Alarm samochodowy z telefonem GSM SIEMENS C/S/M35

AVT–936

Przedstawiamy sposób podłączenia telefonu GSM do systemu z mikrokontrolerem. Napisany dla mikrokontrolera program, to rodzaj alarmu z powiadomieniem, który może zostać zainstalowany na przykład w samochodzie. Jednak opisywaną aplikację należy traktować bardziej jako pewną sugestię, co do wykonania części sprzętowej i interfejsu łączącego mikrokontroler z telefonem GSM, aniżeli gotowe do wykorzystania urządzenie.

***Rekomendacje**: wykorzystanie w dzisiejszych czasach telefonów i modemów GSM jest bardzo powszechne. Artykuł polecamy wszystkim, którzy planują rozszerzenie funkcjonalności budowanego przez siebie sprzętu o łączność bezprzewodową. Przykładowy projekt alarmu z powiadomieniem jest znakomitym punktem startowym.*

PODSTAWOWE PARAMETRY

- Płytka o wymiarach 114 x 72 mm
- Zasilanie 12 VDC
- Telefon nie jest częścią projektu
- Sterowanie telefonu komendami AT
- Wysyłanie powiadomienia (SMS) o włączeniu alarmu
- Możliwość odbioru SMS z komendami (opcja)
- Zdalne wyłączanie alarmu
- Dwa przekaźniki do dowolnego wykorzystania

Funkcje realizowane przez mikrokontroler w prezentowanej aplikacji są następujące:

– wysyłanie przez SMS powiadomienia o załączeniu,
– wyłączenie alarmu przez identyfikację osoby dzwoniącej,
– możliwość odbioru SMS z komendami (nie zaimplementowana, jednak obecna w bibliotekach programowych).

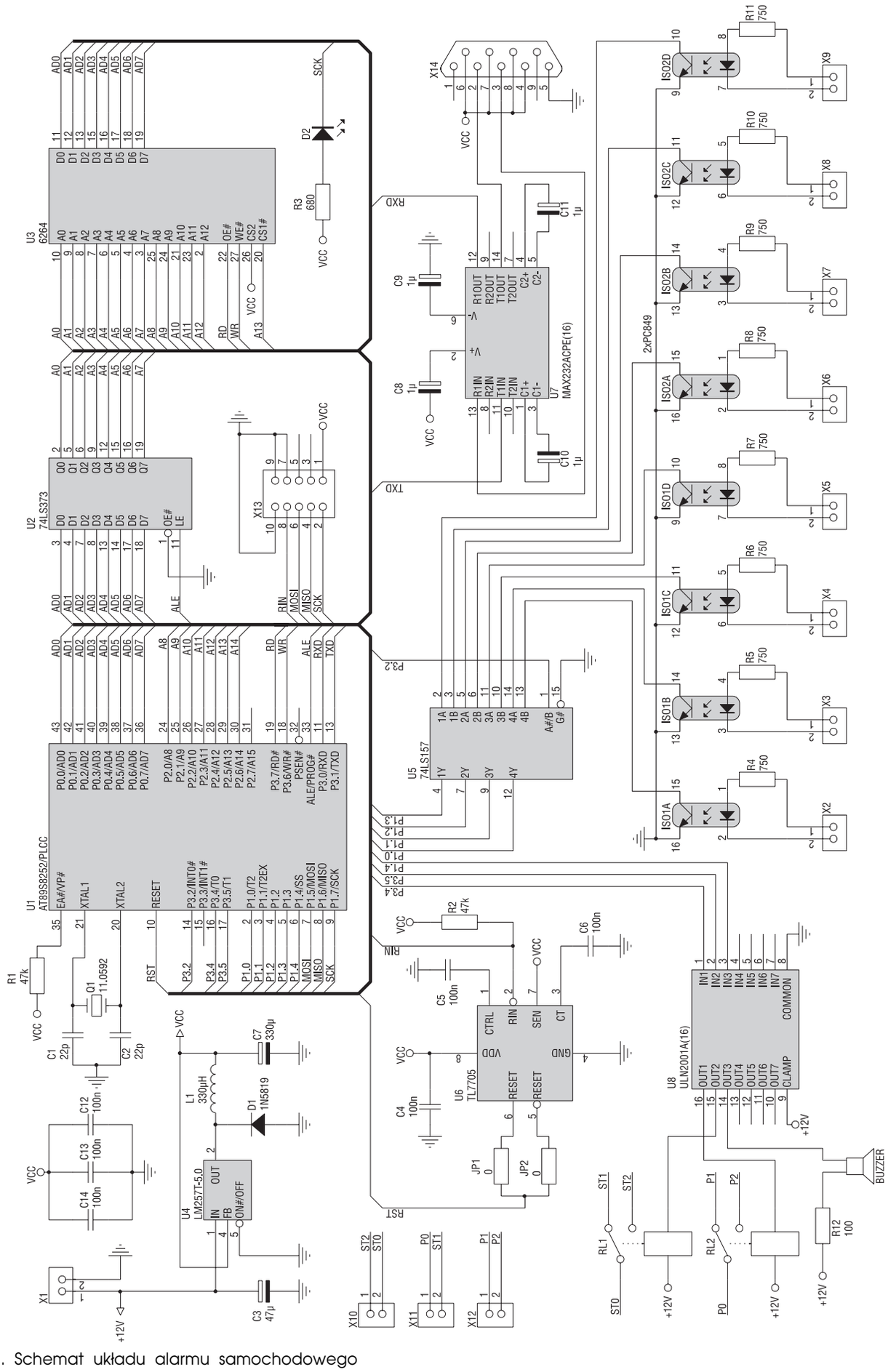
Program sterujący napisany jest w języku C dla mikrokontrolera 8051. Opisywane wyżej funkcje zaimplementowano w bibliotekach i są one gotowe do wykorzystania. Wymagają jednak pewnej inwencji własnej Czytelnika i ewentualnej parametryzacji. Jeszcze raz powtórzę, że aplikację należy traktować jako przykład połączenia mikrokontrolera z telefonem GSM przy pomocy ogólnie dostępnych podzespołów.

Opis układu

Na **rys. 1** przedstawiono schemat układu sterownika alarmu. Sterownik modelowy został zbudowany przy wykorzystaniu mikrokontrolera AT89S8252–24JI (w obudowie PLCC) taktowanego kwarcem o częstotliwości 11,0592 MHz. Mikrokontroler ma dołączoną zewnętrzną pamięć RAM o pojemności 8 kB, która pracuje w obszarze XDATA, wykorzystując adresy od 0x0000 do 0x1FFF. Aplikacja wykorzystuje ją jako bufor od odbioru komunikatów SMS (zmienna UDBuf o rozmiarze 1 kB), pozostawiając sporą jej część na różne inne zmienne użytkownika.

Mikrokontroler AT89S8252 jest zgodny ze standardowym INTEL 8052. Szyna danych ma długość 8 bitów, a szyna adresowa 16 bitów. Ze zgodności tej wynika również fakt, że młodszy bajt adresu urządzenia zewnętrznego oraz 8–bitowe słowo danych są multipleksowane na wyprowadzeniach portu P0. Jako pierwszy wyprowadzany jest adres, później zmienia się stan linii ALE z wysokiego na niski, a następnie odczytywane lub zapisywane jest słowo danych. Sekwencja ta powtarza się dla każdej operacji odczytu lub zapisu komórki pamięci zewnętrznej. W związku z tym adres musi zostać zapamiętany w zewnętrznym przerzutniku. Rolę tę pełni układ U2,z tj. 8–krotny przerzutnik typu „D" 74LS373 (lub 74HCT373).

Na płytce brak jest dekodera adresów rozróżniającego wybrane komórki pamięci. Zakłada się, że pamięć U3 (statyczny RAM typu 6264) pracuje zawsze, gdy linia adresowa A13 ma stan niski. Kierunkiem przepływu danych (zapis/odczyt) sterują sygnały RD (read) i WR (write) wyprowadzane przez



Rys. 1. Schemat układu alarmu samochodowego

mikrokontroler, a zależne od realizowanych przezeń rozkazów zapisu, czy odczytu danych.

Funkcję interfejsu pomiędzy mikrokontrolerem, a telefonem GSM pełni układ U7 typu MAX232 (lub odpowiednik). Sygnały z wyprowadzeń interfejsu UART mikrokontrolera, poprzez układ MAX, doprowadzone są do złącza X14. Jest to złącze 9–wyprowadzeniowe złącze męskie typu DSUB, identyczne z zalecanym dla urządzeń DTE przez standard EIA232 (to jest takie, jak w komputerze PC). Do komunikacji wykorzystywane są tylko dwa sygnały – RXD i TXD. Inne doprowadzenie nie są podłączone, bądź służą do doprowadzenia zasilania. Tu kilka słów wyjaśnienia.

Oryginalnie, do połączenia płytki z telefonem GSM, używane były kable do transmisji danych, przeznaczone dla telefonu ERICSSON T10 lub SIEMENS C/S/M35, w zależności od dołączonego modelu aparatu. Cechą charakterystyczną tych kabli jest fakt, że czerpią one zasilanie z nieużywanych wyprowadzeń interfejsu RS232. W związku z tym, wymagane było dołączenie napięcia zasilającego do odpowiednich wyprowadzeń złącza. Napięcie zasilające płytkę, to jest +5 V, okazało się do tego celu zupełnie wystarczające. Oczywiście nie jest to zgodne ze standardem wyprowadzeń sygnałów i w przypadku użycia zewnętrznego modemu GSM podłączanego przy pomocy kabla do transmisji szeregowej, należy te doprowadzenia napięcia zasilania odciąć.

Rozwiązanie takie ma swoje zalety, mimo swojej dosyć wysokiej ceny: za układ MAX trzeba zapłacić kilka złotych, a za kabel kilkanaście. Zyskuje się jednak w ten sposób możliwość testowania i uruchomienia płytki przy pomocy komputera PC i programu terminalowego (komendy wysyłane są jako łańcuchy znaków i mogą być obserwowane na ekranie PC, natomiast odpowiedzi telefonu można wpisywać ręcznie), a dodatkowo każdy firmowy kabel jest wyposażony w odpowiednie złącze do telefonu. Odpada więc problem z jego zakupem i unika się sytuacji uszkodzenia aparatu przez różne „samoróbki".

Niestety nie rozwiązuje to problemu zasilania telefonu. Oczywiście każdy aparat telefoniczny GSM wyposażony jest w lepszy lub gorszy akumulator. Wymaga on jednak okresowego ładowania, którego częstość zależy od jakości i stopnia zużycia akumulatora. Akumulatora nie można jednak ładować prądem „wprost", co oznacza, że ładowarka powinna być „inteligentna" i dostosowywać ilość dostarczanej energii do stanu naładowania akumulatora. I tu można napotkać pewien problem: niektóre telefony mają tę „inteligencję" wbudowaną, inne wymagają wykonania odpowiedniego układu ładowania. Używany przeze mnie w testach telefon SIEMENS C35 został przez producenta wyposażony w układ sterowania ładowaniem akumulatora i jest do zastosowania w systemach embedded wręcz idealny. Nie dosyć, że do poprawnej pracy wystarcza doprowadzenie napięcia +5 V (maksymalnie do 8 V) podanego pomiędzy doprowadzenia 1 oraz 2 i 3 złącza (opis złącza telefonu SIEMENS C/S/M35 można znaleźć w rozdziale 1.4), to jeszcze na dodatek sterowanie telefonem jest proste i bezproblemowe. Praktycznie niemal wszystkie funkcje telefonu mogą być załączane z użyciem komend AT i prostej transmisji szeregowej przez interfejs UART. Niestety w celu doprowadzenia zasilania, wtyczka oryginalnego kabla musi być rozebrana, a napięcie doprowadzone indywidualnie. Prezentowana aplikacja nie opisuje, w jaki sposób ma to być zrobione. Można to jednak wydedukować w prosty sposób z lektury wcześniejszych rozdziałów.

Sygnały z czujników zewnętrznych są odizolowane przy pomocy transoptorów. Napięcie doprowadzane jest pomiędzy anodę i katodę diody LED sterując załączaniem odpowiedniego tranzystora. Napięcie z kolektora doprowadzane jest na wejście układu multipleksera typu 74LS157 (lub 74HCT157) wybierającego jedno z dwóch wejść, w zależności od stanu doprowadzonego na wejście sterujące A/B (nóżka 1 układu 74LS157).

Użycie jednego z dwóch układów: multipleksera lub bufora szyny danych wybieranego przez odpowiedni adres obecny na szynie adresowej było konieczne ze względu na ograniczenie liczby wyprowadzeń. Jako układ wykonawczy pracuje ULN2003A sterujący pracą przekaźników oraz zasilaniem brzęczka.

Ostatnim elementem wymagającym omówienia jest układ produkcji Texas Instruments, TL7705. Należy on do grupy układów, tak zwanych nadzorców napięcia zasilającego (*voltage supervisors*) i w tej aplikacji służy do wypracowania sygnału zerującego mikrokontroler *reset*. Układ pracuje, o ile doprowadzenie RIN znajduje się w stanie wysokim oraz napięcie doprowadzone do wejścia pomiarowego SEN ma odpowiednią wartość. W związku z tym, że wejście pomiarowe jest dołączone wprost do napięcia zasilającego mikrokontroler, pełni ono funkcję wyłącznie pomiarową. Inaczej jest w przypadku wejścia RIN. Jest ono doprowadzone do złącza interfejsu SPI umożliwiając programowanie mikrokontrolera w układzie. Obecny na tym doprowadzeniu stan niski umożliwia przejście w tryb zapisu pamięci wewnętrznej Flash (wymagany jest dla tego trybu aktywny sygnał zerowania mikrokontrolera *reset*).

Aktywnym sygnałem wyjściowym TL7705 może być stan niski (wyprowadzenie 5) lub wysoki (wyprowadzenie 6). Sygnały wyjściowe doprowadzone są do zworek JP1 i JP2. **Uwaga: może być wlutowana tylko jedna z nich.** Umożliwia to wykorzystanie w płytce mikrokontrolera AT89S8252 (należy wlutować zworkę JP1) lub zgodnego z nim pod względem wyprowadzeń mikrokontrolera AVR, na przykład AT90S8515 wymagającego, aby stanem aktywnym sygnału *reset* był stan niski.

Płytka jest przystosowana do zasilania napięciem +12 V pochodzącym z zewnętrznego źródła napięcia zasilającego. Wartość napięcia to nie tyle wymóg przetwornicy zasilającej (ta pracuje poprawnie w zakresie od +7 do +35 V) ile użytych przekaźników. Zasila ono wprost cewki przekaźników oraz prosty układ przetwornicy *step–down* zbudowanej z użyciem układu typu LM2575–T5.0 produkcji firmy National Semiconductors. Mimo, iż na płytce przewidziano miejsce na radiator, w normalnych warunkach pracy nie musi on być instalowany.

Montaż i użytkowanie

Układ został zmontowany na płytce dwustronnej z metalizacją otworów. Montaż jest mieszany, to jest do konstrukcji urządzenia wykorzystano zarówno elementy do montażu przewlekanego (dioda LED, podstawka mikrokontrolera, rezonator kwarcowy, układy scalone i transoptory, dławik i dioda, terminatory itp.), jak i do montażu powierzchniowego (niemal wszystkie elementy R i C z wyjątkiem kon-

densatorów elektrolitycznych). Nie ma szczególnych zaleceń kolejności montażu – ta może być dowolna. Moim zdaniem osoby nie mające doświadczenia z aplikacjami przetwornic napięcia zasilania, powinny zacząć od wlutowania elementów przetwornicy zasilającej, doprowadzenia napięcia +12 V oraz pomiaru napięcia wyjściowego. Uchroni to układ mikrokontrolera przed uszkodzeniem w przypadku, gdy popełnione zostaną błędy montażowe. Napięcie wyjściowe najlepiej jest mierzyć przy pomocy oscyloskopu, który umożliwi również obserwację jego kształtu. Pozwoli to uniknąć problemów nie tylko z zasilaniem układów TTL, czy mikrokontrolera, ale również z przypadkowymi załączeniami sygnału *reset* na wyjściu TL7705.

Jako aktywny uważany jest stan wysoki napięcia doprowadzonego do złącza X2..X9. Musi być ono na tyle duże, aby zaświecić diodę LED znajdującą się wewnątrz struktury transoptora. W związku z tym rezystory R4...R11 należy dobrać w zależności o przewidywanej wartości napięcia pochodzącego z czujników. Wykorzystując transoptory PC849 lub ich odpowiedniki LT849 (o nieco wyższym prądzie załączenia i gorszym współczynniku przenoszenia) wartość rezystancji R4...R11 można obliczyć jako:

W przypadku doprowadzenia

$$R = \frac{Uwe - 1{,}5[V]}{0{,}012[A]}[\Omega]$$

zewnętrznego napięcia o wartości 12 V, zalecane jest użycie rezystorów o wartości 1...1,2 kΩ. Oczywiście rezystory mogą być różne dla różnych wejść (uwaga: złącze X2 przeznaczone jest do dołączenia przycisku załączającego alarm!). Podobnie wymagane jest dobranie wartości rezystora R12 w zależności od posiadanego buzzer'a:

Program sterujący został skom-

$$R12 = \frac{12[V] - Uzas.buzz.}{Izas.buzz.}[\Omega]$$

pilowany wraz z kodem PIN zapamiętanym jako stała. Kod PIN jest wprowadzany do telefonu tuż po załączeniu zasilania i jest to pierwsze przesyłane przez program polecenie do aparatu GSM. Oczywiście numer ten może być inny dla każdej karty PIN i w związku z tym możliwe są dwa rozwiązania: albo zmiana numeru PIN na zgodny z aplikacją sterującą (w tym przypadku jest to liczba 9861), albo skompilowanie programu z inną stałą. Jej deklarację zawiera plik nagłówkowy „*VARIABLES35.H*" i tam może być ona zmieniona. Jest to o tyle ważne, że po wprowadzeniu kodu PIN aplikacja sterująca bada odpowiedź telefonu i jeśli zwracany komunikat jest różny od „OK", to program zasygnalizuje błąd komunikacji z telefonem i praktycznie zakończy pracę. Będzie o tym mowa dalej, przy omawianiu szczegółów związanych z cechami aplikacji sterującej.

Sygnały z czujników alarmowych muszą być doprowadzone do złącz X2...X9 z tym, że złącze X2 dodatkowo służy do załączania trybu czuwania alarmu. Podanie na wejście X2 napięcia (zgodnie z zasadami opisywanymi dla czujników) przez czas około 1 sekundy, powoduje zmianę trybu z nieaktywnego na czuwania. Po tej zmianie ponowne podanie sygnału (już znacznie krótszego, około 200 ms) na X2 spowoduje załączenie alarmu. Wejście to funkcjonuje identycznie, jak na X3..X9.

Płytka jest wyposażona w dwa, niezależnie załączane przekaźniki. Mogą być one użyte w zupełnie dowolny sposób, zgodnie z inwencją użytkownika. Można na przykład wyposażyć aplikację w prosty interpreter poleceń, który będzie sterował przekaźnikami. W tym momencie, po załączeniu alarmu, RL1 pracuje w sposób przerywany, a RL2, załączany jest na stałe. Pierwszy może być na przykład użyty do sygnalizacji wizualnej przy pomocy mrugających świateł kierunkowskazów, drugi do załączenia sygnalizacji dźwiękowej, przerwania obwodu zapłonu itp.

Alarm jest wyłączany dzięki identyfikacji osoby dzwoniącej. Numer telefonu próbującego nawiązać połączenie jest rozpoznawany i jeśli jest to telefon zapisany w parametrach programu jako zarządzający, alarm jest wyłączany. Połączenie przychodzące jest rozłączane po sprawdzeniu numeru telefonu, toteż jego przeprowadzenie nie pociąga za sobą praktycznie żadnych kosztów.

Alarm nie posiada dołączonego wyświetlacza, nie korzysta również z wyświetlacza telefonu. Komunikacja z użytkownikiem przeprowadzana jest za pomocą diody LED oraz komunikatów SMS. Bufor komunikatów (pamięć) jest na tyle duży, że można alarm wyposażyć w interpreter poleceń przesyłanych za pomocą SMS–ów. Układ jest w stanie bez żadnych problemów odbierać je, jednak praktyczną realizację tego zagadnienia pozostawiono inwencji użytkownika.

Aplikacja sterująca

Można zaryzykować twierdzenie, że najważniejszą funkcją aplikacji sterującej jest ta inicjująca pracę i nastawy telefonu komórkowego. Nosi ona nazwę *initgsm()* i została predefiniowana w bibliotece „*SIEMENS35.C*". Przyjrzyjmy się realizowanym przez nią nastawom. Na **list. 1** umieszczono tę funkcję wraz z fragmentem pliku nagłówkowego *VARIABLES35.H* zawierającego definicje zmiennych i stałych przeznaczonych do wykorzystania przez różne funkcje obsługi komunikacji z aparatem GSM.

Wszystkie komendy AT przesyłane są do telefonu przez interfejs UART za pomocą standardowej funkcji *printf()* predefiniowanej przez producenta kompilatora w bibliotece *stdio.h*, w którą wyposażony

List. 1. Funkcja initgsm()

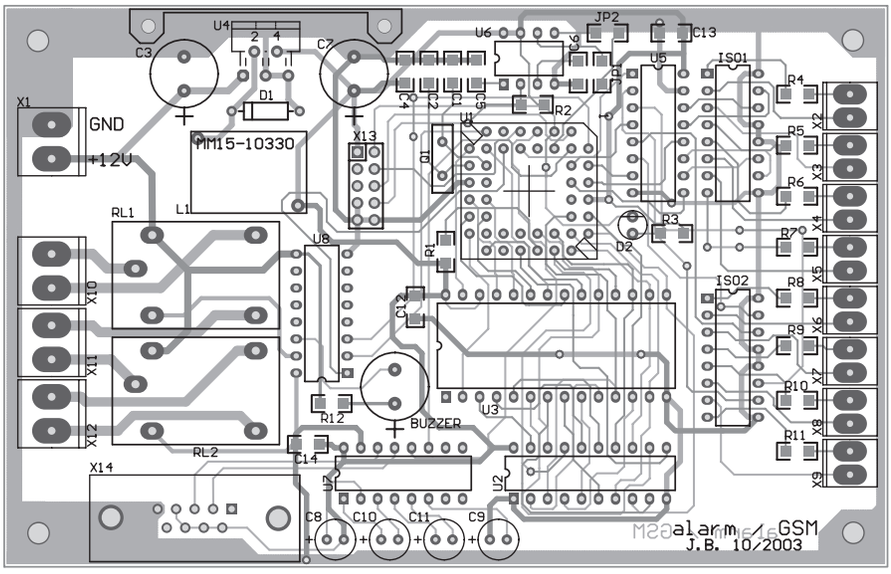
```
code char CPIN[] = „AT+CPIN=9861";  //wprowadzenie numeru PIN
code char ECHOOFF[] = „ATE0";    //wyłączenie echa komendy AT

code char SETNOTICE[] = „AT+CNMI=1,1,0,2"; //ustawienie sposobu powiadamiania
code char SETCLIP[] = „AT+CLIP=1";  //ustawienie sposobu powiadamiania funkcji CLIP

//wysłanie numeru PIN do telefonu, ustawienie sposobu powiadamiania
bit initgsm()
{
   idata char temp[8];

   printf("%s\n",CPIN);    //wprowadzenie numeru pin
   gets(&temp);         //jako pierwsze wysyłane jest 0D+0A
   gets(&temp);         //jeśli PIN był poprawny, to komunikat „OK"+0D+0A
               //jeśli komunikat <>OK, funkcja zwraca 0 i kończy pracę
   if (strncmp(temp, „OK", 2) != 0) return(0);

   printf("%s\n",ECHOOFF); //wyłączenie echa rozkazu
   gets(&temp);
   gets(&temp);
   printf("%s\n",SETNOTICE);  //ustawienie trybu powiadomienia o sms
   gets(&temp);
   gets(&temp);
   printf("%s\n",SETCLIP); //ustawienie trybu powiadomienia funkcji CLIP
   gets(&temp);
   gets(&temp);
   return(1);
}
```



Rys. 2. Schemat montażowy płytki drukowanej

jest praktycznie każdy kompilator języka C. Funkcja przesyła łańcuch znaków (%s) kończąc go znakiem nowej linii (\n).

Jako pierwszy, przy pomocy komendy AT+CPIN=<kod pin> jest przesyłany kod PIN do telefonu. Wprowadzenie poprawnego numeru PIN jest warunkiem załączenia telefonu i realizacji przez niego jakichkolwiek innych komend wykorzystywanych przez układ alarmu. Poprawne wprowadzenie kodu PIN kończy komunikat OK. Po jego przesłaniu telefon loguje się do sieci. Aplikacja nie sprawdza jakości sygnału.

Jako następne przesyłane są następujące polecenia:

– Wyłączenie echa komendy: ATE0.
– Ustawienie sposobu powiadamiania o odebranym przez aparat komunikacie SMS: AT+CNMI=1,1,0,2 (przypomnijmy: na skutek realizacji tego polecenia powiadomienie będzie miało postać +CMTI: <pamięć>, <numer lokalizacji> na przykład +CMTI: ”SM”, 3)
– ustawienie sposobu funkcjonowania powiadomienia o przychodzącym połączeniu: AT+CLIP=1 (połączenie przychodzące skutkuje pojawieniem się komunikatów: RING oraz +CLIP: <numer telefonu>, <numer funkcji> np. RING +CLIP: 0603630461, 129.

Po wykonaniu nastaw sposobu komunikacji z telefonem GSM, rola aplikacji sterującej sprowadza się do sprawdzania i dekodowania rozmaitych komunikatów tekstowych wysyłanych przez telefon oraz testowania stanu czujników. Niestety – operacje na łańcuchach tekstowych są stosunkowo trudne do wykonania, jeśli porównać je do operacji na liczbach. Sprawiają one trudność nie tylko programiście, ale także mikrokontrolerowi pochłaniając gro czasu CPU. Operacje te są również głównym powodem, dla którego dowolny program realizujący interfejs pomiędzy użytkownikiem, a aparatem GSM jest bardzo obszerny i trudny do analizy, a jego implementacja pochłania mnóstwo czasu zużytego na uruchomienie i usuwanie błędów, mimo iż nie realizuje zbyt skomplikowanych funkcji.

Początek programu głównego zawiera inicjalizację stanów portów wyjściowych, zerowanie rejestru licznika układu Watchdog oraz pobranie statusu alarmu z wewnętrznej pamięci EEPROM mikrokontrolera. W przypadku, gdy komórka pamięci zawiera same wartości logiczne „1”, to jest bajt o wartości 0xFF, program interpretując tę sytuację „uważa”, że procesor nie był jeszcze używany (stan po zapisie pamięci Flash). Do komórki pamięci EEPROM jest zapisywana wartość oznaczająca stan wyłączenia alarmu. Później wykonywana jest funkcja inicjalizacji telefonu, opisywana wyżej i umieszczona na list. 1. W przypadku, gdy inicjalizacja przebiegła pomyślnie (funkcja *initgsm()* zwróciła wartość *true*), program przechodzi do wykonywania pętli nieskończonej, sterującej podejmowaniem odpowiedniej akcji, w zależności od stanu zmiennej czujnika oraz statusu alarmu.

Program jest prosty, jego analiza nie powinna nastręczać trudności. Omówienia być może wymaga jeden drobny szczegół.

Po załączeniu alarmu, konieczne jest wykonywanie akcji związanej z obsługą załączenia przekaźników RL1 i RL2. Również w stanie czuwania, wykonywana jest funkcja odczytu stanu czujników alarmu. W związku z ich realizacją wykonywane są kolejne nieskończone pętle, które – zgodnie z założeniem, że identyfikacja numeru aparatu dzwoniącego wyłącza alarm – muszą sprawdzać komunikaty docierające z dołączonego telefonu GSM. Niestety nie nadają się do tego celu standardowe funkcje scanf(), czy gets(), ponieważ z założenia oczekują one na odbiór całego łańcucha znaków zakończonego na dodatek przy pomocy sekwencji CR–LF (0x0A–0x0D). Użycie standardowych funkcji przerwałoby, więc funkcjonowanie pętli, a tym samym sygnalizację stanu załączenia. W związku z tym, do obsługi odbioru komunikatu używane jest przerwanie UART. Jego obsługa jest załączana na czas obsługi funkcji załączenia sygnalizacji alarmon(). Znak docierający do UART powoduje ustawienie flagi RI, a tym samym przejście do obsługi przerwania.

Format wysyłanego przez aparat GSM komunikatu CLIP jest następujący (znaki „białe” tj. nie wyświetlane przez program monitora, zastąpiono ich kodami szesnastkowymi!):

0x0D 0x0A
RING 0x0D 0x0A
0x0D 0x0A
+CLIP: 0603630461 0x0D 0x0A

Łatwo zauważyć, że znaki nowej linii oraz znaki powrotu karetki wysyłane są 4–krotnie. Właściwość tę wykorzystałem przy odbiorze komunikatu. Zmienna *unsigned char OxOA_CNT* pełni rolę licznika odebranych znaków końca linii. Jeśli funkcja odbioru odbierze 4 takie znaki, komunikat jest interpretowany, tzn. funkcja porównuje numer telefonu odebranego z tym zapamiętanym w zmiennej SPHONE i w przypadku zgodności alarm jest wyłączany lub jest zerowany indeks bufora odbioru i program kontynuuje sygnalizację oczekując na następny komunikat. Podobnie jest w trybie czuwania.

Opis funkcji biblioteki SIEMENS35.C

W pliku nagłówkowym o nazwie *SIEMENS35.H* znaleźć można użyteczne funkcje obsługi telefonu tSIEMENS C/S/M35. Przypuszczalnie funkcje te działać będą również z innymi telefonami, jako że komendy AT zgodne są z normami GSM. Mogą jednak wystąpić drobne różnice w ich implementacji w zależności od producenta telefonu.

extern bit initgsm();

Inicjalizacja aparatu GSM, wprowadzenie nastaw dotyczących sposobu funkcjonowania powiadomień o komunikatach SMS oraz przychodzących połączeniach i identyfikacji CLIP.

extern void sendsms(generic char *tekst);

Funkcja wysyłająca SMS. Tutaj zaimplementowana w uproszczonej postaci zakładającej, że program będzie zawierać stałe parametry SMS (łącznie z numerem odbiorcy SMS!), a zmieniać się będzie jedynie przesyłany komunikat. Oczywiście funkcję można zmienić i uprościć, w zależności od posiadanego modelu telefonu. W podanej tu postaci, funkcja działa poprawnie z wszystkimi aparatami obsługującymi tryb PDU w sieci PLUS GSM. Dostosowanie do innej sieci wymaga zmiany lub parametryzacji następujących stałych, które można odnaleźć w pliku *VARIABLES35.H*:

- code char LOSCA=0x07; długość numeru Centrum Usług (dla większości sieci nie będzie wymagać zmiany),
- code char TOSCA=0x91; typ numeracji Centrum Usług (omawiany w rozdziale 2.2 – tutaj numeracja międzynarodowa), podobnie, jak LOSCA nie będzie wymagać zmiany dla większości operatorów,
- code char SCA[]="8406010013F0"; zakodowany zgodnie z zasadami kodowania informacji w PDU numer Centrum Usług, tutaj podany dla sieci PLUS GSM (48601000310); wymaga zmiany w zależności od operatora, zalecane jest zachowanie formatu międzynarodowego,
- code char FO=0x11; pierwszy oktet dla wysyłanego komunikatu SMS, można pozostawić bez zmian,
- code char MR=0x00; numer odniesienia dla komunikatu SMS; dla komunikatów jednoczęściowych nie wymaga zmiany i nie zależy od operatora,
- code char LODA=0x0B; liczba cyfr numeru telefonu ODBIORCA; zalecane jest pozostawienie i zachowanie numeracji międzynarodowej dla następnego parametru,
- code char TODA=0x91; numeracja międzynarodowa dla numeru telefonu ODBIORCA
- code char DA[]="8406630364F1"; numer telefonu odbiorcy komunikatu SMS, tu zadeklarowany jako stały numer w sieci PLUS GSM (48603630461); zalecane jest zachowanie formatu numeracji międzynarodowej,
- code char PID=0x00; identyfikator protokołu komunikacyjnego dla wiadomości SMS; nie zależy od operatora i będzie taki sam dla standardowych SMS wysyłanych przy pomocy różnych sieci,
- code char DCS=0x00; schemat kodowania wiadomości – zalecane pozostawienie, jednak nie można używać znaków narodowych, tylko standardowe ASCII,
- code char SCTS=0x8F; okres ważności dla komunikatu SMS, tutaj 12 godzin,
- code char SPHONE[]= "0603630461"; numer telefonu uprawnionego do wyłączenia sygnalizacji alarmu lub stanu czuwania.

extern void receivesms(generic char *tekst);

Funkcja odbiera i dekoduje komunikat SMS. Działa poprawnie, jeśli aparat otrzymał wcześniej w wyniku inicjalizacji komendę *AT+CNMI=1,1,0,2* ustawiającą sposób powiadamiania o nowym komunikacie SMS. Jako parametr wywołania musi być podany wskaźnik do bufora, w którym znajdzie się zdekodowany komunikat. Bufor musi być na tyle obszerny, aby pomieścić całość komunikatu, nie tylko treść danych użytkownika. Zalecana wielkość to co najmniej 300 bajtów. Po odczycie komunikat jest usuwany z pamięci teelfonu.

extern bit gsmconnected();

Funkcja zwraca wartość logiczną true, jeśli aparat GSM jest podłączony. Sprawdzenie wykonywane jest w nieco prymitywny sposób. Załączany jest układ watchdog z nastawą 2048 ms, a do aparatu wysyłana jest komenda AT i pobierana jest odpowiedź. Jeśli aparat wysłał komunikat „OK”, funkcja odbiera go i zwraca wartość logiczną true i wyłącza watchdog. Jeśli aparat nie wysłał komunikatu zadziała watchdog przeprowadzając zerowanie mikrokontrolera. Jeśli natomiast został zwrócony inny komunikat niż „OK”, funkcja wyłącza watchdog i zwraca wartość logiczną false.

extern void disconnect();

Wywołanie realizacji komendy AT+CHUP powodującej przerwanie połączenia przez „odłożenie słuchawki”.

extern void delay(unsigned int k);

Funkcja realizuje opóźnienie o czasie trwania około k·1 ms. Działa poprawnie dla zegara o częstotliwości 11,0592 MHz oraz mikrokontrolera ze standardowym (1 cykl maszynowy = 12 cykli zegarowych) rdzeniem 8051 lub 8052.

extern void wdtreset();

Zerowanie układu timera układu watchdog.

Jacek Bogusz, EP
jacek.bogusz@ep.com.pl

WYKAZ ELEMENTÓW

Rezystory

R1, R2: 47 kΩ 1206
R3: 680 Ω 1206
R4...R11: 750 Ω 1206
R12: 100 Ω 1206

Kondensatory

C1, C2: 22 pF/50 V 1206
C3: 47 µF/50 V
C5, C6, C12...C14: 0,1 µF/50 V 1206
C7: 330 µF/25 V
C8...C11: 1 µF/50 V

Półprzewodniki

D1: 1N5819
D2: LED (zielona)
U1: AT89S8252 PLCC
U2: 74LS373 (74HCT373) DIP20
U3: 6264 DIP28
U4: LM2575-T5.0 TO-220
U5: 74LS157 (74HCT157) DIP16
U6: TL7705 DIP8
U7: MAX232 (ACPE) DIP16

Inne

JP1, JP2: 0 Ω 1206
L1: 330 µH
ISO1, ISO2: PC849 DIP16
Q1: 11,0592 MHz
RL1, RL2: przekaźnik GUARDIAN-A410/12 V
BUZZER: BUZZER 5 V
X1...X12: terminatory
X13: złącze IDC20
X14: DSUB-9