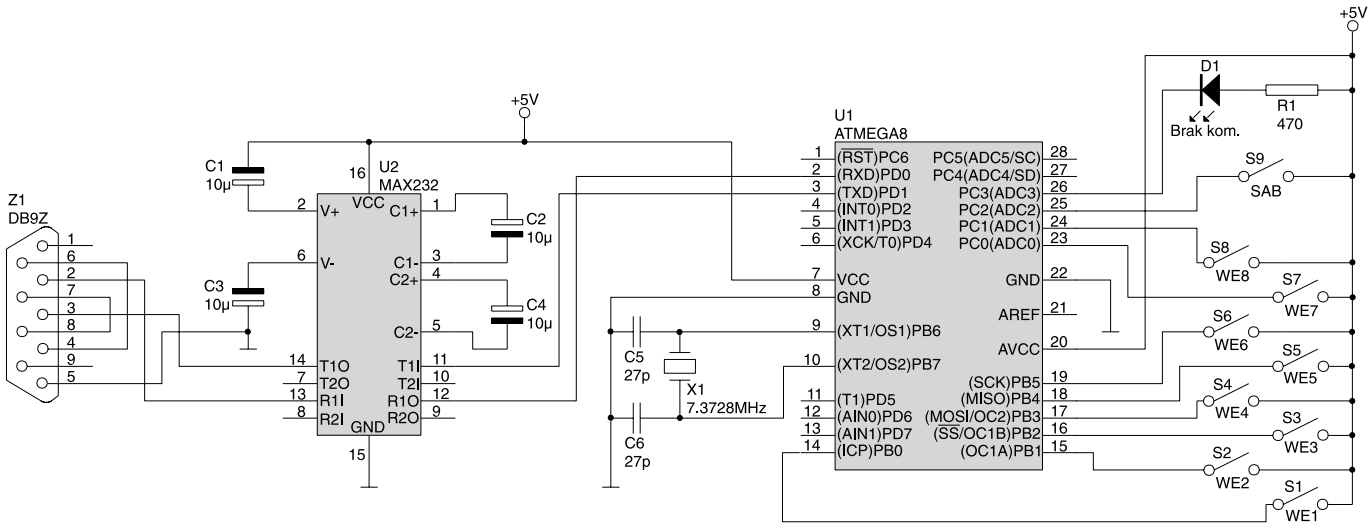
Bluetooth łączy mikrokontrolery, część 2

Bluetooth (w skrócie BT) zdobywa coraz większą popularność w dziedzinie bezprzewodowego przesyłania danych na odległość. Wykorzystuje do tego częstotliwość 2,4 GHz. O popularności BT świadczy coraz częstsze jego występowanie w urządzeniach powszechnego użytku. BT można spotkać nie tylko w komputerach, mamy go w komórkach i wielu innych urządzeniach.

Przykład systemu bezprzewodowego nadajnika oraz odbiornika sygnałów z czujek alarmowych

Prosty system pokazany w tym przykładzie wykorzystuje bezprzewodowe przesyłanie sygnałów z ośmiu oddalonych czujek alarmowych do centrali alarmowej. Sygnały z czujek mogą być przesyłane jako logiczne „0” lub „1”. Do nadajnika może być dołączonych maks. 8 czujek oraz jedna linia sabotażowa. Odbiornik będzie wystawiał na swoich wyjściach identyczne sygnały, jakie wystawiają czujniki dla nadajnika. Wyjścia te można dołączyć do wejść centralki alarmowej. Taki system mógłby być zastosowany, np. w przypadku domku wielopiętrowego, w którym nie ma możliwości przeciągnięcia wielu przewodów. Stosując opisywany system, przewody od czujek należałoby poprowadzić tylko w ramach jednego piętra, piwnicy lub parteru. Do realizacji takiego prostego systemu znakomicie mogą się nadawać moduły BT. Ich dużą zaletą jest wysoka odporność na zakłócenia, co jest bardzo istotne w podobnych aplikacjach. Dodatkowym atutem przy tego typu rozwiązaniu będzie skorzystanie z możliwości autoryzacji i szyfrowania danych, jakie oferują moduły BT firmy ConnectBlue. W opisywanym przykładzie nadajnik sygnałów z czujek będzie pracował z modułem BT skonfigurowanym jako klient, a odbiornik z modułem pracującym jako serwer. W przypadku wykorzystywania funkcji „Multidrop” będzie możliwa jednoczesna praca z wieloma nadajnikami. Jak już pisałem, w tym systemie zostały wykorzystane dodatkowe funkcje modułów BT związane z autoryzacją połączenia i szyfrowaniem danych. Aby była możliwa bezpieczna autoryzacja, a przesyłane dane były szyfrowane, należy włączyć tryby szyfrowania danych w obu komunikujących się modułach BT. Dodatkowo, przy procedurze autoryzacji wymagany jest klucz (pin), który powinien być identyczny dla obu komunikujących się ze sobą modułów. Przy wykorzystaniu funkcji modułów związanych z bezpieczeństwem należy „związać” ze sobą moduły, które będą komunikować się ze sobą (należy zrobić z nich parę). W tym celu, w komunikujących się modułach powinny zostać włączone funkcje parowania modułu. Aby dokonać związania modułów, należy w jednym module wysłać komendę tworzenia kontaktu (pary), a następnie komendy, które umożliwią utworzenie połączenia (tak, jak to było w kliencie temperatury z poprzedniego przykładu). Na **rys. 3** został przedstawiony schemat ideowy nadajnika sygnałów z czujek. Przyciski na schemacie reprezentują sygnały z czujek alarmowych. Przycisk S9 symbolizuje dodatkową linię sabotażową. Dioda LED sygnalizuje, tak jak to było w przypadku serwera temperatury, błąd wykonania komendy (dioda miga) oraz prawidłowość wykonania wszystkich wysyłanych komend (dioda świeci światłem ciągłym). W tym przykładzie moduł BT został również dołączony poprzez konwerter poziomów RS232.

Na **list. 3** został przedstawiony program w języku Bascom sterujący nadajnikiem pracującym jako klient.



Rys. 3. Schemat ideowy nadajnika sygnałów z czujek alarmowych

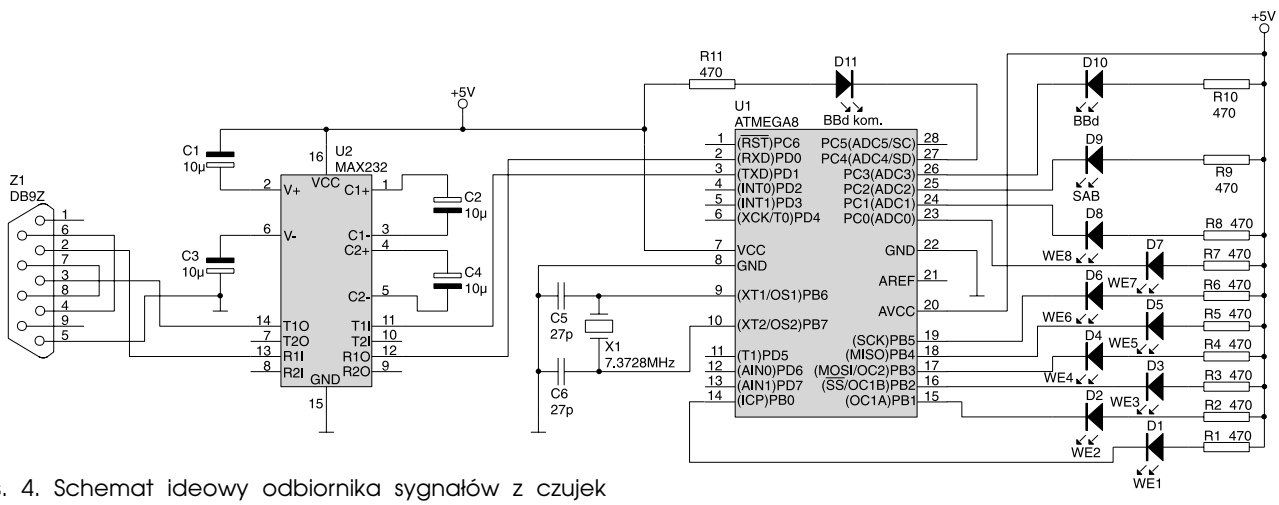
List. 3. Program sterujący nadajnikiem sygnałów z czujek alarmowych

```
'Przykład programu nadajnika radiowego sygnałów z czujek do centrali alarmowej z wykorzystaniem Bluetooth
'Mozliwosc nadawania sygnalow z 8 czujek oraz jednej linii sabotazowej
'Modul Bluetooth skonfigurowany do pracy jako klient
'Transmisja danych pomiedzy nadajnikiem a odbiornikiem kodowana
'stan 8 linii nadajnika jest przesylany w formacie wxxxxxxxx gdzie x to stany (0 lub 1) poszczegolnych wejsc nadajnika
'stan linii sabotazowej jest przesylany w formacie sx gdzie x to stan (0 lub 1) linii sabotazowej nadajnika
'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 Portb = Input                                    'linie portu b jako wejscia
Config Portc = &B00001000                               'linie portu c.3 jako wyjscie, pozostale linie jako wejscia
Config Serialin = Buffered , Size = 10                  'konfiguracja by interfejs rs232 uzywal przy odbiorze transmisji buforowej
                                                        '(bufor o wielkosci 10 znakow)

Declare Sub Sprawdz_stat                                'procedura sprawdzajaca status wykonania wyslanego polecenia at

Dim Odczyt As String * 10                               'zmienna string ktora przechewuje odczytanu status z bluetooth oraz dane odebrane
                                                        'od nadajnika
Dim S As String * 1                                     'pomocnicza zmienna tekstowa
Dim Wej As Byte                                         'zmienna przechowuje wartosc stanu wyjsc czujek dolaczonych do nadajnika

Br_kom_l Alias Portc.3                                  'przypisanie aliasu br_kom_l linii pc.3
Sab Alias Pinc.2                                        'przypisanie aliasu sab do rejestru odbiorczego linii pc.2

Portb = Portb Or &B00111111                             'do linii wejsciowych 0..5 dolaczone zostana rezystory podciagaja ce
Portc = Portc Or &B00001111                             'linie 0..3 portu pc w stanie wysokim
                                                        '(do linii wejsciowych 0..2 dolaczone zostana rezystory podciagajace)
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=2,0”                                     'wlaczenie trybu parowania modulow
Call Sprawdz_stat                                       'sprawdzenie statusu wykonania komendy
Print „at*agsm=2,0”                                     'wlaczenie 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*agfp={034}1199{034},0”                        'zapis pinu 1199 uzywanego podczas zwiazku
Call Sprawdz_stat                                       'sprawdzenie statusu wykonania komendy
Print „at*agln={034}Nadajnik{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*agub=00803719bea4”                            'wykasowanie adresu urzadzenia BT z ktorym bedzie odbywac sie zwiazek
Call Sprawdz_stat                                       'sprawdzenie statusu wykonania komendy
Print „at*agfp?”                                        'odczyt zapisanego wczesniej pinu (pin:1199)

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)

Call Sprawdz_stat                                       'sprawdzenie statusu wykonania komendy
Print „at*agb=00803719bea4”                             'zapisanie adresu urzadzenia BT z ktorym bedzie odbywac sie zwiazek
Call Sprawdz_stat                                       'sprawdzenie statusu wykonania komendy
Print „at*adwdrp=0,00803719bea4,3,0,{034}Odbiornik{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 a takze gdy beda wysylane dane
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

Do                                                      'petla glowna programu
  Wej = Pinb And &B00111111                             'zapisz do temp wartosc stanu 6 najmlodszych linii portu b
  Wej.6 = Pinc.0                                        'zapisz do bitu 6 (0..7) zmiennej wej stan linii pc.0
  Wej.7 = Pinc.1                                        'zapisz do bitu 7 (0..7) zmiennej wej stan linii pc.1
  Print „w” ; Wej                                       'wyslij znak w oraz wartosc zmiennej wej (stan wyjsc 8 czujek)
  If Sab = 1 Then                                       'jesli linia wejsciowa sab (pc.2) = 1 to
    Print „s1”                                          'wyslij do odbiornika znaki „s1”
  Else                                                  'w przeciwnym razie
    Print „s0”                                          'wyslij do odbiornika znaki „s0”
  End If
  Waitms 50                                             'czekaj 50 ms
Loop                                                    'koniec petli glownej programu
End
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 Br_kom_l                                   'zmien na przeciwny stan diody br_kom_l
      Waitms 250                                        'czekaj 250 ms
    Loop                                                'koniec nieskonczonej petli do-loop
  End If
End Sub                                                 'koniec procedury sprawdzajacej status wykonania komendy
```

Na początku programu zostają odpowiednio skonfigurowane linie portów. W tym przypadku zostanie również wykorzystana odbiorcza transmisja buforowa z RS232. Procedura `sprawdz_stat` jest identyczna, jak w programie serwera temperatury. Ponieważ moduł BT w nadajniku zostanie skonfigurowany jako klient, niektóre wysyłane komendy będą identyczne, jak w kliencie temperatury. W tym przypadku pojawią się jednak nowe, dodatkowe komendy. Poniżej zostaną podane jedynie nowe komendy oraz różnice w komendach umożliwiających włączenie bezpieczeństwa komunikacji. W komendzie „agpm...” pierwszy parametr ma tym razem wartość „2”, co oznacza włączenie trybu parowania modułów. Identycznie jest z następną komendą „agsm...”, która z parametrem także równym „2” włącza tryb bezpieczeństwa połączeń i przesyłanych danych. Komenda „agap...” zapisuje w module pin o wartości „1199”, który umieszczony jest w cudzysłowach. Piny zapisane w modułach powinny być identyczne. Pin może się składać maksymalnie z 16 cyfr. Aby możliwe było sparowanie modułów, należy usunąć komendą „agub...” (której parametrem jest adres drugiego modułu) adres modułu, z którym będzie związany (sparowany). Następnie przed wykonaniem parowania wymagane jest odczytanie (zapisanego wcześniej) pinu komendą „agft?...”. Instrukcje po komendzie odczytu pinu służą do zignorowania otrzymanych informacji o pinie, gdyż w tym przypadku nie są one do niczego potrzebne. Instrukcje te odbierają znaki, aż do otrzymania znaku LF (kod ASCII 10). Dopiero po wykonaniu odczytu pinu można wysłać komendę parowania modułów. Parowanie modułów odbywa się po wysłaniu komendy „agb...” z adresem modułu, z którym będzie parowany. Po wykonaniu pozostałych, znanych już komend, podczas nawiązywania połączenia będzie przeprowadzana autoryzacja, a przesyłane dane będą szyfrowane. Oczywiście podczas wykonywania komend związanych z parowaniem wymagane jest, by był już odpowiednio skonfigurowany moduł z którym będzie odbywać się parowanie. Moduł odbiornika musi więc już być skonfigurowany. W opisywanym programie drugi parametr komendy „adwdrp...” został ustawiony na wartość „3”, czyli połączenie modułów będzie odbywało się, kiedy to możliwe oraz gdy zostanie rozpoznane przesyłanie danych. Także temu modułowi została nadana nazwa „Nadajnik”. Program główny odczytuje dane z linii wejściowych i wysyłała je do odbiornika. Stany linii czujek są poprzedzone znakiem „w”, a stan linii sabotażowej znakiem „s”. Każdy bit wartości wysłanej po znaku „w” odpowiada danemu wyjściu dołączonej czujki. Badana jest tylko jedna linia sabotażowa, więc w jej przypadku po znaku „s” może być wysłana wartość 0 lub 1. Wysyłanie danych o stanie wejść nadajnika odbywa się w nieskończonej pętli, co ok. 50 ms.

Na **rys. 4** został przedstawiony schemat ideowy odbiornika sygnałów z czujek alarmowych. Tu także moduł BT został dołączony poprzez konwerter MAX232. Odebrane bezprzewodowo stany sygnałów z czujek są sygnalizowane przez diody D1..D8. Dioda D9 sygnalizuje stan linii sabotażowej. Dioda D10 sygnalizuje brak komunikacji z nadajnikiem (nie są otrzymywane dane z nadajnika), natomiast dioda D11 sygnalizuje swym miganiem, tak jak w przypadku nadajnika, błąd wykonania komendy, a stałym świeceniem poprawne skonfigurowanie modułu BT.

Na **list. 4** został przedstawiony program w języku Bascom sterujący odbiornikiem. Na początku programu konfigurowane zostają odpowiednio linie portów mikrokontrolera. I w tym przypadku zostanie również wykorzystana odbiorcza transmisja buforowa z RS232. Program odbiornika konfiguruje moduł BT do pracy jako serwer i posiada komendy konfiguracyjne, podobne jak w przypadku „serwera temperatury” z poprzedniego przykładu. W tym przypadku zostały one jednak uzupełnione o komendy związane z funkcjami bezpieczeństwa. Tak jak w nadajniku, tak i w odbiorniku (module pracującym jako serwer), zostaje włączony tryb parowania, tryb bezpieczeństwa oraz zostaje zapisany identyczny pin, jak w module nadajnika o wartości „1199”. Wystarczy to, by w przypadku tego modułu wykorzystać funkcje bezpieczeństwa. Resztą zajmuje się moduł BT nadajnika. Modułowi BT odbiornika zostaje nadana nazwa „Odbiornik”. Program główny składa się z nieskończonej pętli, w której druga, wewnętrzna pętla zostanie wykonana 5000 razy, jeżeli nie zostaną odebrane żadne dane od nadajnika. Pętla ta zakończy swe działanie po ok. 5 sekundach, gdyż znajduje się w niej 1 ms opóźnienie, po którym inkrementowana jest wartość zmiennej `licz`. Jeżeli w ciągu 5 sekund nie zostaną odebrane żadne dane od nadajnika, włączana jest dioda D10, sygnalizująca brak komunikacji z nadajnikiem. W przypadku ciągłego odbierania danych z nadajnika zmienna `licz` jest okresowo zerowana, więc



Rys. 4. Schemat ideowy odbiornika sygnałów z czujek

List. 4. Program sterujący odbiornikiem sygnałów z czujek alarmowych

```
'Przykład programu odbiornika radiowego sygnałów z czujek do centrali alarmowej z wykorzystaniem Bluetooth
'Mozliwosc odbioru sygnalow z 8 czujek oraz jednej linii sabotazowej
'Modul Bluetooth skonfigurowany do pracy jako serwer
'Transmisja danych pomiedzy nadajnikiem a odbiornikiem kodowana
'stan 8 linii nadajnika jest przesylany w formacie wxxxxxxxx gdzie x to stany (0 lub 1) poszczegolnych wejsc nadajnika
'stan linii sabotazowej jest przesylany w formacie sx gdzie x to stan (0 lub 1) linii sabotazowej nadajnika
'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 Portb = &B00111111                   'linie 0..5 portu pb jako wyjscia
Config Portc = &B00011111                   'linie 0..4 portu pc jako wyjscia
Config Serialin = Buffered , Size = 10      'konfiguracja by interfejs rs232 uzywal przy odbiorze transmisji buforowej (bufor o wielkosci 10 znakow)

Declare Sub Sprawdz_stat                    'procedura sprawdzajaca status wykonania wyslanego polecenia at do modulu BT

Dim Odczyt As String * 10                   'zmienna string ktora przechewuje odczytanu status z bluetooth oraz dane odebrane od nadajnika
Dim S As String * 1                         'pomocnicza zmienna tekstowa
Dim Licz As Integer                         'zmienna licznikowa czasu braku odpowiedzi od nadajnika
Dim Param As String * 1                     'przechowuje odebrany znak identyfikujacy przesylany parametr
Dim Wej As Byte                             'zmienna przechowuje odebrana wartosc stanu wyjsc czujek dolaczonych do nadajnika
Dim Sab As Byte                             'zmienna przechowuje stan linii sabotazowej

Sab_1 Alias Portc.2                         'przypisanie aliasu sab_1 linii pc.2
Blad_1 Alias Portc.3                        'przypisanie aliasu blad_1 linii pc.3
Br_kom_1 Alias Portc.4                      'przypisanie aliasu br_kom_1 linii pc.4

Portc = Portc Or &B00011111                 'linie 0..4 portu pc w stanie wysokim
Portb = Portb Or &B00111111                 'linie 0..5 portu pb w stanie wysokim

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=2,0”                         'wlaczenie trybu parowania modulow
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agsm=2,0”                         'wlaczenie 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*agfp={034}1199{034},0”            'zapis pinu 1199 uzywanego podczas zwiazku
Call Sprawdz_stat                           'sprawdzenie statusu wykonania komendy
Print „at*agln={034}Odbiornik{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 Blad_1                                'zapala diode LED

Licz = 0                                    'wyzerowanie zmiennej licz

Do                                          'poczatek petli glownej programu
  Do                                        'druga wewnetrzna petla do-loop
    While _rs_head_ptr0 <> _rs_tail_ptr0    'petla while wykonawana dotad dokad parametry w warunku IF sa rozne
      S = Inkey()                           'zapisz do zmiennej s znak odczytany z bufora odbiorczego
      If S = „w” Or S = „s” Then            'jesli odebrany znak w lub s to
        Param = S                           'zapisz odebrany znak w zmiennej param
      Elseif 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)
        If Param = „w” Then                 'oraz zmienna param posiada zapisany znak w to
          Wej = Val(odczyt)                 'zapisz do zmiennej wej przetworzona na postać dziesiętną wartosc zmiennej tekstowej odczyt
          Portb = Wej And &B00111111        'zapisz do portub (linie 0..5) wartosc zmiennej wej z wyzerowanymi dwo najstarszymi bitami
          Portc.0 = Wej.6                   'zapisz stan bitu 6 (0..7) zmiennej wej do linii wyjsciowej pc0
          Portc.1 = Wej.7                   'zapisz stan bitu 7 (0..7) zmiennej wej do linii wyjsciowej pc1
        Elseif Param = „s” Then             'w przeciwnym wypadku jesli zmienna param ma zapisany znak „s” to
          Sab = Val(odczyt)                 'zapisz do zmiennej sab przetworzona na postać dziesiętną wartosc zmiennej tekstowej odczyt
          Sab_1 = Sab                       'zapis stan zmiennej sab do sab_1 ktory wskazuje na linie c.2
        End If
        Set Br_kom_1                        'ustaw linie br_kom_1
        Odczyt = „”                         'zaladowanie do zmiennej string wartosci pustej
        Licz = 0                            'wyzerowanie zmiennej licz
      End If
    Wend                                    'koniec petli while
    Waitms 1                                'czekaj 1 ms
    Incr Licz                               'zwieksz o jeden wartosc zmiennej licz
  Loop Until Licz = 5000                    'wykonuje petle do-loop az licz=5000 (uplynie ok 5 sekund)
  Reset Br_kom_1                            'zeruj linie portu wskazywana przez br_kom_1
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
    Do                                      'poczatek petli nieskonczonej do-loop
      Toggle Blad_1                         'zmien na przeciwny stan diody blad_1
      Waitms 250                            'czekaj 250 ms
    Loop                                    'koniec nieskonczonej petli do-loop
  End If
End Sub                                     'koniec procedury sprawdzajacej status wykonania komendy
```

nie osiągnie wartości 5000 i pętla się nie zakończy. Takie rozwiązanie nie wstrzymuje działania programu, który może zajmować się innymi zadaniami oraz umożliwia sygnalizację braku danych z nadajnika nie od razu, lecz po określonym czasie. Jeśli zmienne pomocnicze `_rs_head_ptr0` oraz `_rs_tail_ptr0` są różne, to w buforze znajdują się odebrane znaki z RS232. Znaki odbierane są w pętli `while`, aż do zrównania się wartości tych zmiennych pomocniczych. Jeśli pierwszym odebranym znakiem jest „w” lub „s”, to jeden z nich jest zapisywany w zmiennej `param`. W przeciwnym przypadku, jeśli odebrany znak ma kod ASCII większy od 31, to znak jest dodawany do zmiennej `odczyt`. W zmiennej tej znajdą się wartości stanów wejść nadajnika (wyjść czujek), które będą zapisane na pozycjach bitowych zmiennej typu bajt. Także w przypadku linii sabotażowej do zmiennej `odczyt` zostanie wpisana wartość „0” lub „1”. Jeżeli odebranym znakiem jest CR, oznacza to, że napotkano na koniec nadawanych w danym momencie informacji z nadajnika. Jeśli tak jest i jeśli wcześniej do zmiennej `param` został zapisany znak „w”, to znaczy, że odebrano informację o stanie wyjść czujek. Zapisana w zmiennej `odczyt` wartość tekstowa stanu linii czujek jest poprzez funkcję `val` zamieniana na postać cyfrową i wysyłana odpowiednio na linie wyjściowe portów PB i PC, do których zostały dołączone diody sygnalizujące stan linii czujek. Jeśli zapisanym znakiem w zmiennej `param` był znak „s”, to oznacza, że otrzymano informację o stanie linii sabotażowej. Otrzymany stan linii sabotażowej także jest zamieniany na postać cyfrową oraz wykorzystany do sterowania linii z dołączoną diodą D9 (SAB). Po otrzymany znaku CR i odpowiednim zinterpretowaniu otrzymanych danych, czyszczone są zmienne `odczyt` oraz `licz`. W tym przypadku procedura odbioru danych i ich interpretacji także okazała się dość prosta.

Oczywiście przedstawiony system można rozbudować o nowe możliwości. Przykład ten miał pokazać sposób skonfigurowania modułów BT w celu uzyskania pewnego bezpieczeństwa przesyłania danych. Przedstawione przykłady systemów można także sprawdzić w działaniu bez modułów BT łącząc układy przewodem RS232. Należy wtedy z programów usunąć instrukcje konfiguracyjne przeznaczone dla modułów BT. Przykłady te niewątpliwie pokazały, że zastosowanie coraz szybciej rozwijającego się i coraz bardziej popularnego systemu Bluetooth nie stwarza specjalnych problemów we własnych zastosowaniach. Z pewnością Bluetooth może znaleźć wiele niebanalnych zastosowań nie tylko w profesjonalnych urządzeniach, ale również amatorskich, przeznaczonych do codziennego, domowego użytku. Przykładowo mogą to być urządzenia służące do zdalnego sterowania światłem itp. Podpierając się pierwszym, przedstawionym przykładem można wykonać np. termostat ze zdalnym czujnikiem temperatury. Bluetooth zapewne już wkrótce stanie się czymś tak normalnym, jak teraz jest USB.

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