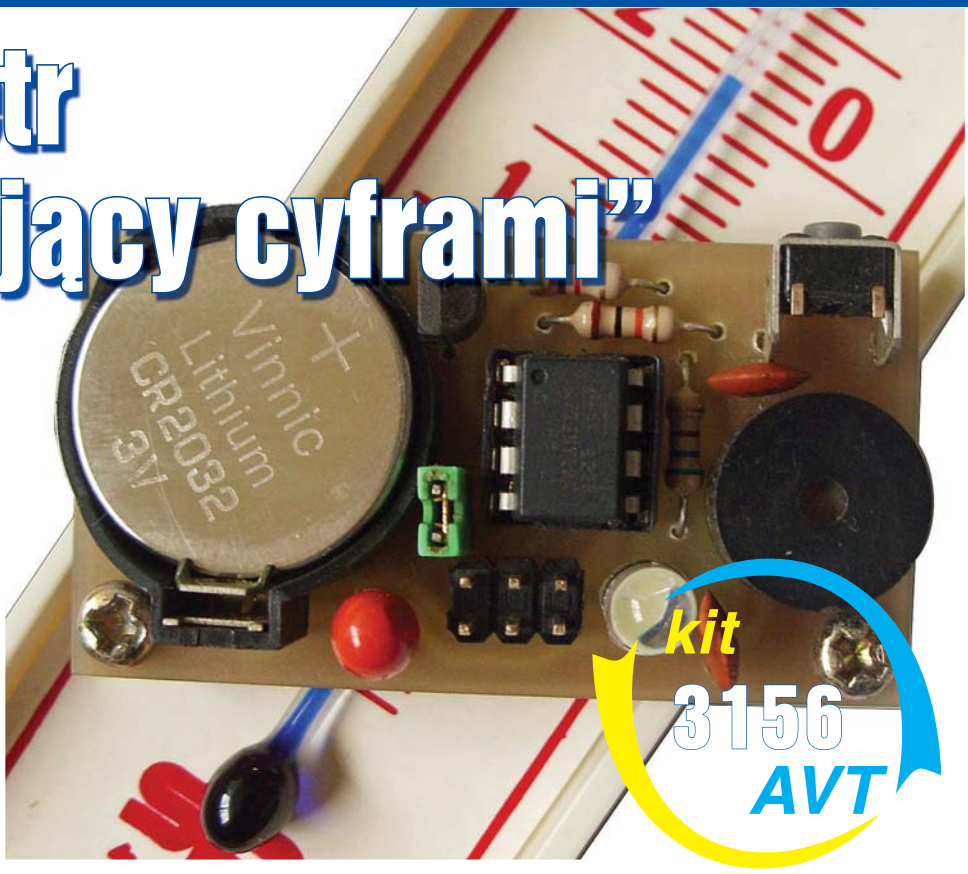
Termometr „ćwierkający cyframi” w kodzie Morse’a



Nieskomplikowany termometr pokojowy z prezentacją wyniku pomiaru kodem Morse’a, tj. impulsami świetlnymi lub/i dźwiękowymi.

Do czego to służy?

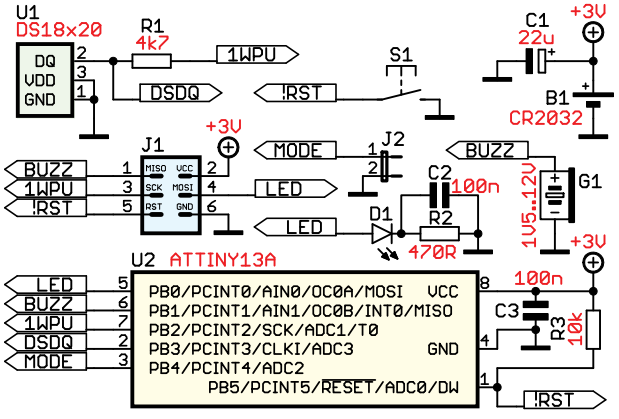
Na łamach prasy elektronicznej pomiar temperatury wydaje się nieśmiertelnym tematem dyżurnym. Mimo że poszczególne rozwiązania różnią się między sobą rozwiązaniem układowym, stopniem komplikacji, liczbą czujników, to najczęściej mają wizualną prezentację wyniku zmierzonej temperatury. W tej roli prym wiodą wszelkiej maści wyświetlacze. W przypadku termometru ogólnego przeznaczenia rodzaj wyświetlacza jest zwykle główną cechą braną pod uwagę przez użytkownika przy podejmowaniu decyzji o kupnie termometru. Mniej istotna jest dokładność i rozdzielczość pomiaru. W opisywanym układzie do prezentacji wyniku pomiaru zastosowano kod wynaleziony w pierwszej połowie XIX wieku przez Samuela Morse’a.

Impulsowa natura kodowania Morse’a pozwala na dwa minimalistyczne interfejsy: dźwiękowy lub/i wizualny (LED). Zakres pomiarowy rozciąga się w zakresie –55..125°C z rozdzielczością 1°C. W przypadku temperatury ujemnej jako pierwszy wytwarzany jest kod znaku „–”. Cyfry można generować na dwa sposoby: kodami o stałej liczbie impulsów lub zmiennej liczbie impulsów, tzw. skróconymi. Być może dźwiękowy sposób prezentacji mierzonej temperatury przypadnie do gustu osobom niewidomym.

Jak to działa?

Na **rysunku 1** przedstawiono schemat ideowy. Sercem układu jest mikrokontroler U2 (ATtiny13A) pełniący funkcję „tłumacza” wartości temperatury odczytywanej z czujnika U1 (DS18B20 lub DS18S20) na impulsy elektryczne w kodzie Morse’a. Buzzer G1 generuje impulsy dźwiękowe, LED D1 impulsy optyczne. Zwora J2 umożliwia wybór typu generowanych cyfr kodu Morse’a między kodami o stałej liczbie impulsów (otwarta) a kodami skróconymi (zamknięta). Generowane przez układ kody Morse’a cyfr i znaku minus zawarto w tabeli na **rysunku 2**. Układ zasilany jest napięciem 3V z baterii B1 CR2032. Kondensator C1 ma za zadanie nie tyle filtrowanie napięcia zasilającego, co gromadzenie energii, zmniejszając tym samym spadki napięcia powstałe na dość dużej rezystancji wewnętrznej B1.

Rys. 1



Program napisany jest dla kompilatora Bascom-AVR. Większość prostych instrukcji wysokiego poziomu zastąpiono wstawkami asemblerowymi, co wpływa na czas wykonywania programu i pośrednio na zużycie baterii. Algorytm działania programu sprowadza się do wykonania pomiaru temperatury, przekonwertowania uzyskanej wartości numerycznej i na jej podstawie wygenerowania kodów cyfr Morse’a. Po tych czynnościach następuje wprowadzenie mikrokontrolera w stan obniżonego poboru energii. Po naciśnięciu przycisku S1 (Reset) algorytm jest powtarzany. Dla zapewnienia minimalnego poboru prądu w trybie uśpienia na początku programu wyłączane są nie- używane bloki peryferyjne mikrokontrolera

Rys. 2

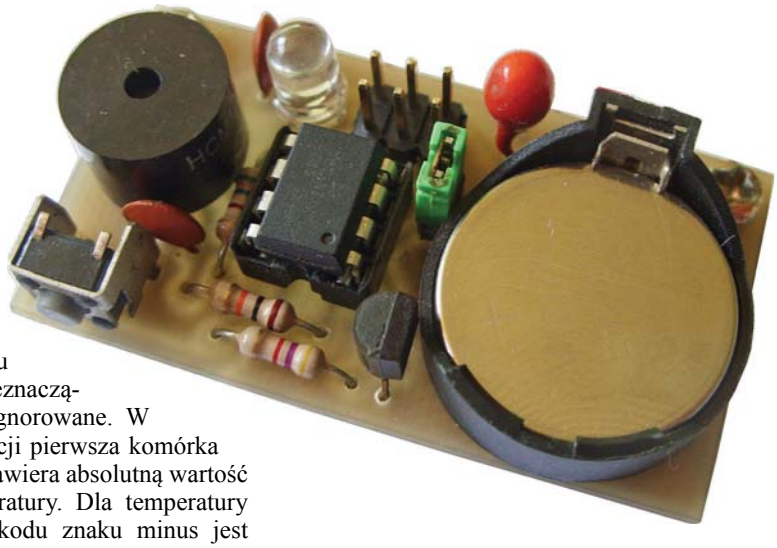
	Zwykły	Skrócony
0	– – – – –	–
1	• – – – –	• –
2	• • – – –	• • –
3	• • • – –	• • • –
4	• • • • –	• • • • –
5	• • • • •	• • • • •
6	– • • • •	– • • • •
7	– – • • •	– • • •
8	– – – • •	– • •
9	– – – – •	– •
-	– • • • • –	– • • • • –

w rejestrach ACSR i PRR. Z tego samego powodu rezystor R1 realizujący podciąganie magistrali 1Wire został dołączony do nóżki mikrokontrolera. Bezpośrednio przed „zaśnięciem" we wszystkie bity (z atrybutem W) rejestru PORTB wpisywane są zera logiczne, następnie wszystkie wyprowadzenia konfigurowane są jako wyjścia w rejestrze DDRB. Zabieg taki wyłącza całkowicie zasilanie czujnika temperatury, minimalizuje prądy upływu oraz zapewnia brak stanów nieustalonych na nóżkach procesora. Bezpośrednio przed „zaśnięciem" jednostki wyłączane jest zasilanie już niepotrzebnego licznika T/C0 w rejestrze PRR.

Po restarcie U2 (naciśnięciu S1) po standardowych poleceniach konfiguracyjnych następuje odczyt temperatury z czujnika U1 (wysokopoziomowa obsługa magistrali 1W) w następującej sekwencji:

1. Przypisanie wyprowadzenia mikrokontrolera do obsługi magistrali 1W.
2. Reset magistrali.
3. Wysłanie rozkazu (&HCC) pomijania odczytu ROM czujnika; nie ma potrzeby adresowania, ponieważ do magistrali podłączony jest jeden czujnik.
4. Wysłanie rozkazu (&H44) inicjującego konwersję temperatury.
5. Oczekiwanie 750ms na konwersję.
6. Ponowny restart magistrali.
7. Ponowne wysłanie rozkazu (&HCC) pomijania odczytu ROM czujnika.
8. Wysłanie rozkazu (&HBE) odczytu zmierzonej temperatury.
9. Odczytanie do tablicy ***DsData*** dwóch bajtów (wartość temperatury) z magistrali czujnika.

Konwersja odczytanej temperatury do wartości dziesiętnej zależna jest od typu czujnika U1 wybieranego na etapie kompilacji programu. Podczas konwersji obliczana jest liczba zajmowanych pozycji dziesiętnych przez wartość temperatury (ilość cyfr). Dzięki temu zera wiodące (nieznaczące) są później ignorowane. W wyniku tej operacji pierwsza komórka tablicy ***DsData*** zawiera absolutną wartość dziesiętną temperatury. Dla temperatury ujemnej indeks kodu znaku minus jest wstawiany do tablicy ***CodeTab*** w wyliczonej (zależnie od ilości cyfr) komórce. W wyniku dzielenia całkowitego i modulo absolutnej wartości temperatury przez 10, 100, zależnie od pozycji dziesiętnej i ilości cyfr, tablica ***CodeTab*** zawiera w swych komórkach indeksy definicji kodów cyfr do wygenerowania. Definicje cyfr w kodzie Morse'a zawarto w dwóch tablicach: ***Long_Morsea*** dla kodu zwykłego, ***Short_Morsea*** dla kodu skróconego. Zależnie od stanu logicznego na wejściu PB.4 (MODE), rejestr roboczy ładowany jest bajtem, kodem cyfry z jednej z tablic spod indeksu przetwarzanej aktualnie cyfry w tablicy ***CodeTab***. Bajtem, gdzie binarne jedynka odpowiada kropce, a zero kresce. By poprawnie generować cyfry o różnej liczbie impulsów, pierwszy bit bajtu za kodem znaku ma wartość jeden, wszystkie bity za nim przyjmują wartość zero. Przesuwanie rejestru roboczego R16 o jeden bit w lewo (do najmniej znaczącego bitu „wsuwane" jest zero) odbywa się w pętli generującej kod Morse'a. Czas trwania impulsu (kropka lub kreska) wyznacza najbardziej znaczący bit rejestru R16. Wyjście BUZZ ustawiane jest w stan H na czas trwania kropki i kreski, włączając brzęczyk z wbudowanym generatorem G1. Wartość &B10000000 w rejestrze R16 oznacza koniec generowania kodu, zakończenie pętli. Procedura generowania kodu Morse'a pojedynczej cyfry (znaku) powtarza się tyle razy, ile pozycji dziesiętnych zajmuje wartość temperatury. W czasie trwania impulsów znaku Morse'a licznik T/C0 pracuje w trybie zerowania licznika przy zgodnym porównaniu (CTC – Clear Timer on Compare Match), co oznacza tyle, że licznik sprzętowo, cyklicznie (z interwałem zależnym od ustawionego podziału zegara, tętna mikrokontrolera) zwiększa zawartość rejestru TIMER0 i porównuje z wpisaną



wcześniej do rejestru OCR0A wartością. W momencie zrównania zawartości obu rejestrów rejestr TIMER0 zaczyna liczyć od zera, a stan wyjścia OC0A zmienia się na przeciwny (ustawiony bit COM0A0 w rejestrze TCCR0A). Stała **Freq** wyznacza częstotliwość generowanego przebiegu na wyjściu OC0A. Wartość wpisywana do rejestru porównania wyznaczana jest ze wzoru: $OCR0A=f_{OSC}[Hz]/_{DIV8}[8]/P_{TC0}/2/Freq$, gdzie: f_{OSC}=częstotliwość zegara systemowego, Fb_{DIV8}=podział zegara systemowego (fusebit), P_{TC0}=podział preskalera TC0. Uzyskana na wyjściu OC0A LED częstotliwość jest kluczowana w takt znaków Morse'a, co zmniejsza pobór prądu, gdy z wyjścia sterowana jest LED. Rolą równolegle włączonego do R2 kondensatora C2 jest zwiększanie prądu płynącego przez LED D1 w momencie pojawienia się zbocza narastającego na wyjściu OC1A dając tym efekt jaśniejszego świecenia.

Możliwa jest współpraca z jednym z dwóch typów czujnika, tj. DS1820 i DS18B20. Ponieważ czujniki różną się rozdzielczością wyniku pomiaru, wyboru czujnika należy dokonać na etapie kompilacji programu, modyfikując stałą **DS_Type**. Wartość stałej determinuje rodzaj współpracującego czujnika, włączając, wyłączając odpowiednie fragmenty kodu między dyrektywami warunkowymi kompilatora #If..#EndIf.

Drugą stałą podlegającą modyfikacji jest **Dot_ms**, która wyznacza czas trwania kropki, zatem ustala zależności czasowe generowanego kodu zgodnie z przyjętymi zasadami kodowania Morse'a, tj. czas trwania kreski równy trzykrotności interwału kropki, czas „ciszy" między impulsami znaku równy interwałowi kropki, czas „ciszy" między znakami równy interwałowi kreski. Wartość tej stałej wpływa na szybkość

Wykaz elementów

Rezystory:

R2	470Ω
R1	4,7kΩ
R3	10kΩ

Kondensatory:

C2, C3	100n ceramiczny
C1	22u/6,3V tantalowy

Półprzewodniki:

D1	LED superjasna Ø 5mm
U1	DS18x20
U2	ATtiny13A

Inne:

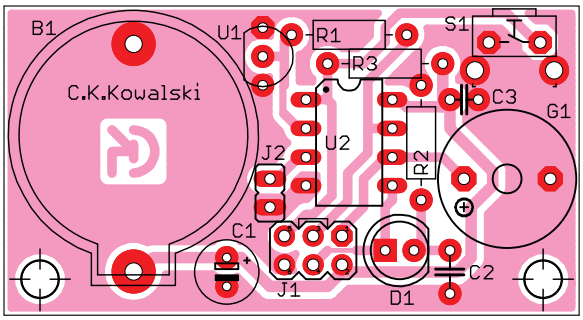
B1	Koszyk + bateria CR2032
S1	Mikrowłącznik B3F-31XX kątowy
G1	HCM1212X buzzer 1V5...12V
J1	Złącze grzebieniowe 2,54mm M2x3
J2	Złącze grzebieniowe 2,54mm M1x2 + Jumper

Komplet podzespołów z płytką jest dostępny w sieci handlowej AVT jako kit szkolny AVT-3156.

generowanego kodu Morse'a, należy ją zmodyfikować wg własnych upodobań.

Montaż i uruchomienie

Jednowarstwowy obwód drukowany widoczny jest na **rysunku 3**. Po sprawdzeniu PCB na występowanie zwarć i pęknięć można przystąpić do montażu. Przed kompilacją programu należy wybrać typ czujnika, modyfikując stałą **DS_Type** wg opisu w poprzednim śródtytule. Program działa przy fabrycznych ustawieniach bitów konfiguracyjnych (fusebit). Do zaprogramowania U2 przewidziano złącze J1 w standardzie firmy Atmel. Termometr po zaprogramowaniu mikrokontrolera nie wymaga uruchamiania i jest gotowy do pracy.



Rys. 3. Skala 150%

Cyprian Kamil Kowalski
c4v2@o2.pl