesz, możesz zmienić rozdzielczość. Po przechwyceniu możesz przejść do folderu zapisu za pomocą polecenia „cd” i odtworzyć wideo za pomocą następującego polecenia:

omxplayer example.mp4

Wideo będzie odtwarzane na pełnym ekranie, jeśli chcesz zamknąć wideo, naciśnij CTRL + C.

6. INFORMACJE I ZOBOWIĄZANIA ZWROTNE

Symbol na sprzęcie elektrycznym i elektronicznym

Ten przekreślony kosz na śmieci oznacza, że sprzęt elektryczny i elektroniczny nie należy do odpadów domowych. Musisz zwrócić stare urządzenia do punktu zbiórki. Przed przekazaniem zużytych baterii i akumulatorów, które nie są zamknięte w zużytym sprzęcie, należy je oddzielić.

Opcje zwrotu
Jako użytkownik końcowy możesz bezpłatnie zwrócić swoje stare urządzenie (które zasadniczo spełnia tę samą funkcję, co zakupione u nas nowe urządzenie) w celu utylizacji przy zakupie nowego urządzenia. Małe urządzenia, których zewnętrzne wymiary nie przekraczają 25 cm, można usuwać w zwykłych ilościach domowych niezależnie od zakupu nowego urządzenia.

Możliwość zwrotu w siedzibie naszej firmy w godzinach otwarcia
Simac GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

Możliwość zwrotu w Twojej okolicy
Wyślemy Ci znaczek do paczki, za pomocą którego możesz bezpłatnie zwrócić urządzenie. Prosimy o kontakt mailowy pod adresem Service @ joy it.net lub telefonicznie.

Informacje na opakowaniu
Jeśli nie masz odpowiedniego materiału opakowaniowego lub nie chcesz używać własnego, skontaktuj się z nami, a wyślemy Ci odpowiednie opakowanie.

Informacje dotyczące utylizacji

a)Produkt

Urządzenie elektroniczne są odpadami do recyclingu i nie wolno wyrzucać ich z odpadami gospodarstwa domowego. Pod koniec okresu eksploatacji, dokonaj utylizacji produktu zgodnie z odpowiednimi przepisami ustawowymi. Wyjmij włożony akumulator i dokonaj jego utylizacji oddzielnie

b) Akumulatory

Ty jako użytkownik końcowy jesteś zobowiązany przez prawo (rozporządzenie dotyczące baterii i akumulatorów) aby zwrócić wszystkie zużyte akumulatory i baterie.
Pozbywanie się tych elementów w odpadach domowych jest prawnie zabronione.
Zanieczyszczone akumulatory są oznaczone tym symbolem, aby wskazać, że unieszkodliwianie odpadów w domowych jest zabronione. Oznaczenia dla metali ciężkich są następujące: Cd = kadm, Hg = rtęć, Pb = ołów (nazwa znajduje się na akumulatorach, na przykład
pod symbolem kosza na śmieci po lewej stronie).

7. Informacje o prawach autorskich

Ten produkt zawiera oprogramowanie, które jest dostępne na warunkach licencji otwartej treści typu GNU General Public License w wersji 2 (GPL) lub X11 License (zwanej również MIT). Pełne teksty licencji zobaczysz na następujących stronach. Bardziej szczegółowe informacje można znaleźć na stronach http://www.gnu.org/licenses/old-licenses/gpl-2.0 i https://www.gnu.org/licenses/license-list.html.

Ponieważ jest to wolne oprogramowanie, nie ma gwarancji, o ile jest to dozwolone przez prawo. Szczegóły hierzu finden Sie in der GNU General Public License und der X11 License. Należy pamiętać, że oczywiście gwarancja na sprzęt nie jest naruszona i obowiązuje w całości

Ponadto dostarczymy kod źródłowy w formie do odczytu maszynowego, obliczony tylko na koszt wytworzenia nośnika. Żądanie należy wysłać na adres service@joy-it.net.

OGÓLNA LICENCJA PUBLICZNA GNU
Wersja 2, czerwiec 1991
Prawa autorskie (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA
Każdy może kopiować i rozpowszechniać wierne kopie
niniejszego dokumentu licencyjnego, ale zmiana go jest niedozwolona.

7. Wsparcie

Wspieramy Cię również po zakupie. W przypadku jakichkolwiek pytań lub problemów, prosimy o kontakt mailowy, telefoniczny lub za pośrednictwem naszego systemu zgłoszeń na naszej stronie internetowej.
E-Mail: service@joy-it.net
System biletów: http://support.joy-it.net
Telefon: +49 (0) 2845 98469 - 66 (11 - 18 Uhr)
Więcej informacji można znaleźć na naszej stronie internetowej:
www.joy-it.net