Programowanie modułu Tibbo EM1206

W kwietniowym numerze Elektroniki Praktycznej rozpoczęliśmy naukę programowania modułu ethernetowego Tibbo EM1206. W tym odcinku zajmiemy się połączeniem płyty ewaluacyjnej em1206 z siecią bezprzewodową.

Listing 1. Deklaracja linii używanych do współpracy z modułem

```
include „global.tbh”

,poniższa deklaracja pinów jest przystosowana do mozaiki ścieżek
,na płycie em1206ev, w układzie docelowym może zostać zmieniona
Const WLN_RST=PL_IO_NUM_11
Const WLN_CS=PL_IO_NUM_15
Const WLN_DI=PL_IO_NUM_12
Const WLN_DO=PL_IO_NUM_13
Const WLN_CLK=PL_IO_NUM_14

‘nazwa SSID Access Pointa, z którym zamierzamy się połączyć
const AP_NAME=”TIBBO”
‘klucz WEP:
‘-pusty brak zabezpieczenia
‘-10 znaków HEX, klucz 40bit (potocznie 64bit)
‘-26 znaków HEX, klucz 104bit (potocznie 128)
const WEP_KEY=””
dim error as no_yes
```

Listing 2. Inicjalizacja systemu

```
sub on_sys_init()
        dim x,y as byte   ‘ zmienne
        dim task as pl_wln_tasks
        dim s as string

        error=NO  ‘flaga błedu
‘sprawdzenie czy wcześniej zadeklarowano
,linie SPI dla GA1000
        if WLN_RST=PL_IO_NULL or WLN_CS=PL_IO_NULL or WLN_DI=PL_IO_NULL or
WLN_DO=PL_IO_NULL or WLN_CLK=PL_IO_NULL then
                sys.halt
                error=YES ‘ błąd sprawdź deklaracje
        end if
,alokowanie buforów
        sock.rxbuffrq(4)
        sock.txbuffrq(4)
        wln.buffrq(6)
        sys.buffalloc
,restart GA1000
        io.num=WLN_RST
        io.enabled=YES
        io.state=LOW
        io.state=HIGH
‘mapowanie zadeklarowanych wcześniej linii
‘SPI dla GA1000
        wln.csmap=WLN_CS
        io.num=WLN_CS
        io.enabled=YES
        wln.dimap=WLN_DI
        wln.domap=WLN_DO
        io.num=WLN_DO
        io.enabled=YES
        wln.clkmap=WLN_CLK
        io.num=WLN_CLK
        io.enabled=YES
‘konfiguracja socketu
        sock.num=0        ‘ numer socketu
‘protokół komunikacji
        sock.protocol=PL_SOCK_PROTOCOL_TCP
,nr portu
        sock.localportlist=”1001”
, dozwolony interfejs
        sock.allowedinterfaces=”WLN”
‘ pozwolenie na przyjmowanie zapytań: od wszystkich
        sock.inconmode=PL_SOCK_INCONMODE_ANY_IP_ANY_PORT    sock.reconmode=PL_
SOCK_RECONMODE_3
‘konfiguracja interfejsu WiFi (GA1000)
        wln.mac=”0.0.102.103.104.2”   ‘przypisanie adresu MAC
        wln.ip=”192.168.0.86”   ‘przypisanie adresu IP
        wln.gatewayip=”192.168.0.1”   ‘brama sieciowa
        wln.netmask=”255.255.255.0”   ‘maska sieci
        wln.domain=PL_WLN_DOMAIN_EU   ‘wybór częstotliwości
```

Do realizacji połączenia z siecią WiFi użyjemy dedykowanego układu interfejsu radiowego WiFi (802.11b/g) oznaczonego GA1000. Komunikacja z nim odbywa się poprzez SPI, zapewniający dużą prędkość wymiany informacji pomiędzy modułami. Należy zaznaczyć, że GA1000 współpracuje również z innymi modułami ethernetowymi z oferty Tibbo, m.in. EM1000, EM1202 i EM1206 oraz stanowi rozwinięcie poprzedniego modelu WA1000 (802.11b).

Producent zadbał, aby konfigurowanie i obsługa modułu nie przyprawiały o ból głowy. Dzięki dostępności gotowych obiektów i metod, ten wysiłek jest ograniczany do niezbędnego minimum.

Sam moduł radiowy GA1000 ma niewielkie wymiary (42×19×6,7 mm), co umożliwia stosowanie go w urządzeniach o niewielkich rozmiarach oraz mobilnych. Wbudowana antena pozwala na samodzielną pracę układu, a ponadto złącze I-PEX MHF umożliwia dołączenie zewnętrznej anteny i wielokrotne zwiększenie zasięgu.

Do komunikacji między modułem em1206 a interfejsem GA1000 potrzebnych jest tylko 5 linii I/O. Linie te mogą być wybrane z dostępnych linii ogólnego przeznaczenia modułu nadrzędnego.

Do zasilania modułu radiowego jest wymagane napięcie stałe 3,3 V ±5%. Maksymalny pobór prądu wynosi około 280 mA podczas nadawania i 200 mA podczas utrzymywania łączności z innym urządzeniem WiFi. Moc transmisji można ustawiać programowo, a nawet całkowicie wyłączyć układ

Dla aplikacji przemysłowych ważna jest również temperatura otoczenia, która zgodnie z danymi katalogowymi może zawierać się w przedziale –20°C...+70°C, przy względnej wilgotności 10...90%.

Interfejs WiFi GA1000 może współpracować zarówno z sieciami korpo-

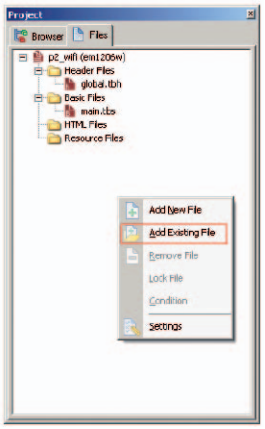
Listing 3. Otwarcie pliku konfiguracji sprzętowej oraz dalsze nastawy modułu Tibbo em1250

```
'włączenie interfejsu WiFi (GA1000)
        romfile.open("ga1000fw.bin")
        if wln.boot(romfile.offset)<>OK then
                sys.halt
                error=YES   'komunikat: sprawdzić połączenie
        end if
'ustawienie zabezpieczeń, szyfrowanie WEP
'funkcja rozpoznaje tryb szyfrowania po długości klucza
        select case len(WEP_KEY)
        case 0:
                ,WEP disabled
        case 10:
                x=wln.setwep(WEP_KEY,PL_WLN_WEP_MODE_64)
                goto a1
        case 26:
                x=wln.setwep(WEP_KEY,PL_WLN_WEP_MODE_128)
a1:             if x<>OK then
                        sys.halt
                        error=YES
                end if
                while wln.task<>PL_WLN_TASK_IDLE
                wend
        case else:
                sys.halt
                error=YES         ,błąd klucza
        end select
'skanowanie w poszukiwaniu Access Pointa
        if wln.scan(AP_NAME)<>OK then
                sys.halt
                error=YES         ,błąd
        end if
        while wln.task<>PL_WLN_TASK_IDLE
        wend
        if wln.scanresultssid<>AP_NAME then
                sys.halt
'Access point o danej nazwie SSID
' nie został odnaleziony
                error=YES
        end if

'połączenie z znalezionym Access Pointem
        if wln.associate(wln.scanresultbssid,wln.scanresultssid,wln.
scanresultchannel,wln.scanresultbssmode)<>OK then
                sys.halt
                error=YES         , błąd
        end if
        while wln.task<>PL_WLN_TASK_IDLE
        wend
        if wln.associationstate<>PL_WLN_ASSOCIATED then
                sys.halt
                error=YES         , błąd, sprawdź klucz
        end if
end sub
```

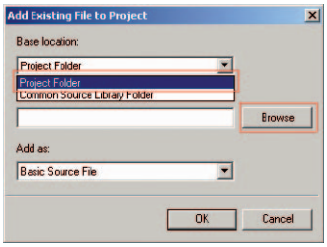
Listing 4. Obsługa zdarzeń od interfejsu WiFi

```
,Zdarzenie od interfejsu sieciowego
sub on_wln_event(wln_event as pl_wln_events)
        dim x as byte
        sys.halt
' zdarzenie, w zmiennej x w trybie debagowania można odnaleźć błąd
' 0 - Wi-Fi sprzęt został odłączony, wyłączony lub działa niepoprawnie
' 1 - interfejs Wi-Fi został odłączony od sieci bezprzewodowej
        x=wln_event
end sub
```

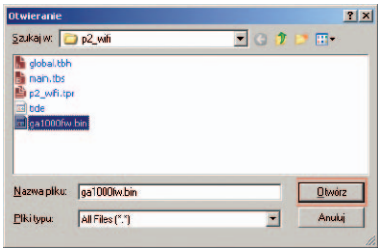
racyjnymi (tryb klienta), jak i Ad hoc (tryb równorzędny). W celu zabezpieczenia komunikacji jest wykorzystywane szyfrowanie WEP (*Wired Equivalent Privacy*) o kluczu 40- lub 104-bitowym.



Rysunek 1. Dodawanie pliku do drzewa projektu



Rysunek 2. Wyświetlenie katalogu projektu



Rysunek 3. Otwarcie pobranego z Internetu pliku ga1000fw.bin

Interfejs WiFi GA1000, poza możliwością wpisania stałych parametrów sieci bezprzewodowej (SSID, tryb działania Ad hoc lub Infrastruktura, klucz WEP itd.), pozwala też na wyszukiwanie dostępnych sieci bezprzewodowych.

Na początku napiszemy jednak najprostszy program pozwalający na ko-

Listing 5. Realizacja procedury echa

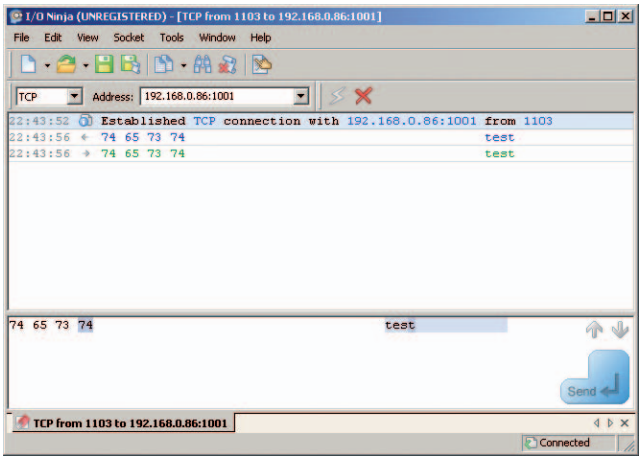
```
'Pętla echo na zadeklarowanym porcie dostępnym od strony sieci bezprzewodowej
poprzez TELNET
sub on_sock_data_arrival()
        sock.setdata(sock.getdata(sock.txfree))
        sock.send
end sub

'Informuje że coś się zmieniło na sockecie
sub on_sock_event(newstate as pl_sock_state, newstatesimple as pl_sock_state_
simple)
        if newstatesimple=PL_SSTS_EST then
                pat.play(„G~”,PL_PAT_CANINT)
        else
                pat.play(„-”,PL_PAT_CANINT)
        end if
end sub
```

Listing 6. Zmiany w konfiguracji modułu przygotowujące go do pracy w trybie Ad hoc.

```
'stała wyboru trybu działania moduły WiFi
const WIFI_MODE = 0
        ' 0 - ADHOC, 1 - AP klient
const ADHOC_CHANNEL = 6    ' kanała dla ADHOC
        if WIFI_MODE = 0 then
                wln.networkstart(AP_NAME,ADHOC_CHANNEL)

                while wln.task<>PL_WLN_TASK_IDLE
                wend

                if wln.associationstate<>PL_WLN_OWN_NETWORK then
                        sys.halt
                        error = YES
                end if
        else

'skanowanie w poszukiwaniu Access Pointa
                if wln.scan(AP_NAME)<>OK then
                        sys.halt
                        error=YES         ,błąd
                end if
                while wln.task<>PL_WLN_TASK_IDLE
                wend
                if wln.scanresultssid<>AP_NAME then
                        sys.halt
'Access point o podanej nazwie SSID
'nie został odnaleziony
                        error=YES
                end if
'połączenie z znalezionym Access Pointem
                if wln.associate(wln.scanresultbssid,wln.scanresultssid,wln.
scanresultchannel,wln.scanresultbssmode)<>OK then
                        sys.halt
                        error=YES         , błąd
                end if
                while wln.task<>PL_WLN_TASK_IDLE
                wend
                if wln.associationstate<>PL_WLN_ASSOCIATED then
                        sys.halt
                        error=YES         , błąd, sprawdź klucz
                end if
        end if
```

munikację z znaną nam siecią. Pisanie aplikacji rozpoczynamy od deklaracji linii IO interfejsu SPI oraz ustawienia parametrów sieci (**listing 1**). Następnie przeprowadzamy konfigurowanie modułu. W tym celu skorzystamy zdarzenie *on_sys_init*, które jest wykonywane jako pierwsze po włączeniu zasilania układu lub jego zerowaniu. Sposób inicjalizacji systemu zamieszczono na **listingu 2**.

W tym miejscu należy dołączyć do projektu plik binarny odpowiadający za konfigurację sprzętową interfejsu WiFi: ga1000fw.bin. Klikamy prawym przyciskiem myszy na drzewie projektu i wybieramy *Add Existing File* (**rysunek 1**). Pojawi się okno, w którym należy wybrać *Project Folder*, a następnie przycisnąć klawisz *Browse* (**rysunek 2**). Na ekranie ukaże się zawartość folderu z projektem. Teraz musimy pobrać z Internetu (za pomocą przeglądarki internetowej *http://tibbo.com/downloads/basic/firmware.html* lub bezpośrednio *http://tibbo.com/downloads/secure/ga1000fw.bin*) plik binarny ga1000fw.bin i skopiować go do folderu z projektem. Gdy już to zrobimy, możemy w oknie wyboru zaznaczyć go i otworzyć (**rysunek 3**). Następnie klikamy OK. Rezultatem powinno być dodanie pliku do drzewa projektu.



Rysunek 4. Efekt działania programu *echo TCP*

Na **listingu 3** zamieszczono dalszą część kodu źródłowego, odpowiedzialną za otwarcie pliku konfiguracji sprzętowej.

W taki sposób poprawnie skonfigurowaliśmy i zainicjowaliśmy moduł em1206 z interfejsem sieciowym do pracy w sieci WiFi jako klienta Access Pointa.

Dalsza część programu odpowiada za obsługę zdarzeń wynikających z pracy w sieci oraz obsługę zdarzeń systemowych (**listing 4**).

Aby pokazać aktywność urządzenia, procedura z **listingu 5** działa jak echo. To znaczy dane które zostaną odebrane przez otwarty port urządzania (dla przypomnienia – 1001), zostaną z powrotem odesłane niezmienione. Dodatkowo użytkownik zostanie o tym poinformowany o mignięciem diody LED. Efekt działania programu prezentuje ekran umieszczony na **rysunku 4**.

Kodu źródłowego z listingu 5 możemy również z powodzeniem użyć do konfigurowanie urządzenia Tibbo do pracy w trybie równorzędnym Ad hoc. Musimy jedynie dodać w ciele deklaracji stałe (jak na **listingu 6**) oraz zmienić procedurę rozpoznającą zadeklarowany tryb.

W celu przetestowania działania układu należy pamiętać, aby ustawić odpowiedni adres IP karty bezprzewodowej komputera.

Moduły Tibbo mają wiele możliwości, które można z powodzeniem wykorzystywać w aplikacjach, zarówno komercyjnych jak i przemysłowych. Na podstawie lektury artykułu dowiedzieliśmy się, jak poprawnie skonfigurować moduł do pracy w sieciach statycznych przewodowych i bezprzewodowych. Po więcej informacji na temat środowiska TAIKO warto sięgnąć do strony producenta *http://www.tibbo.com*. Wśród wielu przykładów odnajdziemy między innymi przykład programu ilustrującego sposób implementacji współpracy z sieciami z dynamicznie przydzielającymi adres IP (DHCP), z protokołem SMTP oraz FTP.

Mikołaj Zarzycki
m.zarzycki@soyter.pl
Soyter Sp. z o.o.