8.1. Technologia Ethernet

Jedną z najbardziej użytecznych form komunikacji dostępnych w Arduino jest technologia Ethernet. Jest to przyjęty standard budowy sieci komputerowych, umożliwiający urządzeniom wszelkiego rodzaju komunikację między sobą poprzez wysyłanie i odbieranie strumieni danych (zwanych **pakietami** lub **ramkami**).

Technologia Ethernet umożliwia wyjątkowo szybką i niezawodną transmisję danych przez sieć. Każde urządzenie posiada unikalny identyfikator zwany adresem IP, umożliwiający komunikację przy użyciu różnych protokołów sieciowych.

Komunikacja Arduino z siecią Internet jest prosta dzięki nakładce i bibliotece Ethernet, ale zanim omówimy te komponenty, poznajmy kilka pojęć związanych z sieciami komputerowymi. Nawet jeżeli już je znasz, warto przypomnieć sobie terminologię i technologie opisane w tabeli 8.1.

Tabela 8.1. Najważniejsze terminy i pojęcia technologii Ethernet

Termin	Opis
Ethernet	Ethernet jest standardową technologią wykorzystywaną do budowy sieci komputerowych, opisującą sposób przesyłania danych pomiędzy komputerami lub innymi urządzeniami przez sieć przewodową.
Protokół	Protokoły są to przyjęte języki komunikacji umożliwiające urządzeniom porozumiewanie się między sobą. Aby dwa urządzenia mogły się ze sobą komunikować, muszą używać tego samego języka. Na przykład protokół HTTP (ang. *Hypertext Transfer Protocol*, protokół przesyłania dokumentów hipertekstowych) jest powszechnie stosowanym protokołem, który możesz wykorzystać w płycie Arduino skonfigurowanej jako serwer WWW. Protokół HTTP określa język, dzięki któremu serwer WWW Arduino rozumie komunikaty i zapytania odbierane od innych systemów, na przykład komputerów z przeglądarkami.
Adres MAC	Adres MAC (ang. *Media Access Control*, kontrola dostępu do medium) jest to unikatowy identyfikator przypisywany urządzeniom wykorzystującym Ethernet lub inną technologię sieciową. Adres MAC umożliwia jednoznaczne zidentyfikowanie urządzenia w sieci, aby mogło komunikować się z innymi. Nakładka Ethernet Arduino posiada etykietę z przypisanym adresem MAC.
TCP/IP	Protokoły TCP (ang. *Transmission Control Protocol*, protokół sterowania transmisją danych) oraz IP (ang. *Internet Protocol*, protokół internetowy) umożliwiają przesyłanie komunikatów przez globalną sieć Internet (będącą fundamentem znanej i lubianej przez nas sieci WWW).
Adres IP	Adres IP jest to unikatowy identyfikator wykorzystywany przez urządzenia i serwery do identyfikacji w globalnej sieci Internet. Na przykład przy otwieraniu strony internetowej *www.helion.pl* usługa DNS (ang. *Directory Name Service*, usługa słownika *nazw*) zamienia adres *http://www.helion.pl* na numeryczny adres IP *188.117.147.100*.
Lokalny adres IP	Lokalny adres IP jest podobny do zwykłego adresu IP, ale jest używany w szczególności do komunikacji pomiędzy komputerami i urządzeniami w lokalnej sieci. Na przykład w Twojej domowej sieci każdy komputer ma przypisany lokalny adres IP, używany do komunikacji z routerem i innymi komputerami.

Sieci komputerowe i technologia Ethernet są złożonymi zagadnieniami, których pełne poznanie może zająć całe lata, ale w tabeli 8.1 wymienione są najbardziej podstawowe pojęcia, które musisz znać, aby zrozumieć pozostałą część tego rozdziału.

Teraz, po zapoznaniu się z tabelą, przejdźmy do biblioteki *Ethernet* i sprawdźmy, do czego służy.

8.1.1. Biblioteka Ethernet

Biblioteka *Ethernet* jest dostarczana razem ze środowiskiem Arduino IDE. Umożliwia ona konfigurację nakładki Ethernet i komunikację ze światem zewnętrznym oraz skonfigurowanie do czterech (w sumie) równolegle działających serwerów i klientów. Serwer przyjmuje połączenia przychodzące od klientów, po czym wysyła i odbiera dane. Natomiast klient najpierw zestawia z serwerem połączenie wychodzące, dzięki któremu może wysyłać i odbierać od niego dane.

Więcej na ten temat powiemy wkrótce, ale teraz spójrzmy na tabelę 8.2, zawierającą przegląd funkcji dostępnych w bibliotece *Ethernet*.

Tabela 8.2. Przegląd funkcji klas Ethernet, Server i Client dostępnych w bibliotece Ethernet

Funkcja	Opis

`Ethernet.begin(mac)`
`Ethernet.begin(mac, ip)`
`Ethernet.begin(mac, ip, brama)`
`Ethernet.begin(mac, ip, brama, ↪podsieć)`
	Inicjuje bibliotekę za pomocą adresu MAC nakładki i automatycznie konfiguruje adres IP za pomocą serwera DHCP. Opcjonalnie można ręcznie podać adres IP, bramę (najczęściej jest to adres IP routera) i maskę podsieci (informującą nakładkę, jak interpretować adres IP; domyślna maska to 255.255.255.0).

`Server(port)`	Tworzy serwer nasłuchujący na określonym porcie.
`Server.begin()`	Uruchamia serwer oczekujący na komunikaty.
`Server.available()`	Zwraca obiekt klienta, jeżeli są już odebrane od niego dane.
`Server.write()`	Wysyła dane do wszystkich podłączonych klientów.
`Server.print()`	Wysyła dane do wszystkich klientów. Liczby są wysyłane jako ciągi znaków ASCII, na przykład liczba *123* jest wysyłana jako ciąg trzech znaków: *'1'*, *'2'* i *'3'*.
`Server.println()`	Działa podobnie jak `Server.print()`, ale dodatkowo wysyła znak nowego wiersza na końcu każdego komunikatu.
`Client(ip, port)`	Tworzy obiekt klienta, który może łączyć się z określonym adresem IP i portem.
`Client.connected()`	Zwraca informację, czy klient jest połączony z serwerem. Jeżeli połączenie jest zamknięte, a jakieś dane wciąż nie są odczytane, funkcja zwraca wartość `true`.
`Client.connect()`	Nawiązuje połączenie.
`Client.write()`	Wysyła dane do serwera.
`Client.print()`	Wysyła dane do serwera. Liczby są wysyłane jako ciągi znaków ASCII, na przykład liczba *123* jest wysyłana jako ciąg trzech znaków: *'1'*, *'2'* i *'3'*.
`Client.println()`	Działa podobnie jak `Client.print()`, ale dodatkowo wysyła znak nowego wiersza na końcu każdego komunikatu.
`Client.available()`	Zwraca liczbę bajtów gotowych do odczytania (liczbę bajtów wysłanych przez serwer).
`Client.read()`	Odczytuje następny bajt odebrany z serwera.
`Client.flush()`	Usuwa wszystkie bajty wysłane do klienta, ale jeszcze przez niego nieodczytane.
`Client.stop()`	Zamyka połączenie z serwerem.

Oprócz klas `Ethernet`, `Server` oraz `Client` biblioteka *Ethernet* zawiera przydatne klasy ogólnego przeznaczenia, obsługujące protokół *UDP* (ang. *User Datagram Protocol*,

protokół danych użytkownika) do rozgłaszania informacji w sieci. Jeżeli serwer lub klient nie jest wymagany, możesz do wysyłania i odbierania danych z Arduino użyć klasy `EthernetUDP`. Tabela 8.3 zawiera szczegółowy opis funkcji tej klasy.

Tabela 8.3. Przegląd głównych funkcji klasy EthernetUDP w bibliotece Ethernet

Funkcja	Opis
`EthernetUDP.begin(port)`	Inicjuje obiekt `UDP` i określa port do nasłuchu.
`EthernetUDP.read(bufor_pakietów, maks_wielkość)`	Odczytuje z bufora pakiety UDP.
`EthernetUDP.write(komunikat)`	Wysyła komunikat do innego urządzenia.
`EthernetUDP.beginPacket(ip, port)`	Określa adres IP i port urządzenia docelowego. Funkcja musi być wywołana przed wysłaniem komunikatu.
`EthernetUDP.endPacket()`	Kończy komunikat. Funkcja wywoływana po wysłaniu komunikatu.
`EthernetUDP.parsePacket()`	Sprawdza, czy jakiś komunikat oczekuje na odczytanie.
`EthernetUDP.available()`	Zwraca ilość danych odebranych i gotowych do odczytania.

Teraz, kiedy znamy już funkcje i klasy biblioteki *Ethernet*, przyjrzyjmy się samej nakładce. Nakładkę stosuje się w celu rozszerzenia funkcjonalności sprzętowych Arduino. Dzięki niej możesz podłączyć płytę do sieci Ethernet.

8.1.2. Nakładka Ethernet z kartą SD

Oryginalna nakładka Ethernet jest kamieniem milowym w rozwoju platformy Arduino. Umożliwia ona komunikację projektów z innymi urządzeniami przez sieć komputerową lub Internet. Nakładka wykorzystuje układ WIZnet W5100 i oferuje obsługę stosu IP wraz z protokołami TCP i UDP poprzez łącza Ethernet 10/100 Mbit/s. Nakładka jest wyposażona w standardowe gniazdo RJ45 umożliwiające połączenie i współpracę z modemem, routerem lub innymi standardowymi urządzeniami.

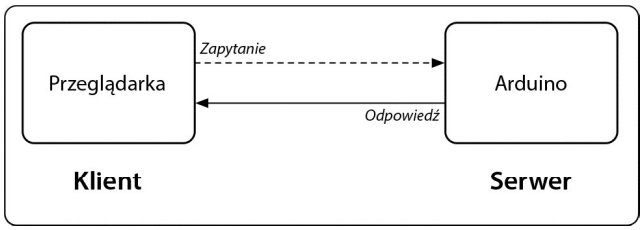
Nowsza, szeroko dostępna wersja nakładki zawiera ulepszenie w postaci slotu na kartę microSD. Dzięki niemu można odczytywać i zapisywać pliki (za pomocą biblioteki *SD*, po umieszczeniu karty SD w slocie). Należy pamiętać, że zarówno układ W5100, jak i karta SD komunikują się z Arduino za pomocą interfejsu SPI (więcej informacji na temat komunikacji SPI znajduje się w podrozdziale 8.6). W większości płyt Arduino wykorzystywane są w tym celu piny nr 11, 12 i 13, natomiast w płycie Mega piny 50, 51 i 52. W obu płytach pin nr 10 jest używany do wybrania do komunikacji układu W5100, natomiast pin nr 4 do wybrania karty SD. Jest to istotne z tego względu, że nie można tych pinów używać jako wejściowych lub wyjściowych pinów ogólnego przeznaczenia.

> **UWAGA:** Zarówno układ W5100, jak i karta SD korzystają z szyny SPI, ale tylko jedno z nich może być aktywne w danej chwili. Aby móc korzystać z karty SD, należy skonfigurować pin nr 4 jako wyjście i ustawić na nim stan wysoki (`HIGH`). Natomiast w przypadku układu W5100 należy jako wyjście skonfigurować pin nr 10 i ustawić na nim stan wysoki. Sprzętowego pinu SS (w większości płyt jest

to pin nr 10, a w płycie Mega nr 53) można nie używać, ale aby móc korzystać z karty SD, biblioteki *Ethernet* i interfejsu SPI, trzeba ustawić go jako wyjście (domyślna konfiguracja).

8.2. Serwer WWW Arduino

Mając w małym palcu podstawy technologii, bibliotekę i nakładkę, możesz zacząć pracę nad swoim pierwszym projektem wykorzystującym Ethernet. Zbudujesz **serwer WWW**, czyli system odpowiadający na zapytania wysyłane przez **klientów** i wysyłający do nich dane (jak przedstawia rysunek 8.1). Aby to osiągnąć, wykorzystasz klasy `Server` oraz `Client` z biblioteki *Ethernet*.



Rysunek 8.1. Schemat komunikacji klienta z serwerem WWW Arduino

8.2.1. Konfiguracja serwera

Do skonfigurowania serwera będziesz potrzebować pewnych informacji.

Po pierwsze, musisz znać adres MAC swojej nakładki Ethernet. Powinien on być wydrukowany na etykiecie na nakładce. W kodzie serwera adres MAC zapiszesz w tabeli bajtów, na przykład tak:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

Pamiętaj, że jest to unikatowy adres sprzętowy, wykorzystywany przez nakładkę do bezpośredniej komunikacji z innymi urządzeniami przez sieć Ethernet.

Jeżeli posiadasz starą nakładkę bez etykiety albo ją zgubiłeś, możesz wykorzystać adres MAC z powyższego przykładu. Jeżeli do sieci jest dołączonych więcej niż jedno urządzenie, ważne jest, aby każde posiadało swój własny unikatowy adres MAC.

W następnym kroku będziesz potrzebować adresu IP. Od wersji 1.0 Arduino wbudowana biblioteka *Ethernet* obsługuje protokół *DHCP* (*Dynamic Host Configuration Protocol*, protokół dynamicznego konfigurowania urządzenia), umożliwiający Arduino automatyczne przypisanie adresu IP. Adres IP zostanie automatycznie skonfigurowany niezależnie od tego, czy płyta jest podłączona bezpośrednio do modemu, czy do routera, o ile tylko na tych urządzeniach jest włączona usługa DHCP (zazwyczaj jest).

Jeżeli w Twojej sieci nie jest używany protokół DHCP albo używasz wersji Arduino starszej niż 1.0, musisz dowiedzieć się, jaki adres IP ma router, i ręcznie przypisać nakładce Ethernet jakiś inny adres IP. W zależności od konfiguracji jest kilka sposobów wykonania tej operacji.

Jeżeli płyta Arduino jest podłączona bezpośrednio do modemu, jego adres IP jest przypisany przez operatora internetowego ISP. W takim przypadku najprostszym

sposobem poznania adresu IP jest podłączenie komputera do modemu i skorzystanie z jednej z wielu stron w Internecie rozpoznających Twój adres IP. (Proste wyszukanie frazy „adres IP" powinno pomóc w znalezieniu odpowiedniej strony, ewentualnie możesz skorzystać z *http://www.adres-ip.pl*.) Zwróć uwagę, że adres IP Twojego routera jest automatycznie przypisywany przez operatora ISP i może się zmieniać od czasu do czasu.

Jeżeli płyta Arduino jest dołączona do sieci przez router, musisz ręcznie przypisać do nakładki jakiś niewykorzystywany adres IP. Aby to zrobić, musisz posiadać nieco więcej informacji na temat konfiguracji swojej sieci. Adres IP routera możesz poznać, korzystając z komputera dołączonego do sieci i opisanej wyżej usługi internetowej. Możesz również otworzyć panel administracyjny swojego routera, wpisując w przeglądarce jego domyślny adres IP. Na przykład w przypadku urządzeń Linksys jest to adres *http://192.168.1.1*, a dla innych marek *http://10.0.0.1*. Jeżeli Twój router ma np. adres 192.168.1.1, musisz nakładce Ethernet nadać adres 192.168.1.x, gdzie *x* oznacza dowolną liczbę od 2 do 254. Każdy komputer lub inne urządzenie w sieci posiada swój unikatowy adres 192.168.1.x, w którym ostatnia liczba identyfikuje to urządzenie w sieci. Dlatego musisz się upewnić, że Twoja nakładka nie wchodzi w konflikt z innymi urządzeniami. Ta sama zasada obowiązuje, gdy Twój router wykorzystuje inną adresację, na przykład 10.0.0.x.

Jeżeli musisz ręcznie przypisać adres IP, możesz wykorzystać obiekt `IPAddress`, na przykład:

```
IPAddress manualIP(192, 168, 1, 2);
```

Liczby w powyższym kodzie oznaczają adres, który chcesz przypisać.

Na koniec jeżeli jesteś podłączony do sieci i wykorzystujesz router do połączenia z Internetem, musisz również ustawić jego adres IP jako bramę. Utwórz tabelę typu `byte` i zapisz w niej adres IP routera, który będzie użyty jako adres bramy. Poniżej przedstawiony jest przykład:

```
byte gateway[] = { 192, 168, 1, 1};
```

Teraz, kiedy znasz już konfigurację IP sieci, przejdźmy do kodu.

8.2.2. Szkic konfigurujący serwer WWW

Listing 8.1 stanowi praktyczne zastosowanie omówionych wcześniej zagadnień do skonfigurowania serwera WWW na płycie Arduino. Serwer będzie miał za zadanie dołączyć się do Internetu (lub do sieci lokalnej), przyjmować połączenia przychodzące od klientów (wysyłane przez przeglądarki) i odpowiadać własnym komunikatem. Listing jest doskonałym szablonem do tworzenia niemal dowolnej aplikacji serwerowej.

Listing 8.1. Serwer WWW na płycie Arduino

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

Przypisanie unikatowego adresu MAC

```
IPAddress manualIP(192,168,1,120);
EthernetServer server(80);

boolean dhcpConnected = false;

void setup()
{
  if (!Ethernet.begin(mac)){
    Ethernet.begin(mac, manualIP);
  }
  server.begin();
}

void loop()
{
  EthernetClient client = server.available();
  if (client) {
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();

        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println();
          client.println("Witaj, jestem Twoim serwerem Arduino!");
          break;
        }
        if (c == '\n') {
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }
    delay(1);
    client.stop();
  }
}
```

Ręczne przypisanie adresu IP, jeżeli DHCP nie działa

❶ Inicjalizacja serwera

❷ Połączenie za pomocą DHCP

❸ Ręczne połączenie, jeżeli DHCP nie działa

❹ Uruchomienie serwera

Oczekiwanie na połączenia od klientów

Połączenie i odczyt danych od klienta

❺ Koniec zapytania i odesłanie odpowiedzi do klienta

Zwłoka na odczytanie danych przez klienta

Zamknięcie połączenia

Najpierw inicjowany jest serwer HTTP na porcie nr 80 ❶. Po zainicjowaniu możesz spróbować podłączyć Arduino do sieci Ethernet, korzystając z usługi DHCP ❷, a jeżeli nie będzie to możliwe, ręcznie przypisując adres ❸. Dalej następuje uruchomienie serwera i w głównej pętli rozpoczyna się nasłuchiwanie połączeń ❹.

Gdy do serwera podłączy się klient i odebrane zostaną dane, wówczas znak nowego wiersza będzie oznaczał koniec komunikatu, po czym do klienta zostanie odesłana odpowiedź ❺.

8.2.3. Załadowanie i test szkicu

Skopiuj dokładnie kod z listingu 8.1 do środowiska Arduino IDE. Teraz możesz załadować szkic do Arduino.

Po załadowaniu i uruchomieniu kodu możesz zdalnie połączyć się z serwerem Arduino, otwierając w dowolnej przeglądarce adres *http://twój_adres_arduino*. Twoja przeglądarka (klient) wyśle do serwera żądanie połączenia, który z kolei odpowie komunikatem „*Witaj, jestem Twoim serwerem Arduino!*". Tak to działa.

Chcesz otrzymywać rzeczywiste dane? Podłącz potencjometr lub czujnik do wejścia analogowego nr 0 i zamień wiersz:

```
client.println("Witaj, jestem Twoim serwerem Arduino!");
```

na następujący:

```
client.println(analogRead(0));
```

Mamy nadzieję, że szkic zadziała bez problemów, ale jeżeli nie otrzymasz żadnej odpowiedzi, przeczytaj poniższy punkt poświęcony usuwaniu usterek.

8.2.4. Usuwanie usterek

Jeżeli nie możesz nawiązać połączenia z Arduino, pierwszą rzeczą, którą należy sprawdzić, są ustawienia adresu IP. Jeżeli jesteś absolutnie pewny, że konfiguracja IP jest poprawna i jesteś podłączony do sieci domowej, możliwe, że musisz skonfigurować na routerze przekierowanie portów. Przekierowanie portów powoduje, że router w specjalny sposób przesyła komunikaty przeznaczone dla Twojego Arduino. Konfiguracja przekierowania portów nie jest trudna; musisz ją wykonać na swoim routerze. Aby uzyskać więcej informacji, zajrzyj do dokumentacji routera, jak ustawić przekierowanie na adres IP Arduino. To powinno rozwiązać problem.

8.3. Ćwir, ćwir — komunikacja z portalem Twitter

Tworzenie serwera WWW komunikującego się ze światem zewnętrznym to wspaniałe zajęcie, ale inną użyteczną opcją jest połączenie z usługami w Internecie. Jedną z usług, z której warto skorzystać, jest portal Twitter.

Zasada działania portalu Twitter jest prosta. Jeżeli masz w nim swoje konto, możesz rozsyłać w całej sieci Twitter tweety (komunikaty) o długości maksymalnie 140 znaków. Użytkownicy mogą zapisywać się do Twojego kanału i automatycznie otrzymywać aktualizacje Twoich tweetów. Ale to nie wszystko. Twitter dobrze współpracuje z innymi usługami, czyli możesz również automatycznie wysyłać swoje tweety do konta w portalu Facebook.

Czy nie byłoby wspaniale skonfigurować kanał Twitter dostarczający aktualnych informacji o tym, co się dzieje z Twoim Arduino? Jest to możliwe. W tym podrozdziale dowiesz się, jak skonfigurować Arduino i nakładkę Ethernet, aby po naciśnięciu przycisku podłączonego do płyty automatycznie wysyłać komunikaty do portalu Twitter.

8.3.1. Twitter i tokeny

Jeżeli nie masz jeszcze konta w portalu Twitter albo na potrzeby tego projektu chcesz założyć nowe, odwiedź stronę *www.twitter.com* i utwórz konto.

Następnie pobierz specjalny token, który umożliwi autoryzację Arduino podczas wysyłania komunikatów przez Twoje konto. Ten token umożliwia pośredniemu serwerowi WWW mediację pomiędzy Arduino a portalem Twitter. Możliwa jest również bezpośrednia komunikacja z portalem, ale korzystanie z usługi pośredniczącej jest lepsze, ponieważ zapobiega przerwaniu wykonywania Twojego kodu w przypadku, gdy Twitter zmieni swój protokół komunikacyjny lub sposób autoryzacji. Również biblioteka *Twitter* jest mniejsza dzięki usłudze pośredniczącej. Oszczędza się w ten sposób cenną pamięć Arduino.

Aby pobrać token, otwórz stronę *http://arduino-tweet.appspot.com* i kliknij odnośnik *Step 1: Get a token to post a message using OAuth* (Krok 1. Pobierz token do wysyłania komunikatów za pomocą usługi OAuth).

Po skonfigurowaniu konta pora przyjrzeć się bliżej bibliotece, z której będziemy korzystać.

8.3.2. Biblioteki i funkcje

Aby móc komunikować się z portalem Twitter, musisz zainstalować bibliotekę *Twitter*, dostępną pod adresem *www.arduino.cc/playground/Code/TwitterLibrary*. Po pobraniu biblioteki zapisz ją w folderze *sketchbook* lub *libraries*.

Tabela 8.4 przedstawia opis funkcji zawartych w bibliotece *Twitter*.

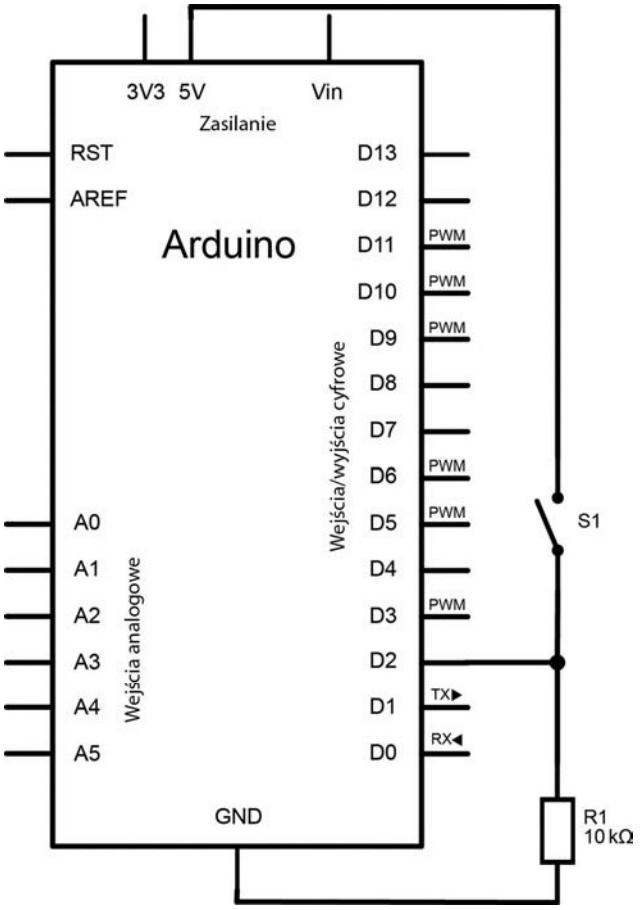
Tabela 8.4. Przegląd funkcji biblioteki Twitter

Funkcja	Opis
`Twitter(string token)`	Konstruktor klasy, przyjmujący token jako argument.
`bool post(const char *komunikat)`	Rozpoczyna wysyłanie komunikatu. Zwraca wartość `true` po pomyślnym nawiązaniu połączenia z portalem Twitter lub `false` w przypadku błędu.
`bool checkStatus(Print *debug)`	Sprawdza, czy żądanie opublikowania komunikatu jest wciąż realizowane (można pominąć argument `debug`, jeżeli nie jest wymagana informacja zwrotna).
`int status()`	Zwraca kod HTTP odpowiedzi z portalu Twitter, na przykład *200 OK* Status jest dostępny dopiero po opublikowaniu komunikatu, gdy funkcja `checkStatus()` zwróci wartość `false`.
`int wait(Print *debug)`	Czeka na zakończenie publikacji komunikatu. Zwraca kod HTTP odpowiedzi z portalu Twitter.

8.3.3. Schemat układu i połączenia komponentów

Jeżeli konto w portalu Twitter i środowisko Arduino IDE są skonfigurowane, zbudujmy prosty układ wysyłający tweeta, gdy użytkownik naciśnie przycisk. Przylutuj przycisk do płyty lub zbuduj układ na płycie prototypowej.

Podłącz jedną z końcówek przycisku z pinem 5 V, a drugą z pinem masy GND poprzez rezystor obniżający 100 kΩ. Tę samą końcówkę przycisku (podłączoną do masy) dołącz do wejścia cyfrowego nr 2, jak pokazuje rysunek 8.2.



Rysunek 8.2. Prosty układ do wysyłania tweetów po naciśnięciu przycisku

8.3.4. Szkic do wysyłania tweeta po naciśnięciu przycisku

Po podłączeniu przycisku i umieszczeniu na Arduino nakładki Ethernet skopiuj poniższy kod (listing 8.2) do środowiska Arduino IDE i możesz zaczynać. Przeczytaj uważnie komentarze w kodzie i upewnij się, że kod zawiera ustawienia odpowiadające konfiguracji Twojej sieci. Jeżeli potrzebujesz wyjaśnień dotyczących jakiegoś pojęcia sieciowego lub terminu, wróć do podrozdziału 8.1.

Listing 8.2. Szkic do wysyłania komunikatu do portalu Twitter po naciśnięciu przycisku

```
#include <SPI.h>
#include <Ethernet.h>
#include <Twitter.h>

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress manualIP(192,168,1,120);

int b1Pin = 2;
int pressCount = 0;
```

Przypisanie unikatowego adresu MAC

Ręczne przypisanie adresu IP, jeżeli DHCP nie działa

Określenie pinu przycisku

```
Twitter twitter("TWOJ-TOKEN");

void setup()
{
  delay(1000);
  if(!Ethernet.begin(mac)){
    Ethernet.begin(mac, manualIP);
  }
  Serial.begin(9600);
}

void sendTweet(const char msgToSend[])
{
  Serial.println("Laczenie...");
  if (twitter.post(msgToSend)) {
    int status = twitter.wait(&Serial);
    if (status == 200) {
      Serial.println("OK.");
    }
    else {
      Serial.print("Blad : kod ");
      Serial.println(status);
    }
  }
  else {
    Serial.println("Polaczenie nieudane.");
  }
}

void loop()
{
  if(digitalRead(b1Pin) == HIGH)
  {
    pressCount++;
    sendTweet("Liczba nacisniec przycisku: " + pressCount);
    delay(2000);
  }
}
```

❶ Inicjalizacja tokena do portalu Twitter

❷ Połączenie za pomocą DHCP

❸ Ręczne połączenie, jeżeli DHCP nie działa

❹ Zestawienie połączenia szeregowego

❺ Sformatowanie i wysłanie tweeta

❻ Sprawdzenie przycisku i wysłanie tweeta

Najpierw Arduino próbuje podłączyć się do sieci Ethernet, korzystając z usługi DHCP ❶, a jeżeli nie będzie to możliwe, za pomocą ręcznie przypisanego adresu IP ❷. Po podłączeniu otwierane jest połączenie szeregowe, poprzez które wysyłane są komunikaty diagnostyczne do monitora portu szeregowego ❸.

Jeżeli połączenie z portalem zostanie nawiązane, zastosowana zostanie funkcja `sendTweet`, odpowiednio formatująca i wysyłająca tweeta ❹. Kod sprawdza, czy został naciśnięty przycisk, po czym wysyła tweeta ❺.

8.3.5. Załadowanie i test szkicu

Jeżeli jesteś pewien, że szkic zawiera poprawne dane, skompiluj go i załaduj do Arduino. Teraz możesz rozgłaszać naciśnięcia przycisku w portalu Twitter. Sprawdź, czy płyta Arduino wykonuje kod i czy jest podłączona do Internetu przewodem Ethernet. To jest cała konfiguracja.

Naciśnij przycisk podłączony do Arduino i zaloguj się do portalu Twitter. Powinieneś w nim zobaczyć ostatni tweet o treści „Liczba nacisniec przycisku: 1", jak pokazuje rysunek 8.3. Po każdym naciśnięciu przycisku pojawi się nowy tweet ze zwiększającą się liczbą naciśnięć. Fajna zabawa.

Rysunek 8.3. Widok tweetu w portalu Twitter po naciśnięciu przycisku

> **UWAGA:** Twitter blokuje wielokrotne wysyłanie w krótkim przedziale czasu tego samego komunikatu. W listingu 8.2 komunikat jest zmieniany po każdym naciśnięciu przycisku (zwiększana jest liczba naciśnięć), a więc nie będzie problemu z wysłaniem tweeta. W większości przypadków okresowego wysyłania komunikatów lub komunikatów różniących się między sobą ten problem nie powinien wystąpić. Niemniej jednak warto o tym pamiętać.

Jak zapewne się domyślasz, możesz rozgłaszać wiele innych pożytecznych informacji, nie tylko dotyczących naciśnięcia przycisku. Na przykład czujnik może wysyłać sygnał, gdy drzwi do Twojego domu zostaną otwarte lub zamknięte. Możesz wysyłać bieżące dane i przedstawiać je w swojej galerii. To tylko zarys możliwości.

8.4. Łączność Wi-Fi

Nakładka Ethernet błyskawicznie podłączy Twoją płytę Arduino do sieci, ale w niektórych sytuacjach przydaje się łączność bezprzewodowa. Na przykład gdy musisz odbierać na bieżąco dane z samobieżnego robota własnej konstrukcji albo gdy Twój układ musi być podłączony do sieci, a w pobliżu nie ma łącza ani routera przewodowego. Sięgnij wtedy po nakładkę WiFi, eleganckie rozwiązanie umożliwiające podłączenie Arduino do sieci bezprzewodowej i przesyłanie danych.

> **UWAGA:** Jeżeli posiadasz nakładkę SparkFun WiFly (popularny odpowiednik oficjalnej nakładki Arduino WiFi) albo z jakiegoś powodu musisz ją wykorzystać w projekcie, na stronie internetowej oryginalnego wydania książki znajdziesz odpowiednio zmienioną wersję tego podrozdziału (w języku angielskim). Odnośnik do niego zamieściliśmy w dodatku E.

8.4.1. Nakładka Arduino WiFi

Nakładka Arduino WiFi umożliwia podłączenie płyty do dowolnej sieci bezprzewodowej typu 802.11b/g. Wykorzystuje moduł bezprzewodowy H&D Wireless HDG104, oferujący uzyskanie zoptymalizowanego, energooszczędnego połączenia radiowego. Nakładka umożliwia komunikację za pomocą protokołów TCP i UDP, a jej użycie jest bardzo proste i polega na zamontowaniu na płycie Arduino i wpisaniu w szkicu kilku wierszy kodu wykorzystującego bibliotekę *WiFi*. Łączówki nakładki posiadają w górnej części gniazda, umożliwiające łatwe wykorzystanie pinów Arduino albo założenie dodatkowych nakładek.

Oprócz obsługi sieci bezprzewodowych w standardzie 802.11b/g nakładka WiFi oferuje szyfrowanie WEP oraz WPA2. Po załadowaniu szkicu i skonfigurowaniu płyty Arduino można ją odłączyć od komputera, zasilić z zewnętrznego źródła i zestawić dwukierunkową komunikację z dowolnego miejsca w zasięgu routera bezprzewodowego.

Ale to nie wszystko. Nakładka WiFi zawiera również slot na kartę microSD, który może być wykorzystany zarówno przez płytę Arduino Uno, jak i Mega, dzięki prostej w użyciu bibliotece *SD*. Jest to bardzo przydatna funkcjonalność, jeżeli zamierzasz zapisać dane, a następnie przesłać je przez sieć. W podrozdziale 8.7 dowiesz się dokładnie, jak korzystać z biblioteki *SD*.

Ważna informacja na temat pinów wejścia/wyjścia

Nakładka Arduino WiFi oraz czytnik kart SD komunikują się z Arduino za pomocą szyny SPI (opisanej niżej w podrozdziale 8.6), którą cechuje kilka istotnych szczegółów związanych z wykorzystaniem pinów wejścia/wyjścia.

W płycie Arduino Uno komunikacja jest realizowana na pinach nr 11, 12 i 13, natomiast w płycie Mega na pinach nr 50, 51 i 52. W obu płytach pin nr 10 jest używany do wybrania do komunikacji układu HDG104, natomiast pin nr 4 do wybrania czytnika karty SD. Sprzętowy pin SS w płycie Mega (pin cyfrowy nr 53) nie jest używany ani przez czytnik kart, ani przez układ HDG104, ale musi być skonfigurowany jako wyjście, aby interfejs SPI działał prawidłowo. Pin cyfrowy nr 7 jest używany do przesyłania danych pomiędzy nakładką WiFi a Arduino. Bardzo ważne jest więc, aby żaden z wymienionych wyżej pinów nie był wykorzystywany do innych operacji wejścia/ wyjścia.

I wreszcie ponieważ zarówno układ HDG104 w nakładce WiFi, jak również czytnik kart SD korzystają z tej samej szyny SPI, tylko jeden z komponentów może być aktywny w danej chwili. Jeżeli wykorzystywane są oba, biblioteki *SD* oraz *WiFi* realizują ich obsługę automatycznie. Ale jeżeli jest wykorzystywany tylko jeden z nich, należy jawnie odseparować drugi (jeżeli nie jest używany czytnik kard SD, należy go ręcznie odseparować), jak pokazuje przykładowy kod.

Konstrukcja nakładki WiFi jest starannie przemyślana i oprócz nawiązywania połączeń bezprzewodowych oferuje szereg przydatnych funkcjonalności. Jej budowa jest całkowicie otwarta. Nakładka jest wyposażona w port Micro-USB na potrzeby przyszłych aktualizacji wbudowanego oprogramowania. Zawiera również serię diod LED dostarczających przydatnych informacji, takich jak status połączenia (zielona dioda LINK), błędy transmisji (czerwona dioda ERROR) i wysyłanie lub odbieranie danych (niebieska dioda DATA).