DEVASTATOR, czyli bombowy zegarek

Estetyczny, praktyczny, efektowny, niezwykle pomysłowy zegar, który zawsze wywołuje efekt WOW wśród rodziny i znajomych.
Ponadto DEVASTATOR to pierwszy naprawdę uniwersalny projekt DIY. Jest odbierany równie entuzjastycznie przez osoby zupełnie niezwiązanie z techniką, jak i przez doświadczonych elektroników.
Czy można wyobrazić sobie lepszy sposób na prezentację wartości naszego wspólnego hobby?

Wbrew pozorom układ nie służy do wysadzania w powietrze budynków i pojazdów. Jest to projekt jak najbardziej „pokojowy", a nawet... kuchenny.

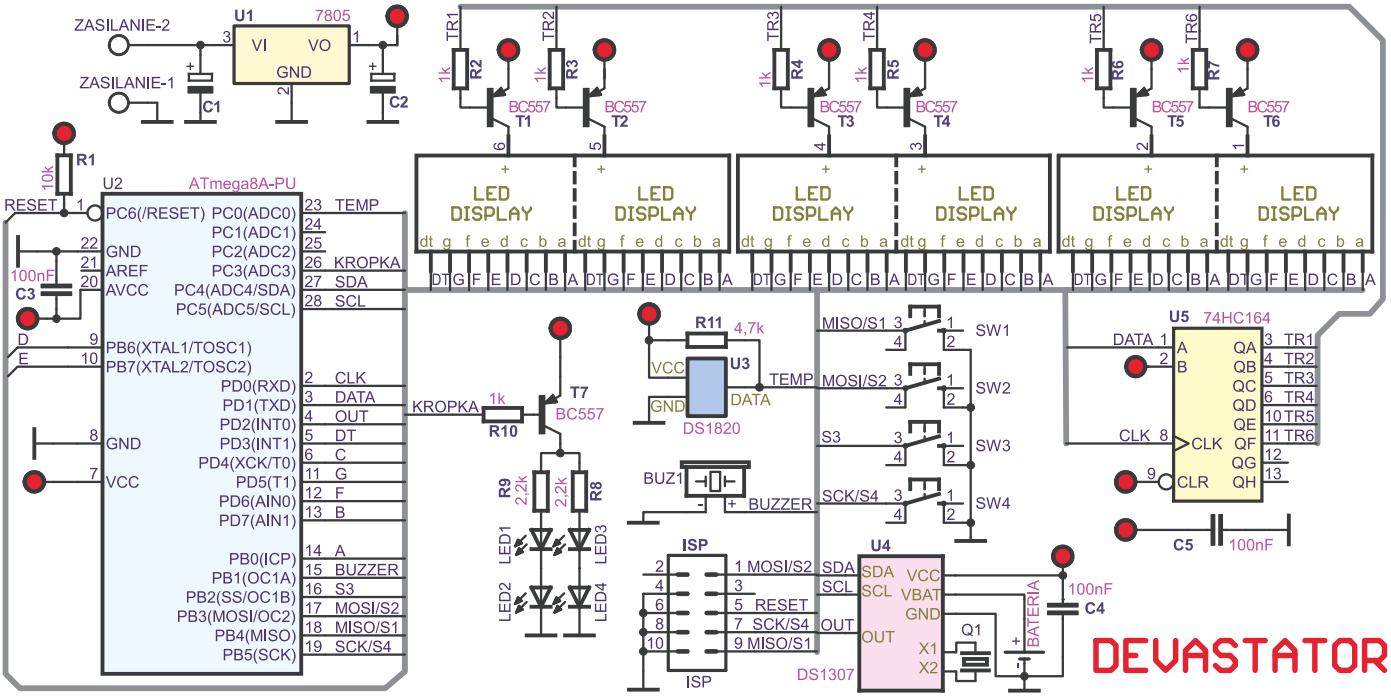
Podczas prac nad prezentowanym urządzeniem zwracałem dużą uwagę na jego walory użytkowe, dlatego projekt DEVASTATOR został zaprojektowany tak, aby był maksymalnie uniwersalny. Poza zegarem i prostym kalendarzem ma w sobie budzik, stoper, minutnik oraz termometr, dzięki czemu bombowy zegarek ma ogromną liczbę możliwych zastosowań. Sprawdzi się zarówno w salonie, jak i w kuchni, warsztacie lub sypialni.

Wiedząc, że wykończenie ma największy wpływ na ogólną prezentację oraz odbiór projektu, świadomie zrezygnowałem z klasycznej obudowy. Płytka drukowana zawierająca duże i charakterystyczne elementy przewlekane jest dumnie umieszczona na pierwszym planie i jako pierwsza zwraca na siebie uwagę. Dzięki temu całość nie sprawia wrażenia pospolitego czarnego pudełka, które działa w magiczny i niezrozumiały sposób – dzięki brakowi klasycznej obudowy projekt jest bardziej przyjazny dla widza. Dopełnieniem całości jest ogromna jaskrawoczerwona bateria imitacji lasek dynamitu, wzorowana na kreskówkach dla dzieci. Kolory i proporcje są dobrane w taki sposób, żeby bomba była realistyczna, ale od razu dająca do zrozumienia, że nie jest prawdziwa.

Dodatkowo imitacja stanowi wygodny stojak – podstawę, co pozwala ustawić płytkę w kilku pozycjach oraz dodaje niezwykle cenny akcent humorystyczny do projektu. W rezultacie bombowy zegarek może być zarówno ciekawym elementem wystroju wnętrza, jak i wartościowym rekwizytem w filmie lub podczas zabawy w ASG. Może również posłużyć do tworzenia rozmaitych żartów, których celem będą bliskie nam osoby z dużym poczuciem humoru.

Opis układu

Schemat ideowy przedstawiony jest na **rysunku 1**. Sercem prezentowanego urządzenia jest mikrokontroler ATmega8 taktowany wewnętrznym generatorem RC. Możliwe jest zastosowanie dowolnej z jego odmian, z tańszą niskonapięciową wersją włącznie. Wokół mikrokontrolera znajdują się standardowe elementy niezbędne do jego prawidłowej pracy. Dwa



Rys. 1

kondensatory odsprzęgające 100nF zostały umieszczone pod mikrokontrolerem i kostką DS1307 – są to jedyne elementy SMD na płytce.

Ze względu na stabilność działania, wymagany długi czas podtrzymywania bez zasilania oraz wygodę w obsłudze, pomiarem czasu zajmuje się zewnętrzny układ RTC DS1307, współpracujący z własnym kwarcem zegarkowym 32kHz. Podtrzymywanie mierzenia czasu przy braku zasilania zapewnia osobna bateria. Dane są przekazywane do mikrokontrolera za pośrednictwem interfejsu I²C (TWI), który został uproszczony przez zastąpienie zewnętrznych rezystorów podciągających rezystorami wbudowanymi w mikrokontroler. Nie wpłynęło to negatywnie na stabilność działania układu. Ponadto został wykorzystany pin OUT układu DS1307, skonfigurowany tak, aby generował stały sygnał 1Hz użyty następnie do generowania przerwania w mikrokontrolerze. Pomiar temperatury realizuje cyfrowy układ DS1820 podłączony do procesora przez interfejs 1-Wire.

Funkcję interfejsu użytkownika pełnią cztery przyciski wykorzystujące wewnętrzny pull-up mikrokontrolera, buzzer z generatorem, cztery diody LED (dwukropki pomiędzy wyświetlaczami) sterowane za pośrednictwem tranzystora PNP oraz trzy wyświetlacze 7-segmentowe ze wspólną anodą. Sterowanie poszczególnymi wyświetlaczami w celu umożliwienia multipleksowania zostało zrealizowane przez podłączenie anod wyświetlaczy do tranzystorów, które z kolei są sterowane przez rejestr przesuwny bez zatrzasku 74HC164. W ten sposób udało się znacznie uprościć projekt płytki. Diody składające się na wyświetlacze są podłączone do mikrokontrolera bezpośrednio, z pominięciem rezystorów ograniczających prąd. Ten kompromis wynika z braku miejsca na płytce, która musiała spełniać ustalone na samym początku założenia dotyczące wymiarów oraz zastosowania elementów przewlekanych. Nie wpływa to negatywnie na działanie układu w normalnych warunkach, ponieważ wyjścia mikrokontrolera wykazują znaczną impedancję. Niestety takie rozwiązanie sprawia, że mikrokontroler staje się bardzo wrażliwy na zbyt wysokie napięcie zasilania. Jeśli napięcie zasilania znacząco przekroczy 5V, to na tranzystorach wbudowanych w port mikrokontrolera może wydzielić się zbyt duża ilość ciepła, dlatego w projekcie został przewidziany stabilizator 7805 i miejsce na mały radiator. Jeśli bomba będzie zasilana zasilaczem dobrej jakości, stabilizator może zostać zastąpiony zworą (tak się stało w prezentowanym egzemplarzu). Oprócz tego na płytce znajduje się gniazdo służące do programowania mikrokontrolera.

Oprogramowanie. Program został napisany w C++ i jest umieszczony w Elportalu wśród materiałów dodatkowych do tego numeru EdW w postaci źródłowej z komentarzami oraz plików wynikowych. Kod jest podzielony na części, w których znajdują się funkcje odpowiedzialne za poszczególne zadania.

Pliki **ds.h** & **ds.cpp** zawierają funkcje niezbędne do obsługi termometru, a w pli-

	Zegar	Stoper	Minutnik	Ustawienia (tryb specjalny)
Przycisk 1	Pokaż datę	Start/pauza	Start/pauza	Dodaj jeden
Przycisk 2	Pokaż temperaturę	Reset	Reset	Odejmij jeden
Przycisk 3	Budzik ON/OFF, podgląd ustawionej godziny	Hold	Ustaw czas początkowy	Następna wartość
Przycisk 4	Następny tryb	Następny tryb	Następny tryb	Zapisz i wyjdź

Tabela 1

kach **onewire.h** & **onewire.c** jest umieszczona obsługa interfejsu 1Wire.

Pliki **rtc.h** & **rtc.cpp** zawierają obsługę układu DS1307 oraz rzeczy niezbędne do zarządzania czasem. Tutaj znajduje się między innymi definicja struktury do przechowywania czasu oraz instrukcje konwersji pomiędzy systemem dziesiętnym, który jest używany w programie, a systemem BCD, który jest zastosowany w układzie DS1307. Pliki **twi.h** & **twi.c** to biblioteka do obsługi wbudowanego w mikrokontroler sprzętowego interface'u TWI (I^2C).

Pliki **user.h** & **user.cpp** zawierają obsługę wyświetlacza, przycisków, kropki oraz buzzera. Tutaj znajdują się również definicje poszczególnych pinów.

Najważniejsze elementy programu zostały umieszczone w pliku **main.cpp**. Znajdują się tutaj: funkcja *main* zawierająca wywołania procedur inicjalizacyjnych, przerwanie *Timer2* pełniące funkcję pętli głównej programu oraz przerwanie INT0, w którym odbywa się aktualizacja globalnej struktury czasu.

Przerwanie *Timer2* wywoływane z częstotliwością 1kHz to najważniejsza część całego programu. Na początku znajdują się deklaracje zmiennych używanych w całym przerwaniu. Następnie realizowana jest obsługa przycisków, sztuczny generator milisekund oraz wybór trybu działania układu. W tym miejscu kolejno po sobie następują procedury obsługi: zegara wraz z wyświetlaniem daty i temperatury, budzika, stopera, minutnika i procedura obsługi specjalnego trybu ustawień, gdzie ustawiany jest aktualny czas, data oraz godzina zadziałania budzika. Umieszczenie wszystkich potrzebnych instrukcji w jednym miejscu, jedna po drugiej, pozwala na wykonywanie zadań wszystkich elementów jednocześnie, pomimo że wyświetlona może być tylko jedna funkcjonalność. Dzięki temu można jednocześnie uruchomić stoper i minutnik w tle oraz wyświetlić aktualną godzinę. Przerwanie kończy umieszczenie na wyświetlaczach danych pochodzących z aktualnie wybranego trybu.

```
//Fragment przerwania Timer2 ok. 1kHz
//Pętla główna całego programu
ISR(TIMER2_COMP_vect)
{
  …
   if(now.msec < 999) //Generator milisekund
   now.msec++;
  …
}
```

Listing 1

```
//Przerwanie zewnętrzne INT0 1Hz
//Pętla główna aktualizacji czasu
//Nieblokowanie pozwala na eliminację migotania wyśw.
ISR(INT0_vect, ISR_NOBLOCK)
{
 time buffer;  //Odczyt może być przerywany
 buffer = rtcGetTime();

 ATOMIC_BLOCK(ATOMIC_FORCEON) //Bezpośrednia aktualizacja
 {                            //jest operacją atomową
        now = buffer;
        now.msec = 0;
 }
}
```

Listing 2

```
//Fragment funkcji odpowiedzialnej za obsługę wyświetlacza LED
void ledProcess(uint8_t content[], uint8_t dot)
{
 volatile static uint8_t current = 5;
 //Zgaszenie wyświetlacza w celu uniknięcia poświaty
        off(LEDA_PORT, LEDA_NUM);
        off(LEDB_PORT, LEDB_NUM);
        off(LEDC_PORT, LEDC_NUM);
        off(LEDD_PORT, LEDD_NUM);
        off(LEDE_PORT, LEDE_NUM);
        off(LEDF_PORT, LEDF_NUM);
        off(LEDG_PORT, LEDG_NUM);
        off(LEDDT_PORT, LEDDT_NUM);
        //Wystarczy przejść na następny
        if(current < 5)
        {
                set(DSP_DATA_PORT, DSP_DATA_NUM);
                set(DSP_CLK_PORT, DSP_CLK_NUM);
                clear(DSP_CLK_PORT, DSP_CLK_NUM);
                current++;
        }
        //Jeśli nie, wróć do początku
        else
        {
                clear(DSP_DATA_PORT, DSP_DATA_NUM);
                set(DSP_CLK_PORT, DSP_CLK_NUM);
                clear(DSP_CLK_PORT, DSP_CLK_NUM);
                current = 0;
        }
        …
}
```

Listing 3

Warto zwrócić uwagę na zastosowanie software'owego generatora milisekund – **listing 1**. Było to konieczne, ponieważ układ DS1307 nie ma takiej funkcjonalności. Taki kod jest umieszczony w przerwaniu wywoływanym z częstotliwością 1kHz i przy każdym wywołaniu inkrementuje zmienną *msec* przechowującą milisekundy ze struktury *now,* w której znajduje się aktualny czas. Zmienna jest zerowana co sekundę w przerwaniu aktualizacji czasu pochodzącym od układu RTC, które zostało zaprezentowane na listingu 2. Zmienna nie będzie inkrementowana, jeśli jej wartość osiągnie 999. W momencie, kiedy software'owy generator milisekund będzie się spieszył w stosunku do zewnętrznego zegara RTC, zapobiegnie to przeładowaniu zmiennej *msec* – **listing 2**.

Precyzja takiego rozwiązania może budzić wątpliwości, ale okazuje się to nie mieć zauważalnego wpływu na działanie układu, ponieważ jest on co sekundę synchronizowany z czasem pochodzącym z układu RTC. Błąd wprowadzany przez to rozwiązanie jest wielokrotnie mniejszy niż czas reakcji na naciśnięcie przycisku, wydłużony dodatkowo przez algorytm eliminacji drgań styków. Przerwanie INT0 zostało zaprogramowane jako nieblokujące, ponieważ długotrwały odczyt danych z układu RTC powodował migotanie wyświetlaczy. Jedynie bezpośrednia aktualizacja struktury przechowującej aktualny czas jest operacją atomową.

Z powodu zastosowania rejestru przesuwnego procedura multipleksowania wyświetlacza różni się od tych najczęściej spotykanych. Zamiast ustawiania stanu wysokiego na kolejnych pinach mikrokontrolera, wysyłane są dane do rejestru przesuwnego – **listing 3**:

Sterowanie układem 74164 jest realizowane przez dwa piny: DATA i CLK. Opadające zbocze pojawiające się na pinie CLK powoduje wpisanie bitu pochodzącego z pinu DATA na pierwsze miejsce rejestru i przesunięcie pozostałych w prawo. Stan wyjść układu odzwierciedla wartości zapamiętanych przez rejestr bitów. Zaświecając pierwszy wyświetlacz, wysyłamy logiczną jedynkę, a następnie, w celu zaświecenia kolejnych wyświetlaczy, przesuwamy ją na kolejne pozycje, wysyłając zera. Unikamy przez to konieczności wysyłania całego bajtu za każdym razem.

Dzięki podziałowi projektu na wiele części, poszczególne fragmenty w łatwy sposób mogą zostać ponownie wykorzystane.

Program jest napisany dla **ATmega8** (i pokrewnych ATmega8A, ATmega8L,…) działających z zegarem **8MHz.** Fusebity powinny być ustawione następująco:

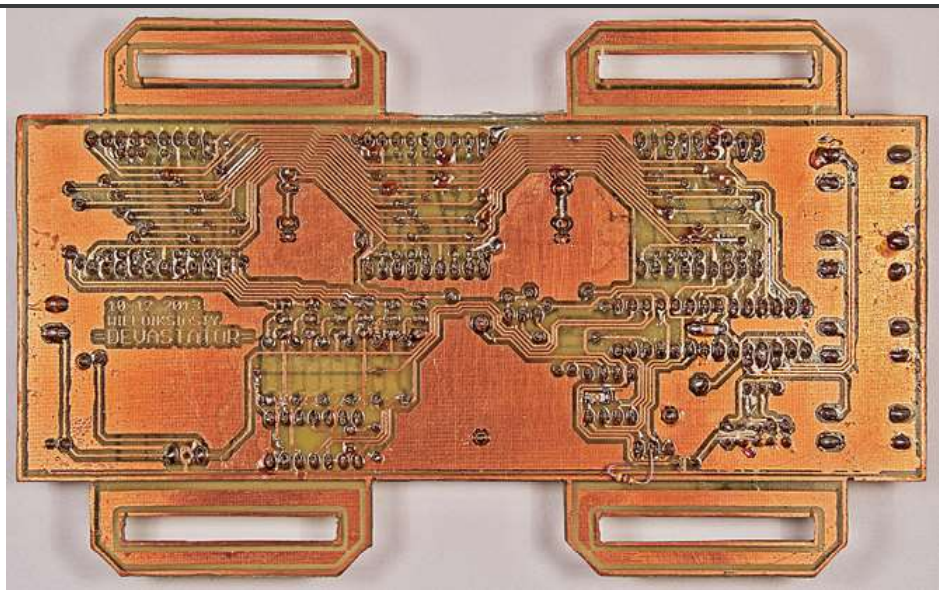
- Low: **E4**
- High: **D9**.

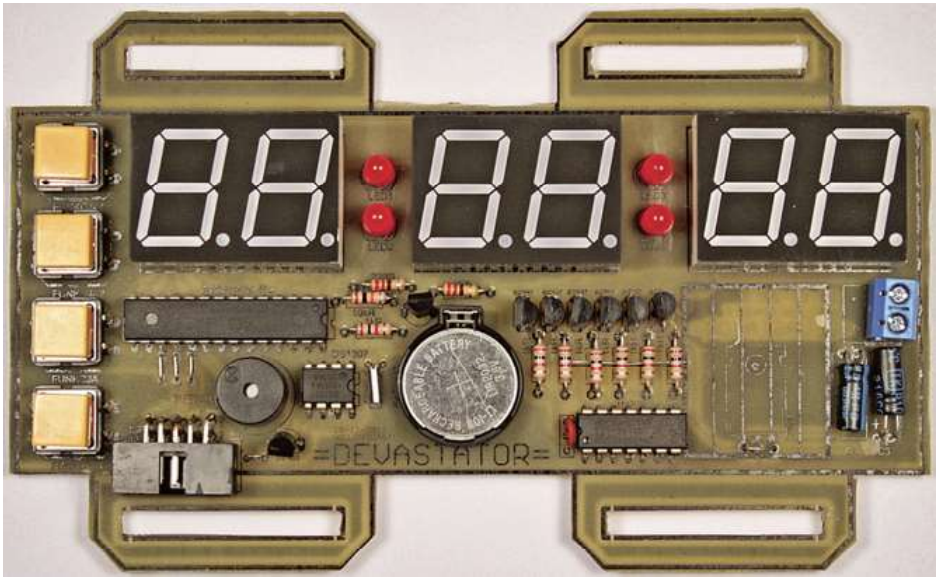
Mikrokontrolery ATmega88, ATmega168 lub ATmega328 również mogą zostać zastosowane, ale będzie to wymagało rekompilacji programu..

Obsługa urządzenia

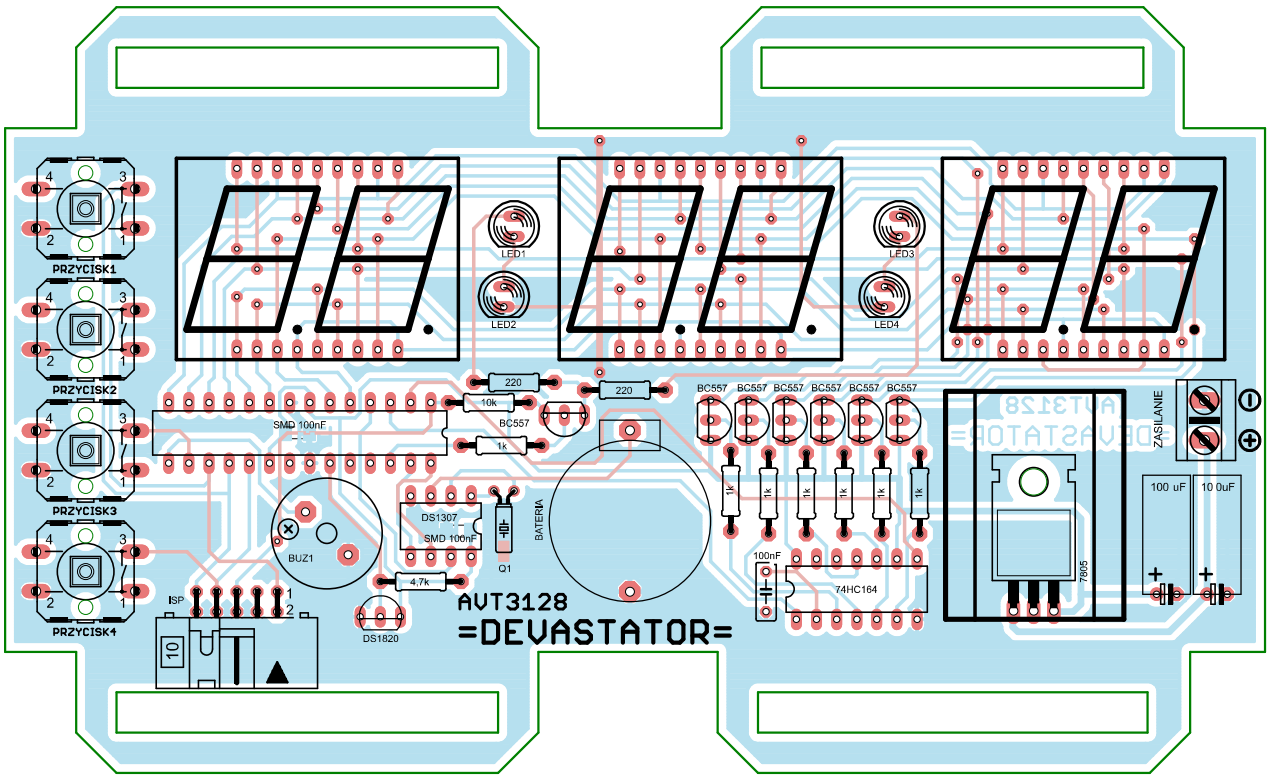
Układ może pracować w czterech trybach: zegara, stopera, minutnika oraz ustawień. Funkcje przycisków w poszczególnych trybach są opisane w **tabeli 1**. Prezentacja działania zegara na filmie znajduje się w materiałach dodatkowych do tego numeru EdW.

W ***trybie zegara*** przytrzymanie kolejnych przycisków pozwala wyświetlić datę, temperaturę oraz aktualnie ustawioną godzinę zadziałania budzika. Dodatkowo każdorazowe naciśnięcie przycisku 3 powoduje przełączenie stanu budzika. Kropka na dole skrajnie prawe-





Rys. 2



go wyświetlacza świeci, gdy budzik jest aktywowany.

Stoper mierzy czas od momentu startu. Reset służy do zerowania stopera, a *hold* to funkcja do zamrożenia wskazania wyświetlacza, bez zatrzymywania działania stopera. Ponadto, kiedy nie ma potrzeby wyświetlania godzin, dane przesuwane są o dwa miejsca w prawo, a zwolnione miejsce zajmowane jest przez milisekundy.

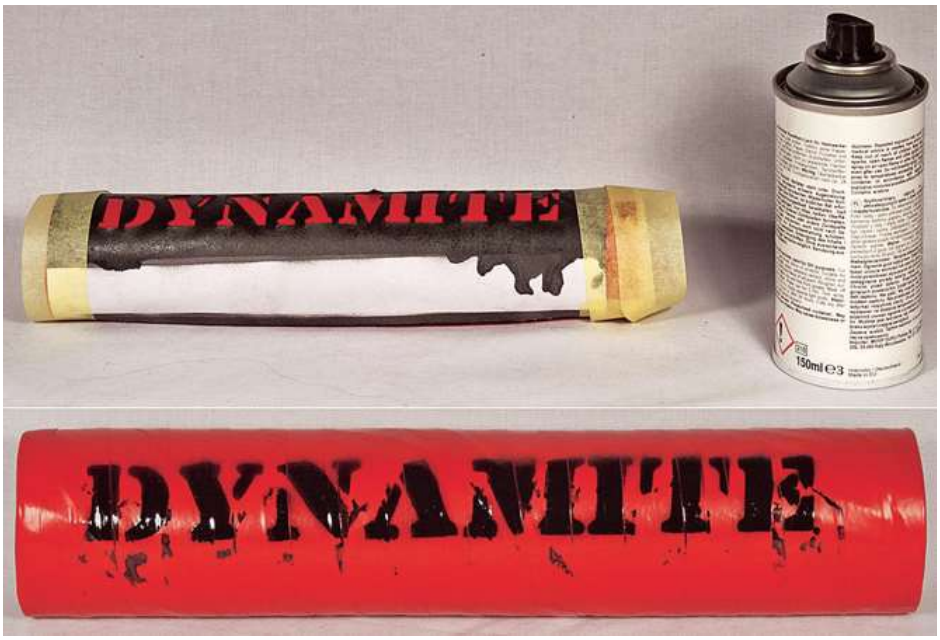
Minutnik działa na zasadzie odliczania czasu do zera i jest to funkcja najbardziej adekwatna do wyglądu zewnętrznego urządzenia. W trybie minutnika w miejscu funkcji *hold* znajduje się wejście w podtryb ustawiania czasu startowego minutnika. Obsługuje się go w taki sam sposób jak specjalny tryb ustawień, ale wyjście następuje samoczynnie po przejściu przez wszystkie trzy wartości za pomocą przycisku 3.

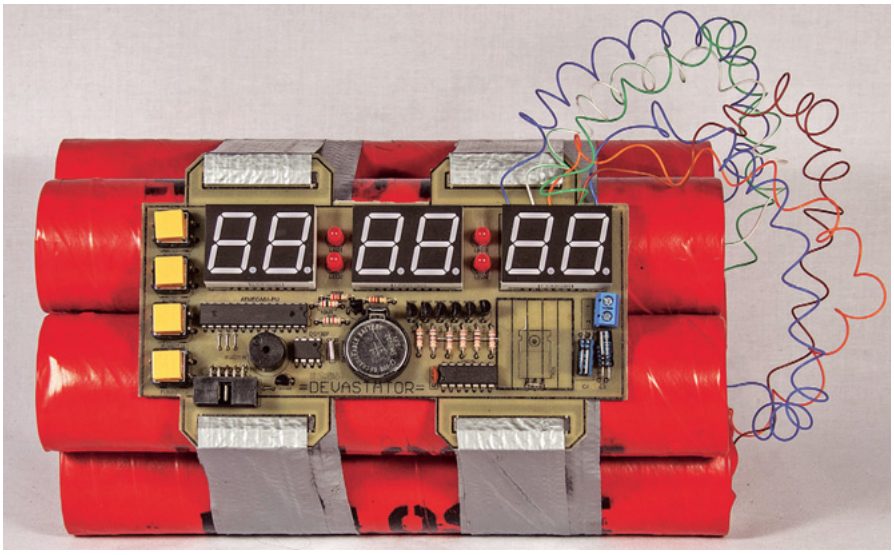
W końcu przytrzymanie przycisku 4 w każdym z trzech pierwszych trybów powoduje wejście w ***tryb ustawień***, gdzie ustawiane są kolejno: data, godzina zadziałania budzika oraz aktualny czas. Kropka na dole wyświetlaczy pozwala zorientować się, na którym etapie ustawień jesteśmy.

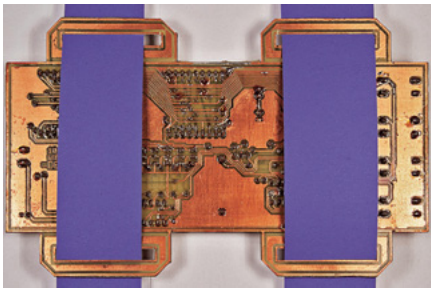
Montaż i uruchomienie

Układ można zmontować na jednostronnej płytce drukowanej, której projekt zamieszczony jest na **rysunku 2**. Na płytce nie są pokazane numery większości elementów. W przypadku tego prostego układu nie jest to problemem, a wprost przeciwnie: na płytce pokazane są wartości kluczowych elementów, co jest wprawdzie nietypowe, ale powinno nawet ułatwić montaż.

Wszystkie elementy są przewlekane, poza dwoma kondensatorami umieszczonymi pod mikrontrolerem i DS1307.







Standardowo montujemy układ, zaczynając od elementów najmniejszych, a kończąc na największych.
Układ nie wymaga żadnego uruchomienia, poza zaprogramowaniem procesora i zaktualizowaniem godziny oraz daty. Zmontowany ze sprawnych elementów powinien od razu prawidłowo pracować. Osoby niedoświadczone powinny poprosić kogoś bardziej zaawansowanego o pomoc w zaprogramowaniu procesora.

Imitacja dynamitu, do której została przymocowana płytka drukowana, jest niezwykle istotnym elementem tego projektu, dlatego przedstawiam szczegóły jej wykonania. Imitacje poszczególnych lasek dynamitu są wykonane z... tekturowych rdzeni ręczników kuchennych, ale można zastosować rurę PCV lub nawet zrolowany papier pakowy o odpowiedniej gramaturze. Rdzeń dynamitu powinien mieć wymiary ok. 220mm x 45mm, tak aby zostały zachowane odpowiednie proporcje. Istotne jest również, aby rdzeń był możliwie jak najsztywniejszy.

Ponadto na dynamit składają się: tekturowe lub kartonowe krążki o średnicy odpowiadającej wewnętrznej średnicy rurek, makulatura oraz czerwona taśma izolacyjna. Warto zwrócić uwagę na szerokość taśmy: im szersza, tym łatwiejsza jest praca.

W celu uzyskania laski dynamitu należy wykonać następujące kroki:
- z jednej strony rury wklejamy tekturowe kółko na głębokość ok. 1cm. Aby ułatwić sobie pracę, można przebić rurkę zapałką, patyczkiem lub cienkim śrubokrętem. Następnie miejsce styku tektury zalewamy klejem.
- Po wyschnięciu kleju do środka wciskamy ciasno zwinięte gazety w celu zwiększenia masy i wzmocnienia konstrukcji. Kolejnym krokiem jest powtórzenie procedury wklejania denka z drugiej strony rury. Miejsce wewnątrz dynamitów może zostać wykorzystane do przechowywania baterii lub akumulatora. W takim wypadku należy umieścić w środku odpowiednio mniejszą ilość makulatury oraz pozostawić nieprzyklejone denko.
- Wreszcie całość owijamy czerwoną taśmą izolacyjną, pamiętając, aby zostawić jak najwięcej materiału na zakładki.
- Wystającą taśmę zawijamy do środka i stabilizujemy kolejnymi tekturowymi krążkami. Z użyciem czarnej farby w sprayu oraz szablonu malujemy ostrzegawczy napis na dynamicie.

Wykaz elementów

R1	10kΩ
R2, R3, R4, R5, R6, R7, R10	1kΩ
R8, R9	2,2kΩ
R11	4,7kΩ
C1, C2	100µF/25V
C3, C4	100nF SMD
C5	100nF
LED1, LED2, LED3, LED4	LED 3MM czerwona
DSP1, DSP2, DSP3	LED 7-seg 1" anoda
T1, T2, T3, T4, T5, T6, T7	BC557
U1	7805
U2	ATmega8A-PU
U3	DS1820
U4	DS1307
U5	74HC164
BAT1	CR2032H
Q1	kwarc 32kHz
BUZ1	buzzer z generatorem
ISP	IDC 2x5 kątowe
SW1, SW2, SW3, SW4	10mm

Ostatnim krokiem jest sklejenie poszczególnych elementów bomby za pomocą taśmy. Najlepszy efekt, moim zdaniem, otrzymujemy przy zastosowaniu popularnej taśmy zbrojonej. Na końcu całość została dopełniona zwiniętymi w spiralę kolorowymi drucikami. W tej roli równie dobrze sprawdzi się np. sznurek, który może udawać lont.

Tomasz Szewczyk
wieloiksiasty@gmail.com