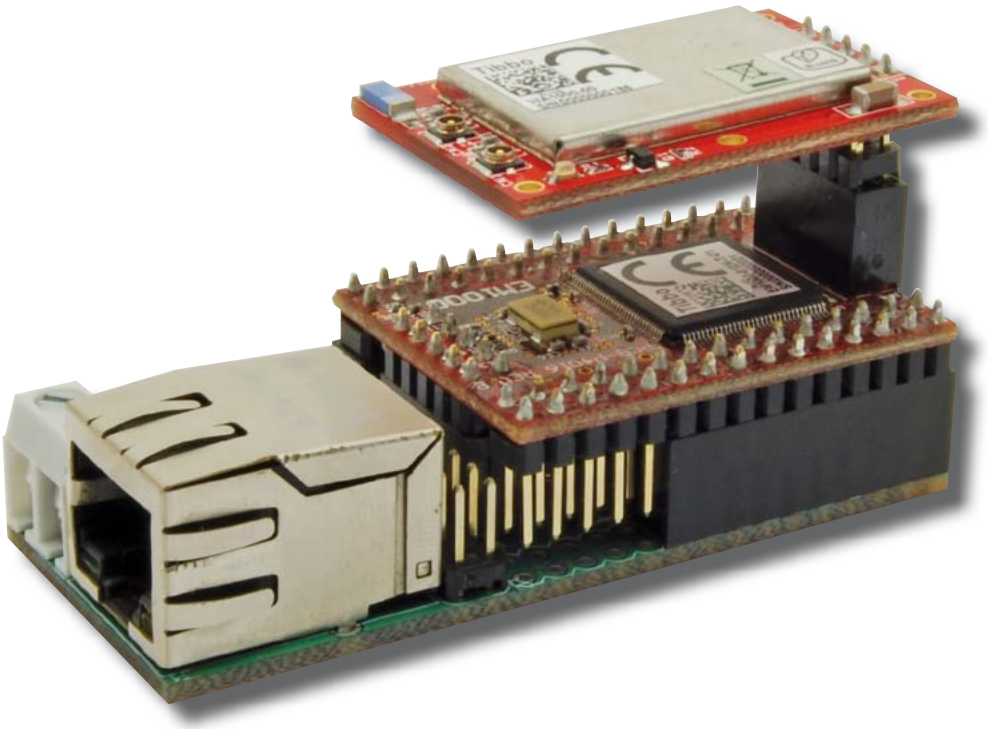
Pilot z WiFi

Zdalne sterowanie sprzętem RTV przez WiFi

Jak większość niekonwencjonalnych projektów opisywanych na łamach EP, tak i ten powstał napędzany przez potrzebę dnia codziennego. Powyższe zdanie jest elokwentną wymówką dla lenistwa autora... Sytuacja, która doprowadziła do powstania prototypu prawdopodobnie zdarza się jednak często w wielu domach, a czy zaszkodzi niewielkie poprawienie sobie komfortu życia?

Rekomendacje:
pomysłowe, praktyczne urządzenie, które przyda się w każdym domu.



W mieszkaniu, w którym w jednym pokoju znajduje się bliżej nieokreślony sprzęt RTV (w przypadku autora był to zestaw audio), a drugi pokój to sypialnia, wynikła potrzeba sterowania sprzętu. Tuż po przebudzeniu, kiedy nie trzeba iść do pracy, miło jest posłuchać np. ulubionej stacji radiowej bądź nagrania. Jednak, aby móc to zrobić, należy WSTAĆ z łóżka, włączyć sprzęt, po czym do niego wrócić. Cały urok wolnego dnia znika!

Oczywiście powyższy opis zawiera nieco przesady, jednak nie od dziś znane są wszelkiego rodzaju „przedłużacze" pilotów IR mające za zadanie pomóc w tego typu sytuacjach. Kilka takich konstrukcji pojawiło się na łamach różnych czasopism. Najczęściej, aby ominąć przeszkodę na drodze wiązki podczerwieni nadajnika (np. ścianę pokoju), urządzenia te wykorzystują fale radiowe. W niektórych konstrukcjach użytkownik np. do sterowania tunerem radiowym używa pilota służącego do jego kontroli, celując w odbiornik „przedłużacza", który z kolei wysyła odebrany rozkaz do swojego nadajnika, umieszczonego przy sterowanym sprzęcie (**rys. 1**).

Po głębszym zastanowieniu urządzenie takie nie rozwiązuje jednak problemu lenistwa autora, bo zawsze przed pójściem spać można zapomnieć o pilocie i zostawić go w innym pokoju. Kolejnym krokiem było zatem znalezienie czegoś, co leniwy bohater zawsze ma przy sobie np. PDA lub telefon komórkowy z interfejsem Wi-Fi, który mógłby znaleźć zastosowanie jako pilot. Jeśli tak jest, to w mieszkaniu znajduje się zapewne Access Point sieci bezprzewodowej, który można użyć jako tor radiowy. Brakuje nam

AVT–5197

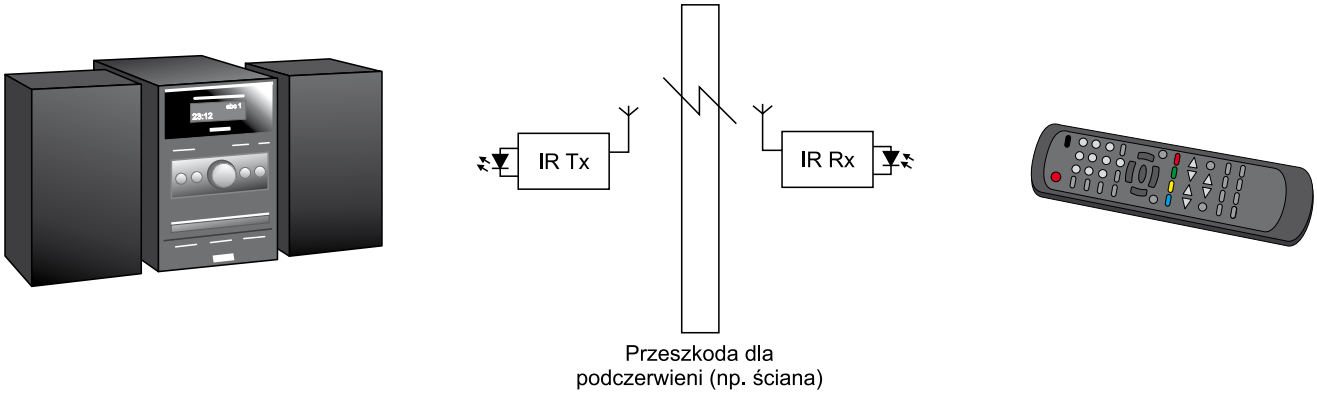
W ofercie AVT:
AVT–5197A – płytka drukowana

PODSTAWOWE PARAMETRY

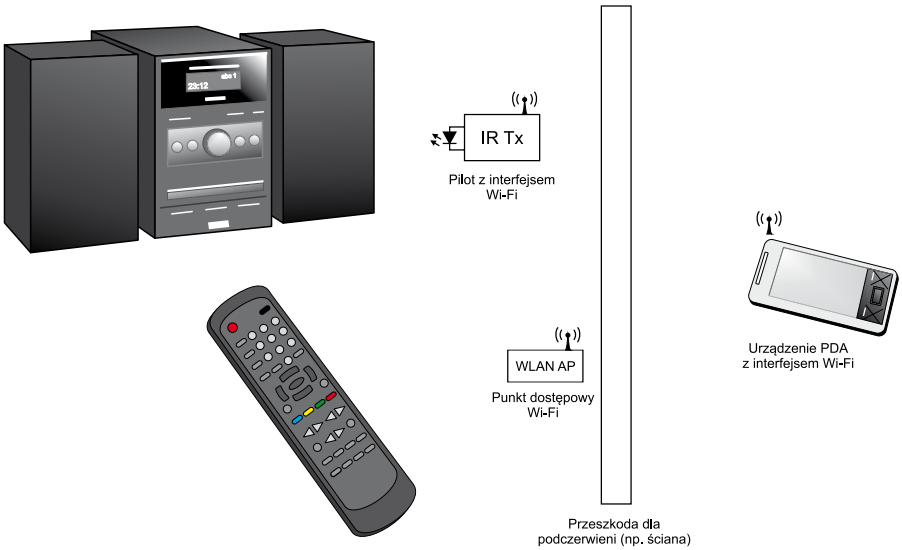
- zasilanie 6...12 VDC
- płytka o wymiarach 65×28 mm
- mikrokontroler ATmega8, moduły Tibbo EM1000 i WM1000
- połączenie pomiędzy PDA/telefonem przez sieć WiFi
- sterowanie za pomocą wbudowanego serwera Web

PROJEKTY POKREWNE wymienione artykuły są w całości dostępne na CD

Tytuł artykułu	Nr EP/EdW	Kit
4-kanałowe radiowe zdalne sterowanie RFM1 z interfejsem MODBUS	EP 8-9/2008	AVT-5143
Bezprzewodowy regulator głośności	EdW 8/2006	AVT-2795
Radiowy przedłużacz pilotów	EP 2/2004	AVT-559
„Przedłużacz" pilota RC5	EP 12/2002	AVT-1359
Radiowy pilot do PC	EP 12/2001	AVT-5032



Rys. 1. Schemat działania tradycyjnego przedłużacza pilotów



Rys. 2. Idea zastosowania pilota zdalnego sterowania z interfejsem Wi-Fi

tylko pilota podczerwieni, który byłby w stanie podłączyć się do sieci Wi-Fi...

Architektura sprzętowa

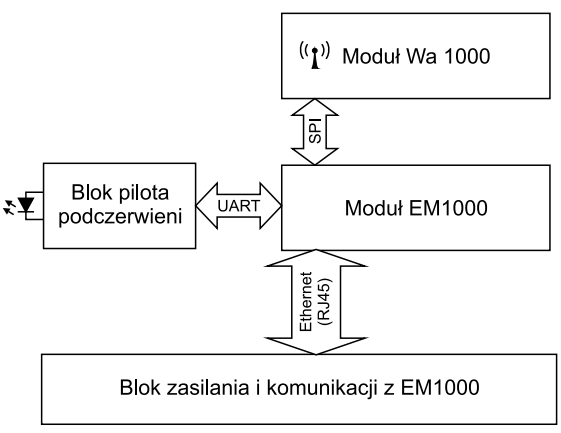
Koncepcję systemu wraz z pilotem opisanym w niniejszym artykule przedstawiono na **rys. 2**.

Część sprzętową pilota podzielono na trzy bloki (**rys. 3**):

- nadajnik podczerwieni zdolny nadawać komendy IR w standardzie zgodnym z obsługiwanym urządzeniem (w tym projekcie jest nim jednostka sterująca zestawem muzycznym Technics ST-CH540),
- programowalny moduł Tibbo EM1000.
- moduł WA1000; jest to moduł Wi-Fi z interfejsem SPI współpracujący z EM1000.

Układ nadajnika podczerwieni zrealizowano z zastosowaniem mikrokontrolera. Jego schemat ideowy przedstawiono na **rys. 4**.

Głównym elementem nadajnika jest mikrokontroler AVR – ATmega8. Pracuje on w podstawowej konfiguracji zewnętrznym rezonatorem kwarcowym 8 MHz. Linia zerująca mikrokontrolera podłączona jest poprzez rezystor 10 kΩ do linii zasilania. Ze względu na pozostałe elementy systemu (w szczególności moduł EM1000) wybrano napięcie zasilające o wartości 3,3 V.



Rys. 3. Architektura sprzętowa urządzenia

Na schemacie pokazano złącze SV1, do którego jest podłączony interfejs ISP umożliwiający programowanie pamięci *Flash* mikrokontrolera. Rozkład wyprowadzeń odpowiada popularnemu programatorowi STK200/300.

Za nadawanie sygnałów-rozkazów sterujących odpowiedzialny jest tranzystor kluczujący diodę podczerwieni. Baza tranzystora sterowana jest za pośrednictwem wyprowadzenia PC0. Interfejsem komunikacyjnym bloku pilota jest UART (wyprowadzenia PD0 i PD1).

Płytka pilota zawiera blok zasilania zbudowany w oparciu o regulator liniowy LDO LD1117-3V3. Zasila on kompletny system, a więc blok pilota, moduł EM1000 oraz moduł Wi-Fi WA1000.

Moduł EM1000 dostarczany jest bez obudowy, z wyprowadzonymi pinami (**fot. 5**). Koniecznie zatem było umieszczenie na płytce pilota złącz o odpowiednim rastrze.

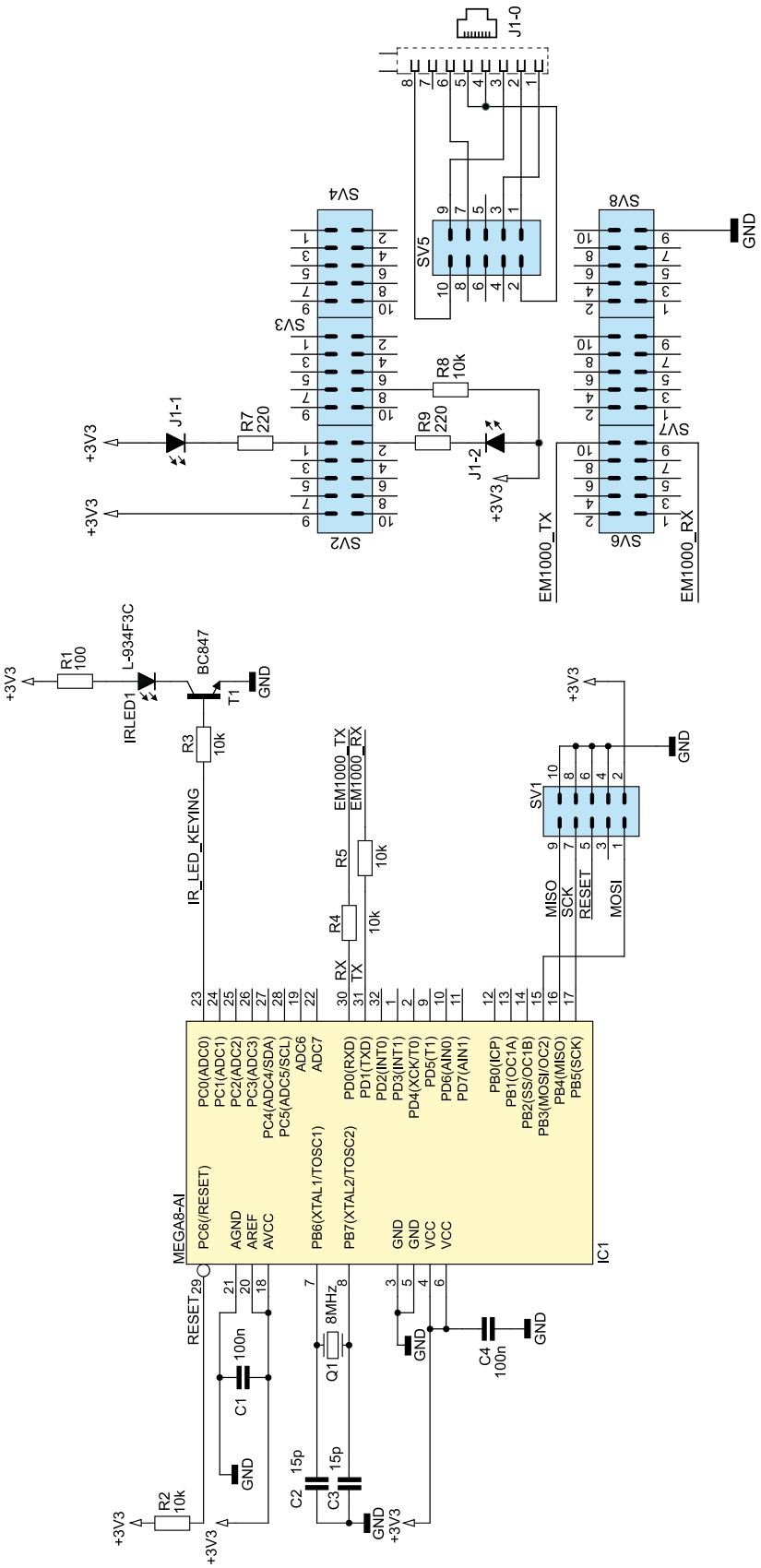
Do programowania modułu używany jest jego interfejs Ethernet, co wymusiło umieszczenie na płytce pilota złącza RJ45 z wbudowanym transformatorem separującym (J1). Do gniazd, w których umieszczony ma być EM1000 doprowadzono linie UART mikrokontrolera (J1) tak, aby możliwe było jego połączenie z UART0 modułu oraz podłączono dwie linie statusowe: SG (*Status Green*) i SR (*Status Red*) do wbudowanych w złącze J1 sygnalizacyjnych diod LED. Ponadto, do EM1000 doprowadzono zasilanie oraz podciągnięto linie MD (*Mode Selection Pin*) do zasilania układu. Moduł nie wymaga zewnętrznego sygnału resetu. Pełny rozkład wyprowadzeń modułu EM1000 pokazano na **rys. 6**, a schemat montażowy płytki bazowej na **rys. 7**.

Blok Pilota Podczerwieni

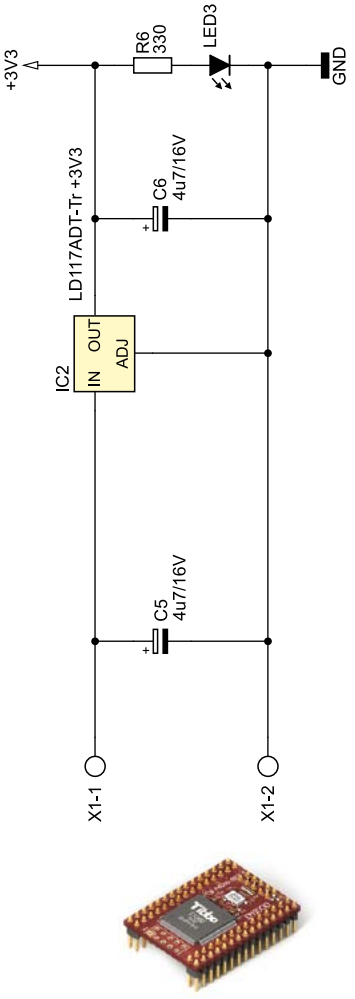
Budowa pilota sprowadziła się do wykonania nadajnika podczerwieni, zdolnego do

WYKAZ ELEMENTÓW
Rezystory (0603)
R1: 100 Ω
R7, R9: 220 Ω
R6: 330 Ω
R2...R5, R8: 10 kΩ
Kondensatory (0603)
C2, C3: 15 pF
C1, C4: 100 nF
C5, C6: 4,7 µF/16 V (EUC-6032/28)
Półprzewodniki
IC1: ATMega8-AI
IC2: LD1117-33CV (3,3 V)
T1: BC847
IRLED1: L-934F3C
LED3: dioda LED
Inne
Q1: kwarc 8 MHz
X1: złącze AK500/2
SV1...SV8: gniazdo goldpin 2,54 mm
J1: złącze 0569564-1

Na CD karty katalogowe i noty aplikacyjne elementów oznaczonych na Wykazie Elementów kolorem czerwonym



Rys. 4. Schemat ideowy bloku zasilania i pilota IR



Rys. 5. Moduł EM1000

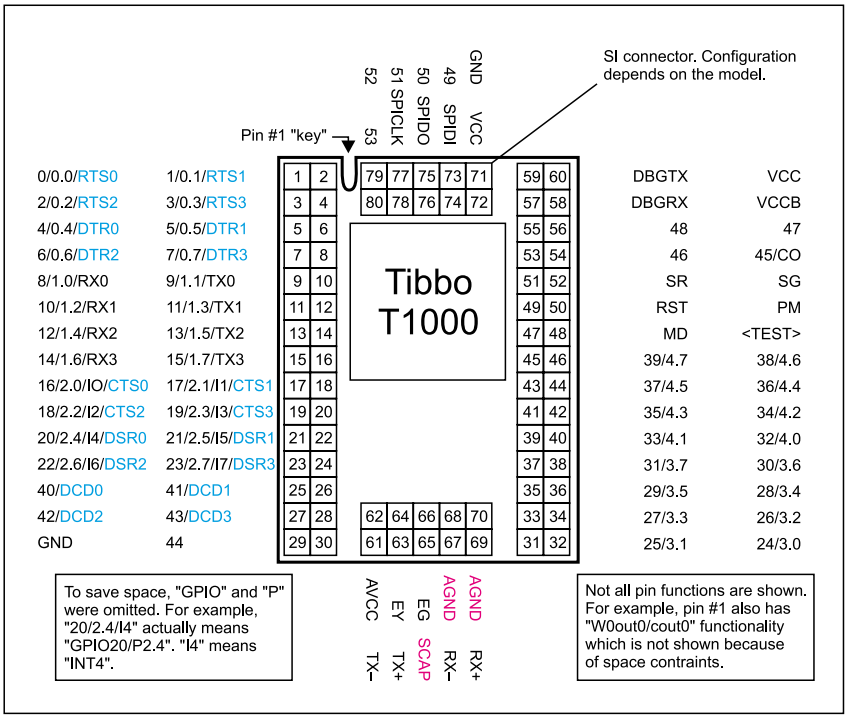
nadawania komend zgodnych z formatem transmisji stosowanym przez firmę Technics.

Zarówno generator fali nośnej sygnału podczerwieni, jak i samo kluczowanie sygnału nośnej, realizowane są przez mikrokontroler. Częstotliwość fali nośnej (prostokąt o wypełnieniu 50%) wynosi 32 kHz. Modulację sygnału można porównać do modulacji OOK, czyli On-Off Keying, często stosowanej w systemach radiowych pracujących w paśmie ISM. Polega ona na reprezentacji przesyłanej informacji bitowej poprzez obecność lub brak fali nośnej. Na **rys. 8** przedstawiono sposób kodowania „zera" i „jedynki".

Ramka danych (**rys. 9**) składa się z preambuły, którą tworzy ciągła fala nośna nadawana przez czas 4 ms. Następnie nośna jest wyłączana na 1,6 ms, po czym następuje nadanie komendy kodowanej w sposób przedstawiony na rys. 8.

Komenda składa się z 48 bitów, z których 32 jest tzw. kodem grupy, a kolejnych 16 bitów to kod rozkazu. Na końcu ramki przesyłany jest znak kończący transmisję, którym jest fala nośna włączona na czas 400 µs. Przed nadaniem kolejnej ramki należy odczekać minimum 76 ms. Ponieważ czas nadawania symbolu reprezentującego wartość 0 różni się o symbolu 1, to długość ramki (w sensie czasu) zależy od nadawanej komendy.

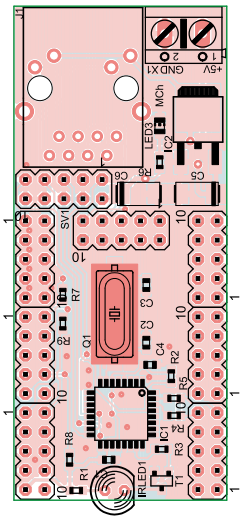
O ile sama implementacja protokołu nie nastręcza wielu problemów, o tyle dotarcie do właściwych kodów komend stanowi już spore wyzwanie (być może jest tylko w przypadku kodowania stosowanego przez Technics). Można go rozwiązać w dwojaki sposób: zbudować odbiornik i dekoder komend lub przechwytywać ramki oscyloskopem, mozolnie odczytywać i dekodować wyświetlony przebieg. W konkretnym przypadku zestawu audio, który miał być kontrolowany przy pomocy pilota, zawiodły też różnego rodzaju internetowe publikacje kodów komend. Najszybszą metodą okazał



Rys. 6. Wyprowadzenia modułu EM1000

Tab. 1 Zestawienie komend zaimplementowanych w module pilota

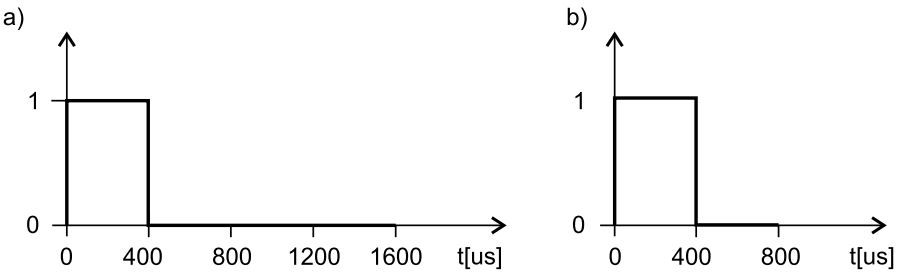
Lp.	Funkcja	Kod grupy	Kod rozkazu
1.	On/Off	0x40040538	0xBC81
2.	Volume +	0x40040538	0x0401
3.	Volume –	0x40040538	0x8481
4.	Tuner: Stacja 1	0x40040538	0x0835
5.	Tuner: Stacja 2	0x40040538	0x88B5
6.	Tuner: Stacja 3	0x40040538	0x4875



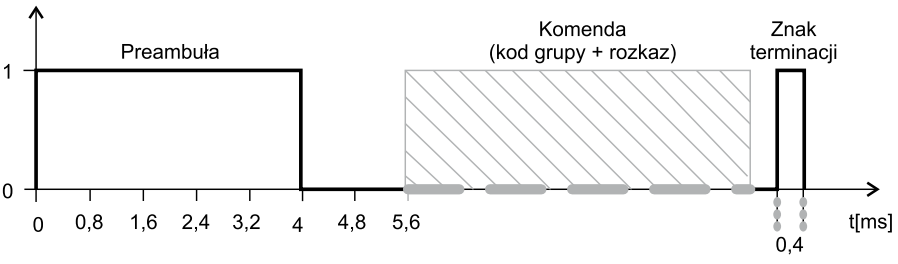
Rys. 7. Schemat montażowy płytki pilota

się oscyloskopowy *reverse engineering*, ale tylko dlatego, iż zgodnie z założeniami potrzebna była kontrola włączania/wyłączania zestawu, zmiana głośności oraz przełączanie pomiędzy trzema stacjami radiowymi – jest to niewiele prostych komend, na których rozpoznanie nie trzeba poświęcić zbyt wiele czasu. Zestawienie kodów tych komend umieszczono w **tab. 1**.

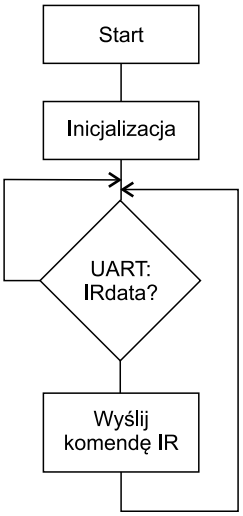
Firmware mikrokontrolera napisano został w środowisku AVR Studio z wykorzystaniem kompilatora AVRGCC. Generator fali nośnej używa Timera 0, w przerwaniu którego naprzemiennie zmieniany jest stan wyprowadzenia PORTC.0 sterującego tranzystorem kluczującym diodę podczerwieni. Za kluczowanie fali nośnej odpowiedzialne są dwie funkcje: *carrierOn()* i *carrierOff()*. Ich wywołanie powoduje odpowiednio włączenie lub wyłączenie przerwania Timera 0.



Rys. 8. Kodowanie zera i jedynki w standardzie Technics



Rys. 9. Ramka danych przyjęta w standardzie Technics



Rys. 10. Algorytm działania programu modułu pilota

Do nadawania właściwych komend podczernieni zastosowano maszynę stanów napędzaną przerwaniem Timera 1. Przerwanie to występuje co najmniejszy, wymagany przez protokół czas, a więc co 400 µs. Maszyna ma dziewięc stanów, które zgrupowano w typ wyliczeniowy:

```
typedef enum technicsState_tag
{
  TS_IDLE = 1,
  TS_INIT,
  TS_SEND_PREAMBLE_ON,
  TS_SEND_PREAMBLE_OFF,
  TS_SEND_GROUP,
  TS_SEND_CODE,
  TS_SEND_ONE,
  TS_SEND_ZERO,
  TS_SEND_STOP
} technicsState_t;
```

Tuż po starcie urządzenia maszyna pozostaje w stanie TS_IDLE, a każdorazowa zmiana wartości stanu na TS_INIT powoduje rozpoczęcie sekwencji maszyny. Do sterowania pilotem zastosowano komunikację szere-

List. 1. Główna pętla programu modułu pilota
```
while(1)
{
  switch(IRdata)
  {
    case '1':
      sendIRcommand(GROUP_TUNER, CH1);
      break;
    case '2':
      sendIRcommand(GROUP_TUNER, CH2);
      break;
    case '3':
      sendIRcommand(GROUP_TUNER, CH3);
      break;
    case '+':
      sendIRcommand(GROUP_AMP, CODE_VOL_UP);
      break;
    case '-':
      sendIRcommand(GROUP_AMP, CODE_VOL_DOWN);
      break;
    case 'o':
      sendIRcommand(GROUP_CTRL, ON_OFF);
      break;
    default:
      break;
  }
  IRdata = 0;
  _delay_ms(TRANSMISSION_DELAY);
}
```

na typy znakowe, konwersji czasu czy kalkulacji sum typu *md5* i *sha1*. Moduł wspiera też największą liczbę obiektów w środowisku TIDE. Z ciekawszych można wymienić obiekt *fd.* (flash disk) dostarczający bardzo wygodnego API do dostępu do wewnętrznej pamięci Flash. Obiekt *lcd.* (Display) – API do obsługi graficznych wyświetlaczy ciekłokrystalicznych. Obiekt *kp.* (Keypad) zdolny obsługiwać klawiaturę macierzową. Jednak najbardziej nas interesującym obiektem jest *wln.* (Wi-Fi) dostarczający podobnej funkcjonalności jak znany obiekt *net.* Jednak operującej na interfejsie bezprzewodowym. Aby można było go użyć należy do porty SPI modułu EM1000 dołączyć moduł WA1000 (**rys. 11**).

Moduł ten dostarcza funkcjonalności interfejsu bezprzewodowego 802.11b. Jest

List. 2. Linie odpowiedzialne za nadanie adresów MAC, IP, Gateway oraz maski podsieci
```
wln.mac="0.0.102.103.104.2"        '<--------- replace with random MAC (see „Setting MAC Address")
wln.ip="192.168.1.93"              '<--------- select an IP address which is compatible with your network
wln.gatewayip="192.168.1.1"        '<--------- this too
wln.netmask="255.255.255.0"        '<--------- and this
```

List. 3. Uruchomienie modułu WA1000
```
romfile.open("wln_fwar.bin")
x=wln.boot(romfile.offset)
```

gową z użyciem UARTa, który w przerwaniu oczekuje na nadchodzące znaki i przepisuje je do zmiennej globalnej: *unsigned char IRdata = 0;*

Na **rys. 10** przedstawiono algorytm programu, a na **list. 1** kod głównej pętli programu.

Pętla, co czas zdefiniowany przez stałą TRANSMISSION_DELAY (80 ms), sprawdza zawartość zmiennej globalnej IRdata. W przypadku, gdy UART odebrał znak rozpoznawany przez instrukcję *switch*, następuje wywołanie funkcji *sendIRcommand()* z właściwymi kodami grupy i rozkazu. Funkcja ta tworzy kopię argumentów i zmienia stan maszyny na TS_INIT, co rozpoczyna proces generowania komendy przez diodę poczerwieni.

Na tym etapie możliwe było przetestowanie układu pilota z dowolnym komputerem wyposażonym z port szeregowy. Kolejnym krokiem było wyposażenie urządzenia w interfejs WLAN.

EM1000+WA1000

Jako baza do implementacji interfejsu bezprzewodowego posłużył programowalny moduł EM1000. Jest to najsilniejsza platforma programowalna firmy Tibbo. Moduł z założenia jest przeznaczony do zaawansowanych aplikacji sieciowych i nie jest dostarczany z jakąkolwiek preprogramowaną aplikacją (jak ma to miejsce w przypadku prostszych modułów). Aplikację musimy stworzyć sami.

Właściwie cały moduł zamyka się w jednym chipie (ASIC). Na jego płytce drukowanej dostrzeżemy ponadto kontroler warstwy sprzętowej (PHY) firmy Davicom (DM9000A-EP), zewnętrzny oscylator kwarcowy oraz zewnętrzną pamięć nielotną typu EEPROM (2 kB). Sercem modułu jest mikrokontroler (prawdopodobnie jest on oparty o rdzeń '51 lub '51 chodź producent nie podaje wprost tej informacji) taktowany zegarem aż 88 MHz. Posiada jeden port Ethernetowy z funkcją Auto-MDIX (rozpoznawanie typu podłączonego kabla: prosty czy skrosowany), 4 szybkie porty szeregowe, 512 kB Flash, z której pierwsze 64 kB są zarezerwowane na obraz systemu operacyjnego natomiast reszta przeznaczona jest na aplikacje użytkownika (z której również mamy programowy dostęp do pamięci flash). Moduł wyposażony jest też w zegar czasu rzeczywistego (RTC) oraz 54 linie GPIO. Część z nich posiada kilka funkcji w zależności od bloku, z jakim są połączone. Moduł posiada też specjalne linie szybkiego interfejsu SPI służącego do komunikacji z układami peryferyjnymi (jak np. moduł Wi-Fi). Pośród linii znajdziemy też wejścia przerwań zewnętrznych, oraz kilka linii sygnalizacyjnych (część z nich opisana jest wcześniej). Do programowania modułu i debugowania kodu nie są wymagane żadne dodatkowe narzędzia (np. JTAG) ponieważ cała komunikacja ze środowiskiem developerskim TIDE odbywa się poprzez połączenie Ethernetowe.

Od strony programowej moduł wspiera wszystkie typy danych: typy całkowite o rozmiarach od jednego do czterech bajtów, typ Real (4 bajty), typ łańcucha znakowego (string), tablice oraz struktury/typy definiowane przez użytkownika. System operacyjny (TiOS) dostarcza wiele procedur systemowych min konwersji typów numerycznych

montowany na stosie na module EM1000 który dodatkowo dostarcza dla niego zasilanie. WA1000 wyposażony jest w antenę typu *chip*, jak również posiada złącze typu *I-PEX MHF* służące do podłączenia zewnętrznej anteny.

Oprogramowanie modułu EM1000 – TIBBO BASIC

Dysponując już wszystkimi elementami sprzętowymi systemu można było przystąpić do ostatniej części projektu: stworzenia firmware'u dla modułu EM1000.

Podstawowy kod źródłowy zawarto w jednym pliku TIBBO Basic oraz w kilku prostych plikach .html implementujących interfejs użytkownika. Działanie programu sprowadza się do przypisaniu odpowiednich adresów MAC, IP, Gateway oraz maski podsieci (**list. 2**).

Kolejnym krokiem jest uruchomienie samego modułu bezprzewodowego. Kod źródłowy programu odpowiedzialnego za współpracę z modułem Wi-Fi pokazano na **list. 3**. Przy pomocy metody romfile.open uzyskujemy wskaźnik na obszar pamięci typu ROM. W tym przypadku był to plik binarny dołączony do projektu. Plik ten zawiera obraz firmware dla modułu WA1000. Następna metoda (*wln.boot*) rozpoczyna właściwy proces transmisji kodu do modułu WA1000. Dopiero po tych czynnościach moduł uaktywnia swój interfejs radiowy.

Jeśli poprzednie operacje przebiegły prawidłowo możemy przystąpić do próby po-

Rys. 11. Wygląd modułu WA1000

List. 4. Sekwencja ustawień niezbędnych do połączenia z siecią bezprzewodową

```
wln.bssmode=PL_WLN_BSS_MODE_INFRASTRUCTURE
wln.defaultibsschannel=6
wln.ssid="AirImprezaKamikaze" '<--------- set the name of your wireless LAN
wln.wepmode=PL_WLN_WEP_MODE_64 '<--------- make correct choice („Setting WEP Mode and Keys")

if wln.wepmode<>PL_WLN_WEP_MODE_DISABLED then
wln.wepkey1="2b176c9071" '<--------- set keys if necessary
        wln.wepkey2="22222222222222222222222222"
        wln.wepkey3="33333333333333333333333333"
        wln.wepkey4="44444444444444444444444444"

        x=wln.setwep
        task=wln.task
        while wln.task<>PL_WLN_TASK_IDLE
        wend
end if
        .
        .
        .
x=wln.associate
```

List. 5. Kod odpowiedzialny za wysłanie do modułu pilota polecenia nadania komendy podczerwieni

```
<?
        include „global.tbh"
?>

<!doctype html public „-//w3c//dtd wc html//en">
<html>
<body>
<meta http-equiv="Refresh" content="1 url=index.html">
<font size="4" face="Verdana">
  S1
  <br>
<<<!!! ------------- SZARE TŁO ODTĄD ------------ !!!>>>
  <?
    'ir command
    ser.setdata(„1")
    ser.send
  ?>
<<<!!! ---------------- DOTĄD -------------- !!!>>>
  Going back to index...
</font>
</body>
</html>
```

łączenia z siecią bezprzewodową. Na **list. 4** pokazano fragment kodu odpowiedzialnego za wybór rodzaju sieci, ustawienia kanału interfejsu radiowego, określenie nazwy sieci (SSID), z którą moduł nawiązać ma połączenie. Kolejnym krokiem jest ustalenie sposobu szyfrowania transmisji oraz przypisanie klucza. Po tych czynnościach możemy nawiązać połączenie z siecią wywołując metodę *wln.associate*.

Na module EM1000 uruchomiony jest serwer WWW dostarczający interfejsu użytkownika pilota. Po wpisaniu w przeglądarce internetowej na dowolnym urządzaniu, które ma dostęp do tego samego segmentu sieci (może to być np. laptop lub PDA), pilot odsyła swoją stronę internetową o wyglądzie takim, jak na **rys. 12**.

Strona składa się z kilku przycisków formularza, za które odpowiada poniższy kod (kod dotyczy jednego przycisku):

```
<form action="s1.html"
method="get">
<input type="submit"
style="height: 40px; width:
100px" value="Roxy FM">
</form>
```

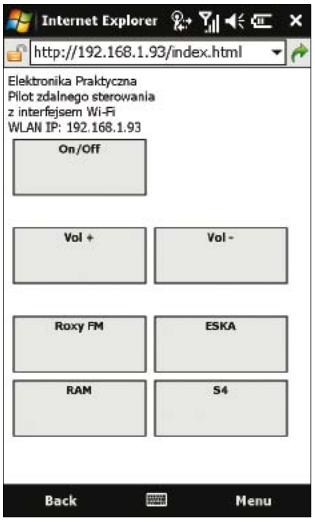
Po wybraniu danego przycisku do użytkownika zwracana jest podstrona taka jak na **rys. 13**, po czym, następuje przekierowanie na stronę główną. Jednak najważniejsza akcja dzieje się właśnie po wywołaniu tej niepozornej strony, której kod widoczny jest na **list. 5**.

Kolorem szarym zaznaczono fragment skryptu Tibbo Basic, który umieszcza w buforze nadawczym UART'u kod ASCII znaku „1" po czym następuje jego wysłanie a tym samym wysterowanie modułu pilota podczerwieni.

Podsumowanie

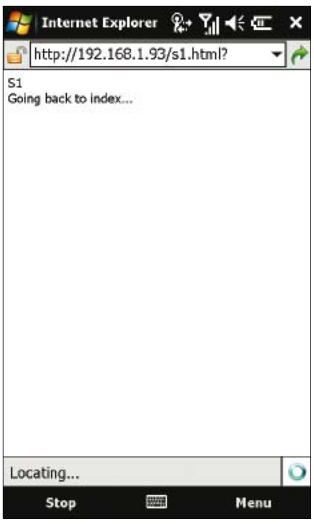
Moduł WA1000 jest przykładem produktu, który po prostu działa. Dostarczany jest ze świetną dokumentacją i przykładami. Dla profesjonalistów zajmujących się integracją systemów sterowania opartych o sieci Wi-Fi ciężko byłoby znaleźć równie dobrą alternatywę. Jednak z drugiej strony, jest to półprodukt, ponieważ zawsze musi być używany razem z modułem EM1000.

Ciekawym posunięciem firmy Tibbo (z punktu widzenia konstruktorów i entuzjastów systemów Wi-Fi) byłoby opublikowanie przez producenta protokołu i zestawu rozkazów SPI, przy pomocy których można komunikować się z modułem. Pozwoliłoby to na szersze jego zastosowanie, ponieważ użytkownik nie byłby zmuszony do używania modułów EM1000 oraz samego języka Tibbo Basic. Szanse są jednak nikłe, bo główną wartością wszystkich produktów sprzętowych Tibbo jest łatwość ich programowania w łatwo przyswajalnym języku Tibbo Basic. Co natomiast byłoby realne (przynajmniej w odczuciu autora), to opracowanie tańszej platformy bazowej dla WA1000. Rzadko bowiem (oczywiście w przypadku prostszych systemów) potrzebujemy platformy sieciowej wyposażonej zarówno w port Ethernet jak i Wi-Fi. Ma to potwierdzenie w tym projekcie, kiedy to interfejs Ethernet był wykorzystywany wyłącznie do programowania i debugowania modułu.



Rys. 12. Wygląd głównej strony pilota

Marcin Chruściel, EP
marcin.chrusciel@ep.com.pl



Rys. 13. Podstrona wysyłana do użytkownika po kliknięciu na przycisk S1 (Roxy FM)