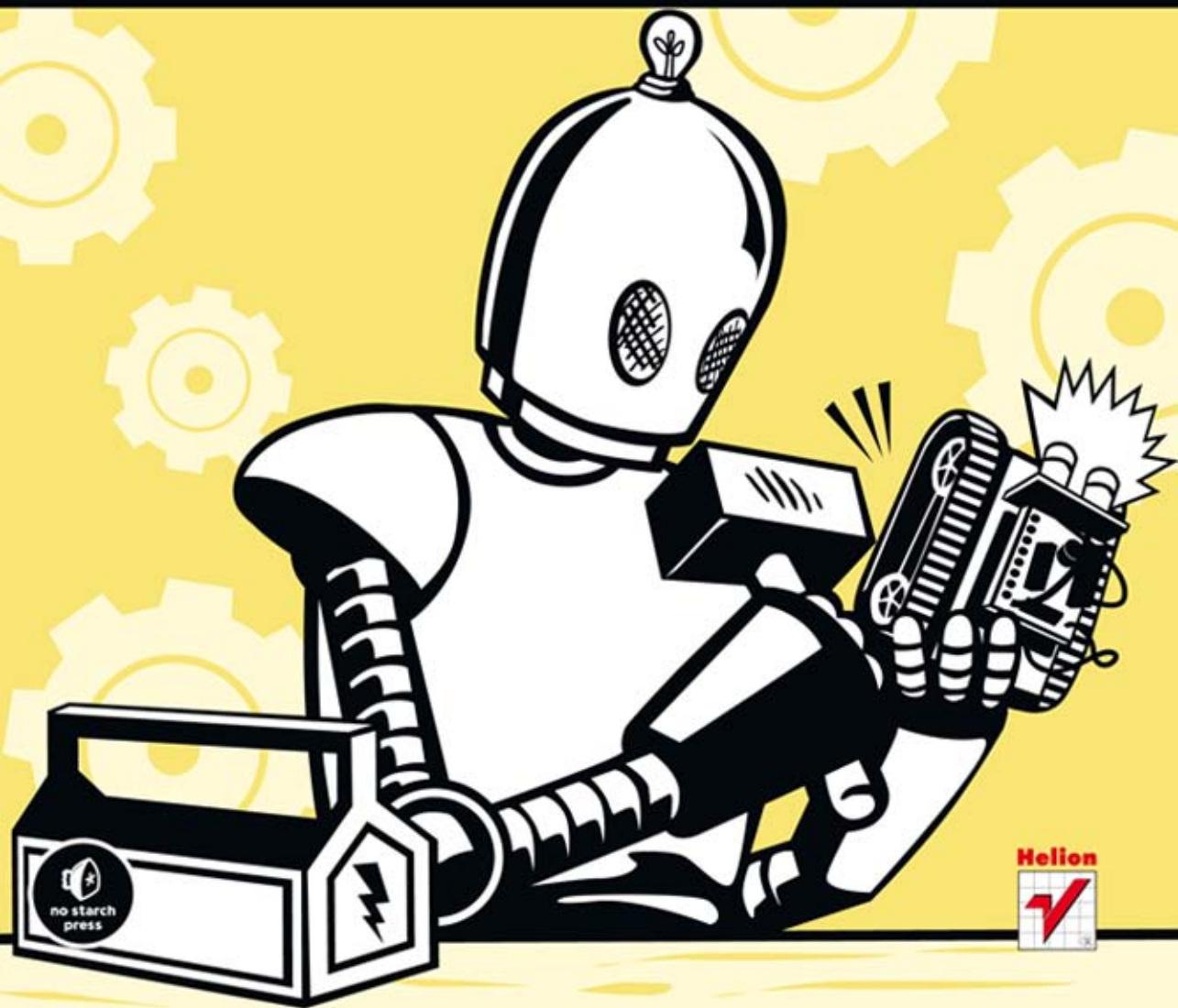


ARDUINO

65 PRAKTYCZNYCH PROJEKTÓW

JOHN BOXALL



no starch
press

Helion



Tytuł oryginału: Arduino Workshop: A Hands-On Introduction with 65 Projects

Tłumaczenie: Mikołaj Szczepaniak

ISBN: 978-83-246-7999-7

Original edition Copyright © 2013 by John Boxall.
All rights reserved.

Published by arrangement with No Starch Press, Inc.

Polish edition copyright © 2014 by Helion SA.
All rights reserved.

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 jakiegokolwiek 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.

Wydawnictwo HELION dołożyło wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie bierze jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Wydawnictwo HELION nie ponosi 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/ardupp.zip>

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

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

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

Spis treści

| | |
|----------------------------|-----------|
| PODZIĘKOWANIA | 17 |
|----------------------------|-----------|

I

| | |
|---|-----------|
| WPROWADZENIE | 19 |
| Nieograniczone możliwości | 20 |
| Popularność | 24 |
| Części i akcesoria | 24 |
| Wymagane oprogramowanie | 25 |
| System Mac OS X | 25 |
| System Windows XP i nowsze | 29 |
| System Ubuntu Linux 9.04 i nowsze | 33 |
| Bezpieczeństwo | 36 |
| Co dalej? | 36 |

2

| | |
|---|-----------|
| PIERWSZE SPOJRZENIE NA PŁYTKĘ ARDUINO I ŚRODOWISKO IDE | 37 |
| Płytką Arduino | 37 |
| Wprowadzenie do środowiska IDE | 42 |
| Obszar poleceń | 43 |
| Obszar tekstu | 44 |
| Obszar komunikatów | 44 |
| Tworzenie pierwszego szkicu w środowisku IDE | 45 |
| Komentarze | 45 |
| Funkcja setup() | 46 |
| Sterowanie sprzętem | 46 |
| Funkcja loop() | 47 |
| Weryfikacja szkicu | 49 |
| Wysyłanie i uruchamianie szkicu | 50 |
| Modyfikowanie szkicu | 50 |
| Co dalej? | 50 |

3

| | |
|---|-----------|
| PIERWSZE KROKI | 51 |
| Planowanie projektów | 52 |
| Kilka słów o elektryczności | 53 |
| Natężenie | 53 |
| Napięcie | 53 |
| Moc | 53 |
| Komponenty elektroniczne | 53 |
| Rezystor | 54 |
| Dioda LED | 57 |
| Płytki uniwersalna | 58 |
| Projekt nr 1: tworzenie fali migających diod LED | 61 |
| Algorytm | 61 |
| Sprzęt | 61 |
| Szkiec | 61 |
| Schemat | 62 |
| Uruchamianie szkicu | 63 |
| Stosowanie zmiennych | 64 |
| Projekt nr 2: powtarzanie instrukcji za pomocą pętli for | 65 |
| Zmiana jasności diod LED za pomocą modulacji szerokości impulsu | 66 |
| Projekt nr 3: przykład zastosowania metody PWM | 67 |
| Dodatkowe komponenty elektroniczne | 68 |
| Tranzystor | 68 |
| Dioda prostownicza | 69 |
| Przekaznik | 70 |
| Obwody zasilane wyższym napięciem | 71 |
| Co dalej? | 72 |

4

| | |
|--|-----------|
| ELEMENTY SKŁADOWE OBWODÓW | 73 |
| Stosowanie schematów obwodów | 74 |
| Identyfikacja komponentów | 75 |
| Połączenia na schemacie obwodu | 77 |
| Analiza przykładowego schematu | 77 |
| Kondensator | 78 |
| Mierzenie pojemności kondensatora | 78 |
| Odczytywanie wartości kondensatorów | 79 |
| Rodzaje kondensatorów | 79 |
| Wejście cyfrowe | 80 |
| Projekt nr 4: przykład użycia cyfrowego wejścia | 82 |
| Algorytm | 82 |
| Sprzęt | 83 |
| Schemat obwodu | 83 |
| Szkiec | 87 |
| Modyfikowanie szkicu | 87 |

| | |
|--|------------|
| Wyjaśnienie szkicu | 87 |
| Tworzenie stałych za pomocą wyrażenia #define | 88 |
| Odczytywanie stanu pinów cyfrowych | 88 |
| Podejmowanie decyzji za pomocą wyrażenia if | 88 |
| Podejmowanie dodatkowych decyzji za pomocą wyrażen if-then-else | 89 |
| Zmienne logiczne | 90 |
| Operatory porównania | 90 |
| Łączenie wielu operacji porównania | 91 |
| Projekt nr 5: sterowanie ruchem samochodowym | 92 |
| Cel | 92 |
| Algorytm | 93 |
| Sprzęt | 93 |
| Schemat | 93 |
| Szkic | 94 |
| Uruchamianie szkicu | 97 |
| Sygnaly analogowe kontra sygnaly cyfrowe | 98 |
| Projekt nr 6: tester baterii | 99 |
| Cel | 99 |
| Algorytm | 99 |
| Sprzęt | 100 |
| Schemat | 100 |
| Szkic | 100 |
| Działania arytmetyczne w systemie Arduino | 102 |
| Zmienne typu float | 102 |
| Operatory porównania liczb | 103 |
| Poprawa precyzji pomiarów sygnału analogowego za pomocą napięcia referencyjnego | 103 |
| Stosowanie zewnętrznego napięcia referencyjnego | 103 |
| Stosowanie wewnętrznego napięcia referencyjnego | 104 |
| Rezystor nastawny | 105 |
| Brzęczyki piezoelektryczne | 106 |
| Schemat elementu piezo | 107 |
| Projekt nr 7: praktyczne wykorzystanie brzęczyka piezo | 107 |
| Projekt nr 8: budowa szybkiego termometru | 108 |
| Cel | 109 |
| Sprzęt | 109 |
| Schemat | 110 |
| Szkic | 110 |
| Doskonalenie szkicu | 112 |
| Co dalej? | 112 |

5

PRACA Z FUNKCJAMI

Projekt nr 9: tworzenie funkcji powtarzającej określone działanie

Projekt nr 10: tworzenie funkcji ustawiającej liczbę cykli włączania diod

Tworzenie funkcji zwracającej wartość

| | |
|--|------------|
| Projekt nr 11: budowa szybkiego termometru z migającymi diodami LED | 117 |
| Sprzęt | 117 |
| Schemat | 118 |
| Szkiec | 118 |
| Wyświetlanie danych odbieranych od płytki Arduino | |
| w oknie monitora portu szeregowego | 120 |
| Monitor portu szeregowego | 120 |
| Projekt nr 12: wyświetlanie temperatury | |
| w oknie monitora portu szeregowego | 122 |
| Diagnostowanie systemów za pomocą monitora portu szeregowego | 123 |
| Podejmowanie decyzji za pomocą wyrażeń while | 124 |
| Konstrukcja do-while | 125 |
| Wysyłanie danych z monitora portu szeregowego do systemu Arduino | 125 |
| Projekt nr 13: mnożenie liczby przez dwa | 126 |
| Zmienne typu long | 127 |
| Projekt nr 14: stosowanie zmiennych typu long | 128 |
| Co dalej? | 129 |

6

| | |
|---|------------|
| LICZBY, ZMIENNE I DZIAŁANIA ARYTMETYCZNE | 131 |
| Generowanie liczb losowych | 132 |
| Generowanie liczb losowych na podstawie napięcia na wolnym pinie | 132 |
| Projekt nr 15: tworzenie elektronicznej kostki do gry | 134 |
| Sprzęt | 134 |
| Schemat | 134 |
| Szkiec | 134 |
| Modyfikowanie szkicu | 137 |
| Krótkie wprowadzenie w świat liczb binarnych | 137 |
| Zmienne typu byte | 137 |
| Zwiększanie liczby dostępnych pinów cyfrowych | |
| za pomocą rejestrów przesuwających | 138 |
| Projekt nr 16: tworzenie wyświetlacza liczb binarnych złożonego z diod LED | 140 |
| Sprzęt | 140 |
| Łączenie rejestru przesuwającego 74HC595 | 140 |
| Szkiec | 142 |
| Projekt nr 17: implementacja binarnego quizu | 143 |
| Algorytm | 143 |
| Szkiec | 143 |
| Tablice | 146 |
| Definiowanie tablicy | 146 |
| Odwoływanie się do wartości w tablicy | 147 |
| Zapisywanie i odczytywanie danych przechowywanych w tablicach | 147 |
| Siedmiosegmentowe wyświetlacze LED | 148 |
| Sterowanie wyświetlaczem LED | 150 |

| | |
|--|------------|
| Projekt nr 18: tworzenie wyświetlacza jednocyfrowego | 151 |
| Sprzęt | 151 |
| Schemat | 151 |
| Szkiec | 151 |
| Wyświetlanie dwóch cyfr | 153 |
| Projekt nr 19: sterowanie dwoma modułami wyświetlaczy siedmiosegmentowych LED | 154 |
| Sprzęt | 154 |
| Schemat | 154 |
| Modulo | 155 |
| Projekt nr 20: budowa termometru cyfrowego | 156 |
| Sprzęt | 157 |
| Szkiec | 157 |
| Moduły wyświetlaczy matrycowych LED | 158 |
| Schemat obwodu wyświetlacza matrycowego LED | 159 |
| Łączenie obwodu | 161 |
| Arytmetyka bitowa | 161 |
| Operator koniunkcji bitowej | 162 |
| Operator alternatywy bitowej | 162 |
| Operator bitowej alternatywy wykluczającej | 163 |
| Operator negacji bitowej | 163 |
| Operatory bitowego przesunięcia w lewo i w prawo | 163 |
| Projekt nr 21: tworzenie wyświetlacza matrycowego LED | 164 |
| Projekt nr 22: wyświetlanie obrazów na wyświetlaczu matrycowym LED | 165 |
| Projekt nr 23: wyświetlanie obrazu na wyświetlaczu matrycowym LED | 167 |
| Projekt nr 24: prezentacja animacji na wyświetlaczu matrycowym LED | 169 |
| Szkiec | 169 |
| Co dalej? | 170 |

7

| | |
|--|------------|
| WYŚWIETLACZE CIEKŁOKRYSTALICZNE | 171 |
| Znakowe moduły LCD | 172 |
| Obsługa znakowego modułu LCD w szkicu | 173 |
| Wyświetlanie tekstu | 174 |
| Wyświetlanie zmiennych i liczb | 175 |
| Projekt nr 25: definiowanie znaków niestandardowych | 176 |
| Graficzne moduły LCD | 178 |
| Łączenie graficznego modułu LCD | 179 |
| Stosowanie modułu LCD | 179 |
| Sterowanie wyświetlaczem | 180 |
| Projekt nr 26: funkcje tekstowe w praktyce | 180 |
| Tworzenie złożonych efektów wizualnych | 181 |
| Projekt nr 27: budowa szybkiego termometru z wyświetlaną historią | 183 |
| Algorytm | 183 |
| Sprzęt | 183 |

| | |
|----------------------------|-----|
| Szkic | 184 |
| Wynik | 185 |
| Modyfikowanie szkicu | 186 |
| Co dalej? | 186 |

8

ROZSZERZANIE MOŻLIWOŚCI PLATFORMY ARDUINO 187

| | |
|--|-----|
| Moduły | 188 |
| Płytką prototypową ProtoShield | 190 |
| Projekt nr 28: tworzenie modułu niestandardowego z ośmioma diodami LED 191 | |
| Sprzęt | 192 |
| Schemat | 192 |
| Układ płytki prototypowej ProtoShield | 192 |
| Projekt | 193 |
| Lutowanie komponentów | 194 |
| Modyfikacja modułu niestandardowego | 195 |
| Rozszerzanie szkiców za pomocą bibliotek | 196 |
| Importowanie bibliotek dla modułów | 196 |
| Karty pamięci microSD | 201 |
| Testowanie karty microSD | 201 |
| Projekt nr 29: zapisywanie danych na karcie pamięci 202 | |
| Projekt nr 30: budowa urządzenia rejestrującego temperaturę 205 | |
| Sprzęt | 205 |
| Szkic | 205 |
| Zarządzanie czasem wykonywania aplikacji za pomocą funkcji millis() i micros() | 208 |
| Projekt nr 31: budowa stopera 210 | |
| Sprzęt | 210 |
| Schemat obwodu | 210 |
| Szkic | 210 |
| Przerwania | 213 |
| Tryby przerwania | 213 |
| Konfiguracja przerwania | 214 |
| Aktywowanie i dezaktywowanie przerwania | 214 |
| Projekt nr 32: stosowanie przerwania 214 | |
| Szkic | 214 |
| Co dalej? | 216 |

9

KLAWIATURY NUMERYCZNE 217

| | |
|--|-----|
| Stosowanie klawiatury numerycznej | 217 |
| Łączenie klawiatury numerycznej | 218 |
| Programowanie obsługi klawiatury numerycznej | 219 |
| Testowanie szkicu | 220 |
| Podjęmowanie decyzji za pomocą konstrukcji switch-case | 220 |

| | |
|---|------------|
| Projekt nr 33: tworzenie zamka sterowanego klawiaturą numeryczną | 221 |
| Szkic | 221 |
| Działanie szkicu | 223 |
| Testowanie szkicu | 223 |
| Co dalej? | 224 |
| | |
| 10 | |
| ODCZYTYWANIE DANYCH WEJŚCIOWYCH UŻYTKOWNIKA ZA | |
| POŚREDNICTWEM EKRAŃÓW DOTYKOWYCH | 225 |
| Ekran dotykowy | 226 |
| Łączenie ekranu dotykowego | 226 |
| Projekt nr 34: adresowanie obszarów na ekranie dotykowym | 227 |
| Sprzęt | 227 |
| Szkic | 227 |
| Testowanie szkicu | 229 |
| Odwzorowywanie punktów dotknięcia ekranu | 229 |
| Projekt nr 35: budowa dwustanowego przełącznika dotykowego | 230 |
| Szkic | 231 |
| Działanie szkicu | 232 |
| Testowanie szkicu | 233 |
| Projekt nr 36: budowa przełącznika dotykowego | |
| podzielonego na trzy obszary | 233 |
| Mapa ekranu dotykowego | 233 |
| Szkic | 234 |
| Działanie szkicu | 235 |
| Co dalej? | 236 |
| | |
| 11 | |
| RODZINA PRODUKTÓW ARDUINO | 237 |
| Projekt nr 37: budowa własnej platformy Arduino na płytce uniwersalnej | 238 |
| Sprzęt | 238 |
| Schemat obwodu | 241 |
| Uruchamianie szkicu testowego | 244 |
| Bogata rodzina płytek Arduino | 247 |
| Płytki Arduino Uno | 249 |
| Płytki Freetronics Eleven | 249 |
| Płytki Freeduino | 250 |
| Płytki Boarduino | 250 |
| Płytki Arduino Nano | 251 |
| Płytki Arduino LilyPad | 251 |
| Płytki Arduino Mega 2560 | 252 |
| Płytki Freetronics EtherMega | 253 |
| Płytki Arduino Due | 253 |
| Co dalej? | 254 |

12

| | |
|--|------------|
| SILNIKI I RUCH | 255 |
| Wprawianie urządzeń w ruch za pomocą silników wykonawczych | 256 |
| Wybór silnika wykonawczego | 256 |
| Łączenie silnika wykonawczego | 257 |
| Uruchamianie silnika wykonawczego | 257 |
| Projekt nr 38: budowa termometru analogowego | 259 |
| Sprzęt | 259 |
| Schemat | 259 |
| Szkic | 260 |
| Stosowanie silników elektrycznych | 261 |
| Tranzystor Darlingtona TIP120 | 262 |
| Projekt nr 39: sterowanie silnikiem | 262 |
| Sprzęt | 262 |
| Schemat | 263 |
| Szkic | 264 |
| Projekt nr 40: budowa robota gąsienicowego i sterowanie tym robotem | 265 |
| Sprzęt | 265 |
| Schemat | 267 |
| Szkic | 270 |
| Wykrywanie kolizji | 272 |
| Projekt nr 41: wykrywanie kolizji robota za pomocą mikroprzełącznika | 272 |
| Schemat | 273 |
| Szkic | 273 |
| Czujniki odległości na podczerwień | 276 |
| Łączenie obwodu | 276 |
| Testowanie czujnika odległości na podczerwień | 276 |
| Projekt nr 42: wykrywanie kolizji robota za pomocą czujnika odległości na podczerwień | 279 |
| Ultradźwiękowe czujniki odległości | 281 |
| Łączenie czujnika ultradźwiękowego | 282 |
| Stosowanie czujnika ultradźwiękowego | 282 |
| Testowanie ultradźwiękowego czujnika odległości | 282 |
| Projekt nr 43: wykrywanie kolizji robota za pomocą ultradźwiękowego czujnika odległości | 284 |
| Szkic | 284 |
| Co dalej? | 287 |

13

| | |
|--|------------|
| STOSOWANIE SYSTEMU GPS NA PLATFORMIE ARDUINO | 289 |
| Czym jest GPS? | 290 |
| Testowanie modułu GPS | 291 |
| Projekt nr 44: budowa prostego odbiornika GPS | 293 |
| Sprzęt | 293 |
| Szkic | 294 |
| Wyświetlanie położenia na ekranie LCD | 295 |

| | |
|---|------------|
| Projekt nr 45: budowa precyzyjnego zegara korzystającego z systemu GPS | 296 |
| Sprzęt | 296 |
| Szkic | 296 |
| Projekt nr 46: rejestrowanie położenia ruchomego obiektu w czasie | 298 |
| Sprzęt | 298 |
| Szkic | 298 |
| Wyświetlanie zarejestrowanych lokalizacji na mapie | 300 |
| Co dalej? | 302 |
| | |
| I 4 | |
| BEZPRZEWODOWE PRZESYŁANIE DANYCH | 303 |
| Stosowanie niedrogich modułów komunikacji bezprzewodowej | 304 |
| Projekt nr 47: zdalne, bezprzewodowe sterowanie urządzeniem | 305 |
| Sprzęt składający się na obwód nadajnika | 305 |
| Schemat nadajnika | 306 |
| Sprzęt składający się na obwód odbiornika | 306 |
| Schemat odbiornika | 306 |
| Szkic nadajnika | 308 |
| Szkic odbiornika | 309 |
| Moduły bezprzewodowego przesyłania danych XBee | |
| — większy zasięg i szybsza transmisja | 310 |
| Projekt nr 48: transmisja danych za pomocą modułów XBee | 312 |
| Szkic | 312 |
| Konfiguracja komputera pod kątem odbierania danych | 313 |
| Projekt nr 49: budowa zdalnie sterowanego termometru | 314 |
| Sprzęt | 314 |
| Układ urządzenia | 315 |
| Szkic | 315 |
| Obsługa urządzenia | 317 |
| Co dalej? | 317 |
| | |
| I 5 | |
| ZDALNE STEROWANIE ZA POMOCĄ PODCZERWIENI | 319 |
| Czym jest podczerwień? | 319 |
| Przygotowanie do użycia podczerwieni | 320 |
| Odbiornik podczerwieni | 320 |
| Pilot | 321 |
| Szkic testowy | 321 |
| Testowanie układu | 322 |
| Projekt nr 50: zdalne sterowanie systemem Arduino na podczerwień | 323 |
| Sprzęt | 323 |
| Szkic | 323 |
| Rozszerzanie szkicu | 325 |

| | |
|---|------------|
| Projekt nr 51: budowa zdalnie sterowanego robota | 325 |
| Sprzęt | 325 |
| Szkiec | 326 |
| Co dalej? | 328 |

16

| | |
|---|------------|
| ODCZYTYWANIE ETYKIET RFID | 329 |
| Wewnętrzna budowa urządzeń RFID | 330 |
| Testowanie sprzętu | 331 |
| Schemat | 331 |
| Testowanie schematu | 331 |
| Projekt nr 52: budowa prostego systemu kontroli | |
| dostępu na bazie technologii RFID | 333 |
| Szkiec | 333 |
| Działanie szkicu | 335 |
| Zapisywanie danych we wbudowanej pamięci EEPROM systemu Arduino | 336 |
| Odczytywanie i zapisywanie danych w pamięci EEPROM | 337 |
| Projekt nr 53: budowa systemu kontroli dostępu RFID | |
| z pamięcią „ostatniej akcji” | 338 |
| Szkiec | 338 |
| Działanie szkicu | 341 |
| Co dalej? | 341 |

17

| | |
|---|------------|
| MAGISTRALE DANYCH | 343 |
| Magistrala I ² C | 344 |
| Projekt nr 54: stosowanie zewnętrznej pamięci EEPROM | 346 |
| Sprzęt | 346 |
| Schemat | 346 |
| Szkiec | 347 |
| Wynik | 349 |
| Projekt nr 55: stosowanie układu ekspandera portów | 350 |
| Sprzęt | 350 |
| Schemat | 350 |
| Szkiec | 352 |
| Magistrala SPI | 353 |
| Łączenie pinów | 353 |
| Implementacja obsługi magistrali SPI | 354 |
| Wysyłanie danych do urządzenia SPI | 355 |
| Projekt nr 56: stosowanie cyfrowego rezystora nastawnego | 356 |
| Sprzęt | 356 |
| Schemat | 357 |
| Szkiec | 357 |
| Co dalej? | 359 |

18

| | |
|---|------------|
| ZEGARY CZASU RZECZYWISTEGO | 361 |
| Łączenie modułu RTC | 362 |
| Projekt nr 57: wyświetlanie daty i godziny | |
| na podstawie zegara czasu rzeczywistego | 362 |
| Sprzęt | 363 |
| Szkiec | 363 |
| Działanie szkicu | 365 |
| Projekt nr 58: tworzenie prostego zegara cyfrowego | 367 |
| Sprzęt | 368 |
| Szkiec | 368 |
| Działanie szkicu i generowane wyniki | 371 |
| Projekt nr 59: budowa systemu kontroli czasu pracy | |
| na bazie technologii RFID | 371 |
| Sprzęt | 372 |
| Szkiec | 373 |
| Działanie szkicu | 377 |
| Co dalej? | 377 |

19

| | |
|--|------------|
| INTERNET | 379 |
| Czego potrzebujemy? | 379 |
| Projekt nr 60: budowa zdalnej stacji monitoringu | 381 |
| Sprzęt | 381 |
| Szkiec | 381 |
| Rozwiązywanie problemów | 384 |
| Działanie szkicu | 385 |
| Projekt nr 61: ćwierkające Arduino | 386 |
| Sprzęt | 386 |
| Szkiec | 386 |
| Sterowanie systemem Arduino za pośrednictwem strony internetowej | 388 |
| Projekt nr 62: konfiguracja usługi Teleduino | |
| i zdalne sterowanie systemem Arduino | 389 |
| Sprzęt | 389 |
| Szkiec | 389 |
| Zdalne sterowanie systemem Arduino | 391 |
| Co dalej? | 392 |

20

| | |
|---|------------|
| KOMUNIKACJA W SIECI TELEFONII KOMÓRKOWEJ | 393 |
| Sprzęt | 394 |
| Przygotowanie modułu zasilania | 395 |
| Konfigurowanie i testowanie sprzętu | 396 |
| Zmiana częstotliwości pracy | 398 |

| | |
|---|------------|
| Projekt nr 63: budowa dzwoniącego Arduino | 400 |
| Sprzęt | 400 |
| Schemat | 401 |
| Szkiec | 401 |
| Działanie szkicu | 402 |
| Projekt nr 64: budowa systemu Arduino wysyłającego SMS-y | 403 |
| Szkiec | 403 |
| Działanie szkicu | 404 |
| Projekt nr 65: konfiguracja systemu sterowanego | |
| za pomocą wiadomości SMS | 405 |
| Sprzęt | 405 |
| Schemat | 405 |
| Szkiec | 405 |
| Działanie szkicu | 408 |
| Co dalej? | 408 |
| SKOROWIDZ | 411 |

16

Odczytywanie etykiet RFID

W tym rozdziale:

- dowiesz się, jak implementować czytniki etykiet RFID na bazie platformy Arduino;
- nauczysz się zapisywać zmienne w pamięci EEPROM na płytce Arduino;
- zaprojektujesz framework dla systemu dostępu do etykiet RFID na bazie Arduino.

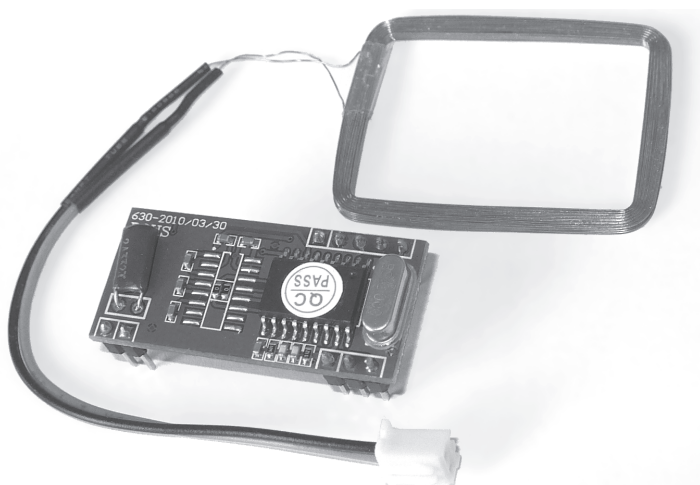
RFID (od ang. *radio-frequency identification*) to bezprzewodowy system wykorzystujący pole elektromagnetyczne do przekazywania danych pomiędzy obiektami bez konieczności bezpośredniego kontaktu tych obiektów. Okazuje się, że na bazie platformy Arduino można zbudować urządzenie odczytujące typowe etykiety i karty RFID. Takie urządzenie może być stosowane w systemach kontroli dostępu lub sterowania cyfrowymi pinami wyjściowymi. Być może miałeś już okazję używać kart RFID na przykład w formie kart dostępu umożliwiających otwieranie drzwi lub kart transportu publicznego zbliżanych do czytników w autobusach lub tramwajach. Na rysunku 16.1 pokazano kilka przykładów etykiet i kart RFID.



Rysunek 16.1. Przykładowe urządzenia korzystające ze standardu RFID

Wewnętrzna budowa urządzeń RFID

Wewnątrz urządzenia RFID znajduje się bardzo mały układ scalony z pamięcią. Dostęp do tego układu wymaga specjalistycznego czytnika. Większość etykiet RFID nie zawiera baterii — są zasilane energią pola magnetycznego generowanego przez czytnik RFID. Pole jest generowane przez cewkę, która jednocześnie pełni funkcję anteny odbierającej dane przesyłane pomiędzy kartą a czytnikiem. Na rysunku 16.2 pokazano cewkę anteny czytnika RFID, który będzie stosowany w tym rozdziale.



Rysunek 16.2. Czytnik RFID używany w tym rozdziale

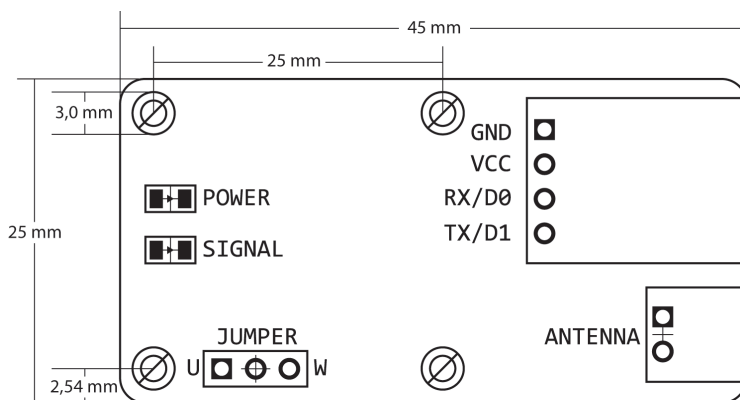
Czytnik kart, który będzie stosowany w tym rozdziale, można kupić w sklepie Seeed Studio (<http://www.seeedstudio.com/>; produkt nr ELB149C5M). Czytnik jest tani i łatwy w użyciu. Ponieważ urządzenie używa częstotliwości 125 kHz, konieczne należy wybrać etykiety RFID operujące na tej samej częstotliwości (na przykład produkt nr RFR103B2B w sklepie Seeed Studio).

Testowanie sprzętu

W tym podrozdziale połączymy czytnik RFID z płytką Arduino, a następnie przetestujemy jego działanie za pomocą prostego szkicu odczytującego sygnał z kart RFID i wysyłającego te dane do monitora portu szeregowego.

Schemat

Na rysunku 16.3 pokazano połączenia niezbędne do budowy modułu RFID.



Rysunek 16.3. Połączenia w ramach modułu RFID

Testowanie schematu

Po połączeniu czytnika RFID z płytką Arduino możemy przetestować ten obwód, umieszczając czarną zworkę w taki sposób, aby łączyła lewy i środkowy pin w sekcji zworek. By połączyć czytnik RFID z płytką Arduino, musisz wykonać następujące kroki (do łączenia obu urządzeń trzeba użyć przewodów połączeniowych z końcówkami żeńskimi i męskimi):

1. Wtyczkę cewki umieść w gnieździe anteny.
2. Pin GND czytnika połącz z pinem GND na płytce Arduino.
3. Pin VCC czytnika połącz z pinem 5 V na płytce Arduino.
4. Pin RX połącz z pinem cyfrowym nr 0 (D0) na płytce Arduino.
5. Pin TX połącz z pinem cyfrowym nr 1 (D1) na płytce Arduino.

UWAGA

Na czas wysyłania szkieców na płytke Arduino połączoną z czytnikiem RFID należy usunąć przewód łączący pin RX tego czytnika z pinem cyfrowym nr 0 na płytce Arduino. Przewód można ponownie podłączyć dopiero po prawidłowym wysłaniu szkiecu. Usunięcie połączenia jest konieczne, ponieważ pin D0 jest używany przez system Arduino także do komunikacji z komputerem i otrzymywania szkieców.

Szkic testowy

Po wpisaniu szkiecu z listingu 16.1 należy go wysłać na płytke Arduino.

Listing 16.1. Szkic testujący moduł RFID

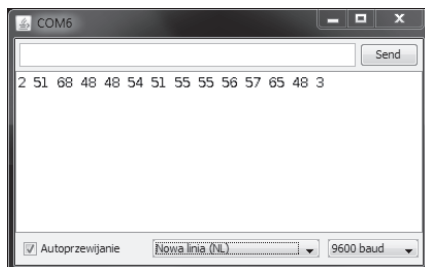
```
// Listing 16.1
int data1 = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  if (Serial.available() > 0) {
    data1 = Serial.read();
    // wyświetla otrzymaną liczbę
    Serial.print(" ");
    Serial.print(data1, DEC);
  }
}
```

Weryfikacja danych w oknie monitora portu szeregowego

Należy teraz otworzyć okno monitora portu szeregowego i wykonać kilka ruchów etykietą RFID nad cewką. Wyświetlony wynik powinien przypominać dane widoczne na rysunku 16.4.



Rysunek 16.4. Przykładowe dane wynikowe wygenerowane przez szkic z listingu 16.1

Warto zwrócić uwagę na wyświetlenie aż 14 liczb w oknie monitora portu szeregowego. Sekwencja liczb to unikatowy identyfikator etykiety RFID. W dalszych szkicach właśnie na podstawie tej sekwencji będziemy identyfikować etykiety. Warto teraz zapisać liczby wyświetlone dla każdej z etykiet, ponieważ będziemy potrzebowali tych identyfikatorów w kilku następnych projektach.

Projekt nr 52: budowa prostego systemu kontroli dostępu na bazie technologii RFID

Czas sprawdzić system RFID w praktycznych zastosowaniach. W tym projekcie pokażę, jak wywoływać zdarzenia systemu Arduino w momencie odczytania prawidłowej etykiety RFID. W kodzie szkicu zapisano identyfikatory dwóch etykiet RFID — w momencie odczytania jednej z tych kart przez czytnik szkic wyświetli w oknie monitora portu szeregowego komunikat *Karta zaakceptowana*. W razie przyłożenia do czytnika niewłaściwej karty w oknie monitora pojawi się komunikat *Karta odrzucona*. Rozwiązanie przygotowane w ramach tego projektu będzie punktem wyjścia dla dalszych prac przy użyciu technologii RFID (polegających między innymi na uzupełnianiu istniejących projektów o ten mechanizm).

Szkic

W środowisku IDE należy wpisać i wysłać następujący szkic. Warto jednak pamiętać o konieczności zastąpienia znaków x w obu tablicach (w wierszach ❶ i ❷) wartościami składającymi się na identyfikatory stosowanych kart RFID, które można sprawdzić za pomocą szkicu opisanego we wcześniejszej części tego rozdziału. (Same tablice zostały omówione w rozdziale 6.)

```
// Projekt nr 52 — budowa prostego systemu kontroli dostępu na bazie technologii RFID
```

```
int data1 = 0;
int ok = -1;
```

```
// identyfikatory etykiet RFID można sprawdzić za pomocą szkicu z listingu 16.1
```

```
❶ int tag1[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
❷ int tag2[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
```

```
int newtag[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0}; // tablica używana do
// porównywania odczytanego identyfikatora
```

```
void setup()
{
```

```
  Serial.flush(); // należy wyczyścić bufor portu szeregowego,
// w przeciwnym razie pierwszy odczyt mógłby być nieprawidłowy
```

```

    Serial.begin(9600);
}

③ boolean comparetag(int aa[14], int bb[14])
{
    boolean ff = false;
    int fg = 0;
    for (int cc = 0 ; cc < 14 ; cc++)
    {
        if (aa[cc] == bb[cc])
        {
            fg++;
        }
    }
    if (fg == 14)
    {
        ff = true;
    }
    return ff;
}

④ void checkmytags() // porównuje oba zapisane identyfikatory z właśnie odczytanym
                    // identyfikatorem
{
    ok = 0; // ta zmienna ułatwia podjęcie decyzji;
           // jeśli ma wartość 1, identyfikatory są takie; jeśli ma wartość 0,
           // identyfikatory są różne;
           // wartość -1 oznacza, że nie odczytano identyfikatora do sprawdzenia
    if (comparetag(newtag, tag1) == true)
    {
        ⑤ ok++;
    }
    if (comparetag(newtag, tag2) == true)
    {
        ⑥ ok++;
    }
}

void loop()
{
    ok = -1;
    if (Serial.available() > 0) // jeśli umieszczono etykietę w sąsiedztwie czytnika,
    {
        // należy odczytać liczbę przekazaną za pośrednictwem pinu szeregowego RX
        delay(100); // czas potrzebny na przesłanie danych
                   // z bufora portu szeregowego
        for (int z = 0 ; z < 14 ; z++) // odczytuje pozostałe liczby identyfikatora
        {
            ⑦ data1 = Serial.read();
            newtag[z] = data1;
        }
    }
}

```

```

    }
    Serial.flush(); // usuwa z bufora ewentualne kolejne odczyty
                  // czas porównać identyfikatory
    ❸ checkmytags();
  }
  ❹ // tutaj należy podjąć kroki zależnie od dopasowania identyfikatorów
  if (ok > 0) // jeśli identyfikatory są takie same
  {
    Serial.println("Karta zaakceptowana");
    ok = -1;
  }
  else if (ok == 0) // jeśli identyfikatory są różne
  {
    Serial.println("Karta odrzucona");
    ok = -1;
  }
}
}

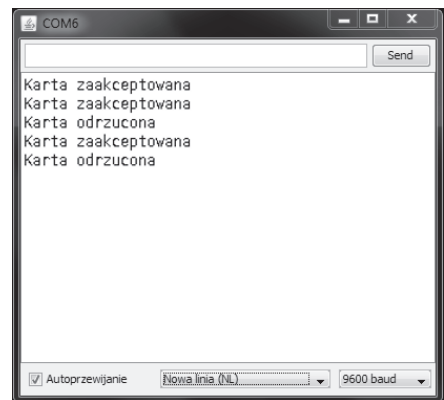
```

Działanie szkicu

Po umieszczeniu etykiety RFID w sąsiedztwie czytnika liczby składające się na identyfikator tej etykiety są wysyłane za pośrednictwem portu szeregowego. W wierszu ❷ szkic pobiera wszystkie 14 liczb i umieszcza je w tablicy newtag[]. Identyfikator jest następnie porównywany z dwoma identyfikatorami zapisanymi w wierszach ❶ i ❷. Do porównania identyfikatorów wykorzystano funkcję checkmytags() (zdefiniowaną w wierszu ❹ i wywołowaną w wierszu ❸). Każda para identyfikatorów jest porównywana za pomocą funkcji comparetag() zdefiniowanej w wierszu ❸.

Funkcja comparetag() otrzymuje na wejściu dwie tablice liczb i zwraca wartość logiczną określającą, czy te tablice są takie same (true), czy różne (false). Jeśli odczytany identyfikator pasuje do któregoś z identyfikatorów zapisanych w kodzie, zmiennej ok jest przypisywana wartość 1 w wierszach ❹ i ❺. I wreszcie w wierszu ❻ szkic decyduje o dalszym działaniu w zależności od ewentualnego dopasowania identyfikatorów.

Po wysłaniu szkicu i ponownym połączeniu pinu cyfrowego nr 0 na płycie Arduino z pinem RX czytnika RFID (patrz rysunek 16.3) należy otworzyć okno monitora portu szeregowego i umieścić kartę RFID w sąsiedztwie czytnika. Wynik wyświetlony na ekranie powinien przypominać dane widoczne na rysunku 16.5.



Rysunek 16.5. Dane wynikowe wygenerowane przez szkic dla projektu nr 52

Zapisywanie danych we wbudowanej pamięci EEPROM systemu Arduino

Zmienne definiowane i stosowane w szkicach dla platformy Arduino przechowują dane tylko do momentu zresetowania tego systemu lub wyłączenia zasilania. Co w takim razie można zrobić, aby zachować te wartości na przyszłość? Jak trwale zapisać na przykład stały kod PIN, który użytkownik będzie mógł wpisywać za pośrednictwem klawiatury numerycznej (jak w rozdziale 9.)? W takim przypadku warto skorzystać z *pamięci EEPROM* (od ang. *electrically erasable programmable read-only memory*). Pamięć EEPROM przechowuje wartości zmiennych w mikrokontrolerze ATmega328. Wartości zapisane w tej pamięci są zachowywane także po wyłączeniu zasilania.

Pamięć EEPROM na płytce Arduino może przechowywać wartości 1024 zmiennych bajtowych na pozycjach oznaczonych numerami od 0 do 1023. Wystarczy przypomnieć sobie, że jeden bajt może reprezentować liczbę całkowitą z przedziału od 0 do 255, aby zrozumieć, że opisywana pamięć wprost doskonale nadaje się do przechowywania liczb składających się na identyfikator etykiety RFID. Warunkiem korzystania z pamięci EEPROM w szkicach jest wywołanie biblioteki *EEPROM* (dołączonej do środowiska Arduino IDE) za pomocą następującego wyrażenia:



Zapisanie wartości w pamięci EEPROM wymaga już tylko jednego prostego wyrażenia:



Parametr *a* reprezentuje pozycję (z przedziału od 0 do 1023), na której dana wartość ma zostać zapisana, natomiast parametr *b* to zmienna zawierająca bajt danych, który ma być zapisany w pamięci EEPROM na wskazanej pozycji.

Aby uzyskać dane z pamięci EEPROM, należy użyć wyrażenia w postaci:



Powyższe wyrażenie odczytuje dane zapisane w pamięci EEPROM na pozycji `position` i zapisuje je w zmiennej `value`.

UWAGA

Żywotność pamięci EEPROM jest dość ograniczona, zatem pamięć po pewnej liczbie operacji zapisu przestanie działać! Według deklaracji producenta (firmy Atmel) pamięć EEPROM platformy Arduino obsługuje maksymalnie 100 tys. cykli zapisu i odczytu na każdej pozycji. Operacje odczytu nie wpływają na czas działania tej pamięci.

Odczytywanie i zapisywanie danych w pamięci EEPROM

W tym punkcie przeanalizujemy przykład szkicu zapisującego dane w pamięci EEPROM i odczytującego te dane. Należy teraz wpisać i wysłać szkic z listingu 16.2.

Listing 16.2. Szkic demonstrujący działanie pamięci EEPROM

```
// Listing 16.2
#include <EEPROM.h>
int zz;

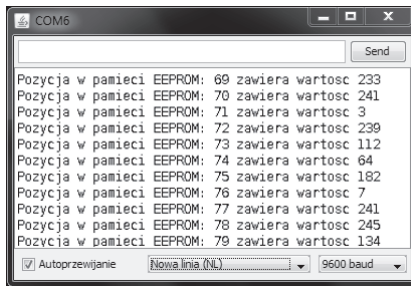
void setup()
{
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop()
{
  Serial.println("Zapisywanie liczb losowych...");
  for (int i = 0; i < 1024; i++)
  {
    zz = random(255);
    ❶ EEPROM.write(i, zz);
  }
  Serial.println();
  for (int a = 0; a < 1024; a++)
  {
    ❷ zz = EEPROM.read(a);
    Serial.print("Pozycja w pamięci EEPROM: ");
    Serial.print(a);
    Serial.print(" zawiera wartość ");
    ❸ Serial.println(zz);
    delay(25);
  }
}
```

W pętli w wierszu ❶ liczba losowa z przedziału od 0 do 255 jest zapisywana na każdej pozycji w pamięci EEPROM. Zapisane wartości są następnie odczytywane w drugiej pętli w wierszu ❷ oraz wyświetlane w oknie monitora portu szeregowego (w wierszu ❸).

Po wysłaniu tego szkicu na platformę Arduino należy otworzyć okno monitora portu szeregowego, w którym powinny zostać wyświetlone dane podobne do tych z rysunku 16.6.

Możemy teraz przystąpić do realizacji projektu korzystającego z pamięci EEPROM.



Rysunek 16.6. Przykładowe dane wynikowe wygenerowane przez szkic z listingu 16.2

Projekt nr 53: budowa systemu kontroli dostępu RFID z pamięcią „ostatniej akcji”

Mimo że w projekcie nr 52 pokazałem, jak używać etykiet RFID do sterowania urządzeniem, na przykład światłem lub zamkiem elektrycznym, musimy pamiętać o tym, że ponowne uruchomienie lub wyłączenie systemu spowoduje utratę zapisanych danych. Jeśli na przykład po włączeniu światła zostało wyłączone zasilanie, po ponownym włączeniu zasilania światło nie zostanie automatycznie włączone. Tym razem chcemy, aby po ponownym włączeniu zasilania system Arduino pamiętał i przywrócił swój stan sprzed wyłączenia zasilania. Spróbujmy rozwiązać ten problem w tym projekcie.

Szkic tego projektu będzie zapisywał ostatnie działanie (akcję) w pamięci EEPROM (tą akcją może być na przykład zamknięcie lub otwarcie zamka). Po ponownym uruchomieniu szkicu po awarii zasilania lub zresetowaniu platformy Arduino system automatycznie odtworzy stan zapisany w pamięci EEPROM.

Szkic

W środowisku Arduino IDE należy wpisać i wysłać następujący szkic. Także tym razem trzeba zastąpić znaki x w obu tablicach w wierszach ❶ i ❷ liczbami składowymi identyfikatorów obu etykiet RFID (tak jak w projekcie nr 52).

```
// Projekt nr 53 — budowa systemu kontroli dostępu RFID z pamięcią „ostatniej akcji”
```

```
#include <EEPROM.h>
int data1 = 0;
int ok = -1;
int lockStatus = 0;
```

```
// identyfikatory etykiet RFID można sprawdzić za pomocą szkicu z listingu 16.1
```

- ```
❶ int tag1[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
❷ int tag2[14] = {x, x, x, x, x, x, x, x, x, x, x, x, x, x};
```



```
int newtag[14] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0}; // tablica używana do
// porównywania odczytanego identyfikatora
```

```
void setup()
{
 Serial.flush();
 Serial.begin(9600);
 pinMode(13, OUTPUT);
 ③ checkLock();
}
```

```
// funkcja comparetag() porównuje dwie tablice i zwraca wartość true, jeśli są identyczne
// funkcja sprawdza się podczas porównywania identyfikatorów RFID
```

```
boolean comparetag(int aa[14], int bb[14])
{
 boolean ff = false;
 int fg = 0;
 for (int cc = 0; cc < 14; cc++)
 {
 if (aa[cc] == bb[cc])
 {
 fg++;
 }
 }
 if (fg == 14)
 {
 ff = true;
 }
 return ff;
}
```

```
void checkmytags()
// porównuje oba zapisane identyfikatory z właśnie odczytanym identyfikatorem
{
```

```
 ok = 0;
 if (comparetag(newtag, tag1) == true)
 {
 ok++;
 }
 if (comparetag(newtag, tag2) == true)
 {
 ok++;
 }
}
```

```
④ void checkLock()
{
 Serial.print("Stan systemu po ponownym uruchomieniu ");
 lockStatus = EEPROM.read(0);
 if (lockStatus == 1)
```

```

 {
 Serial.println("- zablokowany");
 digitalWrite(13, HIGH);
 }
 if (lockStatus == 0)
 {
 Serial.println("- odblokowany");
 digitalWrite(13, LOW);
 }
 if ((lockStatus != 1) && (lockStatus != 0))
 {
 Serial.println("Błąd pamięci EEPROM - zmień płytke Arduino");
 }
}

void loop()
{
 ok = -1;
 if (Serial.available() > 0) // jeśli umieszczono etykietę w sąsiedztwie czytnika,
 {
 // należy odczytać liczbę przekazaną za pośrednictwem pinu szeregowego RX
 delay(100);
 for (int z = 0; z < 14; z++) // odczytuje pozostałe liczby identyfikatora
 {
 data1 = Serial.read();
 newtag[z] = data1;
 }
 Serial.flush(); // zapobiega wielokrotnym odczytom tego samego identyfikatora
 // czas porównać identyfikatory
 checkmytags();
 }
 5 if (ok > 0) // jeśli identyfikatory są takie same
 {
 lockStatus = EEPROM.read(0);
 6 if (lockStatus == 1) // jeśli zablokowany, odblokowuje
 {
 Serial.println("Stan - odblokowany");
 digitalWrite(13, LOW);
 EEPROM.write(0, 0);
 }
 7 if (lockStatus == 0)
 {
 Serial.println("Stan - zablokowany");
 digitalWrite(13, HIGH);
 EEPROM.write(0, 1);
 }
 8 if ((lockStatus != 1) && (lockStatus != 0))
 {
 Serial.println("Błąd pamięci EEPROM - zmień płytke Arduino");
 }
 }
}

```

```

}
else if (ok == 0) // jeśli identyfikatory są różne
{
 Serial.println("Nieprawidłowa etykieta");
 ok = -1;
}
delay(500);
}

```

---

### **Działanie szkicu**

Powyższy szkic jest zmodyfikowaną wersją programu z projektu nr 52. Do symulacji stanu systemu wykorzystano wbudowaną diodę LED, która dobrze demonstruje urządzenie włączane i wyłączane po każdym odczytaniu prawidłowego identyfikatora RFID. Po odczytaniu i dopasowaniu identyfikatora status zamka jest zmieniany w wierszu ⑤. Stan zamka jest zapisywany na pierwszej pozycji w pamięci EEPROM. Stan jest reprezentowany przez wartość liczbową: 0 oznacza odblokowany zamek; 1 oznacza zablokowany zamek. Stan zamka zmienia się (z zablokowanego na odblokowany i odwrotnie) po każdym odczycie prawidłowego identyfikatora RFID w wierszach ⑥ i ⑦.

Szkic zawiera też kod zabezpieczający system na wypadek, gdyby pamięć EEPROM przestała działać. Jeśli wartość zwrócona po odczytaniu zawartości pamięci EEPROM jest różna od 0 i 1, szkic wyświetla odpowiedni komunikat w wierszu ⑧. Co więcej, stan systemu jest sprawdzany po ponownym uruchomieniu szkicu w funkcji checkLock() w wierszach ①, ②, ③ i ④. Funkcja odczytuje wartość zapisaną w pamięci EEPROM, określa na tej podstawie ostatni stan systemu (zamka) i według tego ustawia bieżący stan (zablokowany lub odblokowany).

## **Co dalej?**

W tym rozdziale jeszcze raz wykorzystaliśmy platformę Arduino do budowy prostego urządzenia, które bez tego systemu wymagałoby bardzo złożonego projektu. Wiedza przekazana w tym miejscu wystarczy do stosowania identyfikacji kart RFID w wielu innych projektach i pozwoli tworzyć profesjonalne systemy kontroli dostępu i sterowania cyfrowymi pinami wyjściowymi. Zastosowania tej technologii ponownie zademonstruję w rozdziale 18.



# Skorowidz

## A

- A, 53
- AC, 53
- adres
  - IP, 387, 380
    - modułu sieciowego, 383
    - routera, 383
    - statyczny, 381
  - MAC, 385, 387
  - modułu, 365
- alarm antywłamaniowy, 400
- algorytm, 52, 82
- alternating current, 53
- alternatywa, 91
  - bitowa, 162
  - wykluczająca, 163
- amper, 53
- analog reference, 103
- analogowy
  - sygnał, 98
- analogRead(), 103
- analogWrite, 67, 68
- animacje, 166
- anoda, 57, 69
  - symbol, 75
- antena, 394
- Arduino
  - LilyPad, 249, 251
  - Mega, 249
  - Mega 2560, 252
  - Nano, 249, 251
- Arduino, 19, 37
  - Due, 248, 253
  - LilyPad, 249, 251
  - Mega, 249, 252

- Nano, 249, 251
- piny, 242
- plytka, 20, 24
- plytki alternatywne, 247
- połączenia platformy z komputerem, 25
- programowanie, 25
- przechowywanie danych, 336
- rozszerzanie, 41, 188
- sterowanie z poziomu przeglądarki
  - internetowej, 388
- sterowniki interfejsu USB, 31
- symbol płytki, 75
- szkice platformy, 42
- środowisko programowania, 42
- Uno, 28, 386, 389, 394
- własne płytki, 238
- zdalne sterowanie systemem, 391
- Arduino.app, 27
- AREF, 103
  - tłumienie zakłóceń na pinie, 104
- array, 146
- arytmetyka bitowa, 161
- ATmega2560, 248, 252
- ATmega328, 336
- ATmega328P, 248
  - SMD, 248
- attachInterrupt, 214

## B

- B, 69, 76, 138, 262
- bajt, 139
- Banzy Massimo, 19
- baza, 69
  - symbol, 76

BCD, 365  
bezwładność wzroku, 165  
biblioteka, 196  
  dołączenie, 180  
  I-C, 365  
  instalowanie, 196  
  Mac OS X, 197  
  moduł, 196  
  SerialGSM, 403  
  servo, 258  
  TinyGPS, 293  
  Twitter Arduino, 386  
  Ubuntu Linux, 199  
  VirtualWire, 305  
  Windows, 198  
  Wire, 344  
BIN, 138  
binarne  
  systemy, 137  
  zmienne, 161  
binarycoded decimal, 365  
bit, 90  
bity (porównywanie) 162  
Blu-Tack, 191  
błąd  
  kompilacji, 49  
  w kodzie (znajdowanie), 123  
Boarduino, 250  
boolean variable, 90  
bouncing, 81  
brama sieciowa routera, 380  
breadboard, 58  
brzęczyk  
  piezoelektryczny, 106  
  symbol, 107  
bufor portu szeregowego, 125  
  czyszczenie, 127  
byte, 137, 161

## C

C, 69, 76, 262  
calculateservo, 261  
CAT5E, 380  
CAT6, 380  
cel projektu, 52  
cewka (symbol), 76  
CHANGE, 213  
chip  
  resistors, 56  
  select, 353

circuit diagrams, 74  
clock, 141  
COM, 76  
common, 76  
Cooper Tyler, 21  
crystal oscillator, 240  
CS, 353  
Cuartielles David, 19  
current, 53  
cyfrowy  
  sygnał, 98  
  rezystor nastawnego, 356  
czas  
  pracy, 371  
  wykonywania szkicu, 208  
częstotliwość  
  125 kHz, 331  
  433 MHz, 304  
  migania diody LED, 50  
  pracy modułu GSM, 398  
czołg, 265  
czujnik  
  odległości na podczerwień, 277  
  temperatury, 109  
czyszczenie bufora portu szeregowego, 127

## D

dane  
  rejestrwanie, 201  
  tekstowe (przesyłanie), 304  
  wysłanie do monitora portu szeregowego, 126  
  wysłanie z monitora portu szeregowego, 125, 126  
DC, 53  
DEC, 138  
decimal point, 149  
delay, 48, 50, 62  
DFR0105, 394  
diagram połączeń, 63  
digital storage oscilloscope, 81  
digitalWrite, 62  
dioda  
  LED, 57, 61  
  symbol, 76  
  napięcie przewodzenia, 58  
  oznaczenie, 69  
  prostownicza, 69  
  symbol, 75  
direct current, 53  
DLINE, 291, 312  
długość wiadomości SMS, 404

dodawanie, 102  
  urządzeń, 41  
domyślny stan pinów cyfrowych, 392  
dostosowanie poziomu napięcia, 396  
do-while(), 125  
DP, 149  
drżenie styków, 81, 218  
  zabezpieczenie, 83  
drukowanie szkicu, 43  
DS3232, 361  
DSO, 81  
Duemilanove, 250  
dwójkowy system, 137  
dzielenie, 102  
  napięcia, 104

## E

E, 69, 76, 262  
EEPROM, 336  
  zapisanie wartości w pamięci, 336  
  zewnętrzny, 346  
  żywołność pamięci, 336  
EEPROM.write, 336  
efekty wizualne, 166  
EIA-96 code calculator, 56  
ekran dotykowy, 226  
  budowa, 227  
  łączenie, 226  
  mapa, 229, 234  
  przełącznik, 230  
  testowanie, 229, 233  
ekspander portów, 350, 352  
ELB149C5M, 331  
electrically erasable programmable read-only  
  memory, 336  
elektroniczna kostka do gry, 133  
elektryczność, 53  
  statyczna (pomiar), 132  
element, 146  
EM406, 290  
emiter, 69  
  symbol, 76  
error, 49  
EtherMega, 253  
EtherTen, 380, 386, 389  
expected, 49  
Explorer, 313

## F

FALLING, 213  
false, 90  
fałsz, 90  
farad, 78  
feature creep, 52  
float, 102  
float voltage, 100  
for, 65  
Freeduino, 250  
Freetronics  
  Eleven, 249  
  LCD & Keypad, 293  
Fritzing, 77  
FTDI, 246  
funkcja, 113  
  zwracająca wartość, 116

## G

generowanie sygnałów elektrycznych, 46, 47  
GLCD.  
  CursorTo, 180  
  DrawCircle, 182  
  DrawHoriLine, 182  
  DrawRect, 182  
  DrawRoundRect, 182  
  DrawVertLine, 182  
  FillRect, 182  
  Puts, 180  
  SetDot, 182  
global positioning system, 290  
GND, 53, 77  
gniazdo  
  Ethernet, 380  
  kart pamięci microSD, 380  
GP2Y0A21YK0F, 277  
GPS, 188, 290  
  dane o godzinie, 296  
  odbiornik, 290  
  stan odbiornika, 292  
  szybkość przekazywania danych przez  
    odbiornik, 292  
  testowanie, 291  
GPS-09123, 291  
Greenwich Mean Time, 297  
ground, 53  
GSM  
  częstotliwość pracy, 398  
  status modułu, 398

## GSM

- test, 400
- zasilanie, 394
- rozkaz nawiązania połączenia, 402

## H

- HD44780, 172
- hexTronik HXT900, 257
- HIGH, 48, 62

## I

- I, 59
- I/O, 40
- I-C, 343, 344, 350
  - adres magistrali, 350
- IDE, 25, 42
  - konfiguracja, 27
- if, 88
- if-then, 220
- ikona
  - Monitor portu szeregowego, 44
  - Nowy, 44
  - Otwórz, 44
  - Weryfikuj, 44
  - Załaduj, 44
  - Zapisz, 44
- ilość pamięci dostępnej, 49
- infrared, 319
- INPUT, 47
- instrukcja
  - powtarzanie, 124
  - wykonywane jednorazowe, 46
- int, 64, 161
- integer, 64
- integrated development environment, 25
- interfejs sieci internetowej (Ethernet), 41
- inter-integrated circuit, 343
- interrupt, 213, 214
- IPAddress ip, 384
- IR, 319
- iteracje, 65
- izolacja od obwodu sterującego, 70

## J

- jasność świecenia diody, 233
- jednostki miar (konwersje), 79

## K

- karta
  - dostępu, 329
  - microSD, 201
  - pamięci
    - wpisywanie danych, 202
    - formatowanie, 201
- katoda, 57, 69
  - symbol, 75
- klawiatura numeryczna, 218
  - program obsługi, 219
  - testowanie, 219
- kod 403, 388
  - modyfikowanie, 50
  - paskowy, 54
- kolektor, 69
  - symbol, 76
- komentarz, 45
- kompilowanie, 49
- komponenty
  - wybór, 52
- komunikacja z Arduino
  - za pośrednictwem internetu, 388
- komunikaty środowiska, 44
- kondensator, 78
  - ceramiczny, 79
  - elektrolityczny, 80
  - rozładowanie, 78
  - symbol, 79
- koniunkcja, 91
  - bitowa, 162
- konwersja,
  - jednostek miar, 79
  - liczby binarnej na dziesiętną, 137
- kostka do gry, 133
- KS0066, 172
- KS0108B, 178
- k , 54

## L

- L, 41
- latch, 141
- LCD, 171
  - liczba w systemie
    - binarnym, 175
    - dziesiętnym, 175
    - szesnastkowym, 175
  - moduł graficzny, 178
  - pozycja kursora, 174



- własne znaki, 176
  - wyświetlenie
    - słowa, 175
    - zmiennej, 175
  - wyzerowanie, 174
  - znakowy
    - czarna ramka z białym wypełnieniem, 182
    - koło, 182
    - liczba całkowita, 180
    - odcinek, 182
    - prostokąt z zaokrąglonymi narożnikami, 182
    - tekst, 180
    - włączenie piksela, 182
    - wypełniony prostokąt, 182
  - lcd.
    - begin, 174
    - clear, 174
    - print, 175
    - setCursor, 174
  - least significant bit, 137
  - LED, 40, 57, 61
    - napięcie przewodzenia, 58
    - poziom jasności, 66
    - sterowanie SMS-em, 405
    - symbol, 76
    - świecenie, 48
    - wyłączenie, 48
  - liczba
    - binarna 137, 140
    - całkowita, 64, 127
    - iteracji, 65
    - losowa, 132, 143
    - obrotów wału na minutę, 261
    - wyświetlanie, 149
  - light-emitting diodes, 40
  - Lindsay Philip, 20
  - linia
    - danych, 344
    - zegara, 344
  - liniowy
    - potencjometr, 105
    - regulator napięcia, 239
  - Linux, 25
    - Ubuntu, 33
  - liquid crystal display, 171
  - logarytmiczny potencjometr, 105
  - long, 127
  - loop, 47
  - LOW, 48, 62, 213
  - LSB, 137
  - LSBFIRST, 150
  - lutowanie, 194
- Ł**
- ładowanie szkicu, 43
  - ładunek elektryczny, 78
- M**
- MAC adres, 385
  - Mac OS X, 25
    - biblioteka, 197
  - magistrala
    - danych, 343
    - I-C, 344
    - SPI, 353
  - mapa ekranu dotykowego, 229
  - masa klejąca wielokrotnego użytku, 191
  - master-in, slave-out, 353
  - master-out, slave-in, 353
  - MCP23017, 350, 351, 352
  - MCP4162, 356
  - microSD, 188, 201
    - testowanie, 201
    - zapisywanie danych, 202
  - microSDHC, 201
  - microswitch, 273
  - miernik uniwersalny, 56
  - mierzenie
    - napięcia, 56
    - natężenia prądu, 56
    - oporu elektrycznego, 56
  - miganie
    - diody, 45
    - dostrzegalne, 66
  - mikrofarad, 78
  - mikrokontroler, 39, 41, 240
    - wyjmowanie, 244
  - mikroprzełącznik, 273
  - milliamper, 53
  - millis, 208
  - miniaturowy rezystor nastawny, 106
  - MISO, 353
  - mnożenie, 102
  - moc, 53
    - znamionowa, 56
  - model mostu z sygnalizatorami, 92
  - modem, 380
  - modulacja szerokości impulsu, 66, 108, 236, 271
  - modulo, 155

moduł, 41, 188  
adres, 365  
biblioteki, 196  
dodatkowe, 190  
GSM  
  częstotliwość pracy, 398  
  zasilanie, 394  
  SM5100, 394  
komunikacji radiowej, 310  
niestandardowy, 191  
radiowy, 304  
samodzielne konstruowanie, 190  
sieciowy, 379  
zasilania (przygotowanie), 395  
modyfikowanie kodu, 50  
monitor portu szeregowego, 120, 143, 349  
monitorowanie  
  stanu pinów cyfrowych, 195  
  zdalne, 381  
montaż wielu modułów, 190  
MOSI, 353  
most significant bit, 137, 354  
MSB, 137, 354  
MSBFIRST, 153  
multimetr, 56  
myservo.write, 258  
M , 54

## N

najbardziej znaczący bit, 137, 354  
najmniej znaczący bit, 137  
napięcie, 53, 59  
  dzielenie, 104  
  referencyjne, 103  
  regulator, 239  
  skok, 71  
  stabilizacja, 80  
  wewnętrzne, 104  
natężenie prądu, 53, 59  
  ograniczanie, 54  
nazwa  
  pliku (szkic), 43  
  stosowanej płytki, 44  
NC, 76  
negacja, 90  
  bitowa, 163, 164  
Nintendo DS, 226  
Niski stan, 99

NO, 76  
noInterrupts, 214  
normally  
  closed, 76  
  open, 76  
NPN, 76  
numer  
  portu  
    cyfrowego, 47  
    USB, 44  
  telefonu odbiorcy SMS, 404



obraz  
  odwracanie, 168  
  wyświetlanie, 167  
obsługa przerwań, 214  
obszar  
  komunikatów, 42, 44  
  poleceń, 42, 43  
  tekstu, 42, 44  
obwód  
  filtrujący, 83  
  ochrona, 71  
  własny, 237  
odbiornik  
  podczerwieni, 320  
  testowanie, 321  
odejmowanie, 102  
odwracanie obrazu, 168  
OE, 141  
określanie  
  godziny, 290  
  położenia, 290  
  przybliżonej szybkości, 290  
om, 54  
Oomlout, 22  
open source, 254  
operator  
  alternatywy, 91  
  koniunkcji, 91  
  negacji, 90  
  porównania, 103  
opornik, 54  
oporność, 54, 59  
oprogramowanie, 42  
oscyloskop cyfrowy z pamięcią, 81  
OUTPUT, 47, 61  
output enable, 141

## P

- pamięć
  - dostępna, 49
  - EEPROM, 248
  - flash, 248
  - SRAM, 248
- Parallax Ping, 282
- Pasmo (aktualne ustawienia), 398
- persistence of vision, 165
- pętla for, 65
- pF, 78
- piezoelektryczny brzęczyk, 106
- pikofarad, 78
- pilot, 321
- pin
  - cyfrowy, 46
    - stan domyślny, 392
    - zwiększanie liczby, 138
- numer 1, 241
- pinMode, 47, 61
- plan przedpłacony, 394
- planowanie działań, 52
- plytka
  - prototypowa, 190
  - uniwersalna, 58
- PNP, 76
- pobieranie danych ze zdalnego czujnika, 314
- podczerwień, 319
- podłączanie dodatkowych płytek, 41
- Pololu RP5, 265
- połączenia (symbol), 77
- położenie (rejestrowanie), 298
- pomiar (precyzja), 103
- ponowne uruchamianie systemu, 41
- porównywanie bitów, 162
- port
  - szeregowy, 40, 44
    - monitor, 120
  - USB, 380
- potencjometr, 105
- POV, 165
- PowerSwitch Tail, 112
- powiadomienia o powrocie dzieci ze szkoły, 386
- powtarzanie instrukcji, 124
- poziom jasności diod LED, 66
- prawda, 90
- prawo Ohma, 58, 59
- prąd
  - blądzący, 71, 265
  - maksymalne natężenie, 68
  - natężenie, 53
  - przełączanie, 69
  - sieciowy (przełączanie), 72
  - stały, 53
  - zmienny, 53
- precyzyjny zegar, 296
- prepaid, 394
- prędkość podróży, 298
- program (wstrzymanie wykonywania), 213
- projekt (cel), 52
- projektowania systemów, 52
- ProtoScrewShield, 367
- ProtoShield, 190
  - RESET, 194
  - układ otworów, 192
  - projekt obwodu, 193
- przegrzanie rezystora, 56
- przekazywanie informacji pomiędzy płytką a komputerem, 50
- przełącznik, 70, 71
  - symbol, 76
  - zastosowanie, 71
- przekierowanie portu przy użyciu publicznego adresu IP, 386
- przełączanie prądu, 69, 72
- przełącznik dotykowy, 233
- przepustowość, 121
- przerost funkcji, 52
- przerwanie, 213
  - tryb, 213
- przesunięcie
  - w lewo, 163
  - w prawo, 163
- przesyłanie danych tekstowych, 304
- przetwarzanie dużej liczby powiązanych danych, 146
- przewody, 60
  - symbol, 77
- przewód sieciowy, 380
- przycisk symbol, 82
- pull-
  - down resistor, 83
  - width modulation, 66
- PWM, 66, 236
- PWRIN, 396
- PWROUT, 396

## Q

quiz, 143

## R

R, 59  
radio-frequency identification, 329  
random, 132  
randomSeed, 132  
readKeypad, 223  
reakcja na zdarzenie, 213  
real-time clock, 361  
reference voltage, 103  
regulator napięcia, 239  
rejestr przesuwający, 138  
  zasada działania, 139  
rejestrowanie  
  danych, 201  
  położenia, 298  
  ruchu samochodu, 298  
RESET, 39, 41  
  ProtoShield, 194  
reszta z operacji dzielenia, 155  
revolutions per minute, 261  
rezonator kwarcowy, 240  
rezystancja, 54  
rezystor, 54, 61  
  do montażu powierzchniowego, 56  
  miniaturowy, 106  
  moc znamionowa, 56  
  nastawny cyfrowy, 356  
  nastawny, 105, 356  
  ograniczający natężenie prądu, 58  
  przegrzanie, 56  
  symbol, 75  
  ściągający, 83  
RFID, 329  
  budowa, 330  
  czytnik, 331  
  etykiety, 331  
  identyfikator etykiety, 336  
  system kontroli dostępu, 333  
  testowanie, 331, 332  
RFR103B2B, 331  
RISING, 213  
robot  
  czujnik podczerwieni do wykrywania kolizji, 280  
  gąsienicowy, 265  
  możliwe ruchy, 271  
  zderzenia z obiektami, 273  
rozmiar tablicy, 146  
RPM, 261  
RTC, 361  
RTL-10709, 290

ruch wahadłowy, 92  
RX, 41, 50  
rzut kostką do gry, 134

## S

S, 82  
SAM3X8E, 248  
Sarik John, 22  
SBAND, 399  
schemat obwodu, 74  
schematic diagrams, 74  
Schulz Kurt, 21  
SCK, 353  
SCL, 344, 362  
Scooterputer, 21  
SDA, 344, 362  
seed, 132  
Serial.  
peripheral interface, 343  
  available(), 126  
  begin, 121  
  flush(), 127  
  GSM, 403  
  print(), 138  
  println, 122  
servo, 256  
servomechanism, 256  
servomotor, 256  
shields, 188  
shift register, 138  
shiftOut(), 150  
siedmiosegmentowy wyświetlacz, 148  
silnik, 257  
  elektryczny, 261  
  dodatkowe źródła zasilania, 263  
  napięcie zasilania, 261  
  prąd  
    bez obciążenia, 261  
    zatrzymania, 261, 263  
  sterowanie, 262, 264  
  szybkość przy napięciu zasilacza, 261  
silnik  
  wykonawczy, 256  
    łączenie, 257  
    natężenie pobieranego prądu, 256  
    szybkość, 256  
    uruchamianie, 257  
    wymuszenie ruchu, 258  
    zakres obrotu, 256  
SIM, 394

sketch, 37  
skok napięcia, 71  
slave select, 353  
SM5100, 394  
smartfon, 385  
SMS, 394  
    długość wiadomości, 404  
    w odpowiedzi na zdarzenie, 402  
SPI, 343, 353  
    połączenie, 353  
sprzętu wybór, 52  
SS, 353  
stabilizacja napięcia, 80  
stan  
    niski, 99  
    wysoki, 48, 99  
status modułu GSM, 398  
sterowanie  
    drogą radiową, 305  
    SMS-em, 405  
    urządzeniami zasilanymi siecią, 112  
    za pomocą pilota na podczerwień, 323  
stoper, 210  
strona internetowa, 381  
    z linkami w formie przycisków, 392  
styki  
    drżenie, 81  
    przekaznika (symbol), 76  
Sudoku, 22  
switch bounce, 81  
switch-case, 220  
sygnalizacja świetlna, 92  
sygnał  
    analogowy, 98  
    cyfrowy, 98  
symbol uśmiechniętej buzi, 176  
system  
    binarny, 137, 365  
    kontrola dostępu, 329  
    mierzenia czasu pracy, 371  
    nawigacji satelitarnej, 290  
szerokość i długość geograficzna, 295  
szkic, 42, 45, 52, 87, 94, 114, 134  
    czas wykonywania, 208  
    debugowanie, 123  
    drukowanie, 43  
    elastyczność, 64  
    komentarz, 45  
    kopiowanie, 43  
    ładowanie, 43  
    maksymalny rozmiar, 385

port USB, 43  
przeszukiwanie, 43  
testowy, 244  
uruchamianie, 63  
weryfikacja, 43, 49  
wklejanie, 43  
wybór typu płytki, 43  
zapisywanie, 43  
szybkość działania mikrokontrolera, 240

## Ś

środowisko programowania, 42  
wersja, 43

## T

tablica, 146  
    odczytywanie, 147  
    rozmiar, 146  
    zapisywanie, 147  
tekst  
    wysyłanie do monitora portu szeregowego, 121  
Teleduino, 388  
    klucz, 391  
    konfiguracja, 389  
    liczba mignięć LED, 391  
    stan usługi, 391  
temperatura  
    pomiar, 108  
    rejestracja, 205  
    w zamrażarce, 400  
Terminal, 313  
termometr, 117  
    analogowy, 259  
    cyfrowy, 156  
    LCD znakowy, 183  
    zdalnie sterowany, 314  
tester baterii, 99  
TinyGPS, 300  
TIP120, 262  
TMP36, 315  
tolerancja rezystora, 55  
transmisja bezprzewodowa, 303  
 tranzystor, 68, 71  
    Darlingtona, 262  
    NPN, 76  
    PNP, 76  
    symbol, 76  
    wyłączenie, 69  
trimpot, 106, 172  
true, 90

tryb danych wyjściowych, 47  
trymer, 106  
TSOP4138, 321  
TWI, 344  
Twitter, 22  
    token, 387  
    wpisy, 386  
    wymagania na wpisy, 388  
two wire interface, 344  
TwypeWriter, 22  
TX, 41, 50

## U

U, 59  
UART, 291, 292, 312  
Ubuntu Linux, 33  
    biblioteka, 199  
ultradźwiękowy czujnik odległości, 282  
urządzenia  
    dodawanie, 41  
    nadrzędne, 344  
    podrzędne, 344  
USB, 25, 39, 40  
ustawianie daty i godziny w układzie zegara  
    czasu rzeczywistego, 362  
UTC, 297  
uziemiaenie, 53  
    symbol, 77

## V

V, 53  
variables, 64  
VirtualWire, 305  
void  
    loop, 62, 204  
    setup, 46, 47, 62  
voltage, 53

## W

W, 53  
W5100, 379  
wartość początkowa, 132  
warunek  
    sprawdzanie, 124  
    po wykonaniu kodu, 125  
wał, 53  
wejścia  
    analogowe, 39  
    -wyjście, 40

wersja środowiska IDE, 43  
weryfikacja szkicu, 49  
wewnętrzne napięcie referencyjne, 104  
while(), 124  
wielokrotne wykonanie instrukcji, 47  
Wi-Fi, 385  
Windows, 8, 25, 29  
    biblioteka, 198  
Windows XP, 25  
Wire, 344  
WLS107B4B, 304  
własny obwód, 237  
włączanie klimatyzacji SMS-em, 408  
wolt, 53  
wpisy na Twitterze, 386  
WRL-08687, 311  
WRL-10534, 304  
wspólna  
    anoda, 149  
    katoda, 149  
współczynnik wypełnienia impulsu, 66  
wstrzymanie wykonywania  
    programu, 213  
    szkicu, 48  
wykrywanie sygnałów elektrycznych, 46, 47  
wysoki stan, 99  
wyświetlacz, 21  
    LED (sterowanie), 150  
    liczb binarnych, 140  
    matrycowy, 158  
    siedmiosegmentowy, 148  
wyświetlanie  
    daty i godziny na sznakowym module LCD, 367  
    na module LCD współrzędnych  
        reprezentujących bieżącą pozycję, 293  
wzmacniacz dźwięku, 105

## X

XBee, 305, 310, 315  
    Explorer, 311

## Z

zalanie piwnicy, 408  
zamek z klawiaturą numeryczną, 221  
zapisywanie  
    szkicu, 43  
    zmodyfikowanego szkicu, 118  
zasilanie, 39  
zatrząsk, 141

- zdalne sterowanie robota, 325
- zegar, 141
  - czasu rzeczywistego, 361
  - ustawianie daty i godziny, 362
- precyzyjny, 296
- zewnętrzna pamięć EEPROM, 346

- zintegrowane
  - środowisko programowania, 25, 42
  - kontroler sieciowy, 380
- złącza, 39
- zmienna, 64
  - logiczna, 90
  - tablicowa, 389
  - wyświetlanie wartości, 122
- znakowy moduł LCD, 173





# 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>



GENIALNA LEKTURA DLA KAŻDEGO PASJONATA ELEKTRONIKI!

Arduino to platforma, dzięki której świat elektroniki zyskał wiele nowych, ciekawych rozwiązań i możliwości. Prostota obsługi, świetna dokumentacja oraz specjalnie zaprojektowane środowisko do tworzenia oprogramowania sprawiły, że projekt ten zdobył tysiące sympatyków. Taki sukces przełożył się na ilość dostępnych akcesoriów oraz instrukcji, dzięki którym możesz zbudować dowolny układ elektroniczny.

W tej książce zebrano 65 interesujących projektów o różni-cowanym stopniu trudności. Dzięki nim błyskawicznie opanujesz zasady wykorzystania platformy oraz zbudujesz urządzenia, które przydadzą się w codziennym życiu. Pierwsze projekty pozwolą Ci zapoznać się z Arduino — jeden z nich polega na przykład na tworzeniu fali migających diod LED. Wykonanie kolejnych pozwoli Ci osiągnąć wyższy stopień wtajemniczenia: sterowanie ruchem samochodowym, testowanie baterii, elektroniczna kostka do gry, pomiar temperatury czy wykorzystanie systemu GPS to tylko niektóre z nich. Książka ta jest obowiązkową lekturą dla wszystkich, którzy chcą poznać tajniki platformy Arduino i zbudować niesamowite układy elektroniczne.

### Zbuduj swój własny:

- tester baterii
- odbiornik GPS
- termometr cyfrowy
- stoper
- czytnik RFID



**helion.pl**  
księgarnia  
internetowa

Nr katalogowy: 16342



Księgarnia internetowa  
<http://helion.pl>



Zamówienia telefoniczne:  
**0 801 339900**  
**0 601 339900**



**Helion**

Sprawdź najnowsze promocje:

- <http://helion.pl/promocje>
- Książki najchętniej czytane:
- <http://helion.pl/bestsellery>
- Zamów informacje o nowościach:
- <http://helion.pl/nowosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po **WIĘCEJ**



KOD KORZYSCI

ISBN 978-83-246-7999-7



9 788324 679997

Cena: 69,00 zł

Informatyka w najlepszym wydaniu