Baza programisty

Mike Riley

Inteligentny dom

Automatyzacja mieszkania za pomocą platformy Arduino, systemu Android i zwykłego komputera

Helion

Tytuł oryginału: Programming Your Home: Automate with Arduino, Android, and Your Computer

Tłumaczenie: Mikołaj Szczepaniak

ISBN: 978-83-246-5675-2

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/intdom.zip

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

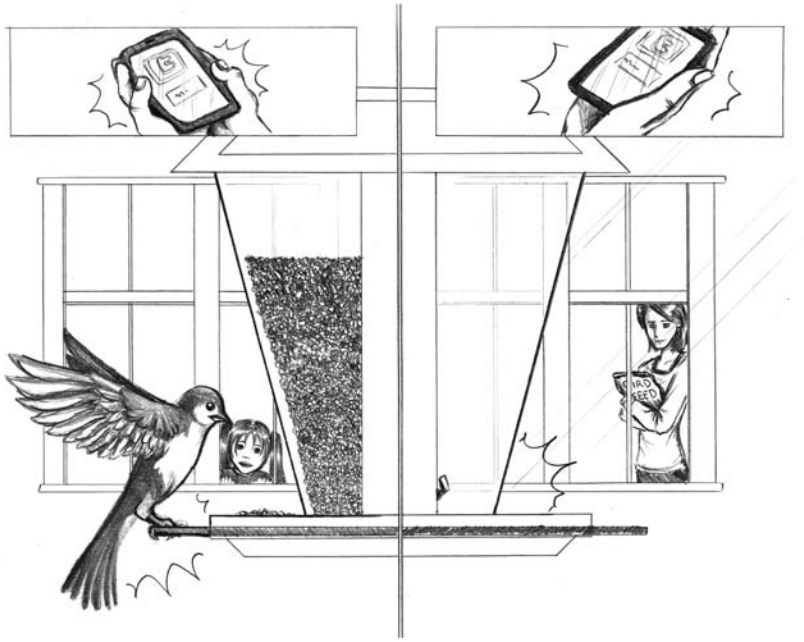
Printed in Poland.

Spis treści

Rozdział 5.
Ćwierkający karmnik dla ptaków

Mam dwójkę dzieci, które wprost uwielbiają ptaki. Już jako małe dzieci miały swoje pierwsze papużki. Lubiły też obserwować dzikie ptaki konstruujące gniazda i posilające się w karmniku zamontowanym przy oknie sypialni. Dokarmianie ptaków wiąże się jednak z pewnym problemem — dzieci ciągle zapominają o uzupełnianiu nasion dla ptaków w karmnikach. Zdarzało się, że z rozmaitych powodów w karmniku brakowało nasion przez całe dnie, a nawet tygodnie. Czyż nie byłoby prościej, gdyby sam karmnik informował nas o konieczności uzupełnienia karmy?

Historia pustego karmnika była inspiracją dla tego projektu. Czy można sobie wyobrazić lepszą formę powiadamiania o konieczności dosypania nasion niż „ćwierkanie" na Twitterze? Zainteresowani znajomi i krewni mogą śledzić konto karmnika, aby wiedzieć, kiedy karmnik jest odwiedzany przez ptaki, kiedy wymaga uzupełnienia karmy i czy nasiona zostały dosypane. (Patrz rysunek 5.1 zatytułowany „Wysyłanie powiadomień przez karmnik dla ptaków za pośrednictwem Twittera").



Rysunek 5.1. Wysyłanie powiadomień przez karmnik dla ptaków za pośrednictwem Twittera (w czasie, gdy karmnik jest odwiedzany przez ptaki, oraz w sytuacji, gdy wymaga uzupełnienia nasion)

Skoro planujemy śledzić na Twitterze komunikaty o konieczności uzupełnienia karmnika, warto rozbudować ten mechanizm o własny czujnik instalowany na grzędzie, który będzie rejestrował wizyty ptaków w karmniku i analizował czas ich przebywania na grzędzie. Przed publikacją wpisów na Twitterze będziemy rejestrować te zdarzenia w bazie danych, aby umożliwić wizualne prezentowanie wzorców karmienia ptaków w czasie.

Czy w kwietniu ptaki są bardziej wygłodniałe niż na przykład w lipcu? Czy ptaki częściej zaglądają do karmnika rano, czy popołudniami? Ile wynosi średni czas przebywania ptaków na grzędzie w karmniku? Ile czasu mija pomiędzy kolejnymi wizytami ptaków? Jak często musimy uzupełniać nasiona w karmniku? Ćwierkający karmnik dla ptaków umożliwia nam prowadzenie ciekawych badań w poszukiwaniu odpowiedzi na te i inne wzorce zachowań ptaków. Czas wzbić się w powietrze!

5.1. Czego potrzebujemy

Ponieważ ćwierkający karmnik będzie naszym pierwszym projektem realizowanym na zewnątrz, koszty niezbędnego sprzętu z kilku powodów będą nieco wyższe. Po pierwsze, jeśli nie chcemy wiercić dziur przez całą grubość ścian zewnętrznych domu ani prowadzić przewodów sieciowych przez otwarte drzwi czy okna, musimy znaleźć sposób wysyłania zdarzeń notowanych przez czujniki przy użyciu technik bezprzewodowych. Na szczęście istnieją niskonapięciowe i stosunkowo tanie urządzenia w formie nadajników radiowych XBee. Początkowa konfiguracja tych urządzeń wymaga dodatkowych nakładów i większego wysiłku, jednak wspomniane urządzenia są dość niezawodne, całkiem łatwo nawiązują połączenia, a raz zainstalowane nie wymagają naszej uwagi.

Po drugie, mimo że moglibyśmy zastosować standardową platformę Arduino Uno (tak jak na schematach połączeń prezentowanych w tym rozdziale), wymiary tej płytki mogą się okazać zbyt duże dla typowego karmnika dla ptaków. W tej sytuacji zachęcam do wydania minimalnie większej kwoty na zakup płytki Arduino Nano. Instalacja platformy Nano z pewnością będzie prostsza, zważywszy na ograniczoną przestrzeń w karmniku. Niewątpliwą zaletą płytki Nano jest niemal identyczna konfiguracja wtyków i układ elementów sprzętowych jak w przypadku większego brata. Płytka Nano oferuje wszystkie możliwości tradycyjnej platformy Arduino, tyle że zajmuje dużo mniej miejsca.

Po trzecie, mimo że zasilanie tych części elektronicznych za pośrednictwem długiego przewodu podłączonego do zewnętrznego gniazdka (zainstalowanego z myślą o dekoracjach świątecznych) jest możliwe, tak zaprojektowany system nie będzie wystarczająco autonomiczny. Co więcej, system karmnika dla ptaków to wprost doskonała okazja do zastosowania ekologicznego źródła energii.

I wreszcie w związku z koniecznością ochrony elektroniki musimy zadbać o dobre zabezpieczenie systemu przed niesprzyjającymi warunkami atmosferycznymi. Oto kompletna lista zakupów (komponenty używane w tym projekcie pokazano też na rysunku 5.2 zatytułowanym „Części systemu ćwierkającego karmnika dla ptaków”):

1. Platforma Arduino Uno lub Arduino Nano[1].

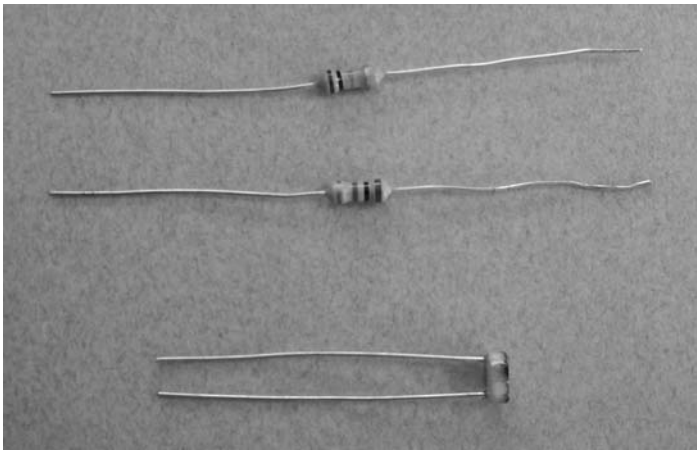
[1] *http://www.makershed.com/ProductDetails.asp?ProductCode=MKGR1*

Rysunek 5.2. Części systemu ćwierkającego karmnika dla ptaków

2. Dwa moduły XBee z zestawami adapterów oraz przewód FTDI[2].
3. Fotokomórka.
4. Kawałek folii aluminiowej.
5. Przewód.
6. Niewielki panel słoneczny z wbudowanym akumulatorem i przewodem USB (na przykład podobny do tego oferowanego w sklepie Solio)[3].
7. Jeden rezystor 10 kΩ i jeden rezystor 10 MΩ — warto sprawdzić kolorowe paski na stosowanych rezystorach: rezystor 10 kΩ powinien być oznaczony paskami brązowym, czarnym, pomarańczowym i złotym, natomiast rezystor 10 MΩ powinien być oznaczony paskami brązowym, czarnym, niebieskim i złotym. Potrzebne rezystory pokazano na rysunku 5.3 zatytułowanym „Rezystory używane w projekcie ćwierkającego karmnika dla ptaków". Na zdjęciu widać również fotokomórkę (nazywaną także fotorezystorem CdS).
8. Karmnik dla ptaków z otworem na nasiona, w którym zmieści się platforma Nano i moduł XBee (zabezpieczone przed czynnikami pogodowymi).

[2] *http://www.adafruit.com*

[3] *http://www.solio.com/chargers/*



Rysunek 5.3. Rezystory używane w projekcie ćwierkającego karmnika dla ptaków

9. Komputer (nie ma go na zdjęciu), najlepiej z systemem Linux lub Mac OS oraz zainstalowanym kompilatorem języka Python 2.6 lub nowszym (na potrzeby skryptu przetwarzającego komunikaty wysyłane przez karmnik dla ptaków).

W razie decyzji o użyciu płytki Arduino Nano zamiast tradycyjnej płytki Arduino Uno trzeba będzie dodatkowo zakupić standardowy przewód USB A-mini B (nie ma go na zdjęciu), aby połączyć płytkę Arduino Nano z komputerem. Co więcej, ponieważ na platformie Arduino Nano zastosowano wtyki męskie zamiast wtyków żeńskich (stosowanych w przypadku płytki Arduino Uno), zamiast standardowych przewodów będziemy potrzebowali odpowiednich końcówek żeńskich (których także nie pokazano na zdjęciu). Odpowiednie końcówki ułatwią łączenie przewodów z wtykami na płytce Nano bez konieczności ich trwałego lutowania.

Projekt jest bardziej złożony niż system powiadamiania o poziomie wody, a najtrudniejszym krokiem procesu budowy tego rozwiązania będzie zapewnienie niezawodnej pracy modułów XBee. Mimo to projekt jest wart niezbędnych nakładów — w jego wyniku będziemy dysponowali nie tylko supernowoczesnym karmnikiem na miarę XXI wieku, ale też skonfigurowanym systemem obejmującym moduły komunikacji radiowej XBee, który będzie nam potrzebny w wielu pozostałych projektach. Gotowy zakasać rękawy? Do dzieła!

5.2. Budowa rozwiązania

Połączenie wszystkich elementów tak, aby pasowały do wnętrza karmnika, może wymagać pewnej pomysłowości, szczególnie jeśli zbiornik na ziarno nie oferuje dostatecznie dużo przestrzeni. Zanim przystąpimy do upychania elektroniki w karmniku, musimy upewnić się, że wszystkie komponenty działają zgodnie z naszymi oczekiwaniami.

1. Zaczniemy od najprostszego kroku, czyli połączenia z płytką Arduino czujnika pojemnościowego z folii aluminiowej oraz napisania funkcji, która w momencie zmiany stanu tego czujnika będzie wysyłała komunikat do okna monitorowania portu szeregowego (a docelowo do modułu XBee podłączonego do tego portu).
2. W dalszej kolejności musimy podłączyć do płytki Arduino fotokomórkę i napisać kod reagujący na zmiany stanu tego czujnika.
3. Zaraz potem przystąpimy do łączenia pary modułów radiowych XBee, tak aby informacje o wspomnianych zdarzeniach były przekazywane pomiędzy nadajnikiem XBee połączonym z płytką Arduino a odbiornikiem XBee połączonym z komputerem za pośrednictwem przewodu FTDI USB.
4. I wreszcie musimy napisać skrypt języka Python, który pobierze dane z bazy danych SQLite, sformatuje je i wyśle w formie gotowego wpisu do publikacji w serwisie Twitter.

Po dopracowaniu i połączeniu wszystkich komponentów będziemy dysponowali systemem złożonym z płytki Arduino (najlepiej w wersji Nano), modułu XBee, czujnika grzędy i fotokomórki — całość będzie zabezpieczona przed czynnikami atmosferycznymi i zainstalowana w karmniku dla ptaków. Po sprawdzeniu, czy wszystko działa prawidłowo, należy wyjść na dwór i przetestować ten system w warunkach polowych.

5.3. Czujnik grzędy

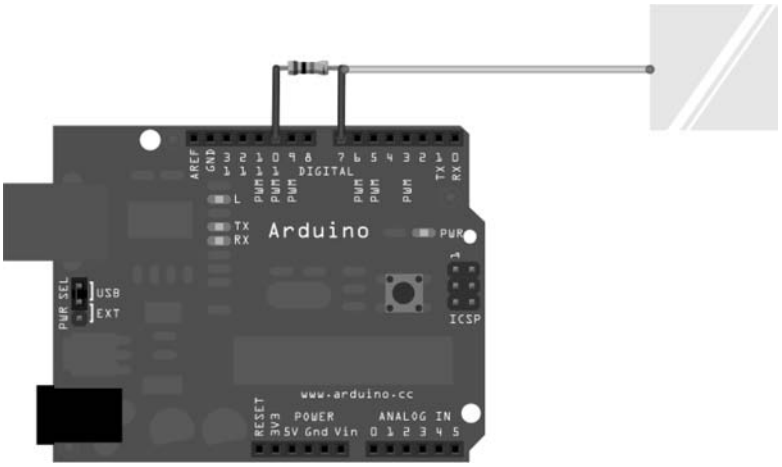
Karmniki dla ptaków maja różne kształty i wymiary. Zdecydowałem się zastosować wyjątkowo proste rozwiązanie w celu wykrywania zdarzeń lądowania ptaków na grzędzie karmnika. Mimo że skonstruowanie mechanizmu wykrywającego nacisk na grzędzie z pewnością byłoby możliwe, czas potrzebny na opracowanie tego rozwiązania i jego koszty byłyby stosunkowo duże, a przecież naszym jedynym celem jest wykrycie, czy coś nie usiadło na grzędzie.

Alternatywnym rozwiązaniem jest monitorowanie zmian pojemności elektrycznej.

Wystarczy owinąć grzędę karmnika folią aluminiową oraz połączyć tę folię z cyfrowymi wtykami na płytce Arduino przy użyciu rezystora. Na podstawie wartości bazowych i zmian wykrywanych przez ten czujnik w momencie lądowania ptaka możemy wyznaczyć wartość progową, której przekroczenie powinno powodować wygenerowanie i wysłanie komunikatu o lądującym ptaku.

Budowa czujnika

Budowa i testowanie czujnika grzędy to najprostszy element tego projektu. Wystarczy użyć kawałka folii aluminiowej wielkości połowy opakowania od listka gumy do żucia i owinąć grzędę. Należy następnie połączyć jeden koniec rezystora 10 MΩ z wtykiem cyfrowym nr 7 na płytce Arduino oraz drugi koniec z wtykiem nr 10. Przewód połączony z folią aluminiową należy połączyć z końcówką rezystora podłączoną do wtyku cyfrowego nr 7. Odpowiedni schemat połączeń pokazano na rysunku 5.4 zatytułowanym „Sposób podłączenia czujnika pojemnościowego".



Rysunek 5.4. Sposób podłączenia czujnika pojemnościowego

Programowanie czujnika

Należy teraz połączyć platformę Arduino z komputerem, po czym uruchomić środowisko Arduino IDE w celu napisania kodu obsługującego czujnik.

Podobnie jak w przypadku projektu systemu powiadamiania o poziomie wody musimy napisać kod realizujący następujące zadania:

1. Wyświetli wartości odczytane przez czujnik pojemnościowy w oknie monitora portu szeregowego środowiska Arduino IDE.
2. Zidentyfikuje wartość bazową tego czujnika.
3. Dostosuje natężenie prądu w momencie dotykania czujnika palcem.
4. Zarejestruje nową wartość używaną w roli wartości progowej generującej powiadomienie.

Aby ułatwić sobie wykrywanie zmian natężenia prądu elektrycznego w momencie dotknięcia folii palcem lub wylądowania ptaka, skorzystamy z rozwiązania opracowanego przez jednego z miłośników platformy Arduino — Paula Badgera. Paul napisał bibliotekę Arduino, dzięki której mierzenie zmian wartości przekazywanych przez czujniki pojemnościowe (podobnych do folii używanej w tym projekcie) jest dziecinnie proste. Biblioteka nazwana Capacitive Sensing[4] umożliwia programistom platformy Arduino przekształcenie dwóch (lub większej liczby) wtyków na płytce Arduino w czujnik pojemnościowy, który może służyć do wykrywania pojemności elektrycznej ludzkiego ciała. Ciało człowieka cechuje się znacznie większą pojemnością niż ciało ptaka, stąd dotknięcie czujnika powoduje dużo większą zmianę wartości. Ponieważ jednak także pojemność elektryczną ptaka można zmierzyć, wystarczy odpowiednio dostroić wartość progową stosowaną przez nasz program.

Należy pobrać tę bibliotekę, rozpakować jej zawartość i skopiować pliki biblioteki do folderu *libraries* platformy Arduino. Więcej informacji na ten temat można znaleźć w dodatku A zatytułowanym „Instalacja bibliotek platformy Arduino".

W następnym kroku musimy utworzyć nowy projekt platformy Arduino i użyć wyrażenia `#include CapSense.h;`.

Z powodu dużo mniejszej wielkości samego ciała i powierzchni styku ciała z folią aluminiową wartości dla ptaka będą zasadniczo różniły się od wartości dla człowieka. Jeśli to możliwe, warto zmierzyć te różnice przy pomocy prawdziwego ptaka. Z radością odkryłem, że papużki moich dzieci są na tyle łakome, że ochoczo uczestniczą w testach, pod warunkiem że grzęda z czujnikiem pozwoli im się dostać do ziaren w karmniku. Moje testowe pomiary

[4] *http://www.arduino.cc/playground/Main/CapSense*

wykazały, że właściwa wartość bazowa powinna mieścić się w przedziale od 900 do 1400 oraz że pojemność elektryczna ciała ptaka zwiększa tę wartość do ponad 1500. Na podstawie tych wartości możemy opracować identyczny kod warunkowy jak w przypadku systemu powiadamiania o poziomie wody, tak aby program zmieniał stan w odpowiedzi na powiadomienia o lądowaniu i odlatywaniu ptaków.

Zacznijmy od napisania kodu, który załaduje bibliotekę CapSense i będzie wyświetlał odczytywane wartości pojemności w oknie monitora portu szeregowego.

Plik **TweetingBirdFeeder/BirdPerchTest.pde**

```
#include <CapSense.h>

#define ON_PERCH 1500
#define CAP_SENSE 30
#define ONBOARD_LED 13

CapSense foil_sensor = CapSense(10,7); // czujnik pojemnościowy
                                       // rezystor mostkujący wtyki cyfrowe nr 10 i 7
                                       // przewód połączony z rezystorem od strony
                                       // wtyku nr 7
int perch_value = 0;
byte perch_state = 0;

void setup()
{
    // na potrzeby komunikatów diagnostycznych w oknie portu szeregowego
    Serial.begin(9600);

    // ustawia wtyk dla wbudowanej diody LED
    pinMode(ONBOARD_LED, OUTPUT);
}

void SendPerchAlert(int perch_value, int perch_state)
{
    digitalWrite(ONBOARD_LED, perch_state ? HIGH : LOW);
    if (perch_state)
        Serial.print("Zdarzenie lądowania na grzędzie, perch_value=");
    else
        Serial.print("Zdarzenie opuszczenia grzędy, perch_value=");
    Serial.println(perch_value);
}

void loop() {
    // czeka sekundę w każdej iteracji pętli
    delay(1000);

    // pobiera wartość czujnika pojemnościowego
    perch_value = foil_sensor.capSense(CAP_SENSE);

    switch (perch_state)
```

```
    {
    case 0: // żaden ptak nie siedzi obecnie na grzędzie
        if (perch_value >= ON_PERCH)
        {
            perch_state = 1;
            SendPerchAlert(perch_value, perch_state);
        }
        break;
    case 1: // jakiś ptak siedzi teraz na grzędzie
        if (perch_value < ON_PERCH)
        {
            perch_state = 0;
            SendPerchAlert(perch_value, perch_state);
        }
        break;
    }
}
```

Warto zwrócić uwagę na wartość stałej `ON_PERCH` (równą 1500), którą porównujemy z zarejestrowaną wartością zmiennej `perch_value`. Z uwagi na różnice dotyczące przewodnictwa elektrycznego zastosowanej folii i samej powierzchni czujnika każdy powinien dostosować wartość progową reprezentowaną przez stałą `ON_PERCH` (tak jak dostosowywaliśmy odpowiednie progi w projekcie systemu powiadamiania o poziomie wody). Należy też zwrócić uwagę na wartość 30 przypisaną stałej `CAP_SENSE`. Ta wartość określa liczbę pobrań próbnych wartości w jednym cyklu mierzenia pojemności.

Skoro dysponujemy już działającym czujnikiem grzędy dla ptaków, czas opracować mechanizm wykrywający niski poziom ziarna. Jak to zrobić? Warto zastosować fotokomórkę.

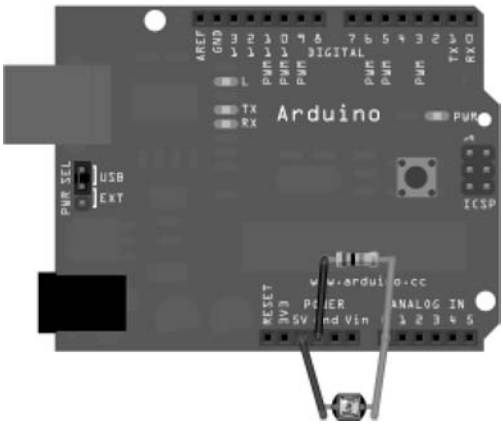
5.4. Czujnik ziarna

Fotokomórka mierzy intensywność światła — większa intensywność przekłada się na wyższe natężenie prądu; słabsze światło powoduje spadek tego natężenia. Szczegółowe wyjaśnienie działania fotokomórek i przewodnik na temat ich stosowania można znaleźć na stronie internetowej Ladyady[5]. Umieszczenie fotokomórki poniżej normalnego poziomu ziaren wsypanych do karmnika umożliwi nam wykrywanie zdarzenia polegającego na spadku poziomu karmy poniżej czujnika — do fotokomórki będzie wówczas docierało więcej światła, a nasz system będzie mógł wygenerować komunikat o konieczności uzupełnienia ziaren.

[5] *http://www.ladyada.net/learn/sensors/cds.html*

Przed wywierceniem dziur w karmniku i zainstalowaniem fotokomórki musimy jeszcze napisać odpowiedni kod i przetestować go w podobny sposób jak w przypadku własnoręcznie skonstruowanego czujnika pojemnościowego.

Jedną końcówkę fotokomórki należy połączyć z wtykiem 5-woltowym na płytce Arduino; drugą końcówkę należy połączyć z wtykiem analogowym nr 0. Wtyk analogowy nr 0 na płytce Arduino należy następnie zmostkować z wtykiem uziemienia za pomocą rezystora 10 kΩ (patrz rysunek 5.5 zatytułowany „Schemat połączenia fotokomórki"). Czy opisany schemat połączeń nie wygląda znajomo? Tak — identyczną konfigurację stosowaliśmy już dla innych czujników łączonych z platformą Arduino. Jest to dość typowy wzorzec łączenia wielu typów czujników z płytką Arduino.



Rysunek 5.5. Schemat połączenia fotokomórki

Po podłączeniu fotokomórki należy połączyć płytkę Arduino z komputerem (za pomocą przewodu szeregowego USB) i uruchomić środowisko Arduino IDE. Możemy teraz zastosować tę samą technikę co w przypadku czujnika pojemnościowego, aby prześledzić wartości dla wtyku analogowego nr 0 (wyświetlane w oknie monitora portu szeregowego środowiska Arduino IDE) i na tej podstawie wyznaczyć wartości bazowe — określić wartość dla odsłoniętej fotokomórki. Warto teraz zasłonić czujnik palcem, aby zablokować dopływ światła. Należy też zanotować, jak zmieniła się wartość odczytana z fotokomórki.

Tak jak w przypadku testów czujnika pojemnościowego, musimy opracować pewne procedury i wyrażenia warunkowe, które sprawdzą progi natężenia światła. W praktyce możemy nawet skopiować i wkleić kod testujący

czujnik pojemnościowy i ograniczyć się do dostosowania nazw zmiennych oraz przypisania odpowiednich wtyków.

Plik **TweetingBirdFeeder/SeedPhotocellTest.pde**

```
#define SEED           500
#define ONBOARD_LED     13
#define PHOTOCELL_SENSOR 0
int seed_value  = 0;
byte seed_state = 0;
void setup()
{
    // na potrzeby komunikatów diagnostycznych w oknie portu szeregowego
    Serial.begin(9600);

    // ustawia wtyk dla wbudowanej diody LED
    pinMode(ONBOARD_LED, OUTPUT);
}

void SendSeedAlert(int seed_value, int seed_state)
{
    digitalWrite(ONBOARD_LED, seed_state ? HIGH : LOW);
    if (seed_state)
        Serial.print("Uzupełnij ziarno, seed_value=");
    else
        Serial.print("Karma uzupełniona, seed_value=");
    Serial.println(seed_value);
}

void loop() {
    // czeka sekundę w każdej iteracji pętli
    delay(1000);

    // sprawdza wartość fotokomórki śledzącej poziom ziarna
    seed_value = analogRead(PHOTOCELL_SENSOR);

    switch (seed_state)
    {
    case 0: // pojemnik na ziarno został napełniony
        if (seed_value >= SEED)
        {
            seed_state = 1;
            SendSeedAlert(seed_value, seed_state);
        }
        break;
    case 1: // pojemnik na ziarno jest pusty
        if (seed_value < SEED)
        {
            seed_state = 0;
            SendSeedAlert(seed_value, seed_state);
        }
        break;
    }
}
```

Zmierzenie i ustalenie odpowiedniej wartości progowej dla fotokomórki (reprezentowanej przez stałą `SEED`) jest dużo prostsze niż w przypadku czujnika pojemnościowego, a uzyskana wartość jest bardziej wiarygodna. Mimo że fotokomórkę można zasłonić palcem, aby zmierzyć wartość w warunkach braku dopływu światła, lepszym rozwiązaniem będzie zasypanie czujnika prawdziwym ziarnem. Jeśli z jakiegoś powodu nie chcemy wiercić dziur w karmniku dla ptaków, aby zainstalować fotokomórkę, możemy umieścić czujnik na dnie papierowego kubka.

Tak jak podczas kalibrowania wartości progowych czujnika w systemie powiadamiania o poziomie wody, należy dodać następujące wiersze za wierszem `seed_value = analogRead(PHOTOCELL_SENSOR);` w głównej pętli programu:

```
Serial.print("seed_value=");
Serial.println(seed_value);
```

Należy zapisać wartość początkową zmiennej `seed_value`, po czym wypełnić pojemnik ziarnem i zmierzyć nową wartość. Na podstawie tych wartości należy określić wartość początkową i wartość progową dla fotokomórki.

Jeśli zasłonięcie fotokomórki nie powoduje żadnej zmiany wartości, warto raz jeszcze sprawdzić wszystkie połączenia. W przypadku mojej fotokomórki wartość bazowa mieściła się w przedziale od 450 do 550. Zasłonięcie czujnika palcem powodowało natychmiastowy spadek tej wartości poniżej 100. Każdy powinien zastosować wartości progowe dobrane na podstawie własnych testów. Musimy też pamiętać o konieczności ponownej kalibracji czujnika już po zamontowaniu w karmniku dla ptaków.

Skoro dysponujemy już działającymi mechanizmami monitorowania czujnika grzędy i fotokomórki, musimy znaleźć sposób sygnalizowania przekroczenia wartości progowych przyjętych dla tych czujników. Prowadzenie przewodu sieciowego od domowego koncentratora do gałęzi drzewa na zewnątrz budynku byłoby niepraktyczne. Sporym wyzwaniem byłaby także instalacja platformy Arduino z dołączonym modułem sieciowym w bardzo ograniczonej przestrzeni karmnika dla ptaków. W tej sytuacji warto zastosować wygodny mechanizm komunikacji bezprzewodowej (wymagający stosunkowo niewiele mocy elektrycznej), który w zupełności wystarczy do przesyłania powiadomień o przekroczeniu wartości progowych przez wskazania czujników. Po nawiązaniu komunikacji bezprzewodowej możemy użyć większej mocy obliczeniowej i większych zasobów pamięciowych do przetwarzania i analizy gromadzonych danych.

5.5. Komunikacja bezprzewodowa

Mimo że istnieją moduły platformy Arduino obsługujące wszechobecny standard Wi-Fi (802.11b/g), na przykład WiFly Shield firmy Sparkfun, w przypadku tej platformy do komunikacji bezprzewodowej częściej stosuje się moduły XBee. Początkowe nakłady związane z zakupem zestawu urządzeń XBee mogą być dość wysokie. Duże koszty wynikają z konieczności zakupu (oprócz pary modułów XBee) przewodu FTDI USB potrzebnego do połączenia jednego z tych modułów z komputerem, tak aby pełnił funkcję bezprzewodowego portu szeregowego.

Drugi moduł XBee najczęściej jest łączony z platformą Arduino. Istnieją też dodatkowe zestawy upraszczające łączenie tych elementów — umożliwiające instalowanie modułów XBee przy użyciu specjalnych wtyczek i wyświetlanie stanu transmisji danych za pomocą wbudowanych diod LED. Takie wizualne wskaźniki mogą być dość przydatne podczas diagnozowania połączenia pary modułów XBee i usuwania ewentualnych błędów. Mimo wszystkich trudności możliwości oferowane przez moduły XBee (niski pobór prądu i stosunkowo duży zasięg — maksymalnie 50 metrów) czynią z tych urządzeń wprost doskonałą technologię komunikacji bezprzewodowej na potrzeby tego projektu.

Z myślą o uproszczeniu łączenia modułów XBee firma Adafruit zaprojektowała zestaw adapterów, który jednak wymaga przylutowania kilku niewielkich komponentów do płytki Arduino. Adaptery należy stosować zgodnie z instrukcjami dostępnymi na stronie internetowej Ladyady[6].

Po połączeniu modułów XBee konfiguracja i nawiązanie komunikacji pomiędzy parą tych modułów nie są trudne. Warto jednak pamiętać, że jedno z najbardziej przydatnych narzędzi ułatwiających konfigurację tych modułów działa tylko w systemie Windows.

Zgodnie z instrukcjami opisującymi schemat łączenia modułów XBee w topologii punkt – punkt dostępnymi na stronie Ladyady[7] należy połączyć wtyki zasilania, uziemienia (*Gnd*), odbioru (*RX*) i transmisji (*TX*) jednego modułu XBee z zamontowanym adapterem odpowiednio do wtyków *5V*, *Gnd*, cyfrowego nr 2 i cyfrowego nr 3 na płytce Arduino. Płytkę Arduino należy następnie połączyć z komputerem, umieścić na platformie program

[6] *http://www.ladyada.net/make/xbee/*

[7] *http://ladyada.net/make/xbee/point2point.html*

testowy, otworzyć okno monitora portu szeregowego środowiska Arduino IDE i upewnić się, że jest ustawiona odpowiednia szybkość transmisji (9600). Należy następnie (jeszcze przed odłączeniem platformy Arduino) połączyć komputer z drugim modułem XBee za pośrednictwem przewodu FTDI USB. W następnym kroku musimy otworzyć sesję terminala portu szeregowego: program Hyperterminal w systemie Windows, polecenie `screen` w systemie Mac lub rozmaite programy do obsługi komunikacji szeregowej dostępne dla systemu Linux, na przykład Minicom[8].

Po nawiązaniu połączenia szeregowego wystarczy wpisać kilka znaków w oknie danych wejściowych użytej aplikacji. Jeśli oba moduły XBee zostały prawidłowo skonfigurowane, wpisane znaki powinny zostać wyświetlone w oknie monitora portu szeregowego środowiska Arduino IDE.

Korzystanie z narzędzia screen

Aplikacja `screen` jest wygodnym narzędziem do monitorowania portu szeregowego dostępnym dla platform na bazie systemu Unix, czyli systemów Mac OS X i Linux. Aby użyć tego programu w systemie OS X, należy określić port szeregowy, do którego podłączono przewód FTDI USB — można to zrobić za pomocą środowiska Arduino IDE. Z menu *Tools* tego środowiska należy wybrać opcję *Serial Port*, aby zidentyfikować przypisany port szeregowy.

W moim przypadku połączenie adaptera FTDI USB z XBee jest oznaczone jako urządzenie */dev/tty.usbserial-A6003SHc*, jednak na innym komputerze to samo połączenie może być reprezentowane w inny sposób (w zależności od pozostałych urządzeń podłączonych do komputera). Po otwarciu aplikacji terminala należy wpisać polecenie `screen /dev/tty.YOURDEVICE 9600`. W wyniku tego polecenia zostanie otwarty port szeregowy i uzyskamy możliwość wpisywania i otrzymywania znaków z szybkością transmisji 9600. Aby zamknąć narzędzie, należy nacisnąć kolejno kombinacje klawiszy *Ctrl+A* oraz *Ctrl+*.

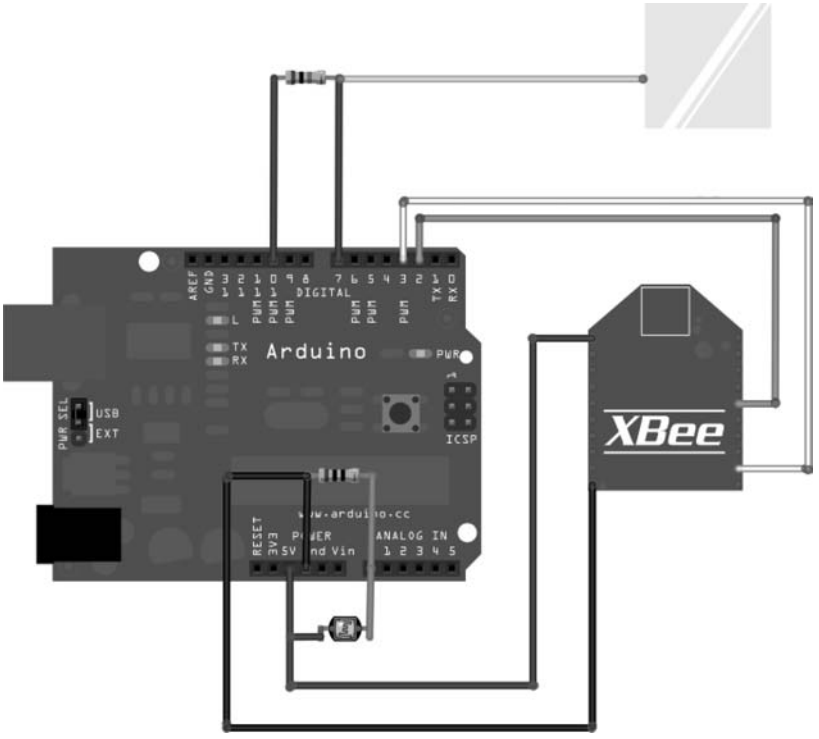
Jeśli do połączenia modułów XBee z płytką Arduino i przewodem FTDI użyto odpowiednich adapterów, w czasie bezprzewodowej transmisji znaków pomiędzy modułami XBee powinny migać diody LED (zielona podczas wysyłania danych i czerwona podczas odbioru danych).

[8] *http://alioth.debian.org/projects/minicom/*

Jeśli w oknie odbieranych danych nie widać żadnych znaków, należy jeszcze raz sprawdzić przewody łączące moduł XBee z odpowiednimi wtykami na płytce Arduino. Warto też zamienić te moduły miejscami, aby sprawdzić, czy oba urządzenia są rozpoznawane po połączeniu z komputerem za pomocą przewodu FTDI USB. W oknie terminala aplikacji portu szeregowego należy wpisać polecenie `AT` i sprawdzić, czy w odpowiedzi otrzymamy potwierdzenie `OK`.

Jeśli moduły XBee wciąż nie mogą nawiązać połączenia, warto poprosić o pomoc sprzedawcę, u którego zakupiono ten sprzęt.

Po udanej próbie nawiązania komunikacji przez parę modułów XBee możemy ponownie podłączyć fotokomórkę i czujnik pojemnościowy do płytki Arduino i połączyć kod obsługujący ten moduł z kodem analizującym warunki progowe obu czujników. Kompletny schemat połączeń tego systemu pokazano na rysunku 5.6 zatytułowanym „Ćwierkający karmnik dla ptaków z czujnikami i modułem XBee podłączonymi do platformy Arduino".



Rysunek 5.6. Ćwierkający karmnik dla ptaków z czujnikami i modułem XBee podłączonymi do platformy Arduino

Warto rozważyć użycie uniwersalnej płytki montażowej lub zlutowanie czujników, użytych przewodów i odpowiednich wtyków na płytce Arduino. Czytelnicy, którzy do testów wolą używać płytki uniwersalnej, muszą pamiętać, że taka płytka najprawdopodobniej nie zmieści się w karmniku, zatem przewody łączące elementy systemu trzeba będzie przylutować dopiero po instalacji w miejscu docelowym. W zależności od położenia i orientacji płytki Arduino Uno lub Nano wewnątrz karmnika być może będziemy musieli użyć dla adaptera XBee prostych wtyków zamiast standardowych wtyków wygiętych w prawo. Naszym celem jest umieszczenie wszystkich potrzebnych elementów wewnątrz karmnika w sposób gwarantujący bezpieczeństwo i jednocześnie możliwość konserwacji systemu. Warto przy tym pamiętać, że w przeciwieństwie do płytki Arduino Uno płytka Arduino Nano stosuje męskie wtyki. Oznacza to, że aby lepiej połączyć męskie wtyki na płytce Nano, należy zastosować przewody z żeńskimi końcówkami.

Kończenie szkicu

Musimy odczytywać wartości obu czujników — czujnika pojemnościowego z folii aluminiowej oraz fotokomórki. Początkowo odczytywane wartości będą trafiały do okna monitora portu szeregowego środowiska Arduino IDE, a docelowo (po wprowadzeniu drobnej zmiany w kodzie) będą wysyłane przez moduł XBee. Na tym etapie naszym celem jest połączenie kodu sprawdzającego wyznaczone wcześniej wartości progowe dla czujnika pojemnościowego i fotokomórki z kodem wysyłającym do modułu XBee ostrzeżenia o przekroczeniu tych wartości. Po dodaniu tej logiki do napisanego wcześniej kodu testującego stan grzędy i pojemnika na nasiona możemy zakończyć pracę nad szkicem dla tego projektu.

Plik **TweetingBirdFeeder/TweetingBirdFeeder.pde**

```
#include <CapSense.h>;
#include <NewSoftSerial.h>

#define ON_PERCH 1500
#define SEED 500
#define CAP_SENSE 30
#define ONBOARD_LED 13
#define PHOTOCELL_SENSOR 0

// ustawia wtyki cyfrowe na potrzeby szeregowego wysyłania/odbioru danych
// przez moduł XBee
NewSoftSerial XBeeSerial = NewSoftSerial(2, 3);
CapSense foil_sensor     = CapSense(10,7); // czujnik pojemnościowy
                                           // rezystor mostkujący wtyki cyfrowe
                                           // nr 10 i 7
```

```
                                        // przewód połączony z rezystorem
                                        // od strony wtyku nr 7
int perch_value  = 0;
byte perch_state = 0;
int seed_value   = 0;
byte seed_state  = 0;

void setup()
{
    // na potrzeby komunikatów diagnostycznych w oknie portu szeregowego
    Serial.begin(9600);

    // na potrzeby transmisji danych za pośrednictwem modułu XBee
    XBeeSerial.begin(9600);

    // ustawia wtyk dla wbudowanej diody LED
    pinMode(ONBOARD_LED, OUTPUT);
}

void SendPerchAlert(int perch_value, int perch_state)
{
    digitalWrite(ONBOARD_LED, perch_state ? HIGH : LOW);
    if (perch_state)
    {
        XBeeSerial.println("przylot");
        Serial.print("Zdarzenie lądowania na grzędzie, perch_value=");
    }
    else
    {
        XBeeSerial.println("odlot");
        Serial.print("Zdarzenie opuszczenia grzędy, perch_value=");
    }
    Serial.println(perch_value);
}

void SendSeedAlert(int seed_value, int seed_state)
{
    digitalWrite(ONBOARD_LED, seed_state ? HIGH : LOW);
    if (seed_state)
    {
        XBeeSerial.println("dosyp");
        Serial.print("Uzupełnij ziarno, seed_value=");
    }
    else
    {
        XBeeSerial.println("ziarnoWNormie");
        Serial.print("Karma uzupełniona, seed_value=");
    }
    Serial.println(seed_value);
}

void loop() {
    // czeka sekundę w każdej iteracji pętli
    delay(1000);
```

```
    // sprawdza wartość czujnika pojemnościowego na grzędzie
    perch_value = foil_sensor.capSense(CAP_SENSE);

    // sprawdza wartość fotokomórki śledzącej poziom ziarna
    seed_value = analogRead(PHOTOCELL_SENSOR);

    switch (perch_state)
    {
    case 0: // żaden ptak nie siedzi na grzędzie
        if (perch_value >= ON_PERCH)
        {
            perch_state = 1;
            SendPerchAlert(perch_value, perch_state);
        }
        break;
    case 1: // jakiś ptak siedzi teraz na grzędzie
        if (perch_value < ON_PERCH)
        {
            perch_state = 0;
            SendPerchAlert(perch_value, perch_state);
        }
        break;
    }

    switch (seed_state)
    {
    case 0: // pojemnik na ziarno został napełniony
        if (seed_value >= SEED)
        {
            seed_state = 1;
            SendSeedAlert(seed_value, seed_state);
        }
        break;
    case 1: // pojemnik na ziarno jest pusty
        if (seed_value < SEED)
        {
            seed_state = 0;
            SendSeedAlert(seed_value, seed_state);
        }
        break;
    }
}
```

Warto zwrócić uwagę na odwołania do biblioteki obsługującej czujnik pojemnościowy i nowej biblioteki obsługującej komunikację za pośrednictwem portu szeregowego na początku tego szkicu. Zmienne, które będą używały odwołań do tych bibliotek, zainicjalizowano w tym samym miejscu, w którym ustawiono wartości zmiennych progowych. W dalszej części kodu konfigurujemy połączenia z oknem monitora portu szeregowego i modułem XBee, a także z wbudowaną diodą LED na płytce Arduino (wtyk nr 13). Po zakończeniu inicjalizacji program uruchamia pętlę i czeka na przekroczenie wartości progowych dla czujników grzędy i (lub) ziarna. W razie wykrycia

zmiany warunków szkic wyśle odpowiednie komunikaty zarówno do okna monitorowania portu szeregowego w środowisku Arduino IDE, jak i do modułu komunikacji radiowej XBee.

Jeśli zarówno czujnik pojemnościowy, jak i fotokomórka prawidłowo przekazują wartości, należy przekierować dane z okna monitorowania portu szeregowego środowiska Arduino IDE do modułu XBee połączonego z platformą Arduino. Warto jeszcze otworzyć okno aplikacji testującej port szeregowy w trybie śledzenia modułu XBee podłączonego za pomocą przewodu FTDI i sprawdzić, czy wszystko działa prawidłowo — jeśli tak, dane wyświetlane w oknie monitora portu szeregowego środowiska Arduino IDE powinny być widoczne także w aplikacji portu szeregowego na komputerze, do którego podłączono przewód FTDI. Czyż komunikacja bezprzewodowa nie jest wspaniała?

Na tym etapie sprzęt potrzebny do realizacji tego projektu jest prawidłowo połączony i przetestowany — cała konstrukcja powinna przypominać karmnik mojego autorstwa pokazany na rysunku 5.7 zatytułowanym „Papużka może pomóc w testowaniu i diagnozowaniu wartości progowych czujnika grzędy dla zdarzeń lądowania i odlatywania ptaków".

Rysunek 5.7. Papużka może pomóc w testowaniu i diagnozowaniu wartości progowych czujnika grzędy dla zdarzeń lądowania i odlatywania ptaków

Zanim jednak przystąpimy do instalowania tego sprzętu w karmniku dla ptaków, musimy zbudować jeszcze jeden ważny komponent. Użyjemy języka

programowania Python do napisania krótkiego programu, który będzie nasłuchiwał komunikatów o lądujących ptakach, sprawdzał stan pojemnika z nasionami i publikował stosowne wpisy na Twitterze. Przejdźmy więc do pisania odpowiedniego kodu.

5.6. Ćwierkanie w Pythonie

Istnieje wiele języków programowania, w których można zaimplementować mechanizm monitorowania i interpretacji komunikatów przychodzących do konsoli portu szeregowego oraz wysyłania komunikatów za pośrednictwem portu szeregowego. Istnieje też wiele bibliotek Twittera dla różnych języków programowania.

Wybrałem język Python dla tego i wielu innych skryptów prezentowanych w tej książce, ponieważ język ten ma łatwą w interpretacji składnię, jest domyślnie instalowany wraz z systemami operacyjnymi Linux i Mac OS X oraz oferuje wiele przydatnych bibliotek (na przykład SQLite) w swojej podstawowej dystrybucji, zatem nie wymaga doinstalowywania tych komponentów. Czytelnikom, którzy chcą się nauczyć programowania w języku Python, polecam książkę *Python. Wprowadzenie* [LA03].

W tym projekcie będziemy potrzebowali prostego skryptu o nazwie *tweetingbirdfeeder.py*, który będzie realizował następujące zadania:

1. Rejestrowanie zdarzeń (wraz z datą i godziną) lądowania i odlotów ptaków na podstawie komunikatów wysyłanych przez czujnik grzędy. Dane mają być zapisywane w tabeli `birdfeeding` bazy danych `tweetingbirdfeeder`.
2. Rejestrowanie daty i godziny zdarzeń polegających na wykryciu braku ziaren i uzupełnieniu pojemnika na karmę. Dane mają być zapisywane w tabeli `seedstatus`, która także należy do bazy danych `tweetingbirdfeeder`.
3. Nasłuchiwanie danych przychodzących i wysyłanie komunikatów za pośrednictwem nadajnika XBee połączonego z komputerem przy użyciu przewodu FTDI. Reagowanie na zdarzenia poprzez zapisywanie danych wraz z datami, godzinami i rejestrowanymi warunkami.
4. Nawiązywanie połączenia z Twitterem za pośrednictwem usługi uwierzytelniania OAuth i wysyłanie wpisów na temat karmienia ptaków i poziomu ziaren w karmniku.

Na potrzeby tego projektu będziemy musieli zainstalować tylko dwie dodatkowe biblioteki Pythona: `pyserial` i `python-twitter`.

Oprócz publikowania wpisów na wybranym koncie na Twitterze warto zadbać o odpowiednią wizualizację trendów opisywanych w tych wpisach, na przykład częstotliwości wizyt ptaków w karmniku, liczby tych odwiedzin według dat i godzin oraz średniego czasu pomiędzy uzupełnianiem karmy. Takie rozwiązanie umożliwi nam śledzenie trendów na podstawie danych zarejestrowanych w ciągu godziny, doby, miesiąca i roku. Warunkiem prezentacji tego rodzaju statystyk jest gromadzenie danych w odpowiednim formacie.

Konfiguracja bazy danych

Ponieważ począwszy od wersji 2.5, język Python oferuje wbudowaną obsługę baz danych SQLite i ponieważ nasze dane nie wymagają wyszukanego, autonomicznego serwera bazy danych, baza SQLite jest wprost idealnym rozwiązaniem dla tego projektu. Mimo że wartości można by zapisywać w zwykłym pliku CSV (z danymi oddzielonymi przecinkami), użycie bazy danych SQLite ma dwie zasadnicze zalety. Po pierwsze, ten sposób przechowywania danych ułatwi wykonywanie zapytań analitycznych w przyszłości. Po drugie, baza danych oferuje większą elastyczność w zakresie gromadzenia danych o różnych rodzajach zdarzeń i zarządzania tymi danymi — w wielu przypadkach wystarczy tylko dodać odpowiednie kolumny do tabeli.

Do utworzenia bazy danych w formacie pliku `sqlite3` należy użyć polecenia wiersza poleceń `sqlite3`. Narzędzie jest domyślnie instalowane wraz z systemem Mac OS X. W większości systemów Linux należy pobrać to narzędzie z repozytorium aplikacji właściwego danej dystrybucji. W dystrybucjach systemu Linux na bazie Debiana, na przykład w systemie Ubuntu, instalacja aplikacji wymaga użycia polecenia `sudo apt-get install sqlite3 libsqlite3-dev`. Użytkownicy systemu Windows będą musieli pobrać narzędzie *sqlite3.exe* z witryny internetowej bazy danych SQLite[9].

Po zainstalowaniu systemu bazy danych należy wpisać polecenie `sqlite3` w oknie terminala. Polecenie spowoduje wyświetlenie komunikatów podobnych do tych pokazanych poniżej:

[9] *http://www.sqlite.org/download.html*

```
SQLite version 3.7.6
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite>
```

Instalacja systemu SQLite na komputerze czytelnika oczywiście może mieć inny numer wersji.

Musimy teraz wpisać wyrażenie języka SQL tworzące nową bazę danych. W tym celu należy opuścić powłokę poleceń `sqlite`, wpisując kolejno znaki `.q` i naciskając klawisz właściwy znakowi powrotu karetki. Zaraz potem należy ponownie uruchomić narzędzie `sqlite3`, tym razem podając nazwę bazy danych, która ma zostać otwarta.

Bazę danych dla tego projektu nazwiemy `tweetingbirdfeeder`, a jej dane będą przechowywane w pliku nazwanym *tweetingbirdfeeder.sqlite*. Ponieważ wspomniana baza danych jeszcze nie istnieje, system SQLite automatycznie utworzy odpowiedni plik. Plik bazy danych zostanie utworzony w katalogu, z którego uruchomiono narzędzie `sqlite3`. Jeśli na przykład polecenie `sqlite3` wpisaliśmy z poziomu katalogu *home*, plik nowej bazy danych zostanie utworzony właśnie w tym katalogu.

W bazie danych *tweetingbirdfeeder.sqlite* należy teraz utworzyć nową tabelę nazwaną `birdfeeding`. Strukturę tej tabeli pokazano poniżej:

Nazwa kolumny	Typ danych	Klucz główny?	Automatyczne zwiększanie?	Dopuszczalne wartości puste?	Unikatowa?
`id`	`INTEGER`	TAK	TAK	NIE	TAK
`time`	`DATETIME`	NIE	NIE	NIE	NIE
`event`	`TEXT`	NIE	NIE	NIE	NIE

Odpowiednią tabelę możemy utworzyć, wpisując następujące wyrażenie języka SQL w wierszu poleceń narzędzia sqlite:

```
[~]$ sqlite3 tweetingbirdfeeder.sqlite
SQLite version 3.7.6
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> CREATE TABLE "birdfeeding" ("id" INTEGER PRIMARY KEY NOT NULL UNIQUE,
"time" DATETIME NOT NULL,"event" TEXT NOT NULL);
```

Po utworzeniu tabeli `birdfeeding` potrzebujemy jeszcze jednej tabeli o podobnej strukturze. Nowa tabela, nazwana `seedstatus`, będzie należała do tej samej bazy danych:

Nazwa kolumny	Typ danych	Klucz główny?	Automatyczna inkrementacja?	Dopuszczalne wartości puste?	Unikatowa?
id	INTEGER	TAK	TAK	NIE	TAK
time	DATETIME	NIE	NIE	NIE	NIE
event	TEXT	NIE	NIE	NIE	NIE

Tak jak w przypadku tabeli `birdfeeding`, utworzenie nowej tabeli `seedstatus` wymaga wpisania odpowiedniego wyrażenia języka SQL w wierszu poleceń narzędzia sqlite:

```
[~]$ sqlite3 tweetingbirdfeeder.sqlite
SQLite version 3.7.6
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> CREATE TABLE "seedstatus" ("id" INTEGER PRIMARY KEY NOT NULL,
"time" DATETIME NOT NULL ,"event" TEXT NOT NULL );
```

Dodatek SQLite Manager

Mimo że narzędzia systemu SQLite obsługiwane z poziomu wiersza poleceń oferują wszystkie elementy niezbędne do tworzenia baz danych i zarządzania tymi bazami, w pewnych przypadkach prostszym rozwiązaniem jest korzystanie z aplikacji oferującej graficzny interfejs użytkownika. Aplikacje z takim interfejsem są szczególnie przydatne, jeśli musimy przewijać dużą liczbę wierszy w jednym oknie. Istnieje wiele aplikacji open source umożliwiających przeglądanie baz danych SQLite i oferujących graficzny interfejs użytkownika. Użytkowników przeglądarki internetowej Mozilla Firefox zachęcam do instalacji dodatku SQLite Manager, który można stosować na wielu różnych platformach[10].

Instalacja tego dodatku jest bardzo prosta. Z menu przeglądarki Firefox należy wybrać opcję *Dodatki*, po czym znaleźć dodatek SQLite Manager i kliknąć przycisk *Zainstaluj*. Po zainstalowaniu dodatku należy otworzyć zakładkę *Rozszerzenia* w oknie dodatków i kliknąć przycisk *Opcje* dla dodatku SQLite Manager. Utworzenie nowej bazy danych sprowadza się do kliknięcia ikony *New Database* na pasku narzędzi dodatku SQLite Manager. Równie proste jest zapisywanie i otwieranie plików baz danych SQLite.

[10] *https://addons.mozilla.org/en-US/firefox/addon/sqlite-manager/*

Dysponujemy już gotową bazą danych, zatem możemy przystąpić do pracy nad kodem importującym tę bazę oraz wspomniane wcześniej biblioteki Serial i Twitter. Musimy też przygotować skrypt nasłuchujący zdarzeń przesyłanych za pośrednictwem portu szeregowego i rejestrujący te dane (wraz ze znacznikami czasowymi) w odpowiedniej tabeli bazy danych.

Procedura przechwytywania zdarzenia powinna się zakończyć publikacją wpisu na Twitterze. Zanim jednak będziemy mogli wysyłać takie wpisy z poziomu programu, musimy utworzyć konto na Twitterze i zarejestrować się, aby uzyskać klucz interfejsu Twitter API wraz z powiązanymi danymi uwierzytelniającymi standardu OAuth. Przejdźmy więc do procesu uzyskiwania klucza do wspomnianego interfejsu API.

Dane uwierzytelniające interfejsu API Twittera

Zanim będziemy mogli umieszczać wpisy na Twitterze, musimy oczywiście założyć konto w tym serwisie. Co więcej, zanim będziemy mogli wysyłać wpisy z poziomu naszego programu, a konkretnie funkcji lub bibliotek języka programowania obsługujących standard uwierzytelniania OAuth[11], musimy utworzyć identyfikator aplikacji przypisany do odpowiedniego konta na Twitterze. Mimo że można wykorzystać do tego celu istniejące konto na Twitterze, moim zdaniem lepszym rozwiązaniem jest utworzenie zupełnie nowego konta specjalnie na potrzeby tego projektu. Dzięki temu użytkownicy śledzący moje dotychczasowe konto nie będą niepokojeni eksperymentalnymi powiadomieniami w związku z tworzonymi projektami. Takie rozwiązanie umożliwia też bardziej selektywny wybór odbiorców postów publikowanych przez tworzoną aplikację. Po rozważeniu wszystkich argumentów należy utworzyć nowe konto i uzyskać identyfikator aplikacji wygenerowany specjalnie dla projektu ćwierkającego karmnika dla ptaków.

Musimy teraz otworzyć stronę *dev.twitter.com*, zalogować się przy użyciu nowych danych uwierzytelniających i wybrać opcję *Create an app*. Na otwartej stronie *Create an application* należy podać unikatową nazwę nowej aplikacji, opis złożony z co najmniej dziesięciu znaków oraz adres strony internetowej rejestrowanej aplikacji. Czytelnicy, którzy nie dysponują własnymi witrynami internetowymi oferującymi możliwość pobrania aplikacji, mogą wpisać jakiś tymczasowy adres. Należy następnie zaznaczyć pole *Client under Application Type* i wybrać opcję *Read & Write* z menu *Default Access*

[11] *http://oauth.net/*

Type. Można też wskazać niestandardową ikonę aplikacji, jednak ten krok nie jest wymagany. Po wpisaniu tekstu zabezpieczenia CAPTCHA należy kliknąć przycisk *Create your Twitter application* na dole strony. Przed przejściem do następnego kroku należy jeszcze zapoznać się z warunkami korzystania z interfejsu Twitter API i zaakceptować proponowane zapisy.

Po zaakceptowaniu żądania wygenerowany zostanie unikatowy klucz API, klucz konsumenta standardu OAuth oraz klucz tajny konsumenta. Aby uzyskać dostęp do tokenu dostępu (*oauth_token*) i klucza tajnego tokenu dostępu (*oauth_token_secret*), należy kliknąć opcję *My Access Token* widoczną po lewej stronie. Obie wartości należy skopiować i zapisać w bezpiecznym, odpowiednio chronionym pliku. Obie wartości będą potrzebne do komunikacji z nowym kontem na Twitterze z poziomu kodu programu. Oczywiście należy zadbać o zachowanie tych wartości w ścisłej tajemnicy! Nie chcemy przecież, aby jakiś złośliwy, pozbawiony skrupułów użytkownik przejął nasz tajny token, wykorzystywał go do wysyłania spamu do naszych przyjaciół i doprowadzał do pasji całą społeczność użytkowników Twittera.

Skoro dysponujemy już kontem na Twitterze i poprawnym kluczem interfejsu API tego serwisu, możemy wykorzystać uzyskane dane uwierzytelniające w kodzie aplikacji języka Python napisanej dla naszego ćwierkającego karmnika dla ptaków.

Biblioteka Python-Twitter

Mimo że mamy dostęp do Twittera za pośrednictwem interfejsu API, wciąż nie dysponujemy mechanizmem komunikacji z Twitterem z poziomu skryptów języka Python. Warto wykorzystać do tego celu rozwiązania dostępne w bibliotece Python-Twitter[12]. Aby zainstalować obie biblioteki potrzebne do realizacji tego projektu, czyli Pyserial i Python-Twitter, należy pobrać najnowsze wersje tych bibliotek i użyć standardowego polecenia `sudo python setup.py install`. W przypadku instalowania tych bibliotek w systemie Mac OS X 10.6 (Snow Leopard) lub nowszym można skorzystać z już zainstalowanego narzędzia instalacyjnego Pythona nazwanego `easy_install`. Aby jednak uniknąć problemów związanych z 64-bitowymi wersjami bibliotek, odpowiednie polecenie należy poprzedzić flagą architektury i386, aby zainstalować bibliotekę Python-Twitter bez żadnych błędów. Kompletne polecenie dla tej biblioteki powinno mieć następującą postać: `sudo env ARCHFLAGS="-arch i386" easy_install python-twitter`.

[12] *http://code.google.com/p/python-twitter/*

Na tym etapie wszystkie niezbędne konta są skonfigurowane, a biblioteki — zainstalowane. Możemy więc dokończyć projekt, czyli napisać skrypt języka Python odpowiedzialny za nasłuchiwanie komunikatów za pośrednictwem modułu XBee podłączonego do portu szeregowego, zapisywanie tych komunikatów w bazie danych i publikowanie odpowiednich postów na Twitterze. Spróbujmy więc napisać skrypt Python implementujący ten złożony proces.

Plik **TweetingBirdFeeder/tweetingbirdfeeder.py**

```
# importuje biblioteki Pythona: DateTime, Serial, SQLite3 i Twitter
from datetime import datetime
import serial
import sqlite3
import twitter

# importuje moduł os w celu wyczyszczenia okna terminala i uruchomienia programu
# w systemie Windows należy użyć polecenia "cls"; w systemach Linux i OS X należy
użyć polecenia "clear"
import os

if sys.platform == "win32":
    os.system("cls")
else:
    os.system("clear")

# nawiązuje połączenie z portem szeregowym; nazwę URZĄDZENIE_SZEREGOWE należy zastąpić
# nazwą portu szeregowego, do którego podłączono moduł XBee (za pomocą przewodu FTDI)
XBeePort = serial.Serial('/dev/tty.URZĄDZENIE_SZEREGOWE', \
                         baudrate = 9600, timeout = 1)

# nawiązuje połączenie z bazą danych SQLite
sqlconnection = sqlite3.connect("tweetingbirdfeeder.sqlite3")

# tworzy kursor bazy danych
sqlcursor = sqlconnection.cursor()

# inicjalizuje obiekt interfejsu Twitter API
api = twitter.Api('klucz_konsumenta_OAuth', 'klucz_tajny_konsumenta_OAuth', \
                  'token_dostępu_OAuth', 'klucz_tajny_tokenu_dostępu_OAuth')

def transmit(msg):
    # uzyskuje i odpowiednio formatuje bieżącą datę i godzinę
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # sprawdza rodzaj komunikatu i przypisuje odpowiednie parametry odpowiedzi
    if msg == "przylot":
        tweet = "Jakiś ptak wylądował na grzędzie!"
        table = "birdfeeding"
    if msg == "odlot":
        tweet = "Ptak odleciał z grzędy!"
        table = "birdfeeding"
    if msg == "dosyp":
        tweet = "Pojemnik na ziarno dla ptaków jest pusty."
```

```
        table = "seedstatus"
    if msg == "ziarnoWNormie":
        tweet = "Ziarno w karmniku zostało uzupełnione."
        table = "seedstatus"

    print "%s - %s" % (timestamp.strftime("%Y-%m-%d %H:%M:%S"), tweet)

    # zapisuje odpowiednie zdarzenie w bazie danych SQLite
    try:
        sqlstatement = "INSERT INTO %s (id, time, event) \
        VALUES(NULL, \"%s\", \"%s\")" % (table, timestamp, msg)
        sqlcursor.execute(sqlstatement)
        sqlconnection.commit()
    except:
        print "Nie można zapisać zdarzenia w bazie danych."
        pass

    # publikuje informację na Twitterze
    try:
        status = api.PostUpdate(msg)
    except:
        print "Nie można opublikować postu na Twitterze"
        pass

# główna pętla programu
try:
    while 1:
        # nasłuchuje znaków przychodzących (wysyłanych przez moduł radiowy XBee
        # zainstalowany w karmniku)
        message = XBeePort.readline()

        # w zależności od rodzaju otrzymanego komunikatu
        # rejestruje odpowiednie zdarzenie w bazie danych i publikuje post na Twitterze
        if "przylot" in message:
            transmit("przylot")
        if "odlot" in message:
            transmit("odlot")
        if "dosyp" in message:
            transmit("dosyp")
        if "ziarnoWNormie" in message:
            transmit("ziarnoWNormie")

except KeyboardInterrupt:
    # przerywa program w momencie wykrycia naciśnięcia kombinacji klawiszy Ctrl+C
    print("\nProgram nasłuchujący komunikatów ćwierkającego karmnika dla ptaków kończy pracę.\n")
    sqlcursor.close()
    pass
```

Po załadowaniu niezbędnych bibliotek `datetime`, `serial`, `sqlite` i `twitter` czyścimy okno terminala (za pomocą polecenia `cls` w systemie Windows lub polecenia `clear` w pozostałych systemach operacyjnych) i nawiązujemy połączenie z modułem XBee (połączonym z portem szeregowym komputera za pomocą przewodu FTDI). Zaraz potem nawiązujemy połączenie

z utworzonym wcześniej plikiem bazy danych *tweetingbirdfeeder.sqlite3* i rozpoczynamy wykonywanie nieskończonej pętli `while` do momentu naciśnięcia kombinacji klawiszy *Control*-C, która powoduje zamknięcie programu. Jeśli moduł XBee otrzyma i prawidłowo rozpozna komunikat, skrypt wywoła funkcję `def transmit(msg)`, która odpowiada za analizę składniową zmiennej `msg`, dodanie opisu danego zdarzenia, zapisanie odpowiedniego komunikatu w bazie danych oraz opublikowanie postu na Twitterze.

Jeśli platforma Arduino jest uruchomiona i jeśli para modułów XBee nawiązała połączenie i jest zasilana, możemy przetestować wykrywanie przekraczania wartości progowych, kilkukrotnie dotykając czujnik grzędy i zasłaniając fotokomórkę, tak aby system wysłał kilka sygnałów o tych zdarzeniach. Jeśli w oknie terminala nie zostały wyświetlone żadne błędy związane z wykonywaniem tego skryptu, warto otworzyć plik *tweetingbirdfeeder.sqlite3* w oknie *Browse and Search* narzędzia SQLite Manager i sprawdzić, czy informacje o zdarzeniach dotyczących obu czujników zostały zarejestrowane i oznaczone odpowiednimi znacznikami czasowymi. Jeśli wszystkie te mechanizmy zadziałały prawidłowo, warto jeszcze zalogować się na Twitterze (do konta używanego do publikowania komunikatów o tych zdarzeniach) i sprawdzić, czy wszystkie posty zostały opublikowane we właściwej kolejności.

Nasz system jest prawie gotowy. Pozostało nam już tylko kilka usprawnień elementów sprzętowych.

5.7. Kończenie projektu

Aby zapewnić pełną funkcjonalność tego projektu, musimy jeszcze zabezpieczyć niezbędne urządzenia (w tym przypadku płytkę Arduino połączoną z modułem XBee) przed warunkami atmosferycznymi i zainstalować w karmniku dla ptaków, zamontować fotokomórkę blisko podstawy karmnika, wsypać ziarno do karmnika, połączyć płytkę Arduino i moduł XBee ze źródłem zasilania oraz umieścić całość na zewnątrz budynku, ale w zasięgu drugiego modułu XBee (podłączonego do komputera).

Każdy, kto nie mieszka w strefie klimatycznej z rzadkimi opadami deszczu, powinien dobrze zabezpieczyć urządzenia elektryczne przed działaniem wody. W moich testów wynika, że umieszczenie tych elementów elektronicznych w dwóch szczelnych workach w zupełności wystarczy do zabezpieczenia płytki Arduino i modułu XBee przed deszczem. Jeśli jednak nie planujemy zasilania tych komponentów za pomocą 9-woltowej baterii, którą można by

umieścić wraz z elektroniką w workach (takie rozwiązanie sprawdziłoby się w przypadku krótkich sesji gromadzenia danych, jednak nasz system bardzo szybko wyczerpałby baterię), musimy doprowadzić do płytki Arduino przewód zapewniający stały dopływ energii.

Aby doprowadzić przewód zasilający, wystarczy zrobić niewielki otwór w workach foliowych, jednak takie rozwiązanie narazi cały system na działanie wilgoci. Aby zminimalizować ryzyko zalania elementów elektronicznych, warto zabezpieczyć miejsce łączenia worków z przewodem mocno naciągniętą folią spożywczą, tak aby szczelnie zamknąć worek i zabezpieczyć łączenie przed poluzowaniem lub wyślizgnięciem się wskutek zmieniającej się pogody.

Stosowanie odpowiednio zabezpieczonego przewodu zasilającego (na przykład przewodu sprzedawanego z myślą o zasilaniu lampek w okresie świąt Bożego Narodzenia) będzie tańsze i łatwiejsze do testowania. Konstruktorzy szczególnie dbający o środowisko naturalne mogą jednak zainwestować nieco większe środki w alternatywne rozwiązanie — w technologię energii odnawialnej w formie fotowoltaicznego systemu zasilania.

Przed zakupem odpowiedniego przenośnego systemu zasilania energią słoneczną warto sprawdzić, czy produkt generuje napięcie 5 V, jest odpowiednio wytrzymały i zawiera wbudowany akumulator, który będzie gromadził energię. Dobrym przykładem stosunkowo niedrogich rozwiązań tego typu są zasilacze z serii Solio Bolt[13].

Czytelnicy, którzy wolą systemy fotowoltaiczne zawierające akumulatory o większych pojemnościach, muszą być przygotowani na nieco większe koszty. Takie firmy jak Sunforce Products mają w ofercie rozmaite rozwiązania podtrzymujące napięcie, układy ładowania akumulatorów i sterowniki projektowane z myślą o większym obciążeniu[14].

Panel słoneczny należy zamontować w odpowiedniej odległości od karmnika, tak aby był wystawiony na promienie słoneczne. Jeśli to możliwe, panel należy zamontować pod kątem dziewięćdziesięciu stopni do promieni słonecznych, aby uzyskać jak najwięcej energii. W zależności od strefy klimatycznej i średniego poziomu nasłonecznienia być może trzeba będzie poszukać alternatywnych lub uzupełniających rozwiązań, jak ładowarka zasilana przez turbinę wiatrową lub nawet dynamo napędzane siłą mięśni.

[13] *http://www.solio.com/chargers/*

[14] *http://www.sunforceproducts.com/results.php?CAT_ID=1*

W ramach tego projektu udało nam się osiągnąć całkiem sporo zupełnie nowych celów — od zastosowania fotokomórki i własnoręcznie zbudowanego czujnika pojemnościowego, przez opanowanie sztuki łączenia w pary modułów XBee i nawiązywania komunikacji bezprzewodowej między nimi, po napisanie skryptu rejestrującego dane w ustrukturyzowanej bazie danych, reagującego na zdarzenia i publikującego posty na Twitterze za pośrednictwem interfejsu API tego serwisu. Zastosowaliśmy także autonomiczny system zasilania płytki Arduino i modułu XBee oraz zabezpieczyliśmy te wrażliwe komponenty elektroniczne przed szkodliwym działaniem warunków atmosferycznych.

Te cenne lekcje z pewnością wykorzystamy przynajmniej w projektach realizowanych w części pozostałych rozdziałów.

5.8. Następne kroki

Czujnik pojemnościowy i fotokomórkę można wykorzystać w najróżniejszych projektach automatyzacji domu. Poniżej opisałem zaledwie kilka pomysłów, które warto rozważyć:

- Płytkę Arduino z podłączonymi fotokomórką i modułem XBee można umieścić (wraz z bateriami) w lodówce lub zamrażarce, aby wykrywać, jak często i jak długo drzwi lodówki są otwarte. Na podstawie zgromadzonych danych można obliczyć energię traconą w poszczególnych miesiącach w związku ze zbyt częstym zaglądaniem do lodówki. W razie stwierdzenia, że koszty traconej w ten sposób energii są zbyt wysokie, system może wysłać wiadomości poczty elektronicznej lub powiadomienia na Twitterze przypominające domownikom o gazach cieplarnianych i globalnym ociepleniu.
- Jeśli uznamy, że szukanie włącznika światła w ciemnościach jest zbyt kłopotliwe, być może powinniśmy zastosować czujnik pojemnościowy z folii aluminiowej i zainstalować ten „włącznik" na ścianie przy wejściu do piwnicy lub garażu bądź na poziomej powierzchni stołu na wprost wejścia.
- Na podstawie wartości analogowych odczytywanych z fotokomórki można mierzyć cykle dni i nocy oraz poziom nasłonecznienia, aby w ten sposób zgromadzić dane przydatne na przykład w ogrodnictwie.

> Czy sadzenie określonych gatunków kwiatów, owoców lub warzyw w odpowiednio wybranym okresie pozwoli przyspieszyć lub opóźnić wzrost roślin? Jak długo rośliny były wystawione na pełne słońce, a przez ile dni niebo było zachmurzone?

Oprócz wymienionych pomysłów istnieje jeszcze mnóstwo sposobów analizowania danych gromadzonych na podstawie zdarzeń dotyczących naszego karmnika dla ptaków. Możemy użyć biblioteki języka Python generującej wykresy (na przykład CairoPlot) do wizualizacji średniego czasu przebywania ptaka w karmniku[15]. Ile czasu zajęło ptakom zjedzenie całego ziarna? Jaki wpływ na godziny przylotów i czas przebywania w karmniku miała pogoda na zewnątrz? Czy zmiana rodzaju ziarna wpłynęła na czas przebywania ptaków w karmniku i częstotliwość przylotów?

Warto rozważyć udostępnienie wpisów na Twitterze innym entuzjastom ptaków, aby zbudować szeroką sieć społecznościową użytkowników czytających i przekazujących dalej dane generowane przez nasz karmnik. Być może wzorce zaobserwowane w jednym karmniku będą nieco inne niż w przypadku karmników zamontowanych na innych obszarach geograficznych — znajomi z serwisu społecznościowego być może będą zainteresowani wspólną analizą trendów dotyczących populacji ptaków, cykli migracyjnych i pozostałych czynników wpływających na zachowania naszych upierzonych przyjaciół.

[15] *http://cairoplot.sourceforge.net/*

Skorowidz

A

B

C

D

E

F

G

H

I

K

M

O

P

R

S

T

U

V

W

X

Już za chwilę możesz mieszkać w niezwykłym i najinteligentniejszym miejscu w okolicy!

Chcesz, żeby Twój dom był inteligentny? Wszystko jest w Twoim zasięgu! Mike Riley, miłośnik technologii, pomoże Ci zrealizować różne nietypowe projekty automatyzacji domu. Przeprowadzi Cię krok po kroku przez proces zdobywania potrzebnych elementów, budowy fizycznych rozwiązań i tworzenia potrzebnego kodu. Nauczysz się używać ciekawych technologii i usług oraz rozwijać dostępne rozwiązania.

System powiadamiający o poziomie wody, elektryczny pies obronny czy system wykrywający dostarczone paczki to tylko część pomysłów, które pozwolą Ci opanować niuanse tworzenia własnych projektów. Dzięki lekturze tej książki samodzielnie zbudujesz karmnik, który będzie publikował na Twitterze wpisy o przylatujących ptakach i konieczności uzupełnienia ziarna. Za pomocą odpowiedniego programu będziesz sterował oświetleniem wewnątrz i na zewnątrz domu. Zabezpieczysz swój dom przed nieproszonymi gośćmi. Stworzysz autonomiczny system podnoszenia i opuszczania zasłon zależnie od temperatury i natężenia światła w pokoju. Sprawisz, że Twój dom przemówi w momencie, gdy przyjdzie do Ciebie e-mail lub odwiedzą Cię goście… Książka ta będzie niewyczerpanym źródłem inspiracji dla kolejnych innowacji w Twoim domu. Spróbuj tego, naprawdę warto!

Sprawdź, jak zaszczepić inteligencję w Twoim domu, i zaprogramuj:

- powiadomienie o poziomie wody
- elektrycznego psa obronnego
- wykrywanie dostarczenia paczki
- internetowy włącznik światła
- zamek sterowany telefonem

Nr katalogowy: 13143

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

ISBN 978-83-246-5675-2

Cena: 49,00 zł

Informatyka w najlepszym wydaniu