AVT 5258

Sterownik domowych odbiorników energii

Urządzenie umożliwia sterowanie odbiornikami energii elektrycznej załączając je zgodnie z harmonogramem. Może symulować obecność domowników lub po prostu ułatwiać im życie. Sterowanie i konfigurowanie jest wykonywane za pomocą nadajnika podczerwieni. ***Rekomendacje:*** *budowę urządzenia szczególnie polecamy majsterkowiczom chcącym nieco ułatwić sobie życie.*

Obecnie obserwuje się gwałtowny rozwój różnego rodzaju systemów sterowania. Popularny na wielu forach staje się temat inteligentnych domów i systemów sterujących poszczególnymi ich elementami. Jednak zastosowanie takich inteligentnych systemów w niewielkim mieszkaniu jest nieopłacalne. Koszt systemu przeznaczonego do sterowania dużą liczbą urządzeń będzie w tym przypadku będzie nieproporcjonalnie wysoki. Właśnie w celu ułatwienia codziennego życia w niewielkim mieszkaniu powstało urządzenie sterujące. Umożliwia ono załączanie odbiorników zasilanych z sieci 230 VAC zgodnie z zadanym harmonogramem. Poza podstawową funkcją włączania/wyłączania sterownik ma także inne możliwości, jak wyłączenie zasilania wybranego gniazda po zadanym czasie, załaczanie/wyłaczanie zasilania wybranego gniazda zgodnie z utworzonym harmonogramem oraz możliwość prostego sterowania mocą np. podłączonego oświetlenia. Dla ułatwienia podłączania kontrolowanych odbiorników energii, sterownik jest wyposażony w klasyczne gniazda z kontaktem ochronnym PE. Konfigurowanie urządzenia odbywa się przy użyciu nadajnika podczerwieni z kodami RC5.

Obsługa urządzenia

Na przedniej płycie sterownika znajdują się trzy gniazda, do których można podłączyć sterowane urządzenia. Umieszczono na niej również diody sygnalizacyjne, informujące czy dany punkt jest włączony oraz sygnalizujące odbiór kodu RC5 z pilota, odbiornik podczerwieni oraz wyświetlacz

AVT-5258 w ofercie AVT:
AVT-5258A – płytka drukowana

Podstawowe informacje:
- Sterowanie trzeba obciążeniami zasilanymi z sieci 230 VAC
- Załączanie i wyłączanie automatyczne (na podstawie harmonogramów) lub za pomocą nadajnika RC5
- Konfigurowanie ustawień za pomocą nadajnika RC5
- Wbudowany zegar z bateryjnym podtrzymaniem zasilania
- Możliwość ściemniania żarówek

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 16719, pass: 8b13241g
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Projekty pokrewne na CD i FTP:
(wymienione artykuły są w całości dostępne na CD)

AVT-2794	Automatyczny sterownik oświetlenia (EdW 8/2006)
AVT-3014	Automatyczny sterownik oświetlenia (EdW 4/2002)
AVT-1223	Czasowy wyłącznik oświetlenia (EP 8/2001)
AVT-445	Inteligentny sterownik oświetlenia (EP 6/1998)
AVT-1133	Inteligentny regulator oświetlenia (EP 12/1997)
Projekt 089	Zdalnie sterowany regulator oświetlenia (EP 8/2001)
Projekt 051	Uniwersalny sterownik oświetlenia (EP 9/1998)

LCD. Płytę czołową pokazano na **rysunku 1**. Po prawej stronie sterownika umieszczono gniazdo zasilania oraz bezpieczniki, oddzielnie dla modułu sterowania oraz kontrolowanych urządzeń. Dodatkowo, po lewej stronie (w otworze) umieszczono przycisk zerowania. Jego działanie jest następujące: po krótkim wciśnięciu następuje restart mikrokontrolera, natomiast po długim wciśnięciu (aż do momentu zapalenia się zielonej diody) następuje wyzerowanie sterownika oraz usunięcie nastaw (usuwana jest tablica zdarzeń i przywracane są standardowe kody PIN).

Do sterowania urządzeniem zastosowano uniwersalny nadajnik podczerwieni firmy Thomson typu ROC3205. Aby współpracował on z urządzeniem należy go skonfigurować. W tym celu należy wcisnąć przycisk SETUP i trzymać go wciśniętym aż do momentu zapalenia się diody, a następnie wcisnąć przycisk TV (przyciśniecie zostanie zasygnalizowane krótkim zgaszeniem się diody) i wprowadzić czterocyfrowy kod **0094**.

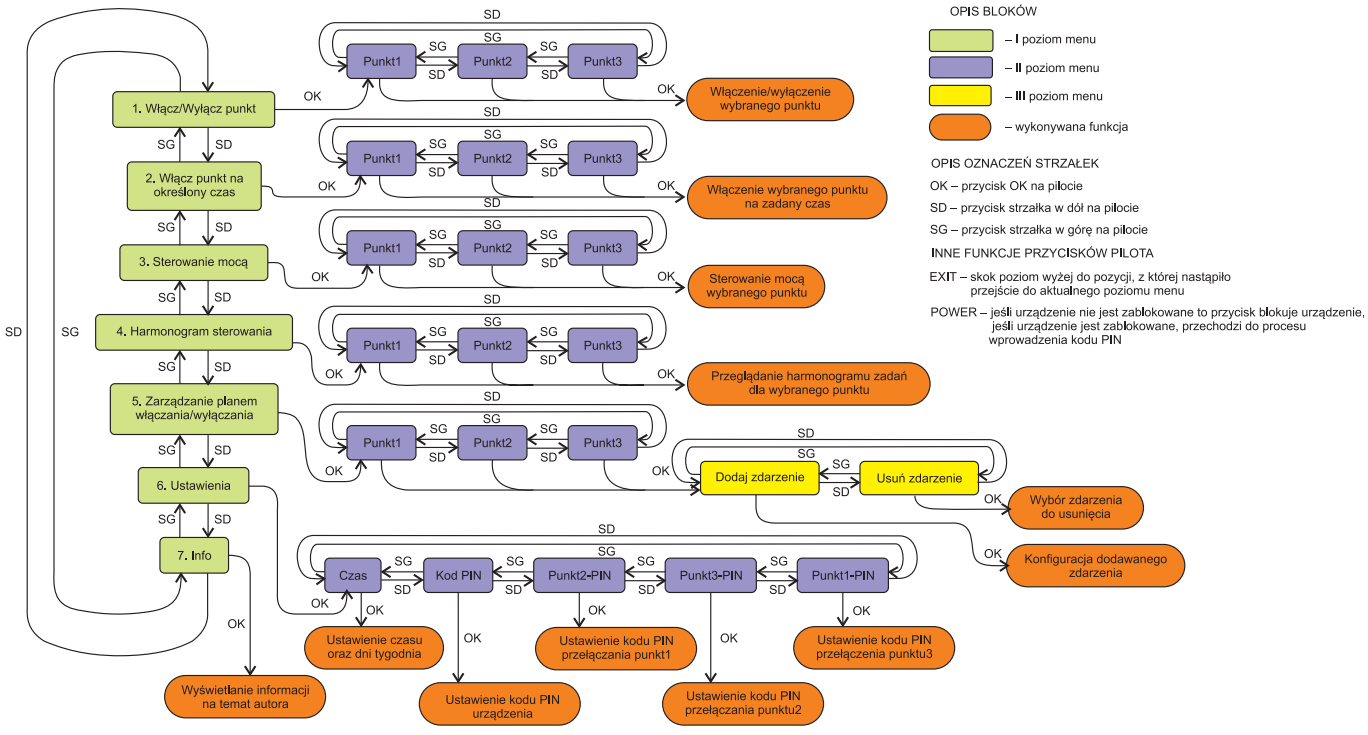
Nadajnik obsługuje do trzech urządzeń. Wybór aktualnie kontrolowanego odbywa się przez naciśnięcie jednego z przycisków: TV, DVD lub SAT. Do obsługi tego sterownika służą kody dostępne po naciśnięciu klawisza TV. Poszczególne klawisze mają następujące funkcje:

- klawisze numeryczne *0...9* używane są do wprowadzania danych numerycznych,
- klawisz *POWER*, jest używany do odblokowania sterownika (jeśli jest on zablokowany) oraz do szybkiego jego zablokowania (jeśli jest on odblokowany),
- klawisz *OK* służy do akceptacji oraz przejścia o 1 poziom niżej w menu,

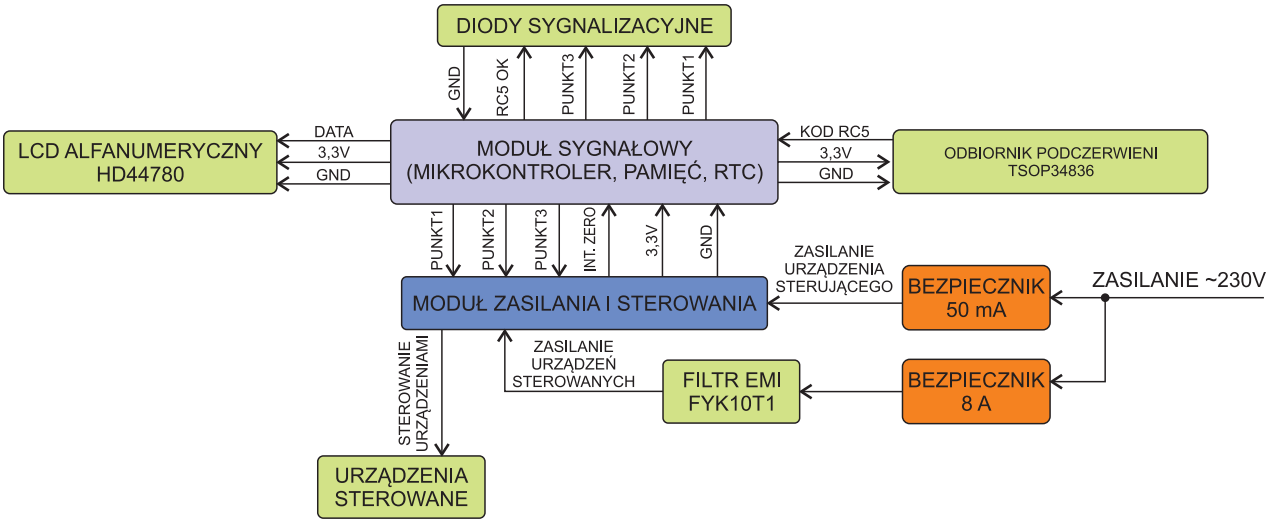
Rysunek 1. Płyta czołowa sterownika

- klawisz *EXIT* służy do cofnięcia wprowadzanych zmian, wyjścia z menu o poziom wyżej,
- *strzałka w górę* służy do zmiany pozycji w menu oraz do zwiększania wartości parametru w czasie wprowadzania ustawień,
- *strzałka w dół* służy do zmiany pozycji w menu oraz do zmniejszania wartości parametru w czasie wprowadzania ustawień,
- *strzałka w lewo* służy do zmiany wskazywanej pozycji podczas wprowadzania ustawień (przesunięcie kursora w lewo).
- strzałka w prawo służy do zmiany wskazywanej pozycji podczas wprowadzania ustawień (przesunięcie kursora w prawo).

W stanie czuwania urządzenie jest zablokowane. Aby je odblokować należy wprowadzić czterocyfrowy kod PIN. W tym celu należy wcisnąć na pilocie przycisk *POWER*, co spowoduje uruchomienie funkcji odczytu kodu PIN. W przypadku pomyłki, poprzez wciśnięcie klawisz *EXIT* można wyczyścić wprowadzone cyfry i ponownie wprowadzić kod. Po 7 sekundach bezczynności następuje powrót do głównego ekranu oraz blokada sterownika.



Rysunek 2. Struktura menu sterownika



Rysunek 3. Schemat połączeń komponentów zewnętrznych z płytkami sterownika

W zależności od wprowadzonego kodu PIN nastąpi odblokowanie urządzenia lub przełączenie odpowiedniego punktu. Domyślne kody PIN są następujące:

- 1111 – wejście do menu urządzenia,
- 2222 – zmiana stanu gniazda punkt 1,
- 3333 – zmiana stanu gniazda punkt 2,
- 4444 – zmiana stanu gniazda punkt 3.

Kody PIN można zmienić w ustawieniach. Należy pamiętać, że ten sam kod nie może zostać przypisany do różnych czynności.

Urządzenie ma rozbudowane menu (**rysunek 2**). Konfigurowanie większości ustawień jest intuicyjne, dokładniejszego wyjaśnienia wymaga jedynie dodawanie nowego zdarzenia do harmonogramu.

Proces ten składa się z trzech etapów: nastawy czasu włączenia, nastawy czasu wyłączenia oraz konfigurowania cykliczności zdarzenia. Przełączanie pomiędzy poszczególnymi nastawami jest wykonywane za pomocą przycisków pilota *strzałka w górę* i *strzałka w dół*. Przy wprowadzaniu czasu włączania/wyłączania dniom odpowiadają klawisze *1...7*, klawiszowi *8* odpowiada codzienne załączanie. Klawisz *0* powoduje usunięcie czasu włączania bądź wyłączania. Przy nastawie cykliczności zdarzenia możemy ją włączyć naciskając klawisz *1* lub wyłączyć naciskając klawisz *2*. Nastawa zadania o wyłączonej cykliczności zostanie po wykonaniu usunięte z pamięci.

Budowa i zasada działania

Sterownik składa się z dwóch modułów: sygnałowego i sterująco-zasilającego. Niektóre elementy są umieszczone poza płytkami i mocowane do obudowy: filtr przeciwzakłóceniowy, odbiornik podczerwieni, gniazda bezpieczników oraz diody sygnalizacyjne. Schemat ich połączeń z płytkami sterownika pokazano na **rysunku 3**.

Moduł sygnałowy spełnia rolę kontrolera obsługującego sterownik. W jego skład wchodzi pamięć typu EN25F10, zegar czasu rzeczywistego RTC4513, bateria podtrzymująca zasilanie, mikrokontroler ATmega168AU, wyświetlacz LCD (2×16 znaków, zasilany napięciem 3,3 V).

Schemat ideowy modułu sygnałowego pokazano na **rysunku 4**. Zamontowano w nim dwa potencjometry. Potencjometr R1 służy do ustawienia kontrastu, natomiast R2 do ustawienia intensywności podświetlenia wyświetlacza LCD. Schemat montażowy modułu sygnałowego pokazano na **rysunku 5**. Przycisk S3 można podłączyć za pomocą przewodów i umieścić go na obudowie sterownika.

Zadaniem modułu sterująco–zasilającego jest zasilanie całego sterownika oraz sterowanie załączaniem poszczególnych triaków. Zasilacz zbudowano z użyciem transformatora i liniowych stabilizatorów napięcia. Na obwód załączania napięcia składają się optotriak oraz triak. Gwarantuje to właściwy poziom izolacji galwanicznej pomiędzy napięciem sieciowym a niskonapięciowymi obwodami mikrokontrolera. Schemat ideowy modułu sterująco-zasilającego zamieszczono na **rysunku 6**, natomiast montażowy na **rysunku 7**.

Do zaprogramowania mikrokontrolera używany jest programator ISP, jednak dla zmniejszenia wielkości płytki zmieniono jego gniazdo ze standardowego IDC na *Crimp Terminal* o rastrze 1,5 mm. Rozmieszczenie sygnałów na gnieździe programatora pokazano na **rysunku 8**. Ze względu na fakt, że te same linie wykorzystywane są do komuniko-

Wykaz elementów

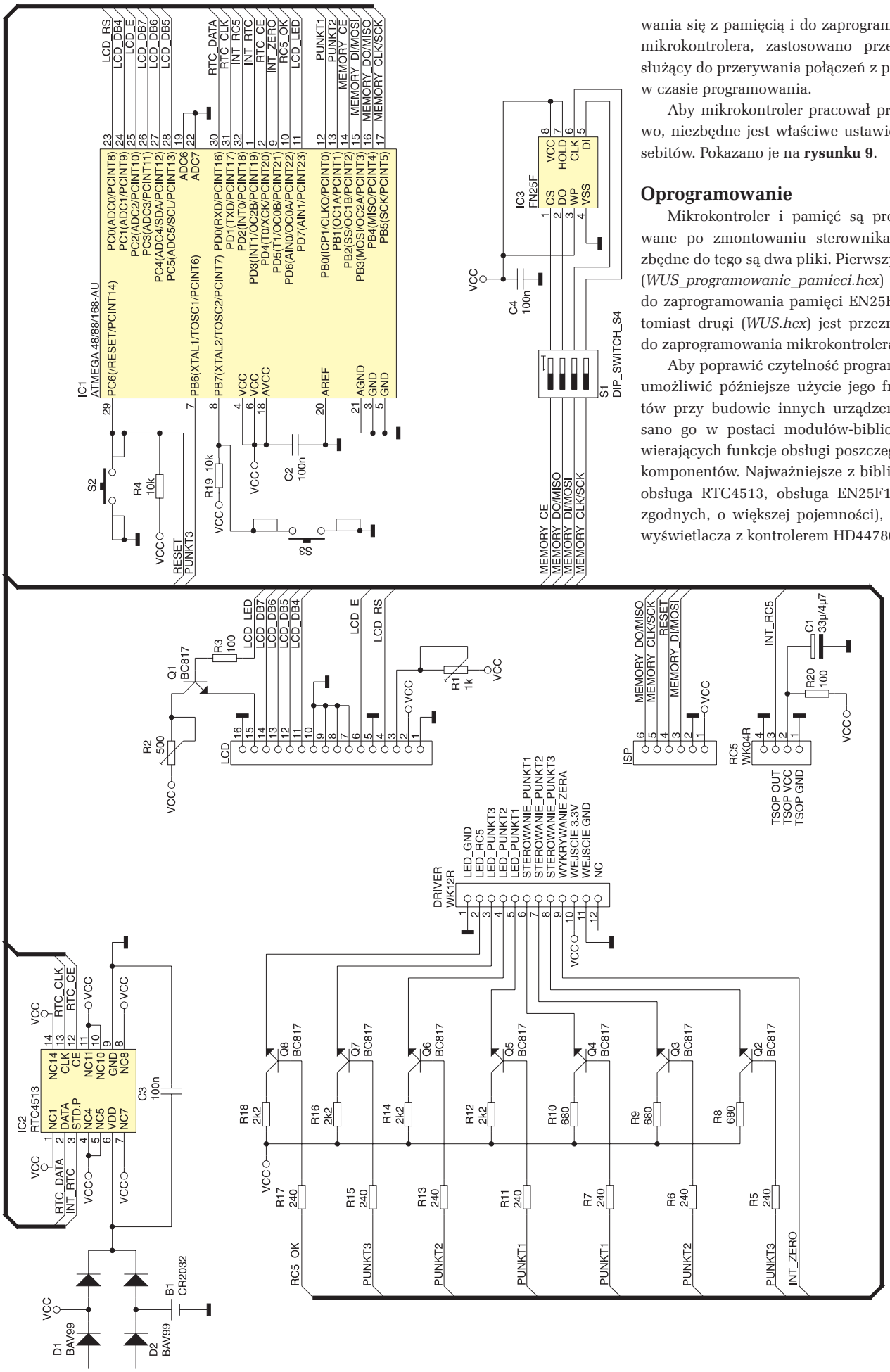
Moduł sygnałowy

Rezystory:
R1: 1 kΩ (potencjometr POT43P)
R2: 500 Ω (potencjometr POT43P)
R3, R20: 100 Ω (SMD 0805)
R4, R19: 10 kΩ (SMD 0805)
R5...R7, R11, R13, R15, R17: 240 Ω (SMD 0805)
R8...R10: 680 Ω (SMD 0805)
R12, R14, R16, R18: 2,2 kΩ (SMD 0805)
Kondensatory:
C1: 4,7 µF (SMD 4×5,8)
C2...C4: 100 nF (SMD 0805)
Półprzewodniki:
D1, D2: BAV99
IC1: ATmega168AU
IC2: RTC4513
IC3: EN25F10
Q1...Q8: BC817
Inne:
S1: DIPS-HD04 (DIP Switch, raster 1,27 mm, 4-biegowy)
S2, S3: Tact Switch (6,0×6,0 mm)
RC5: WK04R (złącze męskie Crimp Terminal, raster 1,5 mm, 4 styki)
LCD: PB16S-LF (listwa stykowa żeńska, raster 2,54 mm, 1×16 znaków)
ISP: WK06R (złącze męskie Crimp Terminal, raster 1,5 mm, 6 styków)
DRIVER: WK12R (złącze męskie Crimp Terminal, raster 1,5 mm, 12 styków)
B1: gniazdo baterii CR2032

Moduł sterująco-zasilający

Rezystory:
R9: 4,7 kΩ (SMD 0805)
R7, R8: 100 kΩ (THT, moc 2 W)
R1...R6: 390 Ω (THT, moc 2 W)
Kondensatory:
C4, C6: 1000 µF/16 V
C1: 1500 µF/16 V
C2, C3, C5, C7: 100 nF (SMD 0805)
Półprzewodniki:
D1: S2J (SMD)
IC1: LM1117-5.0 (TO252)
IC2: LM1117-3.3 (TO252)
OK4: LTV814
T1...T3: BTA24-800BW (TO220AB)
TSOP34836 - odbiornik podczerwieni
Inne:
TR1: transformator TSZZ6/003M (np. firmy Indel)
CON1: WK06R (złącze męskie Crimp Terminal, raster 1,5 mm, 6 styków)
L1, L2, N1, N2, N3, PUNKT1, PUNKT2, PUNKT3: złącza 2×ARK3, 1×ARK2
HK04 (złącza żeńskie Crimp Terminal, raster 1,5 mm, 4 styki)
HK06 (złącza żeńskie Crimp Terminal, raster 1,5 mm, 6 styków)
HK12 (złącza żeńskie Crimp Terminal, raster 1,5 mm, 12 styków)
FYK10T1 - filtr przeciwzakłóceniowy
Gniazdo dla bezpiecznika 5×20 mm – 2 szt.

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym



Rysunek 4. Schemat ideowy modułu sygnałowego

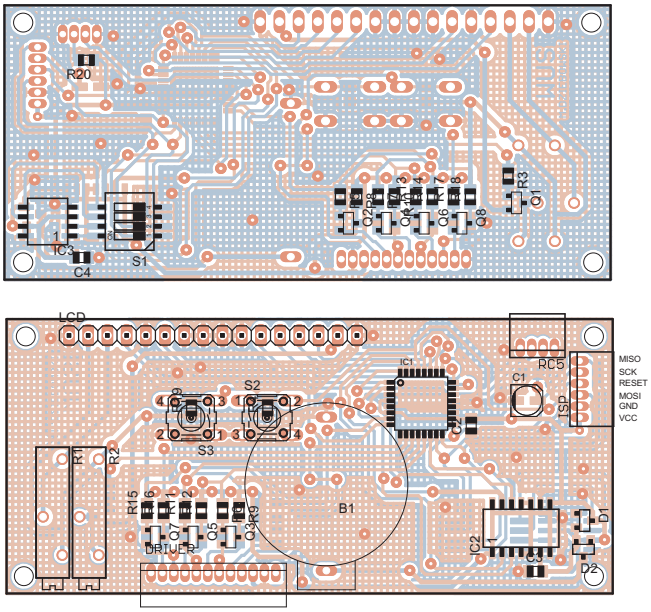
wania się z pamięcią i do zaprogramowania mikrokontrolera, zastosowano przełącznik służący do przerywania połączeń z pamięcią w czasie programowania.

Aby mikrokontroler pracował prawidłowo, niezbędne jest właściwe ustawienie fusebitów. Pokazano je na **rysunku 9**.

Oprogramowanie

Mikrokontroler i pamięć są programowane po zmontowaniu sterownika. Niezbędne do tego są dwa pliki. Pierwszy z nich (*WUS_programowanie_pamieci.hex*) służy do zaprogramowania pamięci EN25F10, natomiast drugi (*WUS.hex*) jest przeznaczony do zaprogramowania mikrokontrolera.

Aby poprawić czytelność programu oraz umożliwić późniejsze użycie jego fragmentów przy budowie innych urządzeń, napisano go w postaci modułów-bibliotek zawierających funkcje obsługi poszczególnych komponentów. Najważniejsze z bibliotek to: obsługa RTC4513, obsługa EN25F10 (oraz zgodnych, o większej pojemności), obsługa wyświetlacza z kontrolerem HD44780.



Rysunek 5. Schemat montażowy modułu sygnałowego

Obsługa RTC4513. Programowa obsługa układu scalonego RTC4513 nie jest kompatybilna z żadnym z popularnych układów scalonych. Z tego powodu przygotowano bibliotekę zawierającą funkcje obsługi układu zegara oraz programową realizację interfejsu sprzętowego. W bibliotece zdefiniowano wyprowadzenia mikrokontrolera, do których jest podłączany RTC. Do obsługi układu wystarczające są następujące funkcje:

– *void RTC_INIT(void)* - inicjalizacja układu, konfigurowanie portów mikrokontrolera,
– *void RTC_READ_TIME(unsigned char *data), void RTC_WRITE_TIME(unsigned char *data)* – funkcje odczytu aktualnej godziny oraz zapisu jej nastawy.

Ponadto, biblioteka zawiera obsługę nastawy i odczytu daty, jednak w sterowniku zrezygnowano z jej obsługi. Zaimplementowano jedynie obsługę dni tygodnia.

Obsługa EN25F10. Pamięć ta ma interfejs SPI. Wykorzystywane są następujące funkcje obsługi pamięci:

– *void MEMORY_CHECK_ID(unsigned char *data)* – odczyt identyfikatora pamięc,.
– *void MEMORY_WE(void)* – ustawienie *write eneable* (zezwolenie zapisu) umożliwiającego zmianę zawartości pamięci,
– *unsigned char MEMORY_CHECK_WEL(void)* – funkcja zwraca „1”, jeśli ustawiony jest bit pozwalający na operacje zapisu/kasowania pamięci,
– *unsigned char MEMORY_CHECK_WIP(void)* – funkcja zwraca „1” jeśli kontroler pamięci wykonuje operacje zapisu/kasowania,
– *unsigned char MEMORY_CHECK_STATUS(void)* – funkcja zwraca wartość bajta statusu,
– *void MEMORY_INIT(void)* – inicjalizacja pamięci, ustawienie parametrów SPI,
– *void MEMORY_WRITE(unsigned char address, unsigned char *data, unsigned char count), void MEMORY_READ(unsigned char address, unsigned char *data, unsigned char count)* – zapis/odczyt danych z pamięci spod wskazanego adresu,
– *void MEMORY_ERASE(void)* – kasowanie zawartości całej pamięci,
– *void MEMORY_ERASE_SECTOR(unsigned char address)* – kasowanie zawartości pojedynczego sektora pamięci.

Dekodowanie kodu RC5. Programowe dekodowanie kodu RC5 składa się z dwóch części: inicjalizacji timera i przerwania zewnętrznego oraz określenie sposobu reagowania na zbocza. W programie zostały zdefiniowane następujące polecenia konfigurujące przerwania:

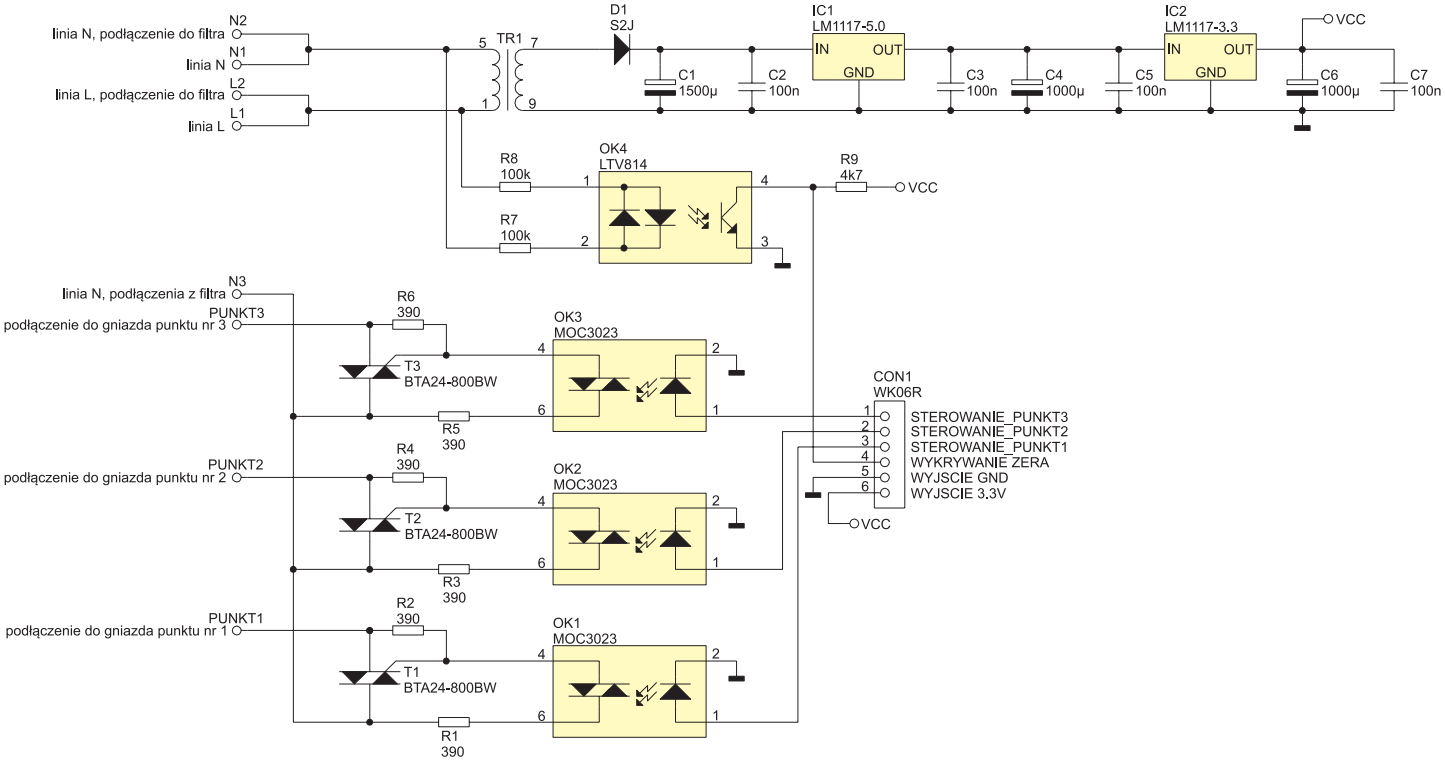
– *#define INT0_ZO {EICRA&=0xFC;EICRA|=0x02;}* //ustawienie wywołania int0 – zbocze narastające;
– *#define INT0_ZN {EICRA&=0xFC;EICRA|=0x03;}* //ustawienie wywołania int0 – zbocze opadające;
– *#define INT0_OFF {EIMSK &= 0xFE;}* // wyłączenie obsługi przerwania int0;
– *#define INT0_ON {EIMSK |= 0x01;}* // włączenie obsługi przerwania int0.

Wstępne skonfigurowanie przerwania jest następujące:

– *INT0_ZO;* //wywołanie przerwania poprzez zbocze opadające;
– *INT0_ON;* //włączenie obsługi przerwania int0.

Konfigurowanie timera wykorzystywanego do pomiaru czasu stanów kodu RC5 przebiega następująco:

TCCR2A = 0x00;



Rysunek 6. Schemat ideowy modułu sterująco-zasilajacego

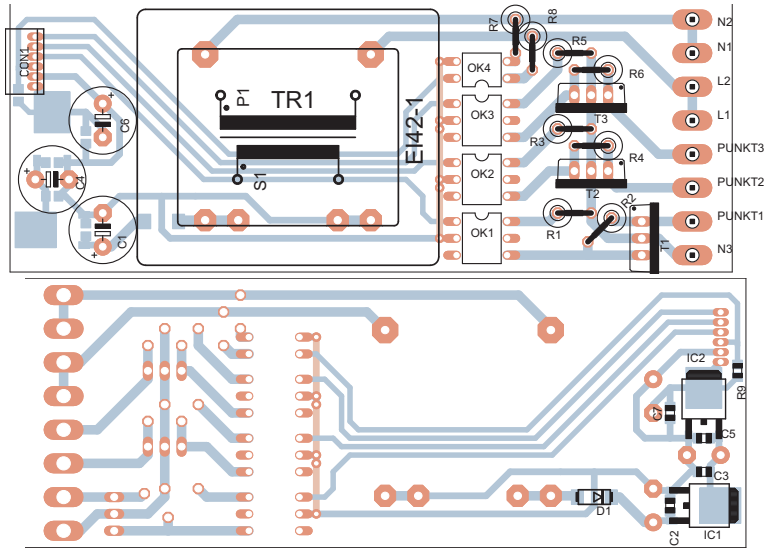
```
TCCR2B = 0x06; //preskaler 256 dla 8MHz
TCNT2 = 0;
```

Natomiast dekodowanie kodu RC5 odbywa się z użyciem przerwań generowanych przez INT0 oraz przez timer. Program dekodujący zamieszczono na **listingu 1**.

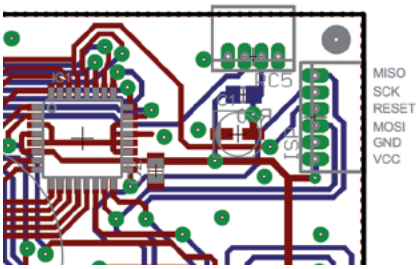
Listing 1. Program dekodujący kod RC5

```
ISR(INT0_vect){
        rc5_tcnt_temp = TCNT_RC5; //wykonanie kopii zawartości timera do późniejszej analizy
        TCNT_RC5 = 0; //synchronizacja timera
//zmiana sygnału aktywującego obsługę przerwania
        if(INT0_STAN == 1) {INT0_ZO;} else {INT0_ZN;}
/* jeśli jest to pierwszy impuls transmisji sygnału to następuje ustawienie zmiennych oraz włączenie zabezpieczenia
zbyt długiego trwania danego stanu */
        if(rc5_i == 0)
        {
                rc5_i = 1;
                rc5_kod[0] = 0;
                rc5_kod[1] = 0;
                rc5_kod[2] = 0;
                rc5_kod[3] = 0;
                OCR2A = czas_bitu_bezpieczenstwa2;
                TIFR2 = 0x02; TIMSK2 = 0x02;
        }
        else
        {
/* aktualizacja stanu w którym sie znajduje dekodowanie rc5 wraz ze sprawdzaniem czy czas trwania impulsu jest poprawny
*/
                if((rc5_tcnt_temp > (czas_bitu_pol - czas_bitu_tolerancja)) && (rc5_tcnt_temp < (czas_bitu_pol + czas_
bitu_tolerancja))) {rc5_i += 1;}
                else if((rc5_tcnt_temp > (czas_bitu_caly - czas_bitu_tolerancja)) && (rc5_tcnt_temp < (czas_bitu_caly
+ czas_bitu_tolerancja))) {rc5_i += 2;}
/* w przypadku błędnego czasu trwania stanu niskiego lub wysokiego następuje wyzerowanie zmiennych oraz zablokowanie
dekodowania sygnału na okresłony czas */
                else
                {
                        rc5_i = 0;
                        rc5_kod[0] = 0;
                        rc5_kod[1] = 0;
                        rc5_kod[2] = 0;
                        rc5_kod[3] = 0;
                        INT0_OFF;
                        OCR2A = 255;
                        TIFR2 = 0x02;
                        TIMSK2 = 0x02;
                }
        }
        _delay_us(10); //niewielkie opóźnienie by nie sprawdzać stanu zaraz po wystąpieniu zbocza
        switch(rc5_i) {
                case 27 : //ostatnie próbkowanie po zboczu synchronizacji
                        if(INT0_STAN == 0) {rc5_kod[1] |= 0x01;}
        //wyłączenie odbioru transmisji
                        rc5_i = 0;
                        INT0_OFF;
                        OCR2A = 255;
                        TIFR2 = 0x02;
                        TIMSK2 = 0x02;
//sprawdzenie czy aktualnie transmitowany kod jest przyciśniętym nowym przyciskiem
                        if(rc5_kod[2] != rc5_toggle)
                        {
                                rc5_toggle = rc5_kod[2];
                                rc5_kod[3] = 1;
                        }
//przypisanie odpowiednich wartości poszczególnym przyciskom
                        CONVERT_RC5(rc5_kod,rc5_klawisz,configuration);
/* jeśli klawisz jest poprawny następuje włączenie diody sygnalizacyjnej informującej o odebraniu sygnału kodu RC5 */
                        if(rc5_klawisz[1] == 1)
                        {
                                RC5_LED_ON;
                                LCD_LED_ON; licznik_diody_rc5 = 0;
                                urzadzenie_zablokowane_licznik = 0;
                        }
                        break;
//próbkowanie stanów chwile po zboczu synchronizacji
                case 25: if(INT0_STAN == 0){rc5_kod[1] |= 0x02;} break;
                case 23: if(INT0_STAN == 0){rc5_kod[1] |= 0x04;} break;
                case 21: if(INT0_STAN == 0){rc5_kod[1] |= 0x08;} break;
                case 19: if(INT0_STAN == 0){rc5_kod[1] |= 0x10;} break;
                case 17: if(INT0_STAN == 0){rc5_kod[1] |= 0x20;} break;
                case 15: if(INT0_STAN == 0){rc5_kod[0] |= 0x01;} break;
                case 13: if(INT0_STAN == 0){rc5_kod[0] |= 0x02;} break;
                case 11: if(INT0_STAN == 0){rc5_kod[0] |= 0x04;} break;
                case 9: if(INT0_STAN == 0){rc5_kod[0] |= 0x08;} break;
                case 7: if(INT0_STAN == 0){rc5_kod[0] |= 0x10;} break;
                case 5: if(INT0_STAN == 0){rc5_kod[2] |= 0x01;} break;
                default: break;
        }
}


ISR(TIMER2_COMPA_vect){
//wyłączenie obsługi transmisji
        INT0_OFF; rc5_licznik_odstepu += 1;
        if(rc5_licznik_odstepu >= 7)
        {
                rc5_licznik_odstepu = 0;
                TCNT_RC5 = 0;
                TIMSK2 = 0x00;
                INT0_ZO;
                INT0_ON;
                rc5_i = 0;
        }
}
```

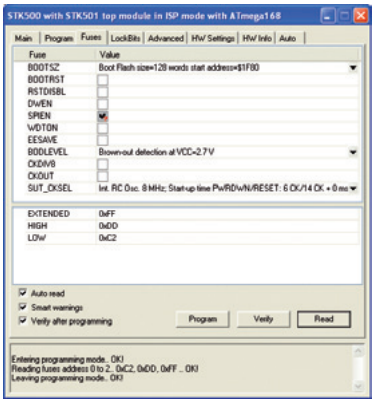


Rysunek 7. Schemat montażowy modułu sterująco-zasilającego



Rysunek 8. Rozmieszczenie sygnałów na gnieździe programatora

W czasie obsługi przerwania wywołanego zboczem jest sprawdzany stan bitu. Następuje to po wystąpieniu zbocza synchronizacji. Do identyfikacji czy przerwanie zostało wygenerowane przez zbocze synchronizujące służy zmienna *rc5_i*. Identyfikuje ona aktualną pozycję w kodzie RC5, numerowane są poszczególne połówki bitu poczynając od drugiej polowy pierwszego bitu (dla przykładu wartość 5 oznacza drugą połowę bitu Toggle – bit 3, natomiast wartość 10 oznacza pierwszą część bitu A2).



Rysunek 9. Ustawienie fusebitów mikrokontrolera ATmega168

W przypadku błędnego kodu (np. niebędącego kodem RC5) następuje wyłączenie obsługi odbioru kodu RC5. Wyłączenie ma także miejsce po poprawnym odebraniu kodu.

Załączanie odbiorników

Sterowanie włączaniem/wyłączaniem zasilania poszczególnych odbiorników odbywa się wskutek wysterowania triaków. Wykorzystano do tego celu programowo generowany sygnał PWM, z synchronizacją sygnału w momencie przejścia napięcia sieci przez zero.

Mariusz Dziębowski
m.dziebowski@gmail.com

R E K L A M A