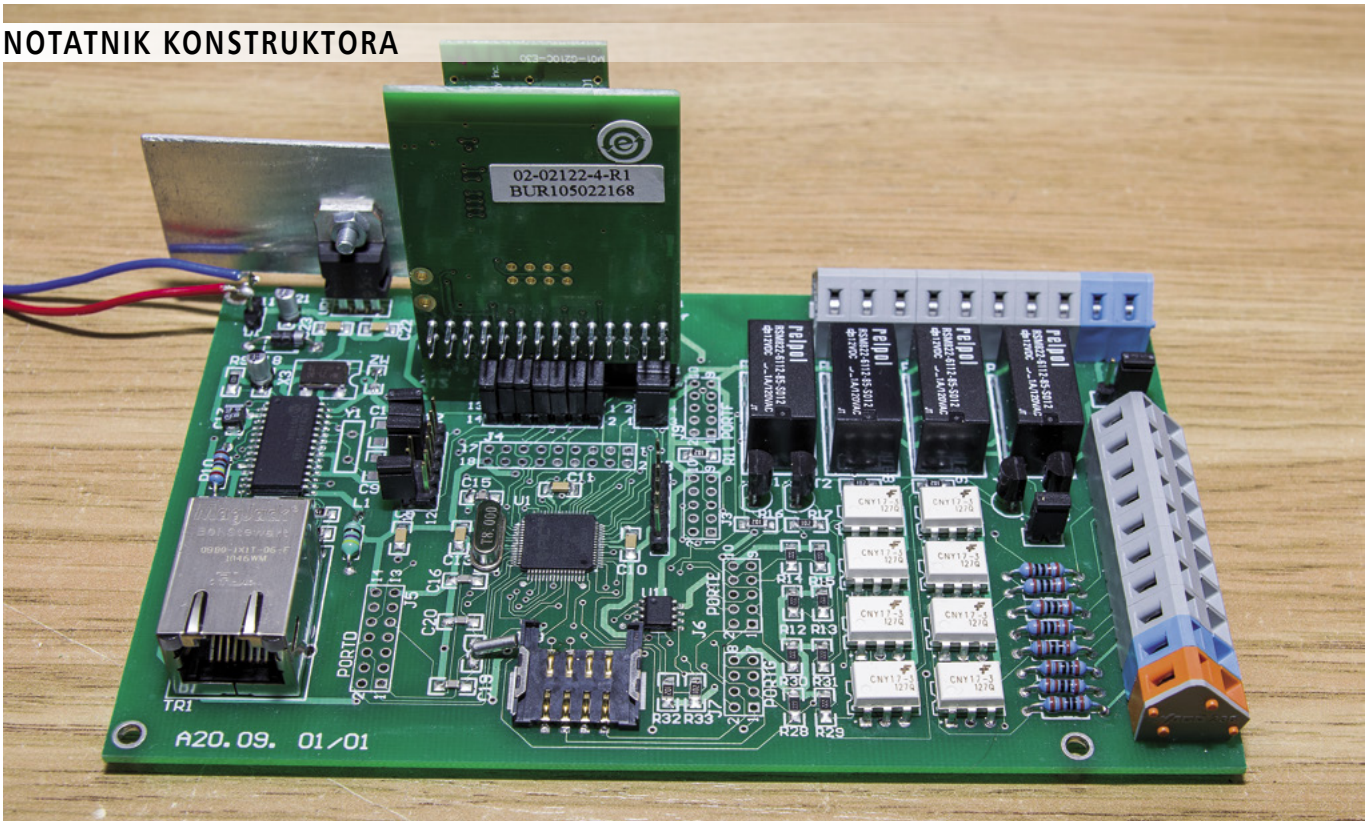


Serwer www – praca z interfejsem Wi Fi

W EP 12/2012 na stronie 39 opisaliśmy projekt serwera WWW mogący pełnić funkcję centrum automatyki domowej. To urządzenie może być dołączone do sieci Ethernet nie tylko za pomocą przewodu, ale również z użyciem fal radiowych. Artykuł wyjaśnia krok po kroku, jak dodać do serwera możliwość komunikowania się poprzez Wi-Fi oraz wykonać program uwzględniający tę funkcjonalność.

Serwer wyposażony w interfejs Ethernet można dołączyć do sieci lokalnej LAN za pomocą kabla. To rozwiązanie jest pewne, niezawodne i tanie. Standard Ethernet zapewnia izolację galwaniczną i sporą prędkość przesyłu danych. Jednak konieczność wykonania połączenia kablowego może być uciążliwa, a pewnych przypadkach trudna lub nawet niemożliwa do wykonania. Znacznie wygodniejsze jest połączenie radiowe pomiędzy klientem (komputerem z uruchomioną przeglądarką) a serwerem. Zarówno w sieciach kablowych jak i w radiowych obowiązują standardy. Chyba każdy użytkownik komputera zna nazwę Wi-Fi. Tak jest nazywany zestaw rozwiązań sprzętowych i programowych przeznaczonych do radiowego łączenia urządzeń w lokalnej sieci LAN. Sieci, w których jest możliwe bezprzewodowe dołączanie urządzeń są bardzo wygodne w użyciu i dlatego błyskawicznie zdobyły olbrzymią popularność. Wszystkie nowe urządzenia mobilne: laptopy, tablety, czy smartfony mają wbudowany interfejs WiFi, poprzez który mogą łączyć się z lokalnymi punktami dostępowymi. Takim punktem dostępowym może być domowy router ADSL z funkcją Wi-Fi, komercyjne punkty dostępu do Internetu lub darmowe punkty dostępu typu Hot Spot.

Zalety połączenia bezprzewodowego można wykorzystać do połączenia naszego serwera z lokalną siecią LAN. Ponieważ funkcje sterownicze mogą wymagać umieszczenia modułu serwera w nietypowych miejscach, to połączenie radiowe może znacznie ułatwić jego zainstalowanie.

W porównaniu do interfejsu Ethernet połączenie radiowe Wi-Fi jest trudniejsze do wykonania z punktu widzenia konstruktora, a konfiguracja sieci wymaga zwrócenia uwagi na możliwe zagrożenia. Jednak wykonanie niezawodnie działającego połączenia radiowego naszego serwera z siecią LAN jest jak najbardziej możliwe.

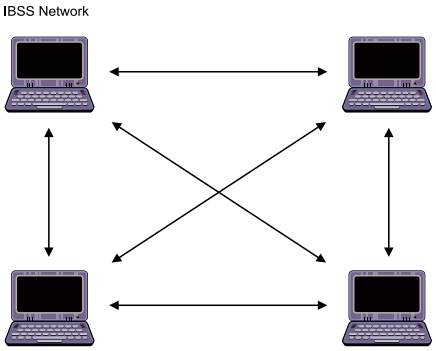
Topologia sieci Wi-Fi

Nazwy Wi-Fi zwykliśmy używać dla standardu łączącego bezprzewodowo komputer z siecią Internet. Jednak tak naprawdę, jest to połączenie opisywane przez normę IEEE 802.11b lub jej odmiany różniące się w nazwie literą końcową. Więcej informacji na temat struktury, topologii i zabezpieczeń sieci z interfejsem radiowym IEEE 802.11b podałem w artykule „Wi-Fi od Microchipa" opublikowanym w EP 9/2012. Na potrzeby tego artykułu przypomnę tutaj niezbędną część zawartych tam informacji.

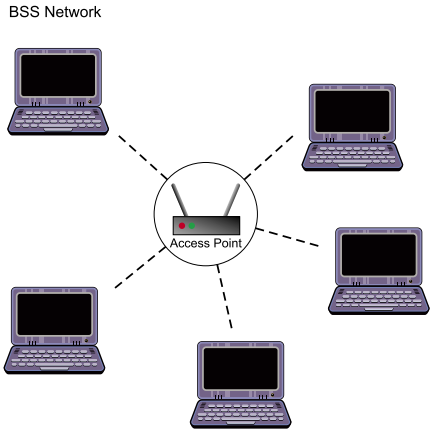
Sieci radiowe Wi-Fi mogą pracować w 2 topologiach:

- Infrastructure,
- Ad Hoc.

Przeciętnemu użytkownikowi połączenia Wi-Fi zwykle jest znana tylko topologia Infrastructure pokazana na **rysunku 1**. Punktem centralnym sieci pracującej w tej topologii jest urządzenie nazywane punktem dostępowym (*Access Point*). Wszystkie dane pomiędzy urządzeniami w sieci przepływają przez punkt dostępowy i nie ma możliwości transmisji bezpośrednio pomiędzy urządzeniami. Topologia Infrastructure najczęściej jest stosowana w sie-



Rysunek 1. Topologia Infrastructure

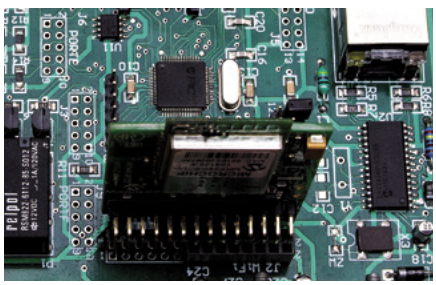


Rysunek 2. Topologia sieci Ad Hoc

ciach domowych umożliwiających dostęp do szerokopasmowego Internetu (router z opcją Wi-Fi), w sieciach publicznych Hot Spot oraz w komercyjnych sieciach radiowych.

Mniej znana jest sieć typu Ad Hoc, której topologię pokazano na **rysunku 2**. Jest to sieć typu peer-to-peer, w której każdy z komputerów może bezpośrednio komunikować się ze wszystkimi komputerami tworzącymi sieć. Ta topologia może być stosowana do tworzenia rozległych sieci, bo jej węzły mogą pracować jako przekaźniki danych pomiędzy bardzo oddalonymi urządzeniami.

Fotografia 3. Moduł Wi-Fi



Fotografia 4. Płytka z modułem Wi-Fi

Listing 1 zmiana definicji linii SPI1

```
#if defined ENC_IN_SPI1
  #define ENC_CS_TRIS  (TRISFbits.TRISF1)  // Comment this line out if you are
using the ENC424J600/624J600, MRF24WB0M, or other network controller.
  #define ENC_CS_IO    (PORTFbits.RF1)
  #define ENC_RST_TRIS (TRISFbits.TRISF0)  // Not connected by default.  It is
okay to leave this pin completely unconnected, in which case this macro should
simply be left undefined.
  #define ENC_RST_IO   (PORTFbits.RF0)
```

Jak wiemy serwer z EP 12/2012 ma wbudowany kompletny interfejs Ethernet. Na etapie projektowania układu nie było jednak pewne, czy w ogóle uda się uruchomić połączenie radiowe w środowisku TCPmakera. Jednak projektując płytkę na wszelki wypadek przewidziałem możliwość wyposażenia serwera w moduł Wi-Fi typu MRF24WB0M Pictail Plus produkowany przez firmę Microchip. Zarówno układ ENC28J60, jak i moduł Wi-Fi komunikują się z mikrokontrolerem poprzez pojedynczy, szeregowy interfejs SPI – w naszym wypadku jest to interfejs sprzętowy SPI1.

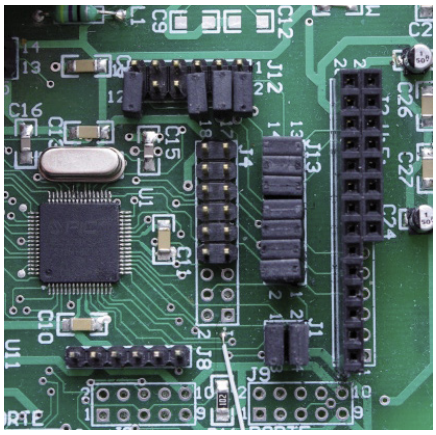
Wygląd modułu Wi-Fi pokazano na **fotografii 3**. Jest on przystosowany do podłączania do firmowych modułów ewaluacyjnych (na przykład Explorer16) głównie za pomocą złącza krawędziowego. Tak się dobrze składa, że ma on też złącze szpilkowe (goldpiny) łatwe do użycia bez konieczności stosowania specjalnego gniazda. Tę właściwość wykorzystałem do połączenia modułu z płytką serwera (**fotografia 4**). Po zainstalowaniu modułu trzeba rozewrzeć wszystkie zworki J12 i zewrzeć wszystkie piny złącza konfiguracyjnego J13 (**fotografia 5**). Linie interfejsu SPI1 zostaną odłączone od ENC28J60 i dołączone do modułu Wi-Fi. Tak skonfigurowany serwer jest sprzętowo gotowy do testowania. W prototypie musiałem wykonać jedną zamianę połączeń. Linia RD8 sterująca przekaźnikiem PRZ1 jest jednocześnie wejściem przerwania zewnętrznego INT0. Oprogramowanie transmisji wykorzystuje to przerwanie i dlatego sterowanie przekaźnikiem musiało być przełączone na linię RB0.

TCPMaker i Wi-Fi

Pierwsze moje próby z połączeniem radiowym i projektem wygenerowanym przez TCPmakera zakończyły się niepowodzeniem. Zazwyczaj w przypadkach, kiedy coś powinno działać a nie działa i nie wiadomo dlaczego, kontaktuję się ze wsparciem technicznym producenta. Jak się okazało, miałem starszą wersję programu, która nie wpierała komunikacji Wi-Fi. Dzięki uprzejmości pana Roberta Millera właściciela firmy Trace Systems Inc., wykonałem aktualizację używanego przez mnie TCPMakera do najnowszej wersji 1.5.0 i prace na projektem mogły zakończyć się sukcesem.

TCPMaker po wybraniu kompilatora MPLAB-C32 domyślnie konfiguruje projekt dla modułu ewaluacyjnego Explorer16 i modułu Ethernet z układem ENC28J60. Połączenie ethernetowe jest uznawane za standardowe i trudno się temu dziwić. Dlatego projekt musiał zostać przekonfigurowany. Nie było to zadanie łatwe, bo nie mogłem znaleźć informacji, jak to zrobić. Dużą pomocą okazała się znajomość przykładowego programu testującego łącze Wi-Fi dostarczanego przez Microchipa. Kiedyś musiałem dobrze zapoznać się z tą aplikacją i dzięki temu udało mi się skonfigurować projekt wygenerowany przez TCPMakera do obsługi modułu MRF24WB0M i połączenia radiowego.

Najnowsza wersja TCPMakera umieszcza w podkatalogu \Configs\ pliki konfigurujące sprzęt dla różnych zestawów sprzętowych modułów ewaluacyjnych Microchipa. Autorzy programu wyszli ze słusznego założenia, że użytkownik powinien najpierw sprawdzić działanie programu na przetestowanym sprzęcie. Oczywiście, nic nie stoi na przeszkodzie by pliki konfiguracyjne przystosować do własnych potrzeb i ja tak właśnie zrobiłem. Aby zmienić konfigurację, należy użyć edytora tekstowego np. Notepad. Na początek wybieramy plik HWP PIC32_GP_SK_MRF24WB.h przeznaczony do konfigurowania modułu PIC32 StarterKit połączonego z modułem MRF24WB0M poprzez interfejs SPI1. Ta konfiguracja byłaby idealna, gdybym w serwerze użył takiego samego mikrokontrolera, jak w PIC32 StarterKit. Jednak mikrokontroler w układzie serwera ma inną liczbę wyprowadzeń i linie SPI są mapowane do innych nóżek. Dlatego w pliku trzeba zmienić definicję linii interfejsu SPI1. Jest to czynność nieskomplikowana i nie powinna sprawić żadnych



Fotografia 5. Konfiguracja zworek dla interfejsu Wi-Fi

Listing 2 zmiany w pliku HardwareProfile.h

```
#if defined(__C30__)
  #include "Configs/HWP EX16_ENC28 C30.h"
  #else
  //#include "Configs/HWP EX16_ENC28 C32.h"
  #include "/Configs/CONFIG_SERWER_WF.h"
#endif
```

problemów. Niezbędne zmiany pokazano na **listingu 1**.

Zmodyfikowany plik zapamiętałem pod nazwą CONFIG_SERWER_WF.h. Aby konfiguracja była dołączona do projektu, trzeba zmienić kolejny plik HardwareProfile.h. Jego zawartość zamieszczono na **listingu 2**.

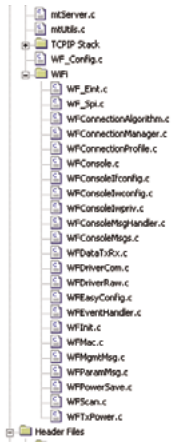
Kolejny krok modyfikacji projektu wygenerowanego przez TCPmaker to dodanie plików źródłowych obsługi interfejsu WiFi. W tym celu do sekcji plików źródłowych dodajemy podkatalog WiFi. Następnie do tego katalogu trzeba przekopiować wszystkie pliki z katalogu \Microchip\TCPIP Stack\WiFi (**rysunek 6**). Oprócz tych plików dodajemy do projektu WF_Config.c z katalogu głównego. I na koniec trzeba usunąć komentarz z definicji w pliku MainDemo.c:

//Used for Wi-Fi assertions

#define WF_MODULE_NUMBER WF_MODULE_MAIN_DEMO.

Po zmianie mikrokontrolera w menu *Configure -> Select Device* na PIC32MX360F512L projekt powinien się skompilować bez błędów.

Teraz pozostaje nam skonfigurowanie pracy sieci WiFi. Do tego celu jest przeznaczony plik o nazwie WF_Config.h. Przed zmianami trzeba zastanowić się w jakiej sieci chcemy pracować. Jeżeli mamy tylko moduł serwera i komputer z kartą Wi-Fi, to możemy zrobić próby i potem pracować docelowo w sieci Ad-Hoc. Dla tej sieci jest wspierane szyfrowanie WEP z kluczem o długości 40 lub 104 bitów. Nie jest to silne zabezpieczenie, ale lepsze niż nic. W sieci z punktem dostępowym (Access Point) można użyć silnego zabezpieczenia WPA lub WPA2. Jest to o tyle ułatwione, że moduł MRF24WB0M potrafi samodzielnie obliczyć klucze binarne. Co prawda, wbudowany mikrokontroler potrzebuje na to trochę czasu (ok. 35 s), ale w praktyce nie jest to problemem, ponieważ te obliczenia są wykonywane tylko raz na początku działania programu, przy pierwszym włączeniu do sieci. Można również obliczyć klucze za pomocą strony internetowej *http://www.wireshark.org/tools/wpa-psk.html* i wtedy uruchomienie programu zajmuje mniej czasu. Jednak potem konfigurowanie sieci z punktu widzenia użytkownika komplikuje się, bo trzeba te wyliczone klucze wpisać do pamięci konfiguracyjnej serwera. Dlatego w naszym projekcie pozostaniemy przy ich wyliczaniu przez moduł Wi-Fi. Najważniejsze fragmenty pliku WF_Config.h pokazano na **listingu 3**.



Rysunek 6. Projekt z plikami źródłowymi Wi-Fi

Rysunek 7. Okno nazw SSID (Windows7)

Listing 3 Fragment pliku WF_Config.h

```
#define MY_DEFAULT_SSID_NAME      "MicrochipDemoAP"
#define MY_DEFAULT_NETWORK_TYPE   WF_INFRASTRUCTURE   /* WF_ADHOC */
#define MY_DEFAULT_CHANNEL_LIST   {1,6,11}            /* Desired channel list */
...
#define MY_DEFAULT_WIFI_SECURITY_MODE WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE
/*WF_SECURITY_WPA_WITH_PASS_PHRASE  WF_SECURITY_WEP_40  WF_SECURITY_OPEN*/
define MY_DEFAULT_PSK_PHRASE "Microchip 802.11 Secret PSK Password"
```

Definicja MY_DEFAULT_SSID_NAME zawiera identyfikator SSID. Dla konfiguracji Ad Hoc jest to nazwa serwera wyświetlana w oknie nazw SSID komputera z charakterystycznym symbolem w postaci 3 komputerów połączonych między sobą (**rysunek 7**). Na jej podstawie można zidentyfikować serwer i się z nim połączyć. W przypadku sieci a punktem dostępowym ta nazwa musi być taka sam jak SSID routera. Ta nazwa również jest wyświetla się w oknie nazw, ale z symbolem poziomu sygnału.

Definicja MY_DEFAULT_NETWORK_TYPE określa typ sieci: Ad Hoc lub Infrastructure. Bardzo istotna jest definicja MY_DEFAULT_WIFI_SECURITY_MODE. Sygnał sieci Wi-Fi jest dostępny dla wszystkich znajdujących się w pobliżu nadajnika. Żeby zapobiec nieuprawnionemu dostępowi do sieci, transmisja musi być zaszyfrowana. Jest możliwe ustawienie transmisji bez zabezpieczeń (WF_SECURITY_OPEN), ale nie polecam tego

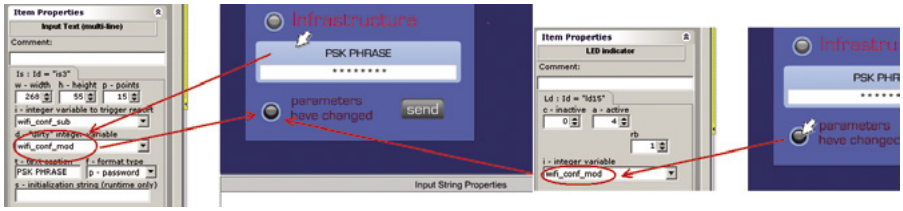
Rysunek 8. Strona służąca do konfigurowania sieci Wi-Fi

ustawienia w trakcie normalnej pracy. Można na chwilę w trakcie testowania lub ustawienia docelowych parametrów połączenia wyłączyć zabezpieczenia, jednak w trakcie normalnej pracy jest to proszenie się o kłopoty.

W konfiguracji Ad Hoc jest wspierane tylko szyfrowanie WEP, a w konfiguracji Infrastructure można włączyć każde z dostępnych zabezpieczeń: WEP, WPA i WPA2. Ja przetestowałem sieć Ad Hoc z zabezpieczeniem WEP z kluczem 40- oraz 102-bitowym i sieć Infrastructure z zabezpieczeniem WPA i WPA2.

Listing 4. Inicjalizacja konfiguracji sieci WiFi

```
void WFInit(void)
{
  char size,i;
  char init [10]={"SerwerInit"};
  //inicjalizacja po pierwszym uruchomieniu
  if(DEFAULT_CONF==0) EEWrite(0xff,5)
  if(EERead(5)!=0x92)
  {
    EEWrite(WF_ADHOC,WF_TYPE);  //sieć AD HOC
    EEWrite(WF_SECURITY_OPEN,WF_SECUR);  //bez zabezpieczeń
    EEWrite(10,SIZE_SSID);  //długość SSID
    for(i=0;i<10i++)  //zapisanie SSID
      EEWrite(*(init+i),BUF_SSID+i);
    EEWrite(0x92,5);
  }
  size=EERead(SIZE_SSID);
  if(size>0&&size<32)
    for(i=0;i<size;i++) *(ssid_name+i)=EERead(BUF_SSID+i);
  if(size>0&&size<42)
  {
    size=EERead(SIZE_WPA);
    for(i=0;i<size;i++)
    *(wpa_pharse+i)=EERead(BUF_WPA+i);
  }
  size=EERead(SIZE_WEP);
  if(size==10)
  {
    for(i=0;i<size;i++) *(wep_key+i)=EERead(BUF_WEP+i);
    for(i=0;i<=5;i++) *(wep_key_bin+i)=char_bin_conv(wep_key+i*2);
  }
  wf_network_type=EERead(WF_TYPE);
  wf_security=EERead(WF_SECUR);
}
```



Rysunek 9. Definiowanie sygnalizacji zmiany parametrów

Ostatecznie, w większości testów wybrałem sieć Infrastructure z zabezpieczeniem WPA2.

Konfigurowanie serwera za pomocą edytowania pliku WF_Config.h jest wygodne i skuteczne, ale wymaga każdorazowego kompilowania programu po zmianie parametrów sieci i jest nie do przyjęcia w aplikacji docelowej. Dlatego wyposażyłem serwer w możliwość konfigurowania z poziomu przeglądarki. Do tego celu do projektu strony serwera dodałem podstronę o nazwie *WiFi Network Configuration* (**rysunek 8**). Jest ona wywoływana z menu głównego za pomocą przycisku nawigacyjnego *WiFi Setup*.

Strona konfiguracji zawiera kilka elementów:

- Okno wprowadzania nazwy SSID.
- Okno wprowadzania klucza 40-bitowego WEP dla konfiguracji Ad Hoc (WEP Encryption).
- Okno wprowadzania znakowego klucza PSK PHRASE dla kodowania PSK i PSK 2 w konfiguracji Infrastructure.
- Przycisku SEND i wskaźnika informującego o zmianie wprowadzanych parametrów z tych trzech okien.
- Przycisk SELECT NETWORK do wyboru rodzaju sieci. Wybrana sieć jest sygnalizowana zmianą koloru wskaźnika.

Każdy wpis do jednego z 3 okien wprowadzania danych powoduje zamianę wskaźnika opisanego jako „Parameter was changed" informując użytkownika o zmianie danych. Działanie tego mechanizmu jest możliwe dzięki zdefiniowaniu i odpowiedniemu wykorzystaniu zmiennej wifi_conf_mod (konfiguracja Wi-Fi została zmodyfikowana). Jak to działa, pokażę na przykładzie okna PSK PHRASE. W definicji tego elementu jest pole „d-dirty" integer variable. Po przypisaniu do tego okna zmiennej wifi_conf_mode, każdy wpis w oknie elementu spowoduje zapisanie do zmiennej wartości 0x01. Jeżeli teraz ta sama zmienna zostanie przypisana do elementu LED Indicator (wskaźnika), to spowoduje to zmianę koloru (**rysunek 9**). Ta sama zmienna jest również przypisana do pól „d-dirty" integer variable pozostałych okien wprowadzania danych nastaw sieci.

Po przyciśnięciu przycisku *Send* wszystkie zmienione parametry są jednocześnie przesyłane do serwera. Do tego celu jest wykorzystywane pole variable to trigger report. We wszystkich elementach typu Input Text jest zdefiniowana zmienna wifi_conf_sub. Jeżeli wartość tej zmiennej zmieni się z 0x00 na 0x01, to przeglądarka wyśle do serwera zawartość buforów wszystkich elementów wprowadzania danych. Ponieważ zmienna wifi_conf_sub jednocześnie jest przypisana do elementu Pushbutton „Send", to po jego naciśnięciu zmienia swoją wartość z 0x00 na 0x01 i zmiany zostają wysłane do serwera.

Rodzaj sieci (Ad Hoc lub Infrastructure) jest zmieniany sekwencyjnie przy klikaniu na przycisk *Select Network*. Wybrana sieć jest sygnalizowana zmianą koloru wskaźnika przy opisie typu sieci.

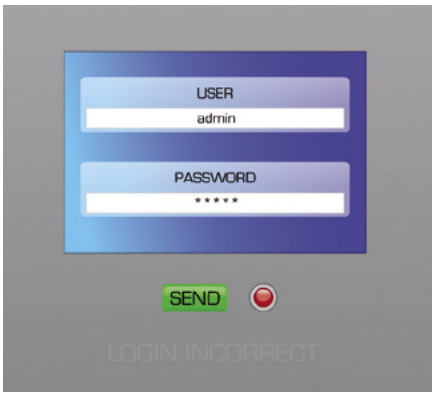
Ciągi znaków z nazwą SSID, kluczami WEP i WPA oraz informacją o typie sieci są zapisywane w pamięci EEPROM i odtwarzane w momencie restartu serwera. Żeby zmiany mogły być wprowadzone, trzeba włączyć i wyłączyć zasilanie układu.

Z punktu widzenia użytkownika, nawiązanie połączenia serwera z siecią LAN poprzez Ethernet jest bardzo łatwe. Wystarczy połączyć się z routerem sieciowym, a reszta dokona się automatycznie. W wypadku sieci Wi-Fi sprawa jest trudniejsza. Aby przyłączyć się do punktu dostępowego, trzeba wpisać w ustawienia nazwę SSID i ewentualnie klucz zabezpieczenia. Zazwyczaj znamy te dane, ale do momentu kiedy się nie połączymy z siecią nie można ich wpisać za pomocą strony pokazanej na rys. 8. Musi być inny sposób na połączenie z serwerem. Można by było wykorzystać połączenia RS232, USB, Ethernet lub lokalny interfejs z wyświetlaczem i klawiaturą. Niestety, w serwerze w trybie Wi-Fi żaden z wymienionych zasobów nie jest dostępny. Interfejsów RS232 i USB nie przewidziano w projekcie, a Ethernet współdzieli SPI z modułem Wi-Fi . Nawet gdyby interfejsy SPI były rozdzielone, to programowe przełączanie pracy z Ethernetu na WiFi nie jest wspierane przez TCPmakera. Interfejs użytkownika również nie jest przewidziany, a nawet gdyby był , to musiałby mieć klawiaturę alfanumeryczną, co jest drogie i niezbyt wygodne. Wyjściem z tej sytuacji jest rozpoczęcie pracy z domyślną konfiguracją Ad Hoc i wyłączonym zabezpieczeniem. Układ po pierwszym uruchomieniu, kiedy pamięć EEPROM nie zawiera ustawień, przełącza się do pracy w sieci Ad Hoc z wyłączonym szyfrowaniem.

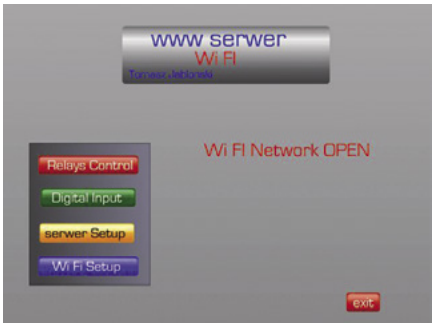
Na **listingu 4** pokazano funkcja inicjalizacji typu sieci i zabezpieczeń po włączeniu zasilania. Jeżeli komórka pamięci EEPROM o adresie 0x05 nie jest zapisana wartością 0x92, to program uznaje, że ma do czynienia z czystą pamięcią i automatycznie uruchamia pracę w sieci Ad Hoc, bez zabezpieczeń, z SSID o nazwie SerwerInit.

W takiej konfiguracji do połączenia przez Wi-Fi nie musimy znać ani nazwy SSID, ani klucza zabezpieczeń. Po połączeniu przechodzimy do strony pokazanej na rys. 8 i ustawiamy docelowe parametry sieci.

Pozostał do rozwiązania kolejny problem – co zrobić, kiedy źle wpiszemy dane i wykonamy restart serwera? Ponownie tracimy możliwość połączenia z serwerem. Rozwiązaniem takiego problemu jest przywrócenie nastaw „fabrycznych", co powoduje ponowne uruchomienie trybu pracy w otwartej sieci Ad Hoc. Jest to możliwe poprzez wymuszenie poziomu niskiego na linii RB14 w momencie uruchamiania serwera. Wtedy do pamięci EEPROM pod adres 0x05 jest wpisywana wartość 0xFF, a procedura konfiguracji sieci



Rysunek 10. Strona startowa



Rysunek 11. Strona główna serwera

zachowa się tak jak przy pierwszym uruchomieniu układu.

W trakcie eksploatowania serwera, niezależnie czy połączonego za pomocą Wi-Fi, czy kabla okazało się, że potrzebne jest zabezpieczenie przed nieuprawnionym dostępem. Dotyczy to szczególnie poleceń sterowniczych i zmiany ustawień sieci oraz czasowych kanałów sterowniczych. Dlatego po uruchomieniu, serwer przesyła do przeglądarki stronę startową z funkcją logowania (**rysunek 10**). Użyłem tu dwóch okien wprowadzania danych Input Text. Pierwsze służy do wprowadzania nazwy użytkownika, a drugie do wprowadzenia hasła. Mechanizm wysyłania wprowadzonych danych do serwera i sygnalizowania zmian jest identyczny, jak w wypadku konfigurowania sieci Wi-Fi. Po naciśnięciu przycisku *Send* wprowadzone ciągi znaków są przesłane do serwera i porównane danymi logowania zapisanymi w pamięci EEPROM. Jeżeli porównanie da wynik pozytywny, to serwer przechodzi do strony głównej (**rysunek 11**). Jeżeli nie, to zostanie podświetlony na czerwono napisz LOGIN INCORRECT i trzeba dane wprowadzić ponownie.

Po pierwszym uruchomieniu serwera domyślnie są ustawione użytkownik admin oraz hasło 12345. W mechanizmie logowania użyłem dwóch funkcji API:

- mtGoToPage – skok do wybranej strony.
- mtSetTextColor – zmiana koloru wskazanego napisu na stronie.

Fragment funkcji sprawdzania prawidłowości logowania pokazano na **listingu 5**. Na **listingu 6** pokazano funkcję przywracająca „nastawy fabryczne", natomiast na **listingu 7**

fragment programu odpowiedzialny za porównanie nazwy użytkownika oraz hasła.

Po pierwszym zalogowaniu nazwę użytkownika i hasło należy zmienić komendami wprowadzanymi z konsoli:

– su(user_name) – ustalenie nazwy użytkownika,
– sp(password) – ustalenie hasła dostępu.

Funkcje obsługi tych komend zapamiętają wprowadzone dane w pamięci eeprom i mogą potem zostać użyte do ponownego logowania. Podobnie jak w przypadku ustawień sieci WiFi domyślne dane do logowania są ustawiane po zwarciu linii RB14 do masy w trakcie uruchamiania serwera. Funkcje obsługi komend su i sp pokazano na **listingu 8**.

Strona konsoli komend (**rysunek 12**) została wyposażona w coś w rodzaju pliku pomocy. Ten krótki opis formatu komend bardzo pomaga z czasie konfigurowania serwera. Oprócz funkcji znanych z serwera z interfejsem Ethernet zostały dodane funkcje ustawienia hasła i nazwy użytkownika na potrzeby logowania. W opisie nie ma 3 dodatkowych komend pozwalających na odczytanie ustawień sieci Wi Fi. Są to:

– @s – wyświetlenie nazwy SSID,
– @p – wyświetlenie klucza WPA,
– @e – wyświetlenie klucza WEP.

Tych komend używałem w trakcie pisania i testowania procedur konfigurowania sieci Wi-Fi i docelowo miały zostać usunięte. Jednak chociaż nie są one jawne, to pozostały. Za ich pomocą można sprawdzić czy zapisane parametry sieci są poprawne. Może to nas uchronić przed utratą kontroli nad połączeniem serwera z przeglądarką.

W porównaniu do poprzedniego opisu serwera z interfejsem Ethernet zmieniłem wygląd strony ze stanami wejść cyfrowych. Oprócz stanu wejść są na niej umieszczone sygnalizacje załączenia lub wyłączenia torów przekaźników (**rysunek 13**).

Serwer bardzo dobrze pracuje w konfiguracji Infrastructure i prawidłowo oblicza klucze szyfrowania WPA i WPA2. To powoduje, że serwer może pracować równie dobrze w domowej jak i przemysłowej sieci Wi-Fi, bez konieczności konfigurowania sieci Ad Hoc. Również praca w konfiguracji Ad Hoc przebiega bez problemów. Testy przeprowadziłem z routerem ASMAX 1004g. Wykrywanie serwera i przypisanie mu adresu IP w lokalnej sieci LAN trwało od momentu włączenia ok. 2...3 minut. Jeżeli router pamiętał adres IP z poprzednich połączeń, to ten czas był krótszy. Należy pamiętać, że moduł Wi-Fi potrzebuje ok. 35 sekund na wyliczenie kluczy WPA. Po nawiązaniu połączenia i uruchomieniu strony o adresie http://mchpboard można testować działanie wszystkich funkcji serwera.

Podsumowanie

Jeszcze nie tak dawno temu wykonanie serwera Web z rozbudowaną stroną WWW

Listing 5. Funkcja sprawdzania prawidłowości logowania

```
void eventsec_access(void)
{
  char size_user,size_pass;
  size_user=strlen(Is4);
  size_pass=strlen(Is5);
  if(sec_access==1)
  {
    memcpy(username,Is4, size_user);
    memcpy(password,Is5, size_pass);
    if(CmdSecurityCmpUser(username,size_user)==1) //porównanie wprowadzonej
nazwy użytkownika z zapisaną
    {
      mtSetTextColor(„tx1”, 0xFF,0x00,0x00);/kolor czerwony
      return;
    }
    if(CmdSecurityCmpPass(password,size_pass)==1) //porównanie wprowadzonego
hasła użytkownika z zapisanym
    {
      mtSetTextColor(„tx1”, 0xFF,0x00,0x00); //kolor czerwony
      return;
    }
    mtSetTextColor(“tx1”, 0x99,0x99,0x99);//kolor tła
    mtGoToPage(„pg5”); //skok do strony 5
  }
}
```

Listing 6. Funkcja inicjalizacji logowania

```
void SecurityInit(void)
{
  char username[]=”admin”;
  char password[]=”12345”;
  char i;
  if(DEFAULT_CONF==0) EEWrite(0xff,2) //linia RB0 jest w stanie niskim
  if(EERead(2)!=0x57)
  {
    EEWrite(0x57,2);
    EEWrite(5,SIZE_U); //wielkosc bufora nazwy uzytkowanika
    for(i=0;i<5;i++) EEWrite(*(username+i),BUF_USER+i);
    EEWrite(5,SIZE_P); //wielkosc bufora nazwy uzytkownika
    for(i=0;i<5;i++) EEWrite(*(password+i),BUF_PASS+i);
  }
}
```

Listing 7. Funkcje porównujące nazwę użytkownika i hasło

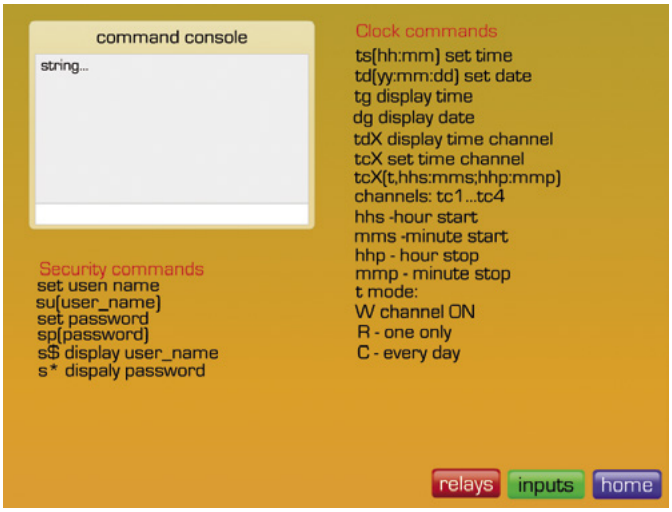
```
//porównanie nazwy uzytkownika
char CmdSecurityCmpUser(char * buffer, char size)
{
  unsigned char i;
  if(size!=EERead(SIZE_U)) return(1); //size error
  for(i=0;i<size;i++)
  {
    if(*(buffer+i)!=EERead(BUF_USER+i))
    return(1);
  }
  return(0);
}

//porównanie hasła
char CmdSecurityCmpPass(char * buffer, char size)
{
  unsigned char i;
  if(size!=EERead(SIZE_P)) return(1); //size error
  for(i=0;i<size;i++)
  {
    if(*(buffer+i)!=EERead(BUF_PASS+i))
    return(1);
  }
  return(0);
}
```

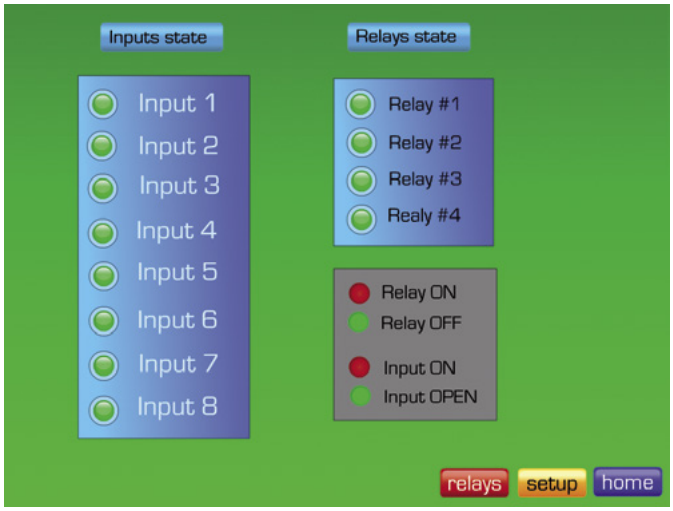
Listing 8. Obsługa komend su i sp

```
//zapamiętanie nowej nazwy użytkownika do logowania
char CmdSetUser(char *buffer)
{
  unsigned char size, i;
  size=strlen(buffer);
  if(*(buffer+1)!=’u’) return(1);
  if(*(buffer+2)!=’(‘) return(1);
  if(*(buffer+size)!=’)’) return(1);
  //zapisz nową nazwę uzytkownika
  for(i=3;i<size-1;i++) EEWrite(*(buffer+i),(BUF_USER+(i-3)));
  return(0);
}

//zapamiętanie nowego hasła do logowania
char CmdSetPass(char *buffer)
{
  unsigned char size, i;
  size=strlen(buffer);
  if(*(buffer+1)!=’p’) return(1);
  if(*(buffer+2)!=’(‘) return(1);
  if(*(buffer+size)!=’)’) return(1);
  //zapisz nowe hasło
  for(i=3;i<size-1;i++) EEWrite(*(buffer+i),(BUF_PASS+(i-3)));
  return(0);
}
```



Rysunek 12. Strona z konsolą wprowadzania komend



Rysunek 13. Strona wyświetlająca stany wyjść

wiązało się ze sporymi kosztami i olbrzymim nakładem pracy. Trudno sobie było wyobrazić by jeden konstruktor samodzielnie i w miarę szybko poradził sobie z tym zadaniem. Wiem z doświadczenia, jak trudno jest napisać „na piechotę" oprogramowanie z nawet bardzo nieskomplikowanym interfejsem dla urządzenia pracującego w sieci LAN. Stos TCP/IP wspiera wszystkie warstwy, oprócz warstwy aplikacji. W wypadku, gdy używamy gotowego stosu, to właśnie warstwa aplikacji i styk pomiędzy nią a stosem stanowi największe wyzwanie dla programisty. Trzeba pamiętać, że Microchip nie dostarcza komercyjnego produktu z pełnym opisem bibliotek, ale dobrze działające, przykładowe programy, które można uruchomić na firmowych modułach ewaluacyjnych. Firma daje za darmo kawał solidnego, stale rozwijanego oprogramowania. Jednak szersza modyfikacja i przystosowanie firmowych rozwiązań dla własnych potrzeb nie jest łatwe. Oczywiście, nie oznacza to, że jest to niemożliwe, ale wymaga nakładu pracy i sporego doświadczenia. Dlatego użycie TCPMakera jest bardzo pomocne, wręcz nieocenione. Wykorzystuje on stos Microchipa, a jednocześnie daje możliwość bardzo szybkiego i wydajnego utworzenia warstwy aplikacji. Programista nie musi się martwić o wiele rzeczy, na przykład jak „zmusić" program do przesłania danych z przeglądarki do serwera i odwrotnie. Nie trzeba znać mechanizmów przyporządkowania wartości zmiennej jakiejś akcji na ekranie – na przykład zapaleniu lub zgaszeniu wirtualnej diody LED. Nawet nie trzeba znać języka HTML, by zaprojektować swój interfejs. Uważam, że strony zaprojektowane z pomocą TCPmakera są czytelne i łatwe w obsłudze. Można je tworzyć i modyfikować w iście ekspresowym tempie.

Podczas pracy nad serwerem okazało się też, że coraz bardziej ważna jest umiejętność analizowania i modyfikowania procedur napisanych przez innych. Bez tej umiejętności nie będziemy w stanie poradzić sobie przy bardziej skomplikowanych zadaniach. W tym wypadku trzeba było odnaleźć fragmenty kodu pobierające informację o konfiguracji sieci z pliku WF_Config.h i zastąpić je własnymi procedurami. Dla tak dużego i skomplikowanego programu nie jest to proste i oczywiste zadanie.

Dziękuję Panu Robertowi Millerowi z firmy TRACESystemInc za udostępnienie programu TCPMaker Pro oraz Panu Piotrowi Ekiertowi z firmy Ekiert za udostępnienie modułów MRF24WB0M i narzędzi firmy Microchip niezbędnych do powstania tego projektu.

Tomasz Jabłoński, EP