Bluetooth łączy mikrokontrolery, część 1

Bluetooth (w skrócie BT) zdobywa coraz większą popularność, świadczy o tym jego coraz częstsze występowanie w urządzeniach powszechnego użytku. BT można spotkać nie tylko w komputerach, mamy go już w komórkach i wielu innych urządzeniach.

Specyficznym rodzajem danych jest dźwięk zapisany w postaci cyfrowej. BT jest więc wykorzystywany w bezprzewodowych zestawach głośnomówiących przeznaczonych do telefonów komórkowych. Upowszechniają się także moduły BT, których zalety można wykorzystać już w urządzeniach budowanych samodzielnie. Dodatkowo są one bardzo odporne na wszelakiego rodzaju zakłócenia. Mogą być zastosowane wszędzie tam, gdzie jest potrzebna bezprzewodowa transmisja danych, przy czym użycie takich modułów wyręcza programistę z konieczności panowania nad protokołem. Przy zastosowaniu modułów BT należy jedynie ustanowić połączenie i przesyłać dane, nie martwiąc się o kontrolę poprawności przesyłanych informacji. Zajmie się tym układ sterujący modułem BT. Moduły BT mają zasięg zazwyczaj do kilkuset metrów. Można dla nich znaleźć wiele różnorodnych zastosowań. Większość modułów BT służących do przesyłania dźwięku jest wyposażona w interfejsy USB, RS232 oraz PCM. W artykule zostaną przedstawione dwa przykłady wykorzystania modułów BT do bezprzewodowej komunikacji ze sobą mikrokontrolerów. Ich oprogramowanie sterujące zostało napisane w Bascomie. Jeden przykład przedstawi system bezprzewodowego (zdalnego) odczytu temperatury, natomiast drugi pokaże, jak zrealizować bezprzewodowy system przesyłania sygnałów z czujek alarmowych do centrali. W drugim przykładzie transmisja będzie szyfrowana. W przykładach zawartych w tym artykule wykorzystane zostały moduły BT firmy ConnectBlue, które były szczegółowo przedstawione wraz z listą komend AT w EP9/03. Moduły te mają jedynie interfejs RS232, przy czym skonfigurowanie modułu do nawiązania połączenia jest niezwykle proste. Konfigurowanie modułów odbywa się za pomocą komend AT – tak jak w modemach oraz niektórych modułach GSM lub GPS. Użycie komend AT niezwykle upraszcza tę czynność. Dodatkowym atutem użytych modułów BT jest to, że po konfiguracji i nawiązaniu połączenia interfejs RS232 modułów staje się „przezroczysty". Użytkownik odnosi wrażenie, jakby moduły były połączone przewodem. Można więc powiedzieć, że moduły BT firmy ConnectBlue tworzą bezprzewodowy interfejs RS232.

Jedną z cech modułów, które wykorzystano w drugim przykładzie, jest możliwość szyfrowania przesyłanych danych. Znajdzie ona zastosowanie wszędzie tam, gdzie będzie wymagane duże bezpieczeństwo przesyłanych danych. Moduł BT firmy ConnectBlue może komunikować się jednocześnie nie tylko z jednym (innym) modułem, ale przy pracy z włączonym trybem „Multidrop" jeden moduł może jednocześnie komunikować się z wieloma modułami BT (maksymalnie z siedmioma). W tym przypadku jeden moduł BT jest modułem głównym (wielopunktowym) – „MultiDrop". Moduł pracujący jako „Multidrop" automatycznie konfiguruje bezprzewodową sieć BT i rozdziela dane do wszystkich modułów dołączonych. Nie jest wymagane dodatkowe oprogramowanie przy pracy modułów BT w tym trybie. W przykładzie pierwszym główny moduł BT (np. z włączonym trybem Multidrop) będzie mógł się komunikować bezprzewodowo z wieloma oddalonymi od siebie bezprzewodowymi czujnikami temperatury.

W dalszej części artykułu postaram się udowodnić, że ich stosowanie jest niezwykle proste. W dwóch opisywanych dalej przykładach, w których połączone będą ze sobą tylko dwa moduły, tryb „MultiDrop" będzie oczywiście wyłączony. Moduły będą sterowane tylko za pośrednictwem mikrokontrolerów. Artykuł ten ma pokazać sposób wykorzystania BT w prostych projektach. Zainteresowani przebiegiem transmisji danych przez interfejs RS232 pomiędzy dwoma mikrokontrolerami, a zwłaszcza sposobem interpretowania odebranych danych, także znajdą dla siebie w przedstawionych dwóch przykładach wiele cennych informacji. Moduły BT można zastosować nie tylko w urządzeniach działających w oparciu o RS232, ale także działających z RS422 lub RS485. Przedstawione dalej przykłady programów można sprawdzić w praktyce (bez potrzeby posiadania modułów BT), łącząc mikrokontrolery przewodem przystosowanym do RS232.

Praktyczne informacje o wykorzystaniu modułów BT firmy ConnectBlue

Moduły BT firmy ConnectBlue mogą pracować w dwóch trybach: w trybie przesyłania danych (*data mode*) oraz w trybie konfigurowania modułu za pomocą komend AT (*AT mode*). Po włączeniu zasilania moduł BT zawsze domyślnie pracuje w trybie wymiany danych. Aby przejść do pracy modułu w trybie komend AT, za pośrednictwem których będzie można dokonać konfiguracji, należy wysłać do modułu odpowiednią sekwencję znaków, które zmienią tryb pracy na AT. Sekwencja przejścia modułu do trybu AT zostanie dokładnie przedstawiona w opisie pierwszego przykładu. Moduł BT może być skonfigurowany jako serwer (*server*) lub jako klient (*client*). Połączenie modułów BT odbywa się zawsze pomiędzy serwerem a klientem lub jednym serwerem i wieloma klientami. Jeżeli inne moduły mają się włączyć do danego modułu BT, to ten moduł musi być skonfiguro-

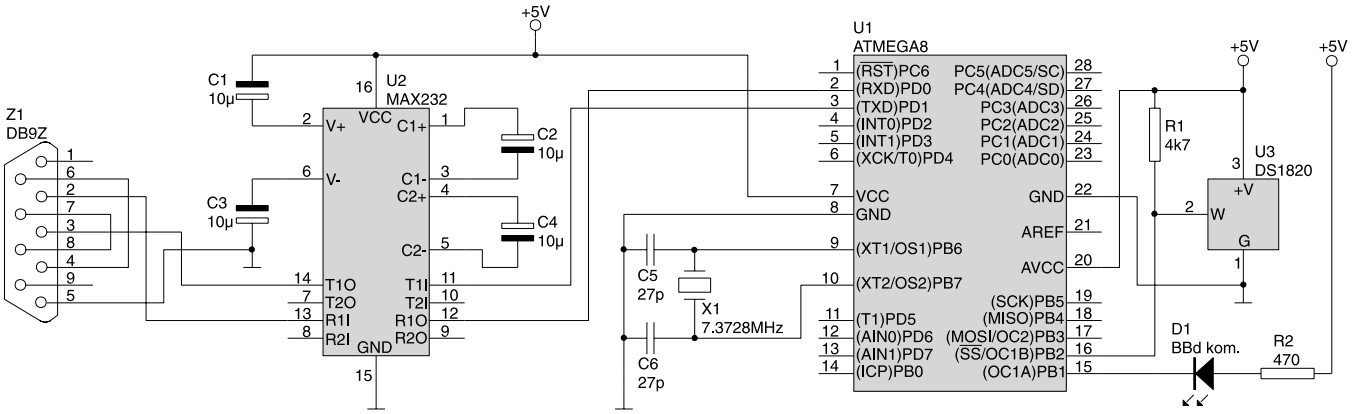
wany do pracy jako serwer. Jeżeli moduł BT (lub moduły) ma ustanawiać połączenie z serwerem, powinien zostać skonfigurowany jako klient (klient zawsze ustanawia połączenie z serwerem). Moduł skonfigurowany jako klient musi znać adres modułu serwera, z którym będzie się łączył oraz jego nazwę. Moduły BT mają swoje adresy, tak jak mają je np. karty sieciowe. Moduły firmy ConnectBlue mają trzy metody wybrania serwera, z którym będą się łączyć. Pierwszą metodą jest znalezienie (wyszukanie) serwera. Drugą, ręczne wpisanie jego adresu i nazwy, natomiast trzecia metoda polega na wyborze serwera z tzw. „listy ulubionych", na którą wcześniej został wpisany. Moduł BT firmy ConnectBlue ma możliwość włączenia opcji pozwalającej na odbiór danych z innego modułu BT. Jeżeli moduł pracuje jako klient i nadchodzi do niego połączenie z odległego modułu BT, zmienia automatycznie swoją konfigurację tak, by dla przychodzącego połączenia stał się serwerem. Wspomniana była tzw. „lista ulubionych". Jest to lista, na którą można wpisać co najwyżej 10 modułów BT pracujących jako serwery, z którymi moduł ten będzie się łączył. Taka lista może służyć do szybkiego wyboru modułów (serwerów), z którymi klient będzie nawiązywał połączenie, przy czym jest pomocna, gdy często jest zmieniany serwer. Kiedy dany moduł komunikuje się z poszczególnymi modułami, to jest konfigurowany jako *master* (układ nadrzędny). Moduł BT pracujący jako *master* ma prawa do ustanawiania połączenia z innym modułem BT, do wykonywania wyszukiwania oraz do akceptacji połączeń przychodzących od innych modułów. W przypadku, gdy moduł BT pracuje jako układ podrzędny (*slave*), ma prawa do łączenia się do innych modułów BT, a także jako *master* do wykonywania wyszukiwania oraz do akceptacji połączeń przychodzących od innych modułów BT. Wszystkie możliwe parametry i funkcje modułów BT można skonfigurować za pomocą komend AT. Dostępnych jest 50 łatwych w użyciu komend AT. Oczywiście, jak wcześniej pisałem, przed użyciem komend AT, moduł BT powinien zostać przełączony z trybu transmisji danych do trybu AT. Konfigurowane parametry oraz tryby pracy modułu BT za pomocą komend mogą być zapisane w modułach na stałe (zapisanie w pamięci nieulotnej modułu) lub mogą być pamiętane, aż do wyłączenia zasilania modułu. W przykładach wybrana została druga opcja, gdyż mikrokontrolery sterujące modułami BT zawsze po włączeniu zasilania zaczynają swoją pracę od konfigurowania modułów BT. Nie było więc potrzeby zapisywania ustawianej konfiguracji w nieulotnej pamięci modułu BT. Dokładny wykaz oraz opis komend modułów BT firmy ConnectBlue można znaleźć na stronie www.connectblue.com, a także w EP9/03 w artykule poświęconym modułom BT tejże firmy.

System bezprzewodowego odczytu temperatury

W tej części artykułu zostanie pokazany przykład bezprzewodowego systemu odczytu temperatury mierzonej w odległym pomieszczeniu. Na marginesie warto zaznaczyć, że pomiar jest wykonywany z dokładnością do 0,1 stopnia. Wykorzystano tu tylko jeden czujnik temperatury, ale nie znaczny to, że nie można zastosować ich więcej. Układ mierzący temperaturę pracuje z modułem BT skonfigurowanym jako serwer. Urządzenie to można więc nazwać „serwerem temperatury". Natomiast urządzenie odczytujące (bezprzewodowo) temperaturę zostało skonfigurowane jako klient. Gdyby „serwer temperatury" został skonfigurowany do pracy w trybie „MultiDrop", mogłoby się z nim łączyć wiele końcówek typu klient. W ten sposób można zrealizować wyświetlanie mierzonej przez serwer temperatury w wielu pomieszczeniach, w których znajdują się „klienci temperatury". W przykładzie występuje tylko jeden klient, więc nie został wykorzystany tryb „MultiDrop". Przykładowy system działa tak, że na zapytanie klienta serwer dokonuje pomiaru i odsyła wartość zmierzonej temperatury. Bezprzewodowy odczyt zmierzonej temperatury jest także możliwy za pomocą jedynie modułu BT i komputerowego terminala.

Serwer temperatury

Na **rys. 1** przedstawiono schemat ideowy serwera temperatury, którym steruje mikrokontroler ATMEGA8. Czujnikiem temperatury jest znany układ DS1820 z magistralą 1-wire. Dioda LED sygnalizuje błąd wykonania komendy AT oraz prawidłowość skonfigurowania modułu BT. Jeżeli konfigurowanie BT zakończy się pomyślnie, dioda LED będzie świeciła światłem ciągłym, w przeciwnym razie będzie migała. Ponieważ moduły BT firmy ConnectBlue posiadają interfejs RS232 zgodny z poziomami napięć -12 V i +12 V (choć nie tylko, bo mają one także linie interfejsu RS232 zgodne z poziomami TTL), zastosowano w układzie dodatkowy konwerter poziomów MAX232. Dopasowuje on sygnały interfejsu RS232 modułu BT do poziomów akceptowanych przez mikrokontroler. Układ MAX232 jest typową przetwornicą pojemnościową, która zwiększa napięcie 5 V oraz je neguje. Na **list. 1** został przedstawiony program realizujący serwer temperatury napisany w Bascomie. W pierwszej kolejności zostają skonfigurowane parametry transmisji RS232. Mikrokontroler sterujący będzie się komunikował z modułem z domyślnymi parametrami BT, czyli: prędkość 57600 bd, dane 8-bitowe, brak parzystości oraz 1 bit stopu. Aby uzyskać taką prędkość transmisji interfejsu szeregowego mikrokontrolera, zastosowany został rezonator kwarcowy o częstotliwości 7,3728 MHz. Taki rezonator gwarantuje uzyskanie wymaganej podstawy czasu dla UART-u. Przy takim wyborze rezonatora błędy transmisyjnej podstawy czasu nie wystąpią nawet dla większych prędkości niż 57600 bd. Oczywiście prędkość interfejsu RS232 modułu BT można zmienić odpowiednimi komendami AT. Linia sterująca diodą LED została skonfigurowana jako wyjście, natomiast linia, do której został dołączony czujnik, jako linia 1-wire. Dodatkowo w programie zastosowano odbiór buforowy danych (z wykorzystaniem przerwania od odbiornika RS232 i bufora). W ten sposób uzyskuje się ochronę przed przeoczeniem znaku otrzymanego z RS232. Bufor odbiorczy został skonfigurowany za pomocą instrukcji `config serialin` na wielkość 10 znaków. W programie zadeklarowano dwie procedury: procedurę sprawdzania poprawności wykonania komendy AT oraz procedurę pomiaru temperatury. Zadeklarowano także kilka zmiennych, z których `odczyt`, `s` i `uniw` służą do interpretacji otrzymanych z RS232 znaków, pozostałe wykorzystywane są przy pomiarze i obliczaniu temperatury. W dalszej części programu, linii sterującej diodą został przypisany alias, dla instrukcji



Rys. 1. Schemat ideowy serwera temperatury

`input` zostało wyłączone echo (instrukcja `input` nie będzie wysyłała zwrotnie otrzymanych znaków) oraz włączone zostają globalne przerwania, by działała transmisja buforowa. Jak pisałem, aby przejść z trybu przesyłania danych do trybu AT, należy do modułu BT wysłać odpowiednią sekwencję znaków. Domyślną sekwencją znaków (można ją zmienić komendami AT), które zmieniają tryb danych modułu BT na tryb AT, jest wysłanie trzech znaków „/”. Ale to nie wystarczy, by wejść w tryb AT. Należy spełnić jeszcze odpowiednie kryteria przed i po wysłaniu tej sekwencji. Przed i po wysłaniu sekwencji znaków „/” przez jedną sekundę nie mogą być przesyłane przez RS232 żadne znaki. W programie przed wysłaniem znaków „/” odczekiwana jest 1 sekunda, a po wysłaniu, dla bezpieczeństwa 2 sekundy. Co najważniejsze, cała sekwencja znaków „/” (czyli wysłanie) „///” musi odbyć się w czasie nie większym niż 200 ms. Tak więc wejście modułu do trybu AT z poziomu komputerowego terminala nie będzie raczej możliwe. Powrót z trybu AT do trybu danych jest możliwy poprzez wykonanie odpowiedniej komendy. W dalszej części programu wysyłane są komendy, na przykładzie których krótko opiszę, co będzie się dziać z modułem BT. Wcześniej przedstawię pokrótce dane o formacie instrukcji AT i informacjach zwrotnych po ich przesłaniu. Komenda AT składa się z trzech części: prefiksu, ciała i zakończenia. Prefiksem są zawsze znaki „AT”. Wielkość liter nie ma znaczenia przy wysyłaniu komend do modułu BT. Ciałem jest łańcuch znaków składających się na rozkaz, a zakończeniem znak CR. Znak CR odpowiada znakowi ASCII o numerze 13 i jest równoważny np. klawiszowi *Enter*. Do modułów BT mogą być wysyłane także komendy rozszerzone, których znaki „AT” są dodatkowo rozszerzone o znak „*”. W programie większość wysyłanych komend to komendy rozszerzone. Należy także zwrócić uwagę na to, że parametry komend AT są oddzielone przecinkami. Jeżeli wysłana komenda ma zwrócić ciąg żądanych z modułu parametrów, otrzymywana informacja jest poprzedzona znakami CR, LF, gdzie CR to znak potwierdzenia, a LF (kod ASCII 10) to znak nowej linii. Poprawne wysłanie i wykonanie komendy przez BT jest sygnalizowane wysłaniem przez BT znaków: CR, LF, „OK”, CR, LF. Znaki „OK” świadczą o prawidłowym wykonaniu przesłanej komendy. W przypadku jakiegoś błędu i niewykonania komendy, moduł BT zwróci CR, LF, „ERROR”, CR, LF – czyli wyśle komunikat „ERROR”. Pierwsza wysyłana w programie komenda „ate0” wyłącza echo znaków, które otrzymuje moduł BT. Domyślnie echo modułu BT jest włączone, ale przy współpracy z mikrokontrolerem jest ono wyłączone, gdyż jest niewykorzystywane. Instrukcja `print`, jeśli nie jest zakończona znakiem „;” sama dba o zakończenie wysyłanych znaków znakiem CR. Po wysłaniu każdej komendy wywoływana jest procedura `sprawdz_stat`, która sprawdza, czy wysłana komenda została poprawnie wykonana. W procedurze tej na początku czyszczona jest zmienna `odczyt` typu *string*. Następnie w pętli `do-loop` odczytywane są za pomocą instrukcji `inkey` znaki, aż do otrzymania znaku LF. Ma to na celu pozbycie się początkowych znaków CR i LF, które uniemożliwią proste porównanie, czy otrzymano znaki „OK”. W kolejnej pętli `do-loop`, także wykonywanej aż do otrzymania znaku LF, odbywa się odczyt znaków i zapis ich ze zmiennej `s` do łańcucha znaków `odczyt`. Jeżeli otrzymany znak to „O” lub „K”, to jest on dodawany do zmiennej `odczyt`. Pojedyncze odebrane znaki są zapisywane do zmiennej `s`. Jeżeli odczytano znak LF kończący wysłany przez BT komunikat, następuje sprawdzenie, czy zmienna `odczyt` posiada znaki różne od „OK”. Jeśli tak, to znaczy że otrzymano inne znaki niż „OK”. Następuje wtedy wejście do nieskończonej pętli `do-loop`, w której naprzemiennie co 250 ms zostaje zmieniony stan diody LED (LED będzie migała). Wyjście z tej pętli będzie możliwe po wyzerowaniu mikrokontrolera. Oczywiście ta procedura nie może być użyta w tej postaci, jeśli po wykonaniu komendy moduł BT będzie odsyłał żądane dane, gdyż będzie to powodowało zawieszenie programu. Należy wtedy napisać nową lub przerobić opisaną procedurę, by dodatkowo umożliwiała odczyt żądanych danych, wysłanych przez BT po wykonaniu danej komendy. Kolejne wysyłane komendy mają znak „*”, czyli są to komendy rozszerzone. Komenda „AGDM...” ma pierwszy parametr równy „1”, który powoduje, że moduł nie będzie widoczny dla innych modułów, tzn. inne moduły nie będą od niego mogły odczytać informacji o adresie, nazwie, nie będzie możliwe skorzystanie z komendy wyszukania modułu itp. Ostatni parametr tej komendy, jak i w kolejnych komendach oznacza, czy ustawienia dokonane daną komendą będą pamiętanie w nieulotnej pamięci modułu, czy nie. Nie ma potrzeby zapisywania ustawień w pamięci nieulotnej, ponieważ jak pisałem, zawsze po włączeniu mikrokontrolera będzie on konfigurował moduł BT. Tak więc ostatnie parametry komend mają wartości „0”. Komenda „AGCM...” z pierwszym pa-

List. 1. Program realizujący serwer temperatury

```
'Serwer Temperatury z Bluetooth
'Przyklad programu konfigurujacego Bluetooth jako serwer wysylajacy na otrzymane
'zapytanie zmierzona temperature czujnikiem DS1820
'przesyłanie dany niekodowane
'Marcin Wiazania
'marcin.wiazania@ep.com.pl

$regfile = „m8def.dat”                      'rejestry mikrokontrolera atmega8
$crystal = 7372800                          'czestotliwosc taktowania mikrokontrolera
$baud = 57600                               'informuje kompilator o predkosci transmisji

Config Pinb.1 = Output                      'linia pb.1 jako wyjscie
Config Serialin = Buffered , Size = 10      'konfiguracja by interfejs rs232 uzywal przy odbiorze transmisji buforowej (bufor o wielkosci 10 znakow)
Config 1wire = Portb.2                      'konfiguracja linii pb2 jako 1wire

Declare Sub Sprawdz_stat                    'procedura sprawdzajaca status wykonania wyslanego polecenia at
Declare Sub Pom_temp                        'procedura pomiaru temperatury z rozdzielczoscia 0,1 st. C

Dim Odczyt As String * 5                    'zmienna string ktora przechewuje odczytanu status z bluetooth oraz z formatowana wartosc temperatury
Dim S As String * 1                         'pomocnicza zmienna tekstowa
Dim Uniw As Byte                            'uniwersalna zmienna pomocnicza
Dim Bd(9) As Byte                           'tablica
                                            'przechowujaca wyniki odczytanych pomiarow temperatury z termometru DS1820
Dim T As Integer                            'zmienna przechowujaca zmierzona temperature
Dim T1 As Integer                           'zmienna pomocnicza do obliczania temperatury
Dim Tmp As Byte                             'zmienna pomocnicza do obliczania temperatury

Led Alias Portb.1                           'przypisanie aliasu led linii pb.1

Set Led                                     'wylaczenie diody led
Echo Off                                    'wylaczenie echa instrukcji input
Enable Interrupts                           'globalne odblokowanie przerwan

Wait 1                                      'czekaj 1 sekunde
Print „///”;                                'wyslij 3x”/” bez
                                            'wysylania dodatkowego kodu 13 (CR - enter) - przelacza modul BT w tryb AT z trybu danych
Wait 2                                      'czekaj 2 sekundy
Print „ate0”                                'wyłączenie echa wysyłanych komend
Call Sprawdz_stat                           'sprawdzenie statusu wykonania wyslanej do BT komendy
Print „at*agdm=1,0”                         'modul BT nie bedzie widoczny dla innych modulow BT
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agcm=2,0”                         'wlaczenie przyjmowania i akceptowania polaczen
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agpm=1,0”                         'wylaczenie trybu parowania modulow
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agsm=1,0”                         'wylaczenie bezpieczenstwa polaczen (autoryzacja i szyfrowanie)
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agmsp=0,0”                        'modul BT w nadchodzacych polaczeniach zawsze bedzie probowal stac sie masterem
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agln={034}Server Temp{034},0”     'nadaje nazwe „Server Temp” modulowi BT
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*aglc=0,0”                         'zapisuje COD modułu BT
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*addcp=255,0”                      'wylaczenie profilu dla clienta (wylaczenie pracy jako clienta)
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*addsp=0,0”                        'wlaczenie profilu portu szeregowego dla serwera (praca jako serwer)
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*adwm=0,0,0”                       'wylaczenie mozliwosci jednoczesnej pracy z wieloma modulami BT (wylaczenie trybu wireless MultiDrop)
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*accb=0,0”                         'wyłacza mozliwosc zdalnej konfiguracji modulu BT
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*addm”                             'przelacza modul BT z powrotem w tryb transmisji danych        '
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Reset Led                                   'zapala diode LED

Do
Input Odczyt                                'odczytanie znakow z bufora odbiorczego
 Uniw = Instr(odczyt , „t?” )               'spawdzenie czy w odczytanym ciagu znakow znajduje sie slowo „t?”
 If Uniw > 0 Then                           'jesli uniw>0 (znajduje sie slowo „t?”) to
   Call Pom_temp                            'wywolanie procedury pomiaru temeratury
   Odczyt = Str(t)                          'konwersja odczytanej temperatury na postac tekstowa
   Print „Temp=” ;                          'wyslanie znakow „Temp=” bez dodatkowego znaku CR (13 - enter)
   Print Format(odczyt , „0.0” )            'waslanie z formatowanej do 0.0 wartosci temperatury
 End If
Loop                                        'koniec petli do-loop
End                                         'koniec programu

Sub Pom_temp                                'procedura pomiaru temperatury z dokladnioscia do 0,1 st C
 1wreset                                    'zerowanie magistrali 1wire
 1wwrite &HCC                               'przeskocz zapis numeru ID
 1wwrite &H44                               'Start pomiaru
 Waitms 750                                 'czekaj 600 ms - czas potrzebny na dokonanie pomiaru temperatury
 1wreset                                    'zerowanie magistrali 1wire
 1wwrite &HCC                               'przeskocz zapis numeru ID
 1wwrite &HBE                               'rozkaz odczytu danych
 Bd(1) = 1wread(9)                          'odczyt danych do tablicy bd
 1wreset                                    'zerowanie magistrali 1wire
                                            'kolejne instrukcje obliczaja temperature z precyzja 0,1 st C
 Tmp = Bd(1) And 1                          'obliczenie wartosci zmiennej tmp
 If Tmp = 1 Then Decr Bd(1)                 'jesli tmp=1 to zmniejsz o jeden pierwszy element tablicy bd
 T = Makeint(bd(1) , Bd(2))                 'laczy dwa bajty w zmienna integer
 T = T * 50 : T = T - 25 : T1 = Bd(8) - Bd(7) : T1 = T1 * 100    'obliczenie temperatury
 T1 = T1 / Bd(8) : T = T + T1 : T = T / 10  'obliczenie temperatury (zmienna T zawiera obliczana temperature)
End Sub                                     'koniec procedury pomiaru temperatury

Sub Sprawdz_stat                            ' procedura sprawdzania statusu wykonania komendy
 Odczyt = „”                                'zaladowanie do zmiennej string wartosci pustej
 Do                                         'poczatek petli
   S = Inkey()                              'zapisz do zmiennej s znak odczytany z bufora odbiorczego
 Loop Until S = Chr(10)                     'zakoncz petle gdy odebrany znak ma kod ascii 13 (CR - enter)
 Do                                         'poczatek drugiej warunkowej petli do-loop
   S = Inkey()                              'zapisz do zmiennej s znak odczytany z bufora odbiorczego
   If S = „O” Or S = „K” Then               'jesli znak zapisany do s to O lub K to
     Odczyt = Odczyt + S                    'dodaj do zmiennej odczyt znak zapisany w zmiennej s
   End If
 Loop Until S = Chr(10)                     'zakoncz petle gdy odebrany znak ma kod ascii 13 (CR - enter)
 If Odczyt <> „OK” Then                     'jesli wartosc zapisana w odczyt rozna ok slowa „OK” to
   Do                                       'poczatek petli nieskonczonej do-loop
     Toggle Led                             'zmien na przeciwny stan diody LED
     Waitms 250                             'czekaj 250 ms
   Loop                                     'koniec nieskonczonej petli do-loop
 End If
End Sub                                     'koniec procedury sprawdzajacej status wykonania komendy
```

rametrem równym „2” włącza przyjmowanie i akceptację połączeń. Będą więc mogły łączyć się do niego moduły BT. Komenda „agpm...” z pierwszym parametrem równym „1” wyłącza tryb parowania modułów, który jest wykorzystywany przy bezpiecznym nawiązywaniu połączenia modułów. Komenda „agsm...” także z pierwszym parametrem równym „1” wyłącza tryb szyfrowanego przesyłania danych. Te dwa wymienione tryby są wykorzystywane w drugim przykładzie. Komenda „agmsp...” określa rolę *mastera* i *slave'a* modułu. Pierwszy parametr o wartości „0” powoduje, że moduł w nadchodzących połączeniach zawsze będzie próbował być *masterem*. Komenda „agln...” nadaje mu ją, ponieważ wszystkie moduły BT mają swoją nazwę. W tym przypadku została mu nadana nazwa „Serwer Temp”. Wysyłana nazwa musi się zawierać w cudzysłowach „”, a zostały już one użyte w instrukcji `print`, więc zostały wstawione przez zastosowanie zapisu {034}, gdzie 034 to znak ASCII cudzysłowu. Komenda „aglc...” zapisuje „COD” modułu o wartości 0. „COD” to klasa modułu (moduł może należeć do klasy audio, sieciowej itp.), który może zostać odczytany podczas przeszukiwania. Na przykład dany moduł może sprawdzić, czy znajduje się inny moduł BT, z którym może się połączyć i ma odpowiednią klasę „COD”. „COD” we wszystkich przykładach został ustawiony na wartość „0”, gdyż moduły w prezentowanych przykładach są wykorzystywane do własnego celu. Komenda „addcp...” z pierwszym parametrem równym „255” powoduje wyłączenie pracy modułu jako „klient”, natomiast komenda „addsp...” z pierwszym parametrem o wartości „0” konfiguruje moduł jako serwer. Komenda „adwm...” z pierwszym parametrem równym „0” wyłącza pracę modułu BT w trybie „MultiDrop”, z kolei drugi parametr określa sposób działania modułu w tym trybie. Komenda „accb...” z pierwszym parametrem o wartości „0” wyłącza możliwość zdalnej konfiguracji modułu (poprzez inny moduł BT). Wykonanie opisanych komend wystarczy, by moduł BT pracował jako serwer i poprawnie komunikował się z innym modułem BT. Aby przesyłać dane, należy z powrotem przejść do trybu danych. Jest to możliwe po wykonaniu komendy „addm”. Po prawidłowym wejściu w tryb danych zapalana jest dioda LED, po czym program wykonuje pętlę główną programu. Za pomocą instrukcji `input` oczekuje w niej na odbiór zapytania o temperaturę, potwierdzonego znakiem CR – np. klawiszem *Enter*. Serwer wysyła klientowi wartość zmierzonej temperatury na zapytanie składające się ze znaków „t?”. Funkcja `instr` zwraca pozycję szukanego ciągu znaków w odebranym łańcuchu, w tym przypadku szuka znaków „t?” w zmiennej `odczyt`. Jeżeli ciąg zostanie znaleziony, funkcja zwróci wartość większą od 0. W tym przypadku zostanie wywołana procedura pomiaru i obliczenia temperatury. Jest to typowa procedura odczytu temperatury z DS1820 i jej przeliczenia dla uzyskania dokładności 0,1 stopnia C, więc nie będzie tu dokładnie omawiana. Po pomiarze temperatury i odpowiednich obliczeniach jej wartość przechowuje zmienna `t`. Wartość zmiennej `t` zamieniana jest funkcją `str` na tekst. Następnie zostaje wysłany przez moduł BT tekst „Temp=” oraz zmierzona i odpowiednio sformatowana wartość temperatury (na przykład może to być: 23.5). Po wysłaniu temperatury, pętla programu powtarza się i następuje oczekiwanie na kolejne zapytanie o temperaturę. Wadą identyfikacji znaków zapytania w przedstawiony sposób – jak łatwo zauważyć – jest to, że temperatura będzie wysyłana nie tylko po odebraniu znaków „t?”, ale i dowolnych znaków przed „t?”, np. otrzymanie łańcucha „abdt?” także spowoduje wysłanie przez serwer informacji o temperaturze. Aby temu zaradzić, należy przerobić program odpowiedzialny za identyfikację zapytania. W tym przypadku nie jest to jednak uciążliwa wada. Po odpowiednim skonfigurowaniu serwera temperatury należy skonfigurować drugi moduł BT, by pracował jako klient i by było możliwe nawiązanie komunikacji z serwerem. Klient temperatury będzie otrzymywał zmierzoną temperaturę po wysłaniu zapytania oraz wyświetlał ją na wyświetlaczu LCD.

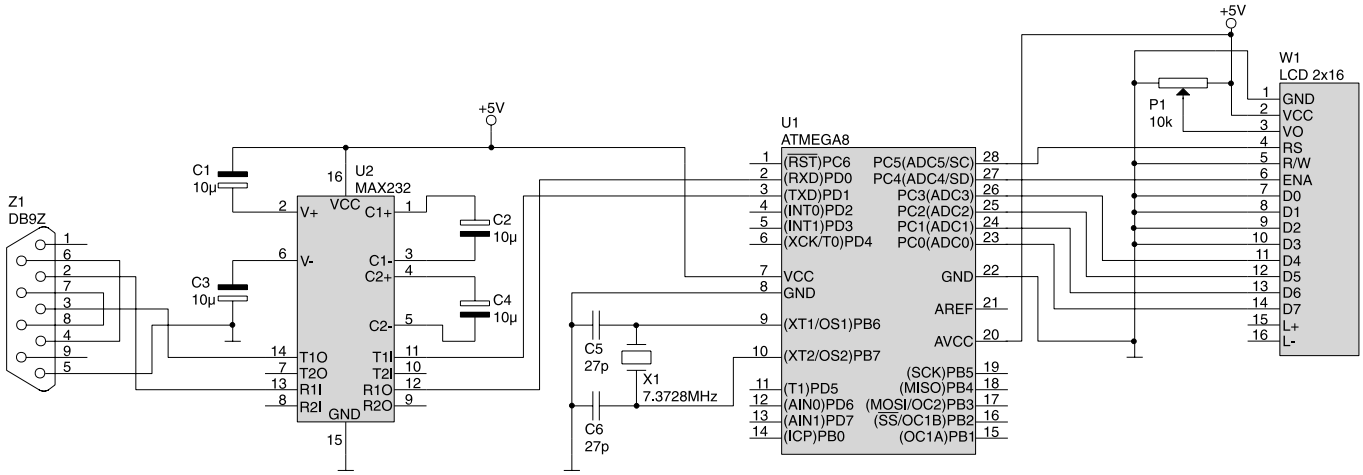
Klient temperatury

Na **rys. 2** przedstawiono schemat ideowy klienta temperatury. Układ klienta temperatury ma jedynie dołączony do mikrokontrolera wyświetlacz LCD, na którym będzie prezentowana temperatura oraz komunikaty. Podobnie jak w serwerze temperatury, układ ma także konwerter poziomów napięcia RS232. Na **list. 2** został przedstawiony program realizujący klienta temperatury.

Także i w tym przypadku wykorzystany został odbiór buforowy danych z RS232. Początkowe instrukcje w programie odpowiednio konfigurują LCD, definiują znak stopnia, wyłączają echo instrukcji `input` oraz włączają przerwania globalne. Po wyświetleniu komunikatu „Konfig BT” następuje konfigurowanie modułu. Niektóre komendy są identyczne jak w przypadku serwera temperatury, dlatego opisane zostaną tylko te, których parametry różnią się lub te, które nie występowały w przypadku serwera temperatury. Komenda „agmsp...” w tym przypadku ma parametr pierwszy nie „0”, lecz „1”. Konfiguruje on moduł tak, by pozwalał drugiej stronie w nadchodzącym połączeniu zadecydować, czy ma być *masterem*, czy *slave'em*. Procedura `sprawdz_stat` jest podobna jak w serwerze temperatury, tylko że informacja o błędzie jest wyświetlana na LCD. Komenda „agln...” nadaje modułowi klienta nazwę „Client BT”. Komenda „addcp...” wraz z pierwszym parametrem o wartości „0” konfiguruje moduł do pracy jako klient, natomiast komenda „addsp...” z wartością pierwszego parametru równą „255” wyłącza pracę modułu jako serwer. W przypadku serwera temperatury było odwrotnie, czyli komenda „addcp” miała parametr równy „255”, a „addsp...” parametr równy „0”. Pierwszy parametr komendy „adnrp...” określa, z iloma modułami BT będzie można się połączyć. W przykładzie wpisano wartość „1”, co oznacza łączenie się tylko z jednym modułem BT. Aby było możliwe połączenie klienta z serwerem, należy zapisać do modułu BT klienta dane o serwerze, z którym ma nastąpić połączenie. Dane te to adres oraz nazwa serwera. Realizuje to komenda „adwdrp...”. Pierwszy parametr to numer (identyfikator) wpisywanych danych o wybranym module serwera. Dla danych pierwszego serwera będzie to wartość „0” (numery są liczone od 0). Drugi parametr to adres modułu serwera zawierający sześć pól reprezentujących wartość szesnastkową składającą się z dwóch znaków. Trzeci parametr, o wartości „2”, określa, w jaki sposób nastąpi połączenie modułów. Do wyboru są warianty, w których: po-

List. 2. Program realizujący klienta temperatury

```
'Client odczytujacy temperature przez Bluetooth
'Przyklad programu konfigurujacego Bluetooth jako client, ktory odbiera w serwera temperatury
'zmierzona temperature
'przesyłanie danych niekodowane
'Marcin Wiazania
'marcin.wiazania@ep.com.pl

$regfile = „m8def.dat”                          'rejestry mikrokontrolera atmega8
$crystal = 7372800                              'czestotliwosc taktowania mikrokontrolera
$baud = 57600                                   'informuje kompilator o predkosci transmisji

Config Lcd = 16 * 2                             'konfiguracja organizacji znakow wyswietlacza LCD
Config Lcdpin = Pin , Db4 = Portc.3 , Db5 = Portc.2 , Db6 = Portc.1 , Db7 = Portc.0 , E = Portc.4 , Rs = Portc.5 'konfiguracja pinow mikrokontrolara do
Config Serialin = Buffered , Size = 20          'konfiguracja by interfejs rs232 uzywal przy odbiorze transmisji buforowej (bufor o wielkosci 20 znakow)
Declare Sub Sprawdz_stat                        'procedura sprawdzajaca status wykonania wyslanego polecenia at
Dim Odczyt As String * 15                       'zmienna string ktora przechowuje odczytanu status z bluetooth oraz otrzymana wartosc temperatury
Dim S As String * 1                             'pomocnicza zmienna tekstowa
Dim Licz As Integer                             'zmienna licznikowa czasu braku odpowiedzi z serwera na zapytania
Dim Czekaj As Byte                              'zmienna licznikowa czasu oczekiwania na odebranie danych o temperaturze
Deflcdchar 0 , 7 , 5 , 7 , 32 , 32 , 32 , 32 , 32  'deklaracja znaku stopnia dla wyswietlacza LCD
Echo Off                                        'wylaczenie echa instrukcji input
Enable Interrupts                               'globalne odblokowanie przerwan
Cursor Off                                      'wylaczenie kursora
Cls                                             'czysc LCD
Lcd „Konfig BT”                                 'informacja o konfiguracji Bluetooth

Wait 1                                          'czekaj 1 sekunde
Print „///”;                                    'wyslij 3x”/” bez wysylania dodatkowego kodu 13 (CR - enter) - przelacza modul BT w tryb AT z trybu danych
Wait 2                                          'czekaj 2 sekundy
Print „ate0”                                    'wyłączenie echa wysyłanych komend
Call Sprawdz_stat                               'sprawdzenie statusu wykonania wyslanej do BT komendy
Print „at*agdm=1,0”                             'modul BT nie bedzie widoczny dla innych modulow BT
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*agcm=2,0”                             'wlaczenie przyjmowania i akceptowania polaczen
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*agpm=1,0”                             'wylaczenie trybu parowania modulow
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*agsm=1,0”                             'wylaczenie bezpieczenstwa polaczen (autoryzacja i szyfrowanie)
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*agmsp=1,0”                            'modul BT w nadchodzacych polaczeniach bedzie pozwalal drugiej stronie zadecydowac, czy ma byc masterem czy slavem
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*agln={034}Client BT{034},0”           'nadaje nazwe „Client BT” modulowi BT
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*aglc=0,0”                             'zapisuje COD modułu BT
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*addcp=0,0”                            'wlaczenie profilu dla clienta (wylaczenie pracy jako serwer)
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*addsp=255,0”                          'wylaczenie profilu portu szeregowego dla serwera (praca jako client)
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*adnrp=1,0”                            'modul BT bedzie mial mozliwosc laczenia sie tylko do jednego odleglego modulu BT
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*adwdrp=0,00803719bea4,2,0,{034}Server Temp{034},0”       'wpisanie adresu modulu bt z ktorym bedzie odbywac sie komunikacja
                                                '(w tym przypadku bedzie to numer modulu serwera) oraz modul bedzie caly czas sie probowal polaczyc
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*adwm=0,0,0”                           'wylaczenie mozliwosci jednoczesnej pracy z wieloma modulami BT (wylaczenie trybu wireless MultiDrop)
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*accb=0,0”                             'wyłacza mozliwosc zdalnej konfiguracji modulu BT
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Print „at*addm”                                 'przelacza modul BT z powrotem w tryb transmisji danych        '
Call Sprawdz_stat                               'sprawdzenie statusu wykonania komendy
Wait 1                                          'czekaj 1 sekunde

Cls                                             'czysc lcd
Lcd „Czekanie na Temp”                          'wyswietla na lcd komunikat oczekiwania na odbior temperatury

Do                                              'petla glowna programu
 Odczyt = „”                                    'zapisz do zmiennej odczyt znak pusty
 Czekaj = 0                                     'zerowanie zmiennej czekaj
 Print „t?”                                     'wysyla zapytanie o temperature
 Incr Licz                                      'zwieksz zmienna licznikowa braku odpowiedzi z serwera
 Do                                             'poczatek petli do-loop
   Incr Czekaj                                  'zwieksz o jeden zmienna czekaj
   While _rs_head_ptr0 <> _rs_tail_ptr0         'petla wykonawana gdy parametry rozne
     Waitms 1                                   'czekaj 1 ms
     S = Inkey()                                'zapisz do zmiennej s pierwszy znak odczytany z bufora odbiorczego
     If S > Chr(31) Then                        'jesli znak zapisany do s ma kod ascii wiekszy niz 31 to
       Odczyt = Odczyt + S                      'dodaj do zmiennej odczyt znak zapisany w zmiennej s
     End If
     If S = Chr(13) Then                        'jesli s ma kod znaku 13 (enter) to
       Licz = 0                                 'wyzeruj zmienna licz
       Cls                                      'czysc LCD
       Lcd Odczyt ; Chr(0) ; „C”                'wyswietl na LCD otrzynana z serwera wartosc temperatury z dodatkowym znakiem stopnia i znakiem C
       Exit Do                                  'opusc petle do-loop
     End If
   Wend                                         'koniec petli while
   Waitms 100                                   'czekaj 100 ms
  Loop Until Czekaj = 30                        'wykonuje petle do-loop az czekaj=30 (uplynie ok. 3 sekundy)
 If Licz = 3 Then                               'jesli licz=3 to
   Cls                                          'czysc LCD
   Lcd „Brak komunikacji”                       'wyswietl komunikat braku otrzymania temperatury po 6 sekundach od czasu wyslania rozkazu jej przeslania
   Licz = 0                                     'wyzeruj zmienna licz
 End If
Loop                                            'koniec petli glownej programu
End                                             'koniec programu

Sub Sprawdz_stat                                ' procedura sprawdzania statusu wykonania komendy
 Odczyt = „”                                    'zaladowanie do zmiennej string wartosci pustej
 Do                                             'poczatek petli
   S = Inkey()                                  'zapisz do zmiennej s znak odczytany z bufora odbiorczego
 Loop Until S = Chr(10)                         'zakoncz petle gdy odebrany znak ma kod ascii 13 (CR - enter)
 Do                                             'poczatek drugiej warunkowej petli do-loop
   S = Inkey()                                  'zapisz do zmiennej s znak odczytany z bufora odbiorczego
   If S = „O” Or S = „K” Then                   'jesli znak zapisany do s to O lub K to
     Odczyt = Odczyt + S                        'dodaj do zmiennej odczyt znak zapisany w zmiennej s
   End If
 Loop Until S = Chr(10)                         'zakoncz petle gdy odebrany znak ma kod ascii 13 (CR - enter)
 If Odczyt <> „OK” Then                         'jesli wartosc zapisana w odczyt rozna ok slowa „OK” to
   Cls                                          'czysc LCD
   Lcd „Blad komendy”                           'wyswietl na LCD komunikat „Blad komendy”
   Do                                           'poczatek petli nieskonczonej do-loop
   Loop                                         'koniec nieskonczonej petli do-loop
 End If
End Sub                                         'koniec procedury sprawdzajacej status wykonania komendy
```



Rys. 2. Schemat ideowy klienta temperatury

łączenie nastąpi po wykryciu przesyłania danych, moduł zawsze będzie próbował się połączyć oraz po wykryciu zewnętrznego sygnału na odpowiednim pinie modułu BT. Metody połączenia można także łączyć ze sobą razem. Wpisanie wartości „2" powoduje, że moduł klienta cały czas będzie się próbował połączyć z serwerem temperatury. Czwarty parametr o wartości „0" spowoduje, że klient będzie się łączył tylko z jednym określonym serwerem i z żadnym innym. Przedostatnim parametrem jest nazwa serwera, która także musi być zapisana w cudzysłowach. W tym przypadku również został wykorzystany zapis {034}, który wstawia dodatkowe cudzysłowy. Adres modułu BT, z którym będzie nawiązywane połączenie, jak i jego nazwę można odczytać zdalnie poprzez wykonanie odpowiedniej komendy. W tym przypadku moduł serwera musi mieć włączoną opcję widoczności dla innych modułów BT. W przykładach ta możliwość nie jest wykorzystywana, lecz adres oraz nazwa serwera są znane i wpisane na stałe w programie. Adres oraz nazwę serwera można także odczytać z „listy ulubionych", jeśli wcześniej zostały na nią wpisane. To już wystarczy, by moduł po wejściu w tryb przesyłania danych nawiązał połączenie z serwerem, którego dane zostały wpisane podczas konfiguracji, czyli nastąpi połączenie z serwerem temperatury.

Po skonfigurowaniu modułu i wyświetleniu odpowiedniego komunikatu na LCD, program przechodzi do wykonywania pętli głównej. Na początku tej pętli czyszczona jest zmienna `odczyt` oraz `czekaj`, która liczy czas oczekiwania na temperaturę od serwera. Wysyłane jest także zapytanie o temperaturę do serwera, czyli znaki „t?" zakończone znakiem CR. Program główny został napisany w taki sposób, by nie dochodziło do jego wstrzymywania. Odliczane jest opóźnienie, po którym w przypadku braku przesłania od serwera temperatury wyświetlany jest stosowny komunikat. Komunikat o braku połączenia z serwerem jest wyświetlony dopiero po trzech nieudanych wysłaniach zapytania o temperaturę. Zmienna `licz` liczy nieudane zapytania (zapytania bez odpowiedzi serwera temperatury). W pętli `do-loop`, która jest wykonywana aż `czekaj` osiągnie wartość 30, zwiększana jest zmienna `czekaj` o jeden. Pętla `while` jest wykonywana dotąd, aż parametry `_rs_head_ptr0` i `_rs_tail_ptr0` (tworzone przy użyciu buforowego przesyłania danych) są różne, czyli gdy w buforze odbiorczym RS232 znajdują się nieodebrane znaki. Jeśli tak jest, do zmiennej `s` ładowane są odczytywane znaki i jeżeli odebrany znak ma kod ASCII większy od 31, to odczytany znak jest dodawany do zmiennej `odczyt`. W ten sposób są ignorowane znaki o kodach ASCII mniejszych niż 32. Gdy zostanie odebrany znak końca wysyłanej przez serwer temperatury (znak CR), zerowana jest zmienna `licz` oraz na LCD zostaje wyświetlona otrzymana temperatura wraz z dodatkowymi znakami stopnia i znaku „C". Po wykonaniu instrukcji `exit do` nastąpi opuszczenie wewnętrznej pętli `do-loop`. Teraz program ponownie wyśle zapytanie o nową wartość temperatury. Gdy po wysłaniu zapytania przez ok. 3 sekundy nie otrzyma się wartości temperatury od serwera, zmienna `licz` zostaje zwiększona o jeden oraz ponownie zostaje wysłane do serwera zapytanie. Pętla wewnętrzna `do-loop` odlicza 100 ms opóźnienia, gdy nie odebrano żadnych danych z serwera. Zliczenie 30 takich opóźnień powoduje opuszczenie pętli – jest to równoznaczne z odliczeniem 3 sekund. Próba wysłania trzech zapytań o temperaturę zakończonych niepowodzeniem spowoduje, że zmienna `licz` przyjmie wartość 3 i zostanie wyświetlony komunikat o braku łączności z serwerem. Po wyświetleniu tegoż komunikatu zmienna `licz` jest zerowana, ale działanie programu nie jest wstrzymywane i program nadal będzie wysyłał zapytania o temperaturę do serwera. Jeśli ją otrzyma, to zostanie ona wyświetlona na LCD.

Jak widać na przykładzie pętli głównej powyższego programu, nie jest trudno napisać program, który nie będzie wstrzymany oczekiwaniem na dane z serwera temperatury (oczekiwaniem na dane z RS232), lecz będzie mógł reagować po odpowiednim czasie na brak oczekiwanych informacji. Przekazywane informacje w tym przykładzie pomiędzy modułami BT nie były utajnione. Dlaczego by nie skorzystać z tej możliwości w urządzeniach, w których byłoby to zaletą, przecież moduły BT firmy ConnectBlue oferują funkcję autoryzacji oraz szyfrowania transmitowanych danych. Funkcje te zostaną wykorzystane w drugim przykładzie.

Marcin Wiązania, EP
marcin.wiazania@ep.com.pl