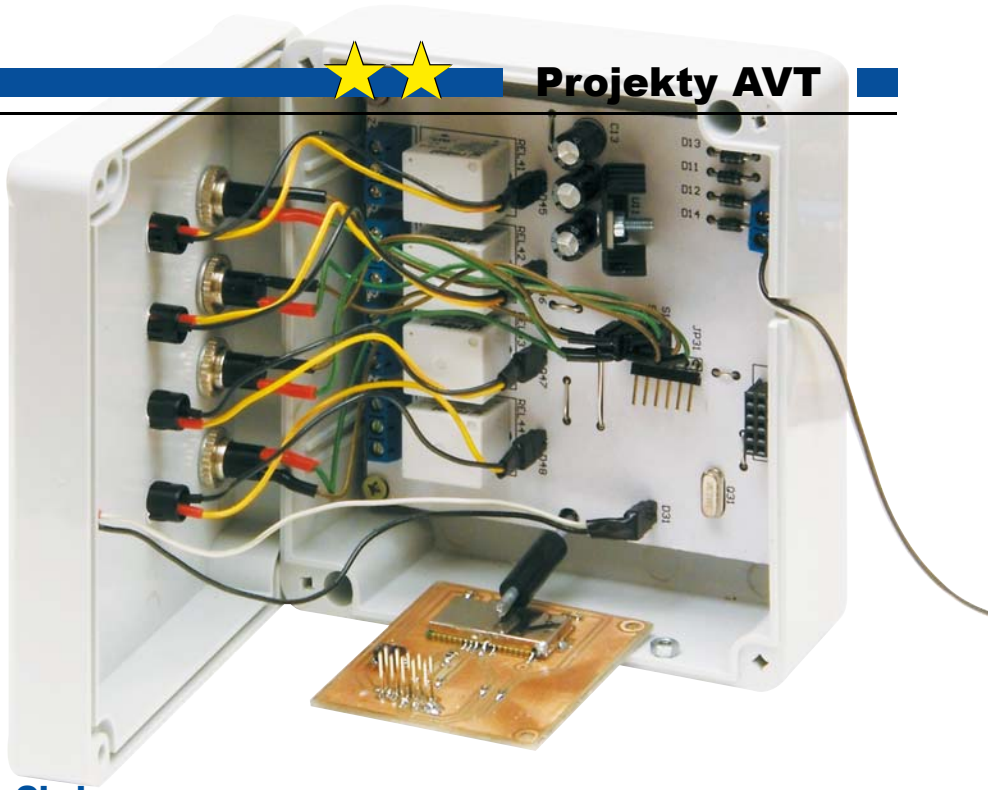
kit 2938 AVT

Blue Supply



Bezpośrednią przyczyną powstania niniejszego projektu było zadanie 159 postawione w Szkole Konstruktorów (oszczędzanie energii) oraz obserwacja, że spora ilość prądu marnuje się z powodu lenistwa. Drukarka stojąca na szafce czasami jest włączona przez parę godzin, bo nie chce mi się odrywać od pracy i iść jej wyłączyć. Stąd właśnie pomysł wykonania urządzenia, które umożliwi zdalne wyłączanie drukarki, światła, wentylatora, etc.

Układ będący przedmiotem niniejszego artykułu został nazwany Blue Supply, ponieważ jest przeznaczone do sterowania zasilaniem urządzeń elektrycznych za pomocą Bluetooth. W Elportalu można znaleźć aplikację komputerową oraz aplet napisany w Javie dla telefonów komórkowych. Programy te umożliwiają niezależne sterowanie czterema przekaźnikami, w które projekt został wyposażony. Oprócz tego do układu dodana została prosta klawiatura złożona z czterech przycisków, którymi można ręcznie zmienić stan dołączonych urządzeń.

Ponieważ projekt zachcą zrealizować także mniej doświadczeni i młodsi wiekiem Czytelnicy, więc z układu zostały wyeliminowane obwody sieci elektrycznej. Konieczny jest zewnętrzny zasilacz DC lub AC na napięcie około 12V (może być niestabilizowany). Do prawidłowej pracy potrzebny jest zaprogramowany mikrokontroler. W Elportalu dostępny jest kod wynikowy, który nie wymaga żadnych zmian, wystarczy nim zaprogramować procesor. W razie problemów można poprosić o pomoc bardziej doświadczonego kolegę lub zamówić zaprogramowany układ w sklepie AVT.

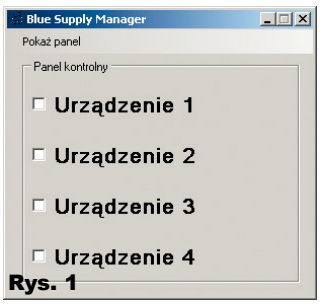
Obsługa

Przed pierwszym uruchomieniem urządzenia należy wywołać program konfiguracyjny. Czynność ta polega na przytrzymaniu dowolnego przycisku i włączeniu zasilania. Dioda LED_STAT zacznie szybko migotać, co będzie oznaczało, że mikrokontroler nadaje nazwę modułowi Bluetooth zainstalowanemu w urządzeniu, ustawia domyślny klucz dostępu i wyłącza echo lokalne. Dodatkowo do EEPROM zostaną zapisane nazwy przekaźników (*DEVICE 1*, *DEVICE 2*, etc.). Uruchomienie programu konfiguracyjnego nie jest niezbędne, ale warto to zrobić, gdyż w przeciwnym razie nazwą urządzenia będzie wartość domyślna producenta (Serial Adaptor), a w panelu kontrolnym wyświetlą się „śmieci" zamiast nazw przekaźników.

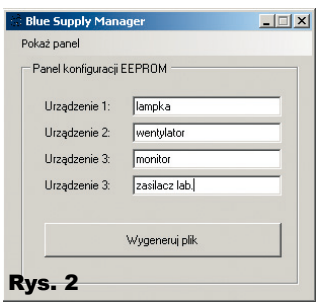
W Elportalu dostępna jest prosta aplikacja *Blue Supply Manager*, która umożliwia zdalne zarządzanie pracą urządzenia. Po pierwszym uruchomieniu aplikacji pokazuje się okienko jak na **rysunku 1**. W menu *Pokaż panel* dostępne są cztery opcje:

– kontrolny – pokazuje panel do sterowania przekaźnikami (rysunek 1); za pomocą pól typu *checkBox* można sterować pracą poszczególnych przekaźników,
– konfiguracji EEPROM – umożliwia wygenerowanie pliku, który należy wgrać do pamięci EEPROM, w efekcie czego każdy przekaźnik otrzyma swoją nazwę; panel ten został pokazany na **rysunku 2**; należy wypełnić wszystkie pola, podając własne nazwy (do 20 znaków) i kliknąć przycisk Wygeneruj plik (pojawi się on w katalogu z programem),
– zdalna konfiguracja nazw – generowanie pliku i programowanie pamięci EEPROM jest dość niewygodne, więc istnieje jeszcze inna możliwość przypisania nazw do przekaźników przez Bluetooth – właśnie za pomocą tego panelu (**rysunek 3**); w polu tekstowym należy wprowadzić nazwę, wybrać urządzenie, którego ma się tyczyć ta nazwa (np. *Urządzenie 1*) i nacisnąć *Wyślij do urządzenia*; po chwili w nawiasie powinna pojawić się nowo nadana nazwa,
– konfiguracja portu szeregowego – po nawiązaniu połączenia z Blue Supply w systemie pojawi się nowy port szeregowy (wirtualny), który należy wskazać na rozwijalnej liście i kliknąć *Zachowaj*; po chwili powinien pojawić się komunikat z informacją o poprawnym otwarciu portu; zostanie on zachowany w pliku, więc przy kolejnym uruchomieniu aplikacji nie trzeba będzie ponownie go konfigurować.

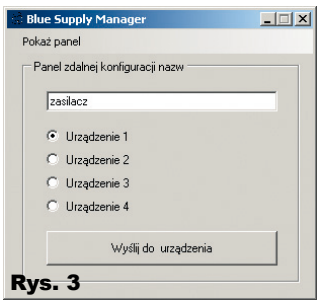
Do ustanowienia połączenia komputera z urządzeniem *Blue Supply* konieczne jest wykonanie kilku prostych kroków. Zostaną one tu pokrótce omówione na przykładzie zewnętrznego układu Bluetooth podłączanego do USB i pracującego ze standardowymi sterownikami Windows XP. Najprościej zabrać się do tego, otwierając *Panel Sterowania* (*Start->Ustawienia*) i klikając ikonę *Urządzenia Bluetooth*. W otwartym okienku (**rysunek 4**) należy kliknąć przycisk *Dodaj*. Spowoduje to otwarcie kolejnego okienka (**rysunek 5**). Wymagane jest tu jedynie zaznaczenie opcji potwierdzającej, że szukane urządzenie jest gotowe do nawiązania komunikacji i kliknięcie *Dalej*. Następnie rozpocznie się wyszukiwanie, którego rezultaty można oglądać w kolejnym okienku (**rysunek 6**). Po odnalezieniu *Blue Supply* należy zaznaczyć urządzenie, klikając na nim raz i wybierając *Dalej*. Przedostatni krok (**rysunek 7**) polega na wprowadzeniu klucza dostępu, którym jest wartość domyślna producenta, czyli *1234*. Po jej



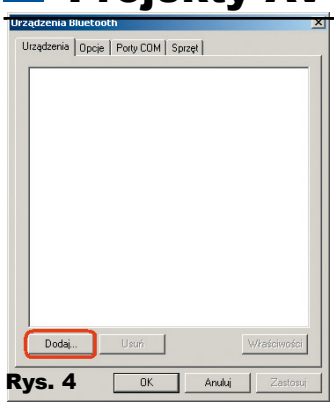
Rys. 1



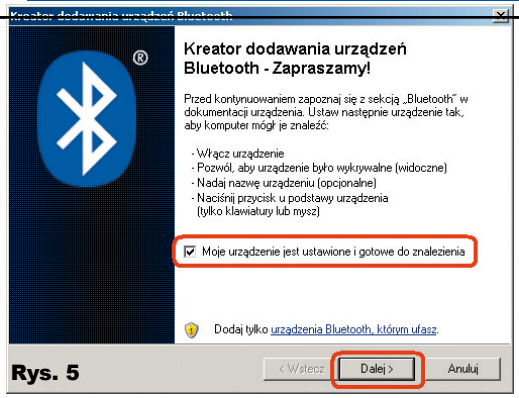
Rys. 2



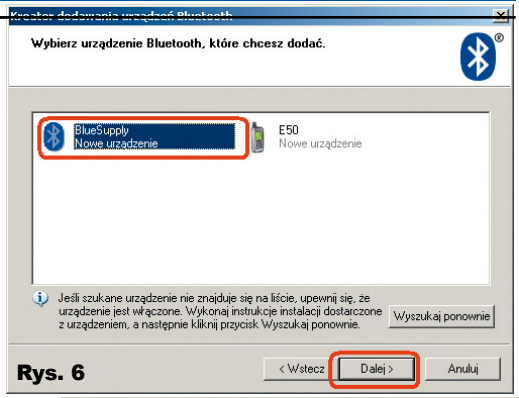
Rys. 3



Rys. 4



Rys. 5



Rys. 6

wprowadzeniu należy kliknąć *Dalej*. Ukaże się podsumowanie, które można zamknąć, zapamiętując uprzednio, pod jakim portem zainstalował się moduł Bluetooth (**rysunek 8**).

Po wprowadzeniu do aplikacji *Blue Supply Manager* nazwy portu z rysunku 8 powinien on zostać otwarty i umożliwić komunikację z modułem. Gdy operacja ta zakończy się sukcesem, dioda LED_STAT zacznie pulsować, do pól w panelu kontrolnym wczytane zostaną stany przekaźników oraz pobrane zostaną ich nazwy.

Omawiając obsługę urządzenia, warto jeszcze wspomnieć o aplikacji przeznaczonej na komórkę, która była testowana na Nokii 2760. Została ona tam załadowana za pomocą tego samego modułu Bluetooth podłączonego przez USB do komputera. Proces wgrywania aplikacji jest prosty i zaczyna się od nawiązania połączenia z komórką podobnie jak na rysunkach 4 do 8. W zasadzie pojawiają się tylko dwie różnice:

– w okienku z rysunku 6 zaznaczamy komórkę zamiast Blue Supply,
– w okienku z rysunku 7 można zaznaczyć opcję *Wybierz dla mnie klucz dostępu* i wygenerowany klucz wprowadzić do komórki (powinna sama o niego poprosić).

Oczywiście, wcześniej należy włączyć moduł Bluetooth w telefonie.

Aplikację wysyła się do komórki poprzez kliknięcie prawym przyciskiem myszy na pliku *BlueSupply.jar* i wybranie pozycji *Urządzenie Bluetooth* (**rysunek 9**). Należy skorzystać z przycisku *Przeglądaj* (**rysunek 10**) i odnaleźć na liście komórkę, zaznaczyć, zaakceptować (przycisk OK) i kliknąć *Dalej*. W ten sposób aplikacja trafi do telefonu komórkowego. Po jej uruchomieniu aparat może wyświetlić pytanie, czy włączyć Bluetooth, na co należy się zgodzić. Rozpocznie się wyszukiwanie pobliskich urządzeń i dodawanie ich do listy. Za pomocą klawiszy nawigacyjnych należy zaznaczyć pozycję *BlueSupply* i zaakceptować wybór przyciskiem OK. Jeżeli pojawi się pytanie, czy aplikacja może korzystać z dostępnych łączy, należy na to zezwolić. Następnie powinno zostać załadowane właściwe menu zawierające cztery pozycje typu *checkBox*, które można zaznaczać i odznaczać powodując, odpowiednio, włączenie i wyłączenie przekaźników.

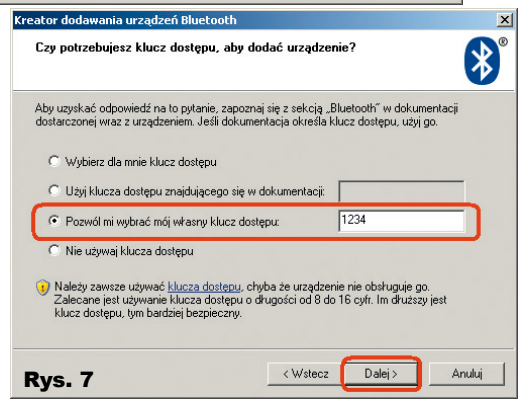
Zmiana stanu przekaźników za pomocą klawiatury umieszczonej na płycie czołowej urządzenia spowoduje również odpowiednią zmianę w panelu kontrolnym aplikacji komputerowej i midletu uruchomionego w telefonie komórkowym.

UWAGA!!! Nie można jednocześnie obsługiwać urządzenia z poziomu komputera i komórki! Może to doprowadzić do zawieszenia oprogramowania.

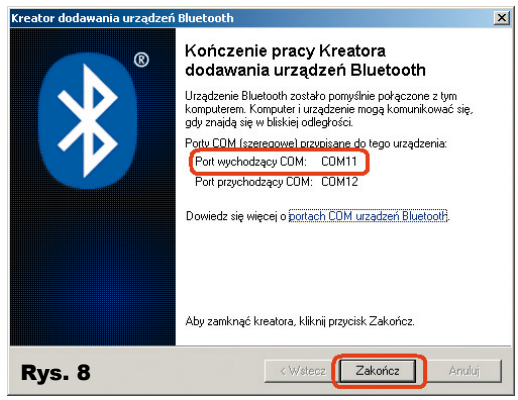
Opis układu

Schemat urządzenia przedstawiono na **rysunku 11**. Zasilacz to typowa aplikacja. Pierwszy stabilizator (7805) w zasadzie nie jest używany, ale ogranicza straty mocy w stabilizatorze U12. Łatwiej na nim zamocować radiator i odprowadzić ciepło niż z układu U12, który jest montowany powierzchniowo. Układ można też zasilać napięciem zmiennym 9...12V (inne będzie nieodpowiednie dla przekaźników). Minimalna wydajność prądowa zasilacza (transformatora) to 200mA, lepiej 300mA lub więcej. Wybór napięcia 3,3V został wymuszony obecnością modułu Bluetooth. Dzięki zasileniu procesora takim samym napięciem możliwe było uproszczenie konstrukcji.

Drugi blok stanowi mikrokontroler ATtiny2313 i jego otoczenie. Komunikacja z modułem Bluetooth przebiega z wykorzystaniem łącza asynchronicznego, stąd obecność rezonatora kwarcowego. Do programowania można wykorzystać tryb ISP i związane z nim linie SCK, MISO, MOSI oraz Reset, które zostały wyprowadzone na złącze JP31. Złącze JP32 zostało przeznaczone do podłączenia modułu Bluetooth BTM-222. Jest on dostępny w polskich sklepach elektronicznych (w Internecie) w cenie, powiedzmy, przystępnej. Moduł Bluetooth musi zostać umieszczony na specjalnej przejściówce, której schemat pokazano na **rysunku 12**.



Rys. 7



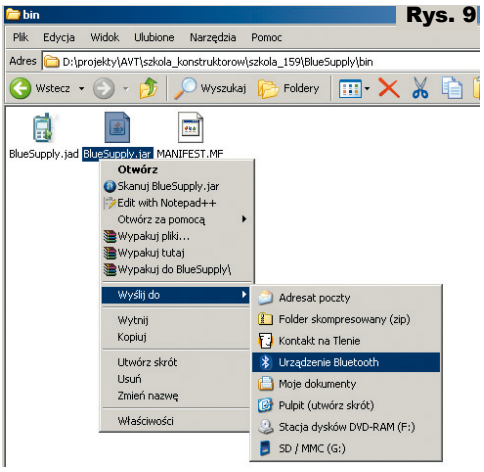
Rys. 8

Ostatni blok stanowią przekaźniki i elementy stowarzyszone – cztery takie same bloki wykonawcze.

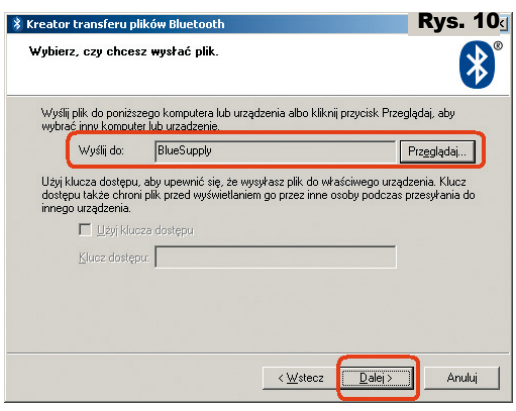
Protokół komunikacyjny

Udostępnione podstawowe aplikacje do sterowania pracą urządzenia są proste – nie zawierają żadnej automatyki. Nie ma możliwości np. okresowego załączania urządzeń.

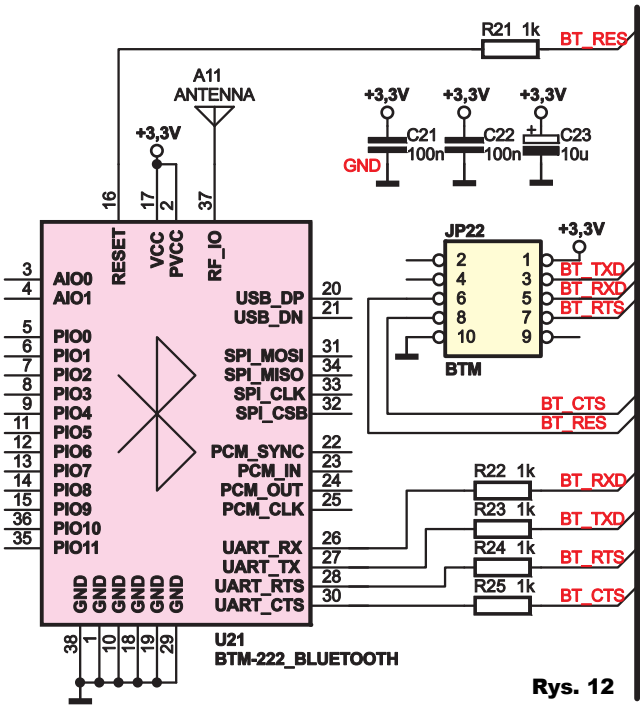
Taka opcja może być przydatna, więc warto przybliżyć szczegóły komunikacji. Pozwoli to na samodzielne napisanie dodatkowego programu i dodanie potrzebnych funkcji.



Rys. 9



Rys. 10



Rys. 12

Otóż komunikacja przebiega z wykorzystaniem typowego portu szeregowego, w sposób wirtualny, kanałem radiowym. Port ten można obsłużyć za pomocą standardowych kontrolek do obsługi portu szeregowego (np. *serialPort* w środowisku Visual Studio).

Najprostszym poleceniem jest bajt o wartości 0xFF (255) powodujący zmianę stanu diody LED_STAT na przeciwny. Jak łatwo się domyślić, obie aplikacje (komputerowa oraz komórkowa) przesyłają to polecenie okresowo i dzięki temu dioda miga po nawiązaniu połączenia.

Następne polecenie to bajt o wartości z zakresu 0xA0..0xAF. Cztery najstarsze bity informują, że należy wymusić stan przekaźników zgodny z zawartością czterech najmłodszych bitów (jedynka włącza, zero wyłącza). Graficzna reprezentacja tego polecenia została przedstawiona w **rysunku 13**. Przykładowo, wysyłanie bajtu o wartości 0xA1 (bitowo: 1010 0001) spowoduje włączenie przekaźnika REL41 i wyłączenie wszystkich pozostałych.

Problem w tym przypadku stanowi nieznajomość stanu początkowego, czyli włączając dowolny przekaźnik, nie wiemy, jaki stan mają inne i co umieścić na pozostałych bitach. Z tego względu trzeba określić, jaki stan mają przekaźniki przed dokonaniem zmiany stanu jednego z nich. Służy do tego polecenie 0xB0. Wysyłając bajt o takiej wartości po chwili otrzymamy inny bajt. Będzie miał on wartość z przedziału 0xB0...0xBF, a cztery ostatnie bity poinformują o stanie przekaźników (**rysunek 14**). Po uruchomieniu aplikacji wysyłany jest bajt 0xB0, odbierana jest odpowiedź i stan przekaźników zostaje zapamiętany w lokalnej zmiennej. Mając tę wartość, można zmienić stan tylko wybranego przekaźnika, pozostawiając pozostałe bez zmiany. Oczywiście może się zdarzyć sytuacja, że naciśnięty zostanie przycisk na obudowie urządzenia i stan przekaźnika zmieni się poza aplikacją. W takiej sytuacji mikrokontroler sam odsyła bajt o wartości 0xB0...0xBF informując o obecnym stanie przekaźników. Informacja ta jest odbierana przez aplikację, zapamiętywana i zmieniane są wartości pól *checkBox*.

W zasadzie te trzy proste polecenia są wystarczające do sterowania pracą urządzenia. Są jeszcze dwie dodatkowe komendy, trochę bardziej złożone, które pozwalają zmienić bądź odczytać nazwę przypisaną do danego przekaźnika. Po wysłaniu bajtu o wartości z przedziału 0xC1...0xC4 mikrokontroler odeśle w odpowiedzi ciąg zaczynający się od tego samego bajtu, następnie ciąg znaków ASCII, będących nazwą przypisaną do danego przekaźnika i na końcu bajt o wartości 0x00. Przykładowo wysyłając 0xC1 odesłanie zostanie następująca sekwencja:

0xC1, ‘L’, ‘A’, ‘M’, ‘P’, ‘A’, 0x00.

Pierwszy bajt określa, że przesyłana jest nazwa urządzenia (najstarsze 4 bity), jest to pierwsze urządzenie (cztery młodsze bity kodowane w systemie BCD). Następnie pojawią się bajty w kodzie ASCII określające nazwę przekaźnika i ciąg znaków kończy znakiem NULL (0x00).

W zasadzie ustawienie nazwy urządzenia wygląda identycznie, z tym że zamiast kodu z przedziału 0xC1...0xC4 wysyłany jest kod z przedziału 0xD1...0xD4. Następnie wysyłane są znaki ASCII stanowiące nazwę (maks. 20) zakończone znakiem NULL. Przykładowo chcąc ustawić nazwę dla przekaźnika REL42 należy przesłać następujący ciąg bajtów:

0xD2, ‘D’, ‘R’, ‘U’, ‘K’, ‘A’, ‘R’, ‘K’, ‘A’, 0x00.

Oprogramowanie

Pełny kod źródłowy sterujący pracą mikrokontrolera udostępniono w Elportalu. Został on napisany w środowisku WinAVR i podzielony na klasy, aby zgrupować pewne funkcje w logiczną całość. Klasa *eeprom* dostarcza dwóch funkcji pozwalających na zapis bądź odczyt pojedynczego bajtu z określonego adresu w wewnętrznej pamięci EEPROM mikrokontrolera.

Klasa *device* udostępnia metody do sterowania pracą przekaźników, diod LED oraz umożliwiającą odczyt stanu klawiatury. Funkcja *key* prawidłowo odczytuje stan klawiatury, ignorujac drgania styków i długość czasu naciskania przycisków. Warto zauważyć, że funkcja

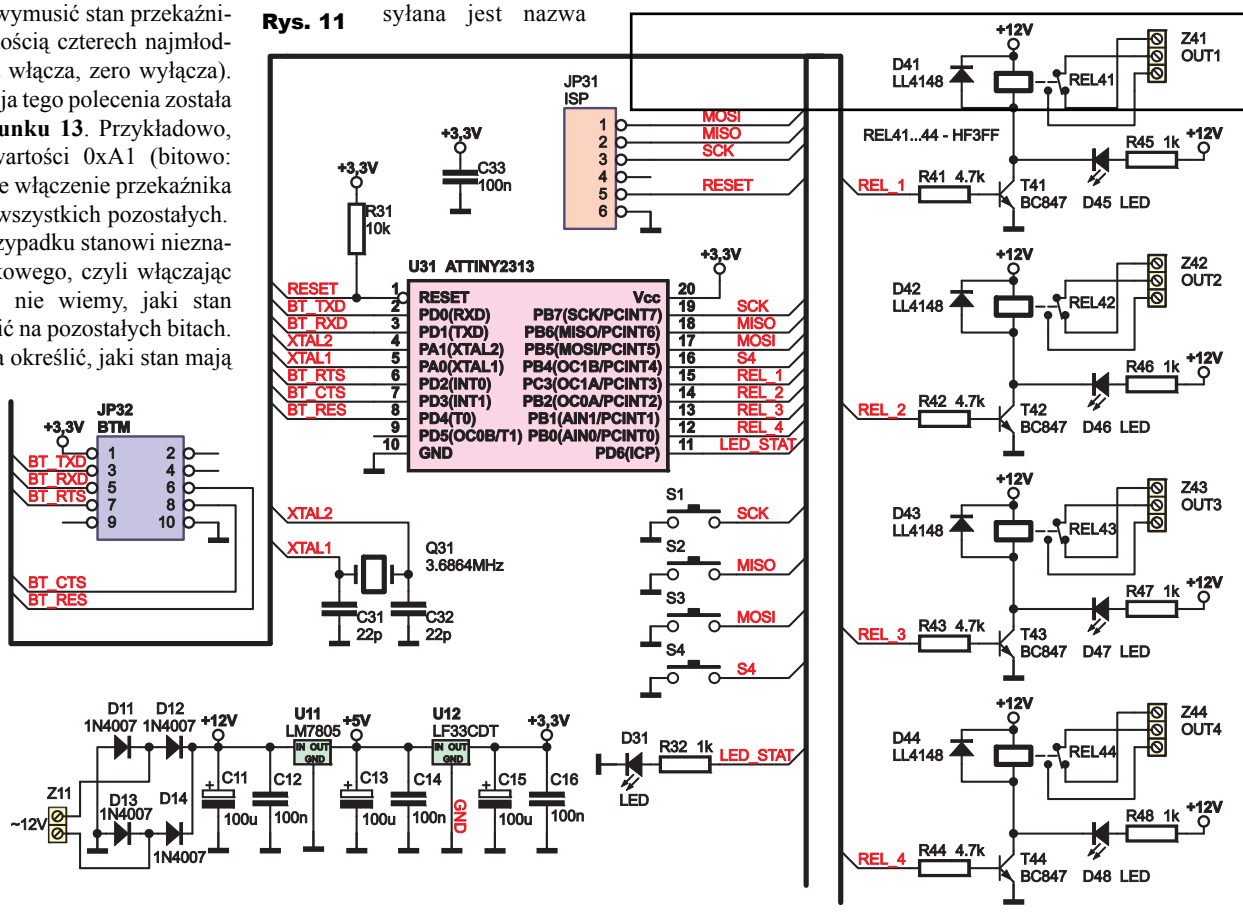
Rys. 13

	MSB							LSB
bit:	7	6	5	4	3	2	1	0
wartość:	1	0	1	0	DEVICE 4	DEVICE 3	DEVICE 2	DEVICE

Rys. 14

	MSB							LSB
bit:	7	6	5	4	3	2	1	0
wartość:	1	0	1	1	DEVICE 4	DEVICE 3	DEVICE 2	DEVICE 1

Rys. 11



obsługująca klawiaturę działa w sposób nieblokujący, czyli program po krótkim „postoju" (20ms) może wykonywać się dalej.

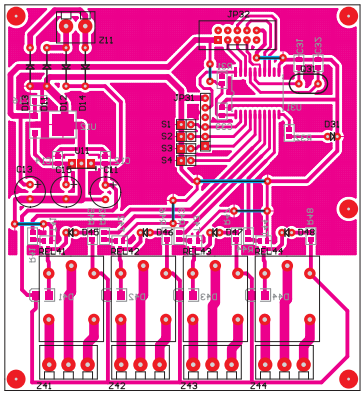
Klasa *btm222* zajmuje się obsługą modułu BTM-222. Konstruktor odpowiada za ustawienie portów I/O, reset modułu, konfigurację układu USART oraz wykonanie funkcji *setupBTM* po stwierdzeniu, że naciśnięto przycisk przed załączeniem zasilania. Włączane są także przerwania. Funkcja *setupBTM* ustawia nazwę modułu BTM-222 na Blue Supply, wyłącza lokalne echo, ustawia klucz dostępu oraz zapisuje w pamięć EEPROM domyślne nazwy urządzeń. Nie jest to może najbardziej intuicyjne miejsce do konfiguracji EEPROM, ale postanowiłem zebrać wszystkie ustawienia w jednym miejscu.

Funkcja *interrupt* jest wywoływana po odebraniu każdego znaku z portu szeregowego. Odpowiada ona za zwiększanie wartości zmiennej określającej liczbę odebranych, nieprzeczytanych bajtów i zapis tych bajtów do bufora w celu późniejszego odczytu. Funkcja *readChar* zwraca najwcześniej odebrany, nieczytany bajt z bufora. Przy okazji przesuwana jest zawartości tablicy stanowiącej bufor, aby zrobić miejsce dla następnych znaków i zmniejszana jest wartość licznika odebranych bajtów. Do wysyłania danych przeznaczona została funkcja *writeChar*, a do sprawdzania, czy jest coś w buforze odbiorczym – *isNewData*.

Program główny, przedstawiony na **listingu 1**, rozpoczyna się od obsługi klawiatury. Po stwierdzeniu, że jeden z klawiszy został wciśnięty, następuje przełączenie przekaźnika w stan przeciwny przy pomocy funkcji *relayInvert* zdefiniowanej wewnątrz klasy *device*. Następnie przesyłany jest bajt z przedziału wartości 0xB0...0xBF informujący aplikację o bieżącym stanie przekaźników. Ostatnim krokiem jest sprawdzenie, czy w buforze portu szeregowego znajdują się jakieś nieodczytane bajty. Jeżeli taka sytuacja ma miejsce, to następuje pobranie bajtu najdłużej zalegającego w buforze (w ten sposób realizowana jest kolejka FIFO) i za pomocą instrukcji *switch* definiowane jest jego znaczenie na podstawie czterech najstarszych bitów. Polecenia operujące na nazwach przekaźników (odczyt i zapis nazwy) powodują wykonanie operacji na pamięci EEPROM z użyciem klasy *eeprom* (jej instancją jest obiekt *memory*). Warto zauważyć, że początkowy adres pamięci jest wyznaczany na podstawie czterech młodszych bitów polecenia wymuszającego zapis nowej nazwy. Kopiowanie przesyłanych bajtów kończy się po 20. znaku lub wystąpieniu znaku NULL.

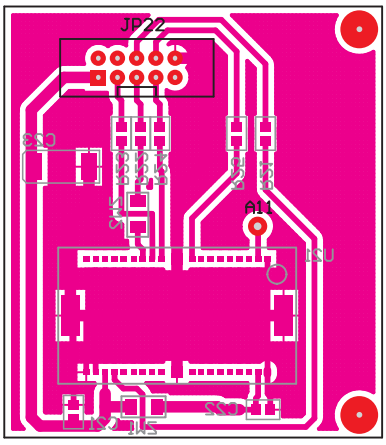
Montaż i uruchomienie

Urządzenie zostało zamontowane na płytkach drukowanych pokazanych na **rysunkach 15** i **16**. Jako obudowę zastosowałem Z-59. układ został zamocowany w obudowie za pomocą śrób, które można wkręcić w tulejki znajdujące się na dnie obudowy. Montaż przycisków nie sprawił większych kłopotów, bo zakupione zostały takie, które mają gwintowany korpus, więc po wywierceniu otworów można je bez większego problemu przykręcić. Podobnie było z diodami LED, które zostały umieszczone wewnątrz specjalnych, plastikowych oprawek. Jedynym problemem może być równomierne rozmieszczenie tych elementów. Na warstwie mechanicznej płytki drukowanej zostały dodane opisy, które można wydrukować, przyłożyć do obudowy i na ich podstawie nawiercić otwory pod diody i przyciski.



Rys. 15 Skala 50%

Rys. 16 Skala 100%



Poszczególne moduły i elementy zostały dołączone za pomocą złączy stykowych, co ułatwi rozłaczanie i ew. serwis. Obecność złączy szpilkowych dla przycisków sprawiła, że trudniej było zamontować płytkę modułu BTM-222. Postanowiłem wlutować złącza szpilkowe od strony druku do płytki z BTM-222, a do płytki sterownika wlutowałem gniazdo goldpinów. Przejściówka została dodatkowo przytwierdzona śrubą do płytki sterownika i dociśnięta do niej tulejką dystansową.

Jakub Borzdyński
jakub.borzdynski@elportal.pl

Wykaz elementów

Płytka sterownika

R31	10kΩ 2012 (0805)
R32,R45-R48	1kΩ 2012 (0805)
R41-R44	4,7kΩ 2012 (0805)
C11,C13,C15	100µF
C12,C14,C16,C33	100nF 2012 (0805)
C31,C32	22pF 2012 (0805)
D11-D14	1N4007
D31,D45-D48	LED
D41-D44	LL4148
T41-T44	BC847
U11	LM7805
U12	LF33CDT
U31	ATtiny2313
JP31	goldpin x6
JP32	goldpin 5x2
Q31	3,6864MHz
REL41-REL44	HF3FF lub JQC-3FF
Z11	ARK2
Z41-Z44	ARK3

Moduł Bluetooth

R21,R22-R25	1kΩ 2012 (0805)
C23	10µF tantalowy SMD
C21,C22	100nF 2012 (0805)
U21	BTM-222 bluetooth
JP22	goldpin 5x2

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-2938.

Listing 1

```
int main(){
 bool stat = 0 ;
 dev.statusLed(1) ;
 //detekcja przytrzymania przycisku
 while(1){
  //obsluga klawiatury
  char result = dev.key() ;
  if(result&DEVICE_KEY_1){dev.relayInvert(DEVICE_RELAY_1) ;}
  if(result&DEVICE_KEY_2){dev.relayInvert(DEVICE_RELAY_2) ;}
  if(result&DEVICE_KEY_3){dev.relayInvert(DEVICE_RELAY_3) ;}
  if(result&DEVICE_KEY_4){dev.relayInvert(DEVICE_RELAY_4) ;}
   //nacisnieto jakis klawisz  ?
    if(result!=0){
     btm.writeChar(dev.getRelayState()|0xB0);¬
     //przeslij zmiane do urzadzenia
   }
   //sprawdzenie, czy nadszedl nowy znak
   if(btm.isNewData()){
    char data = btm.readChar() ; //odebrany znak
    unsigned char index = (data&0x0F)*22 ; //adres pamieci
    //"uklad" decyzyjny
    switch(data&0xF0){
     case 0xA0: //ustawienie stanu przekaznikow
       dev.setRelayState(data&0x0F)  ;
       break ;
     case 0xB0: //prosba o wyslanie stanu przekaznikow
       btm.writeChar(dev.getRelayState()|0xB0);
       break ;
     case 0xC0: //prosba o przeslanie nazwy urzadzenia
       //odeslij nazwe urzadzenia
       btm.writeChar(data) ; //kod polecenia
       //przeslanie nazwy
       for(unsigned char i=0 ; i<20 ; i++){
         char znak = memory.read(index+i) ;
         btm.writeChar(znak) ; //kod polecenia
         if(znak==0){break;}
       }
       break ;
     case 0xD0:
    //oczekuj na zakonczenie transmisji
       _delay_ms(20) ;
       //pobierz bajty i zapisz w pamieci
       while(btm.isNewData()){
         char temp = btm.readChar() ;
         memory.write(index++, temp)  ;
         if(temp==0){break;}
       }
       break ;
     case 0xF0:
       stat = !stat ;
       dev.statusLed(stat) ;
       break ;
    }
   }
  }
}
```