W ofercie AVT*
AVT6059

Najważniejsze parametry:
- źródło napięcia zasilania: bateria CR2032,
- teoretyczny czas pracy na jednej baterii zasilającej: 14 miesięcy*,
- zakres pomiarowy opcjonalnego termometru: 0...55°C,
- dokładność pomiaru temperatury: 0,5°C,
- rozdzielczość pomiaru temperatury: 0,1°C.

* szczegóły w tekście artykułu

Dodatkowe materiały do pobrania ze strony www.ulubionykiosk.pl/media

AVT6043	matrixClock – efektowny zegar stołowy (EP 6–7/2024)
----	Energooszczędny zegar LED (EP 4/2023)
----	Ogromny zegar LCD bez procesora (EP 7/2022)
AVT5920	Licznik czasu pracy z wyświetlaczem LCD (EP 1/2022)
AVT5906	Clock (EP 12/2021)
----	Zegar na rękę (EP 12/2020)

* **Uwaga!** Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania! Podstawową wersją zestawu jest wersja **[B]** nazywana potocznie KIT-em (z ang. zestaw). Zestaw w wersji **[B]** zawiera elementy elektroniczne (w tym **[UK]** – jeśli występuje w projekcie), które należy samodzielnie wlutować w dołączoną płytkę drukowaną (PCB). Wykaz elementów znajduje się w dokumentacji, która jest podlinkowana w opisie kitu. Mając na uwadze różne potrzeby naszych klientów, oferujemy dodatkowe wersje:

- **wersja [C]** – zmontowany, uruchomiony i przetestowany zestaw **[B]** (elementy wlutowane w płytkę PCB),
- **wersja [A]** – płytka drukowana bez elementów i dokumentacji.

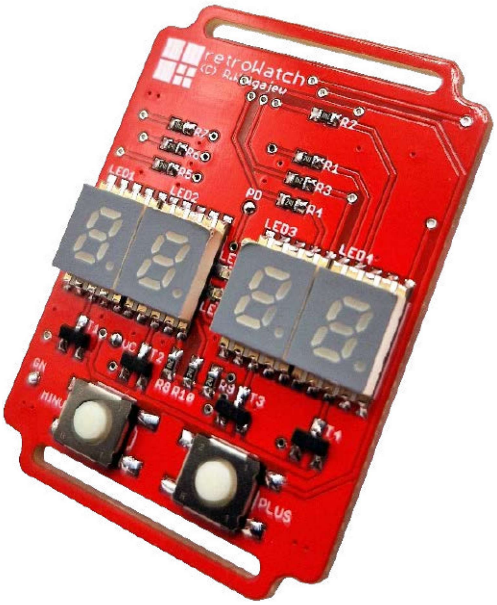
Kity, w których występuje układ scalony wymagający zaprogramowania, mają następujące dodatkowe wersje:
- **wersja [A+]** – płytka drukowana **[A]** + zaprogramowany układ **[UK]** i dokumentacja,
- **wersja [UK]** – zaprogramowany układ.

Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik PDF! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! **http://sklep.avt.pl**

W przypadku braku dostępności na stronie sklepu osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: **kity@avt.pl**.

retroWatch
– zegarek naręczny w stylu retro

W dzisiejszych czasach, gdy postęp w dziedzinie elektroniki użytkowej nabiera rozpędu w sposób co najmniej geometryczny, u wielbicieli klasycznej techniki coraz częściej dochodzą do głosu nuty nostalgii. Materializują się one w postaci urządzeń nawiązujących do stylu retro. I co ciekawe, mimo oczywistej prostoty, jak i nieprzystawania do dzisiejszych możliwości (a zapewne również do części oczekiwań), konstrukcje te z powodzeniem zdobywają coraz większą rzeszę zwolenników.



Ustawienia Fuse-bitów
FREQSEL[1:0]: 01
RSTPINCFG[1:0]: 01*
SUT[2:0]: 111*
EESAVE: 0*
* – ustawienie domyślne producenta

Przypomina mi to sytuację z branży modowej, w której można zauważyć pewne cykliczne powroty popularności danego stylu – i nikt nie zgłasza żadnych wątpliwości co do takiego zjawiska. Pamiętać wszak należy, iż rynkiem rządzi popyt, a on – jak widać – polubił się na dobre ze stylem retro. Dokładnie w te oczywiste oczywistości (jak by powiedział klasyk) wpisuje się mój najnowszy projekt: naręczny zegarek LED, którego pierwowzorem był czasomierz firmy Unitra Warel o oznaczeniu DW-2005, wprowadzony na rynek w latach 80. Owego czasu stał się on bardzo popularny, a to za sprawą generała Mirosława Hermaszewskiego, który właśnie z tym modelem odbył swój pierwszy (i jedyny) lot w kosmos. Dzisiaj urządzenia tego typu, zwłaszcza w stanie kolekcjonerskim, potrafią osiągać na portalach sprzedażowych cenę kilku tysięcy złotych – i nic nie wskazuje, by miało się to zmienić, wszak tych egzemplarzy jest coraz mniej. Właśnie dlatego – na bazie wspomnień i nostalgii – powstał niniejszy projekt, którego schemat ideowy pokazano na **rysunku 1**.

Sercem układu jest mikrokontroler ATtiny416, taktowany wewnętrznym oscylatorem o częstotliwości 1 MHz. Jak nietrudno się domyślić, głównym wyzwaniem przy projektowaniu zegarka była chęć minimalizacji zużycia energii, a co za tym idzie – wydłużenia czasu pracy na jednym ogniwie zasilającym. Mikrokontroler nasz realizuje obsługę 7-segmentowego wyświetlacza LED o organizacji 4 znaków w konfiguracji wspólnej anody (wyprowadzenia PC3...PC0 mikrokontrolera). Korzysta w tym celu z wbudowanego układu czasowo-licznikowego TCA0, który – pracując w trybie NORMAL – wywołuje stosowne przerwanie systemowe (od przepełnienia → TCA0_OVF_vect) 240 razy na sekundę (czyli 60 razy dla każdej wspólnej anody), obsługując właściwy mechanizm multipleksowania. Ponadto mikrokontroler obsługuje prostą klawiaturę złożoną z przycisków PLUS i MINUS, korzystając z wbudowanego układu czasowo-licznikowego TCB0, który pracując w trybie PERIODIC_INTERRUPT, wywołuje przerwanie od przepełnienia (TCB0_INT_vect) 100 razy na sekundę (co 10 ms), przez co – po pierwsze – zapewniona została nieblokująca obsługa klawiatury systemowej z eliminacją zjawiska drgania styków, a po drugie – możliwa stała się obsługa krótkiego i długiego naciśnięcia przycisków. Ostatnią i zarazem fundamentalną funkcjonalnością naszego mikrokontrolera jest zegar czasu rzeczywistego RTC. Zwykle funkcjonalność taką realizuje się, implementując jakiś sprzętowy zegar czasu rzeczywistego (na przykład DS1307 lub MCP79410), lecz w naszym systemie stawiamy na minimalizację zużycia energii, w związku z czym zrealizujemy ją z użyciem wbudowanego w strukturę mikrokontrolera układu RTC. Pracując w trybie PERIODIC_INTERRUPT, wywołuje on przerwanie od przepełnienia (RTC_PIT_vect) raz na sekundę, a tym samym realizuje funkcjonalność zegara RTC. Co więcej, oscylator wspomnianego zegara RTC, taktowany zewnętrznym rezonatorem kwarcowym o częstotliwości 32,768 kHz, pracować może podczas trybu Power Down mikrokontrolera, przez co możliwa stała się realizacja zegara RTC w każdym trybie jego pracy. Tak naprawdę

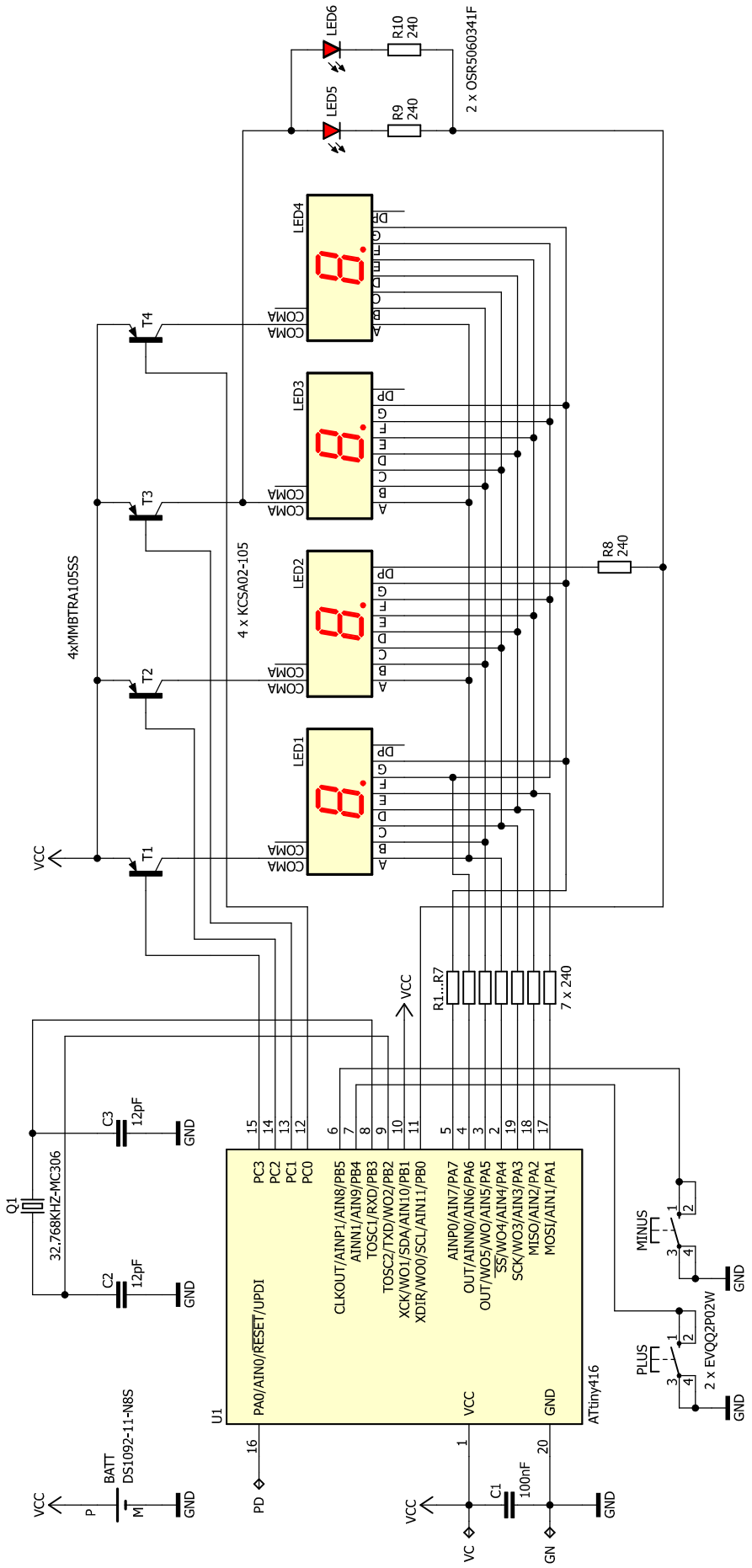
Wykaz elementów:

Rezystory: (SMD 0603)
R1...R10: 240 Ω

Półprzewodniki:
U1: ATtiny416 (SO20)
T1...T4: MMBTRA105SS (SOT23)
LED1...LED4: wyświetlacz 7-segmentowy 0,2", wspólna anoda (typ KCSA02-105)
LED5, LED6: dioda LED SMD czerwona typu OSR5060341F (SMD 0603)

Kondensatory: (SMD 0603)
C1: 100 nF (X7R)
C2, C3: 12 pF (X7R)

Pozostałe:
Q1: rezonator kwarcowy zegarkowy SMD 32768 Hz (typ 32.768KHZ-MC306)
PLUS, MINUS: mikroprzełącznik TACT SMD (typ EVQQ2P02W)
BATT: koszyczek baterii CR2032 SMD (typ DS1092-11-N8S)
VCC: bateria CR2032



Rysunek 1. Schemat ideowy urządzenia

skrajnie niski prąd trybu Power Down rzędu 300 nA (przy taktowaniu mikrokontrolera równym 1 MHz) sprawia, iż opisane rozwiązanie jest dużo korzystniejsze energetycznie, niż stosowanie zewnętrznego zegara RTC (na przykład sterowanego magistralą I²C). Oczywiście wymusza to na nas drobiazgowe przemyślenie stosownych funkcji sterujących, lecz gra warta jest świeczki. Jako że jest to temat ciekawy, poniżej zaprezentuję moduł obsługi zegara RTC nowoczesnych mikrokontrolerów ATtiny 1-series. Na początek plik nagłówkowy, który pokazano na **listingu 1**, a dzięki któremu porządkujemy późniejszy kod źródłowy, czyniąc go bardzo czytelnym – i jednocześnie upraszczamy proces wprowadzania ewentualnych zmian. Plik ten definiuje główne ustawienia sprzętowe oraz wprowadza niezbędne zmienne.

Jak widać, wprowadzono nowy typ danych o nazwie clockType, który integruje w sobie ustawienia zegara czasu rzeczywistego RTC. Dalej, na **listingu 2**, pokazano ciało funkcji, która zwraca wartość maksymalnego indeksu dnia miesiąca dla miesiąca i roku, które są argumentami jej wywołania. Funkcja ta znajduje zastosowanie podczas ustawiania daty zegara RTC.

W tym momencie warto podkreślić, iż konfiguracja RTC wymaga zapisu wartości rejestrów konfiguracyjnych mikrokontrolera, które chronione są przed przypadkową modyfikacją za pomocą rejestru CCP (Configuration Change Protection). Zapis taki musi być poprzedzony wpisem odpowiedniej sygnatury (zezwolenia) do rejestru CCP, po czym, w ciągu 4 taktów zegara mikrokontrolera, musi nastąpić wpisanie wartości modyfikowanego rejestru. Do realizacji takiego zapisu przewidziano funkcję pokazaną na **listingu 3**. Z kolei na **listingu 4** pokazano ciało funkcji konfigurującej układ RTC jako zegar wywołujący przerwanie PIT co 1 sekundę. I na sam koniec, na **listingu 5**, pokazano ciało funkcji realizującej mechanizm zegara RTC.

Tak jak to zostało powiedziane na wstępie, procedura obsługi przerwania zegara RTC wywoływana jest również w trybie Power Down mikrokontrolera (wybudzając go), dzięki czemu możliwa stała się realizacja energooszczędnego zegara RTC wyłącznie przy użyciu zasobów naszego układu, bez udziału peryferiów zewnętrznych.

Schemat montażowy urządzenia pokazano na **rysunku 2**. Przyznać muszę, że w kwestii obwodu drukowanego nie miałem wiele do powiedzenia – jeśli wziąć pod uwagę konieczność jego minimalizacji, a także chęć nawiązania do pierwowzoru. Wszystko to spowodowało, że projekt wymagał sporo uwagi podczas implementacji i nie był tak łatwy do opracowania, jak mogłoby się wydawać. Finalnie powstał obwód drukowany pokazany na rysunku 2,

```
typedef struct
{
    volatile uint8_t Year;  //0..99
    volatile uint8_t Month;  //0..11
    volatile uint8_t Day;  //0..30
    volatile uint8_t Hour;  //0..23
    volatile uint8_t Minute;  //0..59
    volatile uint8_t Second;  //0..59
} clockType;


//Zmienne globalne modułu
extern clockType Clock;

//Wartości maksymalnych indeksów dni poszczególnych miesięcy roku
extern const uint8_t maxDayIndex[];


//Prototypy funkcji
void writeCCP(volatile uint8_t *Register, uint8_t Value);
uint8_t getDayLimit(uint8_t Year, uint8_t Month);
void RTCinit(void);
```

Listing 1. Plik nagłówkowy zegara RTC mikrokontrolerów ATtiny 1-series

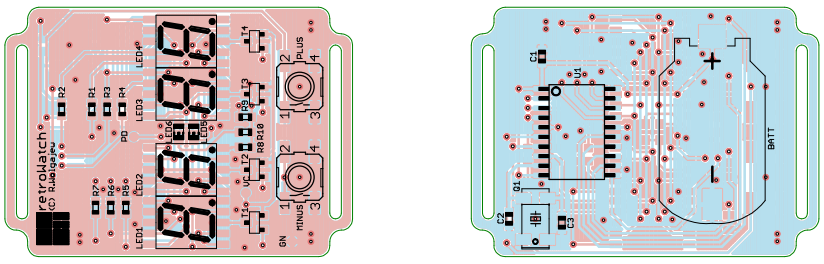
który w zamierzeniach miał odwzorowywać kształtem typowy zegarek naręczny, umożliwiając montaż paska o szerokości 22 mm (poprzez jego przełożenie przez przygotowane „uszy”). Montaż urządzenia rozpoczynamy od warstwy TOP, na której w pierwszej kolejności przylutowujemy diody LED (LED5 i LED6), następnie montujemy wyświetlacze 7-segmentowe LED typu SMD (LED1...LED4), tranzystory sterujące (T1...T4), a na samym końcu elementy bierne oraz przyciski SMD PLUS i MINUS. W tym momencie przechodzimy na warstwę BOTTOM, gdzie w pierwszej kolejności montujemy mikrokontroler o dość

```
uint8_t getDayLimit(uint8_t Year, uint8_t Month)  //Year: 0...99, Month: 0..11
{
    uint8_t dayLimit;

    //Ustalamy maksymalny index dni dla miesiąca i roku będących argumentem funkcji
    dayLimit = pgm_read_byte(&maxDayIndex[Month]);

    //Dla Lutego wyznaczamy wspomniany index w zależności od tego czy rok jest przestępny czy też nie
    if(Month == 1)
    {
        if(((Year+2000)%4 == 0 && (Year+2000)%100 != 0) || (Year+2000)%400 == 0) dayLimit = 28; else dayLimit = 27;
    }

    return dayLimit;
}
```

Listing 2. Funkcja zwracająca wartość maksymalnego indeksu dnia miesiąca dla miesiąca i roku, które są argumentami jej wywołania

```
void writeCCP(volatile uint8_t *Register, uint8_t Value)
{
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
    {
        CCP = CCP_IOREG_gc;  //Zezwolenie na zapis do CCP
        *Register = Value;  //Zapis do chronionego rejestru
    }
}
```

Listing 3. Funkcja realizująca zapis rejestrów mikrokontrolera chronionych przed modyfikacją



Rysunek 2. Schemat montażowy (a – strona TOP, b – strona BOTTOM)

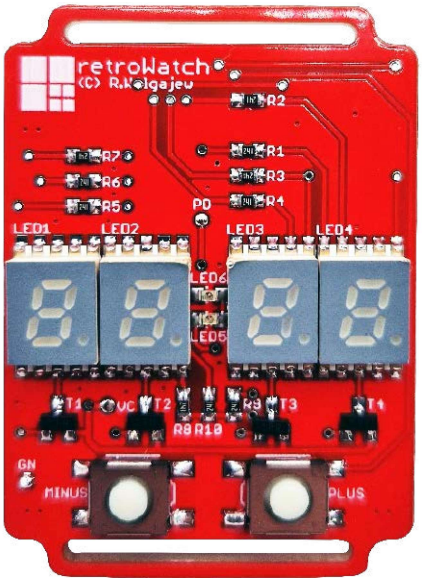
gęstym rastrze wyprowadzeń. Proces ten najłatwiej przeprowadzić przy użyciu stacji lutowniczej na gorące powietrze, jeśli jednak nie dysponujemy tego rodzaju sprzętem, można również zastosować metodę z użyciem typowej stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń układów. Dalej lutujemy elementy bierne, rezonator i – na samym końcu – koszyczek baterii CR2032 (element BATT). Poprawnie zmontowane urządzenie nie wymaga żadnych procedur uruchomieniowych i powinno działać po włączeniu zasilania. Na **fogotrafii 1** pokazano zmontowane urządzenie retroWatch od strony warstwy

```
void RTCinit(void)
{
    uint8_t regValue;

    //Inicjalizujemy oscylator 32.768 Hz. Na początek wyłączamy go, by przeprowadzić niezbędne konfiguracje
    regValue = CLKCTRL.XOSC32KCTRLA;
    regValue &= ~CLKCTRL_ENABLE_bm;
    writeCCP(&CLKCTRL.XOSC32KCTRLA, regValue);
    //Czekamy, aż bit XOSC32KS przyjmie wartość 0 potwierdzając wykonanie zapisu
    while(CLKCTRL.MCLKSTATUS & CLKCTRL_XOSC32KS_bm);


    //Uruchamiamy oscylator 32768 Hz
    regValue = CLKCTRL.XOSC32KCTRLA;
    regValue |= CLKCTRL_ENABLE_bm;
    writeCCP(&CLKCTRL.XOSC32KCTRLA, regValue);

    //Zanim dokonamy jakichkolwiek modyfikacji ustawień zegara RTC czekamy na synchronizację jego rejestrów
    while(RTC.STATUS > 0);
    //Wybieramy rezonator kwarcowy 32.768 Hz, jako źródło taktowania zegara RTC
    RTC.CLKSEL = RTC_CLKSEL_TOSC32K_gc;
    //Czekamy na synchronizację rejestru RTC.PITCTRLA zanim dokonamy jego modyfikacji
    while(RTC.PITSTATUS > 0);
    //Wybieramy preskaler dla zegara RTC powodując, że przerwanie będzie zachodzić co 1s oraz uruchamiamy go
    RTC.PITCTRLA = RTC_PERIOD_CYC32768_gc|RTC_PITEN_bm;
    //Zezwalamy na przerwanie PIT zegara RTC (Periodic Interrupt)
    RTC.PITINTCTRL = RTC_PI_bm;
}
```

Listing 4. Ciało funkcji konfigurującej układ RTC jako zegar wywołujący przerwanie PIT co 1 sekundę

```
ISR(RTC_PIT_vect)
{
    uint8_t dayLimit;

    //Czyścimy flagę przerwania, gdyż nie jest kasowana sprzętowo
    RTC.PITINTFLAGS = RTC_PI_bm;

    //Realizujemy mechanizm zegara RTC
    if(++Clock.Second == 60)
    {
        Clock.Second = 0;
        if(++Clock.Minute == 60)
        {
            Clock.Minute = 0;
            if(++Clock.Hour == 24)
            {
                Clock.Hour = 0;
                //W tym momencie ustalamy maksymalny index dnia dla bieżącego roku i miesiąca
                dayLimit = getDayLimit(Clock.Year, Clock.Month);

                if(++Clock.Day > dayLimit)  //To był ostatni dzień bieżącego miesiąca
                {
                    Clock.Day = 0;
                    if(++Clock.Month == 12)
                    {
                        Clock.Month = 0;
                        if(++Clock.Year == 100) Clock.Year = 0;
                    }
                }
            }
        }
    }
}
```

Listing 5. Ciało funkcji realizującej mechanizm zegara RTC



Fotografia 1. Zmontowane urządzenie retroWatch od strony warstwy TOP

Fotografia 2. Zmontowane urządzenie retroWatch od strony warstwy BOTTOM

TOP, zaś na **fotografii 2** – od strony warstwy BOTTOM.

Obsługa

Przejdźmy zatem do tematu obsługi naszego urządzenia. Tu sprawa jest niezmiernie prosta. Standardowo zegarek większość czasu przebywa w stanie uśpienia, oszczędzając energię baterii zasilającej. Wybudzenie go następuje w dwóch przypadkach:

1. Na skutek wystąpienia przerwania zegara RTC wywoływanego co 1 sekundę, podczas którego mikrokontroler realizuje programową implementację zegara czasu rzeczywistego i po zakończeniu którego ponownie przechodzi do trybu uśpienia.
2. Na skutek naciśnięcia któregokolwiek z przycisków funkcyjnych (PLUS lub MINUS), po którym to mikrokontroler zaczyna prezentować na wyświetlaczu LED informacje stosowne do bieżącego ustawienia menu.

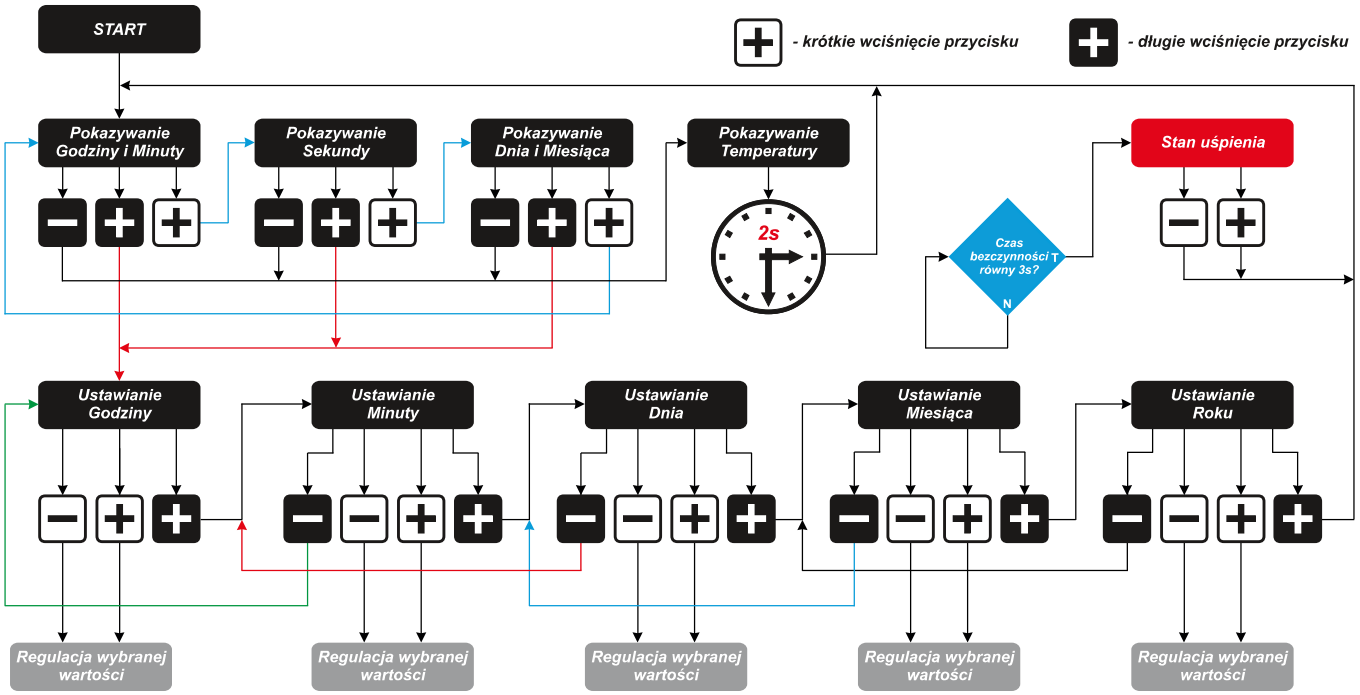
Wybudzenie urządzenia powoduje przejście do funkcji jego obsługi. Jako że chciałem, by urządzenie retroWatch naśladowało funkcjonalność pierwowzoru produkcji Unitry Warel, nie zdecydowałem się (mimo dostępnej, wolnej pamięci Flash) na wzbogacenie jego funkcjonalności – w związku z czym dostępne są wyłącznie 3 podstawowe tryby pracy zegarka, pokazujące na wyświetlaczu LED następujące informacje:

1. Godzinę i minutę,
2. Sekundę,
3. Dzień i miesiąc roku.

Przechodzenie pomiędzy wspomnianymi trybami (w pętli) następuje na skutek krótkiego naciśnięcia przycisku PLUS. Długie naciśnięcie przycisku PLUS powoduje z kolei przejście urządzenia w tryb ustawień, w którym konfiguracji podlegają następujące dane: godzina, minuta, dzień, miesiąc i rok. W tym trybie pracy urządzenia bieżąca funkcjonalność wspomnianych przycisków funkcyjnych zależy w dużej mierze od miejsca w systemie menu, w jakim się ono znajduje. Biorąc pod uwagę, że rozpoznawane jest zarówno krótkie, jak i długie naciśnięcie każdego z nich, nie sposób skrótowo opisać tego zagadnienia – w związku z czym na **rysunku 5** pokazano diagram obrazujący sposób obsługi zegarka retroWatch oraz dostępne opcje menu.

Warto również podkreślić, że zegarek wyposażono dodatkowo w mechanizm wykrywania bezczynności po stronie użytkownika (co ważne, nieaktywny w trybie konfiguracji), który powoduje przejście do stanu uśpienia po 3 sekundach braku obsługi. Poza tym, odchodząc niejako od funkcjonalności pierwowzoru (i tu przepraszam konserwatywnych użytkowników), wyposażyłem układ w opcjonalną możliwość pomiaru temperatury za pomocą zewnętrznego termometru scalonego typu LM35, podłączonego do portu UPDI/RESET (PA0) mikrokontrolera w sposób pokazany na **rysunku 6**.

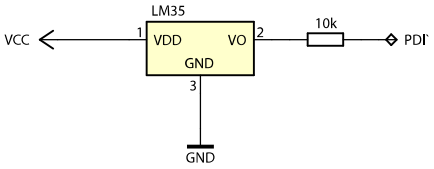
Celowo nie przewidziałem na płytce urządzenia miejsca na tego rodzaju termometr scalony, gdyż chciałem zachować minimalistyczny i zgodny z oryginałem charakter czasomierza. Uważny Czytelnik dostrzeże jednak w tym momencie pewną sprzeczność, jeśli chodzi o wybór wyprowadzenia UPDI/RESET jako realizującego obsługę opcjonalnego termometru. I słusznie. Wyprowadzenie, o którym mowa, standardowo skonfigurowane jest do obsługi programowania mikrokontrolera (dzięki stosownemu ustawieniu fuse-bitu FUSE.SYSCFG0.RSTPINCFG[1:0]) i w takim wypadku nie nadaje się do realizacji jakiejkolwiek funkcjonalności czy też magistrali danych. Co prawda nawet w takiej (domyślnej) konfiguracji możliwe jest odczytywanie zarówno stanu tegoż wyprowadzenia, jak i napięcia na współdzielonym kanale AIN0 przetwornika ADC (mając w pamięci, że pin podciągnięty jest do VCC), lecz już niemożliwe byłoby manipulowanie kierunkiem tego pinu bądź występującym na nim stanem logicznym. W takim wypadku, aby możliwa stała się realizacja obsługi termometru LM35, konieczne jest przestawienie (za pomocą zaprogramowania fuse-bitu FUSE.SYSCFG0.RSTPINCFG[1:0]) funkcji wyprowadzenia UPDI/RESET jako normalnego portu I/O. Jednak należy mieć na uwadze, że po wykonaniu takiej operacji wejście w tryb programowania wymagać będzie programatora HV (wysokonapięciowego), w związku z czym w pierwszej kolejności należy wgrać wsad do pamięci Flash mikrokontrolera, a dopiero w drugim



Rysunek 3. Diagram obrazujący sposób obsługi układu oraz dostępne opcje menu

kroku przestawić fuse-bit FUSE.SYSCFG0. RSTPINCFG[1:0]. Tyle w kwestii obsługi. Na koniec kilka słów komentarza należy się tematyce energooszczędności urządzenia i czasu pracy na jednej baterii CR2032 o typowej pojemności w granicach 240 mAh. Aby ocenić, jak długo urządzenie retroWatch pracować będzie na pojedynczej baterii CR2032, należy zastanowić się, z jakich etapów składa się cykl jego pracy i jakiej wielkości prądy pobiera wtedy ze źródła napięcia zasilającego. Przystępując do obliczeń, przyjąłem następujący podział cyklu pracy urządzenia:

- czas trybu Power Down (uśpienia), który trwa z dużym przybliżeniem 24 h/dobę i podczas którego pobierany jest prąd rzędu 0,8 µA,
- czas obsługi przerwania od zegara RTC, który trwa średnio około 40 µs i podczas którego pobierany jest prąd rzędu 500 µA (prąd mikrokontrolera w trybie Active Mode),
- czas aktywnej pracy urządzenia (załączenia wyświetlacza LED), który trwa 3 s i podczas którego pobierany jest prąd rzędu 22 mA.



Rysunek 4. Sposób podłączenia czujnika temperatury do portu UPDI mikrokontrolera

Założyłem ponadto, że wybudzanie urządzenia (za pomocą klawiatury) i następująca po nim aktywność wyświetlacza LED zachodzi 30 razy na dobę. Przy założeniach jak wyżej otrzymałem teoretyczny, ponad 14-miesięczny czas pracy na pojedynczej baterii CR2032, co wydaje się wartością rewelacyjną (i – w gruncie rzeczy – podobną do pierwowzoru firmy Unitra Warel). Aby zorientować się, w jaki sposób dokonałem stosownych obliczeń, które pozwoliły mi oszacować czas pracy na baterii CR2032, należy spojrzeć do **tabeli 1**, w której zgromadzono wszystkie niezbędne dane.

I na koniec, wyłącznie dla porządku dodam, że mikrokontroler monitoruje na bieżąco poziom napięcia zasilania układu (w czasie jego aktywności), a w przypadku jego nieodpowiedniej wartości (czyli poniżej 1,8 V) wyświetla komunikat „BATT”, po czym usypia urządzenie.

Robert Wołgajew, EP

Tabela 1. Dane wejściowe niezbędne do oceny czasu pracy urządzenia na jednej baterii CR2032

Tryb pracy	Czas [s]	Prąd [mA]	Zużyta energia [mAh]	Liczba cykli pracy na dobę	Sumaryczna zużyta energia [mAh]	Uwagi
Uśpienie (Power Down)	86400	0,0008	0,0192	1	0,0192	24 h/dobę
Wyświetlacz LED załączony	3	22	0,018	30	0,55	
Obsługa zegara RTC (przerwania)	0,00004	0,5	$5{,}5 \cdot 10^{-9}$	86400	0,0005	86400 cykli (co 1 sekundę) po 40 µs (średni czas obsługi)
				SUMA na dobę [mAh]	**0,57**	