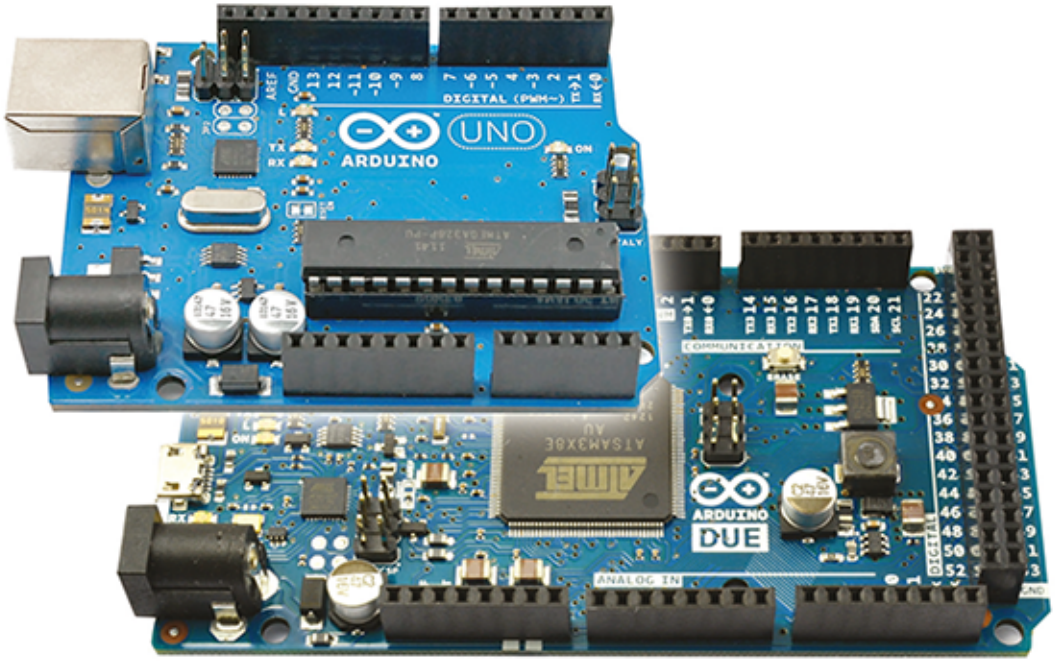
Arduino™
dla początkujących
Kolejny krok

Poznaj tajniki Arduino!



Simon Monk

Helion

Tytuł oryginału: Programming Arduino™ Next Steps: Going Further with Sketches

Tłumaczenie: Konrad Matuk

ISBN: 978-83-283-0013-2

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)

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

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
ftp://ftp.helion.pl/przyklady/arpokk.zip

Printed in Poland.

Spis treści

O autorze ... 9
Podziękowania ... 10
Wstęp ... 11
Pobieranie przykładów ... 12
Czego będę potrzebował? ... 12
Korzystanie z tej książki ... 12

Rozdział 1. Programowanie Arduino ... 15
Czym jest Arduino ... 15
Instalacja i środowisko programistyczne ... 17
Instalacja środowiska programistycznego ... 18
Blink ... 18
Wycieczka po płytce Arduino ... 21
Zasilanie ... 21
Złącza zasilania ... 22
Wejścia analogowe ... 22
Złącza cyfrowe ... 22
Płytki Arduino ... 23
Uno i pochodne ... 23
Duże płytki Arduino ... 24
Małe płytki Arduino ... 25
Płytki LilyPad i LilyPad USB ... 26
Nieoficjalne płytki Arduino ... 27
Język programowania ... 27
Modyfikacja szkicu Blink ... 27
Zmienne ... 29
If ... 30
Pętle ... 31
Funkcje ... 32
Wejścia cyfrowe ... 33
Wyjścia cyfrowe ... 35

Monitor portu szeregowego ... 35
Tablice i macierze ... 37
Wejścia analogowe ... 38
Wyjścia analogowe ... 40
Korzystanie z bibliotek ... 42
Typy danych obsługiwane przez Arduino ... 43
Polecenia Arduino ... 44
Podsumowanie ... 46

Rozdział 2. Pod maską ... 47
Krótka historia Arduino ... 47
Anatomia Arduino ... 47
Procesory AVR ... 48
ATmega328 ... *48*
ATmega32u4 ... *50*
ATmega2560 ... *50*
AT91SAM3X8E ... *51*
Arduino i Wiring ... 51
Od szkicu do Arduino ... 55
AVR Studio ... 56
Instalacja programu rozruchowego ... 58
Instalacja programu rozruchowego za pomocą aplikacji AVR Studio i programatora ... *59*
Instalacja programu rozruchowego za pomocą zintegrowanego środowiska programistycznego Arduino i drugiej płytki Arduino ... *60*
Podsumowanie ... 62

Rozdział 3. Przerwania i zegary ... 63
Przerwania sprzętowe ... 63
Piny przerwań ... *66*
Tryby przerwań ... *67*
Aktywacja wbudowanego rezystora podciągającego ... *67*
Procedury obsługi przerwań ... *67*
Zmienne ulotne ... *68*
Podsumowanie wiadomości na temat procedur obsługi przerwań ... *69*
Włączanie i wyłączanie obsługi przerwań ... 69
Zegary i przerwania ... 70
Podsumowanie ... 73

Rozdział 4. Przyspieszanie Arduino ... 75
Jak szybko działa Twoje Arduino? ... 75
Porównanie płytek Arduino ... 76
Przyspieszanie wykonywania operacji arytmetycznych ... 77
Czy naprawdę musisz stosować wartości typu float? ... *77*
Przeglądanie kontra obliczanie ... 78
Szybkie wejścia-wyjścia ... 80
Podstawowa optymalizacja kodu ... *80*
Bajty i bity ... *82*

Porty układu ATmega328 ..... *82*
Bardzo szybkie działanie wyjść cyfrowych *84*
Szybkie wejścia cyfrowe *84*
Przyspieszanie wejść analogowych 86
Podsumowanie 88

Rozdział 5. Arduino i mały pobór prądu 89
Płytki Arduino i pobór prądu 89
Prąd i akumulatory 91
Zmniejszenie częstotliwości taktowania 92
Wyłączanie komponentów 94
Usypianie 95
Biblioteka Narcoleptic *95*
Budzenie za pomocą zewnętrznych przerwań *97*
Ograniczanie pobieranego prądu za pomocą wyjść cyfrowych 99
Podsumowanie 101

Rozdział 6. Pamięć 103
Pamięć Arduino 103
Korzystanie z minimalnej ilości pamięci RAM 104
Korzystanie z właściwych struktur danych *105*
Zachowaj ostrożność, korzystając z rekurencji *105*
Przechowywanie w pamięci flash stałych będących łańcuchami *107*
Rozpowszechnione błędne przekonania *108*
Pomiar wolnej pamięci *108*
Korzystanie z minimalnej ilości pamięci flash 108
Korzystaj ze stałych *109*
Usuwaj zbędne elementy szkicu *109*
Pomiń program rozruchowy *109*
Statyczna i dynamiczna alokacja pamięci 109
Łańcuchy 111
Tablice elementów typu char *111*
Biblioteka Arduino StringObject *114*
Korzystanie z pamięci EEPROM 115
Przykład korzystania z pamięci EEPROM *116*
Korzystanie z biblioteki avr/eeprom.h *118*
Ograniczenia pamięci EEPROM *120*
Korzystanie z pamięci Flash 120
Zapisywanie danych na kartach SD 121
Podsumowanie 122

Rozdział 7. Korzystanie z magistrali I2C 123
Warstwa sprzętowa 125
Protokół magistrali I2C 126
Biblioteka Wire 126
Inicjacja magistrali I2C *127*
Wysyłanie danych przez urządzenie nadrzędne *127*
Odbieranie danych przez urządzenie nadrzędne *127*

Przykład działania magistrali I2C 128
Radio FM TEA5767 ....*128*
Przesyłanie danych pomiędzy dwoma płytkami Arduino *130*
Płytki z diodami LED *133*
Zegar czasu rzeczywistego DS1307 ....*134*
Podsumowanie 135

Rozdział 8. Praca z urządzeniami wyposażonymi w interfejs 1-Wire 137
Sprzęt obsługujący interfejs 1-Wire 137
Protokół 1-Wire 138
Biblioteka OneWire 139
Inicjalizowanie biblioteki OneWire *139*
Skanowanie magistrali *139*
Korzystanie z układu DS18B20 141
Podsumowanie 143

Rozdział 9. Praca z urządzeniami wyposażonymi w interfejs SPI 145
Operowanie bitami 145
Wartości binarne i szesnastkowe *146*
Maskowanie bitów *146*
Przesuwanie bitów *148*
Sprzęt obsługujący magistralę SPI 150
Protokół SPI 151
Biblioteka SPI 151
Przykład komunikacji za pomocą interfejsu SPI 153
Podsumowanie 157

Rozdział 10. Szeregowa transmisja danych za pośrednictwem układu UART 159
Sprzęt służący do szeregowej transmisji danych 159
Protokół obsługujący szeregową transmisję danych 162
Polecenia służące do obsługi szeregowej transmisji danych 162
Biblioteka SoftwareSerial 164
Przykłady szeregowej transmisji danych 165
Komunikacja pomiędzy komputerem a Arduino za pośrednictwem interfejsu USB *165*
Komunikacja pomiędzy dwoma płytkami Arduino *167*
Moduł GPS *169*
Podsumowanie 172

Rozdział 11. Obsługa interfejsu USB 173
Emulacja klawiatury i myszy 173
Emulacja klawiatury *174*
Przykład emulacji klawiatury *175*
Emulacja myszy *175*
Przykład emulacji myszy *176*
Programowanie hosta USB 176
Płytka USB Host i obsługująca ją biblioteka *177*
Host USB płytki Arduino Due *180*
Podsumowanie 182

Rozdział 12. Obsługa sieci ... 183
Sprzęt sieciowy ... 183
Płytka rozszerzeń wyposażona w kontroler sieci Ethernet ... 183
Arduino Ethernet i Arduino EtherTen ... 184
Arduino i Wi-Fi ... 185
Biblioteka Ethernet ... 185
Nawiązywanie połączenia ... 185
Stawianie serwera sieci Web ... 188
Tworzenie żądań ... 189
Przykład szkicu korzystającego z biblioteki Ethernet ... 189
Sprzętowy serwer sieci Web ... 190
Pobieranie danych w formacie JSON ... 194
Biblioteka WiFi ... 195
Nawiązywanie połączenia ... 195
Funkcje zdefiniowane w bibliotece WiFi ... 196
Przykładowy szkic korzystający z sieci Wi-Fi ... 196
Podsumowanie ... 197

Rozdział 13. Cyfrowe przetwarzanie sygnałów ... 199
Wprowadzenie do cyfrowego przetwarzania sygnałów ... 199
Uśrednianie odczytów ... 200
Wstęp do filtrowania ... 202
Prosty filtr dolnoprzepustowy ... 203
Cyfrowe przetwarzanie sygnałów przez Arduino Uno ... 204
Cyfrowe przetwarzanie sygnałów przez Arduino Due ... 205
Generowanie kodu filtrującego ... 208
Transformacja Fouriera ... 210
Analizator spektrum ... 212
Pomiar częstotliwości ... 214
Podsumowanie ... 214

Rozdział 14. Praca z użyciem tylko jednego procesu ... 215
Zmiana skali ... 215
Dlaczego wątki są zbędne ... 216
Funkcje setup i loop ... 216
Najpierw wykrywaj, a dopiero później reaguj ... 216
Pauza, która nie blokuje mikrokontrolera ... 217
Biblioteka Timer ... 218
Podsumowanie ... 220

Rozdział 15. Tworzenie bibliotek ... 221
Kiedy należy tworzyć biblioteki? ... 221
Stosowanie klas i metod ... 222
Przykładowa biblioteka TEA5767 Radio ... 222
Określ interfejs programistyczny ... 223
Utwórz plik nagłówkowy ... 224
Utwórz plik implementacji ... 225

Utwórz plik ze słowami kluczowymi *226*
Utwórz folder z przykładami *226*
Testowanie biblioteki 227
Publikacja biblioteki 227
Podsumowanie 228

Dodatek A. Podzespoły 229
Płytki Arduino 229
Płytki rozszerzeń 229
Moduły 229
Dystrybutorzy 230

Skorowidz 233

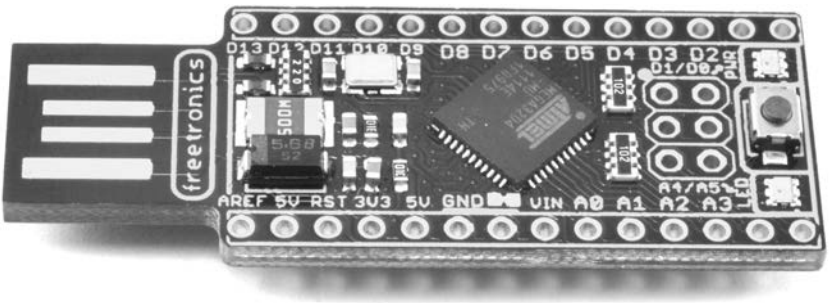
Rozdział 11.

Obsługa interfejsu USB

W tym rozdziale poruszę różne zagadnienia związane z obsługą interfejsu USB przez płytkę Arduino, między innymi emulację klawiatury i myszy przez łytkę Arduino Leonardo. Omówię również odwrotny proces — podłączanie klawiatury i myszy do Arduino.

Emulacja klawiatury i myszy

Oprócz płytki Leonardo również płytki Due i Micro (ich konstrukcja bazuje na płytce Leonardo) mogą emulować klawiaturę lub mysz USB. Taką możliwość oferują również inne płytki kompatybilne z Arduino, takie jak LeoStick produkowana przez firmę Freetronics (zobacz rysunek 11.1).

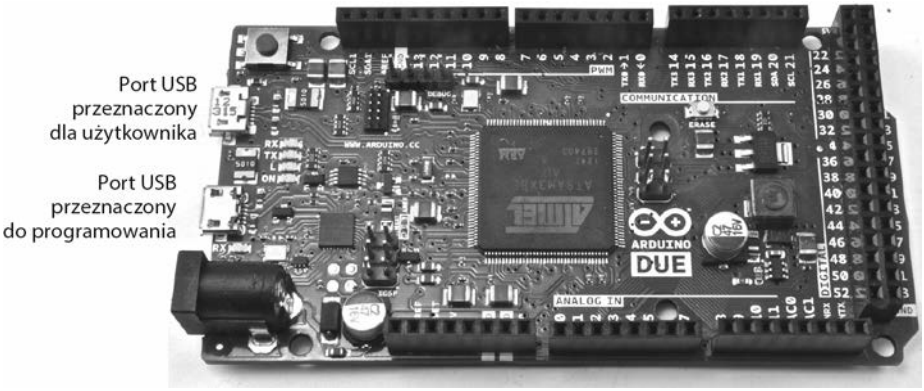


Rysunek 11.1. *Płytka LeoStick*

Wspomniana funkcja przydaje się zwłaszcza podczas pracy z syntezatorami. Pozwala ona płytce Arduino sterować programami takimi jak Ableton Live. Możesz na przykład stworzyć bazujący na Arduino nowatorski instrument muzyczny, którego pracą sterują przyspieszeniomierze, światłomierze lub pedały znajdujące się w pedalboardzie.

Wspomnianą funkcję Arduino można zastosować nieco mniej ambitnie — w celu robienia żartów. Arduino może sprawić, że mysz czyjegoś komputera zacznie żyć własnym życiem, a klawiatura wypisywać losowo wybrane litery.

Płytka Arduino Due jest wyposażona w dwa porty USB. Emulacja klawiatury i myszy może się odbywać za pomocą **portu USB przeznaczonego dla użytkownika**. Płytka ta jest zwykle programowana za pomocą **portu USB przeznaczonego do programowania** (zobacz rysunek 11.2).



Rysunek 11.2. *Porty USB znajdujące się na płytce Arduino Due*

Emulacja klawiatury

Funkcje emulujące klawiaturę są dość łatwe w użyciu. To standardowy element języka, z którego korzystamy, a więc nie ma potrzeby dołączania zewnętrznej biblioteki. Aby rozpocząć emulację klawiatury, umieść następujące polecenie w funkcji `startup`:

```
Keyboard.begin();
```

Tekst „wpisywany” (w dowolnym miejscu, w którym zostanie ustawiony kursor) przez Arduino należy podać jako argument funkcji `print` lub `println`:

```
Keyboard.println("Byla to najlepsza z epok.");
```

Jeżeli chcesz korzystać z klawiszy modyfikujących, na przykład podczas emulacji naciskania klawiszy tworzących kombinację *Ctrl+C*, to zastosuj polecenie `press`:

```
Keyboard.press(KEY_LEFT_CTRL);
Keyboard.press('x');
delay(100);
Keyboard.releaseAll();
```

Funkcja `press` przyjmuje w roli argumentu pojedynczy element typu `char`. Poza normalnymi znakami możesz przekazywać jej wiele zdefiniowanych stałych, takich jak `KEY_LEFT_CTRL`. Po tym, gdy uruchomione jest polecenie `press`, Arduino działa tak, jakby zdefiniowany klawisz był wciśnięty — do momentu uruchomienia polecenia `releaseAll`. Pełną listę klawiszy specjalnych znajdziesz pod adresem *http://arduino.cc/en/Reference/KeyboardModifiers*.

UWAGA

Zaprogramowanie Arduino, podczas gdy mikrokontroler emuluje mysz lub klawiaturę, może okazać się niemożliwe, ponieważ podczas próby zaprogramowania układu może on „pisać" jakiś tekst. Problem ten można rozwiązać, naciskając i przytrzymując przycisk Reset *aż do wyświetlenia się komunikatu „ładowanie" w pasku statusu środowiska programistycznego Arduino.*

Przykład emulacji klawiatury

Poniższy szkic automatycznie (po każdym uruchomieniu Arduino) „wpisuje" tekst określony przez użytkownika (może być to na przykład hasło):

```
// 11_01_klawiatura

char phrase[] = "haslo";

void setup()
{
   Keyboard.begin();
   delay(5000);
   Keyboard.println(phrase);
}

void loop()
{
}
```

Szkic ten byłby lepszy, gdyby proces „pisania" był uruchamiany przez wciśnięcie zewnętrznego przycisku. System operacyjny na komputerze typu Mac wyświetla okno informujące o podłączeniu nowej klawiatury po każdym uruchomieniu Arduino. Tekst nie zostanie „wpisany" przez mikrokontroler dopóty, dopóki okno to nie zostanie zamknięte.

Emulacja myszy

Proces emulacji myszy działa podobnie jak emulacji klawiatury. Tak naprawdę możesz emulować oba urządzenia za pomocą tego samego szkicu.

Emulację myszy należy rozpocząć, stosując polecenie:

```
Mouse.begin();
```

Następnie możesz wykonywać ruchy kursorem za pomocą funkcji `Mouse.move`. Przyjmuje ona trzy parametry: przemieszczanie kursora w płaszczyźnie poziomej i pionowej, a także dane generowane przez pokrętło *scroll*. Wszystkie te trzy wartości są wyrażone w pikselach. Mogą one przyjmować wartość dodatnią (opisując ruch do góry lub w prawo) albo ujemną (opisując ruch w dół lub w lewo). Kursor myszy przemieszcza się o określoną liczbę pikseli względem swojego początkowego położenia. Emulujemy działanie myszy poruszającej kursorem — nie możemy ustawić kursora w dowolnej pozycji bezwzględnej.

Polecenie `click` emuluje kliknięcie przycisku myszy. Jeżeli nie przekażemy tej funkcji żadnych parametrów, to zostanie wykonane kliknięcie lewym przyciskiem myszy. Za pomocą

dodatkowych parametrów możemy wygenerować kliknięcie prawym przyciskiem myszy (`MOUSE_RIGHT`) lub jej środkowym przyciskiem (`MOUSE_MIDDLE`).

Jeżeli chcesz określić czas naciśnięcia przycisku myszy, to możesz to zrobić za pomocą poleceń `Mouse.press` i `Mouse.release`. Polecenie `Mouse.press` może przyjmować te same parametry, co `Mouse.click`. Polecenia te mogą okazać się przydatne w przypadku tworzenia własnej myszy opartej na Arduino. Wtedy kliknięcie myszy może być wywoływane przez przełącznik podłączony do cyfrowego wejścia mikrokontrolera. W ten sposób z łatwością można wygenerować podwójne lub potrójne kliknięcia.

Przykład emulacji myszy

W poniższym przykładowym szkicu kursor znajdujący się na ekranie komputera będzie poruszany w sposób losowy. Aby zatrzymać działanie szkicu i odzyskać kontrolę nad komputerem, musisz wcisnąć i przytrzymać przycisk *Reset* albo odłączyć Arduino od komputera.

```
// 11_02_mysz

void setup()
{
   Mouse.begin();
}

void loop()
{
   int x = random(61) - 30;
   int y = random(61) - 30;
   Mouse.move(x, y);
   delay(50);
}
```

Programowanie hosta USB

Płytki Leonardo, Due i Micro mogą emulować działanie klawiatury lub myszy. Klawiaturę i mysz można jednak podłączyć tylko do gniazda USB płytki Due lub mniej znanej Arduino Mega ADK. Tylko te płytki pozwolą Ci korzystać z klawiatury i myszy jako urządzeń wejścia. Funkcja ta nosi nazwę **USB Host**. Jest ona obsługiwana bezpośrednio tylko przez płytkę Due, ale istnieją również dodatkowe płytki (tzw. shieldy), rozszerzające o tę funkcję możliwości Arduino Uno lub Leonardo.

Wspomniana funkcja może obsłużyć również klawiatury i myszy bezprzewodowe (o ile nie są to urządzenia korzystające z interfejsu Bluetooth). Wystarczy, że ich klucz sprzętowy zostanie włożony do gniazda USB znajdującego się na płytce pełniącej funkcję hosta USB. W ten sposób możesz dodać do Arduino funkcję zdalnego sterowania.

Host USB może również obsługiwać inne, wyposażone w złącza USB, urządzenia peryferyjne, takie jak kontrolery gier, kamery i kontrolery Bluetooth. Możesz również podłączyć do niego telefon pracujący pod kontrolą systemu Android.

Płytka USB Host i obsługująca ją biblioteka

Płytki USB Host i obsługujące je biblioteki są dostępne od kilku lat. Obecnie mogą obsługiwać wiele urządzeń peryferyjnych. Pierwsza tego typu płytka została opracowana przez firmę Circuits@Home (*http://www.circuitsathome.com/*). W ofercie firm SparkFun, SainSmart i innych znajdują się moduły kompatybilne ze wspomnianą płytką. Na rysunku 11.3 przedstawiono host USB firmy SparkFun, podłączony do płytki Arduino Uno. W chwili, gdy piszę tę książkę, tego typu płytki rozszerzeń nie są kompatybilne z Arduino Leonardo, a tak naprawdę w zasadzie nie są kompatybilne z żadnymi płytkami Arduino, będącymi bardziej „egzotycznymi" od Uno. Sprawdź to przed zakupem.

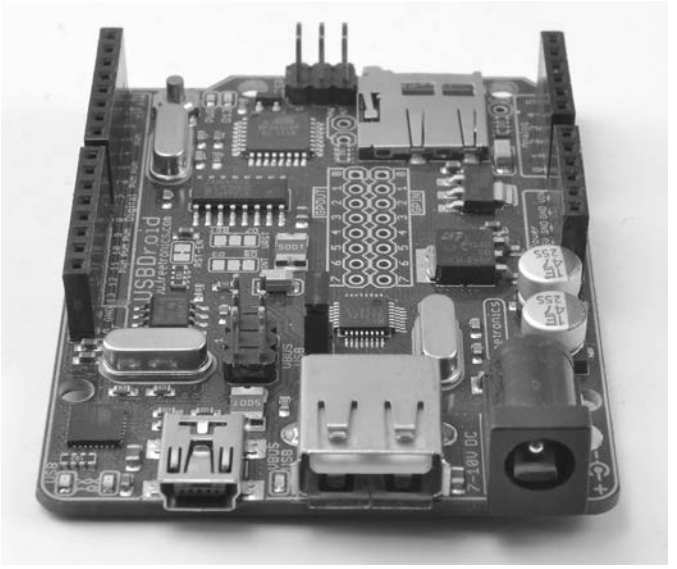
Rysunek 11.3. *Płytka USB Host firmy SparkFun*

Płytka widoczna na rysunku została ponadto wyposażona w przydatną przestrzeń pozwalającą na dolutowanie dodatkowych komponentów. Alternatywnym rozwiązaniem w zakresie korzystania z płytki rozszerzeń jest zakup urządzenia takiego jak USBDroid firmy Freetronics (zobacz rysunek 11.4). Płytka ta została wyposażona w port mikro-USB służący do programowania płytki, a także w pełnowymiarowe złącze USB, do którego możesz podłączyć klawiaturę lub inne urządzenie peryferyjne.

Do pracy z płytką USBDroid lub nieoficjalną płytką rozszerzeń mogącą pełnić funkcję hosta USB niezbędna jest biblioteka USB_Host_Shield firmy Circuits@Home. Jeżeli posiadasz oficjalną płytkę, to możesz korzystać z biblioteki USB_Host_Shield_2 obsługującej więcej urządzeń.

Programowanie za pomocą wspomnianych bibliotek nie jest czymś łatwym. Biblioteka zapewnia dość niskopoziomowy interfejs obsługujący magistralę USB. W pliku *11_03_host_klawiatura.ino*, który możesz pobrać ze strony *http://www.helion.pl/ksiazki/arpokk.htm*, znajdziesz przykładowy szkic obsługujący klawiaturę podłączoną do hosta USB.

Szkic ten powstał na skutek modyfikacji jednego z przykładowych programów dołączonych do biblioteki USB_Host_Shield. Szkic wyświetla wpisywane znaki nie w oknie monitora portu szeregowego (jak oryginalny program), a na wyświetlaczu LCD.



Rysunek 11.4. *Płytka USBDroid firmy Freetronics*

Zarówno ten szkic, jak i oryginalny przykład mogą posłużyć Ci jako szablony pomocne przy tworzeniu własnego kodu. Oba programy prawidłowo obsługują wszystkie klawisze znajdujące się na klawiaturze. Jeżeli chcesz korzystać tylko z klawiszy kierunkowych albo tylko z klawiatury numerycznej, to możesz znacznie uprościć kod szkicu.

Szkic jest zbyt długi, aby umieszczać go tu w całości, a więc skupię się na jego kluczowych elementach. Przed przystąpieniem do lektury warto załadować szkic do pamięci Arduino.

Importowane są trzy biblioteki:

- `#include <Spi.h>`
- `#include <Max3421e.h>`
- `#include <Usb.h>`

Biblioteka *Spi.h* jest wymagana, ponieważ obsługuje interfejs chipa kontrolera hosta USB. Pracujemy z układem Max3421e, a więc potrzebujemy również obsługującej go biblioteki. Ostatnią importowaną biblioteką jest *Usb.h*, która zawiera funkcje uwalniające użytkownika od konieczności kłopotliwego i złożonego korzystania bezpośrednio z chipa.

Pod poleceniami importującymi biblioteki znajdują się definicje stałych, takich jak:

```
#define BANG        (0x1E)
```

W języku C stałe mogą być definiowane na różne sposoby, równie dobrze moglibyśmy zastosować składnię typu:

```
const int BANG = 0x1E;
```

Następnie tworzone są obiekty `MAX3421E` i `USB`, a w funkcji `setup` wywoływana jest funkcja `powerOn` obiektu `Max`:

```
MAX3421E Max;
USB Usb;
```

W funkcji `loop` wywoływane są funkcje `Task` obiektów `Max` i `Usb`. W ten sposób uruchomiony zostaje interfejs sprawdzający aktywność portu USB.

```
void loop() {
   Max.Task();
   Usb.Task();
   if( Usb.getUsbTaskState() == USB_STATE_CONFIGURING ) {  //czeka na stan adresujący
      kbd_init();
      Usb.setUsbTaskState( USB_STATE_RUNNING );
   }
   if( Usb.getUsbTaskState() == USB_STATE_RUNNING ) {
//skanuje klawiaturę
      kbd_poll();
   }
}
```

Po pierwszym uruchomieniu interfejs będzie w trybie `USB_STATE_CONFIGURING`. Połączenie z klawiaturą jest nawiązywane dopiero po wywołaniu funkcji `kbd_init`. Funkcja ta korzysta ze struktury punktu udostępniania danych `ep_record`. W tej strukturze danych umieszczone są fragmenty komunikatu nawiązującego połączenie z klawiaturą:

```
ep_record[ 0 ] = *( Usb.getDevTableEntry( 0,0 ));
    ep_record[ 1 ].MaxPktSize = EP_MAXPKTSIZE;
    ep_record[ 1 ].Interval  = EP_POLL;
    ep_record[ 1 ].sndToggle = bmSNDTOG0;
    ep_record[ 1 ].rcvToggle = bmRCVTOG0;
    Usb.setDevTableEntry( 1, ep_record );
    /* Konfiguracja urządzenia */
    rcode = Usb.setConf( KBD_ADDR, 0, 1 );
```

Po tym, gdy inicjalizacja zostanie zakończona, klawiatura będzie działała w głównej pętli w trybie `USB_STATE_RUNNING`. Wtedy wywołana zostanie funkcja `kbd_poll`, która sprawdza, czy na klawiaturze nie wciśnięto przycisku.

Kluczową linią funkcji `kbd_poll` jest:

```
rcode = Usb.inTransfer( KBD_ADDR, KBD_EP, 8, buf );
```

Funkcja ta sprawdza, czy przycisk na klawiaturze nie został wciśnięty podczas wczytywania kodów odbieranych przez interfejs USB. Kody te nie są kodami ASCII. Dopiero funkcja `HidtoA` konwertuje je na kod ASCII. Jest to najbardziej złożona funkcja w szkicu, ale możesz ją z łatwością zaimplementować we własnym szkicu. Więcej informacji na temat kodów generowanych przez klawiaturę USB i mapowania ich do ASCII znajdziesz na stronie *http://www.win.tue.nl/~aeb/linux/kbd/scancodes-1.html*.

Jedną z ciekawszych funkcji protokołu USB HID (ang. *Human Interface Device*) jest umożliwienie sterowania pracą diod LED informujących użytkownika o trybie pracy klawiszy *Scroll Lock* i *Num Lock*. Pracą tych diod steruje funkcja `kbd_poll` odpowiadająca na wciśnięcie klawiszy *Scroll Lock*, *Caps Lock* i *Num Lock*. Możemy również stworzyć oddzielny szkic sterujący pracą wspomnianych diod. Przykład takiego programu znajdziesz w pliku *11_04_host_scroll_lock.ino*.

Główną funkcją tego szkicu jest:

```
void toggleLEDs( void )
{
   if (leds == 0) {
      leds = 0b00000111;
   }
   else {
      leds = 0;
   }
   Usb.setReport( KBD_ADDR, 0, 1, KBD_IF, 0x02, 0, &leds );
}
```

Trzy najmniej znaczące bity znaku są znacznikami określającymi tryb pracy trzech diod LED znajdujących się na klawiaturze.

Host USB płytki Arduino Due

Płytka Arduino Due może pracować tak, jakby wbudowano w nią host USB. W chwili pisania tej książki funkcja ta jest uznawana przez konstruktorów Arduino za coś eksperymentalnego. Na stronie *http://arduino.cc/en/Reference/USBHost* znajdziesz aktualne informacje dotyczące działania tej technologii i ewentualnych dalszych prac nad nią.

Płytka Due nie ma pełnowymiarowego gniazda USB, a więc nie można do niej podłączyć bezpośrednio klawiatury i myszy. W celu podłączenia urządzeń peryferyjnych do tej płytki musisz kupić kabel Micro USB OTG Host. Przykład takiego przewodu pokazano na rysunku 11.5, na którym widać kontroler bezprzewodowej klawiatury podłączony do Arduino Due. Równie dobrze można korzystać w ten sposób z klawiatury przewodowej.

Rysunek 11.5. *Klawiatura podłączona do Arduino Due za pomocą kabla Micro USB OTG Host*

Biblioteki płytki Arduino Due są prostsze w użyciu od biblioteki USB Host — zamiast kodu interfejsu HID zwracany jest kod ASCII, przypisany klawiszowi wciśniętemu przez użytkownika. Prezentowany przykład ilustruje obsługę klawiatury przez płytkę Arduino Due. Szkic wyświetla w oknie monitora portu szeregowego informację o tym, który przycisk został wciśnięty.

```
// 11_05_klawiatura_due

#include <KeyboardController.h>
```

```
USBHost usb;
KeyboardController keyboard(usb);

void setup()
{
   Serial.begin(9600);
   Serial.println("Uruchomiono program");
   delay(200);
}

void loop()
{
   usb.Task();
}

// Ta funkcja przechwytuje wciśnięcia klawiszy
void keyPressed()
{
   char key = keyboard.getKey();
   Serial.write(key);
}
```

Każde wciśnięcie klawisza powoduje, że biblioteka `KeyboardController` wywołuje funkcję `keyPressed`. Funkcja `keyRelease` może być użyta do przechwytywania zwolnienia klawisza. Aby określić to, który z klawiszy został wciśnięty, na obiekcie `keyboard` należy wywołać jedną z poniższych funkcji:

- `getModifiers` — zwraca maskę bitów określającą, czy wciśnięto jakiś klawisz modyfikujący (*Shift*, *Ctrl* itd.); na stronie *http://arduino.cc/en/Reference/GetModifiers* znajdziesz przykłady użycia tej funkcji.
- `getKey` — zwraca kod ASCII określający to, który klawisz został wciśnięty.
- `getOemKey` — zwraca kod interfejsu HID określający to, który z klawiszy został wciśnięty.

Obsługa myszy jest równie łatwa, a przy tym podobna do obsługi klawiatury. Przykładowy szkic wyświetla w oknie monitora portu szeregowego litery *L*, *P*, *G* i *D* w zależności od tego, czy mysz została poruszona w lewo, w prawo, w górę czy w dół.

```
// 11_06_mysz_due

#include <MouseController.h>

USBHost usb;
MouseController mouse(usb);

void setup()
{
   Serial.begin(9600);
   Serial.println("Uruchomiono program");
   delay(200);
}

void loop()
```

```
{
   usb.Task();
}

// Ta funkcja przechwytuje ruchy myszy
void mouseMoved()
{
   int x = mouse.getXChange();
   int y = mouse.getYChange();
   if (x > 50) Serial.print("P");
   if (x < -50) Serial.print("L");
   if (y > 50) Serial.print("D");
   if (y < -50) Serial.print("G");
}
```

Poza funkcją `mouseMoved` do przechwytywania innych zdarzeń generowanych przez mysz możesz stosować następujące funkcje:

- `mouseDragged` — do tego zdarzenia dochodzi, gdy użytkownik przesuwa mysz, jednocześnie wciskając jej lewy przycisk.
- `mousePressed` — do tego zdarzenia dochodzi, gdy użytkownik wciśnie przycisk myszy. Powinna zostać po nim uruchomiona funkcja `mouse.getButton`, która w roli parametru przyjmuje określenie lewego, prawego lub środkowego przycisku myszy: `LEFT_BUTTON`, `RIGHT_BUTTON` lub `MIDDLE_BUTTON`. Funkcja zwraca wartość logiczną „prawda”, jeżeli dany przycisk jest wciśnięty.
- `mouseReleased` — funkcja ta, stanowiąca uzupełnienie funkcji `mousePressed` , służy do wykrywania zwolnienia przycisku myszy.

Podsumowanie

W tym rozdziale poznałeś zagadnienia związane z łączeniem Arduino z urządzeniami peryferyjnymi za pośrednictwem interfejsu USB.

W kolejnym rozdziale omówię tematykę dotyczącą obsługi przez Arduino sieci przewodowych i bezprzewodowych. Dowiesz się, jak można programować obsługę sieci, a także nauczysz się korzystać z płytek rozszerzeń będących kontrolerami sieci Ethernet oraz Wi-Fi.

Twórz własne projekty z Arduino!

Platforma Arduino — mała płytka o ogromnym potencjale — otworzyła świat elektroniki przed szerokim gronem pasjonatów, którym pozwoliła wreszcie zrealizować wymarzone projekty. Błyskawicznie zdobyła ogromną popularność, na co szybko zareagował rynek — pojawiło się mnóstwo dodatkowych akcesoriów, instrukcji i książek. Wśród tych ostatnich na szczególną uwagę zasługują publikacje Simona Monka.

Elektroniczne projekty jego autorstwa mają liczną grupę fanów. W Twoje ręce oddajemy kolejną książkę poświęconą Arduino. Sięgnij po nią i poznaj tajemnice pracy z tą płytką! W trakcie lektury poznasz krótką historię platformy, a następnie zobaczysz, jak obsługiwać przerwania sprzętowe, współpracować z urządzeniami 1-Wire oraz obsługiwać interfejs USB. Ponadto przyśpieszysz działanie swojej płytki, zoptymalizujesz pobór prądu oraz zmniejszysz zużycie pamięci RAM. Na koniec nauczysz się obsługiwać sieć oraz tworzyć własne biblioteki. To doskonała propozycja dla wszystkich fanów Arduino chcących jeszcze lepiej poznać tę platformę!

Dzięki tej książce:

- poznasz historię platformy Arduino
- zoptymalizujesz zużycie prądu i pamięci
- wykorzystasz magistralę I2C
- nawiążesz komunikację z urządzeniami 1-Wire
- w pełni wykorzystasz potencjał Arduino

Helion
28615 numer katalogowy
księgarnia internetowa
http://helion.pl
zamówienia telefoniczne
0 801 339900
0 601 339900
Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci
Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl
Informatyka w najlepszym wydaniu
sięgnij po WIĘCEJ
KOD KORZYŚCI
ISBN 978-83-283-0013-2
9 788328 300132
cena: 39,00 zł