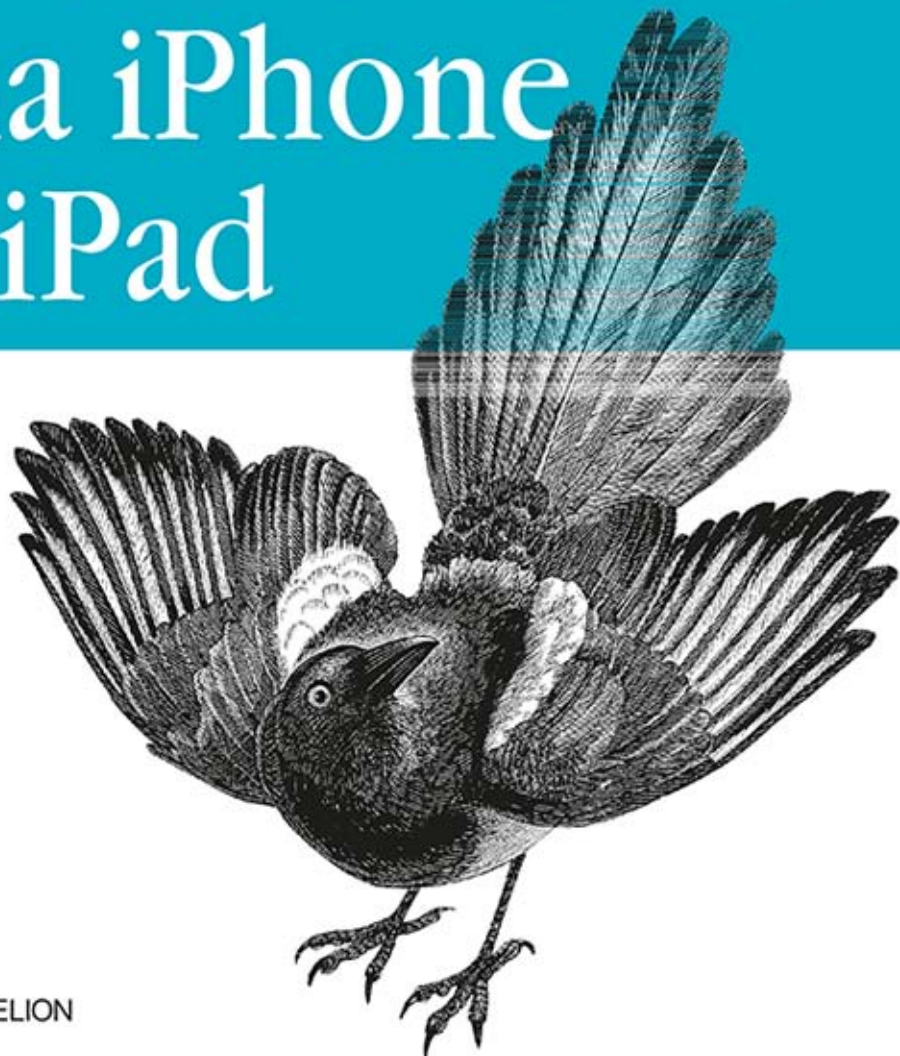


Niesamowite możliwości urządzeń z iOS!

*Niekonwencjonalne gadżety
z technologią Arduino i techBASIC*

Projekty elektroniczne na iPhone i iPad



O'REILLY®

Mike Westerfield

Tytuł oryginału: Building iPhone and iPad Electronic Projects

Tłumaczenie: Robert Górczyński

ISBN: 978-83-246-8890-6

© 2014 Helion S.A.

Authorized Polish translation of the English edition of Building iPhone and iPad Electronic Projects, ISBN 9781449363505 © 2013 James M. Westerfield.

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

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

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

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

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

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

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

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

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

Wprowadzenie	7
1. Wprowadzenie do techBASIC i czujników wbudowanych w urządzenia iOS	13
Własny tricorder	13
Krótkie wprowadzenie do techBASIC	14
Aplikacja techBASIC Sampler	15
Uruchomienie pierwszego programu	15
Tworzenie programu	17
Przyśpieszeniomierz	20
2. Uzyskanie dostępu do innych wbudowanych czujników	33
Żyroskop	34
Radiany czy stopnie?	40
Magnetometr	41
Uzyskanie szybszej odpowiedzi z czujnika	47
Wyznaczanie kursu	51
Położenie	52
Twój własny tricorder	53
3. Budujemy wykrywacz metalu	55
Magnetometr w urządzeniach iPhone i iPad	55
Ziemskie pole magnetyczne	56
Użycie iPhone'a lub iPada jako wykrywacza metalu	58
Konwersja aplikacji Magnetometer na wykrywacz metalu	60
Używanie wykrywacza metalu	61
Co dalej?	63
4. HiJack	65
Co to jest HiJack?	65
Budowa czujnika	67
Zasilanie zewnętrzne dla urządzenia HiJack	71

Program Hello HiJack	73
Kiedy sprawy idą źle	75
Lepsza wersja programu HiJack	75
Co dalej?	80
5. Budujemy wilgotnościomierz za pomocą urządzenia HiJack	81
Dodanie wilgotnościomierza do tricordera	81
Budowa wilgotnościomierza	82
Kalibracja	83
Zebranie danych do kalibracji	83
Przenoszenie plików danych do oraz z techBASIC	85
Użycie danych kalibracji	86
Lepsza wersja oprogramowania	89
Pełny kod źródłowy programu Moisture Meter	96
6. Technologia Bluetooth Low Energy	99
Czym jest technologia Bluetooth Low Energy?	99
Urządzenie SensorTag firmy Texas Instruments	101
Tworzenie programów obsługujących urządzenia typu BLE	103
Przyśpieszeniomierz	115
Co to jest przyśpieszeniomierz?	115
Uzyskanie dostępu do przyśpieszeniomierza	116
Użycie przyśpieszeniomierza	118
Kod źródłowy programu odczytującego wartości z przyśpieszeniomierza	120
Barometr	123
Uzyskanie dostępu do barometru	124
Kod źródłowy programu odczytującego wartości z barometru	128
Żyroskop	132
Uzyskanie dostępu do żyroskopu	132
Użycie żyroskopu	134
Kod źródłowy programu odczytującego wartości z żyroskopu	135
Magnetometr	138
Uzyskanie dostępu do magnetometru	139
Użycie magnetometru	141
Kod źródłowy programu odczytującego wartości z magnetometru	141
Wilgotnościomierz	144
Uzyskanie dostępu do wilgotnościomierza	145
Kod źródłowy programu odczytującego wartości z wilgotnościomierza	147
Termometr	150
Uzyskanie dostępu do termometru	150
Użycie termometru	152
Kod źródłowy programu odczytującego wartości z termometru	153
Co dalej?	156

7. Model rakiety jako sterowane iPhone'em urządzenie typu BLE	157
Odrobina informacji na temat budowy rakiet	158
Listy niezbędnych elementów	159
ST-1	159
ST-2	160
Inne elementy potrzebne dla obu budowanych rakiet	161
Dlaczego używamy SensorTag?	161
Konstrukcja	162
Model rakiety ST-2 przynoszącej iPhone i SensorTag	163
Model rakiety ST-1 przynoszącej tylko SensorTag	168
Program odpowiedzialny za zbieranie danych	169
Oprogramowanie SensorTag działające w zakresie $\pm 8G$	180
Wskazówki dotyczące lotów	182
Silniki	182
Spadochrony	182
Pogoda podczas lotów	182
Start rakiety	183
Dane	183
Analiza danych	183
Analiza danych za pomocą programu Rocket Flight Analysis	185
Prędkość i wysokość	186
Obrót i ciśnienie	189
Czego się dowiedzieliśmy?	189
Wyniki dla rakiety ST-1	189
Wyniki dla rakiety ST-2	191
8. Zdalne sterowanie samochodem	
za pomocą urządzenia typu BLE i mikrokontrolera Arduino	193
Sterowanie samochodem za pomocą urządzenia typu BLE	194
Wybór zdalnie sterowanego samochodu	196
Demontaż samochodu	196
Modyfikacja samochodu	200
Mostek H	201
Układ scalony Texas Instruments SN754410	202
Montaż całego układu elektronicznego	204
Sterowanie mikrokontrolerem Arduino Uno	211
Instalacja oprogramowania Arduino	211
Pobranie oprogramowania Firmata	213
Oprogramowanie	216
Modulacja szerokości impulsów	216
Wracamy do oprogramowania	218
Uruchom silniki!	228

9. Połączenie BLE między urządzeniami iOS	231
Tryb podległy w BLE	231
Program BLE Chat	232
Konfiguracja urządzeń	232
Użycie usług	234
10. Paddles, czyli hołd złożony grze Pong	245
Klasyczna gra Pong	245
Gra Paddles	246
Program obsługujący paletkę	247
Program obsługujący konsolę gry Paddles	251
11. Wi-Fi	263
Komunikacja ze światem	263
Protokoły HTTP, FTP i TCP/IP	264
WiFly	265
Układ elektroniczny	266
Nawiązanie połączenia sieciowego	267
Komunikacja z TCP/IP	268
Prosty program terminala	269
Wi-Fi i Arduino	271
Wczytanie oprogramowania do mikrokontrolera Arduino	271
Układ elektroniczny	273
Komunikacja za pomocą programu terminala	274
12. Serwomechanizmy Wi-Fi	275
Ogólne informacje o serwomechanizmach	275
Kontroler Pololu Serial Servo Controller	276
Układ elektroniczny	279
Maski na Halloween	281
Oprogramowanie	281
Wypróbuj zbudowane urządzenie	285
Serwomechanizmy wykonujące ruch do przodu i do tyłu	285
Zakończenie	288
Skorowidz	289

Uzyskanie dostępu do innych wbudowanych czujników

W tym rozdziale

Przygotowania

Programy przedstawione w tym rozdziale są zmodyfikowanymi wersjami programu omówionego w rozdziale 1. Jeżeli natkniesz się na jakiegokolwiek niejasności, wtedy zapoznaj się z rozdziałem 1.

Wyposażenie

Wymagane jest urządzenie iPhone, iPod touch bądź iPad działające pod kontrolą systemu iOS 5 lub nowszego.

Oprogramowanie

Wymagana jest kopia aplikacji techBASIC lub techBASIC Sampler.

Czego się nauczysz?

Z tego rozdziału dowiesz się, jak uzyskać dostęp do magnetometru i żyroskopu, które są wbudowane w większości urządzeń iOS. Wymienione czujniki można wykorzystać w praktycznie dowolnym celu, począwszy od wyznaczania kursu, aż po rzeczywistość rozszerzoną (ang. *Augmented Reality*).

Zanim zakończysz lekturę rozdziału, dowiesz się nieco więcej na temat techBASIC. Między innymi poznasz inny sposób uzyskania dostępu do czujników w znacznie krótszym czasie. Ponadto przekonasz się, że można używać systemu pomocy w aplikacji techBASIC do wyszukiwania informacji na temat samego języka i oferowanych przez niego poleceń.

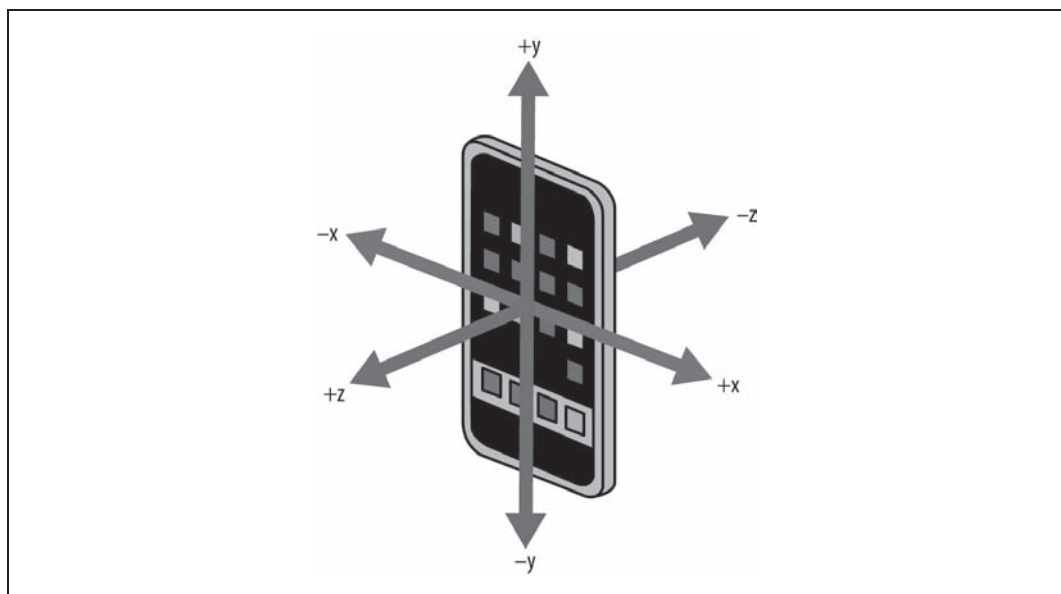
Na końcu rozdziału pokrótce zapoznasz się z dwoma innymi usługami. Wprawdzie to nie są typowe czujniki, ale dostęp do nich odbywa się w taki sam sposób jak do czujników. Pierwsza ze wspomnianych usług to GPS, natomiast druga to wyznaczanie kursu (ang. *heading*), które wykorzystuje magnetometr i kompas w celu znalezienia odpowiedniego kursu.

Utworzenie naszego pierwszego programu opartego na graficznym interfejsie użytkownika i przeznaczonego do wyświetlania danych czujnika wbudowanego w urządzenia iPhone i iPad wymagało nieco pracy. Nic dziwnego, poznajemy nowe środowisko programistyczne, a także

czujniki oferowane przez urządzenia iOS. Mając już podstawową wiedzę, warto ją rozszerzyć i wykorzystać nowe możliwości. Dzięki temu będziemy mogli ukończyć prace nad naszym tricorderem.

Żyroskop

Począwszy od modelu iPhone 4, wszystkie smartfony iPhone są wyposażone w żyroskop trójosiowy. Być może sądzisz, że żyroskop jest niepotrzebny, ponieważ przyśpieszeniomierz może dostarczyć wszelkich informacji o położeniu urządzenia (patrz rysunek 2.1), śledzić położenie i informować o wszelkich zmianach orientacji. Okazuje się jednak, że przyśpieszeniomierz nie sprawdza się jako zamiennik żyroskopu. Jednym z powodów są prawa fizyki. Grawitacja Ziemi to nie jedyne przyśpieszenie wychwytywane przez przyśpieszeniomierz, sam ruch także wiąże się z pewnym przyśpieszeniem. Inny powód jest czysto praktyczny. Przyśpieszeniomierz nie może wykrywać gwałtownych zmian w orientacji tak dobrze jak żyroskop, który został opracowany właśnie do tego celu.



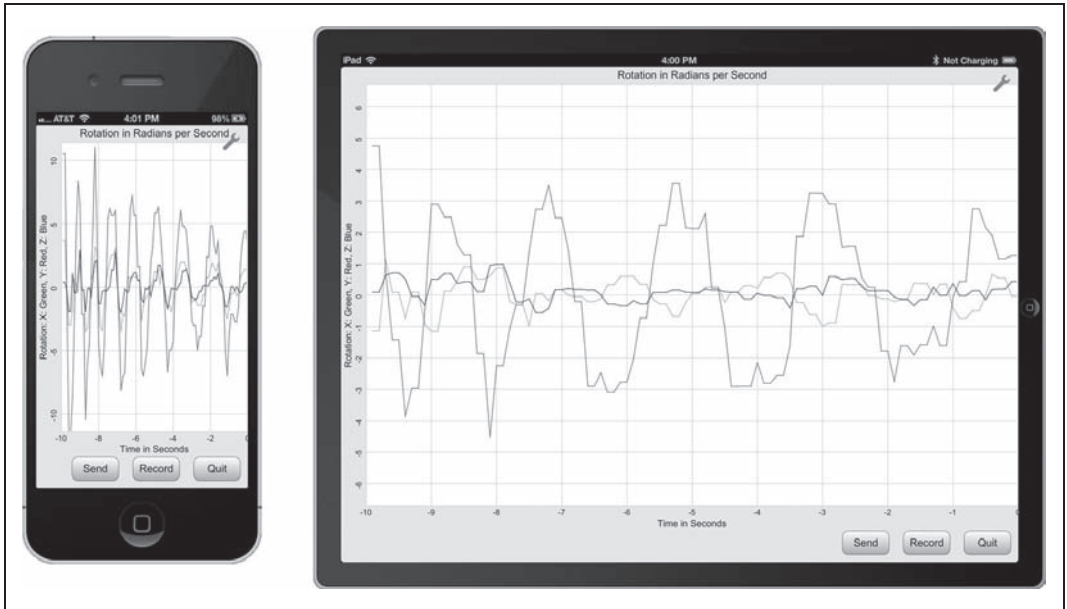
Rysunek 2.1. Orientacja osi jest taka sama w przyśpieszeniomierzu, żyroskopie i magnetometrze

Interfejs użytkownika aplikacji żyroskopu, którą utworzymy w tym rozdziale (patrz rysunek 2.2), jest bardzo podobny do aplikacji przyśpieszeniomierza omówionej w poprzednim rozdziale.

Kod źródłowy aplikacji również jest bardzo podobny do aplikacji omówionej w poprzednim rozdziale. Oczywiście między wspomnianymi aplikacjami występują pewne różnice, ale stworzonego tutaj programu nie będziemy omawiać wiersz po wierszu. Zamiast tego przyjrzymy się dzielącym je różnicom.

Pierwsza i najbardziej oczywista zmiana dotyczy czujnika, z którego będą pobierane wartości. W programie nie pobieramy wartości z czujnika przyśpieszeniomierza:

```
PRINT Sensors.accel
```

Rysunek 2.2. Uruchomiona aplikacja Gyroscope

ale z czujnika żyroskopu, korzystając z bardzo podobnego polecenia:

```
PRINT Sensors.gyro
```

W programie są jeszcze inne subtelne zmiany. Czy wiesz, że żyroskop jest dostępny tylko w niektórych modelach smartfona iPhone? Oznacza to, że bardzo ważne jest sprawdzenie dostępności czujnika, zanim będzie można przystąpić do odczytywania jego danych. Z tego powodu w dwóch miejscach programu wprowadzono odpowiednie zmiany. Pierwsza zmiana znajduje się już na początku programu, w kodzie konfiguracyjnym odpowiedzialnym za pobranie wartości początkowej czasu:

```
! Pobranie i ustawienie wartości początkowej czasu dla żyroskopu.
DIM t0 AS DOUBLE
IF Sensors.gyroAvailable THEN
  WHILE t0 = 0
    r = Sensors.gyro
    t0 = r(4)
  WEND
END IF
```

Polecenie IF sprawdza dostępność żyroskopu, zanim program spróbuje odczytać z niego dane. Kod zawiera także pętlę WHILE, aby zagwarantować pobranie niezerowej wartości czasu. Powód jest prosty: zanim nie osiągnie pełnej gotowości do pracy, żyroskop może zwrócić wartość zero.

Druga zmiana została wprowadzona na końcu podprocedury setUpGUI:

```
! Upewnienie się o dostępności żyroskopu. Jeżeli czujnik jest niedostępny,
! należy wyświetlić odpowiedni komunikat i zakończyć działanie programu.
IF NOT Sensors.gyroAvailable THEN
  msg$ = "This device does not have a gyroscope. "
  msg$ = msg$ & "The program will exit."
  button = Graphics.showAlert("No Gyro", msg$)
  STOP
END IF
```

Powyższy fragment kodu sprawdza, czy urządzenie jest wyposażone w żyroskop. W przypadku jego braku następuje wyświetlenie odpowiedniego komunikatu i zakończenie działania programu.

Inna drobna zmiana została wprowadzona na początku podprocedury setUpGUI. Polecenie ustawiające częstotliwość pobierania danych z czujnika musi być zmodyfikowane, aby zdefiniować częstotliwość dla żyroskopu, a nie przyspieszeniomierza. Poniżej przedstawiono nowy fragment kodu:

```
SUB setUpGUI
! Pobieranie wartości z żyroskopu co 0,05 sekundy.
Sensors.setGyroRate(0.05)
```

Po uruchomieniu programu możesz zauważyć, że wykres jest obsługiwany w nieco odmienny sposób. W aplikacji odczytującej dane przyspieszeniomierza po narysowaniu wykresu do końca jego rysowanie ponownie zaczyna się od początku. Z kolei aplikacja żyroskopu nieustannie uaktualnia wykres, przesuwając istniejące punkty w lewą stronę i umieszczając nowe po prawej stronie. Takie rozwiązanie wymaga wprowadzenia dwóch zmian w programie. Pierwsza to usunięcie zmiennej index, ponieważ nie jest dłużej potrzebna. Drugą zmianą dotyczy podprocedury nullEvent, w której kod należy zmienić z:

```
ax(index, 2) = a(1)
ay(index, 2) = a(2)
az(index, 2) = a(3)
index = index + 1
IF index > 100 THEN index = 1
```

na następujący:

```
FOR i = 1 TO 99
  rx(i, 2) = rx(i + 1, 2)
  ry(i, 2) = ry(i + 1, 2)
  rz(i, 2) = rz(i + 1, 2)
NEXT
rx(100, 2) = r(1)
ry(100, 2) = r(2)
rz(100, 2) = r(3)
```

Możesz zadawać sobie pytanie, czy skopiowanie 297 wartości z jednego miejsca w tablicy do innego nie zabiera większej ilości czasu niż po prostu uaktualnienie jednej wartości i przejście dalej. Masz rację, to wymaga więcej czasu, ale dla nowoczesnego procesora tego rodzaju zmiana naprawdę nie powoduje widocznego pogorszenia wydajności działania programu. Operacja kopiowania wartości tablicy jest przeprowadzana bardzo szybko. Największy spadek wydajności jest związany z uaktualnianiem zawartości wyświetlanej na ekranie. To niewątpliwie należy wziąć pod uwagę, ale po uruchomieniu programu przekonasz się, że spadek wydajności jest niezauważalny.

Wiele opcji

W aplikacjach przyspieszeniomierza i żyroskopu uaktualnienia wykresu są obsługiwane na odmiennie sposoby. Który z nich uważasz za lepszy? Jeżeli preferujesz jeden z nich, wtedy bardzo łatwo możesz zmodyfikować program i otrzymać żądany sposób uaktualniania wykresu.

Ponieważ po prawej stronie wykresu program wyświetla wartość w aktualnej sekundzie, a następnie w lewą stronę umieszczane są dane z poprzednich dziesięciu sekund, zakres dla osi X został zmieniony z od 0 do 10 na od -10 do 0. Konieczne jest więc wprowadzenie

dwóch zmian w podprocedurze setUpGUI. Pierwsza zmiana dotyczy kodu odpowiedzialnego za początkowe przypisywanie wartości x w tablicach wykresu:

```
! Inicjalizacja tablic używanych przez wykres.  
FOR t = 1 TO 100  
  rx(t, 1) = t/10.0 - 10  
  ry(t, 1) = t/10.0 - 10  
  rz(t, 1) = t/10.0 - 10  
NEXT
```

Druga zmiana dotyczy widocznego zakresu wykresu po jego inicjalizacji. Zamiast wyświetlać wartości od 0 do 10 dla osi X, program wyświetla wartości od -10 do 0.

```
! Ustawienie zakresu wykresu i domeny. To trzeba zrobić  
! po dodaniu pierwszego obiektu PlotPoint, ponieważ ten krok  
! również powoduje ustawienie zakresu i domeny.  
p.setView(-10, -10, 0, 10, 0)
```

Jeżeli porównujesz kod źródłowy aplikacji Accelerometer i Gyroscope, to prawdopodobnie zauważyłeś, że zmiany są czysto kosmetyczne. Dotyczą nazw zmiennych, komentarzy i etykiet, co ma na celu odwoływanie się do żyroskopu, a nie przyśpieszoniomierza.

Poniżej przedstawiono pełny kod źródłowy aplikacji odczytującej dane z żyroskopu. Ten program o nazwie Gyroscope znajdziesz również w katalogu *O'Reilly Books* w aplikacjach techBASIC i techBASIC Sampler:

```
! Program wyświetla wykresy wartości pobranych z żyroskopu w ciągu  
! ostatnich 10 sekund w odstępach co 0,1 sekundy. Program oferuje funkcję rejestracji  
! pobieranych wartości i wysłania wyników za pomocą poczty elektronicznej.  
  
! Utworzenie wykresów i tablic przechowujących punkty wykresów.  
DIM p as Plot, px as PlotPoint, py as PlotPoint, pz as PlotPoint  
DIM rx(100, 2), ry(100, 2), rz(100, 2)  
  
! Utworzenie kontrolek.  
DIM quit AS Button, record AS Button, send AS Button  
  
! Utworzenie i inicjalizacja zmiennych globalnych używanych do śledzenia przebiegu działania programu.  
fileName$ = "tempdata.txt"  
recording = 0  
  
! Pobranie i ustawienie wartości początkowej czasu dla żyroskopu.  
DIM t0 AS DOUBLE  
IF Sensors.gyroAvailable THEN  
  WHILE t0 = 0  
    r = Sensors.gyro  
    t0 = r(4)  
  WEND  
END IF  
  
! Utworzenie interfejsu użytkownika.  
setUpGUI  
  
! Utworzenie nowego przycisku i wypełnienie go gradientem.  
!  
! Parametry:  
! x - położenie poziome.  
! y - położenie pionowe.  
! title - tekst wyświetlany przez przycisk.  
!  
! Wartość zwrotna: nowy przycisk.  
FUNCTION newButton (x, y, title AS STRING) AS Button  
DIM b AS Button
```

```

b = Graphics.newButton(x, y)
b.setTitle(title)
b.setBackgroundColor(1, 1, 1)
b.setGradientColor(0.6, 0.6, 0.6)
newButton = b
END FUNCTION

```

*! Ta podprocedura jest wywoływana, gdy nic innego się nie dzieje.
! Sprawdza, czy od ostatniego odczytania wartości żyroskopu
! upłynęło już co najmniej 0,1 sekundy. Jeśli tak, następuje wówczas
! pobranie i wyświetlenie nowych danych.*

! Parametry:

! time - godzina wystąpienia zdarzenia.

```

SUB nullEvent (time AS DOUBLE)

```

```

r = Sensors.gyro

```

```

IF recording AND (t0 <> r(4)) THEN

```

```

  PRINT #1, r(1); ", "; r(2); ", "; r(3); ", "; r(4)

```

```

END IF

```

```

IF r(4) > t0 + 0.1 THEN

```

```

  WHILE r(4) > t0 + 0.1

```

```

    t0 = t0 + 0.1

```

```

    FOR i = 1 TO 99

```

```

      rx(i, 2) = rx(i + 1, 2)

```

```

      ry(i, 2) = ry(i + 1, 2)

```

```

      rz(i, 2) = rz(i + 1, 2)

```

```

    NEXT

```

```

    rx(100, 2) = r(1)

```

```

    ry(100, 2) = r(2)

```

```

    rz(100, 2) = r(3)

```

```

  WEND

```

```

  px.setPoints(rx)

```

```

  py.setPoints(ry)

```

```

  pz.setPoints(rz)

```

```

END IF

```

```

END SUB

```

! Ostatnio zarejestrowane dane będą wysłane za pomocą wiadomości e-mail.

```

SUB sendData

```

```

DIM e AS eMail

```

```

e = System.newEMail

```

```

IF e.canSendMail THEN

```

```

  e.setSubject("Gyroscope data")

```

```

  e.setMessage("Gyroscope data")

```

```

  e.addAttachment(fileName$, "text/plain")

```

```

  e.send

```

```

ELSE

```

```

  button = Graphics.showAlert("Can't Send",
    "Email cannot be sent from this device.")

```

```

END IF

```

```

END SUB

```

! Konfiguracja interfejsu użytkownika.

```

SUB setUpGUI

```

! Pobieranie wartości z żyroskopu co 0,05 sekundy.

```

Sensors.setGyroRate(0.05)

```

! Inicjalizacja tablic używanych przez wykres.

```

FOR t = 1 TO 100

```

```

  rx(t, 1) = t/10.0 - 10

```

```
ry(t, 1) = t/10.0 - 10
rz(t, 1) = t/10.0 - 10
NEXT
```

! Inicjalizacja wykresu i jego wyświetlenie.

```
p = Graphics.newPlot
p.setTitle("Rotation in Radians per Second")
p.setXAxisLabel("Time in Seconds")
p.setYAxisLabel("Rotation: X: Green, Y: Red, Z: Blue")
p.showGrid(1)
p.setGridColor(0.8, 0.8, 0.8)
p.setAllowedGestures($0042)
```

```
px = p.newPlot(rx)
px.setColor(0, 1, 0)
px.setPointColor(0, 1, 0)
```

```
py = p.newPlot(ry)
py.setColor(1, 0, 0)
py.setPointColor(1, 0, 0)
```

```
pz = p.newPlot(rz)
pz.setColor(0, 0, 1)
pz.setPointColor(0, 0, 1)
```

*! Ustawienie zakresu wykresu i domeny. To trzeba zrobić
! po dodaniu pierwszego obiektu PlotPoint, ponieważ ten krok
! również powoduje ustawienie zakresu i domeny.*

```
p.setView(-10, -10, 0, 10, 0)
```

*! Wyświetlenie widoku przedstawiającego wykresy. Przekazanie wartości 1
! dla parametru powoduje przejście do trybu pełnego ekranu.*

```
system.showGraphics(1)
```

! Zablokowanie ekranu w jego bieżącej orientacji.

```
orientation = 1 << (System.orientation - 1)
System.setAllowedOrientations(orientation)
```

! Określenie wymiarów wykresu.

```
p.setRect(0, 0, Graphics.width, Graphics.height - 47)
```

! Wyświetlenie tła.

```
Graphics.setPixelGraphics(0)
Graphics.setColor(0.886, 0.886, 0.886)
Graphics.fillRect(0, 0, Graphics.width, Graphics.height)
```

! Konfiguracja interfejsu użytkownika.

```
h = Graphics.height - 47
quit = newButton(Graphics.width - 82, h, "Quit")
record = newButton(Graphics.width - 174, h, "Record")
send = newButton(Graphics.width - 266, h, "Send")
```

! Jeżeli nie ma żadnych danych do wysłania, wtedy należy wyłączyć przycisk Send.

```
IF NOT EXISTS(fileName$) THEN
  send.setEnabled(0)
END IF
```

*! Upewnienie się o dostępności żyroskopu. Jeżeli czujnik jest niedostępny,
! należy wyświetlić odpowiedni komunikat i zakończyć działanie programu.*

```
IF NOT Sensors.gyroAvailable THEN
  msg$ = "This device does not have a gyroscope. "
  msg$ = msg$ & "The program will exit."
```

```

    button = Graphics.showAlert("No Gyro", msg$)
    STOP
END IF
END SUB

! Podprocedura wywoływana, gdy program powinien rozpocząć
! rejestrację danych. Ta podprocedura zmienia nazwę przycisku
! na Stop, otwiera plik danych wyjściowych, a następnie ustawia
! flagę wskazującą na konieczność rejestracji danych pobieranych
! z żyroskopu.
SUB startRecording
record.setTitle("Stop")
recording = 1
OPEN fileName$ FOR OUTPUT AS #1
END SUB

! Podprocedura wywoływana, gdy program powinien zakończyć
! rejestrację danych. Ta podprocedura zmienia nazwę przycisku
! na Record, zeruje wartość zmiennej recording, a następnie zamyka
! plik danych wyjściowych.
!
! Tę podprocedurę można bezpiecznie wywołać, nawet
! jeśli nie zostały zarejestrowane żadne dane.
SUB stopRecording
IF recording THEN
    record.setTitle("Record")
    CLOSE #1
    recording = 0
    send.setEnabled(1)
END IF
END SUB

! Obsługa naciśnięcia dowolnego przycisku.
!
! Parametry:
! ctrl - przycisk, który został naciśnięty.
! time - godzina, o której wystąpiło zdarzenie.
SUB touchUpInside (ctrl AS Button, time AS DOUBLE)
IF ctrl = quit THEN
    stopRecording
    STOP
ELSE IF ctrl = record THEN
    IF recording THEN
        stopRecording
    ELSE
        startRecording
    END IF
ELSE IF ctrl = send THEN
    stopRecording
    sendData
END IF
END SUB

```

Radiany czy stopnie?

Osoby zajmujące się fizyką, inżynierowie i matematycy bardzo często posługują się radianami i nie mają żadnych problemów w używaniu naturalnej jednostki obrotu, jaką w przypadku żyroskopu są radiany na sekundę. Jeśli jednak Twoją reakcją jest „radi-co?“, to śpieszę wyjaśnić, że bardzo łatwo można zmodyfikować program, aby wartości obrotu były wyrażane w stopniach na sekundę zamiast w radianach na sekundę. Wartość odczytaną z czujnika

wystarczy pomnożyć przez $180/\pi$, co spowoduje jej konwersję z radianów na stopnie. W języku techBASIC znajduje się nawet użyteczna funkcja o nazwie DEG przeznaczona do wspomnianej konwersji. Najlepszym rozwiązaniem jest przeprowadzenie konwersji tuż po odczycianiu wartości z czujnika. W podprocedurze nullEvent umieść więc pętlę FOR:

```
SUB nullEvent (time AS DOUBLE)
r = Sensors.gyro

FOR i = 1 TO 3
  r(i) = DEG(r(i))
NEXT
```

Oczywiście zmianie ulegnie także zakres wartości. W podprocedurze setUpGUI wprowadź poniższą zmianę oznaczającą użycie wartości z zakresu ± 500 :

```
! Ustawienie zakresu wykresu i domeny. To trzeba zrobić
! po dodaniu pierwszego obiektu PlotPoint, ponieważ ten krok
! również powoduje ustawienie zakresu i domeny.
p.setView(-10, -500, 0, 500, 0)
```

Tytuł wykresu również należy zmienić, aby wskazywał użycie nowych jednostek pomiaru. Odpowiedni wiersz także znajduje się w podprocedurze setUpGUI:

```
! Inicjalizacja wykresu i jego wyświetlenie.
p = Graphics.newPlot
p.setTitle("Rotation in Degrees per Second")
```

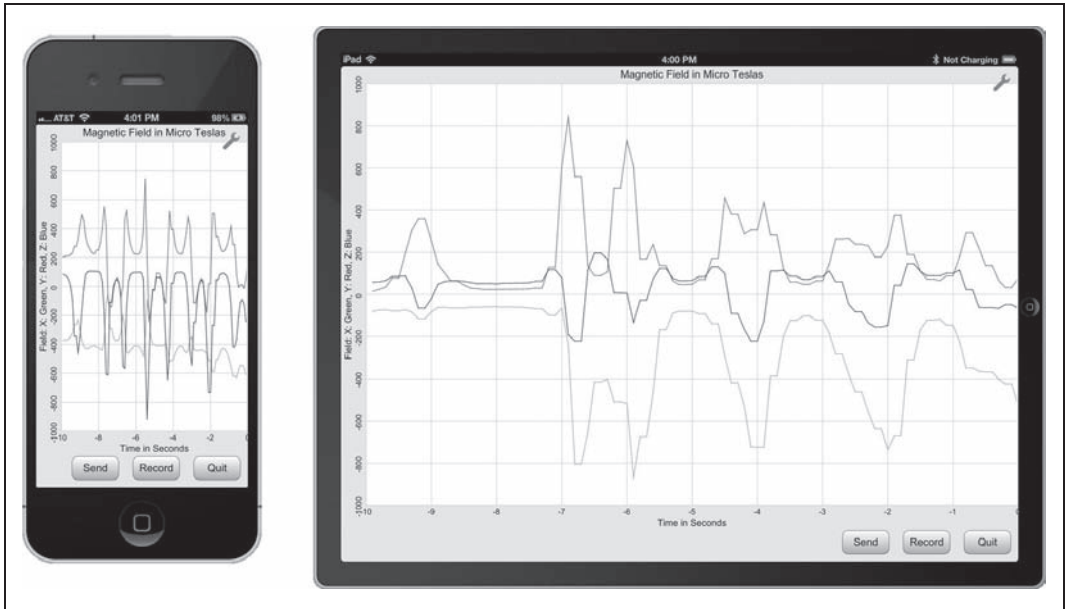
Magnetometr

Począwszy od modelu iPhone 3GS, wszystkie smartfony iPhone są wyposażone w magnetometr trójosiowy. Wymieniony czujnik jest wykorzystywany przede wszystkim jako kompas cyfrowy w aplikacjach opartych na mapach, choć potrafi zmierzyć także pole magnetyczne obecne w pobliżu urządzenia. Tę drugą możliwość wykorzystamy w rozdziale 3. do zbudowania prostego wykrywacza metalu. Natomiast w tym rozdziale utworzymy bardzo prostą aplikację pokazaną na rysunku 2.3. Magnetometru można użyć również do wyszukania przewodów oraz innych źródeł prądu lub po prostu można pobawić się za pomocą urządzenia iOS i magnesów przyczepianych do lodówki.

Oprogramowanie pobierające dane z magnetometru jest niemalże takie samo jak pobierające wartości z żyroskopu. Jak się domyślasz, między nimi istnieją pewne oczywiste różnice, na przykład odczyt danych z zupełnie innego czujnika, ale sama zasada działania jest podobna jak w przypadku programów odczytujących wartości z przyśpieszeniomierza i żyroskopu. W tabeli 2.1 wymieniono odpowiadające sobie polecenia w poszczególnych programach.

Poza tym istnieje jeszcze wiele innych oczywistych zmian kosmetycznych, takich jak dostosowanie nazw zmiennych, komentarzy i ciągów tekstowych, tak aby dotyczyły magnetometru.

Jedyna większa zmiana jest związana z ogromnymi wahaniami siły oddziaływania pól magnetycznych. Wartości maksymalne odczytywane z przyśpieszeniomierza sięgają $\pm 2G$ we wszystkich urządzeniach iOS poza iPhone'em 5, w którym maksymalny zakres wynosi $\pm 8G$. Zdefiniowanie skali pionowej jako $\pm 2G$ sprawdza się doskonale. W przypadku żyroskopu odczytywane wartości rzadko będą przekraczać 10 radianów na sekundę, a tym samym zakres ± 10 dla osi Y w aplikacji Gyroscope sprawdza się doskonale. Jednak magnetometr jest bardzo czuły i może zmierzyć siłę oddziaływania ziemskiego pola magnetycznego. Siła wspomnianego pola magnetycznego nie jest stała i ogólnie zawiera się w przedziale od 30 do 60 mikrotlesli ($30\text{--}60\ \mu\text{T}$).



Rysunek 2.3. Uruchomiona aplikacja Magnetometer

Tabela 2.1. Odpowiadające sobie polecenia w poszczególnych programach

Accelerometer	Gyroscope	Magnetometer
Sensors.accel	Sensors.gyro	Sensors.mag
Sensors.accelAvailable	Sensors.gyroAvailable	Sensors.magAvailable
Sensors.setAccelRate	Sensors.setGyroRate	Sensors.setMagRate

Czujnik może wykryć znacznie silniejsze pola magnetyczne, maksymalnie do 1T, czyli 1000000 μ T. Obsługa wartości w takim zakresie jest sporym wyzwaniem. Poniżej przedstawiono uaktualnioną wersję podprocedury `nullEvent`, która potrafi obsłużyć tak duży zakres wartości:

```

! Ta podprocedura jest wywoływana, gdy nic innego się nie dzieje.
! Sprawdza, czy od ostatniego odczytania wartości magnetometru
! upłynęło już co najmniej 0,1 sekundy. Jeśli tak, następuje wówczas
! pobranie i wyświetlenie nowych danych.
!
! Parametry:
! time - godzina wystąpienia zdarzenia.
SUB nullEvent (time AS DOUBLE)

! Pobranie nowych wartości.
m = Sensors.mag

! Jeżeli zachodzi potrzeba, wartość zostaje zapisana w pliku danych wyjściowych.
IF recording AND (t0 < m(4)) THEN
  PRINT #1, m(1); ", "; m(2); ", "; m(3); ", "; m(4)
END IF

! Uaktualnienie wykresu.
IF m(4) > t0 + 0.1 THEN
  ! Uaktualnienie tablic przechowujących punkty tworzące wykres.
  WHILE m(4) > t0 + 0.1

```



```

t0 = t0 + 0.1
FOR i = 1 TO 99
    mx(i, 2) = mx(i + 1, 2)
    my(i, 2) = my(i + 1, 2)
    mz(i, 2) = mz(i + 1, 2)
NEXT
mx(100, 2) = m(1)
my(100, 2) = m(2)
mz(100, 2) = m(3)
WEND

! Dostosowanie zakresu funkcji na podstawie maksymalnej zarejestrowanej wartości.
max = 0
FOR i = 1 TO 100
    IF ABS(mx(i, 2)) > max THEN max = ABS(mx(i, 2))
    IF ABS(my(i, 2)) > max THEN max = ABS(my(i, 2))
    IF ABS(mz(i, 2)) > max THEN max = ABS(mz(i, 2))
NEXT
range = 10^(INT(LOG(max)/LOG(10)) + 1)
p.setView(-10, -range, 0, range, 0)

! Uaktualnienie wykresów.
px.setPoints(mx)
py.setPoints(my)
pz.setPoints(mz)
END IF
END SUB

```

Zmiana znajduje się na końcu podprocedury, gdzie następuje dynamiczne dostosowanie zakresu. Program analizuje wartości, wyszukując największą. Następnie wykorzystuje pewną sztuczkę matematyczną, to znaczy na podstawie liczby całkowitej z logarytmu dziesiętnego dla największej wartości oblicza prawidłową potęgę dla zakresu pionowego. Przekonajmy się, jak takie rozwiązanie działa.

Zaczynamy od wartości odpowiadającej sile oddziaływania ziemskiego pola magnetycznego, czyli 50 μT . W takim przypadku wynik działania $\text{LOG}(50)/\text{LOG}(10)$ wynosi nieco poniżej 1,7. Wyodrębniamy z niego liczbę całkowitą i dodajemy jeden, otrzymując tym samym wartość 2 — to jest liczba zer, które muszą się znaleźć w zakresie. Podnosimy liczbę 10 do obliczonej potęgi (2) i otrzymujemy zakres ± 100 , doskonale sprawdzający się podczas wyświetlania wartości od 10 do 100. Wypróbuj kilka innych liczb, a przekonasz się, że zastosowany przez nas wzór zawsze zwraca w wyniku 100 dla zakresu, gdy max wynosi więcej niż 10, ale mniej niż 100.

W pobliżu urządzenia iPhone lub iPad umieść źródło względnie silnego pola magnetycznego, a siła takiego pola wzrośnie do kilkuset μT . Spróbuj ponownie wykonać działania matematyczne, a przekonasz się, że dla wartości max od 100 do 1000 zakresem będzie ± 1000 .

To niesamowity widok, gdy aplikacja automatycznie dostosowuje pionowy zakres wykresu w odpowiedzi na zmianę siły pola magnetycznego. Z tym wiąże się jednak pewna wada zastosowanego rozwiązania: ręczne dostosowanie zakresu nie jest możliwe. Wprawdzie można to zrobić, ale program automatycznie dostosuje zakres po pobraniu następnej wartości z czujnika. Jeśli takie rozwiązanie Ci nie odpowiada, to usuń kod, który automatycznie ustala zakres.

Poniżej przedstawiono pełny kod źródłowy aplikacji odczytującej dane z magnetometru. Ten program o nazwie Magnetometer znajdziesz również w katalogu *O'Reilly Books* w aplikacjach techBASIC i techBASIC Sampler:

```
! Program wyświetla wykresy wartości pobranych z magnetometru w ciągu  
! ostatnich 10 sekund w odstępach co 0,1 sekundy. Program oferuje funkcję rejestracji  
! pobieranych wartości i wysłania wyników za pomocą poczty elektronicznej.
```

```
! Utworzenie wykresów i tablic przechowujących punkty wykresów.  
DIM p as Plot, px as PlotPoint, py as PlotPoint, pz as PlotPoint  
DIM mx(100, 2), my(100, 2), mz(100, 2)
```

```
! Utworzenie kontrolek.  
DIM quit AS Button, record AS Button, send AS Button
```

```
! Utworzenie i inicjalizacja zmiennych globalnych używanych do śledzenia przebiegu działania programu.  
fileName$ = "tempdata.txt"  
recording = 0
```

```
! Pobranie i ustawienie wartości początkowej czasu dla magnetometru.  
DIM t0 AS DOUBLE  
IF Sensors.magAvailable THEN  
    WHILE t0 = 0  
        m = Sensors.mag  
        t0 = m(4)  
    WEND  
END IF
```

```
! Utworzenie interfejsu użytkownika.  
setUpGUI
```

```
! Utworzenie nowego przycisku i wypełnienie go gradientem.
```

```
!  
! Parametry:  
! x - położenie poziome.  
! y - położenie pionowe.  
! title - tekst wyświetlany przez przycisk.  
!
```

```
! Wartość zwrotna: nowy przycisk.  
FUNCTION newButton (x, y, title AS STRING) AS Button  
DIM b AS Button  
b = Graphics.newButton(x, y)  
b.setTitle(title)  
b.setBackgroundColor(1, 1, 1)  
b.setGradientColor(0.6, 0.6, 0.6)  
newButton = b  
END FUNCTION
```

```
! Ta podprocedura jest wywoływana, gdy nic innego się nie dzieje.  
! Sprawdza, czy od ostatniego odczytania wartości magnetometru  
! upłynęło już co najmniej 0,1 sekundy. Jeśli tak, następuje wówczas  
! pobranie i wyświetlenie nowych danych.
```

```
!  
! Parametry:  
! time - godzina wystąpienia zdarzenia.  
SUB nullEvent (time AS DOUBLE)
```

```
! Pobranie nowych wartości.  
m = Sensors.mag
```

```
! Jeżeli zachodzi potrzeba, wartość zostaje zapisana w pliku danych wyjściowych.  
IF recording AND (t0 <= m(4)) THEN  
    PRINT #1, m(1); ", "; m(2); ", "; m(3); ", "; m(4)  
END IF
```

```
! Uaktualnienie wykresu.  
IF m(4) > t0 + 0.1 THEN  
    ! Uaktualnienie tablic przechowujących punkty tworzące wykres.
```

```

WHILE m(4) > t0 + 0.1
  t0 = t0 + 0.1
  FOR i = 1 TO 99
    mx(i, 2) = mx(i + 1, 2)
    my(i, 2) = my(i + 1, 2)
    mz(i, 2) = mz(i + 1, 2)
  NEXT
  mx(100, 2) = m(1)
  my(100, 2) = m(2)
  mz(100, 2) = m(3)
WEND

```

! Dostosowanie zakresu funkcji na podstawie maksymalnej zarejestrowanej wartości.

```

max = 0
FOR i = 1 TO 100
  IF ABS(mx(i, 2)) > max THEN max = ABS(mx(i, 2))
  IF ABS(my(i, 2)) > max THEN max = ABS(my(i, 2))
  IF ABS(mz(i, 2)) > max THEN max = ABS(mz(i, 2))
NEXT
range = 10^(INT(LOG(max)/LOG(10)) + 1)
p.setView(-10, -range, 0, range, 0)

```

! Aktualnienie wykresów.

```

px.setPoints(mx)
py.setPoints(my)
pz.setPoints(mz)
END IF
END SUB

```

! Ostatnio zarejestrowane dane będą wysłane za pomocą wiadomości e-mail.

```

SUB sendData
DIM e AS eMail
e = System.newEMail
IF e.canSendMail THEN
  e.setSubject("Magnetometer data")
  e.setMessage("Magnetometer data")
  e.addAttachment(fileName$, "text/plain")
  e.send
ELSE
  button = Graphics.showAlert("Can't Send",
    "Email cannot be sent from this device.")
END IF
END SUB

```

! Konfiguracja interfejsu użytkownika.

```

SUB setUpGUI

```

! Pobieranie wartości z magnetometru co 0,05 sekundy.

```

Sensors.setMagRate(0.05)

```

! Inicjalizacja tablic używanych przez wykres.

```

FOR t = 1 TO 100
  mx(t, 1) = t/10.0 - 10
  my(t, 1) = t/10.0 - 10
  mz(t, 1) = t/10.0 - 10
NEXT

```

! Inicjalizacja wykresu i jego wyświetlenie.

```

p = Graphics.newPlot
p.setTitle("Magnetic Field in Micro Teslas")
p.setXAxisLabel("Time in Seconds")
p.setYAxisLabel("Field: X: Green, Y: Red, Z: Blue")
p.showGrid(1)

```

```
p.setGridColor(0.8, 0.8, 0.8)
p.setAllowedGestures($0042)
```

```
px = p.newPlot(mx)
px.setColor(0, 1, 0)
px.setPointColor(0, 1, 0)
```

```
py = p.newPlot(my)
py.setColor(1, 0, 0)
py.setPointColor(1, 0, 0)
```

```
pz = p.newPlot(mz)
pz.setColor(0, 0, 1)
pz.setPointColor(0, 0, 1)
```

```
! Ustawienie zakresu wykresu i domeny. To trzeba zrobić
! po dodaniu pierwszego obiektu PlotPoint, ponieważ ten krok
! również powoduje ustawienie zakresu i domeny.
p.setView(-10, -10, 0, 10, 0)
```

```
! Wyświetlenie widoku przedstawiającego wykresy. Przekazanie wartości 1
! dla parametru powoduje przejście do trybu pełnego ekranu.
system.showGraphics(1)
```

```
! Zablockowanie ekranu w jego bieżącej orientacji.
orientation = 1 << (System.orientation - 1)
System.setAllowedOrientations(orientation)
```

```
! Określenie wymiarów wykresu.
p.setRect(0, 0, Graphics.width, Graphics.height - 47)
```

```
! Wyświetlenie tła.
Graphics.setPixelGraphics(0)
Graphics.setColor(0.886, 0.886, 0.886)
Graphics.fillRect(0, 0, Graphics.width, Graphics.height)
```

```
! Konfiguracja interfejsu użytkownika.
h = Graphics.height - 47
quit = newButton(Graphics.width - 82, h, "Quit")
record = newButton(Graphics.width - 174, h, "Record")
send = newButton(Graphics.width - 266, h, "Send")
```

```
! Jeżeli nie ma żadnych danych do wysłania, wtedy należy wyłączyć przycisk Send.
IF NOT EXISTS(fileName$) THEN
    send.setEnabled(0)
END IF
```

```
! Upewnienie się o dostępności magnetometru. Jeżeli czujnik jest niedostępny,
! należy wyświetlić odpowiedni komunikat i zakończyć działanie programu.
IF NOT Sensors.magAvailable THEN
    msg$ = "This device does not have a magnetometer. "
    msg$ = msg$ & "The program will exit."
    button = Graphics.showAlert("No Magnetometer", msg$)
    STOP
END IF
END SUB
```

```
! Podprocedura wywoływana, gdy program powinien rozpocząć
! rejestrację danych. Ta podprocedura zmienia nazwę przycisku
! na Stop, otwiera plik danych wyjściowych, a następnie ustawia
! flagę wskazującą na konieczność rejestracji danych pobieranych
! z magnetometru.
SUB startRecording
```

```

record.setTitle("Stop")
recording = 1
OPEN fileName$ FOR OUTPUT AS #1
END SUB

! Podprocedura wywoływana, gdy program powinien zakończyć
! rejestrację danych. Ta podprocedura zmienia nazwę przycisku
! na Record, zeruje wartość zmiennej recording, a następnie zamyka
! plik danych wyjściowych.
!
! Tę podprocedurę można bezpiecznie wywołać nawet
! jeśli nie zostały zarejestrowane jakiegokolwiek dane.
SUB stopRecording
IF recording THEN
    record.setTitle("Record")
    CLOSE #1
    recording = 0
    send.setEnabled(1)
END IF
END SUB

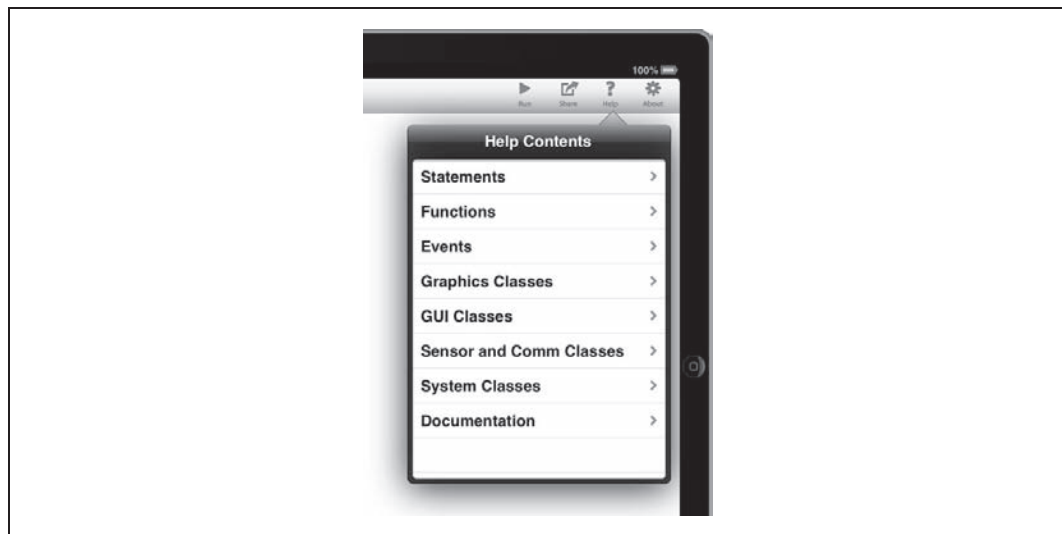
! Obsługa naciśnięcia dowolnego przycisku.
!
! Parametry:
! ctrl - przycisk, który został naciśnięty.
! time - godzina, o której wystąpiło zdarzenie.
SUB touchUpInside (ctrl AS Button, time AS DOUBLE)
IF ctrl = quit THEN
    stopRecording
    STOP
ELSE IF ctrl = record THEN
    IF recording THEN
        stopRecording
    ELSE
        startRecording
    END IF
ELSE IF ctrl = send THEN
    stopRecording
    sendData
END IF
END SUB

```

Uzyskanie szybszej odpowiedzi z czujnika

Zaprezentowane dotąd trzy programy pokazują, jak w zabawny i użyteczny sposób można pobrać niezmodyfikowane wartości z czujników przyspieszeniomierza, żyroskopu i magnetoimetru. Wspomniane programy mają jednak pewną wadę. Ponieważ uaktualnienie środowiska graficznego zabiera dużą ilość czasu, częstotliwość pobierania wartości z czujników jest dość ograniczona. Wprawdzie istnieje możliwość pozbycia się grafiki i jedynie odczytania danych z czujników, ale takie rozwiązanie, choć niewątpliwie zapewnia lepszą wydajność, również może wiązać się z utratą pewnych danych. Język techBASIC oferuje jeszcze inny sposób pobierania danych z czujników — wspomniane rozwiązanie pozwala na skrócenie czasu udzielania odpowiedzi przez czujniki. Pobieranie danych będzie odbywało się z maksymalną szybkością obsługiwaną przez iPhone, ale koszt takiego rozwiązania jest ogromny. Po wykonaniu zapytania program nie reaguje na działania użytkownika aż do chwili zebrania wszystkich danych z czujnika. Przeanalizujemy tego rodzaju rozwiązanie i przy okazji przekonajmy się, jakie są inne źródła pozwalające na jeszcze lepsze poznanie języka techBASIC.

techBASIC jest dostarczany wraz z wbudowanym systemem pomocy zawierającym informacje techniczne o każdym poleceniu, funkcji i klasie. Jeśli korzystasz z iPada, to naciśnij przycisk *Source* i spójrz na kod źródłowy programu, a następnie naciśnij przycisk *Help* znajdujący się na pasku narzędziowym. Z kolei w iPhone'ie musisz przejść do edycji programu, aby przycisk *Help* stał się widoczny. Po naciśnięciu przycisku *Help* na ekranie zostanie wyświetlona lista tematów pomocy pogrupowanych w odpowiednie kategorie (patrz rysunek 2.4).



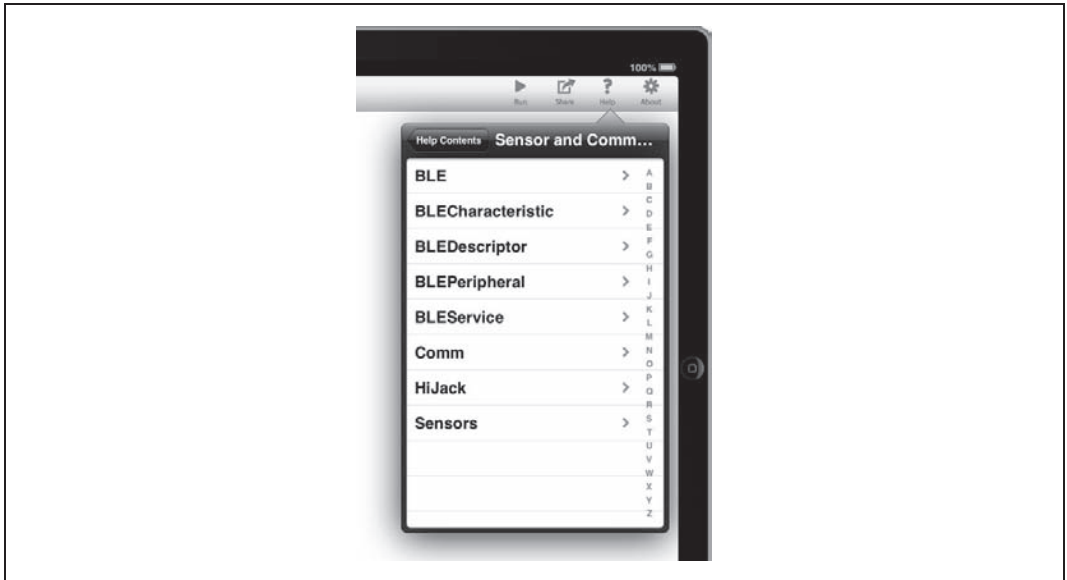
Rysunek 2.4. System pomocy wbudowany w aplikację techBASIC

Omówienie poleceń programu, na przykład polecenia `PRINT` użytego w pierwszym programie przedstawionym w tej książce, znajdziesz w kategorii *Statements*. Omówienie funkcji, na przykład `DEG` przeznaczonej do konwersji radianów na stopnie, znajdziesz w kategorii *Functions*. Z kolei wszystkie podprocedury wymieniono w kategorii *Events*, w której znajdziesz omówienie podprocedur specjalnych, takich jak `nullEvent` i `touchUpInside` wywoływanych przez techBASIC w programach opartych na zdarzeniach. W pozostałych kategoriach pogrupowano predefiniowane klasy.

Chcemy bliżej poznać klasę `Sensors` omówioną w kategorii *Sensor and Comm Classes* pokazanej na rysunku 2.5. Naciśnij tę nazwę. Zobaczysz nazwy różnych klas pozwalających na uzyskanie dostępu do czujników zarówno wewnętrznych, jak i zewnętrznych. Te klasy będziemy intensywnie wykorzystywać w książce, więc warto zapoznać się z poświęconą im sekcją systemu pomocy.

Ostatnia klasa wymieniona na liście to `Sensors`, której używaliśmy w przykładach przedstawionych w tym rozdziale. Naciśnij jej nazwę. Opisy poszczególnych klas rozpoczynają się od ogólnego przedstawienia danej klasy (patrz rysunek 2.6). Zauważysz także kilka metod, które powinny być Ci już znane. My jednak szukamy nowej. Przewiń listę w dół i naciśnij metodę o nazwie `sample`.

Na rysunku 2.7 pokazano okno zawierające pełny opis metody `sample`. Potrafi ona jednocześnie odczytywać wartości z wszystkich trzech czujników, co pozwala na pobieranie z nich danych z maksymalną szybkością dozwoloną przez system operacyjny. Wspomniany opis zawiera wszystkie szczegółowe informacje techniczne o wywołaniu. Znajdziesz tutaj wszystko, czego potrzebujesz do utworzenia programu używającego danego wywołania.

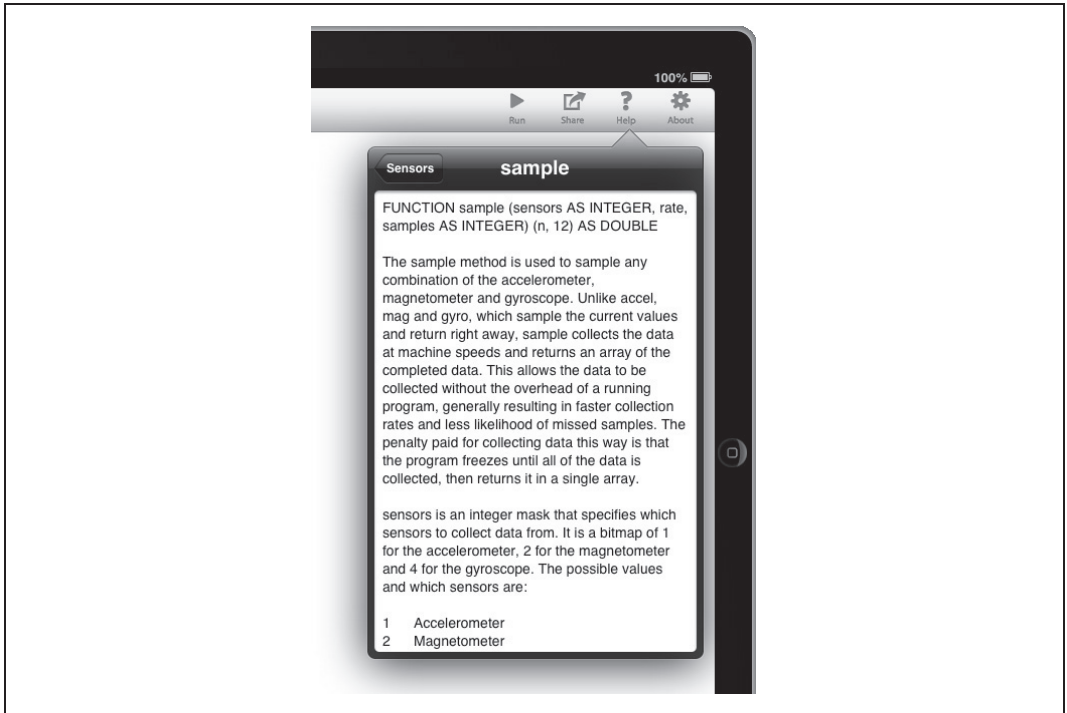


Rysunek 2.5. Kategoria Sensor and Comm Classes w systemie pomocy



Rysunek 2.6. Tematy pomocy dla klasy Sensors

W tym miejscu warto wspomnieć o źródle, które oferuje jeszcze więcej. System pomocy został zaprojektowany w taki sposób, aby można było szybko znaleźć potrzebne informacje i wyświetlić je na niewielkim ekranie. Nie zawiera zbyt wielu przykładowych kodów, a także jest pobawiony rysunków. Na szczęście dostępny jest podręcznik użytkownika techBASIC, w którym znajdziesz przykładowe fragmenty kodu i rysunki.



Rysunek 2.7. Opis metody *sample*



Podręcznik użytkownika dla techBASIC jest dostępny bezpłatnie na stronie Byte Works pod adresem http://www.byteworks.us/Byte_Works/Products.html. To dokument w formacie PDF, co pozwala na jego czytanie za pomocą aplikacji iBooks w urządzeniach iPad i iPhone.

Przeglądając fragment poświęcony metodzie *sample*, znajdziemy przedstawiony poniżej fragment kodu. Podobnie jak większość fragmentów kodu w podręczniku użytkownika to jest pełny, funkcjonujący program:

```
! Pobieranie danych z czujników przez 5 sekund.
samples = 250
samp = Sensors.sample(7, 0.02, samples)

! Określenie wartości średniej i maksymalnej
! dla każdego czujnika.
FOR i = 1 TO samples
  ax = ax + samp(i, 1)
  ay = ay + samp(i, 2)
  az = az + samp(i, 3)

  IF ABS(max) < ABS(samp(i, 1)) THEN max = samp(i, 1)
  IF ABS(may) < ABS(samp(i, 2)) THEN may = samp(i, 2)
  IF ABS(maz) < ABS(samp(i, 3)) THEN maz = samp(i, 3)

  gx = gx + samp(i, 5)
  gy = gy + samp(i, 6)
  gz = gz + samp(i, 7)
```



```

IF ABS(mgx) < ABS(samp(i, 5)) THEN mgx = samp(i, 5)
IF ABS(mgy) < ABS(samp(i, 6)) THEN mgy = samp(i, 6)
IF ABS(mgz) < ABS(samp(i, 7)) THEN mgz = samp(i, 7)

mx = mx + samp(i, 9)
my = my + samp(i, 10)
mz = mz + samp(i, 11)

IF ABS(mmx) < ABS(samp(i, 9)) THEN mmx = samp(i, 9)
IF ABS(mmy) < ABS(samp(i, 10)) THEN mmy = samp(i, 10)
IF ABS(mmz) < ABS(samp(i, 11)) THEN mmz = samp(i, 11)
NEXT

PRINT USING "Max acceleration: ##.##, ##.##, ##.##"; max, may, maz
PRINT USING "Max rotation: ###.##, ###.##, ###.##"; mgx, mgy, mgz
PRINT USING "Max magnetic field: ###.##, ###.##, ###.##"; mmx, mmy, mmz

PRINT USING "Average acceleration: ##.##, ##.##, ##.##"; _
ax/samples, ay/samples, az/samples
PRINT USING "Average rotation: ###.##, ###.##, ###.##"; _
gx/samples, gy/samples, gz/samples
PRINT USING "Average magnetic field: ###.##, ###.##, ###.##"; _
mx/samples, my/samples, mz/samples

```

Choć niezbyt ładny wizualnie, to jest kompletny program pobierający z trzech omówionych dotąd czujników wartości z częstotliwością 50 razy na sekundę. Dane są zbierane przez pięć sekund. Następnie program oblicza i wyświetla wartości średnie i maksymalne.

Ten program nie jest jednym z przykładów zawartych w aplikacji techBASIC. W jaki sposób można go wypróbować w techBASIC? Niestety, firma Apple nałożyła przesadnie ogromne restrykcje mające na celu ochronę użytkowników iOS. Dlatego choć wydaje się to głupie, istnieją tylko dwa sposoby na wypróbowanie wspomnianego programu w techBASIC. Pierwszy wymaga jego samodzielnego wpisania. Drugi polega na wysłaniu kodu źródłowego w wiadomości e-mail, a następnie skopiowaniu tego kodu do nowego, pustego programu. Mimo wszystko użycie poczty elektronicznej nie jest najgorszym rozwiązaniem.

Wypróbuj ten program i przekonaj się, jak działa. Masz już wystarczająco dużą wiedzę, aby go zmodyfikować i rozbudować, jeśli zajdzie potrzeba.

Wyznaczanie kursu

W klasie `Sensor` po wyświetleniu opisu metody `sample` być może dostrzegłeś dwie inne usługi. Pierwsza, o nazwie `heading`, jest tak naprawdę zbiorem przetworzonych informacji pobranych z magnetometru, przyspieszeniomierza i GPS. Zadanie, jakim jest wyszukanie kierunku na podstawie wspomnianych informacji, nie należy do łatwych, o czym się przekonasz w kolejnym rozdziale. Teraz wystarczy wiedzieć, że wyznaczenie kursu jest całkiem łatwe. Podobnie jak w przypadku trzech używanych dotąd czujników wszystko sprowadza się do odczytania wartości, prawdopodobnie po użyciu wywołania `headingAvailable` pozwalającego na sprawdzenie dostępności omawianej usługi w danym urządzeniu iOS. Z tym wiąże się jednak pewien problem. Pierwsze wywołanie `heading` włącza usługę, co wymaga chwili czasu. Zamiast wstrzymać działanie programu na czas włączania usługi, techBASIC zwraca ostatnio pobraną wartość. Należy pobrać kilka wartości, a nie tylko jedną:

```

FOR I = 1 TO 10
  PRINT Sensors.heading
  System.wait(0.5)
NEXT

```

Ten program dziesięciokrotnie odczytuje wartości, czekając pół sekundy między kolejnymi odczytami. Po jego uruchomieniu w moim iPadzie otrzymałem następujące wartości (puste wiersze zostały usunięte):

0	0	-1	22182.984556
90.786407	81.723602	-1	22183.473079
92.786407	83.723602	-1	22183.965387
38.786407	29.723602	40	22184.478982
34.786407	25.723602	40	22184.987621
34.786407	25.723602	40	22185.500116
33.786407	24.723602	40	22185.989068
33.786407	24.723602	40	22186.506593
33.786407	24.723602	40	22186.99292
33.786407	24.723602	40	22187.501888

Jak możesz się przekonać, wartości pochodzące z pierwszego odczytu są bezużyteczne.

Począwszy od wiersza drugiego, dwie pierwsze liczby to poprawne wartości kursu, podobnie jak w kompasie. Powyższe dane wyjściowe pokazują, że iPad był zwrócony w kierunku wschodnim, a następnie przed czwartym odczytem obrócony nieco w kierunku północno-wschodnim.

W rzeczywistości w ogóle nie dotykałem iPada. To tłumaczy potrzebę zastosowania trzeciej liczby w danych wyjściowych. Wspomniana liczba wskazuje dokładność odczytu, dla pierwszych trzech wynosi -1 i oznacza, że wartości w ogóle nie są dokładne, oraz przypomina nam o konieczności odczekania chwili, zanim urządzenie dostarczy prawidłowe dane. Nawet po osiągnięciu pełnej gotowości do pracy i podawania prawidłowych wartości dokładność nie jest zbyt dobra. System iOS wskazuje, że kurs jest podawany z dokładnością do 40 stopni.

Ostatnia wartość w danych wyjściowych wskazuje godzinę, o której nastąpił odczyt wartości.

Położenie

Ostatnia usługa w klasie Sensors to usługa podająca informacje o położeniu. Czy dane pochodzą z nowego czujnika, czy powstają na podstawie informacji pobranych z zewnętrznych czujników tworzących system satelitarnego GPS? Myślę, że to połączenie obu wspomnianych rozwiązań. W każdym bądź razie ta usługa jest używana podobnie jak usługa przeznaczona do wyznaczania kursu, a ponadto również potrzebuje nieco czasu, zanim osiągnie pełną gotowość do pracy i podawania prawidłowych wartości. Poniżej przedstawiono krótki program pokazujący, jak odczytywać informacje o położeniu. Ten fragment kodu został skopiowany bezpośrednio ze znajdującego się w podręczniku użytkownika techBASIC fragmentu prezentującego polecenie location:

```

location = sensors.location(30)
PRINT USING "Latitude   : ####.###"; location(1)
PRINT USING "Longitude  : ####.###"; location(2)
PRINT USING "Altitude   : #####.##"; location(3)
PRINT USING "Horiz. Error: ####.###"; location(4)
PRINT USING "Vert. Error : ####.###"; location(5)
PRINT USING "Speed      : ####.###"; location(6)
PRINT USING "Direction  : #####"; location(7)
PRINT      "Time stamp  : "; location(8)

```

Po uruchomieniu powyższego kodu w moim urządzeniu otrzymałem dane o położeniu, które okazały się całkiem dokładne! W poniższych danych wyjściowych usunąłem pewne liczby pozwalające na dokładne określenie mojego położenia.

```
Latitude   : 35.xxx  
Longitude  : -106.xxx  
Altitude   : 1514.xx  
Horiz. Error: 65.000  
Vert. Error : 44.790  
Speed      : -1.000  
Direction  : -1  
Time stamp : 24355.097656
```

Zrozumienie pierwszych pięciu wartości nie powinno nastręczać trudności, jednostką miary dla odległości jest metr.

Szybkość (Speed) i kierunek (Direction) są obliczane na podstawie serii informacji o położeniu. Jeżeli te wartości nie mogą być obliczone, jak ma to miejsce w omawianym przykładzie, ponieważ iPad leży nieruchomo na biurku, wtedy wyświetlane są wartości -1. Kierunek nie jest kursem według kompasu, ale rzeczywistym kierunkiem ruchu.

Na końcu danych wyjściowych znajduje się znacznik czasu wskazujący godzinę, o której pobrano dane wartości.

Twój własny tricorder

W ten sposób otrzymałeś własny tricorder. Za jego pomocą możesz wyświetlać, rejestrować oraz wysyłać przez pocztę e-mail dane dotyczące przyspieszenia, obrotu i siły pola magnetycznego. Dowiedziałeś się również, jak dzięki kilku dodatkowym wywołaniom uzyskać dostęp do informacji o kursie, położeniu, kierunku ruchu oraz szybkości.

W rozdziale 3. wykorzystamy zdobytą dotąd wiedzę do opracowania wykrywacza metalu opartego na magnetometrze. Nieco później, w rozdziale 7., powrócimy do przyspieszeniomierza, choć wtedy przyspieszeniemierzem będzie zdalne urządzenie. Przekonasz się, jak na podstawie danych przyspieszeniomierza określić szybkość i pokonaną odległość, a także jak użyć przyspieszeniomierza do określenia pułapu dla modelu rakiety.

\$0001, 234
\$0003, 234
\$0006, 234
\$0012, 234
\$00FFFF, 126
\$F4, 228
0xAA01, 151
0xAA02, 150

A

AA11, 117
AA12, 113, 117
AA13, 113, 118
AA21, 146
AA22, 146
AA31, 139
AA32, 139
AA33, 140
AA42, 124
AA43, 125
AA51, 133
accel, 17
Accel, 18
 kod źródłowy programu, 19
Accelerometer, 17, 20, 21
 interfejs użytkownika
 konfiguracja, 24
 przygotowanie, 22
 tworzenie przycisku, 26
 zdefiniowanie funkcji, 27
 kod źródłowy programu, 21
 licznik czasu, 22
 obiekt wiadomości e-mail, 30
 obsługa zdarzeń, 27
 pobieranie i obsługa danych
 z przyspieszeniomierza, 30
 polecenia, 42
 Quit, 28
 Record, 28
 rejestracja danych, 28
 Send, 28, 30
 Stop, 28
 wykresy
 inicjalizacja, 24
 inicjalizacja tablic, 24
 obiekt definiujący osie, 25
 uaktualnienie, 31
 ustawienie zakresu, 25
 wielkość, 26
 wyświetlanie tła, 26
 wyświetlenie widoku, 25
 zablokowanie ekranu w
 bieżącej orientacji, 26
 zakończenie działania, 22
 zapisywanie danych w
 pliku, 28
 zmiennie, 22
 znacznik czasu, 22
addPlant, 93
adres IP, 264
advertise, 233, 236
advertisement, 100
AGND, 69
AK8973, 56, 63
Alldatasheet, 56
analiza danych, 183
 plik CSV, 184
 plik danych, 183
 tag, 184
ANGLE, 226
aplikacja
 Gyroscope, 35
 Magnetometer, 42
 kod źródłowy, 43
 konwersja na
 wykrywacz metalu, 60
rzeczywistości
 rozszerzonej, 135
SensorTag, 100, 103
techBasic, 65
 dostęp do urządzeń, 66
techBasic Sampler, 15, 65
żyroskop, 34
Apps, 86
Arduino
 dioda LED, 274
 Firmata, 211
 przeniesienie
 oprogramowania, 213
 instalacja oprogramowania,
 211
 Serial Port, 214
 sterowanie, 211
 środowisko
 uruchomieniowe, 211
 Wi-Fi, 271
 WiFly, 273
 WiFly Terminal, 274
Arduino Uno, 195, 208
ASCII, 178
atrybut
 0xAA01, 151
 0xAA02, 150
 AA11, 117
 AA12, 117
 AA13, 118
 AA21, 146
 AA22, 146
 AA31, 139
 AA32, 139

- atrybut
 - AA33, 140
 - AA42, 124
 - AA43, 125
 - AA51, 133
- Augmented Reality, 33, 135
- B**
- bajty, 114
 - \$90, 228
 - drugi, 125
 - mapowanie bitowe, 133
 - operacje na bajtach, 125
 - pierwszy, 125
 - użycie, 133
 - w języku BASIC, 249
 - zerowy, 133
- ballX, 258
- ballY, 258
- bar, 123
- barometr, 101
 - identyfikator UUID, 105
 - model rakiety, 171, 175
 - obsługa, 177
 - SensorTag, 123
 - atrybut AA42, 124
 - atrybut AA43, 125
 - BLECharacteristicInfo, 125, 126
 - dane kalibracji, 125
 - konwersja danych
 - temperatury
 - i ciśnienia, 127
 - odczyt danych, 126
 - pełny kod źródłowy, 128
 - profil GATT, 124
 - readCharacteristic, 126
 - setNotify, 126
 - uzyskanie dostępu, 124
 - wartości kalibracji, 127
 - włączenie czujnika, 124
 - wyłączenie, 125
- BASIC
 - pliki, 29
- biblioteka
 - BLEFirmata, 214
- BITAND \$007F, 285
- BITAND \$00FFFF, 126
- BITOR, 125, 228
- bity, 114
 - \$0080, 114
 - drugi, 133
 - mapowanie bitowe, 133
 - najbardziej znaczące, 114
 - najmniej znaczące, 133
 - operacje na bitach, 125
 - użycie, 133
- BLE, 67, 99
 - przeznaczenie, 100
 - SensorTag, 101
 - tryb podległy, 231
- BLE Chat, 232
 - advertise, 233, 236
 - BLECharacteristicInfo, 238
 - BLEChatA, 235
 - BLEChatB, 235
 - BLEDiscoveredPeripheral, 236
 - BLEMutableCharacteristic
 - ↳Info, 239
 - BLEPeripheralInfo, 237
 - identyfikatory UUID, 235
 - interfejs użytkownika, 232, 236, 240
 - pole tekstowe, 240
 - isA, 235
 - konfiguracja urządzeń, 232
 - local, 235
 - localReadyCharacteristic, 241, 242
 - obejście ograniczenia
 - przesyłu danych, 234, 239
 - obsługa połączenia
 - z urządzeniem, 237
 - pakiety danych, 239
 - readyToUpdateSubscribers, 242
 - receiveStatus, 239
 - remote, 235
 - remoteReadyCharacteristic, 239
 - sendText, 241, 242
 - setUpGUI, 240
 - touchUpInside, 243
 - umieszczenie w tablicy
 - wartości maksymalnie 20 bajtów, 241
 - updateValue, 242
 - usługa komunikacji
 - i rozgłaszania, 233
 - użycie usług, 234
 - valueChanged, 241
 - wysyłanie powiadomień, 238
 - wyszukiwanie urządzenia, 236
 - zakończenie działania programu, 243
- BLE Truck, 195
- BLE.connect, 106, 110
- BLE.startBLE, 105
- BLE.stopScan, 106
- BLECharacteristicInfo, 113, 117, 125, 126, 134, 140, 146, 151, 176, 238, 255
- BLEDiscoveredPeripheral, 105, 108, 172, 236
- BLEFirmata, 214
- BLEMutableCharacteristic, 248
- BLEMutableCharacteristicInfo, 239
- BLEMutableService, 233
- BLEMutableService.new
 - ↳Characteristic, 234
- BLEPeripheralInfo, 109, 110, 173, 224, 237, 254
 - characteristics, 112
- BLEPeripheralManager, 233
- BLEServiceInfo, 111, 174, 228, 255
- Bluetooth, 100
 - a BLE, 100
 - dostęp, 100
- Bluetooth 4.0, 99
- Bluetooth LE, 99
- Bluetooth Low Energy, 7, 67, 99, 100
- Bluetooth Smart, 99
- błąd systematyczny, 188
- Break Away Header, 266
- Button, 22
- BYTE, 282
- Byte Works, 50
- C**
- CC Debugger, 102, 162, 180
- cechy charakterystyczne, 104, 111, 233
 - 0xAA52, 133
 - AA12, 113
 - AA13, 113

localReadyCharacteristic, 233
 localTextCharacteristic, 233
 obsługa powiadomień, 113
 pojedynczy odczyt wartości, 113
 przyśpieszoniemierz, 113
 transmisji, 219
 txUUID, 219
 characteristics, 112
 ciśnienie, 123, 127, 189
 atmosferyczne
 czynniki, 124
 wykres, 190
 wyniki pomiarów, 189
 clearConsole, 75
 CLOSE, 29
 CLOSE #1, 29
 color, 94
 Comm.openTCPIP, 268
 Comma Separated Values, 31, 84
 CSV, 31, 85
 ctrl.value, 284
 czujniki
 Digital Humidity Sensor, 145
 Hijack, 67
 SHT21, 145
 T5400, 123
 urządzenia SensorTag, 101
 dostęp, 103
 układ współrzędnych, 116
 wilgotności, 83, 101
 czujniki wbudowane, 13
 magnetometr, 41
 określanie wartości
 średnich i maksymalnych, 50
 przyśpieszoniemierz, 17
 system współrzędnych, 19, 20
 uzyskanie dostępu, 48
 do innych czujników, 33
 uzyskiwanie szybszej
 odpowiedzi, 47
 zużycie energii, 24
 żyroskop trójosiowy, 34

D

DEG, 41, 226
 deklinacja, 141
 dodatnia, 57
 magnetyczna, 63

ujemna, 57
 ziemskiego pola
 magnetycznego, 57
 Delete, 86
 demontaż samochodu, 196
 deskryptory, 104
 deviceID, 284
 Devices, 86
 dewiacja
 kompasu, 63
 magnetyczna, 57
 DFRobot, 280
 die temperature, 150
 Digital Barometric Pressure
 Sensor, 123
 Digital Humidity Sensor, 145
 DIM, 21, 27, 217, 282
 DIM sensorTag AS
 BLEPeripheral, 109
 DIP, 202
 Direction, 53
 discoverCharacteristics, 110, 112
 Discovered SensorTag, 107
 discoverServices, 110
 Dismiss Keyboard, 19
 dodatnie napięcie układu, 69
 dokument
 typu Quick Start, 183
 dokumentacja
 techBASIC, 24
 dozownik M&M, 286
 drawArrow, 222
 Dual Inline Package, 202
 dźwignia
 sterowania, 285
 skrętu, 277

E

Edit, 17
 efekt Halla, 56, 63
 elementy
 TextView, 93
 End Of File, 269
 EOF, 269
 Estes Loadstar II Kit, 160, 169
 etykiety, 91
 receiveStatus, 239
 Status, 178
 Events, 48

F

File Transfer Protocol, 265
 filename\$, 22
 fillRect, 91
 Find, 24
 Firmata, 211
 dostępność, 211
 informacje, 213
 instalacja, 213
 piny cyfrowego wejścia-
 -wyjścia, 225
 pobranie, 213
 poznawanie
 oprogramowania, 228
 wersja oprogramowania, 213
 fizyczny biegun północny, 56
 Flash Programmer, 157, 162, 180
 FOR, 41
 format
 CSV, 31
 FORTRAN, 74
 FTP, 264
 Functions, 48
 funkcja
 ANGLE, 226
 DEG, 41, 226
 Hijack.receive, 74
 pomocnicza getFileName,
 178

G

generator liczb losowych, 235
 Generic Attribute Profile, 116
 GET, 269
 getFileName, 178
 GND, 69, 70
 GPS, 33
 gra
 Paddles, 245, 246
 Pong, 245
 Graphics, 78
 Graphics.newLabel, 92
 Graphics.setPixelGraphics(0),
 178
 Graphics.setToolsHidden, 92
 Grove, 68, 83
 Gyroscope, 35, 37
 polecenia, 42

H

- haveConnection, 219, 224
 - H-Bridge, 193
 - heading, 33, 51
 - headingAvailable, 51
 - Hello HiJack, 73
 - działanie, 73
 - funkcja HiJack.receive, 74
 - pętla WHILE, 73
 - pętla WHILE - WEND, 73
 - polecenie PRINT, 74
 - System.clearConsole, 74
 - System.wait(0.5), 74
 - HiJack, 65
 - AGND, 69
 - budowa czujnika, 67
 - plytka uniwersalna, 68
 - plytka ze złączem USB, 68
 - potencjometr, 68, 69
 - rezystor o zmiennej oporności, 68
 - schemat układu, 69
 - zasilacz, 68
 - clearConsole, 75
 - dane wyjściowe, 76
 - dotąd napięcie układu, 69
 - dostarczanie danych, 79
 - gniazdo żeńskie, 69
 - maksymalne napięcie, 72
 - masa, 69
 - oprogramowanie, 65
 - piny w gniazdach wyjściowych, 70
 - problemy, 75
 - zasilanie zewnętrzne, 75
 - program HiJack, 76
 - DIM, 78
 - lepsza wersja, 75
 - Plot, 78
 - PlotPoint, 78
 - setGrid, 78
 - setTitle, 78
 - showGrid, 78
 - tablica, 77
 - tworzenie wykresu, 78
 - program MFi, 65, 66
 - przygotowania, 65
 - ujemne napięcie układu, 69
 - WHILE, 79
 - wilgotnościomierz, 81
 - wyposażenie, 65
 - zasilanie zewnętrzne, 71
 - bateria 2032, 71
 - lista wymaganych elementów, 71
 - plytka uniwersalna, 71
 - przełącznik, 71
 - rezystory, 71
 - uchwyt na baterię, 71
 - zestaw konstrukcyjny, 68
 - zwracane wartości, 78
 - źródło zasilania, 81
- HiJack Moisture Meter, 90
 - addPlant, 93
 - elementy TextView, 93
 - etykiety, 91
 - newLabel, 92
 - newTextView, 94
 - nullEvent, 95
 - odczytanie i wyświetlenie wartości z czujnika, 95
 - pasek postępu, 95
 - pełny kod źródłowy, 96
 - przycisk zatrzymania programu, 91
 - obsługa kliknięcia, 91
 - tło ekranu, 91
 - touchUpInside, 91
 - wielkość ekranu graficznego, 90
 - HiJack.receive, 74, 95
 - HTTP, 264
 - Hypertext Transfer Protocol, 264
- ## I
- IAP, 15
 - iBooks, 24
 - identyfikator UUID, 105, 116, 171
 - długość, 235
 - gra Paddles, 251
 - lista, 219
 - redBearUUID, 219
 - IEEE 802.11, 264
 - IF, 28, 35
 - ikona narzędzia menu, 23
 - IMU-3000, 132
 - in-app purchase, 15
 - included services, 104
 - index, 22
 - inklinacja magnetyczna, 141
 - inklinometr, 141
 - INPUT, 29
 - Internet Protocol, 264
 - iPad
 - Help, 48
 - HiJack, 65, 81
 - ikona narzędzia, 23
 - iOS 5, 65
 - lista podprocedur, 22
 - magnetometr, 55
 - napięcie w gnieździe słuchawek, 75
 - New, 73
 - orientacja, 19
 - Paddles, 245
 - Programs, 15
 - Run, 19
 - serwomechanizmy, 277
 - Source, 48, 73
 - Stop, 23
 - Subs, 22, 24
 - Wi-Fi, 264, 267
 - wykrywacz metalu, 58
 - wyznaczanie kursu, 52
 - zdalnie sterowany samochód, 194, 218
 - iPad 3, 160
 - iPhone, 160
 - Dismiss Keyboard, 19
 - Find, 24
 - Help, 48
 - HiJack, 65, 81
 - lista programów, 15
 - tematów pomocy, 48
 - magnetometr, 55, 56
 - magnetometr trójosiowy, 41
 - model rakiety, 157
 - napięcie w gnieździe słuchawek, 71, 75
 - New, 73
 - Paddles, 245
 - pole magnetyczne, 58
 - Programs, 15, 74
 - serwomechanizmy, 277

- Source, 73
 - układ AK8973, 56
 - Wi-Fi, 264, 267
 - wykresy
 - odchylenie pola magnetycznego, 59
 - wykrywacz metalu, 58
 - zdalnie sterowany samochód, 194, 218
 - zmiana siły pola magnetycznego, 62
 - żyroskop, 35
 - iPod touch, 14, 160
 - Hijack, 65
 - serwomechanizmy, 277
 - isA, 235
 - iTunes, 85
 - Apps, 86
 - Delete, 86
 - Devices, 86
 - Library, 86
- J**
- Java
 - bezpieczeństwo, 211
 - dostępność, 211
- K**
- kalibracja, 83
 - stałe, 127
 - użycie danych, 86
 - zebranie danych, 83
 - kierunek, 53
 - kind, 110, 112, 173, 237, 254
 - klasy
 - BLEMutableService, 233
 - BLEPeripheralManager, 233
 - Plot, 78
 - PlotPoint, 78
 - Sensors, 17, 48
 - klient FTP, 265
 - kod uzupełnień do dwóch, 114
 - komentarz, 21
 - komora ładunkowa, 163
 - kompas, 135
 - komponenty
 - Grove, 68
 - komunikacja
 - BLE, 232
 - między komputerami, 264
 - dwukierunkowa, 265
 - sieciowa
 - protokoły HTTP, FTP i TCP/IP, 264
 - schemat adresowania, 264
 - z dwoma urządzeniami
 - typu BLE, 251
 - z portem, 265
 - z TCP/IP, 268
 - za pomocą programu terminala, 274
 - ze światem, 263
 - koniec pliku, 269
 - kontrolery
 - Pololu Serial Servo Controller, 275, 276, 277
 - konstrukcja, 278
 - Mini SSC II Mode, 281
 - zestaw poleceń Pololu, 281
 - serwera, 280
 - serwomechanizmu
 - logic-level serial input, 280
 - położenie
 - serwomechanizmu, 284
 - port szeregowy RS-232, 280
 - poruszanie
 - serwomechanizmem, 284
 - tryb awaryjny, 280, 281
 - zasilacz DFRobot, 280
 - złącza, 280
 - kontrolka
 - typu Picker, 185
 - konwersja
 - ciśnienia, 127
 - temperatury, 127, 151
 - wartości kąta, 226
 - wilgotności, 146
 - konwerter A-D, 67
- L**
- lastTime, 219
 - libraries, 213
 - Library, 86
 - licznik czasu, 22
 - LINE INPUT, 269, 270
 - line\$, 234, 241, 242
- lista**
- cech charakterystycznych, 112
 - podprocedur, 22, 23
 - programów, 15, 17
 - My Programs, 18
 - tematów pomocy, 48
 - Events, 48
 - Functions, 48
 - Sensor and Comm Classes, 48
 - Statements, 48
 - usług standardowych, 105
- local**, 235
- localReadyCharacteristic, 233, 241, 242
 - localTextCharacteristic, 233
 - location, 52
 - logic-level serial input, 280
 - lot, 182
 - pogoda, 182
 - silniki, 182
 - B6-4, 182
 - C6-5, 182
 - D12-3, 182
 - E9-4, 182
 - spadochrony, 182
 - start rakiety, 183
- M**
- MAG3110, 138
 - Magnetometer, 42
 - konwersja na wykrywacz metalu, 60
 - nullEvent, 60
 - ogólna siła pola magnetycznego, 60
 - pełny kod źródłowy, 43
 - polecenia, 42
 - utworzenie wykresu i tablic, 60
 - magnetometr, 41, 55, 62, 138
 - dynamiczne dostosowanie zakresu, 43
 - efekty działania, 59
 - nullEvent, 42
 - pełny kod źródłowy aplikacji, 43

- magnetometr
 - SensorTag, 138
 - atrybut AA31, 139
 - atrybut AA32, 139
 - atrybut AA33, 140
 - BLECharacteristicInfo, 140
 - dane wyjściowe, 141
 - kalibracja, 141
 - odczyt danych, 139
 - pełny kod źródłowy, 141
 - profil GATT, 139
 - readCharacteristic, 140
 - setNotify, 140
 - system powiadomień, 139
 - uzyskanie dostępu, 139
 - użycie, 141
 - włączanie, 139
 - wyświetlanie pobranych wartości, 140
 - trójosiowy, 101
 - w urządzeniach iPhone i iPad, 55
 - zapis danych w pliku, 60
 - zasada działania, 56
- magnetyczny biegun północny, 56
- mapowanie bitowe, 133
- masa, 69
- Maski na Halloween, 281
 - oprogramowanie, 281
- master, 100
- Math.poly, 88
- maxState, 220
- Metal Detector, 61
- metody
 - accel, 17
 - BLECharacteristicInfo, 117
 - characteristics, 112
 - discoverCharacteristics, 110, 112
 - discoverServices, 110
 - fillRect, 91
 - Graphics.setToolsHidden, 92
 - Math.poly, 88
 - newPlot, 78
 - PlotPoint.newPlot, 88
 - readCharacteristic, 113, 118
 - sample, 48, 50
 - setColor, 91
 - setGrid, 78
 - setPoints, 31
 - setTitle, 78
 - showGrid, 78
 - startScan, 105
 - writeCharacteristic, 117
- MFi, 65, 66, 100
- mikrokontroler Arduino, 196
 - podłączenie do pinów układu scalonego, 203
 - przeniesienie oprogramowania Firmata, 213
 - Serial Port, 214
 - sterowanie silnikiem, 201
 - Wi-Fi, 271
 - wybór modelu, 215
- mikrokontroler Arduino Uno, 195, 208
 - sterowanie, 211
- Mini SSC II Mode, 281
- model rakiety
 - barometr, 175
 - obsługa, 177
 - budowa raket, 158
 - szyn, 167
- dane, 183
 - analiza, 183
 - analiza za pomocą programu Rocket Flight Analysis, 185
 - błędy pomiaru, 187
 - obrót i ciśnienie, 189
 - prędkość i wysokość, 186
- dotychczasowe komponenty, 162
- identyfikator UUID, 171
- konstrukcja, 162
- niezbędne elementy, 159
- program odpowiedzialny za zbieranie danych, 169
 - BLECharacteristicInfo, 176
 - BLEDiscovered
 - ↳Peripheral, 172
 - BLEPeripheralInfo, 173
 - BLEServiceInfo, 174
 - etykiety, 178
- funkcja pomocnicza getFileNames, 178
- Graphics.setPixel
 - ↳Graphics(0), 178
- interfejs użytkownika, 178
- kind, 173
- pasek stanu, 171
- plik danych wyjściowych, 178, 179
- przycisk Quit, 171, 179
- przyśpieszenie, 177
- setUpGUI, 172, 176, 178
- touchUpInside, 179
- tryb grafiki bitmapowej, 178
- tryb grafiki wektorowej, 178
- wyszukanie urządzenia SensorTag, 172
- zmienna globalna sensorTag, 172
- przyśpieszeniomierz, 174
- rakieta ST-1, 159
 - budowa, 168
 - deska balsowa, 160
 - Estes Loadstar II Kit, 160, 169
 - komora ładunkowa, 170
 - listwy, 160
 - silnik typu B6-4, 160
 - testowanie, 168
 - uchwyt dla urządzenia SensorTag, 169
 - wyniki, 189
- rakieta ST-2, 158, 159, 160
 - budowa, 163
 - deska balsowa, 161
 - dzioby raket NC-80, 161
 - element o numerze 303090, 161
 - komora ładunkowa, 163, 165
 - komplet gum, 161
 - komplet rurek, 161
 - listwy, 161
 - plan statecznika, 164
 - rurki papierowe BT-80, 161
 - spadochron, 161

- stateczniki, 163
- uchwyty, 164, 166, 167
- wyniki, 191
- zestaw montażowy silnika D i E, 161
- waga startowa rakiet, 190
- wskazówki dotyczące lotów, 182
 - pogoda podczas lotów, 182
 - silniki, 182
 - spadochrony, 182
 - start rakiety, 183
- żyroskop, 175
 - obsługa, 177
- modulacja szerokości impulsów, 216
- modularny zestaw Grove, 68
- Moisture Calibration, 86
- Moisture Meter
 - pełny kod źródłowy, 96
- Molex, 198
- Most Significant Bit, 114
- mostek H, 194, 195, 201, 220
 - budowanie, 196
 - ogólny schemat, 201
- moveBall, 257
- movePaddle, 256
- MSB, 114
- Mutable, 233
- My Programs, 18

N

- napięcie wyjściowe z masy, 69
- nazwy domen, 264
- New, 73
- newButton, 27
- newLabel, 92
- newPlot, 78
- newTextView, 94
- nl, 92
- nl.setBackgroundColor, 92
- nl.setText, 92
- NMOS, 281
- ntv.setEditable, 94
- nullEvent, 30, 36, 41, 42, 60, 95, 225, 248, 257, 270

O

- obiekt
 - BLEMutableCharacteristic, 248
 - Button, 22
 - PlotPoint, 21, 60, 79
 - TextView, 90
- obrót, 189
 - wykres, 190
 - wyniki pomiarów, 189
- obsługa plików, 29
- obudowa typu DIP, 202
- odchylenie magnetyczne, 58
- odległość, 119
- ogólna siła pola magnetycznego, 59
 - obliczanie, 60
- okno dialogowe nowego tworzonego programu, 18
- oldPout, 220
- OPEN, 28, 29
- operacja
 - BITAND \$007F, 285
 - BITOR, 228
- oprogramowanie
 - Arduino, 211, 212, 275
 - Firmata, 213
 - firmware 8G, 157, 162
 - Flash Programmer, 180
 - SensorTag, 180
- orientation, 219
- OUTPUT, 29

P

- p, 25
- p% BITAND \$0080, 114
- Paddles, 245, 246
 - BLECharacteristicInfo, 255
 - BLEMutableCharacteristic, 248
 - BLEPeripheralInfo, 254
 - BLEServiceInfo, 255
 - identyfikatory UUID, 251
 - interfejs użytkownika, 250
 - Quit, 248, 250
 - wyświetlenie, 248
 - kind, 254

- komunikacja urządzeń, 251
- konsola gry
 - program obsługujący, 251
- konwersje, 250
- logika gry, 258
- moveBall, 257
- movePaddle, 256
- nullEvent, 248, 256
- obsługa odliczania, 257
- paletki, 246
 - etykiety stanu, 253
 - interfejs użytkownika, 247
 - program obsługujący, 247
 - śledzenie połączenia, 253
- pileczka, 258
 - odbicia, 259
- scanForPaddles, 252
- serve, 258
- setStatus, 253, 257
- setUpGUI, 250, 261
- stopScan, 253
- touchUpInside, 250, 261
- uaktualnienia, 249
- ustalenie kąta urządzenia, 249
- wyszukiwanie paetek, 252
- wyświetlenia ekranu i pileczki, 252
- zachowanie bieżącego znacznika czasu, 250

- paletki, 246
- parametry
 - \$0001, 234
 - \$0003, 234
 - \$0006, 234
 - \$0012, 234
 - color, 94
 - kind, 110, 112, 173, 237, 254
 - plant\$, 94
 - value, 222
- pętla
 - FOR, 41, 77
 - WHILE, 35, 73, 79
 - WHILE - WEND, 73
- ph, 79
- Picker, 185

- piny, 202
 - 1., 204
 - 12., 204
 - 13., 204, 272
 - 16., 204
 - 1A, 203
 - 1Y, 203
 - 2., 267, 274
 - 2A, 203
 - 2Y, 203
 - 3., 267, 274
 - 3A, 203
 - 3Y, 203
 - 4., 204
 - 4A, 203
 - 4Y, 203
 - 5., 204
 - 8., 204
 - 9., 204
 - A6/DAC0, 70
 - cyfrowe wejścia określające, 204
 - GND, 69
 - RX, 274
 - RX-0, 274
 - tabela stanu silnika, 228
 - TX, 274
 - UART_RX, 267
 - UART_TX, 267
 - ustawienie stanu silnika, 225
 - VCC, 69
 - wyjścia dla szeregowego wejścia-wyjścia, 267
- plant\$, 94
- pliki
 - CSV, 184
 - danych
 - iTunes, 85
 - przenoszenie do oraz z techBASIC, 85
 - E9-4a.rkt, 191
 - E9-4b.rkt, 191
 - flight1.rkt, 184, 189
 - flight1i.rkt, 191
 - flight2.rkt, 189
 - flight2i.rkt, 191
 - kodu źródłowego
 - przenoszenie, 86
 - moisture.csv, 85
 - w języku BASIC, 29
- Plot, 21, 78
- PlotPoint, 21, 60, 78, 79
- PlotPoint.newPlot, 88
- płytki zasilania
 - DFRobot Breadboard
 - Power Supply Kit 5 V / 3.3 V, 277
- podprocedura
 - addPlant, 93
 - advertise, 233, 236
 - BLECharacteristicInfo, 113, 125, 176, 238, 255
 - BLEDiscoveredPeripheral, 105, 108, 172, 236
 - BLEMutableCharacteristicInfo, 239
 - BLEPeripheralInfo, 109, 110, 173, 224, 237, 254
 - BLEServiceInfo, 111, 174, 228, 255
 - drawArrow, 222
 - moveBall, 257
 - movePaddle, 256
 - newLabel, 92
 - newTextView, 94
 - nullEvent, 30, 36, 41, 42, 60, 95, 225, 248, 257, 270
 - readyToUpdateSubscribers, 242
 - scanForPaddles, 252
 - sendText, 241, 242
 - serve, 258
 - setStatus, 253, 257
 - setUp, 220
 - setUpGUI, 27, 35, 37, 41, 61, 176, 178, 240, 250
 - startRecording, 28
 - stopRecording, 29
 - touchUpInside, 28, 29, 91, 179, 243, 250
 - updateValue, 242
 - valueChanged, 241, 283
- podusługi, 104
- pole magnetyczne
 - ogólna siła, 59
- pole tekstowe, 240
- polecenia
 - \$90, 228
 - CLOSE, 29
 - DIM, 21, 27, 78, 217, 282
 - DIM sensorTag AS
 - BLEPeripheral, 109
 - GET, 269
 - IF, 28, 35
 - LINE INPUT, 269
 - location, 52
 - OPEN, 28, 29
 - PRINT, 18, 29, 74
 - PUT, 269, 284
 - setEnabled(0), 26
 - value(j + 1) << 8, 125
 - value(j), 125
 - wejścia-wyjścia, 269
- Pololu Serial Servo Controller, 276
 - elementy zestawu, 278
 - obsługiwane protokoły, 281
- połączenia
 - BLE, 219
 - między urządzeniami
 - iOS, 231
 - problemy, 107
 - TCP/IP, 268
 - typu Molex, 209
 - typu null modem, 266
 - zastąpienie Arduino, 271
 - typu P2P, 232
 - położenie, 52
 - Direction, 53
 - Speed, 53
 - użytkownika, 57
 - pomost Wi Fi, 67
 - Pong, 245
 - porty, 264
 - 21, 264
 - position%, 284
 - potencjometr, 68, 69
 - powiadomienie, 112
 - poziom przezroczystości
 - koloru, 92
 - prędkość, 186
 - błędy pomiaru, 187
 - błąd kalibracji, 187
 - błąd systematyczny, 188
 - fluktuacja pomiarów, 187
 - założenia, 189
 - pręt kontrolny, 285
 - PRINT, 18, 29, 74
 - PRINT #1, 29

- profil GATT, 104, 116
 - barometr, 124
 - magnetometr, 139
 - przyspieszeniometer
 - a magnetometr, 140
 - termometr, 150
 - wilgotnościomierz, 145
 - żyroskop, 132
 - program
 - Accel, 18
 - Accelerometer, 17, 20
 - barometr, 128
 - BLE Chat, 232
 - działający w oparciu
 - o zdarzenia, 28
 - edycja, 17
 - Flash Programmer, 157
 - Gyroscope, 37
 - Hello Hijack, 73
 - Hijack, 76
 - Hijack Moisture Meter, 90
 - Metal Detector, 61
 - MFi, 65, 66
 - Moisture Calibration, 86
 - Moisture Meter, 96
 - obsługujący
 - konsolę gry Paddles, 251
 - paletkę, 247
 - urządzenia typu BLE, 103
 - odczytywanie informacji
 - o położeniu, 52
 - odpowiedzialny za
 - zbieranie danych, 169
 - Rocket Acceleration, 184
 - Rocket Data, 176, 183
 - Rocket Flight Analysis
 - analiza danych, 185
 - SensorTag Accelerometer, 115, 117
 - SensorTag Barometer, 124
 - SensorTag Gyroscope, 132
 - SensorTag Humidity, 145
 - SensorTag Magnetometer, 139
 - SensorTag Thermometer, 153
 - Serial Servo, 281
 - SmartRF Flash
 - Programmer, 181
 - sniffer, 108, 110
 - tekst danych wejściowych
 - i wyjściowych, 19
 - terminala, 269, 274
 - TI SensorTag, 117, 124, 132, 139, 145
 - tworzenie, 17
 - uruchomienie, 15
 - WiFi Terminal, 277
 - WiFi Fly Terminal, 270
 - Programs, 15, 18
 - protokół
 - FTP, 264
 - HTTP, 264
 - TCP/IP, 265
 - przekazanie
 - parametru przez referencję, 93
 - parametru przez wartość, 93
 - przełącznik, 204
 - DP, 207
 - DPDT, 206
 - oznaczenia, 207
 - SP, 207
 - SPDT, 206
 - ST, 207
 - suwakowy, 195
 - typy, 207
 - przenoszenie plików
 - danych
 - do oraz z aplikacji
 - techBASIC, 85
 - kodu źródłowego, 86
 - przepustowość łącza, 263
 - przesycenie, 145
 - przewód kontrolny, 285
 - przycisk
 - Perform action, 181
 - Reset, 180
 - przyspieszenie, 119
 - miara, 19
 - przyspieszeniometer, 17, 20, 115, 161
 - cechy charakterystyczne, 113
 - czujnik, 19
 - identyfikator UUID, 105
 - KXTJ9, 115
 - model rakiety, 174
 - błędy pomiaru, 187
 - odczytywanie wartości, 28
 - oddziaływanie pól
 - magnetycznych, 41
 - paletki, 248
 - podawanie wartości, 24
 - powiadomienie, 112
 - profil GATT, 116
 - readCharacteristic, 113
 - SensorTag, 115, 180
 - atrybut AA11, 117
 - atrybut AA12, 117
 - atrybut AA13, 118
 - błędy śledzenia ruchu, 120
 - kalibracja, 118
 - kod źródłowy
 - programu, 120
 - mierzone wartości, 119
 - odczyt wartości, 117
 - uzyskanie dostępu, 116
 - użycie, 118
 - włączenie, 117
 - wysyłanie
 - powiadomień, 117
 - zmiana częstotliwości
 - podawania wartości, 118
 - trójosiowy, 101
 - uaktualnianie wykresu, 36
 - włączenie, 17
 - zdalnie sterowany
 - samochód, 218
 - zmiana orientacji ekranu, 26
 - PUT, 269, 284
- ## Q
- quadrcorder, 81
 - Quit, 17, 22, 28
- ## R
- radiany, 40
 - radiator, 198
 - readCharacteristic, 113, 118, 126, 134, 140, 146
 - readyToUpdateSubscribers, 242
 - receiveStatus, 239
 - Record, 20, 28
 - recording, 22

- RedBearLab BLE Shield, 195, 196, 208, 221
 - Firmata, 213
 - redBearUUID, 219
 - Redpark Serial Cable, 66
 - referencja, 93
 - regresja liniowa, 86
 - regulacja proporcjonalna, 216
 - remote, 235
 - remoteReadyCharacteristic, 238, 239
 - result%, 242
 - RGB, 78
 - Rocket Acceleration, 184
 - Rocket Data, 176, 183
 - Rocket Flight Analysis, 185
 - kontrolka typu Picker, 185
 - uruchomiony program, 186
 - wykrzes przyśpieszenia, 187
 - wykresy danych, 188
 - Roving Networks, 266
 - Roving Networks RN-VX, 265
 - rozszerzenie
 - .rkt, 178
 - .txt, 178
 - znaku, 114
 - RS-232, 280
 - Run, 19
 - rzeczywistość rozszerzona, 33
- ## S
- sample, 48, 50
 - scanForPaddles, 252
 - Seeed Studio, 67, 80
 - komponenty, 83
 - sekcja
 - System-on-Chip, 181
 - Send, 20, 26, 28, 30
 - sendText, 241, 242
 - Sensor and Comm Classes, 48, 49
 - Sensors, 17, 48
 - położenie, 52
 - tematy pomocy, 49
 - wyznaczenie kursu, 51
 - Sensors.accel, 42
 - Sensors.accelAvailable, 42
 - Sensors.gyro, 42
 - Sensors.gyroAvailable, 42
 - Sensors.mag, 42
 - Sensors.magAvailable, 42
 - Sensors.setAccelRate, 42
 - Sensors.setGyroRate, 42
 - Sensors.setMagRate, 42
 - sensorTag, 172
 - SensorTag, 100, 101, 102, 158
 - aplikacje obsługujące urządzenie, 103
 - barometr, 101, 123
 - BLECharacteristicInfo, 113
 - cechy charakterystyczne, 104
 - czujnik wilgotności, 101
 - czujniki, 101
 - dane wyjściowe po wykryciu urządzenia, 111
 - Digital Humidity Sensor, 145
 - kod źródłowy programu, 103
 - konwersje, 250
 - lista elementów, 102
 - magnetometr, 101, 138
 - model rakiety, 161
 - operacje na bitach i bajtach, 125
 - oprogramowanie firmware, 115, 180
 - instalacja, 180
 - przycisk parowania, 107
 - przyśpieszeniomierz, 101, 115
 - specyfikacja protokołu komunikacyjnego, 104
 - termometr, 101, 150
 - układ współrzędnych, 115
 - uruchomiona aplikacja, 101
 - usługi, 104
 - wilgotnościomierz, 144
 - wyszukanie urządzenia, 106
 - żyroskop, 101, 132
 - SensorTag Accelerometer, 115
 - pełny kod źródłowy, 117, 120
 - SensorTag Barometer, 124
 - SensorTag Gyroscope, 132
 - SensorTag Humidity, 145
 - SensorTag Magnetometer, 139
 - SensorTag Thermometer, 153
 - Serial Servo, 281
 - BYTE, 282
 - konfiguracja interfejsu użytkownika, 283
 - przyciski i suwaki, 283
 - polecenie DIM, 282, 284
 - polecenie PUT, 284
 - zakończenie działania programu, 283
 - zmienna deviceID, 284
 - serve, 258
 - services, 103
 - serwomechanizmy, 275, 277
 - interfejs użytkownika programu, 282
 - kolejne łączenie serwomechanizmów, 282
 - kontroler, 276, 277
 - złącza, 280
 - lista elementów do budowy projektów, 277
 - maski na Halloween, 281
 - obsługiwanie sygnałów, 280
 - poruszanie serwomechanizmem, 284
 - ruch do przodu i do tyłu, 285
 - dozownik M&M, 286
 - dźwignia sterowania, 285, 286
 - pręt kontrolny, 285, 286
 - przewód kontrolny, 285, 286
 - sygnały kontrolne, 276
 - typu ROB-09065 RoHS, 277
 - układ elektroniczny, 279
 - port logic-level serial input, 280
 - port szeregowy RS-232, 280
 - układ NMOS, 281
 - zasilacz DFRobot, 280
 - Wi-Fi, 275
 - ogólne informacje, 275
 - zastosowanie, 276
 - złącza, 278, 280
 - setColor, 91
 - setEnabled(0), 26
 - setGrid, 78
 - setNotify, 118, 126, 134, 140, 146
 - setPoints, 31
 - setStatus, 253, 257

- setTitle, 27, 78
 - setUp, 220
 - setUpGUI, 24, 27, 35, 37, 41, 61, 172, 176, 240, 250
 - showGrid, 78
 - sign extension, 114
 - silnik
 - B6-4, 182
 - C6-5, 182
 - D12-3, 182
 - D12-5, 182
 - dostarczanie napięcia, 217
 - E9-4, 182
 - stan, 201, 217
 - siła pola magnetycznego, 56, 59, 62, 138
 - slave, 100
 - słowa, 114
 - SmartRF Flash Programmer, 181
 - SN754410, 194, 196, 198, 202
 - przeciążenie, 200
 - sniffer, 108, 110
 - Source, 48, 73
 - speed, 219
 - Speed, 53
 - Stack Overflow, 251
 - stan silnika, 217
 - macierze, 220
 - standard IEEE 802.11, 264
 - startRecording, 28
 - startScan, 105
 - stateczniki, 163
 - stateMap, 218
 - Statements, 48
 - status, 219
 - sterowanie samochodem
 - za pomocą urządzenia typu BLE, 194
 - Stop, 23, 28, 79
 - stopnie, 40
 - stopRecording, 28, 29
 - stopScan, 253
 - strzałki kierunkowe, 222
 - Subs, 22, 24
 - sygnał
 - analogowy, 65
 - cyfrowy (A-D), 65
 - System.clearConsole, 74
 - System.wait(0.5), 74
 - szybkość, 53
- ## T
- t0, 22
 - tabela
 - stanu, 228
 - tablica
 - stateMap, 218
 - tag, 184
 - target temperature, 150
 - tasiemka, 200
 - TCP/IP, 265, 268
 - kontrolowanie urządzeń, 265
 - techBASIC, 14, 15
 - DEG, 41
 - dokumentacja, 24
 - dostępność, 15
 - funkcje, 20
 - gradienty, 27
 - heading, 51
 - ikona aplikacji, 102
 - catalog O'Reilly Books, 16
 - liczba całkowita, 126
 - obsługa
 - urządzeń typu BLE, 102
 - zdarzeń, 27
 - Plot, 21
 - pobieranie danych
 - z czujników, 47
 - początkowa zawartość ekranu, 16
 - podręcznik użytkownika, 49
 - dostępność, 50
 - powody używania, 14
 - PRINT, 18
 - przenoszenie plików danych, 85
 - regresja liniowa, 86
 - SensorTag, 103
 - wartość domyślna, 79
 - wbudowany system pomocy, 48, 49
 - wielkość liter, 18
 - wprowadzenie, 14
 - tworzenie programu, 17
 - uruchomienie pierwszego programu, 15
 - wykresy, 20
 - gesty, 25
 - techBASIC Sampler, 15
 - dostępność, 15
 - SensorTag, 103
- technologia
 - BLE, 99
 - komunikacja, 263
 - ograniczenie, 231
 - ograniczenie przesyłanych danych, 234
 - Paddles, 245
 - przepustowość łącza, 263
 - zasięg, 263
 - Bluetooth, 100
 - Bluetooth Low Energy, 99
 - Wi-Fi, 263
 - techSampler, 15
 - temperatura
 - celu, 150
 - rdzenia, 150
 - terminal tekstowy, 269
 - termometr, 101
 - SensorTag, 150
 - atrybut 0xAA01, 151
 - atrybut 0xAA02, 150
 - BLECharacteristicInfo, 151
 - dane wyjściowe, 152
 - konwersja wartości, 151
 - odczyt danych, 151
 - pełny kod źródłowy, 153
 - profil GATT, 151
 - uzyskanie dostępu, 150
 - użycie, 152
 - włączanie, 150
 - wysyłanie powiadomień, 151
 - tesla, 56, 138
 - Texas Instruments
 - SN754410, 194
 - TextView, 90, 93
 - TI BLE Sensor Tag, 106, 108
 - TI SensorTag, 117, 124, 132, 139, 145
 - TMP006, 150
 - touchUpInside, 28, 29, 91, 179, 243, 250
 - Transmission Control Protocol/Internet Protocol, 265
 - tricorder, 13, 53
 - przyśpieszeniomierz, 17
 - wilgotnościomierz, 81

- trójosiowy
 - magnetometr, 101
 - przyśpieszeniomierz, 101
 - żyroskop, 101
- tryb
 - podległy, 231
- turn, 219
- tworzenie
 - programu, 17
- txCharacteristic, 226
- txUUID, 219

U

- UART_RX, 267
- UART_TX, 267
- ujemne napięcie układu, 69
- układ
 - AK8973, 56
 - biegun północny pola magnetycznego, 57
 - położenie użytkownika, 57
 - CC2541, 101
 - MAG3110, 138
 - NMOS, 281
 - TMP006, 150
- układ scalony, 202
 - piny, 202
 - cyfrowego wejścia, 203
 - zasilania, 203
 - SN754410, 194, 196, 198, 202, 203
 - przeciążenie, 200
 - wcięcie, 202
 - zasilanie, 204
- unikatowy uniwersalny identyfikator, 105
- Universal Unique Identifier, 105
- updateValue, 242
- uruchomienie
 - pierwszego programu, 15
- urządzenia
 - Bluetooth, 67
 - CC Debugger, 102, 162, 180
 - przycisk Reset, 180
 - centralne, 231
 - główne, 100
 - HiJack, 65
 - dostarczanie wartości, 95

- iOS, 102
- odwrócenie ról, 232
 - konfiguracja urządzeń, 232
- podległy, 100, 231
- podległe BLE, 233
 - gra Paddles, 247
 - konfiguracja, 233
- Pololu Serial Servo Controller, 276
- ReadBearLab BLE Shield, 274
- RedBearLab BLE Shield, 195, 196, 208
- Redpark Serial Cable, 66
- SensorTag, 157, 162
 - podłączanie, 180
 - usługi, 104
 - zasilanie, 180
 - zbieranie danych, 162
- typu BLE, 67, 99, 100, 157
 - bajty, 114
 - komunikacja, 100, 251
 - nawiązywanie
 - połączenia, 105, 106
 - odczyt danych, 112
 - pobór energii, 100
 - program sniffer, 108, 110
 - SensorTag, 101
 - sposób projektowania, 103
 - tworzenie programów obsługujących, 103
 - usuwanie problemów związanych z
 - połączeniem, 107
 - wyszukiwanie, 105, 106, 252
 - wywołania, 105
 - zakończenie
 - wyszukiwania, 106
 - zapisywanie w zmiennej globalnej, 106
 - zdalne sterowanie samochodem, 194
- WiFly, 67, 265, 275
- XBee Breakout, 266, 277
- ZigBee, 265
- usługa, 103, 232
- BLE
 - uruchomienie, 105

- cechy charakterystyczne, 103
- characteristics, 112
- dołączona, 104
- identyfikator UUID, 105
- komunikacji i rozgłaszania, 233
- podusługi, 104
- położenie, 52
- transmisji szeregowej, 224
- wyznaczanie kursu, 51
- usuwanie problemów związanych z połączeniem, 107
- UUID, 105

V

- value, 222
- value(j + 1) << 8, 125
- value(j), 125
- valueChanged, 241, 283
- VCC, 69, 70

W

- waga startowa raket, 190
- WHILE, 35
- Wi-Fi, 263, 264
 - Arduino, 271
 - Board, 271
 - komunikacja za pomocą programu terminala, 274
 - model mikrokontrolera, 271
 - otwieranie połączenia szeregowego, 272
 - port szeregowy, 272
 - Serial Port, 272
 - układ elektroniczny, 273
 - wczytanie
 - oprogramowania, 271
 - serwomechanizmy, 275
 - WiFly, 266
- WiFly, 67, 265, 276, 277
 - Arduino, 273
 - EOF, 269
 - GET, 269
 - komunikacja z TCP/IP, 268
 - LINE INPUT, 269

- lista niezbędnych elementów, 266
 - nawiązanie połączenia sieciowego, 267
 - polecenie, 268
 - nullEvent, 269
 - polecenia wejścia-wyjścia, 269
 - połączenie typu null modem, 266
 - program terminala, 269
 - PUT, 269
 - RX-VX WiFly Module with Wire Antenna, 277
 - układ elektroniczny, 266
 - piny, 267
 - zasilanie, 266
 - WRL-10822, 277
 - WiFly Terminal, 270
 - Arduino, 274
 - dane wyjściowe, 270
 - LINE INPUT, 270
 - nullEvent, 270
 - Wifly-GSX-xx, 267
 - wilgotnościomierz, 81
 - budowa, 82
 - dodanie do tricordera, 81
 - kalibracja, 83
 - konwersja odczytanych wartości, 88
 - przenoszenie plików danych do oraz z techBASIC, 85
 - użycie danych, 86
 - zbieranie danych, 83
 - komponenty, 83
 - lepsza wersja oprogramowania, 89
 - lista elementów, 83
 - program Moisture Meter, 96
 - SensorTag, 144
 - atrybut AA21, 146
 - atrybut AA22, 146
 - BLECharacteristicInfo, 146
 - dane wyjściowe, 147
 - konwersja temperatury, 146
 - odczyt danych, 146
 - pełny kod źródłowy, 147
 - profil GATT, 145
 - readCharacteristic, 146
 - setNotify, 146
 - uzyskanie dostępu, 145
 - włączenie wysyłania powiadomień, 146
 - zwracane wartości, 146
 - tworzenie tablic, 88
 - ustalenie liczby punktów danych, 87
 - wskaźnik wilgotności, 89
 - wykresy, 88
 - wilgotność
 - powietrza, 144
 - względna, 145, 146
 - writeCharacteristic, 117
 - WRL-10822, 266, 277
 - wskaźnik wilgotności, 89
 - wykrywacz metalu, 55, 63
 - efekt wpływu iPhone'a na pomiar, 62
 - konwersja aplikacji Magnetometer, 60
 - nullEvent, 60
 - setUpGUI, 61
 - uaktualnianie wykresu, 60
 - utworzenie wykresu i tablic, 60
 - zapis danych w pliku, 60
 - magnetometr, 55
 - pełny kod źródłowy, 60
 - poziom przybliżenia wykresu, 62
 - szybkość ruchu smartfonem, 62
 - użycie iPhone'a lub iPada, 58
 - używanie, 61
 - ziemskie pole magnetyczne, 56
 - wysokość, 186
 - błędy pomiaru, 187
 - błąd kalibracji, 187
 - błąd systematyczny, 188
 - fluktuacja pomiarów, 187
 - założenia, 189
 - wyznaczanie kursu, 33, 51
- X**
- XBee Breakout, 266, 277
- Z**
- zasięg, 263
 - zasilacz DFRobot, 280
 - zdalnie sterowany samochód, 193, 194
 - ANGLE, 226
 - DEG, 226
 - demontaż samochodu, 196
 - DIM, 217
 - drawArrow, 222
 - elementy, 195
 - haveConnection, 219, 224
 - implementacja stanów silnika, 225
 - interfejs użytkownika, 220
 - nazwa programu, 221
 - strzałki kierunkowe, 222
 - wskaźnik stanu, 221
 - konfiguracja szybkości i stanu silników, 220
 - lastTime, 219
 - maxState, 220
 - mikrokontroler Arduino Uno, 195
 - inicjowanie komunikacji, 220
 - instalacja oprogramowania, 211
 - pobranie oprogramowania Firmata, 213
 - sterowanie, 211
 - modulacja szerokości impulsów, 216
 - stan silnika, 217
 - modyfikacja, 200
 - mostek H, 195, 201
 - nullEvent, 225
 - obsługa zmian w stanach silnika, 226
 - oldPout, 220
 - oprogramowanie, 216
 - orientacja urządzenia iOS, 220
 - orientation, 219
 - ostateczne natężenie prądu, 198
 - plyta podłogowa pojazdu, 206
 - plytka uniwersalna i zworki, 195
 - połączenia
 - cyfrowego wyjścia i silnika, 203
 - typu BLE, 223

- zdalnie sterowany samochód
 - pomiar natężenia prądu, 198
 - wymaganego do poruszenia samochodu, 199
 - wymaganego przez silnik, 199
- przełącznik suwakowy, 195
- przyśpieszeniomierz, 218
- uruchomienie, 221
- RedBearLab BLE Shield, 221
- regulacja proporcjonalna, 216
- ruch do przodu, 201
- ruch do tyłu, 202
- setUp, 220
- speed, 219
- spoczynek, 201
- status, 219
- sterowanie ruchem pojazdu, 195
- turn, 219
- txCharacteristic, 226
- układ elektroniczny
 - Arduino, 206
 - bateria, 210
 - montaż, 204
 - płytki uniwersalna, 208
 - po zamontowaniu, 210
 - połączenie przewodami, 204
 - przełącznik, 204, 206
 - przewody i zworki, 209
 - przewody
 - przedłużające, 210
 - RedBearLab BLE Shield, 208
 - schemat, 205
 - stan, 201
 - układ scalony SN754410, 208
 - złącza dla połączeń typu Molex, 209
 - zworki, 210
- układ scalony SN754410, 198, 202, 208
- uruchomienie, 228
- urządzenie RedBearLab BLE Shield, 195
- usługa transmisji szeregowej, 224
- ustalenie
 - prawkidłowego kierunku, 227
 - prądu pobieranego przez silniki, 197
 - usunięcie nadajnika radiowego, 199
 - wskazanie kierunku poruszania się samochodu, 218
 - wybór, 194
 - wskazówki, 196
 - zamiana sygnałów cyfrowych na napięcie, 201
 - zasilanie
 - dla silników, 204
 - dla układu scalonego, 204
 - zbyt duży, 200
 - zestaw poleceń Pololu, 281
 - ziemskie pole magnetyczne, 41, 56, 63, 138
 - siła, 58
 - ZigBee, 265
 - złącza
 - Break Away Header, 266, 277
 - comm, 278
 - mode, 278
 - trójzworkowe, 278
 - typu Molex, 197, 229
 - XBee Header, 277
 - zmiennie
 - ballVX, 258
 - ballX, 258
 - ballY, 258
 - deviceId, 284
 - filename\$, 22
 - haveConnection, 219, 224
 - index, 22
 - isA, 235
 - lastTime, 219
 - line\$, 234, 241
 - local, 235
 - maxState, 220
 - nl, 92
 - oldPout, 220
 - orientation, 219
 - p, 25
 - ph, 79
 - recording, 22
 - remote, 235
 - sensorTag, 172
 - speed, 219
 - status, 219
 - t0, 22
 - turn, 219
 - znacznik czasu, 22

Ż

 - żyroskop, 34, 161
 - częstotliwość pobierania danych, 36
 - iPhone, 35
 - jednostka wartości obrotu, 40
 - model rakiety, 175
 - obsługa, 177
 - nullEvent, 36, 41
 - oddziaływanie pól magnetycznych, 41
 - pełny kod źródłowy, 37
 - radiany, 40
 - SensorTag, 132
 - 0xAA52, 133
 - atrybut AA51, 133
 - bity i bajty, 133
 - BLECharacteristicInfo, 134
 - IMU-3000, 132
 - kalibracja, 134
 - odczyt danych, 133
 - pełny kod źródłowy, 135
 - profil GATT, 132
 - readCharacteristic, 134
 - setNotify, 134
 - Triple Axis Motion-Processor™, 132
 - uzyskanie dostępu, 132
 - użycie, 134
 - włączanie, 133
 - zastosowanie, 135
 - setUpGUI, 35, 37, 41
 - sprawdzenie dostępności, 35
 - trójosiowy, 101
 - uaktualnianie
 - wykresu, 36
 - zawartości wyświetlanej na ekranie, 36
 - wartość początkowa czasu, 35
 - wykresy
 - inicjalizacja
 - i wyświetlenie, 41
 - inicjalizacja tablic, 37
 - ustawienie zakresu i domeny, 37, 41

PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



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

Zmień swoją stronę WWW
w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA WYDAWNICZA

 **Helion SA**

Projekty elektroniczne na iPhone i iPad

Niekonwencjonalne gadżety z technologią Arduino i techBASIC



Podstawowe funkcje telefonu, czyli dzwonięcie i wysyłanie SMS-ów, nikomu już dziś nie wystarczają. Współczesne smartfony są wykorzystywane na wiele innych sposobów.

Gry, przeglądanie ulubionych stron w internecie, aktywny udział w życiu portali społecznościowych – to tylko niektóre z nich. Przy odrobinie umiejętności możesz użyć Twojego telefonu także do niekonwencjonalnych działań: na przykład jako wykrywacza metali, barometru lub żyroskopu. To urządzenia, które możesz zbudować na podstawie Twojego iPada lub iPhone'a oraz kilku niedrogich urządzeń dodatkowych. Dzięki tej książce jest to naprawdę proste! W trakcie lektury poznasz język techBASIC, który pomoże Ci zbudować działający higrometr oraz przyspieszoniemierz. Ponadto zdobędziesz wiedzę na temat technologii Bluetooth Low Energy oraz nauczysz się sterować zdalnie samochodem za pomocą urządzenia typu BLE i platformy Arduino. Książka ta jest doskonałą lekturą dla wszystkich pasjonatów elektroniki, którzy chcieliby maksymalnie wykorzystać potencjał swoich smartfonów i tabletów. Zaskocz swoich znajomych niesamowitym zastosowaniem telefonu!

Dzięki tej książce:

- poznasz język techBASIC
- uzyskasz dostęp do czujników wbudowanych w Twoje urządzenie
- zbudujesz żyroskop, barometr lub higrometr
- wykorzystasz w pełni możliwości telefonu iPhone i tabletu iPad

Odkryj nowe zastosowania dla Twoich urządzeń iPhone i iPad!

helion.pl
księgarnia
internetowa

Nr katalogowy: 20397



Księgarnia internetowa:
<http://helion.pl>



Zamówienia telefoniczne:
0 801 339900



0 601 339900



Helion

Sprawdź najnowsze promocje:

🔗 <http://helion.pl/promocje>

Książki najchętniej czytane:

🔗 <http://helion.pl/bestsellery>

Zamów informacje o nowościach:

🔗 <http://helion.pl/nawosci>

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel.: 32 230 98 63

e-mail: helion@helion.pl

<http://helion.pl>

sięgnij po WIECEJ



KOD KORZYŚCI

ISBN 978-83-246-8890-6



Cena 59,00 zł