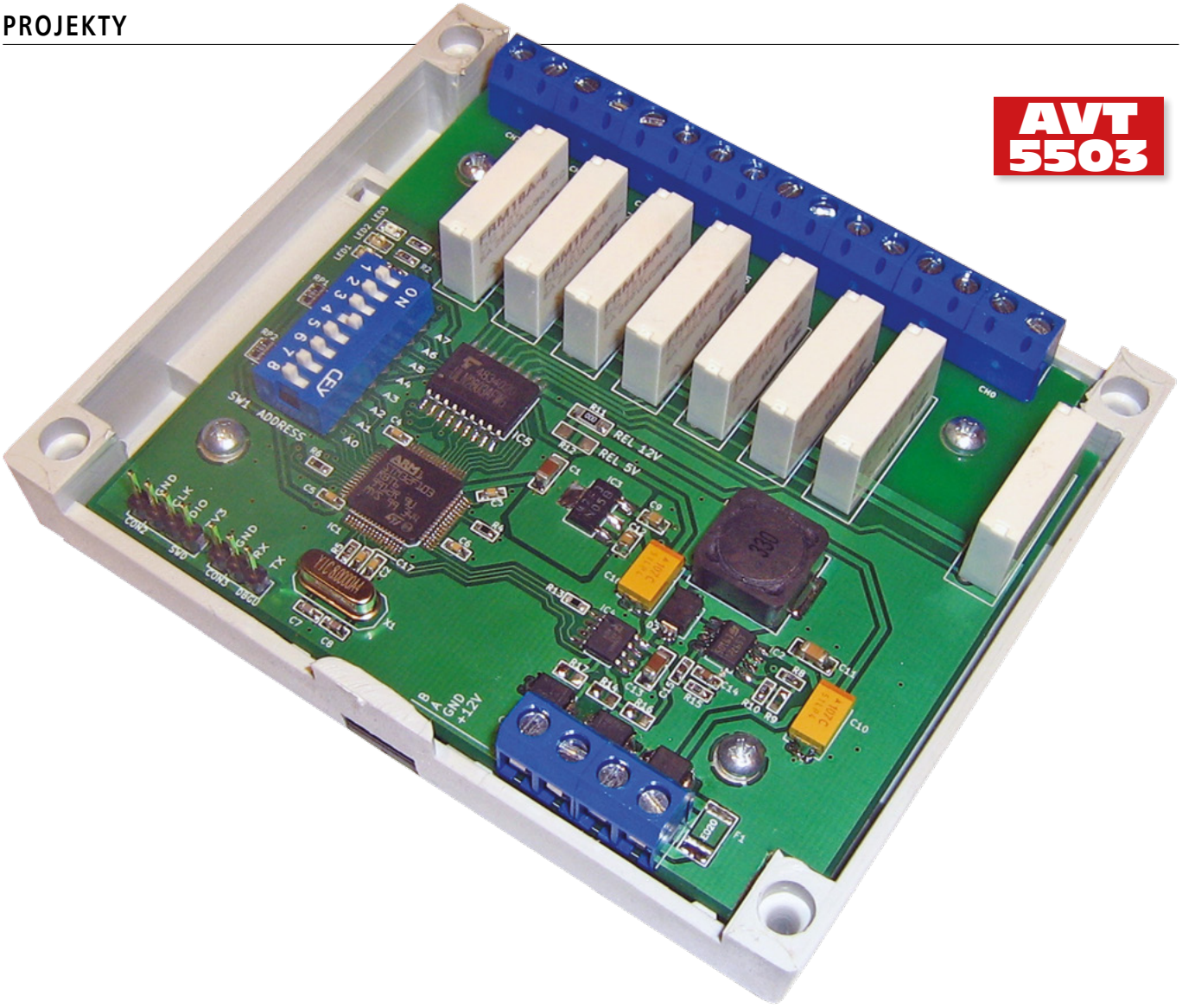


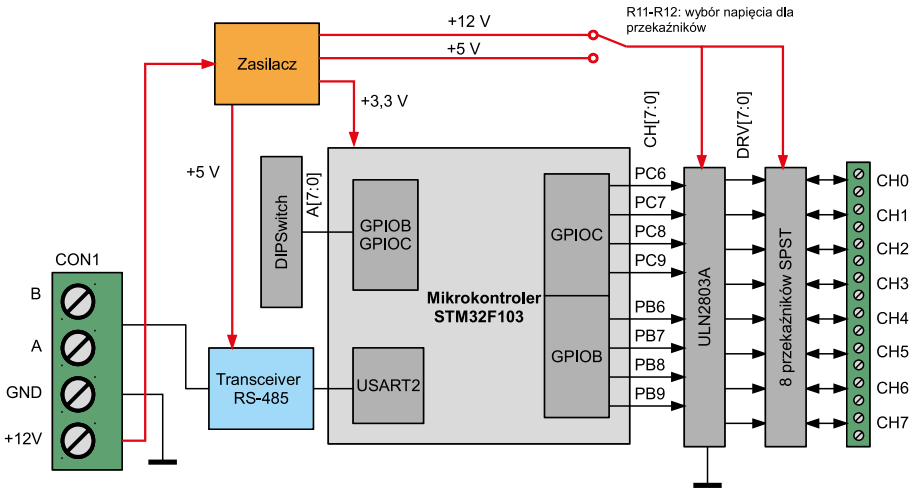
Relayz – 8-kanałowy moduł przekaźników z interfejsem SPPoB

W artykule opisano sterownik z wyjściami przekaźnikowymi, który może mieć szerokie zastosowanie w małej automatyce domowej. Daje on możliwość sterowania 8 wyjściami przekaźnikowymi przez łącze RS-485 za pomocą zaimplementowanego w nim protokołu komunikacyjnego SPPoB. Urządzenie może być łatwo zaadaptowane do różnych zastosowań wymagających nie tylko włączania lub wyłączania odbiorników, ale także uruchamiania ich i wyłączania na określony czas. Sprzętowa część sterownika ma typową budowę i korzysta z ogólnodostępnych elementów. Dodatkowymi atutami jest dość zwarta i łatwa w montażu konstrukcja.

Rekomendacje: *moduł jest przeznaczony do instalacji automatyki domowej.*

Automatyczne włączanie oświetlenia zewnętrznego, akwariowego lub w terrariach – do takich celów programowalne bądź zdalne włączniki często okazują się bardzo przydatne. Jeśli jeszcze rozszerzymy tę listę o możliwość sterowania elektrycznymi roletami bądź chwilowe odcinanie zasilania dla domowego routera, to motywacja do budowy opisanego tutaj zdalnego włącznika wielokanałowego może być jeszcze większa.

Opisany tutaj moduł przekaźników jest w niektórych aspektach podobny do sterownika taśm LED o nazwie „PWMLEDz" przedstawionego szczegółowo w EP 1/2015.



Rysunek 1. Schemat blokowy sterownika

Ponadto korzysta z opracowanego przez autora protokołu SPPoB, również opisanego na łamach EP, w numerach 10 i 11/2014. Z tych względów część redundantnych informacji w niniejszym artykule została skrócona. Głodnych wiedzy zachęcam do zapoznania się z odpowiednimi artykułami w wyżej wymienionych numerach EP.

Przy projektowaniu sterownika przyjęto kilka założeń: przede wszystkim uzyskanie relatywnie dużej ilości obsługiwanych kanałów oraz łatwy montaż mechaniczny w popularnej obudowie na szynę DIN. Te założenia udało się zrealizować uzyskując dość kompaktową konstrukcję, co ważne bez nadmiernego upakowania elementów wewnątrz obudowy oraz bez „piętrowego" montażu modułów. Gęste upakowanie sygnałów wyjściowych zmniejszyłoby bezpieczeństwo separacji galwanicznej między nimi, natomiast piętrowy montaż jest bardziej czasochłonny w wykonaniu i trudniejszy w ew. poprawkach niż pojedyncza płytka. Pewnym kosztem była tutaj konieczność zastosowania przekaźników o mniejszych gabarytach, a co za tym idzie mniejszej obciążalności (3…5 A) w porównaniu z typowymi konstrukcjami opartymi na przekaźnikach 8 lub 16-amperowych.

Podobnie jak w sterowniku „PWMLEDz", także i tutaj zastosowano wydajny mikrokontroler pozwalający Czytelnikom samodzielnie rozbudować funkcjonalność sterownika oraz zwiększyć autonomię jego działania. Rozbudowę i modyfikacje oprogramowania możemy przeprowadzić dysponując dostępnym za darmo kompilatorem GCC dla rdzeni ARM oraz tanim programatorem z interfejsem SWD.

Sprzęt

Schemat blokowy sterownika przedstawiono na **rysunku 1**, natomiast ideowy na **rysunku 2**. Od strony rozwiązań sprzętowych raczej nie znajdziemy tutaj niczego nietypowego, a sam schemat ma wiele elementów wspólnych ze sterownikiem „PWMLEDz".

Uwzględniono podstawowe zabezpieczenia elektryczne przy złączu interfejsu CON1 sterownika. Przed krótko- i długotrwałymi przepięciami oraz odwrotnym podłączeniem zasilania w pewnym stopniu chroni jednokierunkowa dioda D1 typu TVS („Transil") wraz z bezpiecznikiem polimerowym F1. Sygnały interfejsu RS-485 także mogą zostać opcjonalnie wyposażone w diody TVS: D3 i D4. W rozwiązaniu prototypowym do zabezpieczenia magistrali RS-485 zastosowano diody jednokierunkowe o napięciu znamionowym 12 V. Można tutaj także zastosować diody dwukierunkowe o niższym napięciu działania lub w ogóle z nich zrezygnować.

Zasilacz wykonano w postaci konstrukcji dwustopniowej, dostarczającej napięć 5 V i 3,3 V. Napięcie 5 V uzyskuje się z popularnej przetwornicy L5972D. Służy ono do zasilania układu transceivera RS-485 oraz – opcjonalnie – zespołu przekaźników. W urządzeniu można zamontować przekaźniki o znamionowym napięciu cewki wynoszącym 5 V lub 12 V. Zależnie od producenta, mogą one być oznaczane nazwami zaczynającymi się od NPA lub FRM i spotyka się je w wersjach o obciążalności styków 3 A lub 5 A.

Wyboru napięcia pracy dla całego zespołu przekaźników (VREL) dokonujemy wlutowując rezystor o wartości 0 Ω: albo w miejsce R11 dla przekaźników na 12 V, albo w miejsce R12, gdy zastosujemy przekaźniki na 5 V. Napięcie VREL jest podawane bezpośrednio na cewki przekaźników oraz do wewnętrznych diod zabezpieczających w popularnym układzie 8-kanałowego drivera ULN2803A (IC5).

Sterownik „Relayz" został zaprojektowany w oparciu o mikrokontroler STM32F103 (IC1). Do budowy prototypów zastosowano wersję STM32F103RBT6 tego układu, mogącą pracować z częstotliwością taktowania 72 MHz, wyposażoną w 128 kB pamięci Flash i 20 kB pamięci RAM. Trudno tutaj nie dostrzec pewnego rozmachu w doborze mikrokontrolera. W rzeczywistości jednak użycie mniejszej jednostki nie wpłynie znacząco na całkowity koszt urządzenia, podczas gdy w zastosowanej konfiguracji mamy całkiem niezłą zgodność sprzętową i programową z platformą sterownika „PWMLEDz". Jeśli chcemy zbudować sterownik „Relayz" w wersji bardziej oszczędnej lub gdy potrzebujemy większą ilość jego egzemplarzy, dobrym pomysłem może być zastosowanie mikrokontrolera STM32F100 z ekonomicznej serii „Value Line". Wymaga to wprowadzenia niewielkich modyfikacji do konfiguracji programu sterującego i oczywiście ponownej kompilacji źródeł. W takim scenariuszu projekt płytki może pozostać ten sam dzięki dobrej zgodności sprzętowej obu serii mikrokontrolerów.

W ofercie AVT*

AVT-5503 A **AVT-5503 UK**

Podstawowe informacje:

- Możliwość włączania 8 odbiorników energii elektrycznej przez łącze RS-485 i protokół SPPoB (opisany w EP 10/2014 i EP 11/2014).
- Płytka o wymiarach 94 mm×82 mm, możliwość zamontowania w obudowie Z-101 na szynę DIN.
- Nominalne napięcie zasilania: 12 V DC.
- Pobór prądu przy wyłączonych przekaźnikach: ok. 23 mA.
- Pobór prądu przy włączonych wszystkich przekaźnikach typu FRM18A-5 (wersja na 12 V): ok. 100 mA.
- Obciążalność kanałów wyjściowych 3 A lub 5 A zależnie od zastosowanych przekaźników.
- Włączanie, wyłączanie, przełączanie stanu każdego wyjścia.
- Możliwość chwilowego włączania lub wyłączania wyjść na czas z zakresu 100 ms…25,5 s wybierany z rozdzielczością 100 ms.
- Możliwość odczytu stanu wyjść i ustawionych parametrów czasowych.

Dodatkowe materiały na FTP:

ftp://ep.com.pl, user: 07641, pass: yus9jv2r

- wzory płytek PCB

Projekty pokrewne na FTP:

(wymienione artykuły są w całości dostępne na FTP)

AVT-1825 Moduł PWM z interfejsem RS485 (EP 8/2014)

AVT-1745 Miniaturowy moduł przekaźników z RS485 (EP 7/2013)

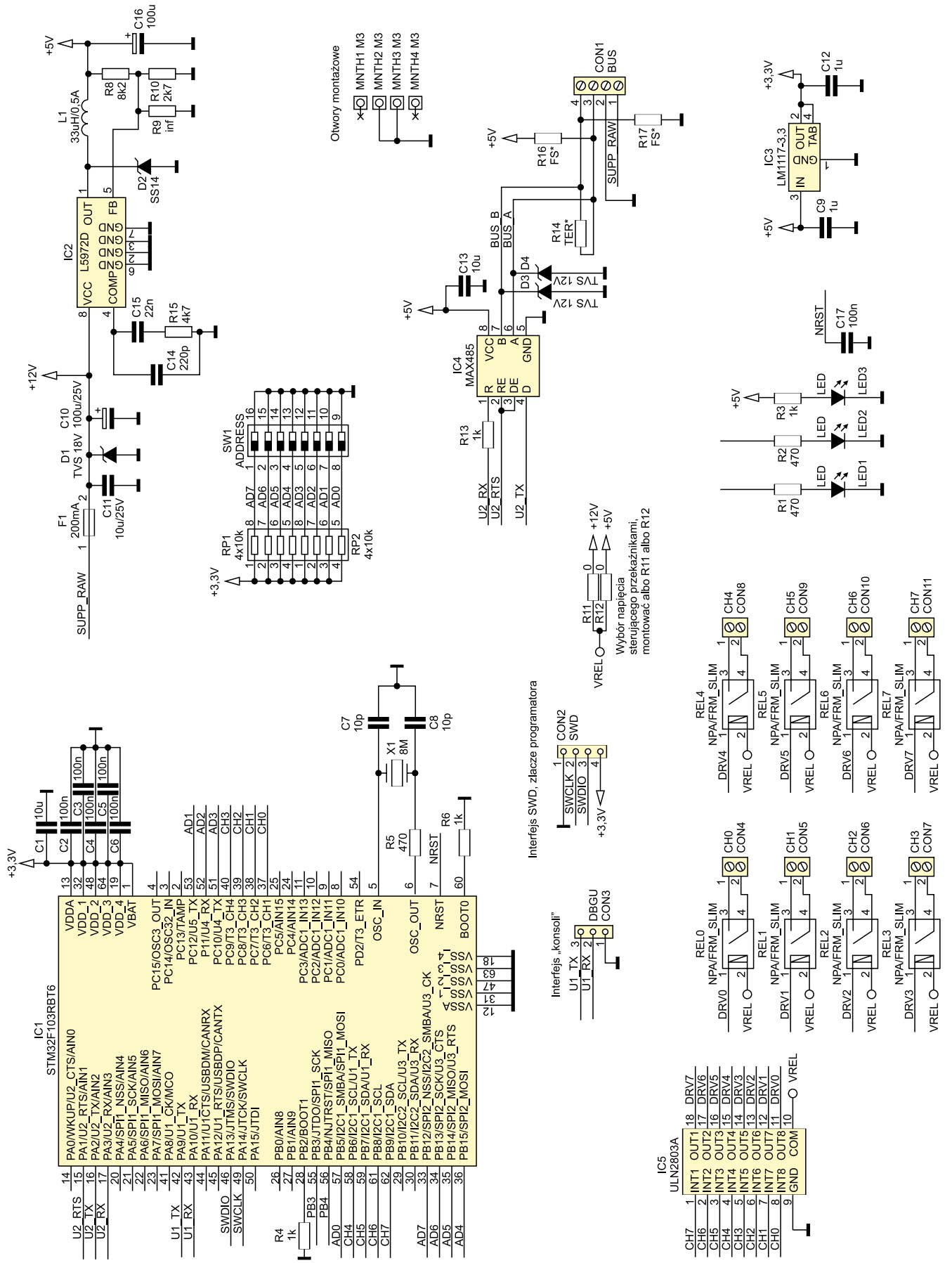
* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:

AVT xxxx UK	to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A	płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+	płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B	płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C	to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD	oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

Montaż i uruchomienie

Schemat montażowy sterownika „Relayz" pokazano na **rysunku 3**. Jeśli planujemy umieszczenie płytki w obudowie Z-101 na szynę DIN, to jeszcze przed rozpoczęciem montażu warto sprawdzić, czy płytka będzie pasowała do konkretnego egzemplarza obudowy, którą dysponujemy. Najpierw należy wyciąć zaznaczone na warstwie opisowej narożniki płytki. Następnie zalecane jest także sprawdzenie, czy na pewno rozstaw otworów montażowych będzie pasował do odpowiednich „nóżek"



Rysunek 2. Schemat ideowy sterownika „Relayz"

w obudowie. W praktyce obudowy od różnych producentów o tym samym oznaczeniu Z-101 mogą mieć różne odległości pomiędzy mocowaniami płytki. Choć są to różnice rzędu 1-2 mm, to i tak może to sprawić niepotrzebny kłopot. Z tego powodu otwory montażowe w płytce są większe niż wkręty mocujące, a może być nawet konieczne ich dalsze rozwiercenie lub delikatne wyfrezowanie w odpowiednią stronę. W egzemplarzach prototypowych do zamocowania płytki w podstawie obudowy zastosowano wkręty do plastiku w rozmiarze 3×6 mm. Gdy płytka jest wstępnie dopasowana do obudowy, przechodzimy do montażu elementów elektronicznych.

Układ może poprawnie działać bez elementów zabezpieczających: D1, D3 i D4. Opcjonalne są także rezystory R14, R16 i R17 mogące pełnić rolę: terminatora (R14) oraz rezystorów *fail-safe* (R16 i R17) dla magistrali RS-485. Bezpiecznik F1 można zastąpić zworą, jednak zalecane jest jego zamontowanie wraz z diodą D1. W egzemplarzach testowych zastosowano bezpiecznik F1 na prąd znamionowy 200 mA, co wydaje się rozsądną wartością, przy której sterownik powinien działać zarówno z przekaźnikami na 5 V jak i na 12 V. Rozmiar bezpiecznika F1, który przewidziano na płytce, to 1812. W projekcie płytki usunięto jednak warstwę miedzi pomiędzy jego polami kontaktowymi, dzięki czemu bez problemu można zamontować F1 w nieco mniejszej obudowie, np. 1206.

Zależnie od tego, na jakie napięcie cewki uda nam się nabyć przekaźniki, należy zamontować w odpowiednim miejscu zworę konfiguracyjną. Zworę stanowić może rezystor o wartości 0 Ω w rozmiarze 1206 zamontowany albo w miejscu R11, albo R12. Gdy przekaźniki będą uruchamiane napięciem 12 V, montujemy zworę 0 Ω w miejscu R11, natomiast w przypadku przekaźników na 5 V, zwora powinna znaleźć się w miejscu R12.

Diody LED nie muszą być montowane. Mogą jednak przydać się przy uruchamianiu urządzenia lub pisaniu własnego, bardziej rozbudowanego oprogramowania dla sterownika. Rezystory R1…R3 ograniczające prąd diod LED mogą mieć oporność nawet ok. 1 kΩ lub więcej przy zastosowaniu współczesnych, jasnych diod, które wyraźnie świecą nawet przy prądzie nieprzekraczającym 1 mA.

Kolejność montażu i uruchamiania można przyjąć dowolnie, jednak przedstawiona dalej zalecana kolejność może uratować mikrokontroler lub układ transceivera w razie ewentualnych pomyłek. W pierwszej kolejności zalecałbym montować „drobnicę", tj.: rezystory, drabinki rezystorowe, diody, kondensatory, dławik, bezpiecznik. Oprócz tego montujemy złącze CON1 i stabilizatory: IC2 i IC3. Zależnie od typu posiadanych przekaźników montujemy albo R11, albo R12.

Tabela 1. Wykaz poleceń obsługiwanych przez sterownik „Relayz"

Kategoria komendy / identyfikatora	Nazwa komendy / identyfikatora w payload[0]	Kod	Ustawienie len	Opis
Ogólne	SPP_ID_ALL_ON	0x04	1	Włącz wszystkie kanały w urządzeniu
	SPP_ID_ALL_OFF	0x05	1	Wyłącz wszystkie kanały w urządzeniu
	SPP_ID_ALL_ON_ALT	0x06	1	Włącz wszystkie kanały chwilowo (zob. opis komendy SPP_ID_TRANSITION)
	SPP_ID_ALL_OFF_ALT	0x07	1	Wyłącz wszystkie kanały chwilowo (zob. opis komendy SPP_ID_TRANSITION)
	SPP_ID_MULTICHANNEL	0x08	≥4	Właściwa komenda zawarta w payload[3] i skierowana do wybranych kanałów, zadanych maskami w payload[1], payload[2]
Dla wybranych kanałów wyjściowych	SPP_ID_CHAN_ON	0x1n	1	Włącz kanał „n" (n=0x0...0x7)
	SPP_ID_CHAN_OFF	0x2n	1	Wyłącz kanał „n"
	SPP_ID_CHAN_TOGGLE	0x3n	1	Zmień stan kanału „n"
	SPP_ID_CHAN_ON_ALT	0x4n	1	Włącz kanał „n" na zadany czas
	SPP_ID_CHAN_OFF_ALT	0x5n	1	Wyłącz kanał „n" na zadany czas
	SPP_ID_CHAN_VALUE	0x9n	1+ilość kanałów do ustawienia	Zadaj wartość (0 lub >0) dla wybranych kanałów, rozpoczynając od „n"; jasność wpisujemy do payload[1, 2, ...], a o ilości kanałów do ustawienia decyduje długość pakietu
	SPP_ID_CHAN_VALUE_ALT	0xAn		J.w., tylko wpisanie wartości niezerowej spowoduje chwilowe włączenie kanału, a zero spowoduje chwilowe wyłączenie kanału
	SPP_ID_TRANSITION	0xBn		Ustawienie czasu chwilowego włącznia/wyłączenia dla wybranych kanałów począwszy od „n"; o ilości konfigurowanych kanałów decyduje długość pakietu; jednostka to „cykl" czyli wielokrotność 100 ms (np. domyślna wartość 5 oznacza 0,5 s)
Komunikacja żądanie/odpowiedź	SPP_ID_REQ / SPP_ID_RESP	0xE0 / 0xF0 0xE2 / 0xF2	1 / 9	Odczyt stanu przekaźników; wartości zwracane są wg typu wyliczeniowego T_RelayAction: 0=wyłączony, 1=włączony, 2=chwilowo włączony, 3=chwilowo wyłączony
		0xE3 / 0xF3		Odczyt ustawionych czasów chwilowego załączenia/wyłączenia (w cyklach 100-milisekundowych)
		0xE4 / 0xF4		Odczyt, ile cykli (100-milisekundowych) zostało do zakończenia chwilowego włączenia/wyłączenia

Uruchomienie

Włączamy zasilanie urządzenia. Dioda LED3 powinna świecić. Sprawdzamy multimetrem obecność napięć zasilania 3,3 V, 5 V oraz 12 V w odpowiednich miejscach, np. na padach odpowiadających wyprowadzeniom zasilania układów scalonych. Napięcie oznaczone jako „+5 V" może być w rzeczywistości trochę niższe ze względu na dobór R8 i R10 spośród łatwo dostępnych wartości. Jeśli nie stwierdzimy niepoprawnego działania układu zasilania, odłączamy cały układ od zasilacza i przechodzimy do dokończenia montażu. Wlutowujemy: mikrokontroler, driver IC5, transceiver IC4 oraz resztę „dużych" elementów, takich jak złącza, przełącznik DIPSwitch, przekaźniki.

Dalej pozostaje już tylko wgrać przygotowany firmware lub ewentualnie zmodyfikować oprogramowanie wedle własnego uznania. Dostarczone oprogramowanie było kompilowane na *arm-none-eabi-gcc* z pakietu Yagarto. Do programowania pamięci Flash mikrokontrolera można użyć wystawionego na złączu CON2 interfejsu SWD posługując się odpowiednim programatorem. Przewidziano także interfejs „konsoli" (terminala) na złączu CON3 – są to po prostu bezpośrednio wyprowadzone linie nadawcza i odbiorcza interfejsu USART1. Może on być przydatny przy prowadzeniu własnych eksperymentów z rozbudową oprogramowania. Domyślnie komunikaty „konsolowe" wysyłane są z następującymi parametrami transmisji: 115200 baud, 8 bitów danych, 1 bit stopu, brak bitu parzystości.

Interfejs i funkcjonalność

Dysponując zmontowanym sterownikiem „Relayz" z wgranym domyślnym programem sterującym, przystępujemy do praktycznego zapoznania się z podstawami jego obsługi. Adres urządzenia ustalamy przełącznikiem DIPSwitch. Adres ten powinien „mieć sens" wg założeń protokołu SPPoB, co m.in. oznacza, że nie jest zalecane ustawienie na nim któregoś z adresów broadcastowych (*0xFF*) lub multicastowych (np. *0x4F*).

W **tabeli 1** umieszczono skrócony opis obsługiwanych poleceń SPPoB. Część poleceń zachowuje taką samą składnię, jaka była zastosowana dla sterownika „PWMLEDz", aby zapewnić możliwie dobry stopień kompatybilności obu tych urządzeń. Przykładowo, sterownik „Relayz" obsługuje polecenie wpisywania arbitralnej wartości do jednego bądź wielu kanałów *SPP_ID_CHAN_VALUE* podobnie jak „PWMLEDz", lecz potrafi on rozróżnić jedynie podane wartości zerowe (wyłącz kanał) i niezerowe (włącz kanał).

Polecenia sterowania „alternatywnego" (przyrostek *_ALT*) służą tutaj do realizacji chwilowego włączania lub wyłączania kanałów. Może to być potrzebne np. do sterowania roletami bądź innymi mechanizmami lub do zerowania routera przez chwilowe odcięcie zasilania. Chwilowe włączanie polega na bezwzględnym włączeniu kanału w momencie otrzymania komendy, odczekania zadanego czasu i bezwzględnego wyłączenia kanału. Po jego wykonaniu kanał pozostanie wyłączony niezależnie od stanu przed wydaniem komendy. Analogicznie działa chwilowe wyłączanie, po którym kanał pozostaje włączony. Tym sposobem polecenia „chwilowe" w połączeniu z odpowiednim stanem początkowym mogą być użyte również do opóźniania wykonania akcji wyłączenia bądź włączenia.

Czas chwilowego włączenia/wyłączenia liczony jest w cyklach. W podstawowej wersji firmware 1 cykl trwa 0,1 s. Po włączeniu zasilania sterownika ilość cykli chwilowego włączenia/wyłączenia dla każdego kanału ustawiona jest na 5, co odpowiada czasowi 0,5 s. Możemy te nastawy zmienić wydając komendę *SPP_ID_TRANSITION* z jednym lub kilkoma parametrami odpowiadającymi nowym wartościom czasu chwilowego włączania/wyłączania.

Odczyt parametrów przeprowadza się w sposób uprzednio opisywany dla sterownika „PWMLEDz". Tutaj jednak, dla zachowania kompatybilności z „PWMLEDz", dwa zapytania (*0xE0* i *0xE2*) zwracają dane, które reprezentują dokładnie to samo, czyli ustawione stany kanałów. W „PWMLEDz" żądanie *0xE0* dotyczyło poziomów docelowej jasności kanałów, a *0xE1* chwilowej – teraz te wielkości nie są rozróżniane, więc opisują je te same dane. Stan i akcje każdego kanału są tutaj reprezentowane przez typ wyliczeniowy *T_RelayAction* zdefiniowany w *drivers/relays.h* (**listing 1**). Dlatego w odpowiedziach na żądania *0xE0* i *0xE2* otrzymamy w pakiecie ciąg 8 liczb opisujących poszczególne kanały. Dla każdego kanału: 0 oznacza, że jest stabilnie wyłączony, 1 oznacza, że jest stabilnie włączony, 2 oznacza chwilowe włączenie, a 3 chwilowe wyłączenie. Czas, jaki pozostał do końca chwilowego włączenia/wyłączenia możemy sprawdzić wysyłając żądanie *0xE4*. W odpowiedzi o identyfikatorze *0xF4* otrzymamy wtedy także 8 liczb, tym razem oznaczających, ile jeszcze cykli pozostało do zakończenia stanu chwilowego włączenia/wyłączenia, bądź 0, jeśli stan kanału jest stabilny.

Przykłady obsługi

Procedury testowania urządzeń z protokołem SPPoB zostały opisane w poprzednich artykułach. Tam też odsyłam w celu uzyskania informacji, jak należy zestawić stanowisko testowe. Tutaj zakładam, że do urządzenia możemy wysyłać przez interfejs RS-485 bajty i całe pakiety. Parametry transmisji pozostają bez zmian względem poprzednich opracowań, tj. 19200 baud, 8 bitów danych, brak bitu parzystości, 1 bit stopu. W przedstawionych dalej przykładach adres źródłowy oznaczamy jako „SRC", docelowy jako „DST", bajt sumy kontrolnej jako „CRC".

W dalszych przykładach liczby zapisane są domyślnie w systemie szesnastkowym bez przedrostka *0x*. Kod komendy jest zaznaczony pogrubioną czcionką. Dla wielu komend, w mniej znaczącej cyfrze kodu komendy zawarty jest numer kanału (tutaj przekaźnika) oznaczony „n". Za „n" oczywiście możemy wstawić cyfrę w zakresie od 0 do 7. Jeśli cyfra reprezentująca kanał nie ma znaczenia (np. dla komend złożonych wielokanałowych), możemy ją przyjąć dowolnie i oznaczona jest przez „x" (w przykładach przyjęto „0").

Dla przypomnienia, ogólna struktura pakietu SPPoB jest następująca:
[7E, DST, SRC, len (długość danych), **payload[0]**, payload[1], ..., payload[len-1], CRC],
a podstawowa komenda-identyfikator zawartości pakietu znajduje się w payload[0]. Pamiętamy także o rozbijaniu bajtów 7E i 7D z wnętrza pakietu na sekwencje „escape".
Włączamy przekaźnik „n" (n=0...7) pakietem: [7E, DST, SRC, 01, **1n**, CRC].
Wyłączamy przekaźnik „n": [7E, DST, SRC, 01, **2n**, CRC].
Zmieniamy stan przekaźnika „n": [7E, DST, SRC, 01, **3n**, CRC].
Włączamy przekaźnik „n" na chwilę: [7E, DST, SRC, 01, **4n**, CRC].
Ustawienie czasu włączenia/wyłączenia w kanale „n" na liczbę EE cykli uzyskamy wysyłając następujący pakiet:

Listing 1. Definicja typu wyliczeniowego T_RelayAction.

```
typedef enum {
    RELAY_OFF = 0,
    RELAY_ON = 1,
    RELAY_PULSE_ON = 2,
    RELAY_PULSE_OFF = 3,
    RELAY_TOGGLE = 4
}T_RelayAction;
```

Listing 2. Funkcja main.

```
int main(void)
{
    init();
    while(1)
    {
//every 1ms
        if(tick_upd)
        {
            tick_upd = 0;
            relProcess();
            if(sppRx(&packet))
            {
                processRxedPacket(&packet);
            }
            sppIdleTimeout();
            blink();
            #if USE_WATCHDOG
                IWDG_ReloadCounter();
            #endif
        }
    }
    return 0;
}
```

Wykaz elementów

Rezystory:
R11, R12: 0 Ω (SMD 1206, montować tylko jeden)
R1, R2, R5: 470 Ω (SMD 0603)
R3, R4, R6, R13: 1 kΩ (SMD 0603)
R10: 2,7 kΩ (SMD 0603)
R15: 4,7 kΩ (SMD 0603)
R8: 8,2 kΩ (SMD 0603)
R9: nie montować
RP1, RP2: drabinka 4×10 kΩ (SMD 1206)
Kondensatory:
C7, C8: 10 pF (0603)
C14: 220 pF (0603)
C15: 22 nF (0603)
C2...C6, C17: 100 nF (0603)
C9, C12: 1 µF (0805)
C11: 10 µF/25 V (1206)
C1, C13: 10 µF/6,3 V (0805)
C10: 100 µF/25 V (tantalowy, SMD, rozm. D)
C16: 100 µF/6,3 V (tantalowy, SMD, rozm. D)
Półprzewodniki:
D1: TVS 18 V jednokierunkowy, np. SMBJ18A
D2: SS14, obud. SMA
D3, D4: 12 V jednokierunkowy, np. SMBJ12A lub dwukierunkowy
IC1: STM32F103RBT6 (LQFP64)
IC2: L5972D (SO-8)
IC3: LM1117-3.3 (SOT-223)
IC4: MAX485 lub zamiennik (SO-8)
IC5: ULN2803A (SO-18)
LED1...LED3: diody LED (SMD 0805)
Inne:
CON1: złącze śrubowe 4p (lub 2×2p), raster 5,0 mm
CON2: goldpin 4p, 2,54 mm
CON3: goldpin 3p, 2,54 mm
CON4...CON11: złącze śrubowe 2p, raster 5,0 mm
F1: bezpiecznik polimerowy ok. 200 mA, maks. rozmiar 1812
L1: dławik mocy, 33 µH, zalecany prąd pracy ≥1 A, maks. rozmiar 12x12 mm, SMD
REL0...REL7: przekaźnik typu NPA lub FRM, np. FRM18A-5 (5 A, 12 V)
SW1: Dipswitch 8p, (DIP-16)
X1: rezonator kwarcowy 8 MHz, przewlekany

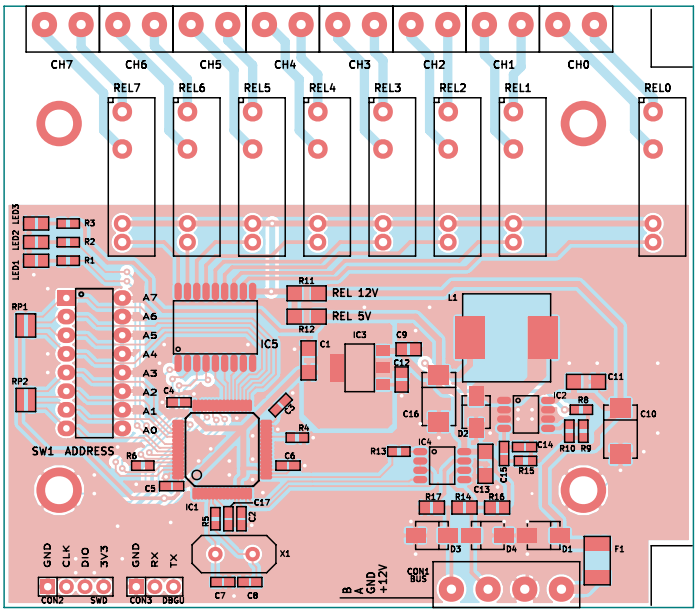
[7E, DST, SRC, 02, **Bn**, EE, CRC],
np. dla n=1:
[7E, DST, SRC, 02, **B1**, EE, CRC].
Ustawienie czasu włączenia/wyłączenia w kanałach 1, 2 i 3 na odpowiednio 11, 22, 33 (hex):
[7E, DST, SRC, 04, **B1**, 11, 22, 33, CRC].
Ustawienie arbitralnych wartości (9n) w kanałach 4, 5, 6 na odpowiednio 1, 1, 0 (0=wyłączony, 1=włączony):
[7E, DST, SRC, 04, **94**, 1, 1, 0, CRC].
Zastosowanie komendy „multichannel" (kod 08) do włączenia (kod 1x) kanałów 7, 5, 2, 0 (maska 00A5):
[7E, DST, SRC, 04, **08**, 00, A5, **10**, CRC]
Zastosowanie komendy „multichannel" (kod 08) do ustawienia arbitralnej wartości 01 (kod 9x) w kanałach 5, 4, 3, 2 (maska 003C):
[7E, DST, SRC, 05, **08**, 00, 3C, **90**, 01, CRC]
Odczyt zadanych czasów wykonamy wysyłając żądanie: [7E, DST, SRC, 01, **E3**, CRC].
Otrzymamy odpowiedź: [7E, DST, SRC, 09, **F3**, CH(0), CH(1), CH(2), CH(3), CH(4), CH(5), CH(6), CH(7), CRC].
Odczyt stanu kanałów wykonamy wysyłając np. żądanie: [7E, DST, SRC, 01, **E0**, CRC].
Otrzymamy odpowiedź: [7E, DST, SRC, 09, **F0**, CH(0), CH(1), CH(2), CH(3), CH(4), CH(5), CH(6), CH(7), CRC].

Struktura programu sterującego

Tę część artykułu polecam Czytelnikom, którzy planują modyfikację oprogramowania sterownika lub po prostu chcą dowiedzieć się, „jak to działa".

Podstawowe parametry SPPoB możemy zmodyfikować typowo, jak we wcześniej opisywanych urządzeniach SPPoB, czyli edytując plik *spp_cfg.h* umieszczony w katalogu głównym projektu. Większość kodu specyficznego dla sterownika „Relayz" znajduje się w funkcji *main* (**listing 2**) oraz modułach programowych z podkatalogu *drivers*. W głównej pętli programowej cały kod wykonywany jest co ok. 1 ms, czyli za każdym razem, kiedy ustawiona będzie flaga *tick_upd*. Flaga ta jest ustawiana w funkcji obsługi przerwania *SysTick_Handler*.

W głównej pętli programowej cyklicznie wywoływane są funkcje odpowiadające za podstawowe elementy projektu sterownika. Funkcja ***relProcess*** (z modułu *drivers/relays.c*) służy do zapewnienia niezależnego sterowania wszystkimi przekaźnikami, w tym do odliczania czasu trwania chwilowego włączenia/wyłączenia przy pomocy zestawu liczników i instrukcji warunkowych. Po jej wywołaniu następuje sprawdzenie, czy nie przyszedł nowy pakiet SPPoB. Jeśli tak – zostaje wywołana funkcja ***processRxedPacket*** (*drivers/spp_proc.c*), do której dodatkowo przekazany jest wskaźnik do struktury reprezentującej odebrany pakiet. Dalej następuje analiza zawartości pakietu, wykonanie odebranych poleceń i ew. odesłanie odpowiedzi. Kolejnym krokiem pętli głównej jest odliczenie czasu „timeoutu" dla zajętości magistrali przez wywołanie *sppIdleTimeout* (*spp.c* w katalogu nadrzędnym). Wreszcie, już całkowicie „nadobowiązkowo", wywołana jest funkcja odpowiadająca za miganie diodą LED1 (urządzenie potwierdza, że pracuje) i ewentualne resetowanie licznika watchdog jeśli odpowiednia makrodefinicja ma wartość niezerową.



Rysunek 3. Schemat montażowy sterownika

Obsługa przekaźników od strony programowej odbywa się przez dedykowany sterownik zaimplementowany w module *drivers/relays.c*. Korzysta z niego wysokopoziomowy moduł przetwarzania komend SPPoB z *drivers/spp_proc.c*. Podstawową funkcją jest tutaj

int ***relSet****(uint8_t relId, T_RelayAction newState)*

przyjmująca jako pierwszy argument numer przekaźnika (od 0 do 7), którego stan chcemy zmodyfikować oraz, w drugim argumencie, akcję, jaką chcemy wykonać. Akcje zakodowane są we wspomnianym wcześniej (list. 1) typie wyliczeniowym *T_RelayAction*.

Czas chwilowego włączenia bądź wyłączenia dla poszczególnych przekaźników możemy ustalić wywołując funkcję ***relSetPulseTime***. Czas ten dla każdego kanału jest przechowywany wewnątrz modułu *relays.c* w tablicy struktur *relay* typu *T_Relay* wraz z licznikami odpowiedzialnymi za odliczanie czasu trwania impulsu. Takie podejście zwalnia funkcje nadrzędne od liczenia czasu i przenosi logikę sterowania do niższych warstw.

Pozostałe funkcje sterownika z modułu *relays.c* raczej nie wymagają szerszego komentarza i nie powinno być problemów z ich samodzielną analizą.

Podsumowanie

Opisany moduł przekaźników nie jest ani czymś nietypowym, ani skomplikowanym. Jest to natomiast jeden z najbardziej podstawowych elementów wykonawczych przy budowaniu wszelkiego rodzaju domowej (i nie tylko) automatyki. Zaprezentowany tutaj moduł „Relayz" starałem się w miarę możliwości obudować w choć trochę bardziej zaawansowaną funkcjonalność niż „włącz-wyłącz", co pozwoliłoby na jego bardziej wszechstronne zastosowanie. Stąd m.in. pomysł na czasowe włączanie bądź wyłączanie kanałów – raczej rzadko spotykane w innych konstrukcjach hobbystycznych, a które w bardzo wygodny sposób zwalnia „logikę nadrzędną" z konieczności odmierzania czasu dla wielu kanałów jednocześnie.

Sterownik „Relayz" nie posiada w swoim oprogramowaniu zabezpieczeń przed pakietami częściowo niemającymi sensu, w których np. zawarto komendę jednobajtową bez parametrów, podczas gdy długość danych w pakiecie została ustawiona na kilka bajtów. W takim przypadku, jeśli tylko struktura pakietu jest poprawna, sterownik wykona komendę z pierwszego bajtu, a resztę danych po prostu zignoruje. Zabezpieczenie przed tego typu sytuacjami jest łatwe do zaimplementowania, choć jest to dość żmudna czynność. W przedstawionej implementacji przyjęto założenie, że pakiety są komponowane i wysyłane przez urządzenia upoważnione do tego, więc takie, które – można by powiedzieć – *wiedzą co robią*. Natomiast użytkownik końcowy nie jest upoważniony do komponowania treści pakietów.

Otwarte oprogramowanie z prostym szkieletem i „wysokopoziomowym" sterownikiem przekaźników ma nie tylko pozwalać, ale wręcz zachęcać Czytelników do jego modyfikacji bądź wykorzystania we własnych projektach. Prezentowana tutaj część sprzętowa jest na tyle mało specjalizowana (chciałoby się rzec *generyczna*), że można ją z powodzeniem zastosować do projektu przekaźnikowego sterownika z zupełnie innym protokołem niż tutaj zaproponowany lub też z bardziej zaawansowaną wewnętrzną logiką działania.

Robert Brzoza-Woch
robert.brzoza@gmail.com