Spraw, by rzeczy przemówiły

Programowanie urządzeń elektronicznych z wykorzystaniem Arduino

Tom Igoe

ZAPROJEKTUJ MIKROKONTROLERY, KOMPUTERY OSOBISTE, SERWERY I SMARTFONY TAK, BY SAME KOMUNIKOWAŁY SIĘ ZE SOBĄ!

O'REILLY®

Helion

Tytuł oryginalny: *Making Things Talk: Using Sensors, Networks, and Arduino to see, hear, and feel your world*

Tłumaczenie: Joanna Celej-Kobalczyk

ISBN: 978-83-246-5012-5

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/sprawb.zip

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

Printed in Poland.

Spis treści

6

Komunikacja bezprzewodowa

Jeśli, jak większość osób, jesteś zainteresowany tą dziedziną, to czytając wcześniejsze rozdziały, zastanawiałeś się zapewne: „A co z sieciami bezprzewodowymi?”. Być może jesteś tak gorliwy, że przeskoczyłeś prosto do tego rozdziału. Jeśli tak zrobiłeś, wróć i przeczytaj resztę książki! Szczególnie jeśli nie jesteś zaznajomiony z komunikacją szeregową pomiędzy komputerami a mikrokontrolerami, powinieneś przeczytać wcześniej rozdział 2. Niniejszy rozdział wyjaśnia podstawy komunikacji bezprzewodowej pomiędzy obiektami. Dowiesz się w nim o dwóch typach komunikacji bezprzewodowej, a następnie zbudujesz kilka działających przykładów.

◂◂ **Zygoty Alexa Beima** (*www.tangibleinteraction.com*) to lekkie, dmuchane gumowe kule rozświetlone przez światło LED. Kule zmieniają kolor w reakcji na zmianę ciśnienia na ich powierzchni i używają radia ZigBee do komunikacji z centralnym komputerem. Sieć zygot na koncercie sprawia, że odbiorcy mają bezpośredni wpływ nie tylko na same kule, ale także na muzykę i projekcje wideo, z którymi są one połączone w sieć.
Zdjęcie zamieszczamy dzięki uprzejmości Alexa Beima.

Zaopatrzenie do rozdziału 6.

OZNACZENIA DOSTAWCÓW:

- **A** — Arduino Store (*http://store.arduino.cc/ww*)
- **AF** — Adafruit (*http://adafruit.com*)
- **D** — Digi-Key (*www.digikey.com*)
- **F** — Farnell (*www.farnell.com*)
- **J** — Jameco (*http://jameco.com*)
- **MS** — Maker SHED (*www.makershed.com*)
- **RS** — RS (*www.rs-online.com*)
- **SF** — SparkFun (*www.sparkfun.com*)
- **SS** — Seeed Studio (*www.seeedstudio.com*)

PROJEKT 9. Sterownik na podczerwień do cyfrowego aparatu fotograficznego

» **1 moduł Arduino.** Arduino Uno lub coś, co bazuje na Arduino Uno, ale projekt powinien działać również na innych płytkach Arduino i zgodnych z Arduino. **D** 1050-1019-ND, **J** 2121105, **SF** DEV-09950, **A** A000046, **AF** 50, F 1848687, **RS** 715-4081, **SS** ARD132D2P, **MS** MKSP41

» **1 dioda LED na podczerwień.** **J** 106526, **SF** COM-09469, **F** 1716710, **RS** 577-538, **SS** MTR102A2B

» **1 przycisk.** Można zastosować dowolny przycisk. **D** GH1344-ND, **J** 315432, **SF** COM-10302, **F** 1634684, **RS** 718-2213

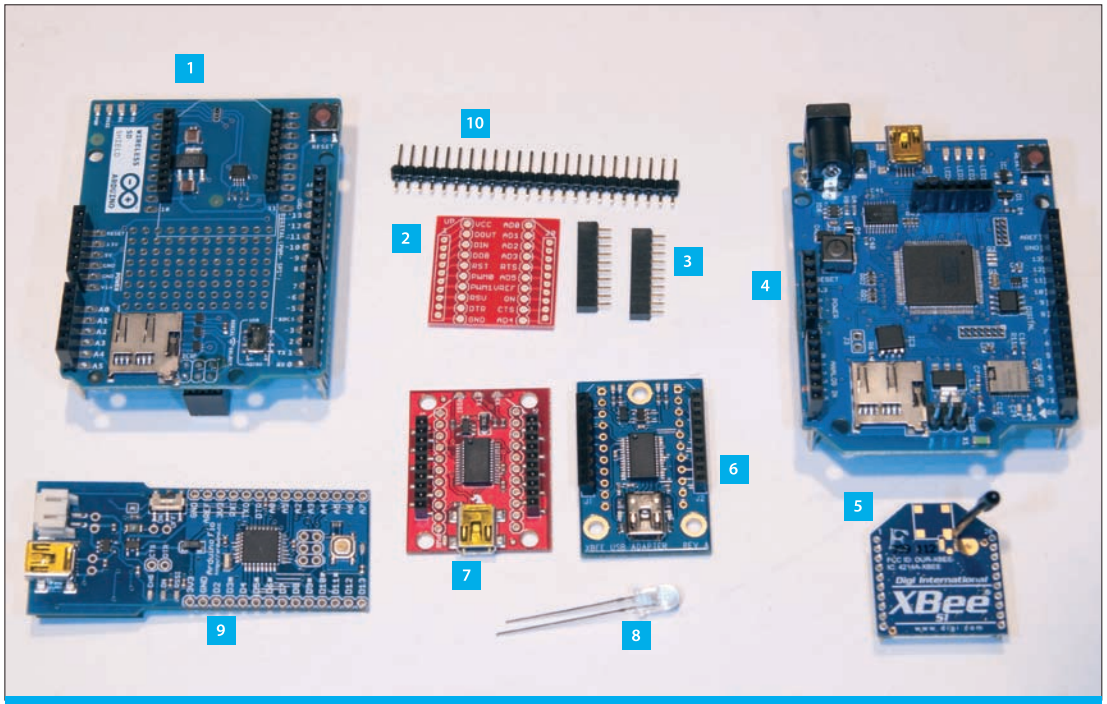
» **1 opornik 220 Ω.** **D** 220QBK-ND, **J** 690700, **F** 9337792, **RS** 707-8842

» **1 opornik 10 kΩ.** **D** 10KQBK-ND, **J** 29911, **F** 9337687, **RS** 707-8906

» **1 płytka stykowa.** **D** 438-1045-ND, **J** 20723 lub 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M lub STR102C2M, **MS** MKKN21

PROJEKT 10. Dupleksowa transmisja radiowa

» **2 płytki stykowe.** **D** 438-1045-ND, **J** 20723 lub 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M lub STR102C2M



Rysunek 6.1. Nowe części w tym rozdziale: **1.** Tarcza bezprzewodowa Arduino **2.** Płytka stykowa Spark Fun XBee **3.** Gniazda do listew kołkowych 2 mm **4.** Tarcza Arduino Wi-Fi **5.** Moduł Digi XBee 802.15.4 OEM **6.** Adapter Adafruit XBee na USB **7.** Spark Fun XBee Explorer **8.** Diody podczerwieni **9.** Arduno Fio. Nie zapomnij o mnóstwie listew kołkowych do płyt z wyprowadzeniami dla układów scalonych

- **2 moduły Arduino.** Modele Arduino Fios świetnie sprawdzają się we współpracy z XBee, ale projekt powinien działać również na innych płytkach zgodnych z Uno. **SF** DEV-10116
- **2 moduły RF Digi XBee 802.15.4.** **J** 2113375, **SF** WRL-08664, **AF** 128, **F** 1546394, **SS** WLS113A4M, **MS** MKAD14
- **2 tarcze bezprzewodowe Arduino.** Możesz nie używać tarcz i zamiast tego użyć części wymienionych poniżej. **A** A000064 lub A000065. Tarcze alternatywne: **SF** WRL-09976, **AF** 126, **F** 1848697, **RS** 696-1670, **SS** WLS114A0P
- **2 potencjometry.** **J** 29082, **SF** COM-09939, **F** 350072, **RS** 522-0625
- **1 adapter USB-XBee.** Poniższe części są niezbędne, jeśli nie używasz tarcz bezprzewodowych. **J** 32400, **SF** WRL-08687, **AF** 247
- **2 regulatory napięcia 3,3V.** **J** 242115, **D** 576-1134-ND, **SF** COM-00526, **F** 1703357, **RS** 534-3021
- **2 kondensatory 1μF.** **J** 94161, **D** P10312-ND, **F** 8126933, **RS** 475-9009
- **2 kondensatory 10μF.** **J** 29891, **D** P11212-ND, **F** 1144605, **RS** 715-1638
- **2 płytki z wyprowadzeniami dla XBee.** **J** 32403, **SF** BOB-08276, **AF** 127
- **4 listwy kołkowe z rozstawem 2,54 mm.** **J** 103377, **D** A26509-20ND, **SF** PRT-00116, **F** 1593411
- **4 gniazda do listew kołkowych 2 mm.** **J** 2037747, **D** 3M9406-ND, **F** 1776193
- **6 diod LED.** **D** 160-1144-ND lub 160-1665-ND, **J** 34761 lub 94511, **F** 1015878, **RS** 247-1662 lub 826-830, **SF** COM-09592 lub COM-09590

PROJEKT 11. Radiostacje Bluetooth

- **2 moduły Arduino.** Wybierz coś, co bazuje na Arduino Uno, ale projekt powinien działać również na innych płytkach Arduino i zgodnych z Arduino. **D** 1050-1019-ND, **J** 2121105, **SF** DEV-09950, **A** A000046, **AF** 50, **F** 1848687, **RS** 715-4081, **SS** ARD132D2P, **MS** MKSP4
- **2 płytki stykowe.** **D** 438-1045-ND, **J** 20723 lub 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M lub STR102C2M
- **2 diody LED. D** 160-1144-ND lub 160-1665-ND, **J** 34761 lub 94511, **F** 1015878, **RS** 247-1662 lub 826-830, **SF** COM-09592 lub COM-09590
- **2 potencjometry.** Można zastosować dowolny sensor analogowy. **J** 29082, **SF** COM-09939, **F** 350072, **RS** 522-0625
- **2 przyciski.** Można zastosować dowolne. **D** GH1344-ND, **J** 315432, **SF** COM-10302, **F** 1634684, **RS** 718-2213
- **2 oporniki 220 Ω.** **D** 220QBK-ND, **J** 690700, **F** 9337792, **RS** 707-8842
- **2 oporniki 10 kΩ.** **D** 10KQBK-ND, **J** 29911, **F** 9337687, **RS** 707-8906
- **1 adapter FTDI łącza szeregowego na USB.** Obie wersje — zarówno 5 V, jak i 3,3 V — będą działać; są produkowane jako przewody lub osobne moduły. **SF** DEV-09718 lub DEV-09716, **AF** 70, **A** A000059, **MS** MKAD22, **SS** PRO101D2P, **D** TTL-232R-3V3 lub TTL-232R-5V
- **2 moduły Bluetooth Mate.** **SF** WRL-09358 lub WRL-10393

PROJEKT 12. Witaj, Wi-Fi!

- **1 tarcza Arduino Wi-Fi. A** A000058
- **1 moduł Arduino.** Wybierz coś, co bazuje na Arduino Uno, ale projekt powinien działać również na innych płytkach Arduino i zgodnych z Arduino. **D** 1050-1019-ND, **J** 2121105, **SF** DEV-09950, **A** A000046, **AF** 50, F 1848687, **RS** 715-4081, **SS** ARD132D2P, **MS** MKSP41
- **1 połączenie Wi-Fi Ethernet z Internetem.**
- **3 oporniki 10 kΩ.** **D** 10KQBK-ND, **J** 29911, **F** 9337687, **RS** 707-8906
- **3 fotorezystory (oporniki światłoczułe).** **D** PDVP9200-ND, **J** 202403, **SF** SEN-09088, **F** 7482280, **RS** 234-1050
- **1 płytka stykowa.** **D** 438-1045-ND, **J** 20723 lub 20601, **SF** PRT-00137, **F** 4692810, **AF** 64, **SS** STR101C2M lub STR102C2M1, **MS** MKKN2
- **3 filtry oświetlenia.** Jeden podstawowy czerwony, jeden podstawowy zielony i jeden podstawowy niebieski. Dostępne u lokalnych dostawców sprzętu oświetleniowego lub fotograficznego.

Początkowa część niniejszego rozdziału opisuje sposób działania sieci bezprzewodowej, dając Ci pewne podstawy i punkty odniesienia do rozwiązywania problemów. Druga połowa rozdziału zawiera przykłady. Temat jest tak szeroki, że nawet pobieżny przegląd kilku różnych urządzeń dotyka jedynie wierzchołka góry lodowej.

Z tego powodu ćwiczenia w niniejszym rozdziale nie będą w pełni dopracowanymi aplikacjami tak jak w poprzednich. Zamiast tego po prostu otrzymasz przykład z podstawowym „Witaj, świecie!" dla kilku rodzajów urządzeń bezprzewodowych.

X

Dlaczego nie wszystko jest bezprzewodowe?

Zaleta komunikacji bezprzewodowej wydaje się oczywista: brak kabli! Powoduje to, że konstrukcja fizyczna jest znacznie prostsza dla każdego projektu, gdzie urządzenia muszą się przemieszczać i komunikować ze sobą. Ubrania z systemami czujników, cyfrowe instrumenty muzyczne i zdalnie sterowane pojazdy mają uproszczoną konstrukcję fizyczną dzięki komunikacji bezprzewodowej. Jednak istnieją pewne ograniczenia komunikacji bezprzewodowej, które trzeba rozważyć przed jej zastosowaniem.

Komunikacja bezprzewodowa nigdy nie jest tak niezawodna jak komunikacja przewodowa

Masz mniejszą kontrolę nad źródłami zakłóceń. Można ochronić i osłonić przewód odpowiadający za komunikację danych, ale nigdy nie da się całkowicie zaizolować bezprzewodowych łączy radiowych lub podczerwonych. Zawsze będzie jakaś forma interferencji, więc musisz mieć pewność, że wszystkie urządzenia w systemie wiedzą, co zrobić, kiedy otrzymają zniekształcony komunikat (lub w ogóle go nie otrzymają) od swoich odpowiedników.

Komunikacja bezprzewodowa nigdy nie jest komunikacją tylko jeden do jednego

Urządzenia radiowe i na podczerwień emitują sygnały w taki sposób, że wszyscy mogą je usłyszeć. Czasami oznacza to kolizję z komunikacją między innymi urządzeniami. Na przykład: Bluetooth, większość radioodbiorników Wi-Fi (802.11b, g i n) oraz radioodbiorniki ZigBee (802.15.4) pracują na tym samym zakresie częstotliwości: 2,4 GHz (802.11n będzie również działać na 5 GHz). Są zaprojektowane tak, aby nie powodować wzajemnie nadmiernych zakłóceń, ale jeśli masz dużą liczbę radioodbiorników ZigBee pracujących w tej samej przestrzeni co bardzo aktywna sieć Wi-Fi, pojawią się zakłócenia.

Komunikacja bezprzewodowa nie oznacza zasilania bezprzewodowego

Nadal musisz dostarczyć energię do urządzeń i jeśli przemieszczają się, to oznacza, że będą potrzebne baterie. Baterie zwiększają ciężar urządzenia i nie są wieczne. Awaria baterii podczas testowania projektu może spowodować wszelkiego rodzaju błędy, które mógłbyś przypisać do innych przyczyn. Klasycznym przykładem jest „tajemniczy błąd radiowy". Wiele odbiorników radiowych zużywa dodatkową energię podczas transmisji. Powoduje to niewielki spadek napięcia źródła zasilania. Jeśli radio nie ma kondensatora oddzielającego zasilanie i masę, napięcie może spaść do tak niskiego poziomu, że zresetuje radio. Może to wyglądać tak, że radio będzie działać normalnie, kiedy wysyłasz do niego wiadomości szeregowo, ale nigdy nie będzie transmitować i nie będzie wiadomo, dlaczego tak się dzieje. Podczas tworzenia projektów sieci bezprzewodowej warto najpierw upewnić się, że komunikacja działa prawidłowo z wykorzystaniem podłączonego zasilacza stabilizowanego, a następnie stworzyć stabilne zasilanie bateryjne.

Komunikacja bezprzewodowa generuje promieniowanie elektromagnetyczne

Można łatwo o tym zapomnieć, ale radio, którego używasz, emituje energię elektromagnetyczną. Taka sama energia, jaka gotuje Twoje jedzenie w kuchence mikrofalowej, wysyła Twoje pliki MP3 w Internecie. I chociaż istnieje wiele badań wykazujących, że jest to bezpieczne na niskim poziomie operacyjnym używanych tutaj odbiorników radiowych, po co zwiększać szum w eterze, jeśli nie ma takiej potrzeby?

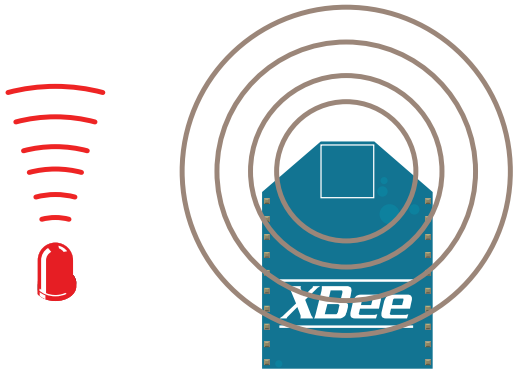
Najpierw zrób wersję przewodową

Omówione tutaj radio i urządzenia nadawczo-odbiorcze na podczerwień zastępują komunikację przewodową używaną w poprzednich rozdziałach. Zanim podejmiesz decyzję o dodaniu sieci bezprzewodowej do dowolnej aplikacji, wpierw upewnij się, że możliwa jest podstawowa wymiana wiadomości pomiędzy urządzeniami komunikującymi się przewodowo.

Podstawowe media sieci bezprzewodowej: podczerwień i radio

W życiu codziennym większość ludzi styka się z dwoma głównymi typami komunikacji bezprzewodowej: komunikacją z użyciem światła podczerwonego i komunikacją radiową. Główną różnicą pomiędzy nimi, z punktu widzenia użytkownika lub programisty, jest ich **kierunkowość** (ang. *directionality*).

Piloty do zdalnego sterowania telewizorem zazwyczaj korzystają z komunikacji w podczerwieni (IR). W przeciwieństwie do radia, jest ona zależna od wzajemnego położenia nadajnika i odbiornika. Musi istnieć nieprzesłonięta niczym przestrzeń pomiędzy obydwoma. Czasami IR może działać przez odbicie wiązki od innej powierzchni, ale nie jest to niezawodne. Ostatecznie odbiornik jest urządzeniem optycznym, więc musi „widzieć" sygnał. Piloty do otwierania drzwi samochodu, telefony komórkowe, zdalne sterowanie drzwi garażowych i wiele innych urządzeń korzysta z radia. Działają niezależnie od tego, czy nadajnik i odbiornik są skierowane ku sobie. W niektórych przypadkach mogą nawet działać przez ściany. Innymi słowy ich transmisja jest **wielokierunkowa** (ang. *omnidirectional*). Ogólnie IR jest stosowane do aplikacji bliskiego zasięgu w linii wzroku, a radio jest używane do wszystkich innych zadań (rysunek 6.2 ilustruje tę różnicę).



Rysunek 6.2.
Sygnał z diody LED (po lewej) emanuje na zewnątrz w wiązce promieniowania LED, podczas gdy sygnał anteny radiowej, tak jak w radiu XBee (po prawej), promieniuje w wielu kierunkach

Nadajniki, odbiorniki i urządzenia nadawczo-odbiorcze

Istnieją trzy typy urządzeń wspólnych dla systemów IR i RF: **nadajniki** (ang. *transmitters*), które wysyłają sygnał, lecz nie mogą go odebrać, **odbiorniki** (ang. *receivers*), które odbierają sygnał, lecz nie mogą wysyłać, i **urządzenia nadawczo-odbiorcze** (ang. *transceivers*), które mogą robić obie te rzeczy. Być może zastanawiasz się, dlaczego nie wszystko jest urządzeniem nadawczo-odbiorczym, skoro jest to najbardziej elastyczne urządzenie. Zbudowanie urządzenia nadawczo-odbiorczego jest zadaniem bardziej złożonym niż zbudowanie jednego z dwóch pozostałych. W przypadku urządzenia nadawczo-odbiorczego musisz upewnić się, że odbiornik nie odbiera transmisji swojego nadajnika i że nie będą się one wzajemnie zakłócać i nie będą słuchać innych urządzeń. Dla wielu zastosowań taniej jest użyć pary nadajnik-odbiornik i obsługiwać błędy po prostu przez wielokrotne przekazywanie wiadomości, aż odbiornik ją odbierze. Tak działa na przykład zdalne sterowanie telewizorem. To sprawia, że komponenty są dużo tańsze.

Coraz częściej aplikacje radiowe są budowane po prostu na urządzeniach nadawczo-odbiorczych i dołącza się mikrokontroler do zarządzania filtrowaniem nadajnika i odbiornika. Wszystkie radia Bluetooth, ZigBee i Wi-Fi pracują w ten sposób. Jednak nadal możliwe jest zakupienie pary nadajnik-odbiornik radiowy i są one tańsze od odpowiadającego im urządzenia nadawczo-odbiorczego.

Pamiętaj o rozróżnieniu pomiędzy parą nadajnik-odbiornik a urządzeniem nadawczo-odbiorczym podczas planowania swoich projektów i podczas robienia zakupów. Rozważ, czy komunikacja w Twoim projekcie musi być dwukierunkowa, czy może tylko jednokierunkowa. Jeśli jest jednokierunkowa, zadaj sobie pytanie, co się stanie, jeśli komunikacja nie powiedzie się. Czy odbiornik może działać bez pytania o wyjaśnienie? Czy można rozwiązać problem poprzez wielokrotne nadawanie do momentu otrzymania wiadomości? Jeśli odpowiedź na to pytanie brzmi „tak", to możesz spróbować użyć pary nadajnik-odbiornik i zaoszczędzić trochę pieniędzy.

Jak działa podczerwień?

Komunikacja IR działa przez wysyłanie w wyznaczonych odstępach czasowych impulsów diody IR LED i odbieranie impulsów przy użyciu fotodiody IR. Jest to po prostu komunikacja szeregowa przekazywana z wykorzystaniem podczerwieni. Ponieważ istnieje wiele powszechnych źródeł światła IR (słońce, żarówki żarowe, inne źródła ciepła), konieczne jest odróżnienie sygnału danych IR od innych źródeł energii IR. Aby to zrobić, szeregowe wyjście jest wysyłane do oscylatora przed wysłaniem go do wyjścia LED. Fala utworzona przez oscylator, nazywana **falą nośną** (ang. *carrier wave*), to regularne impulsy modulowane przez impulsy sygnału danych. Odbiornik odbiera całość światła IR, ale odfiltrowuje wszystko, co nie wibruje na częstotliwości nośnej. Następnie odfiltrowuje się częstotliwość nośną, więc wszystko, co pozostaje, jest sygnałem danych. Metoda ta pozwala na transmitowanie danych przy użyciu podczerwieni bez ryzyka zakłóceń z innych źródeł światła IR — chyba że zdarzy się, że oscylują z taką samą częstotliwością jak Twoja fala nośna.

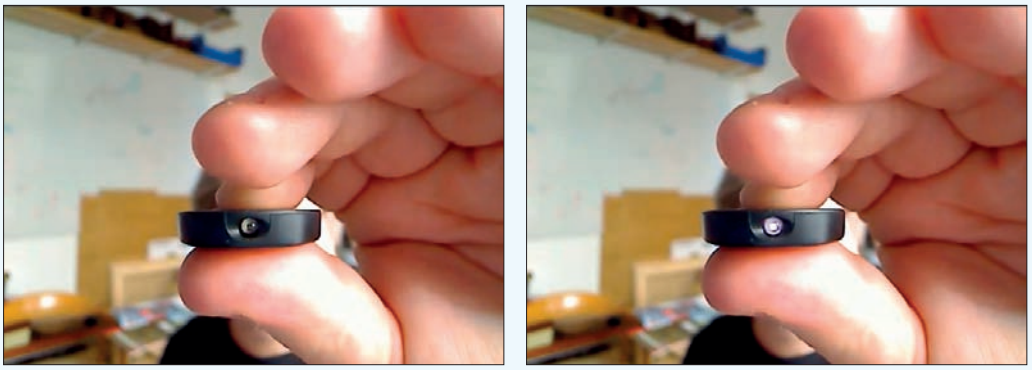
Kierunkowa natura podczerwieni czyni ją bardziej ograniczoną, ale jest tańsza niż radio i wymaga mniej energii. Ponieważ transfer drogą radiową staje się coraz tańszy, a jego zasilanie jest bardziej niezawodne, coraz trudniej znaleźć komputer z portem IR. Jednak nadal jest to rozwiązanie zarówno opłacalne, jak i efektywne w zużyciu energii dla aplikacji zdalnego sterowania, w których nadawca i odbiorca znajdują się w linii wzroku.

Protokoły danych dla pilotów zdalnego sterowania IR w większości domowej elektroniki różnią się w zależności od producenta. Aby je zdekodować, musisz znać częstotliwość nośną i strukturę wiadomości. Większość komercyjnych urządzeń zdalnego sterowania IR działa z użyciem fali nośnej pomiędzy 38 a 40 kHz. Częstotliwość fali nośnej ogranicza prędkość, z jaką można wysyłać dane na tej fali, więc transmisja IR zwykle odbywa się z niską prędkością danych, zazwyczaj między 500 a 2000 bitów na sekundę. Nie jest to doskonałe rozwiązanie do przesyłania dużych ilości danych, ale jeśli wysyłasz tylko wartości kilku przycisków na pilocie, to jest to akceptowalne. W odróżnieniu od protokołów szeregowych, które widziałeś dotychczas w tej książce, nie wszystkie protokoły IR używają 8-bitowego formatu danych. Na przykład protokół Sony Control-S ma trzy formaty: 12 bitów, 15 bitów i 20 bitów. Format Philips RC5, popularny w wielu pilotach, używa formatu 14-bitowego.

Jak zobaczyć podczerwień?

Istnieją dwa narzędzia, które są naprawdę przydatne podczas pracy z nadajnikami i odbiornikami IR: aparat fotograficzny i oscyloskop.

Nawet jeśli Ty nie możesz zobaczyć światła podczerwonego, aparat fotograficzny je widzi. Jeśli nie masz pewności, czy dioda IR LED działa, szybkim sposobem sprawdzenia jest zrobienie zdjęcia diody LED. Jeśli działa, na zdjęciu zobaczysz zaświeconą diodę. Rysunek 6.3 pokazuje diodę IR LED w domowym pilocie zdalnego sterowania, oglądaną przez kamerę internetową podłączoną do komputera. Ten efekt można nawet zobaczyć na wizjerze LCD aparatu cyfrowego. Jeśli próbujesz to zrobić ze swoją diodą IR LED, może być konieczne wyłączenie świateł lub zasunięcie zasłon, aby można było zobaczyć efekt. Niektóre kamery internetowe mają wbudowany filtr IR, więc najpierw należy sprawdzić urządzenie IR, o którym wiesz, że działa, takie jak pilot do telewizora, zanim użyjesz kamery do sprawdzenia, czy działa Twój projekt.



Rysunek 6.3. Aparat fotograficzny jest przydatny podczas rozwiązywania problemów z projektami IR

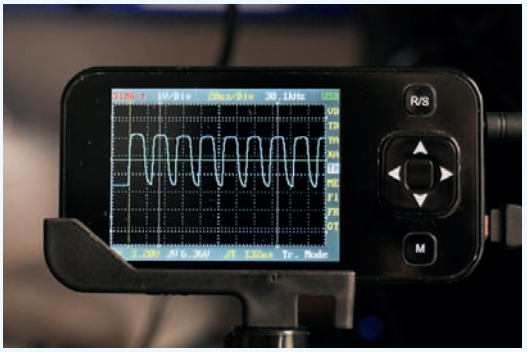
Jeśli musisz wysyłać lub odbierać sygnały pilotów zdalnego sterowania, to zaoszczędzisz mnóstwo czasu, wybierając układ scalony modulatora IR, zamiast próbować odtworzyć protokół samodzielnie. Na szczęście istnieje wiele dobrych stron WWW, które wyjaśniają różne protokoły. Reynolds Electronics (*www.rentron.com*) oferuje wiele pomocnych samouczków i sprzedaje mnóstwo przydatnych modulatorów i demodulatorów IR. EPanorama ma szereg przydatnych linków opisujących wiele bardziej powszechnych protokołów IR na *www.epanorama.net/links/irremote.html*. Istnieje również wiele bibliotek napisanych dla Arduino ułatwiających wysyłanie i odbieranie sygnałów IR dla różnych protokołów. Wiele z nich jest wymienionych na stronie Arduino pod adresem *http://arduino.cc/playground/Main/InterfacingWithHardware*. W kolejnym projekcie zobaczysz działanie jednego z nich.

Jeśli konstruujesz zarówno nadajnik, jak i odbiornik, zadanie jest dość proste. Wystarczy oscylator, za pomocą którego można przekazać dane szeregowo do diody podczerwonej LED, i odbiornik, który nasłuchuje fali nośnej i demoduluje sygnał danych. Można zbudować własny modulator IR przy użyciu układu scalonego czasomierza 555, ale równie dobrze można kupić szereg niedrogich modułów do modulacji lub demodulacji sygnału IR.
X

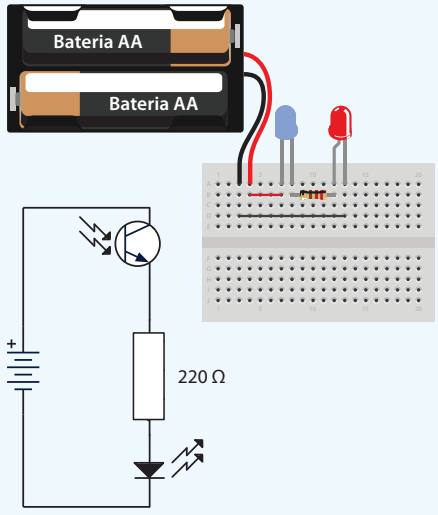
Podsłuchiwanie sygnałów w podczerwieni

Gdy próbujesz odszyfrować sygnał IR, przydatny jest również oscyloskop (patrz rysunek 6.4). Być może nie znasz protokołu swojego odbiornika, ale możesz go rozpracować, obserwując przekazywany sygnał. Jeśli połączysz fototranzystor podczerwieni, opornik i zwykłą diodę LED szeregowo, tak jak pokazano na rysunku 6.5, to kiedy skierujesz pilot na fototranzystor, dioda LED powinna się zaświecić.

Aby wyświetlić sygnał pilota na oscyloskopie, podłącz jego sondy do masy i emitera fototranzystora i włącz pilot skierowany na fototranzystor. Kiedy zaobserwujesz działanie, dopasuj podziałkę napięcia i czasu na oscyloskopie, aż zobaczysz czytelny odczyt aktywności. Większość oscyloskopów podpowie Ci częstotliwość sygnału automatycznie. Ustawienie oscyloskopu w tryb wyzwalacza jednokrotnego (ang. *single-shot trigger mode*) **pomoże Ci przechwytywać rzeczywisty sygnał. Teraz, kiedy możesz zobaczyć impuls każdego sygnału w czasie, możesz odtworzyć go, generując własne impulsy na diodzie IR LED. Aby uzyskać więcej informacji na ten temat, zapoznaj się z którymś z doskonałych blogów, w których znajdziesz wiele zapisów na temat zdalnego sterowania IR przy użyciu Arduino. Na przykład przeczytaj bardzo dobre objaśnienie na blogu Kena Shirriffa pod adresem** *www.arcfn.com*.



Rysunek 6.4. Oscyloskop może Ci pomóc zaobserwować strukturę sygnału IR



Rysunek 6.5. Fototranzystor IR i dioda LED połączone szeregowo dobrze się sprawdzają przy testowaniu odbioru IR

MATERIAŁY:

- **1 płytka stykowa,**
- **1 opornik 220 Ω,**
- **1 fototranzystor Digi-Key nr części: 365-1068-ND,**
- **1 dioda LED,**
- **1 bateria lub źródło zasilania 5 V lub mniej,**
- **1 oscyloskop DSO Nano (pokazany tutaj).**

Projekt 9.

Sterownik na podczerwień do cyfrowego aparatu fotograficznego

W tym przykładzie użyjemy diody LED na podczerwień i Arduino do sterowania cyfrowym aparatem fotograficznym. Jest to chyba najprostszy projekt sterowania IR, jaki można zrobić.

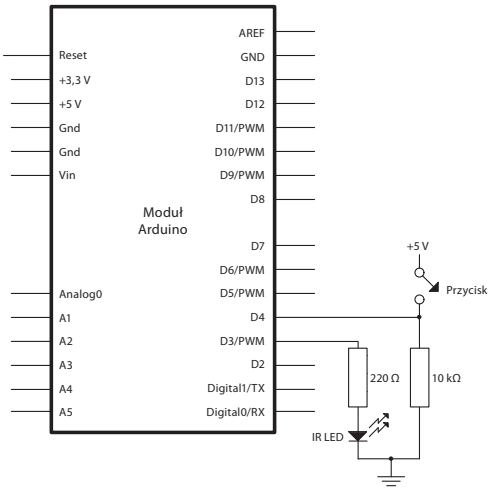
MATERIAŁY:

- **1 moduł Arduino,**
- **1 dioda LED na podczerwień,**
- **1 przycisk,**
- **1 opornik 220 Ω,**
- **1 opornik 10 kΩ,**
- **1 płytka stykowa lub tarcza prototypowa.**

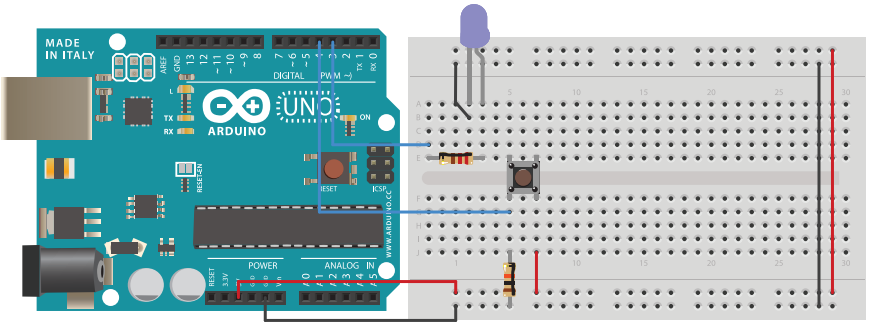
Większość dostępnych na rynku cyfrowych aparatów SLR ma możliwość zdalnego sterowania przez podczerwień. Każda marka używa nieco innego protokołu, ale charakteryzują je te same podstawowe polecenia: wyzwalacz migawki, wyzwalacz z opóźnieniem i ustawienie ostrości. Sebastian Setz jest autorem biblioteki Arduino, która pozwala wysyłać sygnały do większości popularnych aparatów. Została przetestowana z aparatami Canon, Nikon, Olympus, Pentax i Sony. Jeśli masz aparat SLR którejkolwiek z tych marek, możesz nim sterować za pomocą tej biblioteki.

Obwód dla tego projektu jest prosty. Połącz przycisk z pinem 4 mikrokontrolera (z opornikiem stanu niskiego 10 kΩ) i połącz diodę LED na podczerwień z pinem 3 mikrokontrolera, tak jak pokazano na rysunku 6.6.

Pobierz bibliotekę Multi Camera IR Control z *http://sebastian.setz.name/arduino/my-libraries/multi-camera-ir-control* i skopiuj ją do katalogu `libraries` (biblioteki) w katalogu szkiców Arduino. Jeśli nigdy wcześniej nie instalowałeś biblioteki, musisz utworzyć ten katalog. Kiedy już tam jest, ponownie uruchom aplikację Arduino, a nowa biblioteka powinna być widoczna w menu *Sketch/Import Library* (szkic/import biblioteki) pod nazwą *MultiCameraIrControl*.



Rysunek 6.6.
Mikrokontroler z dołączoną diodą IR LED i przyciskiem



Wypróbuj Aby uruchomić szkic, zaimportuj bibliotekę *MultiCameraIrControl*. Zainicjuj bibliotekę do przesyłania sygnałów na pinie 3, do którego podłączona jest dioda LED. Następnie skonfiguruj kilka zmiennych do śledzenia stanu przycisku.

```
/*
  Sterowanie aparatem fotograficznym przez IR
  Kontekst: Arduino

  Ten szkic steruje aparatem cyfrowym za pomocą podczerwonej diody LED.
*/

// dołącz bibliotekę do sterowania kamerą:
#include <multiCameraIrControl.h>

const int pushButtonPin = 4;
// ustaw pin 3 do sterowania diodą IR LED.
// Zmień to ustawienie w zależności od marki aparatu fotograficznego:
Nikon camera(3);

// zmienne, które reprezentują:
int buttonState = 0;         // bieżący stan przycisku
int lastButtonState = 0;     // poprzedni stan przycisku
```

Metoda `setup` inicjuje pin przycisku jako wejście.

```
void setup(){
  // zainicjuj przycisk jako wejście:
  pinMode(pushButtonPin, INPUT);
}
```

Główna pętla nasłuchuje, czy stan przycisku się zmienił. Ponieważ nie chcemy, żeby aparat robił zdjęcia przez cały czas, wyzwolimy go tylko wtedy, gdy przycisk zmienia stan z OFF na ON. Aby to zrobić, główna pętla porównuje stan przycisku ze stanem poprzednim, zachowując bieżący stan jako ostatni na końcu każdej pętli.

Oto cały program. Teraz skieruj diodę LED na aparat fotograficzny i zacznij robić zdjęcia zdalnie. Może zajść potrzeba przełączenia aparatu w tryb zdalnego sterowania. Sprawdź w instrukcji aparatu fotograficznego, jak to zrobić, ponieważ zależy to od aparatu.

```
void loop(){
  // odczytaj pin wejściowy przycisku:
  buttonState = digitalRead(pushButtonPin);

  // porównaj buttonState ze stanem poprzednim.
  // Jeśli został zmieniony i teraz jest wysoki, to
  // właśnie naciśnięto przycisk:
  if (buttonState != lastButtonState && buttonState == HIGH) {
    // wyślij sygnał otwarcia migawki:
    camera.shutterNow();
  }
  // zapisz bieżący stan jako ostatni stan,
  // dla następnej iteracji pętli
  lastButtonState = buttonState;
}
```

A. czujnik PIR

Rysunek 6.7.
Ten interwalometr został zbudowany przy użyciu tych samych metod co powyżej. Arduino w pudełku wykrywa zmiany z czujnika PIR i wysyła sygnał IR do aparatu fotograficznego, aby zrobić zdjęcie

Jak działa radio?

Radio bazuje na właściwości elektrycznej o nazwie **indukcja** (ang. *induction*). Za każdym razem, kiedy zmienia się natężenie prądu elektrycznego w przewodzie, generowane jest odpowiadające mu pole magnetyczne, rozchodzące się od przewodu. To zmienne pole magnetyczne indukuje prąd elektryczny w innych przewodach będących w zasięgu pola. Częstotliwość pola magnetycznego jest równa częstotliwości prądu w oryginalnym przewodzie. Oznacza to, że jeśli chcesz wysłać sygnał bez przewodu, można wygenerować prąd zmienny z daną częstotliwością w jednym przewodzie i dołączyć obwód do wykrywania prądu zmiennego na tej częstotliwości do drugiego przewodu. Tak właśnie działa radio.

Odległość przesyłania sygnału radiowego zależy od siły sygnału, czułości odbiornika, rodzaju anteny i wszelkich przeszkód, które blokują sygnał. Im silniejszy oryginalny prąd i bardziej wrażliwy odbiornik, tym dalej od siebie mogą być nadawca i odbiorca. Dwa przewody działają jako anteny. Dowolny przewodnik może być anteną, ale niektóre działają lepiej od innych. Długość i kształt anteny oraz częstotliwość sygnału wpływają na transmisję. Projektowanie anteny jest samo w sobie odrębną gałęzią nauki, więc nie uda mi się tutaj zbyt wiele przekazać, ale zdroworozsądkowa zasada przy konstruowaniu anteny z prostego drutu jest następująca:

Długość anteny = 5,616 cala / częstotliwość w MHz = 14,266.06 cm / częstotliwość w MHz

Aby uzyskać więcej informacji, skonsultuj się ze specyfikacją techniczną konkretnego radia, którego używasz. Instrukcje dotyczące wykonania dobrej anteny są zazwyczaj podawane w dokumentacji radia.

Transmisja radiowa: cyfrowa i analogowa

Tak jak ze wszystkim innym w świecie mikrokontrolerów, ważne jest rozróżnienie pomiędzy cyfrową a analogową transmisją radiową. W analogowych radiach analogowy

sygnał elektryczny, taki jak sygnał audio, jest nakładany na częstotliwości radiowe — i w ten sposób transmitowany. Częstotliwość radiowa działa jako fala nośna przenosząca sygnał audio. Cyfrowe radia nakładają sygnały cyfrowe na falę nośną, więc musi istnieć urządzenie cyfrowe na obu końcach, aby zakodować lub odkodować te sygnały. Innymi słowy, radia cyfrowe są zasadniczo modemami, konwertującymi dane cyfrowe na sygnały radiowe i sygnały radiowe na dane cyfrowe.

Zakłócenia radiowe

Chociaż anteny, których będziesz używać w niniejszym rozdziale, są wielokierunkowe, sygnał radiowy może być blokowany przez przeszkody, szczególnie metalowe. Na przykład duży arkusz metalowej blachy raczej odbije sygnał radiowy, niż pozwoli na przejście przez niego. Zasada ta jest wykorzystywana nie tylko w projektowaniu anteny, ale również przy projektowaniu **osłon radiowych** (ang. *radio frequency (RF) shield*). Jeśli kiedykolwiek rozcinałeś przewód komputerowy i znalazłeś wewnątrz cienką folię owiniętą wokół przewodów, to napotkałeś osłonę RF. Osłony są używane, aby zapobiec zakłócaniu danych przesyłanych w przewodzie przez losowe sygnały radiowe. Osłona nie musi być pełnym arkuszem metalu — siatka z przewodzącego metalu także będzie blokować sygnał radiowy, jeśli oczka siatki są wystarczająco małe. Skuteczność projektowanej siatki zależy od częstotliwości, jaką ma zablokować. Istnieje możliwość zablokowania sygnałów radiowych na danym obszarze poprzez otoczenie obszaru właściwą osłoną i uziemienie jej. Mówi się o tym powszechnie jako o tworzeniu **klatki Faradaya**. Efekt ten został nazwany na cześć Michaela Faradaya, który jako pierwszy go wykazał i udokumentował.

Czasami transmisja radiowa jest blokowana przez niezamierzone osłony. Jeśli masz kłopot z przesłaniem sygnału radiowego, poszukaj metalu, który może być przeszkodą dla niego. Transmisja z wnętrza samochodu może czasami być kłopotliwa, ponieważ nadwozie auta działa jak klatka Faradaya. Wyprowadzenie anteny na zewnątrz kabiny poprawia odbiór. Woda może równie skutecznie blokować RF. Jest to prawdą dla prawie każdej obudowy radiowej.

Wszystkie rodzaje urządzeń elektrycznych emitują fale radiowe jako efekt uboczny ich eksploatacji. Każdy prąd zmienny może wygenerować sygnał radiowy, nawet prąd, który napędza sprzęty w Twoim domu lub biurze. Dlatego właśnie słychać buczenie, gdy umieścisz przewody głośnika równolegle z kablem zasilającym. Sygnał prądu zmiennego zakłóca prąd w przewodach głośnika, a głośniki odczytują zmiany prądu jako dźwięk. Z podobnych powodów możesz mieć problemy, operując bezprzewodową siecią danych w pobliżu kuchenki mikrofalowej. Wi-Fi działa na częstotliwościach w zakresie gigaherca, powszechnie nazywanych **zakresem mikrofalowym**, ponieważ długość fali tych sygnałów jest bardzo krótka w porównaniu z sygnałami na niższych częstotliwościach. Aby ugotować jedzenie, kuchenka mikrofalowa generuje energię w tym zakresie do pobudzenia (nagrzewania) cząsteczek wody w żywności. Część tej energii wycieka z kuchenki przy małej mocy i właśnie dlatego otrzymasz różnego rodzaju szum radiowy w zakresie gigaherca w pobliżu kuchenki mikrofalowej.

Generatory i silniki są szczególnie podstępnymi źródłami szumów radiowych. Silnik działa również poprzez indukcję; w szczególności para magnesów zamocowanych do wału wiruje wewnątrz cewki z przewodów. Podłączając przewód do prądu, tworzysz pole magnetyczne, które przyciąga lub odpycha magnesy, powodując ich obracanie. Podobnie, używając siły mechanicznej do obracania magnesami, generujesz prąd w przewodzie. Tak więc silnik (lub generator) jest zasadniczo małym radiem, które generuje szum elektryczny o takiej częstotliwości, z jaką obraca się jego wirnik.

Ponieważ istnieje tak wiele źródeł szumu radiowego, istnieje wiele sposobów zakłócania sygnału radiowego. Ważne jest, aby pamiętać o tych możliwych źródłach szumu podczas pracy z urządzeniami radiowymi. Wiedza na ten temat jest bardzo cenna przy rozwiązywaniu problemów radiowych.

Multipleksowanie i protokoły

Kiedy transmitujesz przez radio, każdy, kto ma zgodny odbiornik, może odbierać Twój sygnał. Nie ma przewodu, który zamykałby sygnał, więc jeśli dwa nadajniki emitują sygnał w tym samym czasie, to będą ze sobą kolidować. To jest największą bolączką radia: dany odbiornik nie może w żaden sposób wiedzieć, kto wysłał odebrany sygnał. Dla kontrastu rozważ szeregowe połączenie przewodowe: kiedy otrzymasz impulsy elektryczne w przewodzie szeregowym, to możesz być rzeczywiście pewien, że pochodzą z urządzenia na drugim końcu przewodu. Nie masz takiej gwarancji w przypadku radia. To tak, jakbyś miał zawiązane oczy na przyjęciu i wszyscy inni goście mieli taki sam głos. Aby dowiedzieć się, kto do Ciebie mówi, należałoby ustalić ścisłe zasady — każda osoba powinna wyraźnie zidentyfikować się na początku i na końcu konwersacji, i nikt nie powinien przerywać jej w tym czasie. Innymi słowy — chodzi o protokoły.

Pierwszą rzeczą, którą wszyscy na tym przyjęciu musieliby zrobić, byłoby ustalenie kolejności mówienia. W ten sposób każdy mógłby mieć na chwilę Twoją uwagę. Współużytkowanie w komunikacji radiowej jest nazywane **multipleksowaniem** (ang. *multiplexing*), a ta forma współdzielenia jest nazywana **multipleksowaniem z podziałem czasu** (ang. *time-division multiplexing*). Każdy nadajnik ma nadany wycinek czasu, w którym transmituje.

Oczywiście to zależy od tego, czy wszystkie nadajniki są zsynchronizowane. Gdy nie są, multipleksowanie z podziałem czasu może nadal działać dość dobrze, jeśli wszystkie nadajniki mówią mniej, niż nasłuchują (pamiętaj o pierwszej regule miłości i sieci z rozdziału 1.: słuchaj więcej, niż mówisz). Jeśli dany nadajnik wysyła sygnał tylko przez kilka milisekund w każdej sekundzie i jeżeli istnieje ograniczona liczba nadajników, to szanse, że dwie wiadomości będą się nakładać lub **kolidować** ze sobą, są stosunkowo niewielkie. Niniejsze wytyczne, w połączeniu z prośbą o wyjaśnienie od odbiornika (reguła numer trzy), mogą zapewnić dobrą komunikację RF.

Wróćmy na przyjęcie. Jeśli każda osoba mówiłaby w innej tonacji, mógłbyś na tej podstawie odróżnić poszczególne osoby. W warunkach radiowych nazywa się to **multipleksowaniem z podziałem częstotliwości** (ang. *frequency-division multiplexing*). Oznacza to, że odbiornik musi mieć możliwość odbierania na kilku częstotliwościach jednocześnie. Ale jeśli istnieje koordynator przydzielający częstotliwości dla każdej pary nadajnika i odbiornika, jest to dość skuteczne.

Różne kombinacje multipleksowania z podziałem czasu i częstotliwości są używane w każdym systemie transmisji radia cyfrowego. Dobrą wiadomością jest to, że przez większość czasu nie trzeba o tym myśleć, ponieważ radia obsługują to za Ciebie.

Multipleksowanie pomaga w transmisji poprzez organizację włączania się nadajników i rozróżnienie ich pomiędzy sobą z użyciem częstotliwości, ale nie zajmuje się treścią tego, co zostanie przekazane. W tym miejscu pojawiają się protokoły danych. Jak już wiesz, protokoły danych umożliwiły powstanie sieci przewodowych, a w sieciach bezprzewodowych odgrywają równie istotną rolę. Aby upewnić się, że wiadomość jest jasna, powszechnie stosuje się protokoły danych poza multipleksowaniem. Na przykład Bluetooth, ZigBee i Wi-Fi nie są niczym więcej niż protokołami danych sieciowych nałożonymi na sygnał radiowy. Wszystkie trzy mogą być równie łatwo zrealizowane w sieci przewodowej (i w pewnym sensie tak jest z Wi-Fi: używa tej samej warstwy TCP/IP, z której korzysta sieć Ethernet). Założenia tych protokołów są takie same jak w sieci przewodowej, co umożliwia zrozumienie transmisji bezprzewodowej danych, nawet jeśli nie jesteś specjalistą w technologii radiowej. Pamiętaj o zasadach i metodach rozwiązywania problemów, które stosowałeś przy sieciach przewodowych, ponieważ będziesz z nich korzystać ponownie w projektach sieci bezprzewodowej. Metody wymienione w tym miejscu to tylko nowe narzędzia w Twoim zestawie narzędzi do rozwiązywania problemów. Będziesz ich potrzebował w kolejnych projektach.

Nadajniki, odbiorniki i radiowe urządzenia nadawczo-odbiorcze

Kiedy wybrać parę nadajnik-odbiornik, a kiedy urządzenie nadawczo-odbiorcze? Najprostsza odpowiedź brzmi następująco: jeśli potrzebujesz informacji zwrotnych z urządzenia, do którego transmitujesz, to potrzebujesz urządzenia nadawczo-odbiorczego. W większości przypadków najprościej jest używać urządzenia nadawczo-odbiorczego. Faktem jest, że odkąd urządzenia nadawczo-odbiorcze stały się tańsze w produkcji (a co za tym idzie w sprzedaży), to parę nadajnik-odbiornik coraz trudniej znaleźć.

Dostępnych jest wiele różnych rodzajów urządzeń nadawczo-odbiorczych. Najprostsze cyfrowe urządzenia radiowe nadawczo-odbiorcze na rynku podłącza się bezpośrednio do pinów szeregowej transmisji i odbioru mikrokontrolera. Wszystkie dane szeregowe, które wysyłasz po linii transmisji, są nadawane bezpośrednio jako sygnał radiowy. Wszelkie impulsy otrzymane przez urządzenie nadawczo-odbiorcze są wysyłane do linii odbioru Twojego mikrokontrolera. Jest to proste połączenie, ale trzeba samodzielnie zarządzać całą konwersacją. Jeżeli otrzymujące urządzenie nadawczo-odbiorcze zgubi bit danych, otrzymasz zniekształconą wiadomość. Wszelkie znajdujące się w pobliżu urządzenia radiowe pracujące na tym samym zakresie częstotliwości mogą mieć wpływ na jakość odbioru. Dopóki pracujesz tylko z dwoma odbiornikami radiowymi bez żadnych zakłóceń, urządzenia nadawczo-odbiorcze tego typu sprawdzają się dość dobrze. Jest to jednak rzadki przypadek.

Obecnie większość dostępnych na rynku urządzeń nadawczo-odbiorczych implementuje protokoły sieciowe zajmujące się zarządzaniem konwersacją. W rozdziale 2. modem Bluetooth ignorował sygnały z innych radioodbiorników, z którymi nie był skojarzony, i obsługiwał sprawdzanie błędów. Radia XBee, których będziesz używać w następnym projekcie, będą robić to samo i o wiele więcej, o czym przekonasz się w rozdziale 7. To wymaga, abyś dowiedział się trochę więcej o protokołach sieciowych, ale korzyści, jakie zyskujesz, są warte poniesienia tych niewielkich kosztów.

Największą różnicą pomiędzy sieciowymi radiami a prostymi modułami nadawczo-odbiorczymi jest to, że każde urządzenie w sieci ma adres. Oznacza to, że musisz zdecydować, do którego urządzenia zamierzasz mówić (być może chcesz mówić do wszystkich innych urządzeń w sieci).

Z powodu komplikacji zarządzania siecią wszystkie radia sieciowe mają zwykle dwa tryby działania: tryb polecenia i tryb danych (zgodnie z opisem w rozdziale 2.). Zapoznając się z protokołem komunikacji dla radia sieciowego, jedną z pierwszych rzeczy, o których się dowiadujesz, jest informacja o sposobie przełączania się z trybu poleceń do trybu danych i z powrotem.

X

Projekt 10.

Dupleksowa transmisja radiowa

W tym przykładzie podłączysz urządzenie nadawczo-odbiorcze RF i potencjometr do mikrokontrolera. Każdy mikrokontroler będzie wysyłał sygnał do drugiego mikrokontrolera, kiedy odczyt jego potencjometru zmieni się o więcej niż 10 punktów. Kiedy którykolwiek odbierze wiadomość, zaświeci się dioda LED. Każde urządzenie ma również diodę LED do przekazywania lokalnej informacji zwrotnej.

Urządzenia nadawczo-odbiorcze RF używane w tym projekcie implementują bezprzewodowy protokół sieciowy 802.15.4, na którym oparty jest ZigBee. W tym przykładzie nie zostały faktycznie wykorzystane korzyści ZigBee i niewiele z korzyści 802.15.4. Protokoły 802.15.4 i ZigBee zostały opracowane w celu umożliwienia wielu różnym obiektom komunikacji w elastycznym schemacie sieci. Każde radio ma adres i każdorazowo, kiedy wysyła wiadomość, musi określić adres, na który wysyła. Może również wysyłać **wiadomość rozgłoszeniową** (ang. *broadcast message*), skierowaną do każdego radia w zasięgu — więcej informacji na ten temat znajdziesz w rozdziale 7. Na razie podasz każdemu ze swoich dwóch radioodbiorników adres drugiej strony, aby mogły przekazywać między sobą wiadomości.

Jest wiele rzeczy, które mogą pójść nie tak z bezprzewodową transmisją, i w związku z tym że transmisje radiowe nie są wykrywalne bez działającego radia, rozwiązanie problemu może być trudne. Z tego powodu zbudujesz ten projekt w kilku etapach. Po pierwsze, skomunikujesz się z samym modułem radiowym szeregowo, aby ustawić jego lokalny adres oraz adres docelowy. Następnie napiszesz program, aby mikrokontroler wysyłał wiadomości, gdy zmienia się wartość potencjometru, i nasłuchiwał wiadomości z drugiego radia podłączonego do komputera osobistego. Na koniec sprawisz, że dwa mikrokontrolery będą komunikować się ze sobą za pomocą radia.

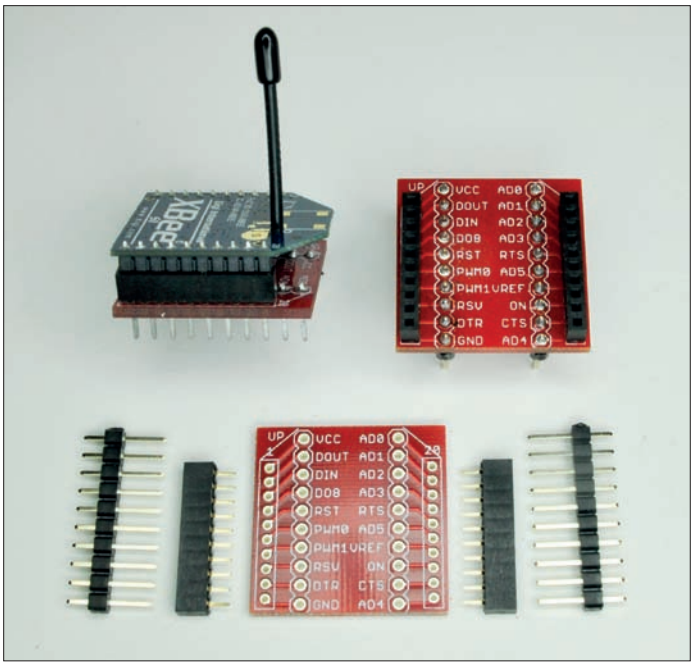
MATERIAŁY:

- **2 płytki stykowe,**
- **1 adapter XBee na USB,**
- **2 Arduino** (modele Arduino Fio są przyjazną alternatywą zaprojektowaną do pracy z XBee),
- **2 moduły Digi XBee 802.15.4 RF,**
- **2 tarcze bezprzewodowe Arduino.** Jeśli nie zdecydujesz się na tarczę bezprzewodową lub Arduino Fio, możesz użyć części wyszczególnionych poniżej, aby połączyć XBee z Arduino:
- **2 regulatory napięcia 3,3 V,**
- **2 kondensatory 1 μF,**
- **2 kondensatory 10 μF,**
- **2 płytki z wyprowadzeniami połączeń dla XBee,**
- **4 listwy kołkowe z rozstawem 2,54 mm,**
- **4 gniazda do listew kołkowych 2 mm,**
- **6 diod LED,**
- **2 potencjometry.**

Krok 1. Konfiguracja szeregowa modułów XBee

Najprostszym sposobem podłączenia XBee do komputera osobistego jest użycie adaptera szeregowego XBee na USB. Odkąd opublikowano pierwsze wydanie tej książki, popularność XBee rośnie wykładniczo i obecnie dostępnych jest wiele wersji (większość dostawców elektroniki hobbystycznej oferuje swoją wersję). Wszystkie są zasadniczo adapterami łącza szeregowego na USB zamontowanymi na płytce z pinami rozstawionymi tak, aby dopasować radio XBee. Pierwsze zdjęcie na rysunku 6.12 pokazuje dwie opcje: pytkę adaptera XBee na USB firmy Adafruit i XBee Explorer firmy Spark Fun. Oba mają zamontowane diody LED do wskazania szeregowej transmisji i odbioru. Model Adafruit ma również diody LED wskazujące, czy radio jest skojarzone z siecią lub czy jest w stanie uśpienia. Wskaźnik LED trybu uśpienia jest dołączony do pinu 13, który ma stan niski, gdy radio jest w trybie uśpienia, i wysoki, gdy jest aktywne. Wskaźnik skojarzenia LED jest dołączony do pinu 15. Kiedy radio jest skojarzone, dioda LED miga.

Podłącz swój moduł XBee do adaptera, następnie podłącz go do portu USB komputera i otwórz swój ulubiony program terminala szeregowego.



Rysunek 6.8.
Płytka stykowa XBee, w różnych etapach. U dołu: goła płytka z niezbędnymi listwami kołkowymi. U góry po prawej: skończona płytka. U góry po lewej: skończona płytka z zamontowanym XBee

Montaż radia XBee na płytce z wyprowadzeniami

Radia XBee mają piny w rozstawie 2 mm od siebie — za wąsko, aby zmieściły się na płytce stykowej. Możesz albo przylutować przewody do każdego pinu, aby rozszerzyć nóżki, albo zamontować moduł na płytce z wyprowadzeniami. SparkFun ma taką płytkę — jest to płytka z wyprowadzeniami dla modułu XBee (nr części BOB-08276). Kiedy masz już płytkę stykową, przylutuj listwy do wewnętrznych rzędów. Będą one podłączane do płytki stykowej. Następnie dołącz gniazda z rozstawem 2 mm. XBee będzie w nie wetknięty, więc może okazać się konieczne dopasowanie ich do płytki przez założenie ich najpierw na XBee, a następnie włożenie go do płytki razem z dołączonymi gniazdami.

Protokół poleceń XBee jest wymagający co do tego jak kończysz polecenia, oczekując, że każde polecenie w wierszu jest zakończone tylko znakiem powrotu karetki (\r lub ASCII 13). Jednak większość programów terminali szeregowych umożliwia ustawienie, co jest wysyłane, gdy naciśniesz klawisz *Return*.

W CoolTerm dla OS X i Windows kliknij przycisk *Options* (opcje) i zmień *Enter Key Emulation* (wprowadź emulację klawiszy) na *CR* (patrz rysunek 6.10). W PuTTY dla Windows i Ubuntu Linux wybierz zakładkę *Terminal Configuration* (konfiguracja terminala) i zaznacz *Implicit LF in every CR* (ukryj LF w każdym CR — patrz rysunek 6.11).

Kiedy już skonfigurujesz terminal szeregowy, otwórz port i wpisz:

```
+++
```

Nie naciskaj klawisza *Return* (*Enter*) ani żadnego innego klawisza przez co najmniej jedną sekundę. XBee powinien odpowiedzieć tak:

```
OK
```

Krok ten jest podobny do konfiguracji modemu Bluetooth w rozdziale 2., gdzie wpisywałeś $$$, aby przejść w tryb poleceń. XBee używa zestawu poleceń w stylu AT i +++ przełącza go w tryb poleceń. Jednosekundowa pauza po tym ciągu jest nazywana **czasem ochronnym**. Jeśli nic nie zrobisz, moduł porzuci tryb poleceń po 10 sekundach. Więc jeśli czytasz to podczas pisania, może zajść potrzeba wpisania +++ ponownie przed następnym krokiem.

Które radio XBee kupić?

Digi produkuje kilka odmian modułu XBee i wybór odpowiedniego może być trudny. Moduły typu punkt do wielu punktów są najbardziej podstawowe, zapewniają możliwość wysyłania skierowanych wiadomości lub emisji wiadomości do wszystkich radioodbiorników w sieci. Tworzą sieci o topologii gwiazdy, jak opisano w rozdziale 3., i są najłatwiejsze do konfiguracji. Moduły kratowe (ang. *mesh*) **umożliwiają utworzenie wielowarstwowych sieci kratowych, ale ich konfiguracja i obsługa są bardziej skomplikowane. Doskonała książka Roberta Faludiego** *Building Wireless Sensor Networks* **(Budowa sieci czujników bezprzewodowych) dogłębnie opisuje radiowe sieci kratowe. Jednak dla większości projektów hobbystycznych sieci kratowe to przesada, więc w tej książce używane są radia typu punkt do wielu punktów.**

Najprostszy model, moduł małej mocy XBee 802.15.4, jest najtańszy i ma nominalny zasięg transmisji około 300 m. Modele o poszerzonym zasięgu XBee-PRO 802.15.4 mają większy zasięg — około 1 mili (1,6 km) (jednak w praktyce nigdy nie udało mi się osiągnąć więcej niż ćwierć mili — 400 m), ale zużywają więcej mocy. Zarówno modele małej mocy, jak i modele PRO mogą być używane zamiennie do projektów z tej książki.

Radia Digi mają kilka opcji anteny. Tylko dwie opcje, antena przewodowa lub antena w układzie scalonym, nie wymagają żadnych dodatkowych części, więc polecam je do tych projektów.

Poniżej wyszczególnione są numery tych modeli:

Moduł małej mocy Digi XBee 802.15.4: XB24-AWI-001 lub XB24-ACI-001.

Moduł o rozszerzonym zasięgu Digi XBee-PRO 802.15.4: XBP24-AWI-001 lub XBP24-ACI-001.

Rysunek 6.9.
Moduły XBee 802.15.4. Po lewej moduł małej mocy XBee z anteną w układzie scalonym, a po prawej moduł z rozszerzonym zasięgiem XBee-PRO z anteną przewodową
Zdjęcia zamieszczamy dzięki uprzejmości Digi International.

Po uzyskaniu odpowiedzi OK ustaw adres XBee. Protokół XBee używa adresów 16- lub 64-bitowych, więc adres składa się z dwóch części: słowo niskie i słowo wysokie (w pamięci komputera dwa lub więcej bajtów używanych dla pojedynczej wartości jest czasami określane jako słowo (ang. *word*)). W tym projekcie będziesz używać 16-bitowego adresowania, w związku z tym możesz wybrać własny adres. Do tego celu potrzebujesz tylko niższego słowa adresu. Wpisz:

```
ATMY1234\r
```

Pamiętaj! `\r` oznacza, że powinieneś nacisnąć *Enter* lub *Return*). Aby potwierdzić, że ustawiłeś adres, należy wpisać:

```
ATMY\r
```

Moduł powinien odpowiedzieć:

```
1234
```

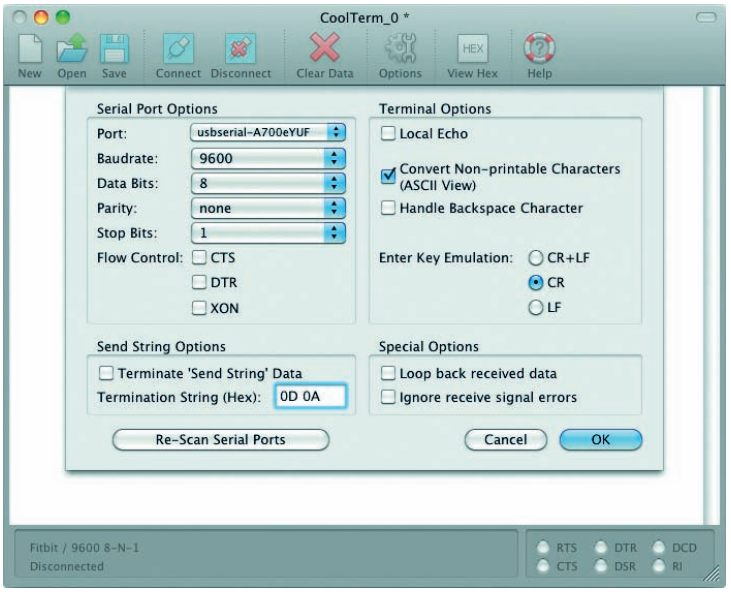
Następnie ustaw *docelowy adres* XBee (adres, na który będą wysyłane wiadomości). Upewnij się, że jesteś w trybie polecenia (+++), a następnie wpisz `ATDL\r`.

Prawdopodobnie otrzymasz:

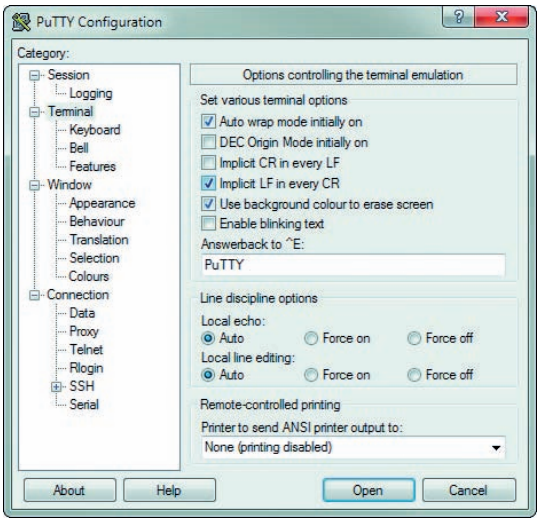
```
0
```

Domyślnym adresem docelowym tych modułów jest 0. Adres docelowy to dwa słowa, więc aby zobaczyć wyższe słowo, wpisz:

```
ATDH\r
```



Rysunek 6.10.
Menu opcji CoolTerm. Aby użyć CoolTerm do konfiguracji radioodbiorników Digi (XBee i innych), ustaw Enter Key Emulation na CR



Rysunek 6.11.
Menu Terminal Configuration w PuTTY. Aby użyć PuTTY do konfiguracji radia Digi (XBee i innych), wybierz „Implicit LF in every CR"

Tą parą poleceń można również ustawić adres docelowy:

```
ATDL5678\r
ATDH0\r
```

Te radia mają również identyfikator grupy, czyli Personal Area Network (PAN). Wszystkie radia z tym samym identyfikatorem PAN mogą komunikować się ze sobą i ignorować radia o innym identyfikatorze PAN. Ustaw identyfikator PAN dla swojego radia następująco:

```
ATID1111\r
```

XBee odpowie na to polecenie, podobnie jak na wszystkie polecenia:

```
OK
```

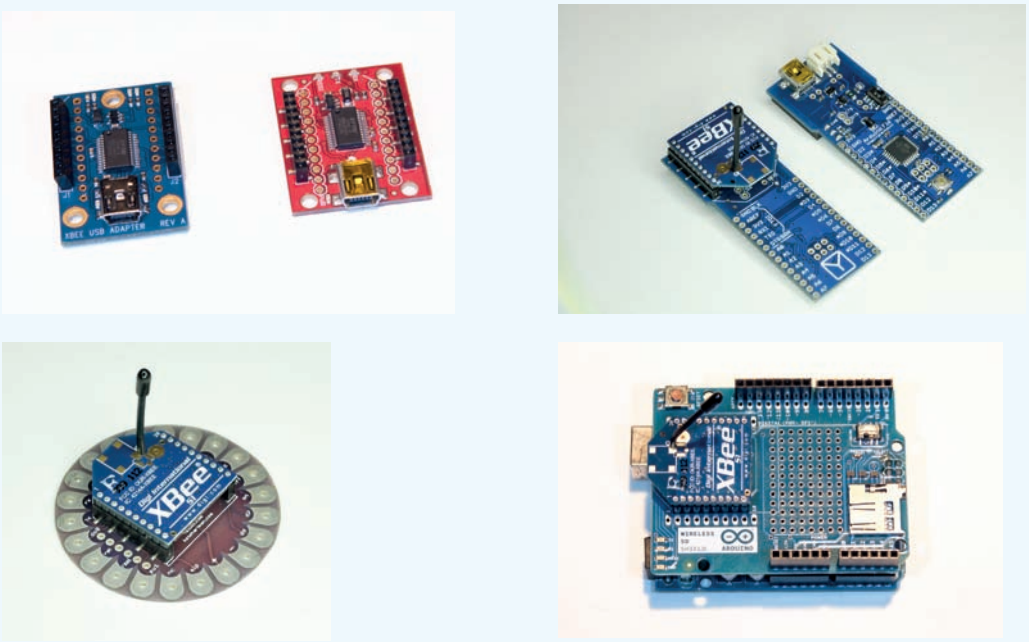
Upewnij się, że dodasz `WR` po ostatnim poleceniu, które zapisuje parametry w pamięci radia. W ten sposób pozostaną skonfigurowane tak, jak chcesz, nawet po wyłączeniu radia. Na przykład:

```
ATID1111,WR\r
```

Jakie akcesoria XBee kupić?

Na rynku dostępnych jest tak wiele akcesoriów XBee, że wybór właściwej części może być trudny. Istnieją trzy sposoby wykorzystania XBee w projektach w tej książce:

- **XBee do komputera za pośrednictwem adaptera XBee na USB. Dostępnych jest wiele takich modeli. Rysunek 6.12 pokazuje dwa modele, czerwony SparkFun XBee Explorer i niebieską płytę adaptera XBee na USB — dostępną u dostawców Adafruit i Parallax.**
- **XBee do mikrokontrolera. Do tego celu można użyć płytki z wyprowadzeniami, na której zainstalujesz XBee, lub bezprzewodowej tarczy dla standardowych Arduino. Można również użyć Arduno Fio, który ma wbudowane gniazdo dla XBee.**
- **XBee samodzielnie. Do tego celu płytka z wyprowadzeniami XBee świetnie się nadaje, podobnie jak płytka XBee LilyPad. XBee LilyPad, XBee Explorer Regulated (nr części Spark Fun: WRL-09132) i zestaw adaptera Adafruit XBee (nr części Adafruit: 126) posiadają wbudowane regulatory napięcia, więc możesz użyć swojego XBee z różnego rodzaju źródłami zasilania.**



Rysunek 6.12.
Akcesoria XBee. Zgodnie z kierunkiem wskazówek zegara, począwszy od lewego górnego rogu: adapter XBee na USB, XBee Explorer, Arduino Fio, tarcza bezprzewodowa Arduino, XBee LilyPad

Po skonfigurowaniu swojego radia odłącz program terminala szeregowego i odłącz płytkę od komputera. Następnie usuń XBee z obwodu, włóż drugi moduł i skonfiguruj go przy użyciu tej samej procedury. Nie ustawiaj adresu docelowego radia na tę samą wartość jego adresu źródłowego, bo będzie rozmawiał tylko z samym sobą! Możesz użyć dowolnego 16-bitowego adresu dla swojego radia. Poniżej przedstawiona jest typowa konfiguracja dla dwóch odbiorników radiowych, które będą komunikować się ze sobą (nie zapomnij dodać `WR` do ostatniego polecenia).

	ATMY	ATDL	ATDH	ATID
Radio 1.	1234	5678	0	1111
Radio 2.	5678	1234	0	1111

Można łączyć polecenia w tym samym wierszu, oddzielając je przecinkami. Na przykład, aby uzyskać oba słowa adresu źródłowego modułu, należy wpisać:

```
ATDL, DH\r
```

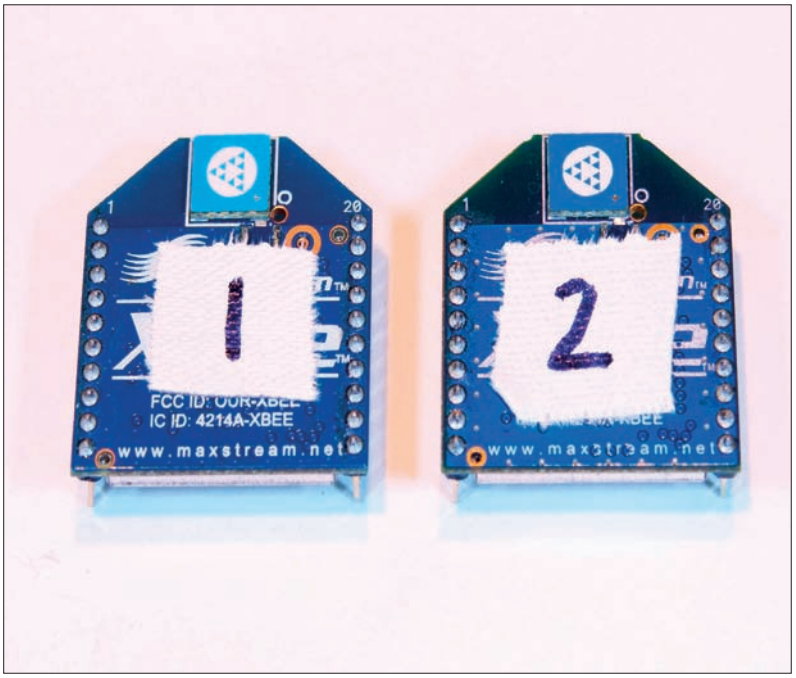
Moduł odpowie oboma wyrazami naraz. Podobnie, aby ustawić oba słowa adresu docelowego, a następnie sprawić, że moduł zapisze je w pamięci — aby pamiętał adres, również kiedy jest wyłączony — wpisz:

```
ATDL5678, DH0, WR\r
```

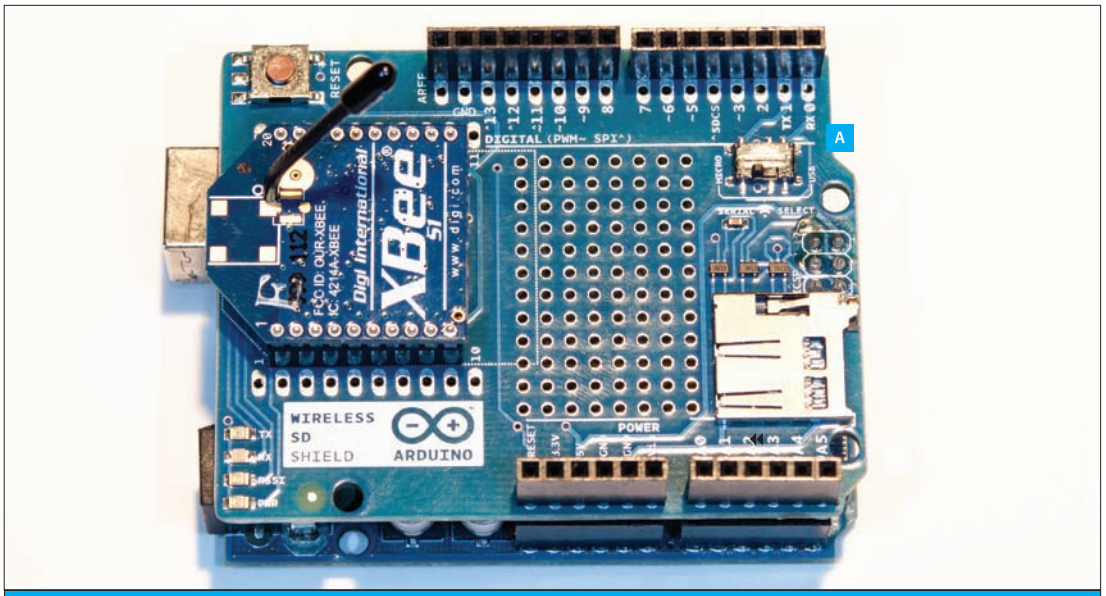
Moduł odpowie na wszystkie trzy polecenia naraz:

```
OK OK OK
```

X



Rysunek 6.13.
Aby rozróżnić radia, przyklej im etykiety z kawałków taśmy. Będziesz je przełączać między adapterem łącza szeregowego na USB i mikrokontrolerem parokrotnie — wówczas łatwo je pomylić



A. Przełącznik szeregowy ustawiony na Micro (po lewej).

Tarcza bezprzewodowa Arduino

Obecnie kilka firm produkuje bezprzewodowe tarcze dla Arduino, które będą działać w tym projekcie. Oryginalna tarcza Arduino dla XBee używana w pierwszym wydaniu tej książki została znacząco przeprojektowana. Obecnie jest nazywana tarczą bezprzewodową Arduino (ponieważ inne urządzenia radiowe z tym samym rozstawem gniazd mogą również pracować na tej tarczy), posiada kilka przydatnych funkcji, takich jak obszar do tworzenia prototypów, opcjonalne gniazdo kart microSD (zobaczysz przykład z kartą SD w dalszej części książki) i przełącznik wyboru szeregowy pozwalający zmienić piny dla połączenia szeregowego XBee.

Kiedy przełącznik wyboru szeregowego tarczy bezprzewodowej jest ustawiony na „Micro", XBee zostanie podłączony do komunikacji z mikrokontrolerem ATMega328 na Arduino. Po przełączeniu na „USB" będzie podłączony do komunikacji bezpośredniej przez konwerter portu szeregowego na USB na Arduino, z pominięciem mikrokontrolera. W tej pozycji przełącznika możesz użyć konwertera łącza szeregowego na USB Arduino, aby skonfigurować swoje XBee.

Kiedy programujesz Arduino, dobrym pomysłem jest usunięcie XBee, aby szeregowa komunikacja radiowa nie zakłócała ładowania programu.

Aby skonfigurować radio XBee na tarczy przy użyciu płyty Arduino jako konwertera łącza szeregowego na USB, zaprogramuj Arduino pustym szkicem, takim jak:

```
void setup() {
}

void loop() {
}
```

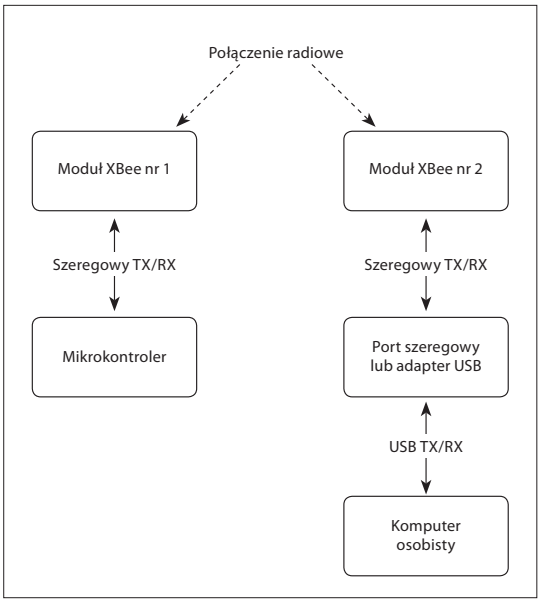
Następnie ustaw przełącznik wyboru szeregowego na USB. Otwórz połączenie szeregowe terminala do portu szeregowego na płycie Arduino i wysyłaj polecenia, jak pokazano w punkcie „Krok 1. Konfiguracja szeregowa modułów XBee". Kiedy skonfigurujesz radio, odłącz XBee od tarczy, ustaw przełącznik wyboru szeregowego z powrotem na pozycji „Micro" i programuj swoje Arduino tak jak zwykle.

Krok 2. Zaprogramowanie mikrokontrolera, aby używał modułu XBee

OK! Teraz jesteś gotowy, aby sprawić, że dwa mikrokontrolery będą rozmawiać ze sobą bezprzewodowo za pośrednictwem radia XBee. Na razie zostawisz jedno XBee podłączone do komputera za pośrednictwem adaptera USB. Kiedy potwierdzisz, że to działa, zastąpisz komputer drugim Arduino i XBee. Rysunek 6.14 pokazuje schemat tego, co jest z czym połączone w tym kroku. W kroku 3. usuniesz komputer osobisty i zastąpisz go drugim Arduino.

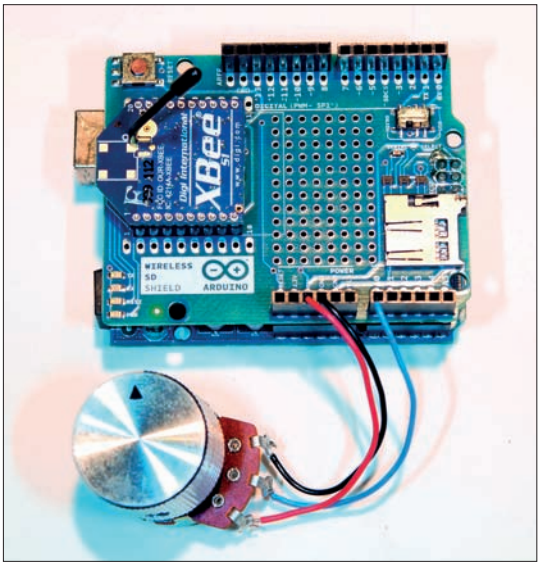
Rysunek 6.15 pokazuje moduł XBee dołączony do standardowego Arduino przy użyciu bezprzewodowej tarczy Arduino. Rysunek 6.16 pokazuje, jak to zrobić w przypadku użycia tylko płytki z wyprowadzeniami dla XBee zamiast tarczy. Zauważ, że XBee jest podłączony do regulatora 3,3 V. Szeregowe połączenie wejścia-wyjścia XBee toleruje 5 V, co oznacza, że może akceptować sygnały danych 5 V, ale moduł działa na 3,3 V. Arduino Uno i nowsze modele mają bardziej niezawodny regulator 3,3 V, który może zasilać XBee, ale nie zaszkodzi dodanie dedykowanego elementu, jak pokazano tutaj.

Po podłączeniu modułu nadszedł czas na oprogramowanie mikrokontrolera do wysyłania danych poprzez XBee. W tym programie mikrokontroler skonfiguruje adres docelowy XBee podczas uruchamiania, a następnie wyśle odczyt analogowy, gdy odczyt na analogowym pinie 0 znacząco się zmieni.



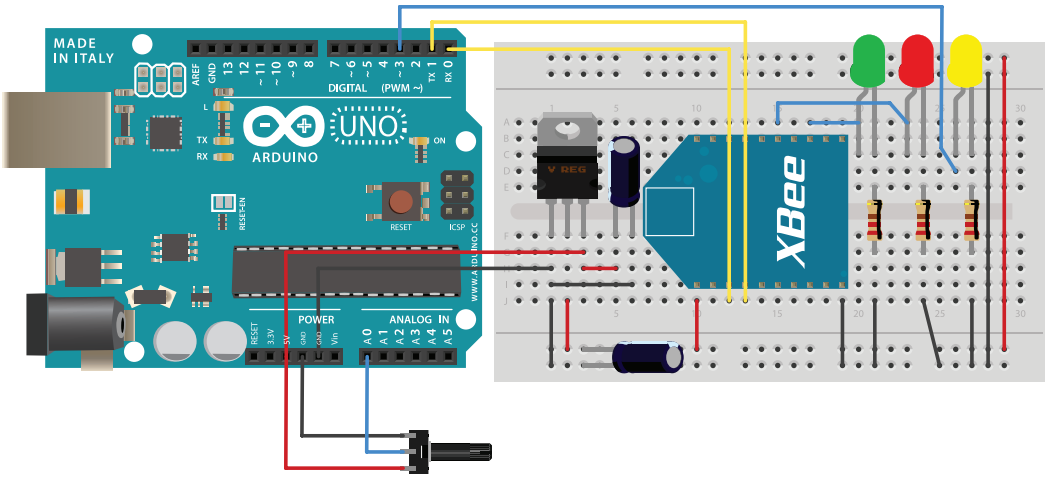
U góry, po prawej

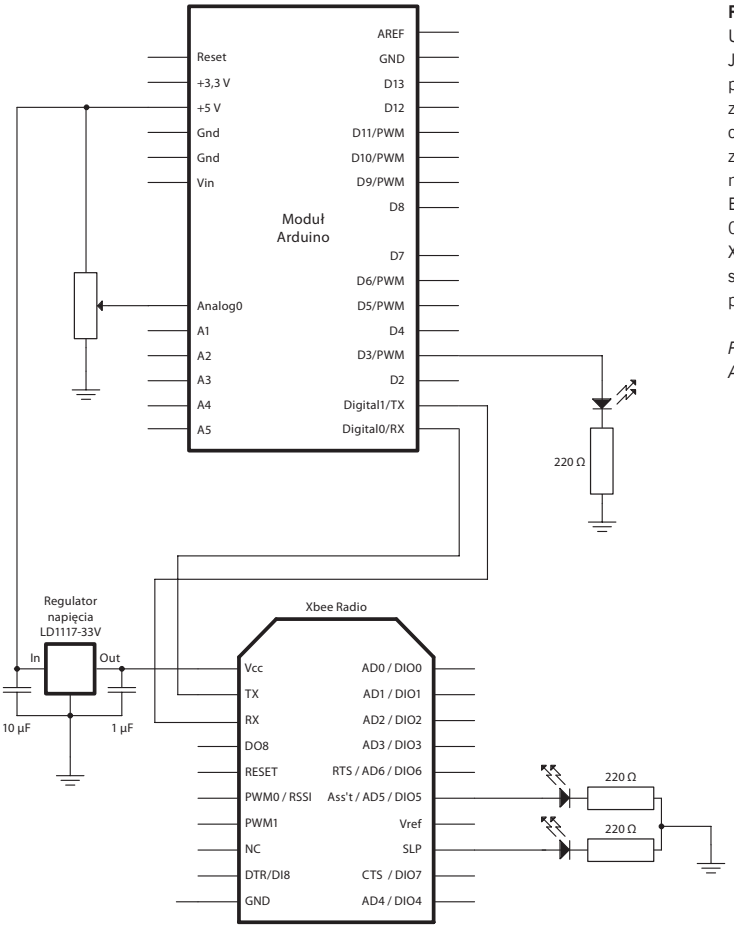
Rysunek 6.14.
XBee nr 1 jest podłączone do mikrokontrolera. XBee nr 2 jest połączone za pomocą konwertera USB lub portu szeregowego do komputera PC. Umożliwia to połączenie bezprzewodowe między komputerem PC a mikrokontrolerem



U dołu po prawej

Rysunek 6.15.
Arduino i tarcza bezprzewodowa z potencjometrem podłączone do analogowego pinu 0. Obwód jest taki jak na rysunku 6.16, ale nie ma diody LED na pinie 9. Diody LED dołączone do XBee są wbudowane w tarczę





Rysunek 6.16.
U góry: XBee podłączone do Arduino. Jeśli nie chcesz używać tarczy, możesz połączyć te elementy za pomocą płytki z wyprowadzeniami. Zauważ, że ten obwód zakłada, że posiadasz gołą płytkę z wyprowadzeniami, bez regulatora napięcia. Jeśli używasz Spark Fun XBee Explorer Regulated (nr części WRL-09132) lub zestawu adaptera Adafruit XBee (nr części 126), nie potrzebujesz stabilizatora napięcia, ponieważ obie te płytki mają wbudowane regulatory 3,3 V

Po lewej: Schemat obwodu dla połączenia Arduino-XBee.

Zrób to Najpierw nadaj nazwy pinom wejścia-wyjścia i skonfiguruj niektóre zmienne do śledzenia zmian potencjometru.

```
const int sensorPin = A0;      // czujnik wejściowy
const int analogLed = 3;       // dioda LED, która zmienia jasność zgodnie
                               // z otrzymanymi wartościami
const int threshold = 10;      // próg dla zmiany czujnika

int lastSensorReading = 0;     // poprzedni stan czujnika

String inputString = "";
```

» Następnie w metodzie `setup()` skonfiguruj transmisję szeregową, ustaw tryby na pinie wejścia-wyjścia i skonfiguruj adres docelowy XBee.

```
void setup() {
  // konfiguracja komunikacji szeregowej:
  Serial.begin(9600);

  // konfiguracja pinu wyjściowego:
  pinMode (analogLed, OUTPUT);

  // ustaw adres docelowy XBee:
  setDestination();
  // zamigaj diodą TX, wskazując, że główny program zaraz się zacznie:
   blink(analogLed, 3);
}
```

» Konfiguracja XBee, obsługiwana przez metodę `setDestination()`, przebiega w ten sam sposób jak poprzednio, tylko obecnie uczymy mikrokontroler, jak to zrobić.

```
void setDestination() {
  // przełącz radio w tryb poleceń:
  Serial.print("+++");
  // poczekaj, aż radio odpowie „OK\r”
  char thisByte = 0;
  while (thisByte != '\r') {
    if (Serial.available() > 0) {
      thisByte = Serial.read();
    }
  }

  // ustaw adres docelowy przy użyciu 16-bitowego adresowania.
  // Jeśli używasz dwóch modułów, adresem docelowym jednego radia
  // powinien być adres drugiego radia, i odwrotnie:
  Serial.print("ATDH0, DL5678\r");
  // ustaw mój adres przy użyciu 16-bitowego adresowania:
   Serial.print("ATMY1234\r");
  // ustaw PAN ID. Jeśli pracujesz w miejscu, gdzie wiele osób
  // używa XBee, należy ustawić własne PAN ID,
  // inne niż w pozostałych projektach:
  Serial.print("ATID1111\r");
  // przełącz radio w tryb danych:
  Serial.print("ATCN\r");
}
```

» **Zmień adres docelowy na adres tego radia, które dołączasz do komputera osobistego, a nie tego, które jest podłączone do mikrokontrolera.**

Metoda `blink()` jest taka sama jak prezentowane już wcześniej w tej książce. Miga diodą LED, aby wskazać, że zakończyła się metoda `setup`.

```
void blink(int thisPin, int howManyTimes) {
  // zamigaj diodą LED:
  for (int blinks=0; blinks< howManyTimes; blinks++) {
    digitalWrite(thisPin, HIGH);
    delay(200);
    digitalWrite(thisPin, LOW);
    delay(200);
  }
}
```

Główna pętla obsługuje przychodzące dane szeregowe, odczytuje potencjometr i wysyła dane, jeśli pojawiła się wystarczająca zmiana w odczycie potencjometru.

```
void loop() {
  // nasłuchuj przychodzących danych szeregowych:
  if (Serial.available() > 0) {
    handleSerial();
  }
  // nasłuchuj potencjometru:
  int sensorValue = readSensor();

  // jeśli jest coś do wysłania, wyślij to:
  if (sensorValue > 0) {
    Serial.println(sensorValue, DEC);
  }
}
```

Dwie inne metody są wywoływane z pętli głównej: `handleSerial()`, która nasłuchuje ciągów znaków numerycznych ASCII i konwertuje je na bajty, aby ustawić jasność diody LED na wyjściu PWM; oraz `readSensor()`, która odczytuje potencjometr i sprawdza, czy zmiana na nim jest wystarczająco duża, aby wysłać nową wartość przez radio. Oto te metody:

```
void handleSerial() {
  char inByte = Serial.read();

  // zapisz tylko znaki numeryczne ASCII (ASCII 0 - 9):
  if (isDigit(inByte)){
    inputString = inputString + inByte;
  }

  // jeśli otrzymasz znak ASCII dla nowej linii:
  if (inByte == '\n') {
    // skonwertuj ciąg znaków na liczbę :
    int brightness = inputString.toInt();
    // ustaw analogowe wyjście LED:
    analogWrite(analogLed, brightness);
    // wyczyść ciąg wejściowy dla następnej wartości:
    inputString = "";
    Serial.print(brightness);
   }
}

int readSensor() {
  int result = analogRead(sensorPin);
```

UWAGA! **Podczas programowania odłącz połączenia transmisji i odbioru XBee do mikrokontrolera (jeśli używasz tarczy bezprzewodowej Arduino, użyj przełącznika wyboru szeregowego). Komunikacja szeregowa z XBee może zakłócać komunikację szeregową z programującym komputerem. Po zaprogramowaniu mikrokontrolera można ponownie podłączyć linie transmisji i odbioru.**

Dokończenie z poprzedniej strony

```
  // czekaj na zmianę w stosunku do ostatniego odczytu,
  // która jest większa niż próg:
  if (abs(result - lastSensorReading) > threshold) {
    result = result/4;
    lastSensorReading = result;
  } else {
    // jeśli zmiana nie jest znacząca, zwróć 0:
    result = 0;
  }
  return result;
}
```

Zauważ, że w pętli głównej nie używasz żadnych poleceń AT. Dzieje się tak dlatego, że XBee wraca automatycznie do trybu danych (nazywanego **trybem jałowym** (ang. *idle mode*) w podręczniku użytkownika XBee) po wydaniu polecenia `ATCN` w metodzie `setDestination()`.

Pamiętaj, że w trybie danych wszelkie bajty wysłane do modemu typu AT przechodzą przez niego słowo w słowo. Jedynym wyjątkiem od tej reguły jest to, że po otrzymaniu ciągu +++ modem przełącza się w tryb poleceń. To zachowanie jest identyczne z zachowaniem modułu Bluetooth z rozdziału 2. (prawie każde urządzenie, które implementuje tego rodzaju protokół, działa w ten sposób). Jest to wspaniałe, ponieważ oznacza, że kiedy bieżącym trybem jest tryb danych, możesz wysyłać dane bez żadnych dodatkowych poleceń, pozwalając radiu samodzielnie obsługiwać całą korekcję błędów.

Kiedy zaprogramujesz mikrokontroler, ustaw adres docelowy XBee komputera na adres radia mikrokontrolera. (Jeśli zrobiłeś to we wcześniejszym kroku, nie musisz robić tego ponownie). Następnie przekręć potencjometr mikrokontrolera. Powinieneś otrzymać następujący komunikat w oknie terminala szeregowego:

```
120
```

Bieżąca liczba będzie się zmieniać, jeśli przekręcisz potencjometr. Może się nadpisywać w oknie szeregowym — w zależności od aplikacji terminala szeregowego — ponieważ nie wysyłasz znaku nowego wiersza. Gratulacje! Właśnie stworzyłeś pierwsze łącze bezprzewodowe między urządzeniami nadawczo-odbiorczymi. Pokręć potencjometrem, dopóki się nie znudzisz, a następnie przejdź do kroku 3.

X

Krok 3. Dwukierunkowa bezprzewodowa komunikacja między mikrokontrolerami

Ten krok jest prosty. Wszystko, co musisz zrobić, to zastąpić komputer z poprzedniego kroku drugim mikrokontrolerem. Połącz Arduino z drugim modułem XBee, jak pokazano na rysunku 6.16, lub użyj tarczy bezprzewodowej. Programy obu mikrokontrolerów będą prawie takie same, tylko adresy docelowe radia XBee będą różne. Ten program będzie zarówno wysyłał, jak i odbierał dane z modułów. Przekręcenie potencjometru spowoduje wysłanie liczby do drugiego mikrokontrolera. Kiedy mikrokontroler otrzymuje liczbę na porcie szeregowym, wykorzystuje ją do ustawienia jasności diody LED na pinie 3.

Najpierw podłącz drugi moduł XBee do drugiego mikrokontrolera. Następnie zaprogramuj oba mikrokontrolery poprzednim programem, upewniając się, że ustawiasz adres docelowy, jak zaznaczono w programie.

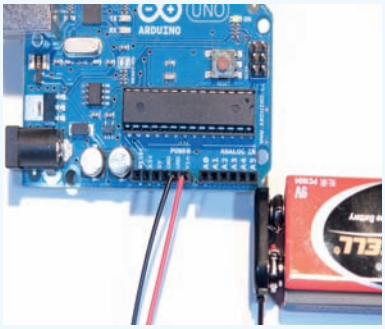
Kiedy zaprogramujesz oba moduły, włącz zasilanie i przekręć potencjometrem kilka razy. Podczas gdy kręcisz potencjometrem, dioda LED na pinie 3 drugiego modułu powinna się rozjaśniać i przyciemniać. Teraz masz możliwość dwukierunkowej bezprzewodowej komunikacji między dwoma mikrokontrolerami. Otwiera to możliwość wszelkiego rodzaju interakcji.

X

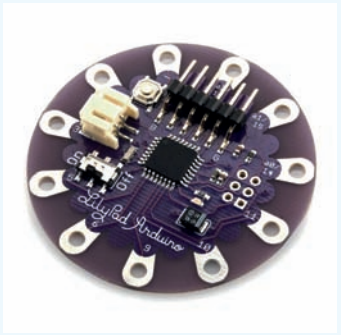
Bezprzewodowe i mobilne

Skoro masz już możliwość komunikacji bezprzewodowej, być może chciałbyś, aby Twój mikrokontroler również był mobilny. W tym celu wystarczy zasilić go z baterii. Istnieje prosty sposób, aby to zrobić dla standardowej płyty Arduino, i istnieje kilka modeli Arduino — i różne modele pochodne — zaprojektowanych do zasilania z baterii.

Najprostszym rozwiązaniem jest połączenie baterii z terminalem wejścia zasilania, jak pokazano na rysunku 6.17. Pin Vin (wejście napięciowe — ang. *Voltage input*) może przyjąć napięcie 6 – 15 V (Arduino Uno będzie działać przy niższym napięciu — udało mi się uruchomić je przy 3,7 V — ale nie zawsze jest to niezawodne). Możesz albo podłączyć baterię do pinów masy i Vin, albo zrobić wtyczkę i podłączyć ją do gniazda zasilania.



Rysunek 6.17.
Moduł Arduino zasilany z baterii 9 V

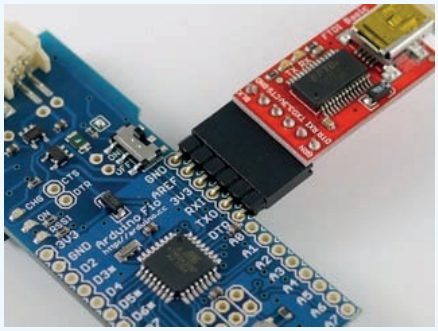


Rysunek 6.19.
LilyPad Arduino Simple, z gniazdem akumulatora litowo-polimerowego
Zdjęcie zamieszczamy dzięki uprzejmości Spark Fun.

Arduino Fio znakomicie nadaje się do projektów XBee. Posiada gniazdo dla XBee i gniazdo baterii dla baterii litowo-polimerowych 3,7 V, jak pokazano na rysunku 6.18. Gniazdo mini-USB na Fio tak naprawdę nie komunikuje się z mikrokontrolerem, tylko ładuje baterię. Aby zaprogramować Fio, potrzebujesz adaptera USB w stylu FTDI, jak opisano w rozdziale 2., możesz też zaprogramować go bezprzewodowo przez XBee. Programowanie Fio bezprzewodowo wymaga większego zrozumienia XBee, więc dobrym pomysłem jest zaprogramowanie go najpierw przy użyciu połączenia przewodowego. Więcej informacji na temat programowania Fio znajdziesz pod adresem *http://arduino.cc/en/Main/ArduinoBoardFioProgramming*.

LilyPad Arduino zostały stworzone do stosowania w ubraniach i innych miękkich przedmiotach. Je również programuje się przy użyciu adaptera szeregowego typu FTDI. Dostępnych jest kilka zasilaczy LilyPad dla akumulatorów litowo-polimerowych. LilyPad Arduino Simple ma gniazdo zasilania, więc można przyłączyć akumulator litowo-polimerowy bezpośrednio do płytki (patrz rysunek 6.19).

Dobrym pomysłem jest podłączenie modułu mikrokontrolera do zasilacza sieciowego lub zasilania USB podczas programowania i debugowania. Gdy bateria zaczyna słabnąć, moduł nie będzie działać spójnie, co może uniemożliwić debugowanie.



Rysunek 6.18.
Szczegóły Fio, pokazujące gniazdo baterii, i adapter portu szeregowego na USB podłączony do oprogramowania

Projekt 11.

Radia Bluetooth

W rozdziale 2. nauczyłeś się podłączać mikrokontroler do komputera osobistego przy użyciu radia Bluetooth. W tym przykładzie pokazano, jak połączyć dwa mikrokontrolery przy użyciu Bluetooth w podobny sposób.

Jak wspomniano w rozdziale 2., Bluetooth był zaprojektowany jako protokół zastępujący sieć między dwoma urządzeniami. W rezultacie wymaga bliższego połączenia między urządzeniami, niż widziałeś w poprzednim projekcie XBee. W tamtym projekcie radio wysyłało sygnał bez informacji, czy odbiornik otrzymał wiadomość, i mogło wysyłać sygnał do różnych odbiorników, po prostu zmieniając adres docelowy. Natomiast odbiorniki radiowe Bluetooth muszą ustanowić połączenie przed wysyłaniem danych w danym kanale i muszą przerwać to połączenie przed rozpoczęciem konwersacji z innym radiem przez ten kanał. Zaletą technologii Bluetooth jest to, że obecnie jest ona wbudowana w wielu komercyjnych urządzeniach, więc jest to wygodny sposób połączenia projektów mikrokontrolerowych z komputerami osobistymi, telefonami i innymi urządzeniami. Wszystkie komplikacje tej technologii rekompensuje wiarygodna transmisja danych.

Moduły używane tutaj — radioodbiorniki Bluetooth Mate ze Spark Fun — używają radia z Roving Networks. Zestaw poleceń używanych w tym miejscu został zdefiniowany przez Roving Networks. Moduły Bluetooth innych producentów używają zestawów poleceń w podobnym stylu i mogą wykonywać podobne funkcje, ale ich składnia nie jest taka sama. Niestety, producenci radia Bluetooth nie ustalili standardowej składni dla swoich urządzeń.

MATERIAŁY:

- **2 płytki stykowe,**
- **1 adapter portu szeregowego TTL na USB,**
- **2 moduły Arduino,**
- **2 moduły Bluetooth Mate,**
- **2 potencjometry lub inne czujniki analogowe,**
- **2 diody LED,**
- **2 oporniki 220 Ω,**
- **2 oporniki 10 kΩ,**
- **2 przyciski.**

Krok 1. Obwody

Zamiast parowania radia Bluetooth z radiem Bluetooth komputera, jak w rozdziale 2., sparujesz dwa radia dołączone do mikrokontrolerów. Kiedy skończysz, komputer nie będzie już potrzebny.

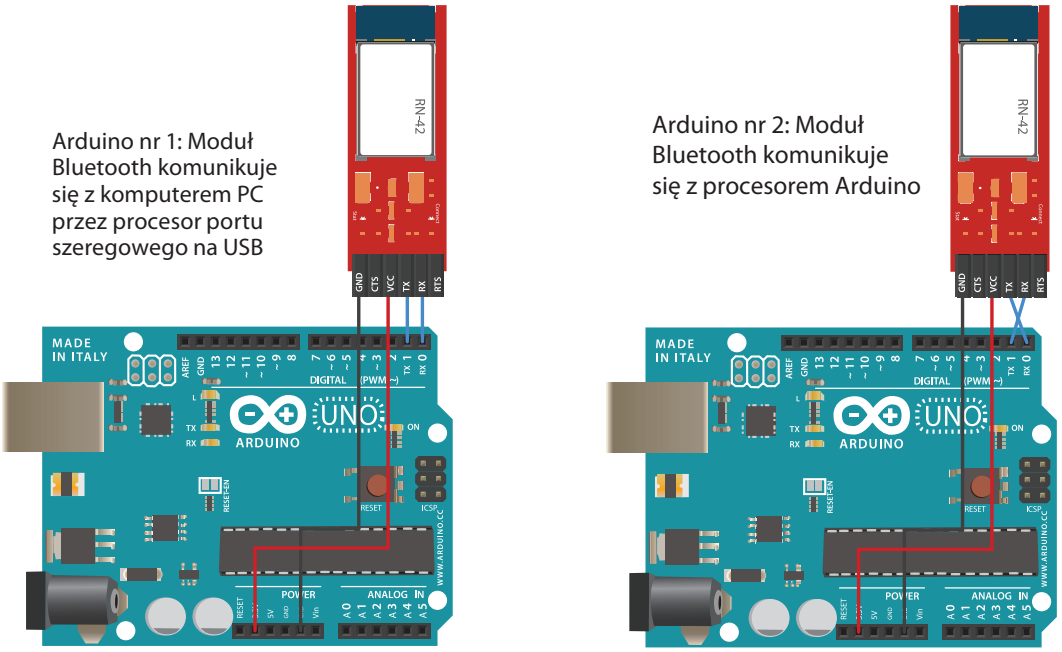
Ponieważ proces zestawienia połączenia Bluetooth obejmuje wiele kroków, najłatwiej jest dowiedzieć się i zrozumieć je przy użyciu programu terminala szeregowego, jeszcze przed napisaniem kodu. Nawet kiedy programujesz, program terminala szeregowego będzie użytecznym narzędziem diagnostycznym. Możesz użyć modułu portu szeregowego na USB pokazanego w projekcie 4., „Negocjacje w Bluetooth", w rozdziale 2., ale celem tego projektu jest uzyskanie dwóch mikrokontrolerów komunikujących się ze sobą przez Bluetooth bez pośrednictwa komputera. Zatem zamiast adaptera portu szeregowego na USB, można skonfigurować płytkę Arduino do przekazywania danych szeregowych z komputera do radia Bluetooth. Następnie, gdy będziesz gotowy do usunięcia komputera w kroku 3., po prostu zmienisz szkic i wprowadzisz niewielkie zmiany obwodu, aby usunąć komputer PC.

Aby skonfigurować Arduino do przekazywania danych z USB bezpośrednio do portu szeregowego, najpierw zaprogramuj Arduino pustym szkicem. Może do tego posłużyć szkic `BareMinimum` z *Basics examples* (podstawowe przykłady). Wygląda on tak:

```
void setup() {
}

void loop() {
}
```

Kiedy załadujesz ten szkic, mikrokontroler nie będzie robił nic, więc możesz użyć jego połączenia do procesora portu szeregowego na USB do komunikacji z radiem Bluetooth. Podłącz radio, jak pokazano po lewej na rysunku 6.20 (będziemy je nazywać *Arduino nr 1* i *radio nr 1*), następnie podłącz je do komputera i za pomocą programu terminala szeregowego otwórz połączenie szeregowe, ustawiając jego prędkość transferu na 115 200 bitów na sekundę.

Przez chwilę będziesz używał drugiego Arduino tylko do wysłania prostej wiadomości, żeby sprawdzić, czy wszystko działa. Podłącz je do drugiego Bluetooth Mate, wykorzystując obwód znajdujący się po prawej stronie na rysunku 6.20 (będziemy je nazywać *Arduino nr 2* i *radio nr 2*). Docelowo oba moduły będą połączone w ten sposób.



Rysunek 6.20
Moduł Arduino po lewej jest używany jako adapter portu szeregowego na USB dla modułu Bluetooth Mate. Zauważ, że RX jest dołączony do RX i TX do TX. To dlatego, że Bluetooth Mate faktycznie komunikuje się z komputerem za pomocą adaptera portu szeregowego na USB, a nie procesora głównego Arduino, więc połączenia szeregowe są odwrócone. Jeśli chcesz, aby moduł Bluetooth komunikował się z głównym procesorem Arduino, zamień te dwa połączenia, tak jak pokazano po prawej

Program Arduino nr 2 używa podstawowego szkicu szeregowego, jak poniżej:

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println("Witaj, Bluetooth!");
}
```

Po otwarciu połączenia Bluetooth do tego radia zobaczysz następujący komunikat, wysyłany raz po raz:

```
Witaj, Bluetooth!
Witaj, Bluetooth!
Witaj, Bluetooth!
```

Krok 2. Poznajemy polecenia

Polecenia i konfiguracja radia Bluetooth Mate korzystają z zestawu poleceń szeregowych. Konfiguracja ma dwa tryby — tryb poleceń i tryb danych, tak samo jak radia XBee. Zaraz po włączeniu zasilania Bluetooth Mate i połączeniu się z jego interfejsem szeregowym (na przykład za pomocą Arduino nr 1 z rysunku 6.20) jest ono w trybie danych. Aby sprawdzić, czy moduł działa, wpisz $$$. Odpowie:

```
CMD\r
```

Wszystkie odpowiedzi radia będą zakończone znakiem powrotu karetki, jak pokazano tutaj. Również wszystkie Twoje polecenia wejściowe powinny być zakończone znakiem powrotu karetki (naciśnij *Enter* lub *Return*).

Kiedy już jesteś w trybie poleceń, możesz uzyskać pewne podstawowe informacje na temat radia, wpisując D, podobnie jak w rozdziale 2. Otrzymasz status radia, który zawiera jego adres:

```
***Settings***

BTA=000666112233
BTName=FireFly-7256
Baudrt(SW4)=115K
Parity=None
Mode =Slav
Authen=0
Encryp=0
PinCod=1234
Bonded=0
Rem=NONE SET
```

Pierwszy wiersz, zaczynający się od BTA=, jest adresem radia w zapisie szesnastkowym. Zapisz ten adres lub skopiuj go do dokumentu tekstowego — za chwilę będziesz go potrzebować. Następnie sprawdź stan jego połączenia, wpisując GK\r. Odpowie tak:

```
0
```

Gdy radio jest podłączone, odpowie 1 zamiast 0.

Teraz chcesz zobaczyć, jakie inne radia są dostępne. Wpisz następujące polecenie (wielka litera *i*):

```
I\r
```

Radio odpowie listą radioodbiorników, tak jak to się stało w rozdziale 2.:

```
Found 2
442A60F61837,Tom Igoe...s MacBook Air,38010C
000666481ADF,RN42-1FDF,1F00
Inquiry Done
```

Widać, że każde urządzenie ma inny adres, nazwę i kod urządzenia. Możesz otrzymać listę tylko z jednym określonym kodem adresu, wpisując:

```
IN 0,001F00\r
```

Jest to przydatne, kiedy chcesz zobaczyć tylko inne Bluetooth Mate w danym obszarze. Jak być może zgadłeś, kodem urządzenia jest 001F00. Polecenie IN również eliminuje nazwy tekstowe, więc można uzyskać tylko adresy i kody urządzeń.

Teraz, kiedy masz adres swojego odbiornika radiowego Bluetooth (który powinien być widoczny na listach powyżej, jeśli jest podłączony do drugiego Arduino), możesz połączyć się z nim tak:

```
C, 000666481ADF\r
```

Zastąp 000666481ADF adresem radia podłączonego do Arduino nr 2. Radio nr 1 odpowie:

```
TRYING
```

Kiedy nawiąże prawidłowe połączenie, zielone światełko połączenia na obu Bluetooth Mate powinno się zapalić, radia automatycznie przejdą w tryb danych i powinien pojawić się następujący komunikat z Arduino nr 2:

```
Witaj, Bluetooth!
Witaj, Bluetooth!
Witaj, Bluetooth!
```

Połączenie jest oczywiście dwukierunkowe. Wszystko, co wpiszesz w oknie terminala szeregowego, będzie wysyłane do Arduino nr 2. Nie zostało ono zaprogramowane, aby odpowiedzieć, więc nie zobaczysz żadnych informacji zwrotnych od niego. Gdy będziesz chciał zamknąć połączenie, najpierw wróć do trybu poleceń, wpisując:

```
$$$\r
```

Radio potwierdzi to ponownie przez CMD, po czym należy wpisać:

```
K,
```

Połączenie zostanie przerwane, a Ty zobaczysz:

```
KILL
```

Istnieją jeszcze inne polecenia statusu, ale te są najważniejsze.

X

Krok 3. Łączenie dwóch radioodbiorników Bluetooth

Teraz, gdy znasz podstawy podłączania i rozłączania, nadszedł czas, aby pozwolić mikrokontrolerowi to zrobić. W tym kroku połączysz się za pomocą tej samej konfiguracji fizycznej, z kilkoma dodatkowymi częściami. Będziesz pracował na Arduino nr 2 zamiast Arduino nr 1. Na chwilę pozostaw go podłączony jako konwerter portu szeregowego na USB dla radia Bluetooth. Pozostaw również otwarty terminal szeregowy do niego, dzięki czemu będziesz mógł zobaczyć, co się dzieje.

Najpierw uzyskaj adresy obu radioodbiorników Bluetooth. Już zapisałeś jeden. Zastąp go drugim radiem w Twoim obwodzie konwertera portu szeregowego na USB i wykonaj te same czynności, aby uzyskać adres również tego radia. Kiedy otrzymasz te adresy, wykonaj jedną dodatkową konfigurację na obu radiach, tak jak poniżej:

```
SO,BT\r
```

To ustawia ciąg statusu na BT, więc podczas łączenia lub rozłączania radioodbiorniki wysyłają ciąg, abyś o tym wiedział, taki jak:

```
BTCONNECT\r
```

lub:

```
BTDISCONNECT\r
```

Następnie dodaj potencjometr i przycisk, jak pokazano na rysunku 6.21. Podobnie jak w przykładzie z XBee, mamy tu potencjometr podłączony do analogowego pinu, abyś mógł wysyłać jego wartości. Możesz użyć dowolnego analogowego czujnika na pinie A0. Przyciskiem będziesz nawiązywał lub przerywał połączenie między dwoma odbiornikami radiowymi. Jest to zasadnicza różnica pomiędzy XBee i odbiornikami radiowymi Bluetooth — te drugie muszą zostać sparowane, zanim będą mogły się komunikować.

UWAGA! Dobrym pomysłem jest usunięcie Bluetooth Mate podczas programowania płyty Arduino, podobnie jak robiłeś to już dla innych urządzeń szeregowych.

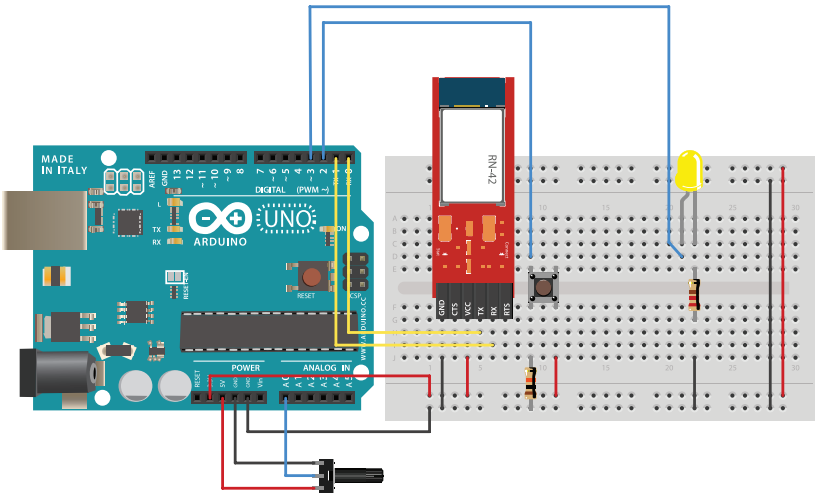
Kiedy przycisk zostanie naciśnięty po raz pierwszy, poniższy program łączy się z drugim Bluetooth Mate o ustalonym adresie. Kiedy się połączy, wysyła wartość potencjometru jako ciąg ASCII, zakończony znakiem powrotu karetki.

Podobnie jak w przykładzie z XBee, ten program również czeka na przyjście ciągów ASCII i konwertuje je, aby użyć ich jako wartości PWM do przyciemnienia diody LED na pinie 3. Będzie korzystać z biblioteki TextFinder, którą widziałeś w rozdziale 4. Jeśli nie zainstalowałeś jej jeszcze, zrób to, korzystając z instrukcji z tamtego rozdziału.

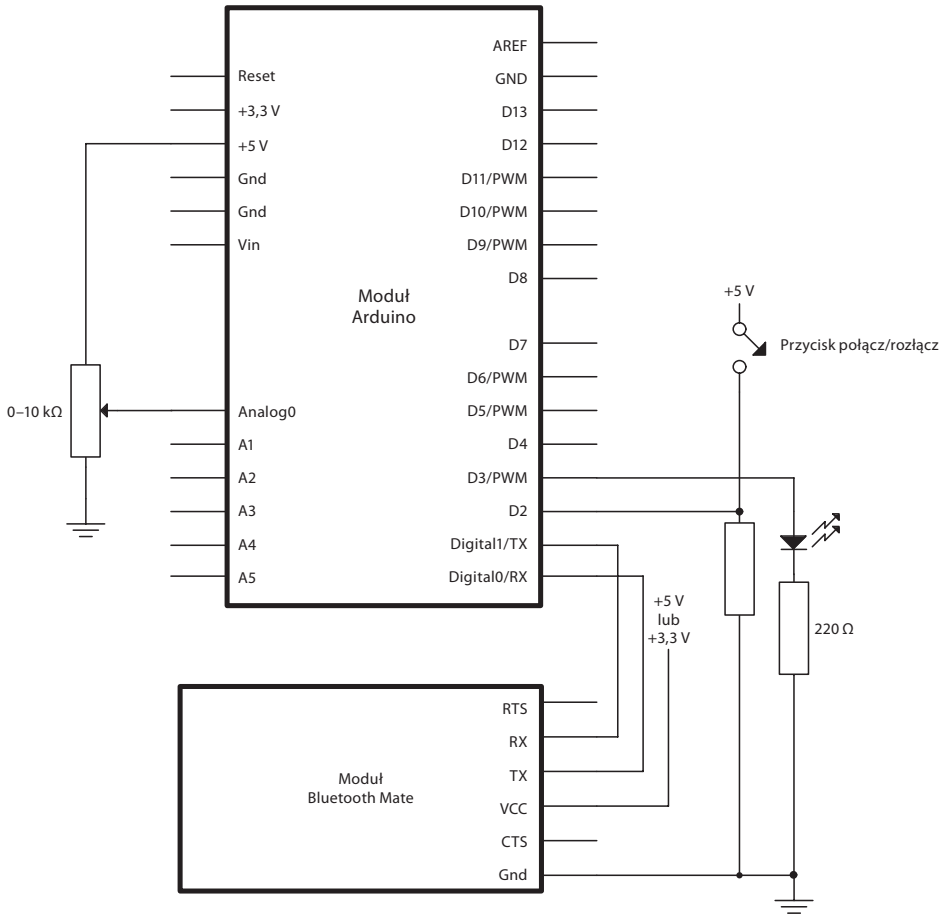
Na początek mamy stałe i zmienne dla tego programu.

```
/*
 Analogowe przesyłanie dwukierunkowe Bluetooth
 Kontekst: Arduino
*/
#include <TextFinder.h>

const int sensorPin = A0;          // czujnik wejścia analogowego
const int analogLed = 3;           // dioda LED, która zmienia jasność
const int threshold = 20;          // próg zmiany czujnika
const int debounceInterval = 15;   // używane do wygładzania odczytów przycisku
const int connectButton = 2;       // przycisk do podłączania
int lastButtonState = 0;           // poprzedni stan przycisku
int lastSensorReading = 0;         // poprzedni stan czujnika
long lastReadingTime = 0;          // czas wcześniejszego odczytu czujnika

// adres zdalnego radia BT. Zamień na adres
// swojego zdalnego radia
String remoteAddress = "112233445566";
String messageString = "";         // wiadomości przychodzące na port szeregowy

boolean connected = false;         // jesteś połączony czy nie
boolean commandMode = false;       // jesteś w trybie poleceń czy danych

TextFinder finder(Serial);         // do przeszukiwania danych z portu szeregowego
```

Zamień na adres drugiego radia.



Rysunek 6.21.
Radio Bluetooth Mate podłączone do Arduino. Obwód jest podobny do opisanego wcześniej obwodu mikrokontrolera XBee. Przycisk będzie łączył lub rozłączał dwa radia



Metoda `setup()` ustawia stany pinów, inicjuje port szeregowy i miga diodą LED, jak zwykle.

```
void setup() {
  // konfiguracja komunikacji szeregowej:
  Serial.begin(115200);

  // konfiguracja pinów wyjściowych:
  pinMode (analogLed, OUTPUT);

  // zamigaj diodą LED TX, sygnalizując, że
  // główny program zaraz się zacznie:
  blink(analogLed, 3);
}
```

Główna pętla nasłuchuje przychodzących danych szeregowych i sprawdza, czy przycisk został naciśnięty. Jeśli tak, odpowiednio: łączy lub rozłącza. Jeśli radia są połączone, odczytuje analogowe dane wejściowe i wysyła je, jeśli nastąpiła znacząca zmiana

```
void loop() {
  // przeczytaj przychodzące dane szeregowe i sparsuj je:
  handleSerial();

  // sprawdź, czy przycisk jest wciśnięty:
  boolean buttonPushed = buttonRead(connectButton);

  // jeśli przycisk został właśnie naciśnięty:
  if (buttonPushed) {
    // jeśli klient jest podłączony, rozłącz:
    if (connected) {
      BTDisconnect();
    } // jeśli klient jest rozłączony, spróbuj połączyć się:
    else {
      BTConnect();
    }
  }
  // jeśli połączony, odczytaj czujnik:
  if (connected) {
    // zapisz bieżący czas w milisekundach:
    long currentTime = millis();
    // jeżeli upłynęło wystarczająco dużo czasu od ostatniego odczytu:
    if (currentTime - lastReadingTime > debounceInterval) {
      // odczytaj czujnik analogowy, podziel przez 4, aby uzyskać zakres 0 - 255:
      int sensorValue = analogRead(A0)/4;
      //  jeśli istnieje istotna różnica między
      //  bieżącym i ostatnim odczytem czujnika, wyślij go:
      if (abs(sensorValue - lastSensorReading) > threshold) {
        Serial.println(sensorValue, DEC);
      }
      // zapisz czas ostatniego odczytu
      // i ostatni odczyt czujnika:
      lastReadingTime = currentTime;
      lastSensorReading = sensorValue;
    }
  }
}
```

» Metoda `blink()` jest taka sama jak we wcześniejszym przykładzie z XBee.

```
void blink(int thisPin, int howManyTimes) {
  for (int i=0; i< howManyTimes; i++) {
    digitalWrite(thisPin, HIGH);
    delay(200);
    digitalWrite(thisPin, LOW);
    delay(200);
  }
}
```

» Metoda `buttonRead()` również będzie znajoma — jest taka sama jak u klientów Pong w rozdziale 5.

```
// ta metoda odczytuje przycisk, aby zobaczyć, czy właśnie się zmienił
// ze stanu niskiego na wysoki, i filtruje odbicie przycisku w przypadku
// zakłóceń elektrycznych:

boolean buttonRead(int thisButton) {
  boolean result = false;
  // tymczasowy stan przycisku:
  int currentState = digitalRead(thisButton);
  // końcowy stan przycisku:
  int buttonState = lastButtonState;
  // uzyskaj bieżący czas do odliczenia czasu interwału odbijania:
  long lastDebounceTime = millis();

  while ((millis() - lastDebounceTime) < debounceInterval) {
    // odczytaj stan przełącznika do zmiennej lokalnej:
    currentState = digitalRead(thisButton);
    // jeśli przycisk zmienił się z powodu zakłóceń:
    if (currentState != buttonState) {
      // resetuj czasomierz odbijania
      lastDebounceTime = millis();
    }
    // niezależnie od odczytu, czekamy już dłużej
    // niż opóźnienie odbijania, dlatego należy przyjąć go jako rzeczywisty bieżący stan:
    buttonState = currentState;
  }
  // jeśli stan przycisku zmienił się i jest wysoki:
  if(buttonState != lastButtonState && buttonState == HIGH) {
    result = true;
  }
  // zapisz bieżący stan na następny raz:
  lastButtonState = buttonState;
  return result;
}
```

Ponieważ jest to połączenie dedykowane dla dwóch radioodbiorników, musisz śledzić jego stan. Gdy ustanawiane jest nowe połączenie, Bluetooth Mate wysyła wiadomość szeregową, zanim przejdzie w tryb danych (dzięki Twojej ostatniej zmianie konfiguracji). Wiadomość szeregowa wygląda tak:

```
BTCONNECT\r
```

Gdy połączenie jest zrywane, wysyła taką wiadomość i pozostaje w trybie poleceń:

```
BTDISCONNECT\r
```

Podczas próby połączenia Mate wysyła:

```
TRYING\r
```

Można zignorować wiadomość `TRYING`, ponieważ po niej zawsze następuje `CONNECT` lub — jeśli Mate nie uda się nawiązać połączenia:

```
CONNECT failed\r
```

Wyszukaj te ciągi w przychodzących danych i użyj ich do śledzenia stanu połączenia. Bluetooth Mate przechodzi w tryb danych natychmiast po wysłaniu jednego z tych dwóch komunikatów, więc są one również wygodnym sposobem na śledzenie trybów. Metody `BTConnect()`, `BTDisconnect()` i `handleSerial()` używają tych ciągów, aby wykonać swoje zadanie, jak pokazano poniżej.

`BTConnect()` sprawdza, czy radio jest w trybie poleceń, czy w trybie danych, a następnie próbuje nawiązać połączenie. Jeśli próba nie powiedzie się, pozostaje w trybie poleceń.

```
void BTConnect() {
  // jeśli w trybie danych, wyślij $$$
  if (!commandMode) {
    Serial.print("$$$");
    // czekaj na odpowiedź:
    if (finder.find("CMD")) {
      commandMode = true;
    }
  }
  // kiedy jesteś w trybie poleceń, wyślij polecenie połączenia:
  if (commandMode) {
    Serial.print("C," + remoteAddress + "\r");
    // czekaj na odpowiedź:
    finder.find("CONNECT");
    // jeśli wiadomość to „CONNECT failed”:
    if (finder.find("failed")) {
      connected = false;
    }
    else {
      connected = true;
      // radio automatycznie przejdzie w tryb danych,
      // kiedy się połączy:
      commandMode = false;
    }
  }
}
```

`BTDisconnect()` jest podobna do `BTConnect()`, ale działa odwrotnie. Przechodzi w tryb poleceń i wysyła komunikat rozłączenia, którego użyłeś wcześniej.

```
void BTDisconnect() {
  // jeśli w trybie danych, wyślij $$$
  if (!commandMode) {
    Serial.print("$$$");
    // czekaj na odpowiedź:
    if (finder.find("CMD")) {
      commandMode = true;
    }
  }
  // kiedy jesteś w trybie poleceń,
  // wyślij polecenie rozłącz:
  if (commandMode) {
    // próba połączenia:
    Serial.print("K,\r");
    // poczekaj na komunikat o pomyślnym rozłączeniu:
    if (finder.find("BTDISCONNECT")) {
      connected = false;
      // radio automatycznie przejdzie w tryb danych,
      // kiedy jest rozłączone:
      commandMode = false;
    }
  }
}
```

`handleSerial()` również czeka na wiadomość tekstową, ale nie używa `TextFinder`, ponieważ musi mieć możliwość odebrania jednej z trzech różnych opcji. `TextFinder` wykonuje tylko jeden przebieg przez strumień szeregowy, więc nie można sprawdzić kolejnego ciągu, jeśli nie znajdzie pierwszego ciągu.

Jeśli zostanie znaleziona prawidłowa liczba, metoda używa jej do ustawienia jasności diody LED na pinie 3.

```
void handleSerial() {
  // sprawdź ciąg wiadomości:
  // jeśli to BTCONNECT, connected = true;
  // jeśli to BTDISCONNECT, connected = false;
  // jeśli to CONNECT failed, connected = false;
  // jeśli to liczba, ustaw LED
  char inByte = Serial.read();

  // dodaj dowolne znaki alfanumeryczne ASCII
  // do ciągu wiadomości:
  if (isAscii(inByte)) {
    messageString = messageString + inByte;
  }

  // obsłuż wiadomości CONNECT i DISCONNECT:
  if (messageString == "BTDISCONNECT") {
    connected = false;
  }
  if (messageString == "BTCONNECT") {
    connected = true;
  }
  if (connected) {
    // konwertuj ciąg znaków na liczbę:
    int brightness = messageString.toInt();
    // ustaw analogowe wyjście LED:
```

Na koniec `handleSerial()` czeka na znak powrotu karetki; kiedy go znajdzie, czyści `messageString`, aby odebrać następną wiadomość.

Dokończenie z poprzedniej strony

```
    if (brightness > 0) {
      analogWrite(analogLed, brightness);
    }
  }

  // jeśli otrzymasz znak ASCII powrotu karetki:
  if (inByte == '\r') {
    // wyczyść ciąg wejściowy dla
    // następnej wartości:
    messageString = "";
  }
}
```

Oto cały program. Uruchom go na swoim mikrokontrolerze, wprowadzając adres radia w Arduino nr 2 do tablicy `remoteAddress` powyżej. Po naciśnięciu przycisku na pierwszym mikrokontrolerze, zostanie utworzone połączenie do drugiego i rozpocznie się przesyłanie wartości czujnika. W terminalu szeregowym początkowe wiadomości powinny wyglądać następująco:

```
BTCONNECT
121
132
83
```

Po nawiązaniu połączenia możesz odpowiedzieć tak, jakby były wysyłane wartości czujnika. Mikrokontroler odpowiednio przyciemni diodę LED na pinie 3. Wpisz:

```
12\r
120\r
255\r
1*
```

Dioda LED powinna zacząć się przyciemniać, rozjaśniać, potem osiągnie najjaśniejszy odcień, a następnie uzyska bardzo ciemny. W odpowiedzi powinieneś otrzymać cztery odczyty czujnika. A kiedy ponownie naciśniesz przycisk, powinieneś otrzymać:

```
BTDISCONNECT
```

Kiedy uda Ci się połączyć i rozłączyć kilka razy, jesteś gotowy na ostatni krok.

Krok 4. Łączenie dwóch mikrokontrolerów przez Bluetooth

Jeśli wcześniej zauważyłeś podobieństwa między tym przykładem a przykładem z XBee, to prawdopodobnie wiesz, co się stanie. Zbuduj ten sam obwód dla swojego drugiego mikrokontrolera, przy użyciu drugiego radia, przez dodanie przycisku i potencjometru i zamianę połączenia szeregowego TX i RX. Zmień adres Bluetooth w powyższym szkicu na adres pierwszego radia i zaprogramuj drugi mikrokontroler. Następnie zresetuj oba mikrokontrolery. Po naciśnięciu przycisku na jednym z nich, spróbuje on połączyć się z drugim i rozpocząć wymianę danych. Po naciśnięciu przycisku po raz drugi, rozłączą się.

Teraz, gdy masz już podstawy, możesz zmodyfikować kod tak, aby obsłużyć różne sytuacje z radioodbiornikami Bluetooth.

X

Zakup radia

W niniejszym rozdziale widziałeś już kilka różnych rodzajów modułów bezprzewodowych. Mimo że dobrze spełniają swoje zadanie, nie są to jedyne opcje dostępne na rynku. Zdecydowanie powinieneś rozejrzeć się za modułami, które spełniają Twoje potrzeby. Oto kilka rzeczy, które należy wziąć pod uwagę przy wyborze swojego radia.

Najmądrzejszą rzeczą w przypadku zakupu radia jest zakup całego zestawu. Dopasowywanie nadajnika od jednego producenta do odbiornika od innego to dopraszanie się o problemy. Mogą oni mówić, że działają na tym samym zakresie częstotliwości, ale nie ma żadnej gwarancji. Podobnie próby hakowania radia analogowego — takiego jak elektroniczna niania lub walkie-talkie — mogą wydawać się rozwiązaniem łatwym i tanim, ale ostatecznie będą Cię kosztować sporo czasu i energii. Podczas wyszukiwania radia poszukaj czegoś, co może pobrać szeregowe dane wyjściowe Twojego mikrokontrolera. Większość mikrokontrolerów wysyła dane szeregowe na poziomach TTL, z 0 V dla logicznego 0 i 3,3 lub 5 V dla logicznej 1. Konwertowanie danych wyjściowych do poziomów RS-232 również jest dość proste, więc radia, które mogą pobierać te sygnały, nadają się do Twoich celów.

Należy rozważyć szybkość przesyłu danych potrzebną Twojej aplikacji — a dokładniej jej części bezprzewodowej. Możesz nie potrzebować szybkiej sieci bezprzewodowej. Jednym z popularnych zastosowań komunikacji bezprzewodowej w technice filmowej jest uzyskiwanie danych z ciał artystów w celu sterowania scenicznymi urządzeniami MIDI, takimi jak samplery i regulatory oświetlenia. Mógłbyś pomyśleć, że do tego celu Twoje radio musi pracować z prędkością danych MIDI, ale nie ma takiej potrzeby. Możesz wysłać dane czujnika od wykonawcy bezprzewodowo, z niską prędkością transmisji danych, do stacjonarnego mikrokontrolera, następnie mikrokontroler wyśle dane za pośrednictwem MIDI z wyższą szybkością transferu.

Należy rozważyć protokoły urządzeń, które już masz do dyspozycji. Na przykład, jeśli konstruujesz obiekt, który ma komunikować się z telefonem komórkowym lub komputerem typu laptop, i istnieje tylko jeden obiekt zaangażowany, rozważ Bluetooth. Większość laptopów i wiele telefonów komórkowych ma już wbudowane radia Bluetooth, więc będziesz potrzebował tylko jedno radio, aby wykonać to zadanie. Sprawienie, by Twój obiekt był zgodny z poleceniami specyficznymi dla istniejących urządzeń, może wymagać trochę pracy, ale jeśli możesz skupić się na tym zamiast na uzyskaniu spójnej transmisji radiowej, to i tak zaoszczędzisz mnóstwo czasu.
X

A co z Wi-Fi?

Do tej pory widziałeś w działaniu najbardziej podstawowe radia szeregowe w projekcie nadajnik-odbiornik i bardziej zaawansowane radia pracujące w projektach z urządzeniami nadawczo-odbiorczymi. Jeśli myślisz o sieci mikrokontrolerów, prawdopodobnie zastanawiasz się, czy możesz podłączyć swoje projekty do Internetu i połączyć je ze sobą nawzajem za pomocą Wi-Fi. Możesz, ale należy rozważyć pewne komplikacje.

Do niedawna Wi-Fi nie było zbyt popularne w projektach mikrokontrolerów ze względu na koszty i trudności związane z zasilaniem. Dostępne na rynku moduły konwersji mikrokontrolera do Wi-Fi są droższe niż równoważne urządzenia nadawczo-odbiorcze implementujące inne protokoły. Na szczęście to zaczyna się zmieniać. Wiele starszych modułów Wi-Fi zachłannie konsumowało zasilanie, ale to także się zmienia.

Obecnie dostępnych jest kilka rozwiązań Wi-Fi na rynku. Spark Fun produkuje tarczę WiFly, a Digi właśnie ogłosiło wersje Wi-Fi swoich modułów XBee. Nadchodzi również tarcza Arduino Wi-Fi, w założeniu łatwa do podłączenia do istniejącego projektu Ethernet (wymaga niewielkich zmian w kodzie). Poniżej znajdziesz krótkie wprowadzenie do tych zagadnień.
X

Projekt 12.

Witaj, Wi-Fi!

Pamiętasz serwer koloru światła dziennego, który zbudowałeś w projekcie 6.? W tym projekcie przebudujesz go za pomocą tarczy Arduino Wi-Fi. Zobaczysz, że większość kodu i obwód są dokładnie takie same. Zmienia się tylko warstwa fizyczna komunikacji.

MATERIAŁY:

- **1 tarcza Arduino Wi-Fi,**
- **1 moduł mikrokontrolera Arduino,**
- **1 połączenie Wi-Fi Ethernet z Internetem,**
- **3 oporniki 10 kΩ,**
- **3 fotokomórki (oporniki fotoelektryczne),**
- **1 płytka stykowa,**
- **3 filtry oświetlenia.**

Zestawianie połączenia

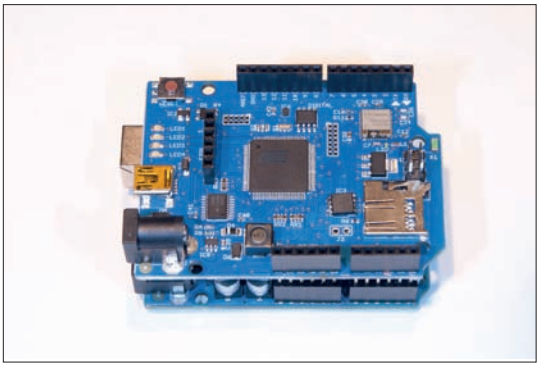
Tarcza Wi-Fi komunikuje się z Arduino za pośrednictwem SPI podobnie jak tarcza Ethernet, więc zbuduj obwód tak jak przedstawiono na rysunku 4.4, ale zastąp tarczę Ethernet tarczą Wi-Fi.

UWAGA! Tarcza Wi-Fi nie może pracować z tarczą Arduino Ethernet lub Ethernet, ponieważ wszystkie trzy używają tych samych pinów wyboru SPI.

Aby utworzyć połączenie sieciowe, musisz znać nazwę sieci Wi-Fi, z którą się łączysz (nazywaną również SSID), i wiedzieć, jakiego typu zabezpieczeń używa. To ta sama informacja, której używasz, łącząc inne urządzenia bezprzewodowe do rutera Wi-Fi. Tarcza Wi-Fi może połączyć się z otwartymi sieciami lub sieciami zabezpieczonymi WEP (40-bitowymi i 128-bitowymi) szyfrowaniem WPA lub WPA2. Do WPA i WPA2 będzie potrzebne hasło. Do WEP potrzebujesz klucza i indeksu klucza. Klucz WEP jest długim ciągiem cyfr szesnastkowych używanym jako hasło. 40-bitowe klucze WEP mają 10 znaków ASCII, a klucz WEP 128-bitowy ma 26 znaków. Rutery WEP mogą przechowywać do czterech kluczy, więc indeks klucza wskazuje, którego używasz. Przez większość czasu będziesz używał indeksu klucza 0. Poniżej znajdują się przykłady typowych kombinacji WEP i WPA.

```
WPA network name: noodleNet
WPA password: m30ws3rs!

WEP network name: sandbox
WEP key index: 0
WEP 40-bit key: 1234567890

WEP network name: sandbox
WEP key index: 0
WEP 128-bit key: 1A2B3C4D5E6FDADADEEDFACE10
```

Jeśli korzystasz z domowego rutera, są szanse, że Ty lub ktoś z Twojej rodziny skonfigurowaliście ruter bezprzewodowy tak, że znasz te informacje. Jeśli łączysz się z ruterem w szkole lub instytucji, zapytaj administratora sieci o te szczegóły. Po uzyskaniu tych informacji jesteś gotowy do rozpoczęcia programowania.



Rysunek 6.22.
Tarcza Arduino Wi-Fi

Tarcza Arduino Wi-Fi, pokazana na rysunku 6.22, to nowy produkt, więc jej interfejs programowania może ulec zmianom w trakcie rozwoju oprogramowania. Najnowsze aktualizacje dla tarczy Wi-Fi, biblioteki Wi-Fi i przykłady jej zastosowania znajdziesz w sekcji *Hardware* (sprzęt) na stronie *http://arduino.cc*.

Skonfiguruj to (WPA)

Pierwszą rzeczą, którą powinieneś zrobić, jest uzyskanie informacji o sieci (jak opisano powyżej) i wprowadzenie ich do zmiennych, żebyś mógł nawiązać połączenie. Będziesz nadal potrzebował serwera, jak również zmiennej globalnej `lineLength` z poprzedniego szkicu serwera RGB.

```
Skonfiguruj to (WPA)
/*
  Serwer WWW RGB Wi-Fi
  Kontekst: Arduino
*/

#include <SPI.h>
#include <WiFi.h>

char ssid[] = "mojaSieć";          // nazwa Twojej sieci
char password[] = "tajneHasło";    // hasło, którego używasz do połączenia
int status = WL_IDLE_STATUS;       // status radia Wi-Fi

Server server(80);
int lineLength = 0;                // długość przychodzącego wiersza tekstu
```

» Zmień tak, żeby odpowiadało to Twojej sieci.

Skonfiguruj to (WEP)

Jeżeli używasz szyfrowania WPA, zmienne konfiguracyjne będą bardziej przypominały poniższy kod (zmiany zostały wyróżnione).

```
char ssid[] = "mojaSieć";              // nazwa Twojej sieci
char keyIndex = 0;                     // sieci WEP mogą mieć wiele kluczy
// 128-bitowy klucz WEP, używany do połączenia:
char key[] = "FACEDEEDDADA01234567890ABC";
int status = WL_IDLE_STATUS;           // status radia Wi-Fi
```

» Zmień tak, aby odpowiadało to Twojej sieci. Jeśli ruter ma wartości dla więcej niż jednego indeksu klucza, użyj indeksu klucza 0.

» Metoda `setup()` wygląda inaczej niż w poprzednim szkicu, ponieważ ustawiasz połączenia Wi-Fi zamiast sieci przewodowej Ethernet. Jeśli uzyskasz połączenie, wypisz nazwę sieci. Jeśli nie, zatrzymaj program właśnie tutaj. Tarcza Wi-Fi domyślnie korzysta z protokołu DHCP, więc nie ma potrzeby zmieniać czegokolwiek w kodzie, aby uzyskać adres za pośrednictwem DHCP.

Reszta szkicu jest identyczna ze szkicem w projekcie 6. Stamtąd można skopiować pozostałą część kodu. Ponieważ Wi-Fi jest również Ethernetem, interfejsy biblioteki klienta i serwera są takie same, więc można łatwo przełączać się między Ethernetem i Wi-Fi.

```
void setup() {
  // inicjuj port szeregowy:
  Serial.begin(9600);

  Serial.println("Próba połączenia z siecią...");
  // próba połączenia za pomocą szyfrowania WPA:
  status = WiFi.begin(ssid, password);

  // lub użyj tego do próby połączenia za pomocą 128-bitowego szyfrowania WEP:
  // status = WiFi.begin(ssid, keyIndex, key);

  Serial.print("SSID: ");
  Serial.println(ssid);

  // jeśli nie udało się połączyć, zatrzymaj się tutaj:
  if ( status != WL_CONNECTED) {
    Serial.println("Nie można nawiązać połączenia Wi-Fi");
    while(true);
  }
}
```

Diagnostyka Wi-Fi

Tarcza Wi-Fi daje Ci możliwość robienia projektów w sieci bezprzewodowej, a ponadto udostępnia także kilka innych użytecznych funkcji do diagnozowania poprawności połączenia. Przed rozpoczęciem projektu Wi-Fi uruchom kilka diagnostyk, aby upewnić się, że zestawiłeś niezawodne połączenie. Podobnie jak wszelkie połączenia bezprzewodowe, jest ono niewidoczne i łatwo przeoczyć coś podczas rozwiązywania problemu, kiedy problemem jest po prostu to, że nie masz połączenia.

Niezależnie od tego, z jakim modułem Wi-Fi pracujesz, prawdopodobnie pojawią się jakieś problemy przy próbie połączenia Twojego mikrokontrolera. Poniżej wymieniłem kilka najczęściej spotykanych.

Jeśli pracujesz w szkole lub firmie, gdzie nie masz kontroli nad ruterami Wi-Fi, upewnij się z wyprzedzeniem, że masz wszystkie informacje o konfiguracji i że odpowiada to temu, co może zrobić Twój moduł. Protokoły typu Enterprise, takie jak WPA2 Enterprise, nie są dostępne dla modułów opartych na mikrokontrolerach.

Niektóre sieci publiczne korzystają z **portali utrzymywanych** (ang. *captive portal*), które są otwartymi sieciami Wi-Fi, ale musisz zalogować się za pośrednictwem strony WWW, zanim uzyskasz pełny dostęp do Internetu. Jest to utrudnienie, ponieważ trzeba zaprogramować mikrokontroler (a nie moduł Wi-Fi), aby najpierw nawiązywał połączenie HTTP z portalem utrzymywanym z informacjami logowania. Lepiej użyć sieci, która nie używa portalu utrzymywanego.

Niektóre moduły Wi-Fi, takie jak tarcza Arduino Wi-Fi, nie widzą sieci, której SSID jest ukryte. Jeśli jest to możliwe, upewnij się, że Twoja sieć jest widoczna publicznie — nawet jeśli zabezpieczenie jest włączone.

Ponieważ nie masz interfejsu, przez który mógłbyś otrzymać informację zwrotną o błędach, porównaj konfigurację z laptopem, telefonem komórkowym lub innym urządzeniem, z którym możesz połączyć się prawidłowo.

Poniżej przedstawiono dwa małe wycinki kodu diagnostycznego dla tarczy Arduino Wi-Fi, które również mogą być przydatne: umożliwiają skanowanie w poszukiwaniu dostępnych sieci i określanie siły sygnału sieci, do której jest podłączony mikrokontroler.

Oto sposób skanowania w poszukiwaniu dostępnych sieci:

```
// skanuj w poszukiwaniu pobliskich sieci:
byte numSsid = WiFi.scanNetworks();

// wyświetl listę widocznych sieci:
Serial.print("Lista SSID:");
Serial.println(numSsid);
// wyświetl numer sieci i nazwę
// dla każdej wykrytej sieci:
for (int thisNet = 0; thisNet<numSsid; thisNet++) {
  Serial.print(thisNet);
  Serial.print(") Sieć: ");
  Serial.println(WiFi.SSID(thisNet));
}
```

Oto kod określający siłę sygnału, kiedy jesteś podłączony do sieci:

```
// wyświetl poziom siły odebranego sygnału:
long rssi = WiFi.RSSI();
Serial.print("RSSI:");
Serial.println(rssi);
```

Obie techniki są przydatnymi narzędziami diagnostycznymi. Zależnie od Twojego środowiska i konfiguracji sieci, Twoja sieć bezprzewodowa może nie być stabilna przez cały czas. W związku z tym być może zechcesz ująć całość głównej `loop()` w instrukcji `if`, która sprawdza, czy nadal masz połączenie z siecią:

```
void loop() {
  if ( status == WL_CONNECTED) {
    // tutaj zrób wszystko
  } else {
    // niech użytkownik wie, że nie ma połączenia
  }
}
```

X

Podsumowanie

Komunikacja bezprzewodowa pociąga za sobą pewne znaczące różnice w porównaniu z komunikacją przewodową. Z powodu komplikacji nie można liczyć na przekazanie wiadomości w taki sposób jak w przypadku połączenia przewodowego, więc należy określić, do czego chcesz zastosować komunikację bezprzewodową.

Jeśli decydujesz się na najmniej kosztowne rozwiązania, możesz zaimplementować jednokierunkowe łącze bezprzewodowe przy użyciu pary nadajnik-odbiornik i wysyłać wiadomość ponownie, mając nadzieję, że ostatecznie zostanie otrzymana. Jeśli wydasz trochę więcej pieniędzy, możesz zaimplementować połączenie dwustronne tak, że każda strona może odpytać drugą i potwierdzić odpowiedź. Odkąd różnica w kosztach jest minimalna i zmniejsza się dostępność pary nadajnik-odbiornik, połączenie dwustronne stało się normą. Niezależnie od tego, którą metodę wybierzesz, musisz przygotować się na nieuniknione zakłócenia, jakie pojawią się przy połączeniu bezprzewodowym. Jeżeli używasz podczerwieni, światła żarowe i ciepło będą źródłem szumu; jeśli używasz radia, wszystkie rodzaje źródeł elektromagnetycznych będą źródłem szumu — od kuchenek mikrofalowych, przez prądnice, aż po telefony bezprzewodowe. Możesz napisać własne procedury korekcji błędów, ale w coraz większym stopniu bezprzewodowe protokoły, takie jak Bluetooth i ZigBee, pozwalają Ci o tym zapomnieć, ponieważ moduły implementujące te protokoły zawierają własne korekcje błędów.

Tak jak rozpocząłeś naukę o sieci, pracując z najprostszymi sieciami typu jeden do jednego w rozdziale 2., w niniejszym rozdziale rozpocząłeś pracę z połączeniami bezprzewodowymi, obserwując proste pary nadajnik-odbiornik. W następnym rozdziale przyjrzysz się sieciom równorzędnym (ang. *peer-to-peer network*), w których nie istnieje centralny kontroler i każdy obiekt w sieci może komunikować się z każdym innym obiektem. Zobaczysz przykłady realizowane zarówno w sieci Ethernet, jak i w sieciach bezprzewodowych.
X

»» „Sonar miejski" stworzony przez Kate Hartman, Kati London i Sai Sriskandarajaha
Kurtka zawiera cztery czujniki ultradźwiękowe i dwa czujniki tętna. Mikrokontroler w kurtce komunikuje się za pośrednictwem Bluetooth z telefonem komórkowym. Rejestrowana przez czujniki przestrzeń osobista wokół użytkownika oraz zmiany rytmu pracy serca, będące wynikiem zmian w przestrzeni osobistej, odwzorowują specyficzny portret użytkownika, który jest wysyłany przez telefon do wizualizacji w Internecie.

Skorowidz

D

E

F

G

H

I

N

O

P

Q

R

S

Ś

T

U

V

W

X

Z

Zacznij eksperymentować i spraw, by rzeczy robiły to, czego chcesz!

O'REILLY®

Z odrobiną wiedzy o elektronice, niedrogimi zestawami mikrokontrolerów i modułami sieciowymi pozwalającymi im komunikować się ze sobą możesz od razu zacząć budowę własnych projektów.

Ciężko nam w to uwierzyć, ale jeszcze całkiem niedawno komputery były odizolowanymi jednostkami, niezdolnymi do komunikowania się między sobą. Dzisiaj potencjał samych komputerów, tabletów i telefonów połączonych w sieć jest oszałamiający. Budowanie projektów elektronicznych, które prowadzą interakcję ze światem fizycznym, to dobra zabawa. A kiedy urządzenia, które budujesz, zaczynają komunikować się między sobą, staje się to naprawdę interesujące. Trzydzieści trzy łatwe projekty z tej książki pokazują, jak sprawić, by Twoje gadżety komunikowały się z Tobą i Twoim środowiskiem. To idealna propozycja dla ludzi z niewielką wiedzą techniczną, ale dużym zainteresowaniem tematem!

Dzięki tej książce dowiesz się, jakie urządzenia i narzędzia będą Ci potrzebne, przygotujesz stanowisko pracy i rozpoczniesz tę niesłychaną przygodę! Na początek zbudujesz najprostszą sieć i prześlesz pierwsze komunikaty (także bezprzewodowo). W kolejnych rozdziałach zaczniesz konstruować coraz bardziej zaawansowane układy, poznasz szczegóły komunikacji bezprzewodowej, identyfikacji oraz lokalizacji. Twoją ciekawość powinien wzbudzić rozdział poświęcony umieszczaniu w sieci mikrokontrolerów. Ta możliwość daje Ci do ręki potężne narzędzie. Czy już wiesz, jak je wykorzystać?

- **Blink** — Twój pierwszy program
- **Monski Pong** — sterowanie grą wideo za pomocą puszystej różowej małpki
- **Sieciowy miernik stanu zanieczyszczenia powietrza** — ładowanie i wyświetlanie najświeższego raportu dla Twojego miasta
- **Czujnik toksyn XBee** — używanie ZigBee, czujników i małpki z talerzami do ostrzegania o toksycznych wyziewach
- **Bluetooth GPS** — budowanie zasilanego z baterii GPS-a, który podaje swoją lokalizację przez Bluetooth
- **Tweetowanie z RFID** — odczytywanie strumieni Twittera przez pomachanie czytnikiem RFID

Nr katalogowy: 13295

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

helion.pl
księgarnia internetowa

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Cena: 79,00 zł
ISBN 978-83-246-5012-5

9 788324 650125

Informatyka w najlepszym wydaniu