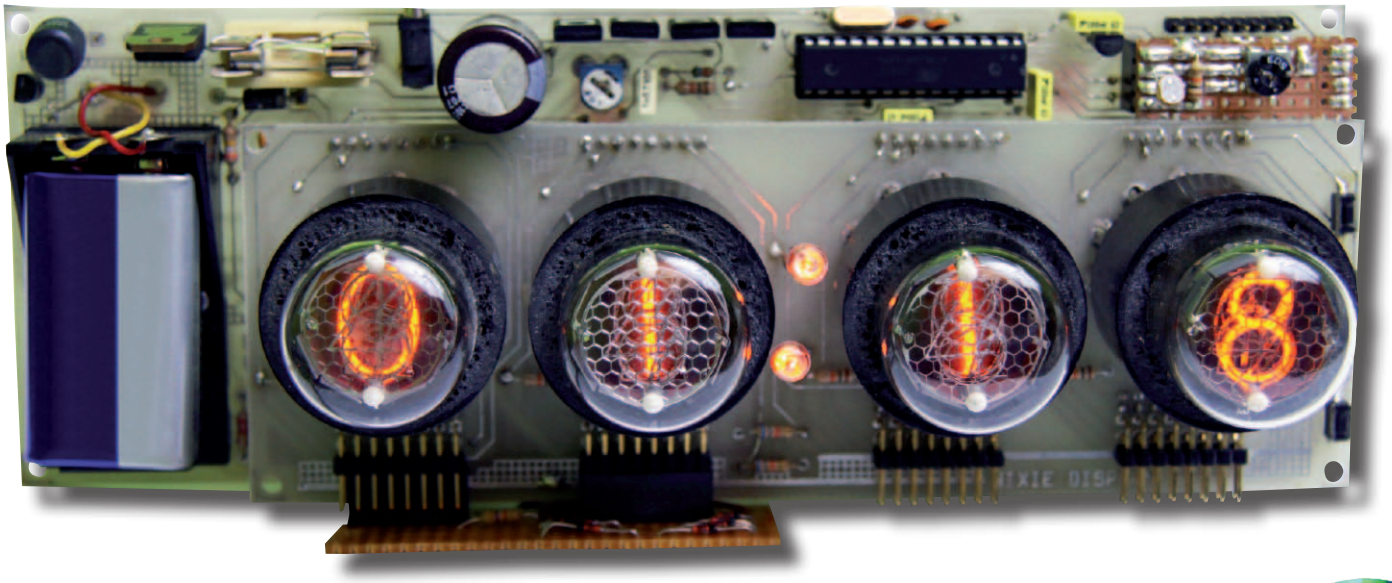


Zegar w stylu retro

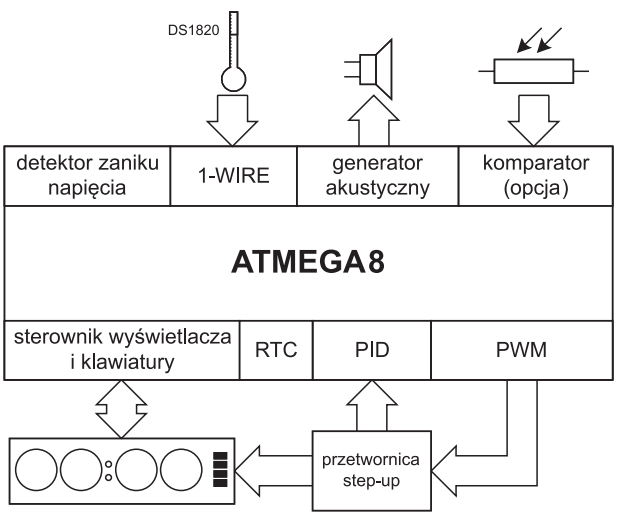
Zegarki w stylu retro, budowane w oparciu o wyświetlacze Nixie są popularnym urządzeniem budowanym przez elektroników hobbystów. Niespotykane już w komercyjnych rozwiązaniach wyświetlacze, w połączeniu z elegancką obudową, wyglądają bardzo estetycznie i nie są dostępne dla przeciętnych konsumentów na rynku masowej elektroniki.

Schemat blokowy zegara umieszczono na **rys. 1**. Najważniejsze cechy funkcjonalne i rozwiązania układowe mojego urządzenia to:

- Funkcja budzenia melodiami wielotonowymi (5 melodii).
- Baterijne podtrzymanie zasilania: budzik działa nawet po wyłączeniu zasilania.
- 2 tryby wyświetlania czasu: „dzień”, w którym jest podwyższona jasność cyfr na wyświetlaczach, miga znak „:” oraz „noc”, w którym jest zmniejszona jasność świecenia, a znak „:” gaszony; o wyborze trybu decyduje czujnik natężenia światła zewnętrznego.
- Termometr.
- Kalendarz.
- Możliwość połączenia płytki wyświetlacza i sterującej zegara na dwa sposoby: równolegle i prostopadle do siebie.

Urządzenie jest zasilane z pojedynczego źródła o napięciu 12 V. Napięcie do zasilania lamp jest wypracowywane przez przetwornicę step-up sterowaną mikrokontrolerem.

Dodatkowym atutem jest fakt, że całe urządzenie zostało zaprojektowane z wykorzystaniem narzędzi dostępnych za darmo z użyciem tzw. wolnego oprogramowania. Do wykonania schematów ideowych i projektów PCB użyłem programu KiCad [1], który staje się coraz bardziej popularny wśród elektroników hobbystów. Do stworzenia programu dla mikrokontrolera użyłem kompilatora AVR-GCC [2] wraz z niezbędną biblioteką avr-libc [3], a jako narzędzia do edycji – edytora VIM [4]. Do zaprogramowania mikrokontrolera użyłem AVRDUDE [5]. Całość pracuje pod kontrolą darmowego systemu operacyjnego Debian [6], który zawiera w sobie wszystkie wymienione pakiety i stanowi doskonałą platformę do tego typu zadań



Rys. 1. Schemat blokowy (funkcjonalny) zegara

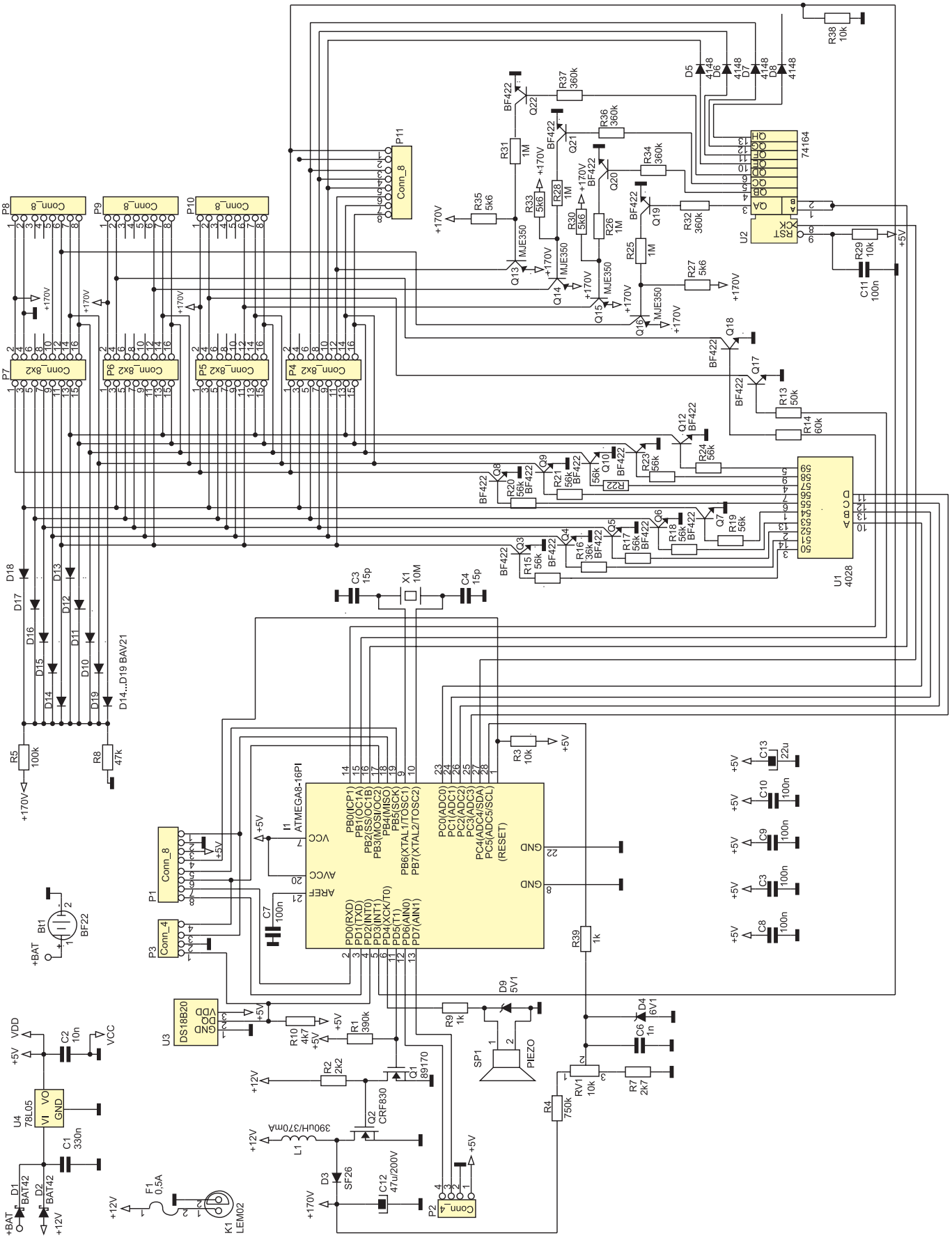
Schemat Elektryczny

Na **rys. 2** przedstawiono schemat płytki głównej zegara. Najważniejszym elementem układu jest mikrokontroler I1. Za pośrednictwem U1 steruje tranzystorami Q3...Q12, które służą do załączania równolegle połączonych katod wyświetlaczy Nixie pracujących w trybie multipleksowania. Układ U2 pośredniczy w wybieraniu wyświetlacza przy multipleksowaniu oraz przy podczas wybierania przycisku, którego stan ma być odczytany. Q13...Q16 to elementy wykonawcze układu sterowania anodami, a tranzystory Q19...Q22 mają za zadanie zapewnić odpowiedni potencjał gwarantujący nasycenie tranzystorów wykonawczych. Diody D10...D19 wraz z rezystorami R5, R6 służą do zapewnienia różnicy potencjałów na aktualnie niewybranych lampach poniżej napięcia zapłonu (w prototypie wysłanym do redakcji diody są umieszczone na dodatkowej płytce).

Tranzystor Q2 i dioda D3 to klucze klasycznej przetwornicy *step-up*, której energia gromadzona jest w dławiku L1 i kondensatorze C12. Q1 służy do wysterowania odpowiednio wysokim napięciem bramki Q2. Dzielnik napięcia zbudowany z R4, RV1, R7 dopasowuje napięcie z przetwornicy do poziomu akceptowanego przez wbudowany w ATmega8 przetwornik analogowo-cyfrowy. Q17 i Q18 sterują neonówkami, służącymi do wyświetlania znaku „:”. Złącza P2, P3 tworzą gniazdo rozszerzeń służące do dołą-

czenia dodatkowych modułów współpracujących z zegarem. Jako przykład takiego modułu zbudowałem czujnik natężenia światła, wpływający na jasność świecenia wyświetlaczy oraz na miganie znaku „:", dzięki czemu zmniejszone natężenie świecenia zegara nie przeszkadza wieczorami np.: podczas oglądania filmu czy snu.

P1 to gniazdo serwisowe z wyprowadzonymi sygnałami programującymi oraz portu szeregowego, ułatwiające uruchamianie programu mikrokontrolera. Gniazda P4...P11



Rys. 2. Schemat ideowy płytki głównej zegara

służą do podłączenia płytki wyświetlacza. Jej schemat pokazano na **rys. 3**.

Schemat opcjonalnego modułu czujnika oświetlenia pokazano na **rys. 4**. Wykorzystuje on wbudowany w ATmega8 komparator. Na jedno z jego wejść podawane jest napięcie z dzielnika zbudowanego na potencjometrze montażowym PV1, służące do konfiguracji poziomu załączania, a na drugie sygnał z fototranzystora. Prostota konstrukcji, a w szczególności brak filtrowania zakłóceń przy przejściu przez próg zadziałania, wymaga odpowiedniej obsługi programowej czujnika.

Dodatkowo na **rys. 5** zamieszczam schemat programatora wraz z konwerterem poziomów napięć z TTL na RS232. Łącze szeregowe nie jest potrzebne do normalnej pracy zegara (służy do odczytu informacji serwisowych), dlatego też budowa takiego programatora jest również opcjonalna.

Program

Ze względu na stosunkowo prosty hardware, kod programu jest dość rozbudowany.

PWM. Ten blok realizuje zadanie wytwarzania przebiegu prostokątnego o zadanym wypełnieniu do sterowania przetwornicą *step-up*. Do jego realizacji użyłem Timer2 w trybie *Fast PWM*. Timer2 ma wyłączony preskaler, co oznacza, że generowany przebieg powinien mieć okres równy 16 µs, jednak ze względu na celowo wprowadzone opóźnienie, okres rzeczywistego przebiegu jest o około 2 µs dłuższy.

PID. Zadaniem tej funkcjonalności jest obliczenie najkorzystniejszego w danej chwili wypełnienia przebiegu sterującego przetwornicą na podstawie odczytanego napięcia z przetwornika A/C. Do obliczenia skorzystałem z prostej, wyznaczonej eksperymentalnie reguły matematycznej będącej uproszczoną implementacją sterowania PID.

$$D = 16U_e + \frac{1}{512}\int\left(\frac{U_e}{4}\right)dt + 2\frac{dU_e}{dt}$$

gdzie:

Wykaz elementów

Płytka główna zegara

Rezystory:
R1: 390 kΩ
R2: 2,2 kΩ
R3, R29, R38: 10 kΩ
R4: 750 kΩ
R5: 100 kΩ
R6: 47 kΩ
R7: 2,7 Ω
R9, R39: 1 kΩ
R10: 4,7 Ω
R13, R14: 50 kΩ
R15...R24: 56 kΩ
R25, R26, R28, R31: 1 MΩ
R27, R30, R33, R35: 5,6 kΩ
R32, R34, R36, R37: 360 kΩ
RV1: potencjometr montażowy 10 kΩ

Kondensatory:
C1: 330 nF
C2: 10 nF
C3, C7...C11: 100 nF
C4, C5: 15 pF
C6: 1 nF
C12: 47 µ/200 V
C13: 22 µF

Półprzewodniki:
D1, D2: BAT42
D3: SF26
D4: 5V1
D5...D8: 1N4148
D9: 5V1
D10...D19: BAV21
I1: ATmega8-16PI
U1: 4028
U2: 74164
U3: DS18B20
U4: 78L05
Q1: BS170
Q2: IRF830
Q3...Q12, Q17...Q22: BF422
Q13...Q16: MJE350

Inne:
BT1: gniazdo na baterię+bateria 6F22
F1: gniazdo bezpiecznika + bezpiecznik 0,5 A
K1: gniazdo zasilania męskie, kołek ϕ1,3 mm, otwór ϕ4,5 mm
L1: 390 µH/370 mA
P1, P8...P11: PB8 (męskie)
P2, P3: PB4 (męskie)
P4...P7: PB8×2 (żeńskie)
SP1: głośnik PIEZO
X1: kwarc 16M

Płytka wyświetlacza

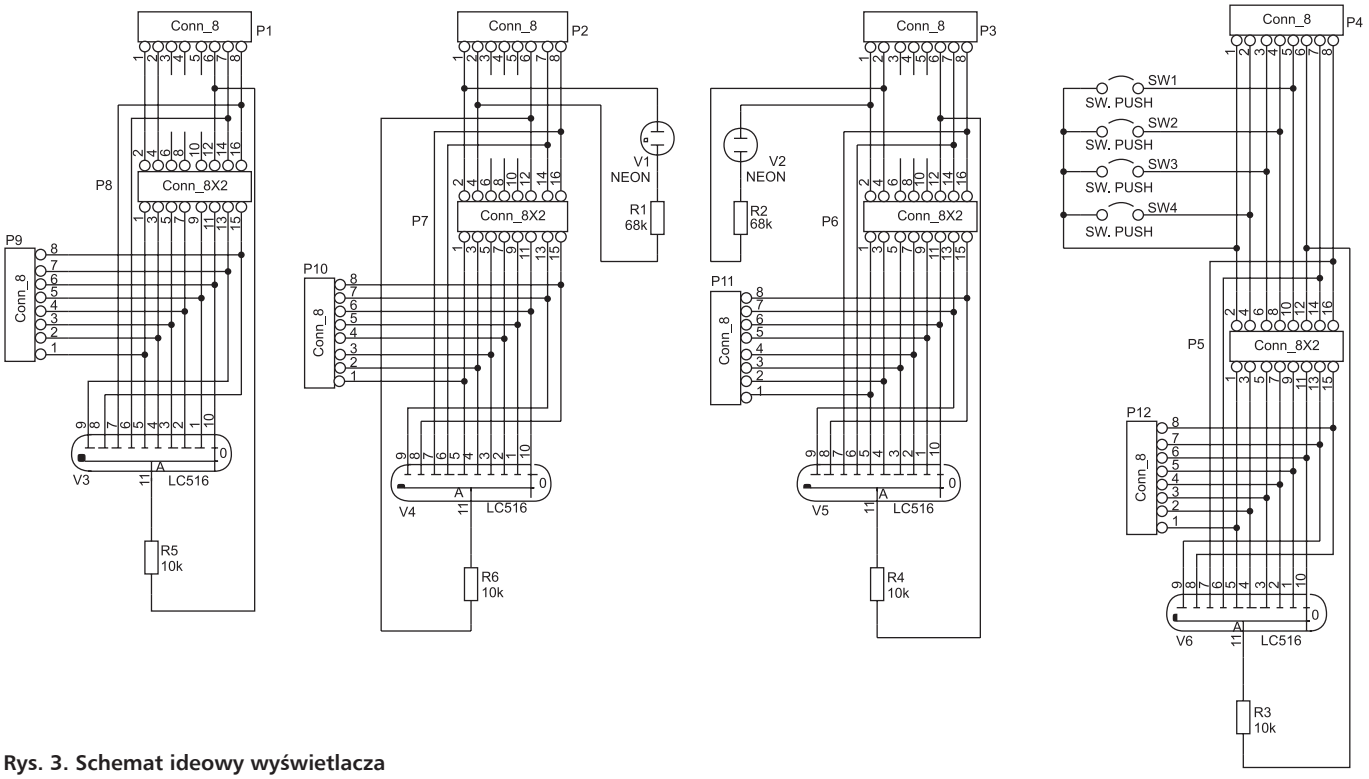
P1...P4, P9...P12: PB8 żeńskie
P5...P8: PB8×2 żeńskie, kątowe
R1, R2: 68 kΩ
R3, R6: 10 kΩ
SW1...SW4: microswitch
V1, V2: neonówki
V3...V6: LC516 lampy Nixie

Czujnik światła

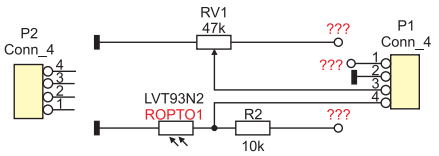
P1, P2: PB4 żeńskie
R2: 10 kΩ
ROPTO1: fotorezystor LVT93N2
RV1: potencjometr montażowy 47 kΩ

Programator

C1...C4: 1 µF
C5: 100 nF
D1: 1N4148
J1: DB25
P1: PB8 żeńskie
P2: PB3 żeńskie
R1: 100
U1: MAX232



Rys. 3. Schemat ideowy wyświetlacza

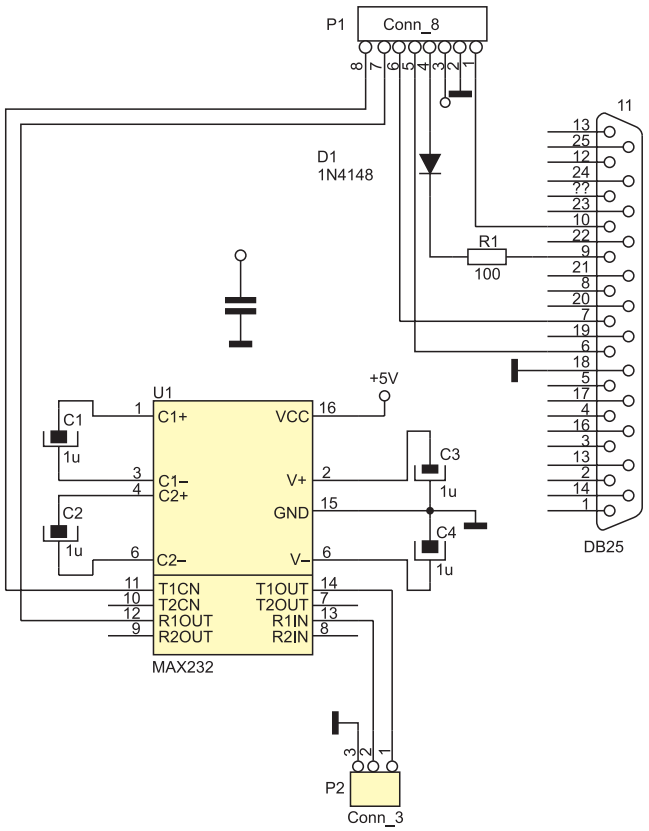


Rys. 4. Schemat ideowy czujnika światła

U_e – czynnik proporcjonalny do wartości napięcia uchybu, liczony jako różnica wartości zadanej, a odczytanej z przetwornika A/C. Całka liczona jest jako suma wszystkich wartości U_e od startu programu, a pochodna, to po prostu różnica między poprzednią a aktualną wartością U_e. Wszystko liczone jest w kontekście przerwania od przetwornika A/C.

Sterowanie przetwornicami najczęściej buduje się w oparciu o specjalizowane układy scalone. Programowa realizacja tego zadania na ATmedze8 nie pozwala na szybką reakcję, jednak w przypadku tego konkretnego układu zasilania nie jest to problemem. Wymagania, co do napięcia zasilania wyświetlaczy Nixie nie są tak krytyczne. Zalety takiego podejścia to niejako wbudowana możliwość regulacji zadanego napięcia wyjściowego oraz dobra zabawa związana z eksperymentowaniem przy realizacji.

RTC. To funkcjonalność odpowiedzialna za odmierzanie aktualnej daty i czasu. Do odmierzania cykli użyłem przerwania z przetwornika A/C użytego przy budowie bloku PID. Jak podaje dokumentacja mikrokontrolera, dla ciągłego trybu pracy, częstość jego pojawiania się wynosi 13 cykli sygnału zegarowego dostarczanego do przetwornika. Ja dodatkowo użyłem preskalera zaprogramowanego na podział /128. Niestety, dla kwarcu 16 MHz taki współczynnik podziału (13×128) nie daje w wyniku wartości całkowitej. Problem ten obszedłem poprzez zliczanie kumulującego się błędu (reszty z dzielenia całkowitoliczbowego) i na tej podstawie wprowadzaniu odpowiednich korekt.



Rys. 5. Schemat ideowy programatora

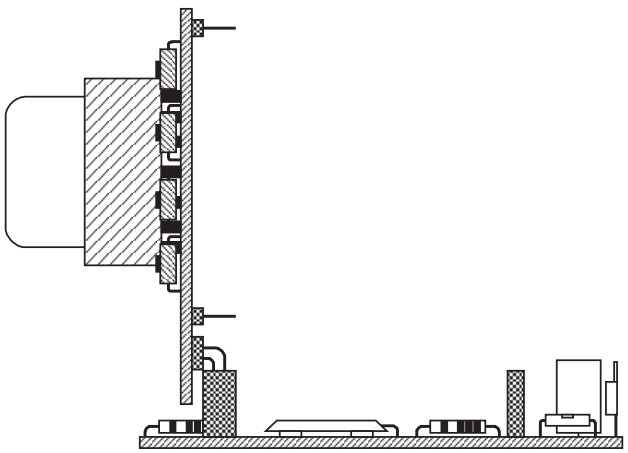
Sterownik wyświetlacza i klawiatury. Zadanie modułu polega na generacji odpowiednich sygnałów sterujących do realizacji multipleksowego wyświetlania oraz odczytu stanu przycisków. To zadanie również wykonywane jest w kontekście przerwania ADC.

1-wire. Blok ma za zadanie skomunikować się ze scalonym termometrem DS1820. Ze względu na okna czasowe w specyfikacji tego interfejsu, jego implementacja była sporym wyzwaniem. Dokładne odliczanie czasu z rozdzielczością do µs powinno się odbywać przy wyłączonych przerwaniach jednak ze względu na architekturę systemu niewskazane jest, aby blokować przerwania na zbyt długo. Należało, więc minimalizować momenty, w których przerwania będą blokowane i jednocześnie zachować akceptowalny poziom liczby błędnych odczytów. Kod wykonywany jest w pętli głównej.

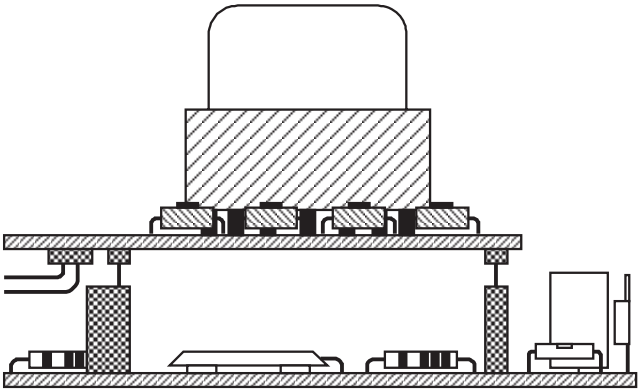
Generator akustyczny. Funkcjonalność realizowana w kontekście przerwania PWM. Ze względu na niską rozdzielczość impulsów PWM częstotliwości generowanych kolejnych półtonów obarczone są błędem rosnącym wraz z wysokością dźwięku. Konstrukcja programu umożliwia stosunkowo łatwą rozbudowę o nowe melodie – nuty zapisywane są w postaci list inicjalizujących tablice.

Komparator. Jest on potrzebny tylko w sytuacji, gdy zegar będzie wyposażony w opcjonalny, progowy czujnik oświetlenia. Wykorzystuje wbudowany w mikrokontroler komparator do określenia jasności świecenia wyświetlacza. Podstawową trudnością w realizacji był sposób zmniejszania jasności. Ostatecznie zdecydowałem się na wprowadzenie przerw pomiędzy okresami świecenia kolejnych lamp (sterowanie jasnością poprzez współczynnik wypełnienia) i dodatkowo lekkie obniżenie napięcia zasilania (zbyt niska wartość, poniżej napięcia zapłonu nie pozwola na poprawne świecenie lamp). Ze względu na brak histerezy w układzie komparatora, odczyt wyniku porównania napięcia z fotorezystora z wartością odniesienia z potencjometru uznawany jest za ważny, gdy kilka tysięcy odczytów pod rząd daje ten sam wynik.

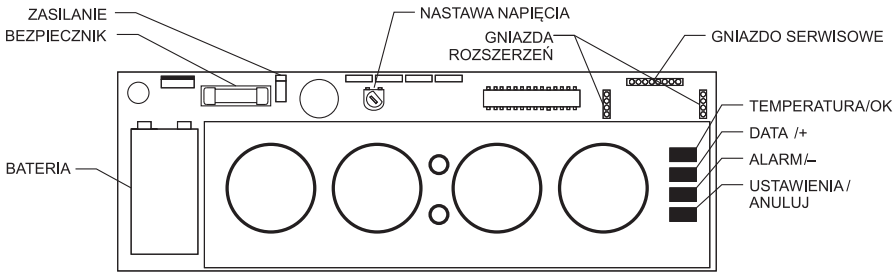
Detektor zaniku napięcia. funkcjonalność realizowana w kontekście ADC i korzystająca z odczytów ADC. Wykrycie zaniku napięcia powoduje przejście w tryb oszczędzania energii. Działają wtedy tylko 2 funkcję RTC i Generator akustyczny.



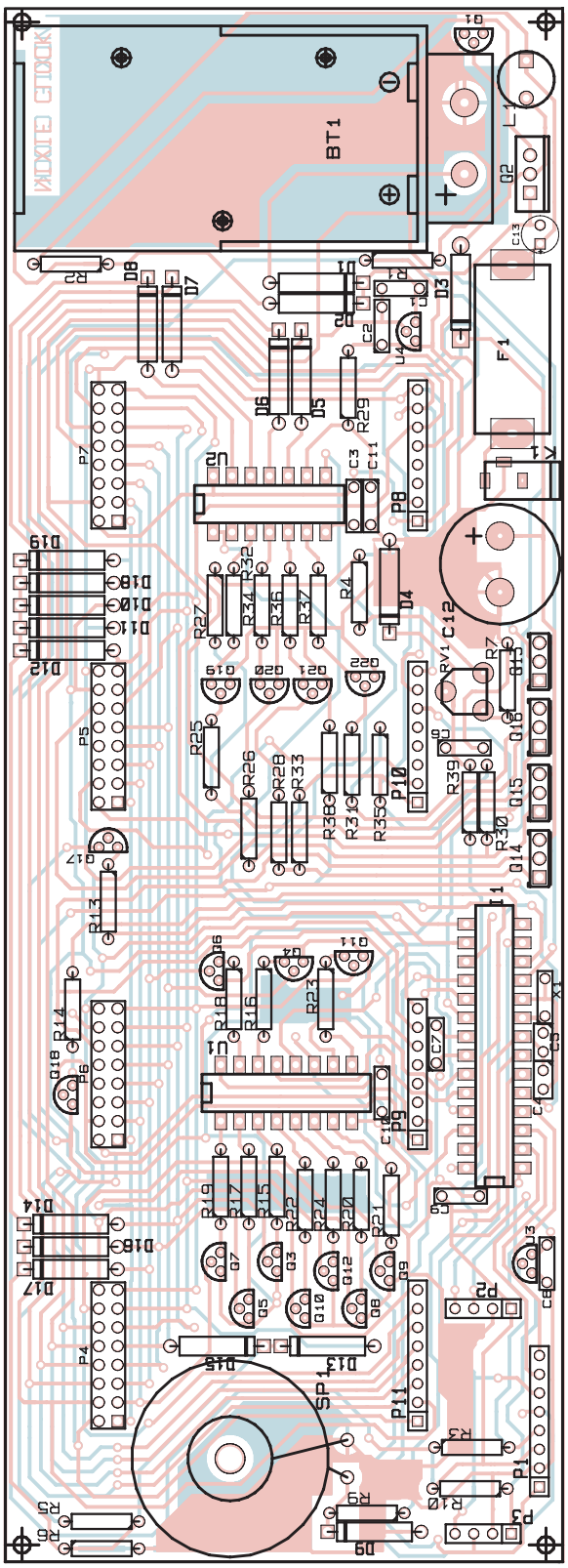
Rys. 6. Płytki sterująca i wyświetlacza połączone prostopadle do siebie



Rys. 7. Płytki sterująca i wyświetlacza połączone równolegle do siebie



Rys. 8. Znaczenie funkcjonalne poszczególnych elementów zegara



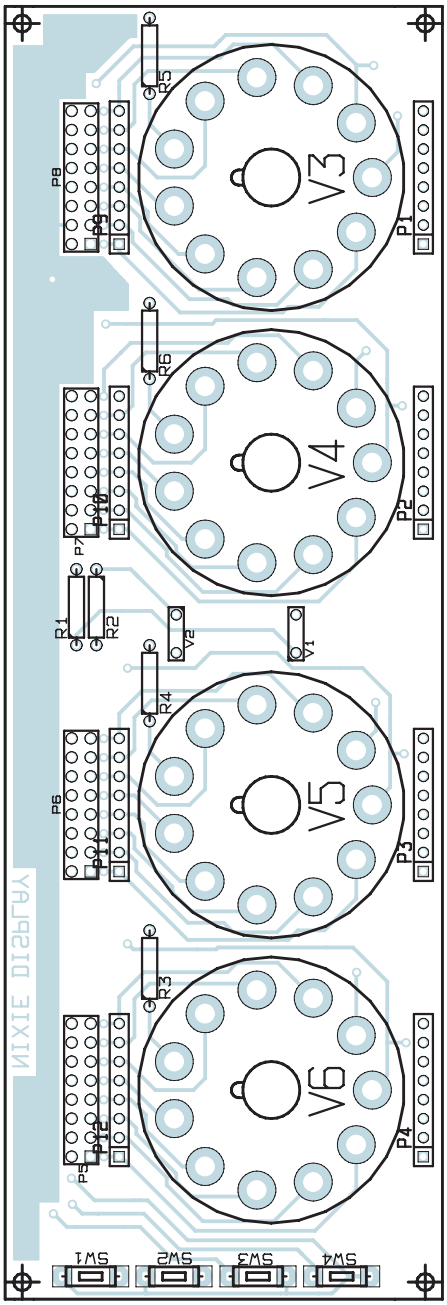
Rys. 9. Schemat montażowy płytki głównej zegara

Poza omówionymi elementami w pętli głównej programu zaszyta jest cała logika związana w obsługą zegara, czyli akcję przypisane do konkretnych przycisków i stanów zegara..

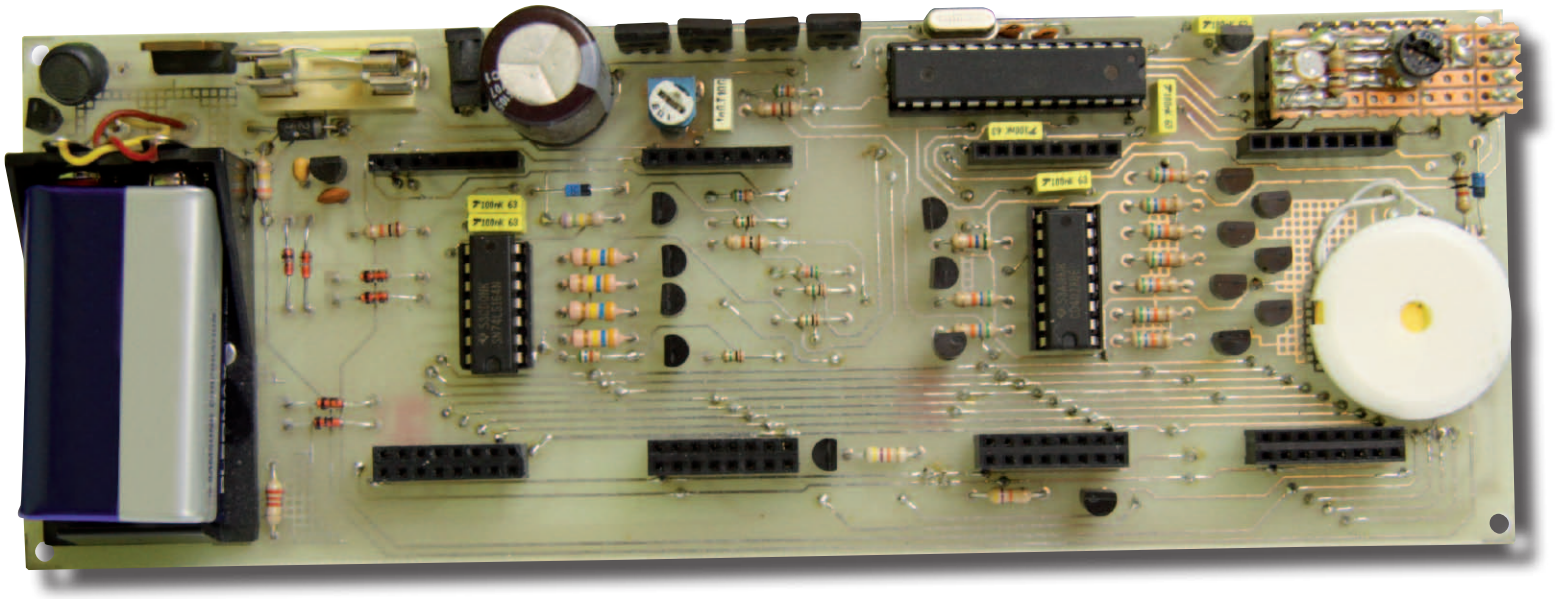
Montaż i uruchomienie

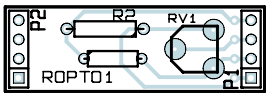
Lampy warto zamontować na płytce używając odpowiednich podstawek lub w przypadku ich braku można zastosować rurki metalowe. Na **rys. 6** i **rys. 7** przedstawiono sposoby łączenia płytek składających się na zegar. Należy zwrócić uwagę, że złącza na płytce wyświetlacza są lutowane od strony elementów. Jeśli z góry decydujemy się na konkretny sposób łączenia to można nie montować części złącz.

Pierwsze uruchomienie wymaga regulacji napięcia zasilania wyświetlaczy (poprawna wartość to 170 V). Regulacji tej dokonujemy jeszcze przed złożeniem płytek najlepiej z załadowanym programem dla wersji bez czujnika światła (nixie.hex/nixie.bin). W tym celu, przed podłączeniem zegara do zasilacza, ustawiamy potencjometr RV1 na minimalny podział (maksymalną wartość wyjściową napięcia). Następnie doprowadzamy zasilanie i regulujemy nastawę potencjometru tak, aby napięcie wyjściowe przetwornicy (mierzone np. na dodatniej elektrodzie C12) wynosiło 170 V. Po tej regulacji można połączyć obie płytki. Jeśli zegar będzie wyposażony w czujnik światła, to należy wgrać nowy program zawierający jego obsługę (usunięty komentarz z linii *OPTO_SENSOR* na początku pliku *nixie.c* lub zaprogramować procesor plikiem *nixie_opto.hex/nixie_opto.bin*), zamontować płytkę w odpowiednie złączach, a następnie dokonać regulacji progu zadziałania potencjometrem.



Rys. 10. Schemat montażowy płytki wyświetlacza





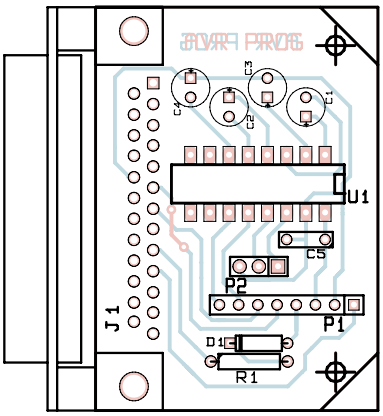
Rys. 11. Schemat montażowy czujnika oświetlenia

Obsługa

Zegar po pierwszym włączeniu pokazuje godzinę 1:00. Aktualna temperatury, data lub godzina budzenia wyświetlane są w czasie, gdy wciśnięty jest odpowiedni przycisk (odpowiednio TEMPERATURA, DATA, ALARM). Rozmieszczenie przycisków pokazano na **rys. 8**.

Wciśnięcie przycisku USTAWIENIA przełącza zegar w tryb zmiany parametrów. Zaraz po wciśnięciu zegar powinien zamiast godziny wyświetlić „00”, co oznacza zakładkę ustawienia godziny alarmu. Za pomocą przycisków +/– możemy zmieniać wyświetlaną wartość w zakresie od „00” do „03”. Wartości te oznaczają:

- „00” – ustawienie godziny budzenia,
- „01” – ustawienie melodyjki lub wyłączenie alarmu,
- „02” – ustawienie daty,
- „03” – ustawienie czasu,



Rys. 12. Schemat montażowy programatora

Wejście do odpowiedniej zakładki następuje po wciśnięciu OK. Następne kroki zależą od wybranej wcześniej zakładki:

- „00”/alarm – ustawiamy godzinę, przyciski „+”/„–” –> „OK” –> ustawiamy minuty, przyciski „+”/„–” –> „OK” –> sygnał potwierdzający zapamiętanie parametrów.
- „01”/melodia – wybieramy melodię, przyciski „+”/„–”; wartość „00” oznacza, że alarm jest nieaktywny –> „OK” –> sygnał potwierdzający zapamiętanie parametrów.
- „02”/data – ustawiamy rok, przyciski „+”/„–” –> „OK” –> ustawiamy miesiąc, przyciski „+”/„–” –> „OK” –> ustawiamy dzień, przyciski „+”/„–” -> „OK” –> sygnał potwierdzający zapamiętanie parametrów.
- „03”/czas – ustawiamy godzinę, przyciski „+”/„–” –> „OK” –> ustawiamy minuty, przyciski „+”/„–” –> „OK” –> sygnał potwierdzający zapamiętanie parametrów.

W każdej chwili możemy opuścić tryb ustawiania parametrów bez zapamiętywania zmian za pomocą przycisku ANULUJ.

Marcin Orłowski
2lero@wp.pl

Przypisy

[1] http://kicad.sourceforge.net/wiki/index.php
[2] http://gcc.gnu.org
[3] http://www.nongnu.org/avr-libc/
[4] http://www.vim.org/
[5] http://www.bsdhome.com/avrdude/
[6] http://www.debian.org/