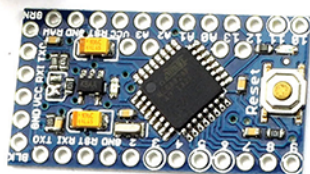
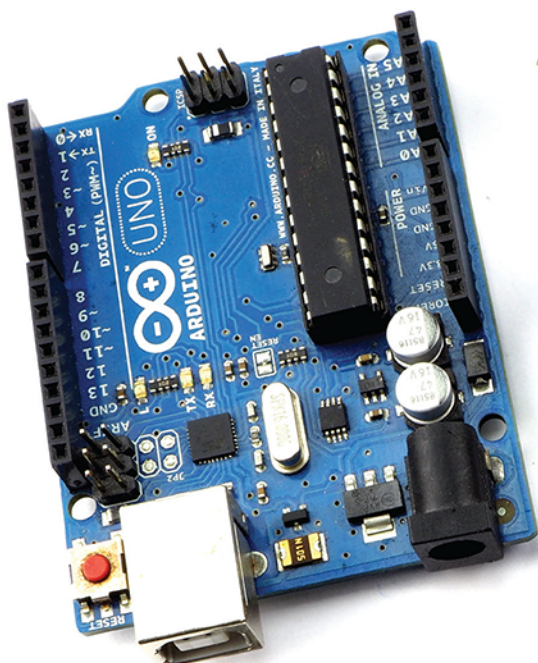
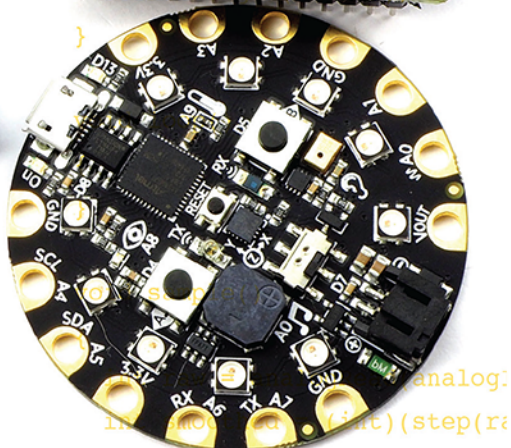
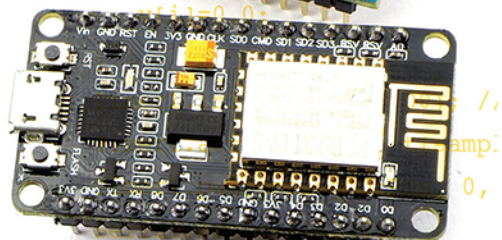


Arduino® dla początkujących

Kolejny krok

```
#3
const byte PS_128 = (1 << ADPS0);
const byte PS_16 = (1 << ADPS1);
float v[5];
```

```
void setup()
{
  DCSRA &= ~PS_128; // remove PS_128
  DCSRA |= PS_16; // add PS_16
  pinMode(0, INPUT);
}
```



Tytuł oryginału: Programming Arduino Next Steps: Going Further with Sketches, Second Edition

Tłumaczenie: Anna Mizerska

z wykorzystaniem fragmentów Arduino dla początkujących. Kolejny krok
w przekładzie Konrada Matuka

ISBN: 978-83-283-7548-2

Original edition copyright © 2019, 2014 by McGraw-Hill Education. All rights reserved.

Polish edition copyright © 2021 by Helion SA. All rights reserved.

McGraw-Hill Education, the McGraw-Hill Education logo, TAB, and related trade dress are trademarks or registered trademarks of McGraw-Hill Education and/or its affiliates in the United States and other countries and may not be used without written permission.

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.

Autor oraz Helion SA 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 Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/arpok2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

O autorze	9
Podziękowania	10
Przedmowa	11
Wstęp	13
Rozdział 1. Programowanie Arduino	17
Czym jest Arduino	17
Instalacja i środowisko programistyczne	20
<i>Instalacja środowiska programistycznego</i>	20
<i>Blink</i>	20
Wycieczka po płytce Arduino	23
Zasilanie	23
Złącza zasilania	24
Wejścia analogowe	24
Złącza cyfrowe	24
Płytki Arduino	25
<i>Uno i pochodne</i>	25
<i>Duże płytki Arduino</i>	26
<i>Małe płytki Arduino</i>	27
<i>Nieoficjalne płytki Arduino</i>	28
Język programowania	29
Modyfikacja szkicu Blink	29
Zmienne	31
If	32
Pętle	32
Funkcje	34
Wejścia cyfrowe	35
Wyjścia cyfrowe	37
Monitor portu szeregowego	37
Tablice i macierze	39
Wejścia analogowe	40

Wyjścia analogowe	42
Korzystanie z bibliotek	44
Typy danych obsługiwane przez Arduino	46
Polecenia Arduino	47
Podsumowanie	49
Rozdział 2. Pod maską	51
Krótka historia Arduino	51
Anatomia Arduino	52
Procesory AVR	52
<i>ATmega328</i>	53
<i>ATmega32u4</i>	53
<i>ATmega2560</i>	55
<i>AT91SAM3X8E</i>	55
Arduino i Wiring	55
Od szkicu do Arduino	59
AVR Studio	61
Instalacja programu rozruchowego	62
<i>Instalacja programu rozruchowego za pomocą aplikacji AVR Studio i programatora</i>	63
<i>Instalacja programu rozruchowego za pomocą zintegrowanego środowiska programistycznego Arduino i drugiej płytki Arduino</i>	64
Podsumowanie	66
Rozdział 3. Kiedy Arduino to nie Arduino?	67
Rozszerzalna architektura Arduino IDE	67
Adafruit Circuit Playground Express	68
NodeMCU	71
ESP32	73
Mikrokontrolery ATtiny	73
<i>ATtiny44</i>	73
<i>Arduino jako programator</i>	74
<i>Instalacja ATtinyCore w IDE</i>	75
<i>Zegary, kryształy i bezpieczniki</i>	76
<i>Minimalne Arduino</i>	77
Podsumowanie	77
Rozdział 4. Przerwania i zegary	79
Przerwania sprzętowe	79
<i>Piny przerwań</i>	82
<i>Tryby przerwań</i>	83
<i>Aktywacja wbudowanego rezystora podciągającego</i>	83
<i>Procedury obsługi przerwań</i>	84
<i>Zmienne ulotne</i>	84
<i>Podsumowanie wiadomości na temat procedur obsługi przerwań</i>	85
Włączanie i wyłączanie obsługi przerwań	85
Zegary i przerwania	86
Podsumowanie	89

Rozdział 5. Przyspieszanie Arduino	91
Jak szybko działa Twoje Arduino?	91
Porównanie płytek Arduino	92
Przyspieszanie wykonywania operacji arytmetycznych	93
<i>Czy naprawdę musisz stosować wartości typu float?</i>	93
Przeglądanie kontra obliczanie	94
Szybkie wejścia-wyjścia	96
<i>Podstawowa optymalizacja kodu</i>	97
<i>Bajty i bity</i>	98
<i>Porty układu ATmega328</i>	98
<i>Bardzo szybkie działanie wyjść cyfrowych</i>	100
<i>Szybkie wejścia cyfrowe</i>	100
Przyspieszanie wejść analogowych	102
Podsumowanie	103
Rozdział 6. Arduino i mały pobór prądu	105
Płytki Arduino i pobór prądu	105
Prąd i akumulatory	107
Zmniejszenie częstotliwości taktowania	108
Wyłączanie komponentów	109
Usypianie Arduino opartych na ATmega	111
<i>Biblioteka Narcoleptic</i>	111
<i>Budzenie za pomocą zewnętrznych przerwań</i>	113
Usypianie ESP8266	115
Usypianie ESP32	116
Ograniczanie pobieranego prądu za pomocą wyjść cyfrowych	118
Podsumowanie	120
Rozdział 7. Pamięć	121
Pamięć Arduino	121
Korzystanie z minimalnej ilości pamięci RAM	123
<i>Korzystanie z właściwych struktur danych</i>	123
<i>Przechowywanie w pamięci flash stałych będących łańcuchami</i>	124
<i>Rozpowszechnione błędne przekonania</i>	124
<i>Pomiar wolnej pamięci</i>	124
Korzystanie z minimalnej ilości pamięci flash	125
<i>Korzystaj ze stałych</i>	125
<i>Usuwać zbędne elementy szkicu</i>	125
<i>Pomiń program rozruchowy</i>	126
Stacyczna i dynamiczna alokacja pamięci	126
Łańcuchy	127
<i>Tablice elementów typu char</i>	128
<i>Biblioteka Arduino StringObject</i>	130
Korzystanie z pamięci EEPROM	131
<i>Przykład korzystania z pamięci EEPROM</i>	132
<i>Korzystanie z biblioteki avr/eeprom.h</i>	134
<i>Ograniczenia pamięci EEPROM</i>	136

Korzystanie z pamięci Flash	136
Zapisywanie danych na kartach SD	138
Podsumowanie	139
Rozdział 8. Interfejsy Arduino	141
System binarny	141
Typy danych Arduino i system binarny	142
System szesnastkowy	143
Maskowanie bitów	144
Przesunięcie bitowe	145
Dane przesyłane szeregowo	146
Podsumowanie	150
Rozdział 9. Korzystanie z magistrali I2C	151
Warstwa sprzętowa	153
Protokół magistrali I2C	154
Biblioteka Wire	155
<i>Inicjacja magistrali I2C</i>	<i>155</i>
<i>Wysyłanie danych przez urządzenie nadrzędne</i>	<i>155</i>
<i>Odbieranie danych przez urządzenie nadrzędne</i>	<i>155</i>
Przykład działania magistrali I2C	156
<i>Radio FM TEA5767</i>	<i>156</i>
<i>Przesyłanie danych pomiędzy dwoma płytkami Arduino</i>	<i>158</i>
<i>Płytki z diodami LED</i>	<i>161</i>
<i>Zegar czasu rzeczywistego DS1307</i>	<i>162</i>
Podsumowanie	163
Rozdział 10. Praca z urządzeniami wyposażonymi w interfejs 1-Wire	165
Sprzęt obsługujący interfejs 1-Wire	166
Protokół 1-Wire	166
Biblioteka OneWire	167
<i>Inicjalizowanie biblioteki OneWire</i>	<i>167</i>
<i>Skanowanie magistrali</i>	<i>167</i>
Korzystanie z układu DS18B20	168
Podsumowanie	171
Rozdział 11. Praca z urządzeniami wyposażonymi w interfejs SPI	173
Operowanie bitami	173
Sprzęt obsługujący magistralę SPI	174
Protokół SPI	175
Biblioteka SPI	175
Przykład komunikacji za pomocą interfejsu SPI	177
Podsumowanie	180

Rozdział 12. Szeregowa transmisja danych za pośrednictwem układu UART	183
Sprzęt służący do szeregowej transmisji danych	183
Protokół obsługujący szeregową transmisję danych	186
Polecenia służące do obsługi szeregowej transmisji danych	186
Biblioteka SoftwareSerial	189
Przykłady szeregowej transmisji danych	190
<i>Komunikacja pomiędzy komputerem a Arduino za pośrednictwem interfejsu USB</i>	190
<i>Komunikacja pomiędzy dwoma płytkami Arduino</i>	192
<i>Moduł GPS</i>	194
Podsumowanie	197
Rozdział 13. Obsługa interfejsu USB	199
Emulacja klawiatury i myszy	199
<i>Emulacja klawiatury</i>	200
<i>Przykład emulacji klawiatury</i>	201
<i>Emulacja myszy</i>	201
<i>Przykład emulacji myszy</i>	202
<i>Host USB płytki Arduino Due</i>	202
Podsumowanie	205
Rozdział 14. Obsługa sieci oraz internet rzeczy	207
Sprzęt sieciowy	207
<i>Płytką rozszerzeń wyposażoną w kontroler sieci Ethernet</i>	207
<i>Arduino Ethernet i Arduino EtherTen</i>	208
Biblioteka Ethernet	209
<i>Nawiązywanie połączenia</i>	209
<i>Stawianie serwera sieci Web</i>	211
<i>Tworzenie żądań</i>	211
Przykład szkicu korzystającego z biblioteki Ethernet	212
<i>Sprzętowy serwer sieci Web</i>	212
<i>Pobieranie danych w formacie JSON</i>	216
Oficjalna biblioteka Arduino do obsługi WiFi	219
<i>Nawiązywanie połączenia</i>	219
<i>Funkcje zdefiniowane w bibliotece WiFi</i>	220
Przykładowy szkic korzystający z sieci Wi-Fi	220
Przykład ESP8266/ESP32 WiFi	221
Internet rzeczy	223
<i>dweet.io</i>	224
<i>Programowanie NodeMCU lub Wemos D1 Mini</i>	225
<i>Podłączanie TMP36</i>	226
<i>Strona internetowa wyświetlająca temperaturę</i>	226
Podsumowanie	228

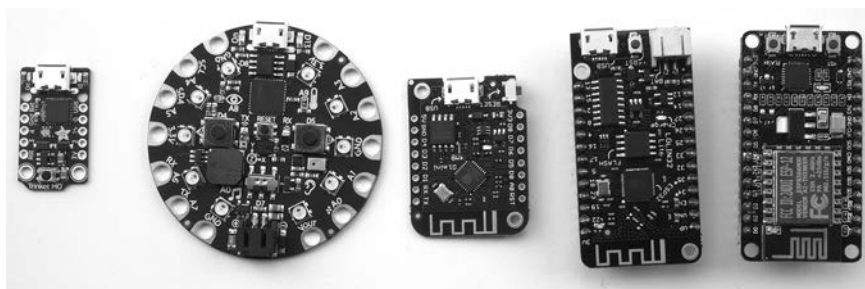
Rozdział 15. Cyfrowe przetwarzanie sygnałów	229
Wprowadzenie do cyfrowego przetwarzania sygnałów	229
Uśrednianie odczytów	230
Wstęp do filtrowania	232
Prosty filtr dolnoprzepustowy	233
Cyfrowe przetwarzanie sygnałów przez Arduino Uno	234
Cyfrowe przetwarzanie sygnałów przez Arduino Due	235
Generowanie kodu filtrującego	238
Transformacja Fouriera	241
<i>Analizator spektrum</i>	241
<i>Pomiar częstotliwości</i>	244
Podsumowanie	244
Rozdział 16. Praca z użyciem tylko jednego procesu	247
Zmiana skali	247
Dlaczego wątki są zbędne	248
Funkcje setup i loop	248
<i>Najpierw wykrywaj, a dopiero później reaguj</i>	248
<i>Pauza, która nie blokuje mikrokontrolera</i>	249
Biblioteka Timer	250
Diagramy stanów	251
Automaty skończone w Arduino	253
Podsumowanie	254
Rozdział 17. Tworzenie bibliotek	255
Kiedy należy tworzyć biblioteki?	255
Stosowanie klas i metod	256
Przykładowa biblioteka TEA5767 Radio	256
<i>Określ interfejs programistyczny</i>	257
<i>Utwórz plik nagłówkowy</i>	258
<i>Utwórz plik implementacji</i>	259
<i>Utwórz plik ze słowami kluczowymi</i>	259
<i>Utwórz folder z przykładami</i>	260
Testowanie biblioteki	260
Publikacja biblioteki	261
Publikacja biblioteki na GitHubie	261
<i>GitHub</i>	261
<i>Tworzenie repozytorium</i>	262
Podsumowanie	264
Dodatek A. Podzespoły	265
Płytki Arduino	265
Komponenty i moduły	265
Dystrybutorzy	266
Zestawy startowe Arduino	267

Rozdział 3.

Kiedy Arduino to nie Arduino?

Zespół Arduino wykonał świetną pracę, przez co środowisko programistyczne Arduino jest modułowe i można łatwo dodawać do niego nowe płytki. Co za tym idzie, możliwe jest dodanie nowego oprogramowania wspierającego płytki kompatybilne z Arduino i programowanie ich za pomocą IDE z użyciem tych samych poleceń.

Możesz programować nie tylko gotowe płytki, które widać na rysunku 3.1, lecz możesz również użyć Arduino IDE do programowania samych mikrokontrolerów (innych niż te zastosowane w płytkach Arduino).



Rysunek 3.1. Od lewej do prawej: *Trinket m0*, *Circuit Playground Express*, *Wemos D1 Mini*, *LOLIN32*, *NodeMCU*

Rozszerzalna architektura Arduino IDE

Arduino IDE, a tak naprawdę cały projekt Arduino, ma otwarte źródło (licencja *open source*). Oznacza to, że możesz przeanalizować i skopiować na własne potrzeby cały kod Arduino IDE i kod, który uruchamiany jest na płytce Arduino. Mimo że formalności kłóciłoby się ze stwierdzeniem, że sprzęt Arduino całkowicie spełnia założenia licencji *open source*, faktem jest, że wszystkie schematy i informacje potrzebne do zrobienia własnego Arduino są ogólnie dostępne.

Ponadto Arduino IDE jest elastyczne i nie ma tak naprawdę potrzeby zmiany jego kodu źródłowego. Jednym z kluczowych powodów takiego rozwiązania jest możliwość łatwego i szybkiego podłączenia innych platform lub jednostek bazowych do Arduino IDE.

Najprostszym sposobem na dodanie sterownika innej płytki jest skorzystanie z menedżera płytek (rysunek 3.2), który znajduje się w *Narzędzia/Płytki/Menedżer płytek*.



Rysunek 3.2. Menedżer płytek

Domyślnie nie znajdziesz tam wszystkich płytek, a szczególnie tych mniej popularnych, pochodzących z mniej oficjalnych źródeł. Możesz je jednak dodać poprzez wprowadzenie adresu URL w polu *Dodatkowe adresy URL do menedżera płytek* w oknie preferencji, które otworzysz z poziomu menu *Plik* (rysunek 3.3).

W kolejnym podrozdziale dodamy wsparcie dla popularnej płytki Adafruit Circuit Playground.

Adafruit Circuit Playground Express

Płytką Adafruit Circuit Playground (rysunek 3.4) może być programowana za pomocą wielu różnych języków i środowisk, w tym Arduino. Po części płytką została zaprojektowana do celów edukacyjnych, więc ma już wbudowanych kilka urządzeń peryferyjnych, między innymi: 10 diod LED RGB „Neopixel”, brzęczyk, dwa przyciski, akcelerometr i mikrofon.

W przeciwieństwie do Arduino ta płytką nie ma procesora z rodziny Mega i jej mikrokontroler oparty jest na architekturze Arm. Teraz użyjemy tej płytki, aby pokazać zastosowanie menedżera płytek.

Otwórz menedżera płytek, w polu wyszukiwania wpisz **Circuit Playground Express** (rysunek 3.5) i wybierz pierwszą pozycję z listy. Zapewne zauważysz, że ten sterownik daje wsparcie dla dużej liczby płytek, w tym Circuit Playground Express. Naciśnij przycisk *Instaluj*, który znajduje się obok informacji o wersji.

Preferencje

Ustawienia Sieć

Lokalizacja szkieletownika:

Język edytora: (wymagany restart Arduino)

Rozmiar czcionki edytora:

Powiększenie interfejsu: automatyczne % (wymagany restart Arduino)

Motyw: (wymagany restart Arduino)

Pokaż szczegółowe informacje podczas: kompilacji wgrzywania

Ostrzeżenia kompilatora:

Wyświetl numery lini Włącz zawijanie tekstu

Zweryfikuj kod po wgraniu Użyj zewnętrznego edytora

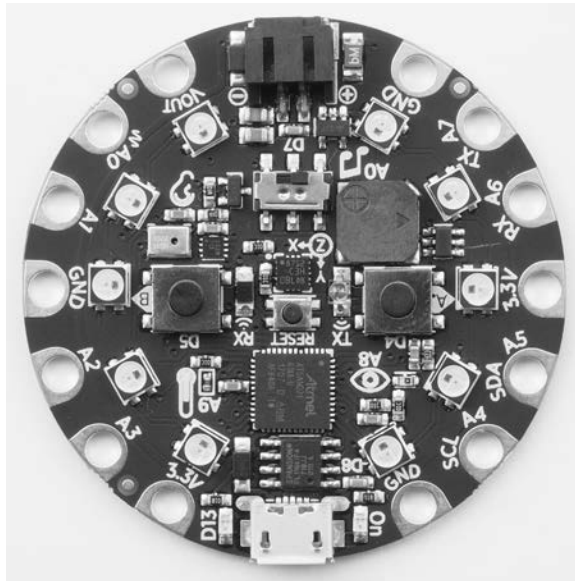
Sprawdź aktualizację przy starcie Zapisuj przy weryfikacji lub ładowaniu

Use accessibility features

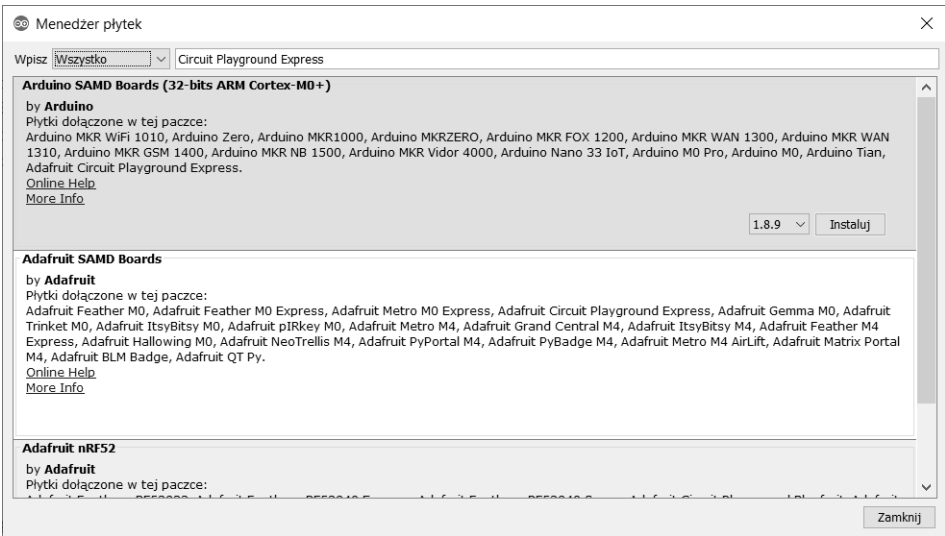
Dodatkowe adresy URL do menedżera płytek:

Więcej preferencji może być edytowanych bezpośrednio w pliku
 C:\Users\Laptop\AppData\Local\Arduino15\preferences.txt
 (edytuj tylko kiedy Arduino nie pracuje)

Rysunek 3.3. Dodawanie adresów URL do menedżera płytek

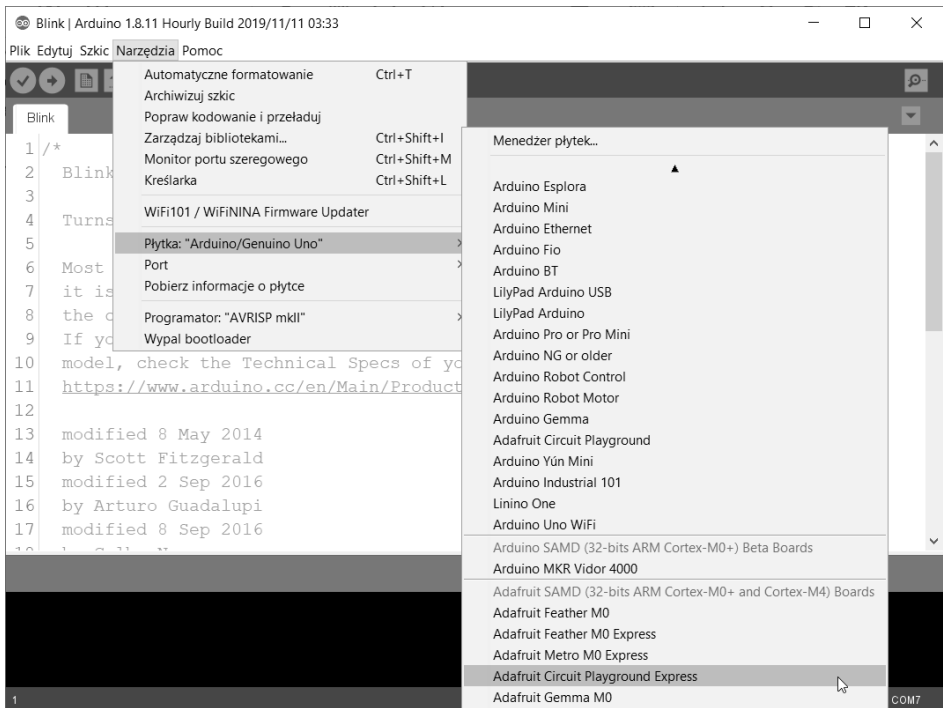


Rysunek 3.4. Płytki Adafruit Circuit Playground



Rysunek 3.5. Dodawanie płytki za pomocą menedżera płytek

Po zainstalowaniu sterownika na liście dostępnych płytek powinna pojawić się płytka Circuit Playground Express (rysunek 3.6).



Rysunek 3.6. Circuit Playground Express na liście płytek

Aby w pełni wykorzystać wbudowane urządzenia peryferyjne płytki Circuit Playground Express, dodatkowo musisz zainstalować odpowiednie biblioteki Arduino. W tym celu otwórz menedżera bibliotek i wyszukaj *Playground Express*. Przetestuj jeden z przykładowych szkiców

dołączonych do biblioteki, który znajdziesz w menu *Plik/Przykłady*, choćby szkic *HelloNeoPixels* z kategorii *Hello Circuit Playground*. Przejrzyj pozostałe przykładowe szkice biblioteki *Circuit Playground*, aby zobaczyć, co możesz jeszcze zrobić z płytką, programując ją tak jak standardową płytkę Arduino.

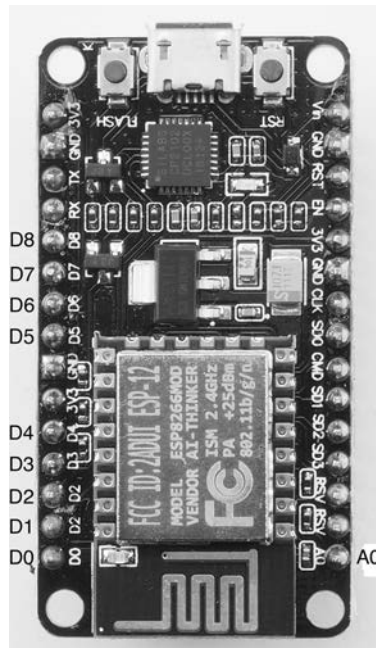
Uwaga: jeśli korzystasz z systemu Windows, to aby móc używać płytki, musisz zainstalować sterownik dla tego systemu (zobacz <https://learn.adafruit.com/introducing-circuit-playground/windows-driver-installation>). Użytkownicy systemów Linux i macOS nie muszą tego robić.

Wszystkie informacje na temat płytki Playground Express znajdziesz na stronie: <https://learn.adafruit.com/introducing-circuit-playground/>.

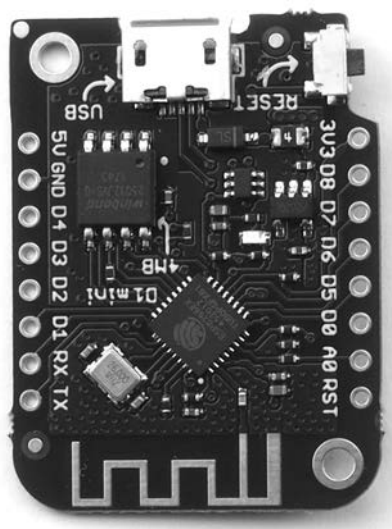
Firma Adafruit ma w swojej ofercie także duży wybór płytek kompatybilnych z Arduino. Odwiedź ich stronę i wyszukaj „feather”, „trinket” i „flora”, aby sprawdzić inne produkty tej firmy. Niektóre z nich mają specjalne funkcjonalności, na przykład wbudowaną antenę radiową, Wi-Fi czy ładowarkę baterii LiPo.

NodeMCU

Płytką NodeMCU i płytką z tej samej rodziny, Wemos D1 Mini (rysunki 3.7 i 3.8), to bardzo tanie płytki kompatybilne z Arduino z wbudowanym modulem Wi-Fi. Obie są oparte na module mikrokontrolera ESP8266 WiFi. Można go programować za pomocą wielu języków programowania, w tym Arduino.



Rysunek 3.7. Płytką NodeMCU



Rysunek 3.8. Płytką Wemos D1 Mini

Pomimo że na pierwszy rzut oka wydaje się, że płytką NodeMCU ma tyle samo pinów wejścia/wyjścia ogólnego przeznaczenia (GPIO) co Arduino Pro Mini, w rzeczywistości ma ich dużo mniej — tylko 1 wejście analogowe i 9 cyfrowych wejść/wyjść. Płytką Wemos D1 Mini korzysta z tego faktu i dzięki temu jest mniejsza.

Odwołując się do tych pinów, należy dodawać literę *A* lub *D*, na przykład A0, D4. To podejście różni się od standardowej płytki Arduino, gdzie litera *D* może być pominięta. Zauważ, że płytką NodeMCU, pokazana na rysunku 3.7, ma błąd w oznaczeniach pinów — D2 występuje dwukrotnie.

Jeśli używasz systemu Windows lub Linux, możliwe, że aby móc stosować te płytki, będziesz musiał zainstalować sterowniki dla CP2102 (dla NodeMCU) lub interfejsu USB CH340 (dla Wemos D1 Mini) (zobacz, odpowiednio, www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers lub www.arduinoed.eu/ch340-windows-8-driver-download).

Żeby dodać wsparcie dla tych płytek do Arduino IDE, otwórz menedżera płytek, wyszukaj ESP8266 i zainstaluj *esp8266 by ESP8266 Community*. Jeśli wgrasz szkic o nazwie *03_01_node_mcu_blink*, zobaczysz, jak miga wbudowana dioda LED (która w przypadku NodeMCU zazwyczaj podłączona jest do pinu D0, ale czasami do pinu D1 lub D2, a na Wemos D1 Mino — do pinu D4). Wszystkie przytoczone w tej książce szkice znajdziesz na stronie <http://www.helion.pl/ksiazki/arpok2.htm>.

Należy pamiętać, że wejścia i wyjścia tych płytek działają pod napięciem 3,3 V, dlatego nie podłączaj urządzeń wyjścia 5 V do wejść 3,3 V, bo zniszczysz płytkę. Każde cyfrowe wyjście może dostarczyć 12 mA. Jest to mniej niż w przypadku Arduino Uno, które może dostarczyć 40 mA, ale jest to wciąż wystarczająca ilość prądu, aby dioda LED jasno świeciła.

W rozdziale 14. znajdziesz przykłady wykorzystania wbudowanej funkcji Wi-Fi tych płytek oraz płytki ESP32 z kolejnej części.

ESP32

Często ograniczona liczba pinów GPIO płytki NodeMCU i innych płytek kompatybilnych z Arduino opartych na ESP8266 nie stanowi problemu. Jednak czasami możesz potrzebować ich więcej lub będziesz chciał użyć Bluetootha. W takich przypadkach wybierz płytkę z ESP32.

Firmy SparkFun i Adafruit produkują wysokiej jakości płytki oparte na ESP32, dla których stworzona jest rzetelna dokumentacja (na przykład SparkFun ESP32 Thing i Adafruit Huzzah32 Feather Board). Obie płytki mają wbudowaną możliwość ładowania baterii LiPo, co sprawia, że są doskonałym wyborem dla projektów mobilnych. Jeśli chcesz wydać mniej, w internecie możesz znaleźć tańsze alternatywy, między innymi pokazaną na rysunku 3.1 płytkę LOLIN32.

Aby móc dodać płytki oparte na ESP32 z poziomu menedżera płytek, musisz najpierw dodać adres URL: https://dl.espressif.com/dl/package_esp32_index.json w oknie preferencji, w polu *Dodatkowe adresy URL do menedżera płytek* (zobacz rysunek 3.3). Dzięki temu w menedżerze płytek znajdziesz płytkę *esp32 by Espresif Systems*.

Więcej informacji na temat projektu ESP32 jest dostępnych na stronie: <https://github.com/espressif/arduino-esp32>. Znajdziesz tam również obecny status projektu, ponieważ podczas pisania tej książki niektóre funkcjonalności nie zostały jeszcze wprowadzone, na przykład wyjście PWM (analogWrite).

ESP32 jest godnym uwagi urządzeniem z podwójnym procesorem, dużą ilością pamięci i naprawdę niskim zużyciem mocy, dzięki czemu doskonale sprawdzi się w projektach związanych z internetem rzeczy. Do tego zagadnienia wrócimy w rozdziale 14.

Mikrokontrolery ATtiny

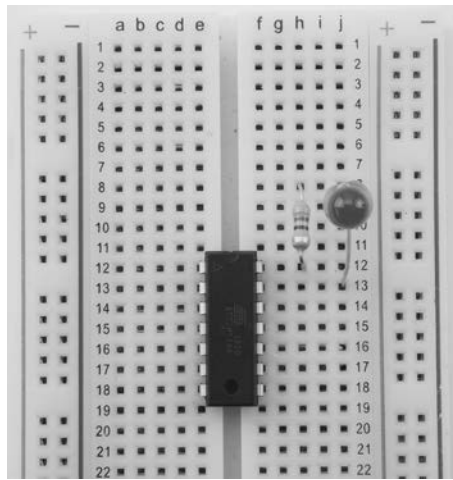
Niektóre projekty wymagają niewiele, jeśli chodzi o piny wejścia i wyjścia i zużywają zaledwie niewielki procent dostępnej 32-kilobajtowej pamięci flash mikrokontrolera ATmega328. W przypadku takich projektów możesz rozważyć zastosowanie płytki opartej na pokrewnym mikrokontrolerze z rodziny ATtiny. Te mikrokontrolery mają wiele wspólnych cech z ATmega, ale jak sama nazwa wskazuje (ang. *tiny* — małutki), ogólnie mają wszystkiego mniej, włączając w to piny i ilość pamięci każdego typu. Zaletami oczywiście są niska cena i mały rozmiar. ATtiny jest doskonałym kolejnym krokiem w przygodzie z Arduino Uno. Możesz zrezygnować z większości płytek Arduino i opierać swoje projekty tylko na jednym mikrokontrolerze.

ATtiny44

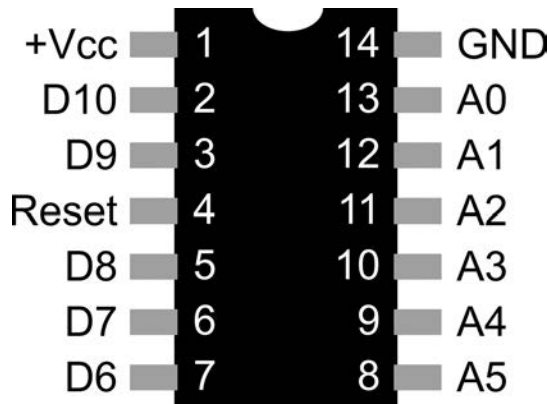
Dla przykładu przyjrzyjmy się, jak używa się mikrokontrolera ATtiny44. Ten chip plasuje się między pełnowymiarowymi mikrokontrolerami i naprawdę małymi 8-bitowymi kontrolerami ATtiny85.

ATtiny44 ma tylko 4 kB pamięci flash i 256 bajtów pamięci RAM. Jeśli zabraknie Ci miejsca na program, możesz zawsze użyć mikrokontrolera ATtiny84 z kompatybilnymi pinami i 8-kilobajtową pamięcią flash.

Na rysunku 3.9 widać mikrokontroler ATtiny44 z podłączoną diodą LED (sprawimy, że będzie migać) i opornikami umieszczonymi na płytce stykowej. Chip ma 14 pinów: 2 zasilania, 1 resetu, a pozostałe 11 to piny GPIO (zobacz rysunek 3.10).



Rysunek 3.9. Mikrokontroler ATtiny44 na płytce stykowej



Rysunek 3.10. Układ pinów ATtiny44

Arduino jako programator

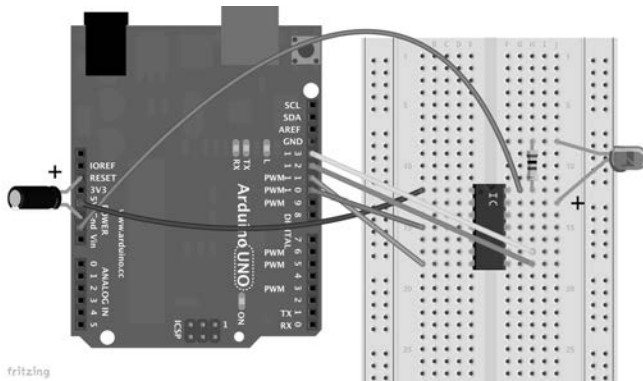
Może pamiętasz, że w rozdziale 2. Arduino Uno pełnił funkcję programatora w celu wgrania programu rozruchowego na inną płytkę Arduino. Arduino Uno możemy również użyć do zaprogramowania mikrokontrolera ATtiny, lecz zamiast wgrzywać program rozruchowy, na mikrokontrolerze umieścimy cały szkic (w tym przypadku program sprawiający, że dioda LED miga).

Zanim zrobisz cokolwiek innego, musisz wgrać szkic *ArduinoISP* na Arduino Uno. Znajdziesz go w menu *Plik*, w przykładach. Następnie możesz podłączyć Arduino Uno do ATtiny w opisany niżej sposób. Ważne jest, aby najpierw wgrać szkic *ArduinoISP*, ponieważ po podłączeniu kondensatora 10 µF do pinu Reset Arduino, nie będziesz mógł wgrać żadnego szkicu, dopóki kondensator nie zostanie usunięty.

Do wykonania pokazanych w tabeli 3.1 połączeń między Arduino Uno i ATtiny można użyć przewodów podłączeniowych zakończonych obustronnie końcówką męską. Połączenia pokazano także na rysunku 3.11.

Tabela 3.1. Połączenie Arduino Uno z ATtiny44 w celu programowania

Funkcja	Arduino Uno	ATtiny44
SCK (zegar)	13	9
MISO (wejście urządzenia nadrzędnego, wyjście urządzenia podrzędnego)	12	8
MOSI (wyjście urządzenia nadrzędnego, wejście urządzenia podrzędnego)	11	7
Reset	10	4



Rysunek 3.11. Połączenie Arduino Uno z ATtiny44 w celu programowania

Można łatwo określić, w którą stronę umieścić chip na płytce stykowej, gdyż ATtiny44 ma małą dziurkę na górnej krawędzi, obok pinu 1 (zobacz rysunek 3.10). Możliwe, że aby dopasować mikrokontroler do otworów płytki stykowej, będziesz musiał delikatnie wygiąć piny po obu stronach. Zazwyczaj sprawdza się lekkie przyciśnięcie obu stron do blatu stołu.

Kondensator ma jedną nóżkę dłuższą. Jest to przewód dodatni i należy podłączyć go do pinu Reset na Arduino. Podobnie jest w przypadku diody LED — dłuższa nóżka jest dodatnia.

Instalacja ATtinyCore w IDE

Dla ATtiny dostępnych jest wiele oprogramowań, które możesz dodać do swojego Arduino IDE. W niniejszej książce wykorzystano oprogramowanie o nazwie ATtinyCore, a jego dokumentację można znaleźć pod adresem: <https://github.com/SpenceKonde/ATtinyCore>.

Na tej stronie znajdziesz również pełną listę obsługiwanych przez ATtinyCore płytek wraz z bardzo przydatnymi schematami układu ich pinów.

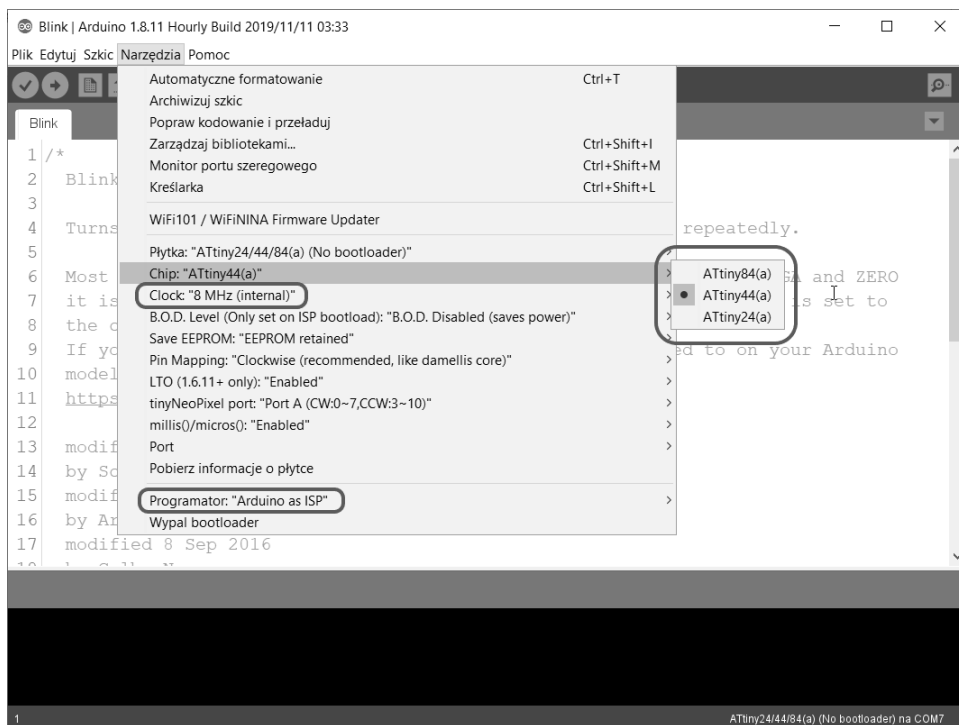
Oprogramowanie jest dostępne z poziomu menedżera płytek, jednak najpierw musisz dodać URL (http://drizzy.com/package_drizzy.com_index.json) w oknie preferencji (zobacz rysunek 3.4).

Po dodaniu adresu URL otwórz menedżera płytek, wyszukaj *attinycore* i zainstaluj najnowszą wersję.

Zegary, kryształy i bezpieczniki

Już wszystkie przewody są podłączone i Arduino Uno może pełnić funkcję programatora dla ATtiny. Jesteśmy prawie gotowi do wgrania szkicu na mikrokontroler, ale uprzednio musimy nieco skonfigurować Arduino IDE.

Zacznij od wyboru płytki ATtiny24/44/84 z sekcji ATtinyCore (zobacz rysunek 3.12).



Rysunek 3.12. Wybór płytki ATtiny24/44/84

Spowoduje to dodanie kilku opcji dotyczących ATtiny w menu *Narzędzia*. Dla większości z nich można zostawić ustawienia domyślne. Powinieneś zmienić opcję *Chip* na *ATtiny44* (zobacz rysunek 3.12). Upewnij się również, że opcja *Clock* (zegar) ma wartość *8 MHz (internal)*, a w menu *Narzędzia* ustaw *Programator* (druga pozycja od końca) na *Arduino as ISP*. Jeśli masz inny programator, to możesz go również wybrać w tym miejscu.

Niektóre opcje tego menu mają wpływ na „bezpieczniki”, które określają zachowanie mikrokontrolera, a nawet konfigurację danych pinów. Na przykład jeśli ustawisz opcję *Clock* na *8 MHz (internal)*, to informujesz Arduino IDE, że piny 2 i 3 ATtiny nie będą potrzebne rezonatorowi kwarcowemu, więc mogą być używane jako piny GPIO. Tutaj musimy być ostrożni, bo jeśli na przykład ustawimy opcję *Clock* na *16 MHz (external)*, to gdy będziemy chcieli zaprogramować ATtiny, okaże się, że nie jest to możliwe bez rezonatora kwarcowego. W rezultacie „zabetonowalibyśmy” nasz chip. Na szczęście wprowadzone w tych opcjach zmiany nie dają żadnego efektu, dopóki nie naciśniesz funkcji *Wypal bootloader*, która znajduje się na końcu menu *Narzędzia*. W rzeczywistości ta funkcja nie wgrzywa programu rozruchowego na ATtiny, a jedynie ustawia bezpieczniki.

W końcu możesz wgrać szkic na mikrokontroler. Możesz użyć szkicu o nazwie *03_02_attiny44_blink*, który znajduje się na stronie <http://www.helion.pl/ksiazki/arpok2.htm>. Aby wgrać szkic na ATtiny, wystarczy, że naciśniesz przycisk *Wgraj*, tak jak w przypadku standardowego Arduino. Po zakończonym wgrywaniu dioda LED powinna zacząć migać.

Minimalne Arduino

Jak pewnie zauważyłeś, większą część Arduino Uno zastąpiliśmy chipem umieszczonym na płytce stykowej. Zatem jak udało nam się pozbyć tak wielu elementów i co poświęciliśmy w zamian?

Przede wszystkim utraciliśmy źródło zasilania. Nie mamy już stabilizatora napięcia. W tym momencie ATtiny korzysta z zasilania Arduino Uno. Jednak ATtiny zadowolony jest każdym źródłem zasilania o napięciu z zakresu od 2,7 V do 5,5 V. Więc gdy zaprogramujesz już ATtiny, możesz zasilać go napięciem 3 V z dwóch baterii AA.

Z powodu tego, że użyliśmy Arduino Uno jako programatora, nie potrzebowaliśmy chipu interfejsu USB i powiązanych z nim komponentów. Nie mamy również oscylatora kwarcowego 16 MHz, małego srebrnego komponentu, który możesz znaleźć na Arduino Uno. ATtiny ma swój własny wbudowany oscylator, który działa o połowę wolniej. Jest to rekompensowane przez oprogramowanie i tak funkcja `delay` wciąż zatrzymuje program na zadany czas. Wbudowany oscylator nie jest tak dokładny jak zewnętrzny kryształ. Według dokumentacji dokładność oscylatora ATtiny44 wynosi ± 2 procent, natomiast margines błędu zewnętrznego oscylatora to $\pm 0,003$ procent. Jednak w rzeczywistości dokładność oscylatora ATtiny jest dużo lepsza od tej podanej w dokumentacji.

Podsumowanie

W tym rozdziale poznaliśmy kilka płytek, które stanowią alternatywę dla oficjalnych płytek Arduino, włączając w to możliwość stworzenia własnego Arduino za pomocą mikrokontrolera i płytki stykowej. Niektórych z tych urządzeń będziemy używać w kolejnych rozdziałach.

W następnym rozdziale przyjrzymy się przerwanom i zegarom, dzięki którym Arduino może reagować na określone zdarzenia zewnętrzne i na te zaplanowane.

PROGRAM PARTNERSKI

— GRUPY 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
Helion

Zostań mistrzem Arduino!

W pewnych kwestiach nic się nie zmienia — Arduino pozostaje praktycznym, tanim mikrokontrolerem, który może komunikować się ze światem zewnętrznym i obsługiwać przeróżne urządzenia elektroniczne. Daje też pole do popisu licznym amatorom i hobbystom. Środowisko programistyczne wciąż ma prosty interfejs, a programowanie płytki samo w sobie jest łatwym zadaniem nawet dla ludzi bez doświadczenia w pisaniu kodu. Zmiany w świecie Arduino dotyczą większych możliwości oraz sukcesywnie dodawanych funkcjonalności, takich jak obsługa bibliotek, czy kolejnych interfejsów.

To drugie, starannie zaktualizowane wydanie przewodnika po programowaniu Arduino, przeznaczonego dla osób, które opanowały podstawy i chcą zdobyć umiejętności na wyższym, profesjonalnym poziomie. W książce dodatkowo ujęto zagadnienia wykorzystania Arduino do komunikacji z urządzeniami szeregowymi oraz z internetem rzeczy (IoT). Dowiesz się również, jak używać środowiska programistycznego Arduino do programowania kompatybilnych płytek. Prezentowane treści zostały zilustrowane praktycznymi przykładami stosowania omówionych technik. Nie zabrakło licznych kodów do pobrania, które ułatwią Ci pracę z nawet bardzo ambitnymi projektami.

Dzięki książce dowiesz się, jak:

- konfigurować Arduino IDE i tworzyć efektywne szkice
- poprawić wydajność pracy przy zmniejszeniu natężenia prądu pobieranego przez Arduino
- pracować z różnymi interfejsami: I2C, 1-Wire, SPI, a także z układem TTL, USB i UART
- korzystać z Ethernetu, Bluetootha i DSP oraz z zasobów internetu
- tworzyć i udostępniać własne biblioteki

Dr Simon Monk jest inżynierem cybernetykiem, informatykiem i uzdolnionym hakerem. Po kilku latach pracy na uczelni został przedsiębiorcą. Obecnie dzieli swój czas pomiędzy pisanie rozchwytywanych książek i projektowanie produktów dla MonkMakes — firmy, którą prowadzi wraz z żoną Lindą. Mieszka w Preston w Wielkiej Brytanii.

  helion.pl	<i>Sprawdź nasze szkolenia!</i>  AKADEMIA IT & BUSINESS HELIONSZKOLENIA.PL	KOD KORZYŚCI <i>Sięgnij po więcej!</i>   ISBN 978-83-283-7548-2  9 788328 375482
INFORMATYKA W NAJLEPSZYM WYDANIU		Cena: 54,90 zł