Zrób to sam

Generowanie ruchu, światła i dźwięku za pomocą Arduino i Raspberry Pi

Simon Monk

Helion

Tytuł oryginału: Make: Action: Movement, Light, and Sound with Arduino and Raspberry Pi

Tłumaczenie: Konrad Matuk

ISBN: 978-83-283-3897-5

© 2018 Helion S.A.

Authorized Polish translation of the English edition of Make: Action ISBN 9781457187797 © 2016 Simon Monk, published by Maker Media Inc.

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

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

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

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

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

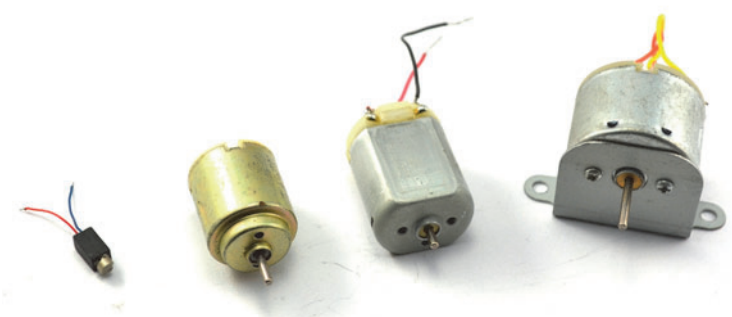
Kup książkę Poleć książkę

Kup książkę Poleć książkę

Silniki, pompy i siłowniki 7

W rozdziale 4. rozpoczęliśmy eksperymentowanie z silnikami prądu stałego. Wiele zagadnień związanych ze sterowaniem tego typu silnikami przydaje się podczas sterowania pracą innych urządzeń za pomocą płytek Arduino i Raspberry Pi.

Na rysunku 7.1 przedstawiono kilka silników prądu stałego. Jak widzisz, komponenty te mogą mieć różne kształty i rozmiary.



Rysunek 7.1. Silniki prądu stałego

Silniki stanowią siłę napędową wielu innych przydatnych urządzeń wyjściowych takich jak pompy i siłowniki liniowe. W tym rozdziale znajdziesz informacje również o tych komponentach.

Podczas lektury podrozdziału „Eksperyment: sterowanie pracą silnika" znajdującego się w rozdziale 4. dowiedziałeś się, że silniki prądu stałego muszą być zasilane prądem większym od dostarczanego przez piny płytek Raspberry Pi i Arduino, dlatego w celu ich włączania i wyłączania należy korzystać z tranzystorów.

W tym rozdziale dowiesz się, jak można sterować prędkością obrotową silników prądu stałego.

Kup książkę Poleć książkę

Działanie silników prądu stałego

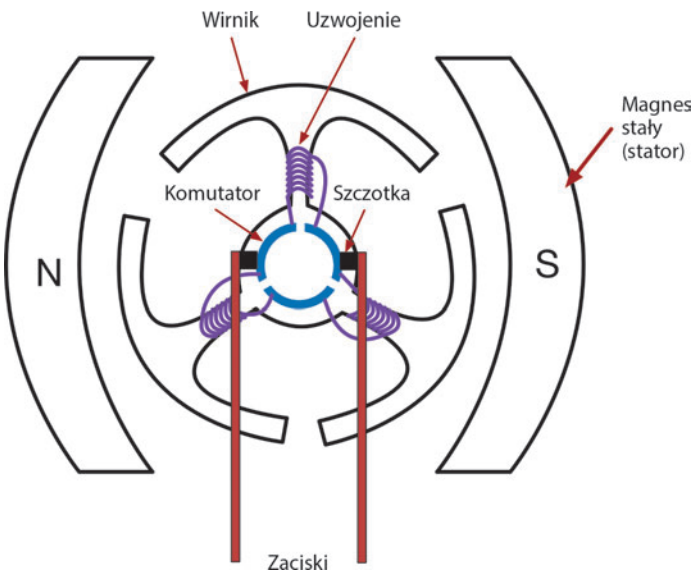
Silniki prądu stałego składają się z trzech głównych elementów konstrukcyjnych (rysunek 7.2): magnesów stacjonarnych (statorów) znajdujących się na zewnątrz silnika, a także wirników (ruchomych elementów silnika) i komutatorów.

Wokół wirnika nawinięte są zwoje drutu. Na rysunku 7.2 widać trzy uzwojenia nawinięte na poszczególne gałęzie wirnika. Uzwojenia te są podłączone do komutatora, który zasila cewki prądem o odpowiedniej polaryzacji we właściwym czasie. Silnik obraca się dzięki temu, że cewki odpychają się we właściwym momencie od magnesów stałych tworzących stator.

Komutator składa się z pierścienia podzielonego na segmenty (na rysunku 7.2 widać trzy segmenty) i ze szczotek stykających się z kolejnymi segmentami podczas obracania się wirnika silnika.

Większość małych silników prądu stałego (przykładowe urządzenia pokazano na rysunku 7.1) składa się z trzech uzwojeń.

Przydatną cechą silnika prądu stałego jest to, że po zmianie polaryzacji dostarczanego do niego prądu silnik obraca się w przeciwnym kierunku.



Rysunek 7.2. Elementy konstrukcyjne silnika prądu stałego

Sterowanie prędkością obrotową (PWM)

W rozdziale 6., w podrozdziale „Eksperyment: mieszanie kolorów", korzystałeś z modulacji czasu trwania impulsu (patrz ramka „Modulacja czasu trwania impulsu" w rozdziale 6), aby regulować jasność diody LED. Technologia ta może być użyta również do sterowania prędkością obrotową silnika.

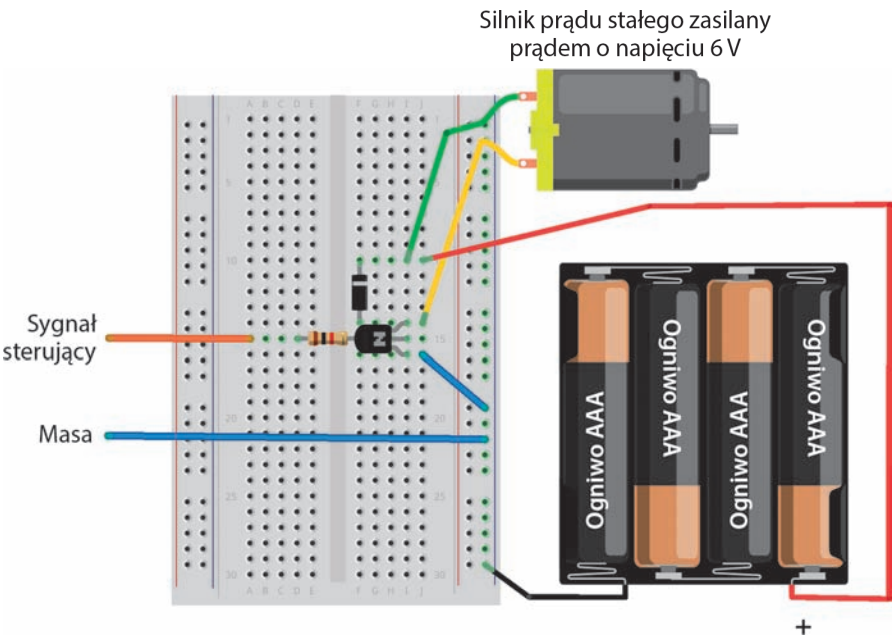
Kup książkę Poleć książkę

Eksperyment: sterowanie prędkością obrotową silnika prądu stałego

W tym eksperymencie będziemy korzystać z tego samego obwodu, którego używaliśmy w podrozdziale „Eksperyment: sterowanie pracą silnika” w rozdziale 4., ale zamiast tylko włączać i wyłączać silnik będziemy sterować jego prędkością obrotową.

Obwód

Jeżeli jeszcze tego nie zrobiłeś, wykonaj obwód opisany w rozdziale 4., w podrozdziale „Eksperyment: sterowanie pracą silnika”. Dla przypomnienia przedstawiam schemat tego obwodu na rysunku 7.3.



Rysunek 7.3. Schemat wykonawczy obwodu sterującego pracą silnika

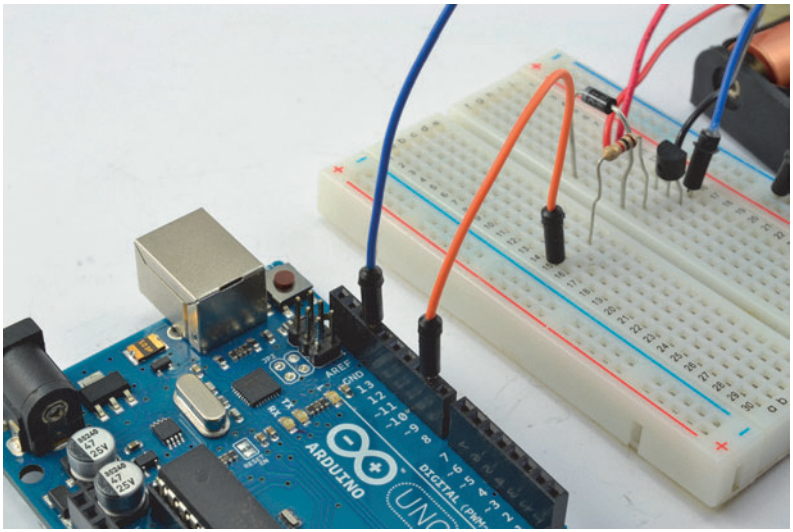
Podłączanie obwodu do Arduino

Podłącz Arduino do płytki obwodu zgodnie z rysunkiem 7.4. Połącz masę płytki prototypowej z pinem *GND* płytki Arduino, a przewód sygnału sterującego podłącz do pinu oznaczonego etykietą *D9*.

Kod Arduino

Szkic Arduino sterujący pracą tego obwodu jest w folderze *arduino\eksperymenty\pwm_silnik* (w miejscu, w którym rozpakowałeś archiwum z kodem). Więcej informacji na ten temat znajdziesz w rozdziale 2., w podrozdziale „Instalowanie szkiców opisanych w tej książce”.

Kup książkę Poleć książkę



Rysunek 7.4. Obwód podłączony do płytki Arduino

Szkic pozwala na definiowanie cyklu roboczego za pomocą monitora portu szeregowego. Czas przeanalizować ten kod:

```
const int controlPin = 9;

void setup() {                      // ❶
   pinMode(controlPin, OUTPUT);
   Serial.begin(9600);
   Serial.println("Zdefiniuj cykl roboczy (0-100)");
}

void loop() {                       // ❷
   if (Serial.available()) {        // ❸
      int duty = Serial.parseInt();
      if (duty < 0 || duty > 100) {  // ❹
         Serial.println("Od 0 do 100");
      }
      else {
         int pwm = duty * 255 / 100;
         analogWrite(controlPin, pwm);
         Serial.print("Aktualny cykl roboczy ");
         Serial.println(duty);
      }
   }
}
```

❶ Funkcja `setup` definiuje pin sterujący (`controlPin`) oraz tryb jego pracy (wyjście), a także uruchamia szeregową transmisję danych za pomocą polecenia `Serial.begin`. Transmisja danych pozwoli na definiowanie cyklu roboczego za pomocą komputera podłączonego do płytki Arduino.

Kup książkę Poleć książkę

❷ Funkcja `loop()` przy użyciu instrukcji `Serial.available` sprawdza, czy do Arduino dotarły za pośrednictwem interfejsu USB dane czekające na przetworzenie.

❸ Jeżeli przesłane dane mają formę liczby, to liczba ta jest odczytywana z łańcucha znaków i przekształcana na wartość całkowitą liczbową (`int`) za pomocą funkcji `parseInt`.

❹ Program sprawdza, czy wartość mieści się w zakresie od 0 do 100. Jeżeli się nie mieści, użytkownik zobaczy komunikat przypominający o tym zakresie w oknie monitora portu szeregowego.

Tekst i liczby

Bardzo często w projektach opisanych w tej książce będziesz wczytywać liczby wprowadzone na monitorze szeregowym do szkiców Arduino za pomocą powyższej techniki. W związku z tym warto, abyś wiedział, jak działa funkcja `parseInt` i jak przebiega przesyłanie wiadomości do płytki Arduino za pomocą interfejsu USB.

W sekcji „Szeregowa transmisja danych" w rozdziale 5. pisałem o tym, że komunikację z płytką Arduino (a także z płytką Raspberry Pi) można nawiązać za pomocą interfejsu szeregowego. Interfejs szeregowy płytki Arduino Uno jest połączony z pinami cyfrowymi oznaczonymi etykietami *D0* i *D1*. Pinów tych nie należy używać w charakterze standardowych złączy wejścia-wyjścia, ponieważ są połączone z szeregowym interfejsem używanym przez płytkę Arduino do komunikacji z komputerem za pośrednictwem kontrolera USB, który pośredniczy w wymianie danych pomiędzy interfejsem USB a bezpośrednim interfejsem szeregowym obsługiwanym przez układ Arduino.

Po wpisaniu wiadomości w oknie monitora szeregowego i wysłaniu jej do Arduino wpisany przez Ciebie tekst jest zamieniany na ciąg bitów (wysokich i niskich impulsów), które odebrane przez Arduino są łączone w ciągi ośmiu impulsów, czyli bajtów. Każdy z tych bajtów jest kodem numerycznym litery alfabetu łacińskiego. Wszystkim cyfrom i literom przypisany jest unikalny kod zdefiniowany przez standard ASCII (ang. *American Standard Code for Information Interchange*).

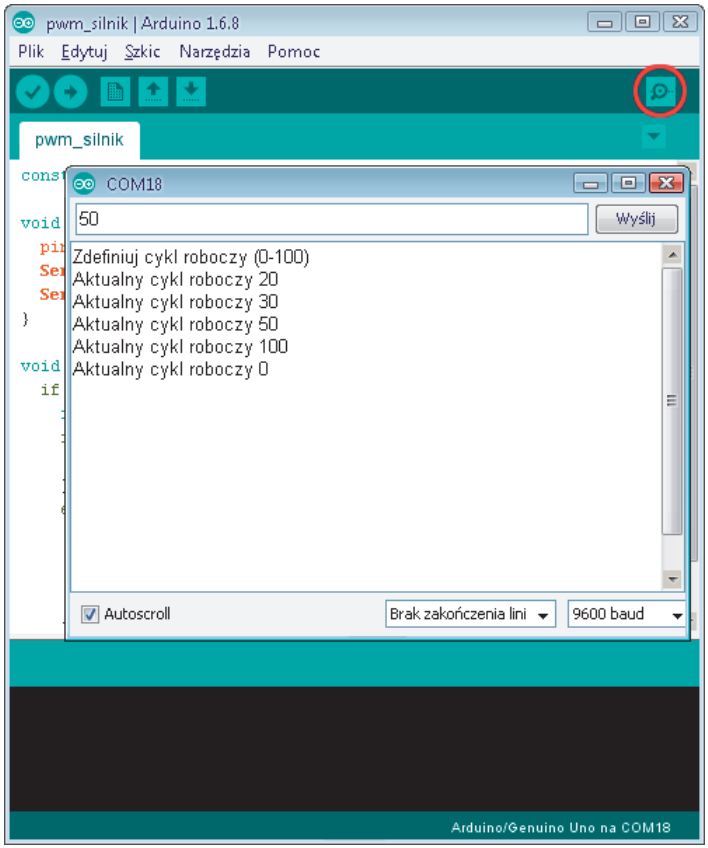
Szkic Arduino, odebrawszy dane przesłane za pomocą interfejsu szeregowego, może odczytywać je po jednym bajcie (literze) lub skorzystać z funkcji `parseInt`, która odczytuje znaki w formie cyfr, a następnie tworzy z nich liczbę. Na przykład liczba 154 zostanie przesłana jako trzy kolejne znaki: 1, 5 i 4. Funkcja `parseInt` przerywa wczytywanie znaków, gdy trafi na znak końca wiersza, spację lub znak, który nie jest cyfrą, i po odczytaniu trzech powyższych znaków zwraca wartość 154 jako dane typu `int`. Funkcja `parseInt` przerywa wczytywanie znaków również wtedy, gdy po odczytaniu ostatniej liczby nastąpi krótka przerwa w transmisji danych.

Jeżeli wprowadzona liczba znajduje się w zakresie od 0 do 100, jest konwertowana na wartość z zakresu od 0 do 255, a następnie kierowana do funkcji `analogWrite()`, która konfiguruje sygnał PWM. Rozwiązanie to zastosowano, ponieważ funkcja `analogWrite()` przyjmuje wartości definiujące cykl roboczy w zakresie od 0 do 255, gdzie wartość 0 to 0%, a wartość 255 to 100 % cyklu roboczego.

Eksperymentowanie z Arduino

Podłącz Arduino do komputera, uruchom środowisko programistyczne tej płytki i kliknij ikonę z lupą (*Monitor szeregowy*) znajdującą się w prawym górnym rogu okna (ikonę tę wyróżniono na rysunku 7.5).

Kup książkę Poleć książkę



Rysunek 7.5. Sterowanie prędkością obrotową silnika za pomocą okna monitora szeregowego

Zostaniesz poproszony o zdefiniowanie cyklu roboczego, czyli podanie wartości z zakresu od 0 do 100. Poeksperymentuj, wprowadzając różne wartości, i zobacz, jak wpływają na prędkość obrotową silnika.

Wprowadzanie niskich wartości takich jak 10 czy 20 może sprawić, że wirnik silnika przestanie się obracać, a silnik zacznie wydawać charakterystyczny terkot świadczący o tym, że dopływa do niego zbyt mało mocy, aby pokonać wewnętrzne siły tarcia i wprawić go w ruch.

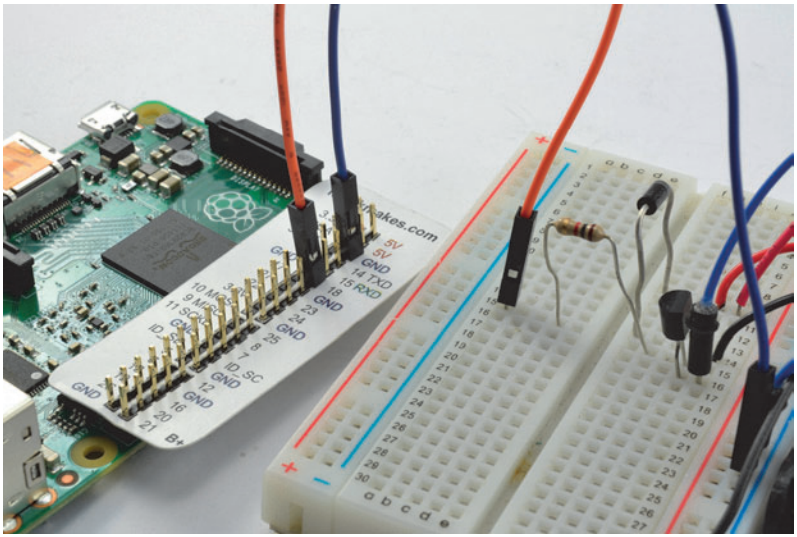
Jeżeli planujesz zastosowanie silnika do czegoś praktycznego, zaprezentowany szkic pozwoli Ci określić minimalną wartość cyklu roboczego pracy silnika.

W celu przerwania pracy silnika zdefiniuj zerową wartość cyklu roboczego.

Podłączanie obwodu do Raspberry Pi

Podłącz płytkę obwodu do Raspberry Pi zgodnie z rysunkiem 7.6. Skorzystaj z przewodów połączeniowych, które z jednej strony są zakończone wtykami męskimi, a z drugiej żeńskimi, i połącz masę płytki z jednym z pinów oznaczonych etykietą *GND*, a przewód sygnału sterującego podłącz do pinu nr 18.

Kup książkę Poleć książkę



Rysunek 7.6. Obwód podłączony do płytki Raspberry Pi

Kod Raspberry Pi

Konstrukcja kodu wykonywanego przez płytkę Raspberry Pi przypomina konstrukcję szkicu Arduino. W przypadku płytki Raspberry Pi program poprosi Cię o podanie wartości cyklu roboczego, zgodnie z którą będzie sterował pinem nr 18.

Program napisany w Pythonie sterujący pracą tego obwodu jest w pliku *pwm_silnik.py* umieszczonym w folderze *python|eksperymenty*. Informacje dotyczące instalowania programów w Pythonie wykorzystywanych w projektach opisanych w tej książce znajdziesz w rozdziale 3., w podrozdziale „Kod zaprezentowany w tej książce". Czas przeanalizować kod:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

control_pin = 18      # ❶

GPIO.setup(control_pin, GPIO.OUT)
motor_pwm = GPIO.PWM(control_pin, 500)  # ❷
motor_pwm.start(0)                      # ❸

try:
    while True:                         # ❹
        duty = input('Zdefiniuj cykl roboczy (0-100): ')
        if duty < 0 or duty > 100:
            print('Od 0 do 100')
        else:
            motor_pwm.ChangeDutyCycle(duty)
```


Kup książkę Poleć książkę

```
finally:
    print("Czyszczenie")
    GPIO.cleanup()
```

❶ Pierwsza część kodu jest taka sama jak początek programu opisanego w podrozdziale „Eksperyment: sterowanie pracą silnika” (rozdział 4.).

❷ W tej linii pin jest przełączany w tryb pracy wyjścia PWM. Biblioteka `RPi.GPIO` pozwala na użycie każdego z pinów gniazda GPIO jako wyjścia PWM. Parametr `500` określa częstotliwość pracy układu PWM równą 500 Hz (impulsów na sekundę).

❸ Sygnał PWM nie jest generowany aż do momentu wywołania polecenia start. Parametr tego polecenia określa początkową wartość cyklu roboczego. Przypisaliśmy mu wartość równą `0`, ponieważ chcemy, aby nasz silnik na początku był wyłączony.

❹ W głównej pętli program prosi użytkownika o zdefiniowanie wartości cyklu roboczego (`duty`) za pomocą polecenia `input`, a następnie sprawdza, czy wprowadzona wartość mieści się w zakresie od 0 do 100. Jeżeli warunek ten jest spełniony, wartość jest przekazywana do funkcji `ChangeDutyCycle`.

Eksperymentowanie z Raspberry Pi

Uruchom program z uprawnieniami administratora i skorzystaj z polecenia `sudo`, żeby poeksperymentować z różnymi wartościami cyklu roboczego. Program powinien zmieniać prędkość obrotową silnika, tak jak to miało miejsce w przypadku szkicu Arduino:

```
pi@raspberrypi ~/ python/eksperymenty $ sudo python pwm_silnik.py
Zdefiniuj cykl roboczy (0-100): 50
Zdefiniuj cykl roboczy (0-100): 10
Zdefiniuj cykl roboczy (0-100): 100
Zdefiniuj cykl roboczy (0-100): 0
Zdefiniuj cykl roboczy (0-100):
```

Sterowanie silnikami prądu stałego za pomocą przekaźników

Jeżeli płytka Arduino lub Raspberry Pi ma tylko okazjonalnie włączać i wyłączać silnik, możesz skorzystać z przekaźnika. Co prawda rozwiązanie to uważane jest za staromodne, ale ma wiele zalet:

- Jest proste i wymaga tylko kilku komponentów.
- Pozwala na solidne odizolowanie silników generujących zakłócenia od delikatnej elektroniki sterującej płytek Arduino i Raspberry Pi.
- Daje możliwość sterowania przepływem prądu o wysokim natężeniu (w przypadku zastosowania odpowiedniego przekaźnika).
- Umożliwia korzystanie z gotowych modułów przekaźników przygotowanych z myślą o pracy z Raspberry Pi lub Arduino.

Oto wybrane wady zastosowania przekaźnika lub modułu z przekaźnikami:

- Przekaźniki są dość duże.
- Rozwiązanie to pozwala tylko na włączanie i wyłączanie silnika, a nie na sterowanie jego prędkością obrotową.

Kup książkę Poleć książkę

- Przekaźniki są urządzeniami elektromechanicznymi zaprojektowanymi zwykle z myślą o przepracowaniu 10 000 000 operacji przełączania. Później mogą ulec uszkodzeniu.

Przekaźnik elektromechaniczny

Na rysunku 7.7 przedstawiono najpopularniejszy typ przekaźnika — tak zwaną kostkę (kształt tego przekaźnika jest zbliżony do kostki cukru, ale zwykle ma czarny kolor).

Ogólna zasada działania przekaźnika elektromagnetycznego polega na tym, że gdy przez jego cewkę płynie prąd (o natężeniu około 50 mA), cewka ta działa jak elektromagnes i zwiera dwa metalowe styki działające jak włącznik. Przez te styki może płynąć prąd o znacznie większym napięciu i natężeniu (nawet dziesiątek amperów).

Schemat przekaźnika | Obudowa przekaźnika | Przekaźnik typu kostka

Rysunek 7.7. Przekaźniki

W tym rozdziale skorzystamy z przekaźnika w celu włączania i wyłączania silnika, ale tak naprawdę przekaźnik może być użyty do sterowania pracą dowolnego urządzenia elektrycznego.

Przekaźniki tego typu określa się mianem przekaźników przełączających jeden biegun (SPCO). Są wyposażone w trzy, a nie w dwa zaciski: zacisk wspólny (zwykle oznaczony etykietą *COM*), zacisk normalnie otwarty (*NO*) i zacisk normalnie zamknięty (*NC*). Termin „normalnie" oznacza „bez dopływu prądu do cewki przekaźnika", a więc styki *NO* i *COM* będą otwarte (rozwarte) aż do podłączenia cewki przekaźnika do prądu. Zacisk NC działa odwrotnie, co oznacza, że zaciski NC i COM będą normalnie połączone, ale po podłączeniu cewki przekaźnika do prądu zostaną rozwarte.

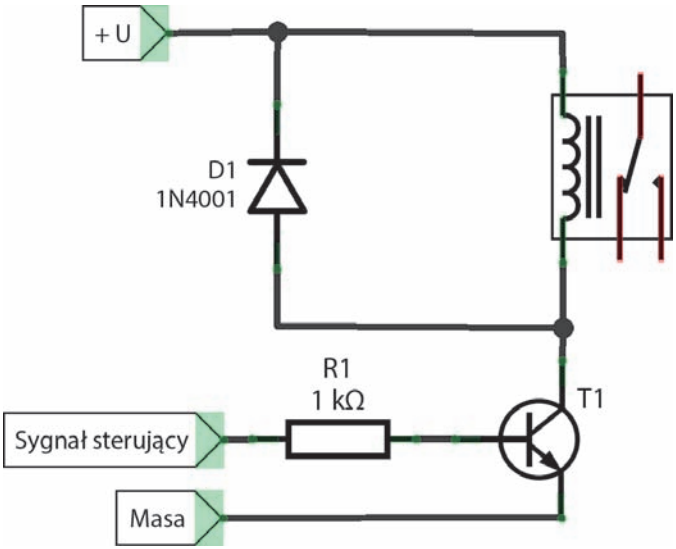
Ogólnie rzecz biorąc, w celu włączania i wyłączania urządzeń korzysta się tylko z zacisków NO i COM.

Sterowanie pracą przekaźnika za pomocą płytek Arduino i Raspberry Pi

Z płytkami Arduino i Raspberry Pi mogą współpracować przekaźniki, których cewki są przeznaczone do używania pod napięciem 5 V. Aby można było podłączyć cewki przekaźników pobierających zbyt dużo prądu (około 50 mA) bezpośrednio do pinów płytki Arduino lub Raspberry Pi, a więc w celu sterowania pracą przekaźnika, należy skorzystać z tranzystora.

Na rysunku 7.8 przedstawiono schemat ideowy obwodu sterującego pracą przekaźnika.

Kup książkę Poleć książkę



Rysunek 7.8. Sterowanie pracą przekaźnika za pośrednictwem małego tranzystora

Cewka przekaźnika przystosowana do pracy pod napięciem 5 V pobiera prąd o natężeniu 50 mA. To nieco przerasta możliwości pinów GPIO płytek Arduino i Raspberry Pi. W związku z tym, podobnie jak w podrozdziale „Eksperyment: sterowanie pracą silnika" (rozdział 4.), skorzystasz z pośrednictwa małego tranzystora (tym razem odbiornikiem podłączonym do tranzystora będzie przekaźnik, a nie silnik).

Takie rozwiązanie ma sens tylko wtedy, gdy silnik (lub inny odbiornik, którego pracą chcesz sterować) charakteryzuje się poborem tak dużego prądu, że nie można sterować nim bezpośrednio za pomocą tranzystora.

Cewka przekaźnika podczas włączania i wyłączania może generować impulsy napięcia, podobnie jak ma to miejsce w przypadku silnika, a więc na schemacie obwodu ponownie znalazła się dioda.

Ze schematu przedstawionego na rysunku 7.8 wynika, że obwód, którego pracą steruje przekaźnik, jest całkowicie odizolowany galwanicznie od cewki. W związku z tym odbiornik podłączony do przekaźnika będzie w mniejszym stopniu zakłócał pracę płytki Arduino i Raspberry Pi.

Cewka przekaźnika pobiera prąd o natężeniu zaledwie 50 mA, a więc możemy ją z powodzeniem podłączyć do prostego tranzystora 2N3904, który kosztuje zaledwie kilkadziesiąt groszy.

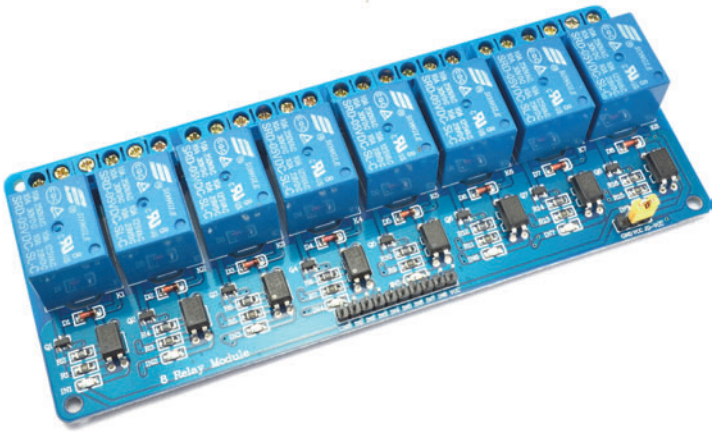
Moduły przekaźników

Jeżeli w swoim projekcie chcesz sterować pracą kilku urządzeń i może to ograniczyć się tylko do włączania i wyłączania, to warto, abyś rozważył zakup modułu przekaźników (rysunek 7.9).

Moduły tego typu znajdziesz w popularnych internetowych serwisach aukcyjnych i w sklepach. Są tanie i zawierają przekaźniki oraz tranzystory, a także małe diody LED sygnalizujące pracę poszczególnych przekaźników. W związku z tym moduły takie mogą być podłączone bezpośrednio do płytki Raspberry Pi lub Arduino.

Nie musisz korzystać ze wszystkich przekaźników znajdujących się w module. W razie potrzeby z pewnością znajdziesz moduły zawierające więcej niż 8 przekaźników.

Kup książkę Poleć książkę



Rysunek 7.9. Moduł przekaźników z 8 kanałami

Moduły te są zwykle wyposażone w następujące złącza:

- *GND* (masa).
- *VCC* lub *5V* — pin ten należy połączyć z wyjściem napięcia 5 V płytki Arduino lub Raspberry Pi. Złącze to zasila cewki aktywowanych przekaźników.
- *Data* — złącza oznaczone tą etykietą służą do sterowania pracą poszczególnych przekaźników. Czasami są dodatkowo oznaczone etykietą *active high*, która świadczy o tym, że cewki przekaźników są włączane po podaniu wysokiego sygnału. Etykieta *active low* oznacza, że cewki przekaźników są włączane po podaniu niskiego sygnału.

Moduły przekaźników zawierają ponadto rząd zacisków śrubowych, które są połączone bezpośrednio ze stykami przekaźników.

Eksperyment: sterowanie pracą silnika elektrycznego za pośrednictwem modułu przekaźników

W tym eksperymencie będziesz włączać i wyłączać silnik elektryczny za pośrednictwem przekaźnika.

W projekcie będziemy korzystać z jednego przekaźnika, a więc możesz kupić moduł mniejszy od mojego (pracowałem z modułem, na którym znajduje się 8 przekaźników).

Lista elementów

Oto lista elementów, które są niezbędne do przeprowadzenia tego eksperymentu, niezależnie od tego czy korzystasz z Arduino, Raspberry Pi, czy masz obie te płytki:

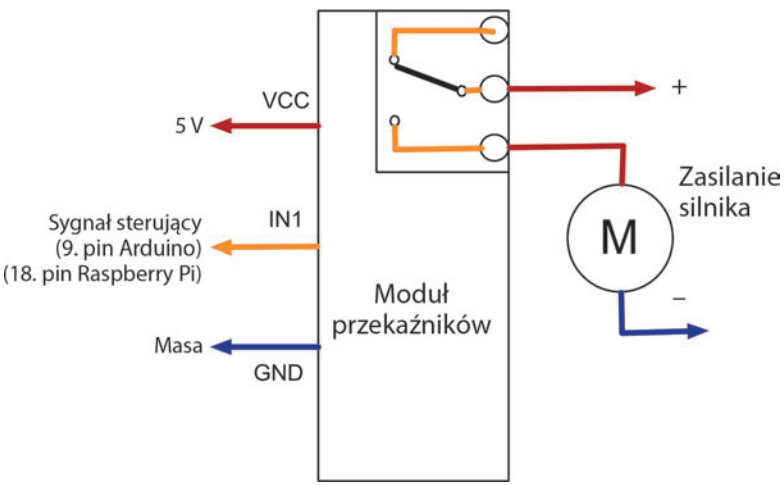
Kup książkę Poleć książkę

Element	Źródło
Mały silnik prądu stałego przystosowany do pracy pod napięciem 6 V	Adafruit: 711
Moduł przekaźników	Allegro
Koszyk na 4 baterie AA (6 V)	Adafruit: 830
Przewody połączeniowe pasujące do wybranego modułu przekaźników	Patrz dodatek A

Niektóre moduły przekaźników są wyposażone w piny, a niektóre w gniazda połączeniowe. W celu podłączenia modułu z męskimi pinami do płytki Arduino będziesz potrzebował przewodów z jednej strony zakończonych końcówkami żeńskimi, a z drugiej męskimi (Adafruit: 826), aby połączyć taki moduł z Raspberry Pi, powinieneś użyć przewodów z końcówkami żeńskimi (Adafruit 266).

Łączenie obwodu

Schemat ideowy tego obwodu umieściłem na rysunku 7.10.



Rysunek 7.10. Schemat ideowy obwodu sterującego pracą silnika prądu stałego za pomocą modułu przekaźników

Styki wewnątrz przełącznika działają jak przełącznik, który może znajdować się w jednym z dwóch położeń. Przekaźnik jest wyposażony w złącze **normalnie otwarte** (*NO*) i **normalnie zamknięte** (*NC*). Gdy do cewki nie dopływa prąd, zacisk wspólny (*COM*) jest połączony z zaciskiem normalnie zamkniętym. Podczas przepływu prądu przez cewkę przekaźnika przełącznik zostaje przestawiony w drugie położenie, a zacisk wspólny jest wtedy połączony z zaciskiem normalnie otwartym.

Kod uruchomiony na Arduino i Raspberry Pi jest prawie identyczny jak przedstawiony w podrozdziale „Eksperyment: sterowanie pracą silnika” (rozdział 4.). Różnicę w działaniu programów zauważysz tylko wtedy, jeżeli Twój moduł podobnie jak mój aktywuje przekaźniki przy niskim sygnale.

Kup książkę Poleć książkę

Kod Arduino

Szkic Arduino sterujący pracą tego projektu jest w folderze *arduino\eksperymenty\przekaznik* (w miejscu, w którym rozpakowałeś archiwum z kodem). Więcej informacji na ten temat znajdziesz w rozdziale 2., w podrozdziale „Instalowanie szkiców opisanych w tej książce".

Szkic włącza przekaźnik (a więc i silnik) na 5 sekund, wyłącza go na kolejne 2 sekundy i powtarza te operacje. Przeanalizuj kod tego programu:

```
const int controlPin = 9;

void setup() {
   pinMode(controlPin, OUTPUT);
}

void loop() {
   digitalWrite(controlPin, LOW);   // ❶
   delay(5000);
   digitalWrite(controlPin, HIGH);
   delay(2000);
}
```

❶ Kod jest praktycznie taki sam jak w podrozdziale „Eksperyment: sterowanie diodą LED" (rozdział 4.). Jedyną różnicę stanowi to, że parametry `LOW` i `HIGH` są przekazywane do funkcji `digitalWrite` w sposób odwrotny. Jeżeli po uruchomieniu programu zauważysz, że silnik jest włączany na 2 sekundy i wyłączany na 5 sekund, to znaczy, że Twój moduł aktywuje przekaźniki przy wysokim sygnale i powinieneś zamienić parametry `LOW` i `HIGH` (tak jak były używane w podrozdziale „Eksperyment: sterowanie diodą LED").

Kod Raspberry Pi

Program napisany w Pythonie sterujący pracą tego układu znajdziesz w pliku *przekaznik.py* umieszczonym w folderze *python\eksperymenty* (szukaj go w miejscu, w którym rozpakowałeś archiwum z kodem).

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

control_pin = 18

GPIO.setup(control_pin, GPIO.OUT)

try:
    while True:
        GPIO.output(control_pin, False)
        time.sleep(5)
        GPIO.output(control_pin, True)
        time.sleep(2)

finally:
    print("Czyszczenie")
    GPIO.cleanup()
```


Kup książkę Poleć książkę

Wybieranie właściwego silnika

Silniki prądu stałego mają różne kształty i rozmiary. Wybierając silnik do swojego projektu, musisz znaleźć model o odpowiedniej mocy. Dwa najważniejsze parametry, które należy wziąć pod uwagę, decydując się na zakup silnika, to siła, jaką może on wygenerować (jego moment obrotowy), i jego maksymalna prędkość obrotowa.

Jeżeli silnik obraca się szybciej, niż tego potrzebujesz, ale generuje za mały moment obrotowy, możesz rozwiązać ten problem za pomocą przekładni.

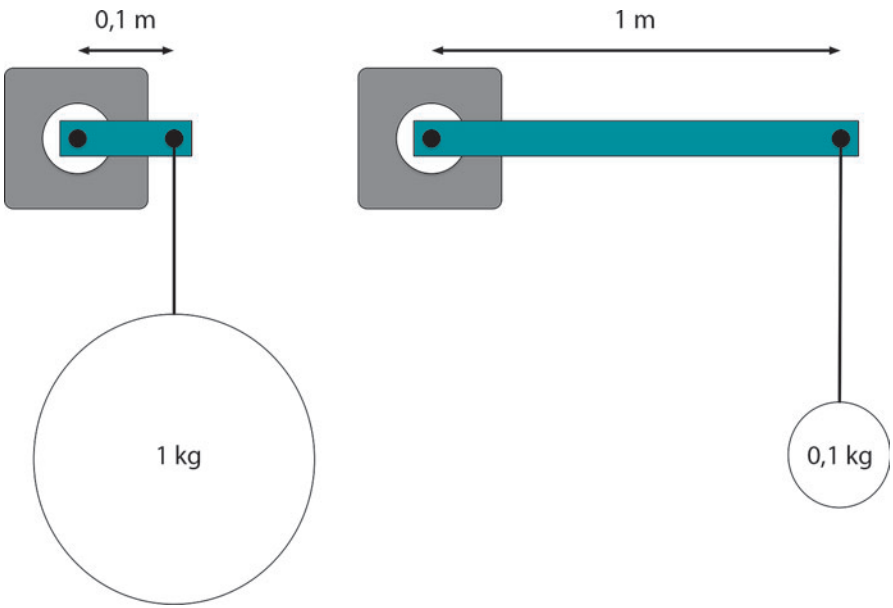
Moment obrotowy

Moment obrotowy określa siłę generowaną przez silnik. Im wyższa wartość momentu, tym większa siła generowana przez obracający się wał silnika.

Moment obrotowy to iloczyn siły i odległości. Siła jest mierzona w niutonach (N), a odległość w metrach (m). Siła momentu obrotowego jest często definiowana jako siła wymagana do podniesienia pewnej masy.

Odległość jest brana pod uwagę, ponieważ im dalej od wału silnika, tym mniej siły silnik może wygenerować. Jeżeli silnik charakteryzuje się na przykład momentem obrotowym 1 Nm (niuton · metr), to będzie w stanie utrzymać masę 1 kg (co odpowiada sile 10 N) w odległości 0,1 m od środka wału. Silnik nie będzie w stanie podnieść tej masy na większą wysokość, ale jednocześnie masa ta nie będzie spadać. Ten sam silnik przy dziesięciokrotnie większej odległości masy od osi będzie w stanie utrzymać dziesięciokrotnie mniejszą masę.

Na rysunku 7.11 pokazano zależność pomiędzy masą i odległością od osi silnika.



Rysunek 7.11. Moment obrotowy (w skrócie)

Kup książkę Poleć książkę

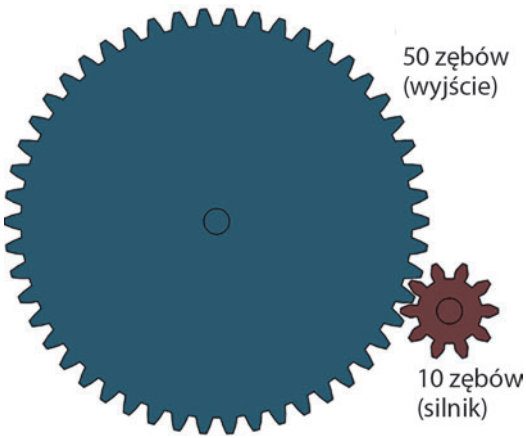
Prędkość obrotowa

Silniki prądu stałego osiągają dość wysokie prędkości obrotowe. W związku z tym bardzo często paruje się je z przekładniami (silniki z wbudowanymi przekładniami określamy mianem silników przekładniowych, opiszę je w kolejnych sekcjach). Standardowy wał silnika zasilanego prądem stałym o niskim napięciu obraca się z prędkością 10 000 obrotów na minutę. Wał takiego silnika obraca się około 166 razy w ciągu sekundy.

Podczas lektury tej książki poznałeś już sposoby zmniejszania prędkości obrotowej silnika elektrycznego, ale techniki te wpływają również na energię dostarczaną do silnika, co zmniejsza generowany przez niego moment obrotowy.

Przekładnie

Przekładnie pozwalają na zwiększenie momentu obrotowego przy zmniejszeniu prędkości obrotowej. Jeżeli skorzystałbyś z przekładni dającej przełożenie 5:1 (zobacz rysunek 7.12), w której jedna zębatka miałaby 50 zębów, a druga 10 zębów, to na każde pięć obrotów silnika przypadałby tylko jeden obrót wyjściowego wału przekładni, ale moment obrotowy na wyjściu przekładni byłby dziesięciokrotnie większy od dostępnego bezpośrednio na silniku.



Rysunek 7.12. Koła zębate przekładni

Silniki przekładniowe

Silniki są w praktyce tak często parowane z przekładniami, że rozsądnym rozwiązaniem jest zakup silnika przekładniowego, czyli silnika zamkniętego we wspólnej obudowie z przekładnią.

Firma Pololu (*http://www.pololu.com*) ma w swojej ofercie szeroki wybór silników przekładniowych.

Jeżeli chcesz zaoszczędzić na zakupie takiego silnika, rozważ model z plastikowymi kołami zębatymi, które są co prawda mniej wytrzymałe od kół wykonanych z metalu, ale generują taki sam moment obrotowy.

Kup książkę Poleć książkę

Pompy

Wewnątrz pompy znajduje się zwykle silnik prądu stałego lub bezszczotkowy silnik prądu stałego (więcej informacji na temat tego komponentu znajdziesz w rozdziale 10., w podrozdziale „Bezszczotkowy silnik prądu stałego") napędzający mechanizm przeganiający ciecz z miejsca na miejsce.

Do najpopularniejszych pomp używanych przez majsterkowiczów należą pompa przewodowa (perystaltyczna) i pompa wirowa. Porównano je na rysunku 7.13 (pompa przewodowa znajduje się po lewej stronie).

Rysunek 7.13. Pompa przewodowa (po lewej) i pompa wirowa (po prawej)

Obie pompy są wyposażone w silniki prądu stałego, ale pracują w inny sposób. Pompy przewodowe stosuje się wtedy, gdy potrzebny jest powolny i miarowy przepływ. Pompy wirowe używane są, gdy ważna jest prędkość przepompowywania cieczy.

Pompy przewodowe

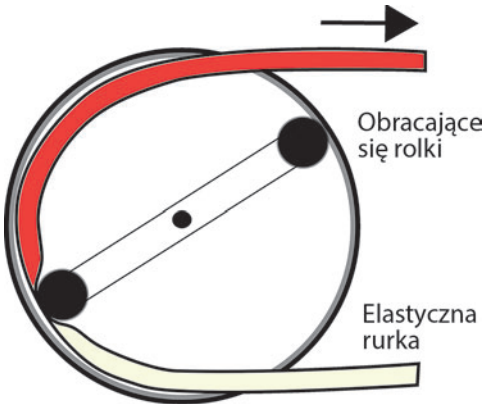
Pompy perystaltyczne są zaprojektowane z myślą o miarowym poruszaniu cieczami. W praktyce stosuje się je często w urządzeniach medycznych i badawczych w celu poruszania dość precyzyjnie odmierzonymi objętościami cieczy. Aby zwiększyć precyzję pracy tych pomp, do ich napędzania stosuje się silniki krokowe (zobacz rozdział 10.).

Na rysunku 7.14 przedstawiono działanie pompy przewodowej.

Wewnątrz takiej pompy znajduje się silnik przekładniowy obracający rolki ściskające giętki wąż, w wyniku czego dochodzi do przemieszczania się płynu wewnątrz węża. Nie powinno nikogo dziwić, że rurka w wyniku ciągłego ściskania w końcu się zużyje i będzie wymagać wymiany. Pompy przewodowe są zwykle skonstruowane tak, aby umożliwić wymianę tego elementu.

Po podłączeniu silnika przekładniowego pompy przewodowej do wyjścia obsługującego technologię PWM i mostka H (ten komponent zostanie opisany w rozdziale 8.) będziesz mógł sterować ilością przepompowywanej cieczy, a także kierunkiem jej przepływu.

Kup książkę Poleć książkę



Rysunek 7.14. Działanie pompy przewodowej

Pompa przewodowa nie wymaga dodatkowego odpowietrzania podczas rozruchu: jeżeli zostanie umieszczona nieco wyżej od zasobnika z cieczą, wytwarzane przez nią podciśnienie będzie w stanie zassać wodę do pompy i uruchomić proces pompowania.

Objętościowe natężenie przepływu

Objętościowe natężenie przepływu określa objętość cieczy, która może zostać przepompowana przez pompę w jednostce czasu. Objętość cieczy i czas mogą być zdefiniowane w kilku różnych jednostkach. Mała pompa przewodowa może charakteryzować się objętościowym natężeniem przepływu na poziomie 50 ml/min (milimetrów/minutę). Pompa wirowa przeznaczona do podlewania ogrodu — objętościowym natężeniem przepływu na poziomie 5 l/min (litrów/minutę).

Pompa wirowa

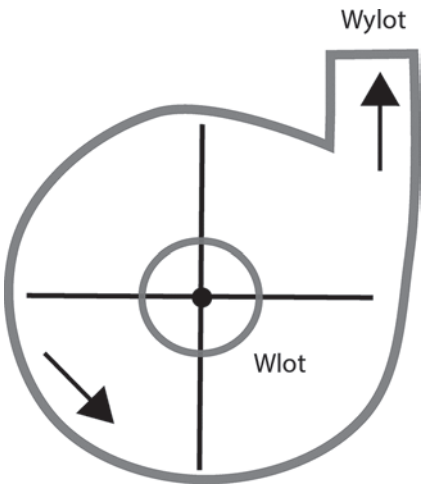
Jeżeli bardziej zależy Ci na szybkim przepompowaniu cieczy, to lepiej, abyś zamiast pompy przewodowej zastosował pompę wirową. Na rysunku 7.15 przedstawiono działanie najpopularniejszego rodzaju pompy wirowej, czyli pompy odśrodkowej.

Ciecz jest zasysana do pompy przez otwór znajdujący się z przodu (rysunek 7.15) na osi silnika napędzającego wirnik. Wirnik generuje siłę odśrodkową, w wyniku czego ciecz wyrzucana jest na zewnątrz, na obudowę pompy (jest kierowana do wylotu).

Pompy wirowe nie potrafią odpowietrzyć się samodzielnie: podczas rozruchu ciecz musi znajdować się na wysokości wlotu. W przeciwieństwie do pomp przewodowych pompy wirowe umożliwiają przepływ cieczy, gdy są wyłączone. Niektóre z tego typu pomp są przeznaczone do ogrodowych sadzawek lub akwariów i mogą zostać całkowicie zanurzone w wodzie.

W przeciwieństwie do pomp przewodowych pompy wirowe nie mogą odwracać kierunku przepływu pompowanej cieczy. Niektóre z nich są wyposażone w bezszczotkowy silnik prądu stałego i elektronikę sterującą jego pracą zamkniętą w tej samej obudowie, dzięki czemu możliwe jest osiągnięcie maksymalnej mocy przy małych wymiarach urządzenia.

Kup książkę Poleć książkę



Rysunek 7.15. Działanie pompy wirowej

Projekt: podlewanie roślin przy użyciu Arduino

Ten prosty projekt sterowany za pomocą Arduino (zobacz rysunek 7.16) służy do codziennego podlewania rośliny określoną ilością wody, która jest dostarczana za pośrednictwem pompy przewodowej (przyda się wszystkim wyjeżdżającym na wakacje).

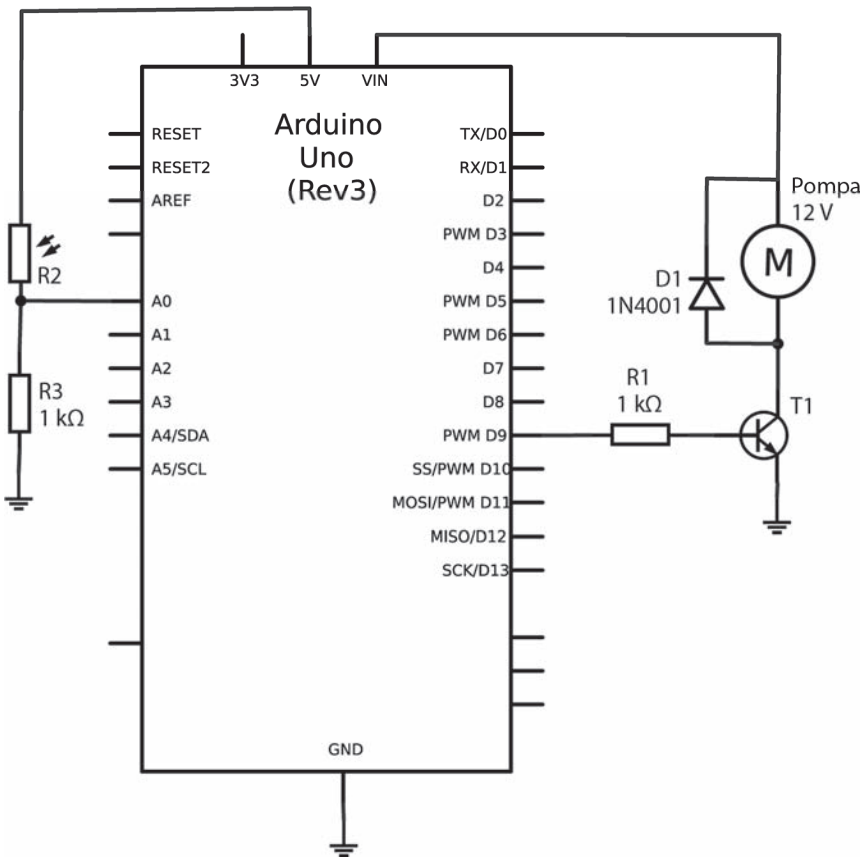
Rysunek 7.16. Urządzenie podlewające domowe kwiaty

Kup książkę Poleć książkę

W projekcie opisanym w tym podrozdziale nie wykorzystano układu mierzącego czas. Zamiast niego zastosowano układ mierzący natężenie światła: rośliny są podlewane, gdy się ściemnia.

Założenia projektowe

Na rysunku 7.17 pokazano schemat ideowy tego projektu.



Rysunek 7.17. Schemat ideowy urządzenia podlewającego domowe kwiaty

Płytka Arduino włącza i wyłącza silnik pompy za pośrednictwem tranzystora MPSA14. Dioda *D1* chroni płytkę przed skokami napięcia.

Po lewej stronie schematu znajdują się fotorezystor i standardowy rezystor, które tworzą dzielnik napięcia umożliwiający pomiar natężenia światła na złączu analogowym *A0* płytki Arduino.

Im więcej światła pada na fotorezystor, tym mniejsza jest jego rezystancja, co powoduje zbliżanie się napięcia odbieranego przez pin *A0* do poziomu 5 V.

Obwód jest dość prosty do zbudowania. Najtrudniejszą rzeczą będzie przylutowanie przewodów do zacisków silnika pompy (większość pomp nie jest wyposażona w przewody połączeniowe).

Kup książkę Poleć książkę

Lista elementów

Oto lista elementów, które są niezbędne do wykonania tego projektu:

Symbol	Element	Źródło
	Płytka Arduino Uno	
T1	Tranzystor MPSA14 w układzie Darlingtona	Mouser: 833-MPSA14-AP
R1, R3	Rezystory 1 kΩ	Mouser: 291-1k-RC
R3	Fotorezystor (1 kΩ)	Adafruit: 161 Sparkfun: SEN-09088
D1	Dioda 1N4001	Adafruit: 755 Sparkfun: COM-08589 Mouser: 512-1N4001
	Pompa perystaltyczna (przewodowa) 12 V	Allegro
	Bezstykowa płytka prototypowa (400 otworów montażowych)	Adafruit: 64
	Przewody połączeniowe obustronnie zakończone wtykami męskimi (tylko płytka Arduino)	Adafruit: 758
	Rurka pasująca do pompy o długości około 1m	Sklep budowlany
	Zasilacz dostarczający prąd o napięciu 12 V i natężeniu 1 A z wtykiem pasującym do płytki Arduino	Botland: ZAS-05045
	Duży pojemnik na wodę	
	Drut połączeniowy o średnicy 0,6 mm (do przylutowania do zacisków silnika pompy)	Adafruit: 1311

W tym projekcie zdecydowałem się na zastosowanie pompy przewodowej przeznaczonej do użycia w akwariach, która jest tania, a jej zakup nie powinien sprawiać problemów.

Rurki, z których korzystałem, pochodzą z zestawu do podlewania znalezionego w markecie budowlanym. Do rurek dołączone były plastikowe złączki, które doskonale sprawdziły się podczas łączenia rurek wychodzących z pompy z rurkami stanowiącymi ich przedłużenie. Połączenia muszą być szczelne. W przeciwnym wypadku pompa nie będzie działać.

Budowa projektu

Projekt wymaga wykonania prostych prac związanych z przygotowaniem płytki obwodu, a także pomajsterkowania w celu przystosowania zbiornika na wodę.

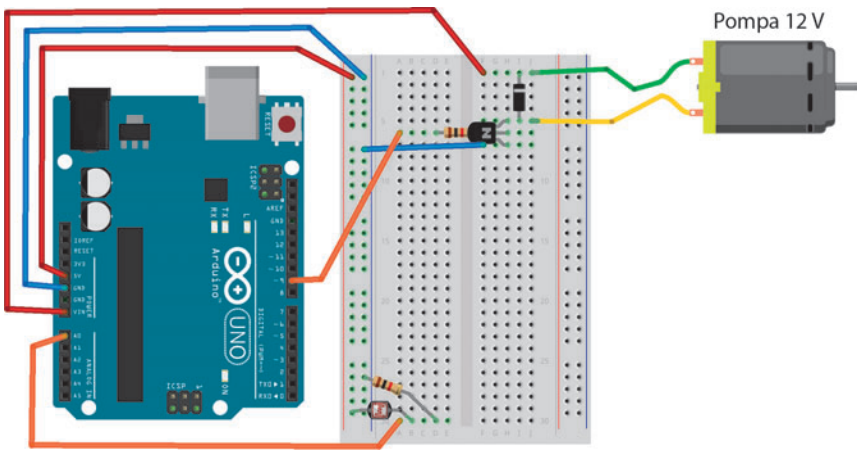
Krok 1: Przylutuj przewody do silnika

Przylutuj przewody do zacisków silnika pompy (oczywiście czynność tę wykonaj, jeżeli pompa nie została wyposażona w fabryczne przewody zasilające). Przewody muszą być na tyle długie, aby połączyć pompę z płytką prototypową i Arduino. Przyjmij, że odpowiednią długością będzie 0,5 m.

Kup książkę Poleć książkę

Krok 2: Połącz obwód na płytce prototypowej

Zamontuj komponenty obwodu na płytce prototypowej, korzystając ze schematu wykonawczego przedstawionego na rysunku 7.18.



Rysunek 7.18. Schemat wykonawczy obwodu sterującego procesem podlewania kwiatów

Upewnij się, że tranzystor i dioda są zwrócone we właściwym kierunku.

Krok 3: Przyczep rurki do pompy

Potrzebujesz dwóch rurek. Jedna z nich powinna sięgać od pojemnika z wodą do wlotu pompy (rurkę tę należy wprowadzić do pojemnika z wodą od góry, tak aby sięgała dna). Druga rurka musi połączyć wylot pompy z rośliną, którą chcesz podlewać. Na rysunku 7.19 pokazano pompę z doczepionymi rurkami.

Rysunek 7.19. Pompa, do której podłączono rurki

Kup książkę Poleć książkę

Zwykle na obudowie pompy nie znajdują się żadne etykiety oznaczające wlot i wylot, ale kierunek pracy pomp przewodowych jest odwracalny, więc jeżeli okaże się, że Twoja pompa zasysa, a nie podlewa, to zamiast zamieniać ze sobą rurki możesz zamienić ze sobą przewody zasilające.

Krok 4: Wykonaj pozostałe prace

Podczas pracy nad projektem odkryłem, że wygodnym rozwiązaniem jest zamontowanie pompy na górze zbiornika z wodą, tak aby głowica pompy była wciśnięta w jego otwór (silnik będzie wtedy zwrócony ku górze, rurka zasysająca wodę zwrócona w dół, a rurka, którą woda ma płynąć do rośliny, będzie zwrócona w bok). W celu zamontowania pompy w ten sposób musiałem naciąć nieco szyjkę pojemnika na mleko, którego użyłem w charakterze zbiornika z wodą.

Na rysunku 7.16 pokazano moją wersję tego projektu. Jeżeli chcesz, możesz zamontować pompę u dołu, obok płytki z obwodem, ale wtedy będziesz musiał przyczepić do niej dłuższą rurkę.

Kod Arduino

Szkic Arduino sterujący pracą tego obwodu znajdziesz w folderze *arduino\projekty\podlewanie* (w miejscu, w którym rozpakowałeś archiwum z kodem). W folderze *projekty* umieszczono również szkic *podlewanie_test* pozwalający na wykalibrowanie czujnika światła.

```
const int motorPin = 9;    // ❶
const int lightPin = A0;

const long onTime = 10 * 1000; // 60 sekund ❷
const int dayThreshold = 200;              // ❸
const int nightThreshold = 70;

boolean isDay = true;                      // ❹

void setup() {
   pinMode(motorPin, OUTPUT);
}

void loop() {        // ❺
   int lightReading = analogRead(lightPin);
   if (isDay && lightReading < nightThreshold) { // Ściemniło się. ❻
      pump();
      isDay = false;
   }
  if (!isDay && lightReading > dayThreshold) {   // ❼
    isDay = true;
   }
}

void pump() {   // ❽
   digitalWrite(motorPin, HIGH);
   delay(onTime);
   digitalWrite(motorPin, LOW);
}
```

❶ Na początku szkicu znajdują się definicje dwóch pinów Arduino, z których korzystamy: pinu sterującego pracą silnika i wejścia analogowego fotorezystora mierzącego natężenie światła (`lightPin`).

Kup książkę Poleć książkę

❷ Stała `onTime` określa czas pracy pompy włączanej co wieczór. Podczas testowania projektu pompa nie powinna pracować zbyt długo, dzięki czemu nie będziesz musiał czekać na zakończenie całego procesu.

Najciekawszą częścią szkicu jest kod wykrywający zmierzch. Płytka Arduino nie ma wbudowanego zegara czasu rzeczywistego, więc jeżeli nie podłączysz do niej takiego zegara w formie zewnętrznego modułu, to mikrokontroler nie będzie wiedział, która jest godzina. W tym projekcie chcemy, aby roślina była podlewana raz dziennie, dlatego wykrycie zmierzchu jest dobrym czynnikiem uruchamiającym proces podlewania. Po podlaniu rośliny mikrokontroler powinien odczekać kolejny dzień (przeczekać długi okres jasności).

❸ W celu odróżnienia dnia od nocy zdefiniowano dwie stałe: `dayThreshold` i `nightThreshold`. Prawdopodobnie będziesz musiał zmienić te wartości i dostosować je do warunków panujących w miejscu, w którym znajduje się Twoja roślina, oraz do czułości fotorezystora. Ogólnie rzecz biorąc, mikrokontroler rozpoznaje dzień po tym, że natężenie światła jest większe od wartości progowej `dayThreshold`. Noc jest rozpoznawana po natężeniu światła mniejszym od `nightThreshold`. Dlaczego zastosowano dwa progi, a nie jeden? Ponieważ o zmierzchu, gdy zaczyna robić się ciemno, natężenie światła może oscylować przez chwilę w okolicy wartości `dayThreshold`, co mogłoby wywołać wielokrotną aktywację procesu podlewania.

❹ Zmiennej logicznej `isDay` przypisywana jest wartość określająca, czy jest dzień. Jeżeli zmiennej przypisana jest wartość `true`, szkic zakłada, że jest dzień.

❺ W funkcji `loop` znajduje się kod logiki określającej konieczność podlania rośliny. Kod ten na początku odczytuje natężenie światła.

❻ Jeżeli obecnie jest dzień, ale poziom natężenia światła spadł poniżej progu `nightThreshold`, to właśnie się ściemniło, a więc kod wywołuje funkcję sterującą pompą w celu podlania rośliny. Następnie zmiennej `isDay` przypisywana jest wartość `false` w celu określenia tego, że jest noc i na razie roślina nie powinna być powtórnie podlana.

❼ Drugie polecenie `if` znajdujące się w funkcji `loop` sprawdza, czy jest noc (`!isDay`) i czy poziom natężenia światła jest większy od progu `dayThreshold`. Gdy oba warunki są spełnione, zmiennej `isDay` przypisywana jest wartość logiczna `true`.

❽ Funkcja `pump` włącza pompę, odczekuje czas określony przez wartość zmiennej `onTime` i przerywa proces pompowania.

Korzystanie z projektu

Zanim uruchomisz właściwy szkic, do pamięci Arduino załaduj program testowy *podlewanie_test.ino*. Pozwoli to na ustalenie właściwych wartości `dayThreshold` i `nightThreshold`. Załaduj szkic do pamięci Arduino i otwórz okno monitora szeregowego.

Na ekranie powinieneś zobaczyć serię wartości będących bieżącymi odczytami z czujnika światła. Zapisz wartości odczytywane w dzień, gdy pogoda jest dość pochmurna. Do zmiennej `dayThreshold` przypisz połowę tej wartości, projekt będzie dzięki temu pracował poprawnie nawet w mało słoneczne dni.

Poczekaj, aż się ściemni, i dokonaj ponownego pomiaru natężenia światła padającego na miejsce, w którym znajduje się roślina. Wartość tę możesz ustalić również przez zasłonięcie czujnika światła palcem. Zmiennej `nightThreshold` przypisz dwukrotność tej wartości. Pamiętaj o tym, że wartość zmiennej `nightThreshold` powinna być znacznie mniejsza od wartości zmiennej `dayThreshold`. W związku z tym być może będziesz musiał dojść do pewnego kompromisu podczas ustalania tych dwóch wartości.

Kup książkę Poleć książkę

Teraz możesz zdefiniować wartości zmiennych `dayThreshold` i `nightThreshold` we właściwym programie (*podlewanie.ino*) i załadować go do pamięci Arduino.

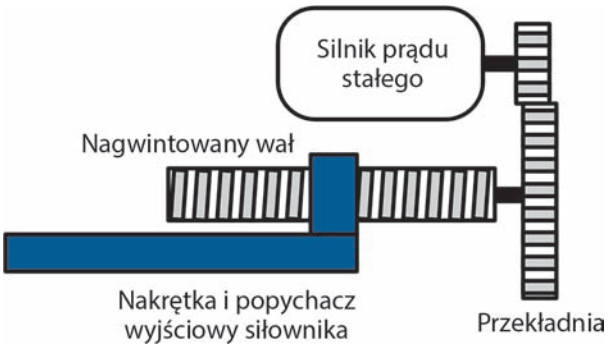
Zmrok możesz symulować, zasłaniając fotorezystor palcem. Powinno to powodować włączenie pompy.

Pompa zastosowana w mojej wersji projektu pompowała około 90 ml cieczy na minutę. W celu określenia czasu pracy pompy możesz skorzystać z kubka z miarką i określić objętościowe natężenie przepływu pompy, a następnie przypisać zmiennej `onTime` wartość pozwalającą na przepompowanie odpowiedniej ilości wody.

Siłowniki liniowe

Siłowniki liniowe zamieniają ruch obrotowy silnika prądu stałego na ruch liniowy. Są często używane do otwierania i zamykania drzwi lub okien.

Siłowniki liniowe są wyposażone w nagwintowany wał napędowy i coś, co jest tak naprawdę nakrętką, która nie może się obracać wokół własnej osi, ale może przemieszczać się wzdłuż nagwintowanego wału, poruszając końcówką siłownika do środka lub na zewnątrz. Działanie takiego siłownika przedstawiono na rysunku 7.20, a na rysunku 7.21 pokazano przykład typowego siłownika liniowego.

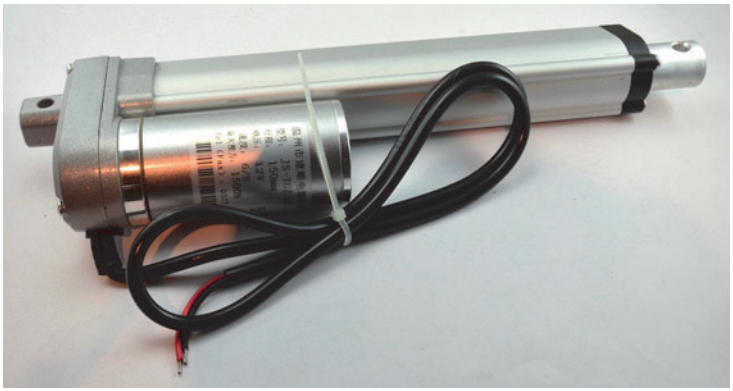


Rysunek 7.20. Działanie siłownika liniowego

Popychacze wyjściowe siłowników liniowych poruszają się dość wolno, ponieważ długi nagwintowany wał i znajdująca się na nim nakrętka działają jak przekładnia redukcyjna, a ponadto siłowniki tego typu są wyposażone w standardową przekładnię znajdującą się bezpośrednio przy silniku. Mała prędkość i mocne silniki sprawiają, że siłowniki liniowe generują dość dużą siłę ciągnącą lub rozpychającą. Siłownik pokazany na rysunku 7.21 generuje siłę ciągnącą lub pchającą na poziomie 1500 N (niutonów), która pozwala na podniesienie masy 150 kg. Silnik siłownika liniowego pracującego z pełną mocą może z łatwością pobierać prąd o natężeniu 5 A przy napięciu 12 V.

Zwykle silnikiem siłownika liniowego steruje się za pomocą mostka H mogącego dostarczyć prąd o maksymalnym natężeniu przynajmniej 5 A, co pozwala na poruszanie siłownikiem w obu kierunkach. Na końcach większości siłowników znajdują się czujniki automatycznie odcinające dopływ prądu do silnika w skrajnych położeniach popychacza, co zapobiega powstawaniu uszkodzeń mechanizmu przekładniowego. Rozwiązanie to

Kup książkę Poleć książkę



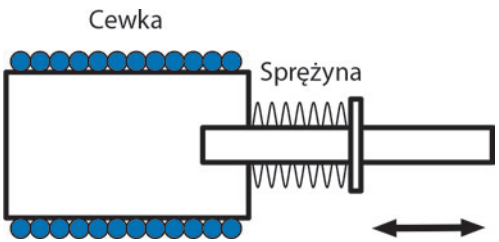
Rysunek 7.21. Siłownik liniowy

ułatwia sterowanie pracą silnika, ponieważ zaprogramowanie mostka H wystarcza wtedy do zasilania silnika w jednym kierunku przez określony czas, który powinien być na tyle długi, aby popychacz siłownika mógł zostać w pełni wyciągnięty.

W rozdziale 8., w podrozdziale „Projekt: zgniatarka do puszek sterowana za pomocą Arduino", zastosowałem w praktyce siłownik pokazany na rysunku 7.21.

Solenoidy

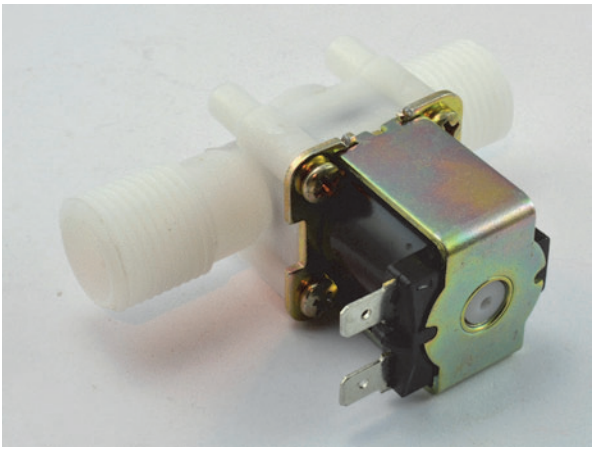
Solenoidy są używane w zamkach do drzwi i w zaworach. Wykonują ruch liniowy podobny do ruchu siłowników liniowych, ale są o wiele prostszymi urządzeniami. To tak naprawdę elektromagnesy poruszające tworzywem, który jest wypychany na zewnątrz lub wciągany do środka. Zakres jego ruchu jest niewielki: zwykle to kilka milimetrów. Działanie solenoidu pokazano na rysunku 7.22, a na rysunku 7.23 przedstawiono elektrozawór sterowany prądem o napięciu 12 V, w którym zastosowano solenoid.



Rysunek 7.22. Działanie solenoidu

Po podłączeniu cewki do prądu ruchomy trzpień zostanie wciągnięty do jej wnętrza (siła wygenerowana przez cewkę będzie większa od siły sprężyny). Po odłączeniu prądu ruchomy trzpień zostanie wysunięty z cewki i znajdzie się w położeniu początkowym.

Kup książkę Poleć książkę



Rysunek 7.23. Zawór wodny sterowany prądem o napięciu 12 V

Zawór wodny pokazany na rysunku 7.23 jest zaprojektowany z myślą o sterowaniu przepływem wody pod ciśnieniem w domowej instalacji hydraulicznej. Zawór blokuje przepływ wody, gdy nie jest zasilany. Po podłączeniu cewki zaworu do prądu ruchomy trzpień przesuwa się, pozwalając na przepływ wody przez zawór aż do momentu odłączenia cewki od prądu.

Zaprezentowany zawór jest przeznaczony do sterowania prądem o napięciu 12 V. Tego typu zawory można znaleźć w pralkach. W sklepach dostępne są zawory przystosowane do sieciowego prądu przemiennego o napięciu 230 V, ale woda i tak wysokie napięcie to dość niebezpieczne połączenie, dlatego radzę Ci, abyś w swoich projektach stosował zawory i pompy sterowane prądem o napięciu 12 V.

Solenoidy są używane również w elektrycznych zatrzaskach do drzwi.

Podsumowanie

W tym rozdziale dowiedziałeś się, jak działają silniki prądu stałego. Ponadto nauczyłeś się włączać i wyłączać takie silniki za pomocą płytek Arduino i Raspberry Pi, a także sterować ich prędkością obrotową.

W kolejnym rozdziale dowiesz się, jak zmieniać kierunek obrotów wału silnika za pomocą mostka H.

Kup książkę Poleć książkę

Skorowidz

A

B

C

Kup książkę Poleć książkę

D

E

F

G

H

I

J

K

Kup książkę Poleć książkę

L

M

N

O

P

Kup książkę Poleć książkę

R

S

Kup książkę Poleć książkę

T

U

W

Z

Kup książkę Poleć książkę

Kup książkę Poleć książkę

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl

GRUPA WYDAWNICZA
Helion SA

Arduino i Raspberry Pi — steruj światłem, dźwiękiem, ruchem!

Elektronika jest dziedziną dla wymagających. Wydaje się bardzo skomplikowana, a przyswojenie choćby samych jej podstaw wymaga nie lada wysiłku. Skoro jednak żyjemy w świecie zdominowanym przez elektronikę, warto jednak pokusić się o znajomość jej prawideł. Zwłaszcza że dzięki płytkom Arduino i Raspberry Pi rozpoczęcie nauki jest bardzo proste. Istnieje tylko jedno niebezpieczeństwo: te płytki niepostrzeżenie mogą rozbudzić niepohamowaną ciekawość i stać się prawdziwą pasją!

Ta książka jest przewodnikiem po elektronice dla początkujących. Szczegółowe i praktyczne instrukcje przeprowadzą Cię krok po kroku przez wiele projektów, dzięki czemu mimochodem przyswoisz podstawy elektroniki i równocześnie odkryjesz potencjał płytek Arduino i Raspberry Pi. Szybko nauczysz się sterować pracą diod LED, różnych silników, cewek, urządzeń zasilanych prądem przemiennym, grzejników, urządzeń chłodniczych, wyświetlaczy i generatorów dźwięku. Dowiesz się, jak możesz monitorować pracę tych urządzeń i kierować nimi przez internet. Dzięki tym popularnym platformom możesz nawet zaprojektować inteligentny dom z Twoim własnym systemem sterowania!

W tej książce między innymi:

- wiele wciągających projektów, od najprostszych po bardziej złożone
- wyjaśnienia dotyczące zastosowań Arduino i Raspberry Pi oraz różnic między nimi
- przystępnie podane podstawy elektroniki
- wskazówki niezbędne przy samodzielnym tworzeniu systemów sterujących

Dr Simon Monk — doktor inżynierii oprogramowania, przez kilka lat pracownik akademicki, później współzałożyciel firmy programistycznej Momote Ltd. Elektronika jest jego ukochanym hobby od wczesnej młodości. Autor wielu lubianych książek dla pasjonatów elektroniki — zachęcony ich popularnością, w 2015 roku wraz z żoną Lindą założył spółkę MonkMakes Ltd, gdzie zajmuje się wymyślaniem nowych publikacji.

Helion
helion.pl
0 801 339900
0 601 339900
INFORMATYKA W NAJLEPSZYM WYDANIU

Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-283-3897-5

Cena: 49,00 zł

Make:
makezine.com