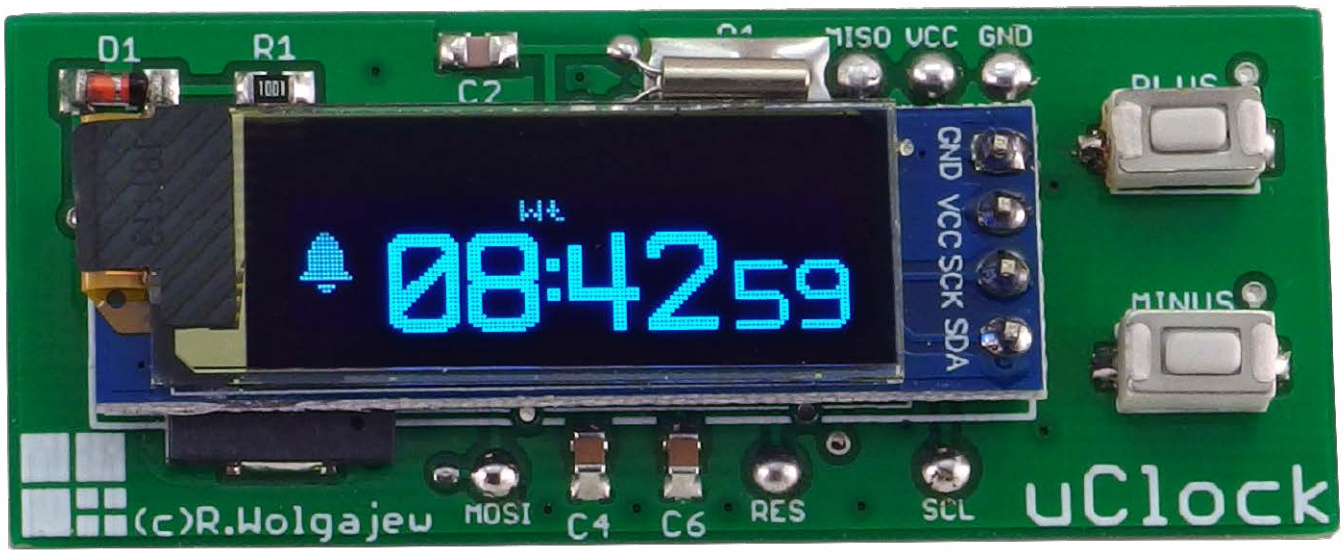


µClock

Zegar to w portfolio każdego elektronika konstruktora przysłowiowe „must have”. Nie ma w tym nic dziwnego, wszak są to urządzenia dość proste, lecz dające dużo niekłamanej satysfakcji podczas własnoręcznej implementacji.

Zegara nie mogło zabraknąć również wśród moich konstrukcji. Postanowiłem jednak powiesić poprzeczkę nieco wyżej i zaprojektować urządzenie wyróżniające się następującymi cechami użytkowymi:

- prostota konstrukcji,
- niska cena implementacji,
- łatwość montażu,
- atrakcyjny wygląd,
- duża funkcjonalność,
- duża ergonomia obsługi.

Niektóre z tych cech wydają się przeciwstawne i z pozoru trudno wyobrazić sobie ich wzajemne współistnienie w ramach jednego projektu, lecz uwierzcie mi na słowo – zaprezentowany poniżej system mikroprocesorowy µClock integruje wszystkie wspomniane powyżej założenia konstrukcyjne.

Pierwszym z dylematów, z jakim miałem się zmierzyć, konstruując to urządzenie był wybór zegara czasu rzeczywistego (RTC). Początkowo wydawało się, iż tego rodzaju

Dodatkowe materiały do pobrania ze strony www.media.avt.pl

W ofercie AVT* AVT-5805

Podstawowe parametry:
- zegar, stoper, kalendarz i budzik,
- prostota i niska cena konstrukcji,
- duża ergonomia obsługi i atrakcyjny wygląd,
- zasilanie 3…5 V, 10 mA.

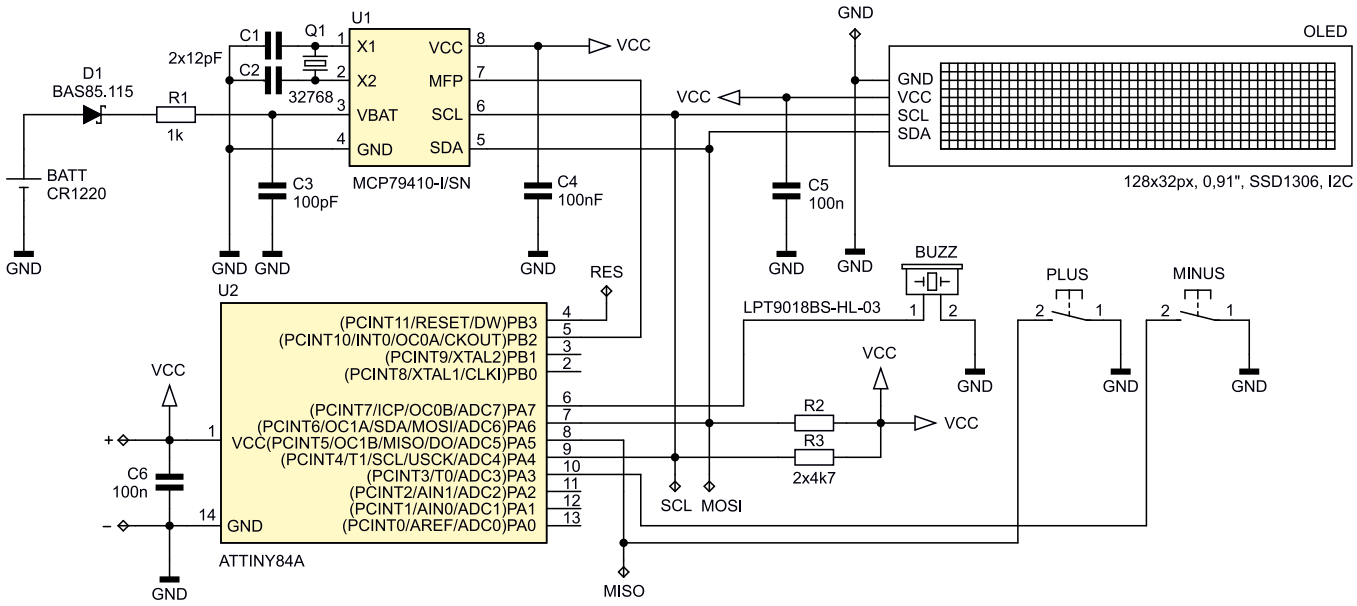
Projekty pokrewne na www.media.avt.pl:

AVT-5735	Estetyczny zegar (EP 1/2020)
AVT-5677	Zegar ClockRDS (EP 6/2019)
AVT-5640	Rozbudowany zegar (EP 7/2018)
AVT-5522	Zegar ustawiany za pomocą GPS (EP 9/2015)

*** 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 ***http://sklep.avt.pl***, osoby zainteresowane zakupem płytek drukowanych (PCB) prosimy o kontakt via e-mail: kity@avt.pl.



Rysunek 1. Schemat ideowy urządzenia

peryferium z powodzeniem możemy zrealizować programowo przy współudziale wbudowanego w strukturę mikrokontrolera układu czasowo-licznikowego Timer2, który w przypadku wielu mikrokontrolerów z rodziny AVR może pracować w trybie asynchronicznym, realizując zadanie zegara czasu rzeczywistego. Jednak w przypadku takiej implementacji nierozwiązany pozostaje problem podtrzymania zasilania układu RTC w czasie braku zasilania systemu mikroprocesorowego. Zresztą samo podtrzymanie bateryjne nie rozwiązuje w tym przypadku problemu, gdyż co prawda asynchroniczny układ czasowo-licznikowy Timer2 może pracować w trybie głębokiego uśpienia mikrokontrolera (pobierającego w tym stanie pracy niewielki prąd z podtrzymującej baterii zasilającej), lecz nie znaczy to, że z automatu realizuje on w tym czasie zadania typowego układu RTC. Aby wspomniany Timer2 realizował zadania sprzętowego układu RTC, potrzebny jest odpowiedni fragment kodu obsługi aplikacji (obsługi przerwania Timera2) a więc zachodzi konieczność pracy mikrokontrolera. Można to oczywiście ominąć, wybudzając mikrokontroler co 1 s i w krótkim czasie po przebudzeniu realizować treść obsługi przerwania Timera2, czyli de facto realizując zadania programowego zegara RTC, po czym wracać do stanu uśpienia, lecz nie jest to moim zdaniem rozwiązanie optymalne, a już na pewno nie jest proste w implementacji. Ostatecznie i biorąc pod uwagę fakt, że w konstrukcji urządzenia µClock wykorzystuję magistralę I²C do współpracy z zastosowanym wyświetlaczem OLED, a więc część kodu obsługi sprzęgu I²C i tak musi się znaleźć w treści kodu naszej aplikacji, zdecydowałem, że jako układu RTC użyję zewnętrznego, taniego i wygodnego w implementacji układu firmy Microchip typu MCP-79410. Ma on ważną zaletę – obsługuje automatyczny mechanizm podtrzymania bateryjnego zegara RTC, pobierając w takim trybie pracy prąd rzędu kilkuset nA.

Wykaz elementów:
Rezystory: (SMD0805)
R1: 1 kΩ
R2, R3: 4,7 kΩ

Kondensatory: (SMD 0805)
C1, C2: 12 pF
C3: 100 pF
C4…C6: 100 nF

Półprzewodniki:
U1: MCP79410-I/SN (SO08)
U2: ATtiny84A (SOIC14)
D1: BAS85.115 (MINIMELF)
OLED: wyświetlacz OLED 128×32px, 0,91", sterownik SSD1306, magistrala I2C, wymiary 38×12 mm

Pozostałe:
Q1: rezonator kwarcowy zegarkowy 32768 Hz
PLUS, MINUS: mikroprzycisk SMD typu DTSM31NB
BATT: gniazdo baterii CR1220 typu CONNFLY DS1092-12-N8S
BUZZ: przetwornik piezoelektryczny typu LPT9018BS-HL-03-4.0-12-R
VCC: bateria CR1220

Co więcej, układ ten cechuje się następującymi, wybranymi walorami użytkowymi:

- obsługa dokładnego zegara RTC oraz kalendarza do roku 2399,
- możliwość regulacji dokładności wbudowanego oscylatora w zakresie ±129 ppm,
- funkcja automatycznego podtrzymania zasilania bateryjnego ustawień RTC,
- funkcja automatycznego znacznika czasu wypadnięcia/przywrócenia zasilania,
- wielofunkcyjne wyjście MFP wyposażone w możliwość generowania przebiegów prostokątnych o wybranej częstotliwości,
- wbudowana pamięć RAM o wielkości 64 B podtrzymywana bateryjnie,
- wbudowana pamięć EEPROM o wielkości 1 kB,
- szeroki zakres napięć zasilających 1,8...5,5 V.

Budowa i działanie

Układ MCP-79410 idealnie wpisuje się w wymagania naszej aplikacji, w związku z tym przejdźmy od razu do schematu aplikacyjnego urządzenia µClock, który pokazano na **rysunku 1**. Zaprojektowano bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler firmy Microchip o oznaczeniu ATtiny84A. Realizuje on kilka kluczowych zadań w ramach programu obsługi aplikacji, a mianowicie: steruje pracą wbudowanego wyświetlacza OLED o organizacji 128×32 piksele stanowiącego element interfejsu użytkownika, angażując w tym celu interfejs I²C, zarządza współpracą ze wspomnianym wcześniej scalonym układem RTC typu MCP-79410 angażując, jak poprzednio, ten sam sprzęg I²C oraz obsługuje dwa przyciski funkcyjne PLUS i MINUS oraz buzzer piezoelektryczny BUZZ, będące elementami interfejsu użytkownika. Warto w tym miejscu zaznaczyć, że obsługa przycisków funkcyjnych odbywa się bez jakichkolwiek opóźnień (typu *_delay_ms*), gdyż korzysta z wbudowanego w strukturę mikrokontrolera układu czasowo-licznikowego Timer1 pracującego w trybie CTC. Jego zadaniem jest generowanie i obsługa sprzętowych opóźnień w ramach funkcji obsługi przycisków, której zadaniem jest, po pierwsze, eliminacja niepożądanego zjawiska drgania styków (*debouncing*), a po drugie, obsługa długiego i krótkiego naciśnięcia, co wykorzystywane jest w ramach interfejsu użytkownika urządzenia µClock.

Kolejnym peryferium, jakie wykorzystuje nasz program obsługi aplikacji, jest układ czasowo-licznikowy Timer0 pracujący również w trybie CTC. Jest on odpowiedzialny za generowanie przebiegu prostokątnego o częstotliwości 4 kHz na wyjściu OC0B (PA7) mikrokontrolera, który zasila buzzer piezoelektryczny odpowiedzialny za efekty dźwiękowe (w funkcji budzika).

Ostatnim elementem mikrokontrolera, z którego dobrodziejstw korzysta program obsługi aplikacji, jest przerwanie zewnętrzne INT0 wyzwalane opadającym zboczem na wyprowadzeniu PB2 mikrokontrolera. Do tego wyprowadzenia mikrokontrolera podłączono wielofunkcyjne wyprowadzenie

Listing 1. Plik nagłówkowy modułu obsługi układu MCP-79410

```
//Definicje typów strukturalnych
//do obsługi zegara RTC
typedef struct{
    uint8_t Hour;  //0...23
    uint8_t Minute;  //0...59
    uint8_t Second;  //0...59
}timeType;

typedef struct{
    uint8_t weekDay;  //1...7
    uint8_t Day;  //1...31
    uint8_t Month;  //1...12
    uint8_t Year;  //0...99
}dateType;

//Definicje adresów układu MCP79410
//w trybie zapisu/odczytu
#define MCP79410_WRITE_ADDR 0xDE
#define MCP79410_READ_ADDR 0xDF

//Definicje najważniejszych rejestrów
//sterujących i ich właściwości
#define TIME_START_REG 0x00
    #define START_OSCILLATOR (1<<7)
    #define STOP_OSCILLATOR (0<<7)

#define DATE_START_REG 0x03
    #define VBAT_ENABLE (1<<3)
    #define VBAT_DISABLE (0<<3)

#define CONTROL_REG 0x07
    #define MFP_AS_OUTPUT_1 (1<<7)
    #define MFP_AS_OUTPUT_0 (0<<7)
    #define MFP_AS_SQUARE_OUTPUT (1<<6)
    #define MFP_AS_NORMAL_OUTPUT (0<<6)
    #define NO_ALARMS_ACTIVE (0<<4)
    #define ALARM0_ACTIVE (1<<4)
    #define ALARM1_ACTIVE (2<<4)
    #define BOTH_ALARMS_ACTIVE (3<<4)
    #define EXTERNAL_OSCILLATOR (1<<3)
    #define INTERNAL_OSCILLATOR (0<<3)
    #define SQUARE_1HZ 0x00
    #define SQUARE_4096HZ 0x01
    #define SQUARE_8192HZ 0x02
    #define SQUARE_32768HZ 0x03

#define RAM_START_REG 0x20
```

Listing 2. Funkcja odpowiedzialna za konfigurację układu MCP-79410

```
void RTCinit(uint8_t Settings){
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy rejestru ustawień zegara RTC
    i2cWriteByte(CONTROL_REG);
    //Ustawienia zegara RTC
    i2cWriteByte(Settings);
    i2cStop();
}
```

MFP układu RTC MCP-79410, na którym, po odpowiednim skonfigurowaniu, pojawia się bardzo dokładny sygnał 1 Hz niezbędny w programie obsługi aplikacji. Po pierwsze, służy do cyklicznego odświeżania ekranu interfejsu użytkownika, zaś pod drugie, obsługuje mechanizm programowego stopera będącego jedną z funkcji urządzenia.

Tyle w kwestiach konstrukcyjnych. Przejdźmy do zagadnień programowych, a te skupią się głównie na obsłudze naszego zegara RTC. Rozpoczniemy od pliku nagłówkowego, który jak zawsze staram się napisać w taki sposób, aby bez lektury dokumentacji układu możliwe było określenie niezbędnych funkcji peryferium. Zawartość pliku pokazano na **listingu 1**. Wprowadzono dwa dodatkowe typy strukturalne (*timeType, dateType*) odpowiedzialne za przechowywanie i przetwarzanie czasu i daty wbudowanego zegara RTC.

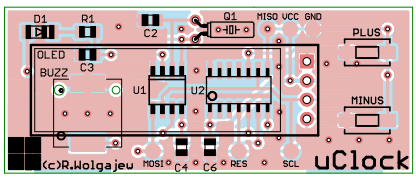
Na **listingu 2** pokazano funkcję, której zadaniem jest konfiguracja cech sprzętowych zegara RTC. Argument funkcji decyduje o ustawieniach zegara (np. ustawieniach wyjścia MFP) i w przypadku naszego urządzenia przyjmuje wartość:

```
MFP_AS_SQUARE_OUTPUT|NO_
ALARMS_ACTIVE|INTERNAL_
OSCILLATOR|SQUARE_1HZ;
```

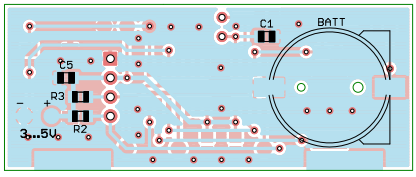
Kontynuując, na **listingu 3** pokazano proste funkcje narzędziowe umożliwiające odczyt i zapis czasu zegara RTC, zaś na **listingu 4** bliźniacze funkcje odczytu i zapisu, ale tym razem daty.

Dalej, korzystając z faktu, że nasze peryferium wyposażono w 64 bajty pamięci RAM podtrzymywanej bateryjnie, na **listingu 5** zostały pokazane proste funkcje umożliwiające odczyt i zapis wbudowanej pamięci RAM układu. Analogicznie można zaimplementować podobne funkcje, lecz tym razem przeznaczone do obsługi pamięci EEPROM wbudowanej w układ MCP-79410. Należy jednak w tym przypadku mieć na uwadze dwie istotne cechy. Po pierwsze, układ MCP-79410 ma w tym wypadku inny adres w przestrzeni adresowej magistrali I²C, niż ma to miejsce w przypadku ustawień RTC i RAM, zaś pod drugie, zapis do pamięci EEPROM, jak to zwykle bywa, zajmuje pewien określony i wcale nie taki krótki czas, co należy wziąć pod uwagę, aplikując stosowne funkcje. W naszym programie obsługi aplikacji pamięć EEPROM układu nie była wykorzystywana, w związku z czym nie istniała potrzeba implementacji stosownych funkcji narzędziowych.

Na koniec kilka słów na temat obsługi stopera, jako że ta funkcja została zaimplementowana programowo z użyciem



Rysunek 2. Schemat płytki PCB wraz z rozmieszczeniem elementów, strona TOP



Rysunek 3. Schemat płytki PCB wraz z rozmieszczeniem elementów, strona BOTTOM

Ustawienia Fuse-bitów:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1
EESAVE: 0

Listing 3. Funkcje przeznaczone do odczytu i zapisu czasu układu MCP-79410

```
void RTCwriteTime(timeType *Time){
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy rejestru czasu
    //(w tym przypadku sekund)
    i2cWriteByte(TIME_START_REG);
    //Sekundy w zapisie BCD + start oscylatora
    i2cWriteByte(((Time->Second/10)<<4)|(Time->Second%10)|START_OSCILLATOR);
    //Minuty w zapisie BCD
    i2cWriteByte(((Time->Minute/10)<<4)|(Time->Minute%10));
    //Godziny w zapisie BCD (standardowo zapis 24-godzinny)
    i2cWriteByte(((Time->Hour/10)<<4)|(Time->Hour%10));
    i2cStop();
}

void RTCreadTime(timeType *Time){
    uint8_t readByte;
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy rejestru czasu
    //w tym przypadku sekund)
    i2cWriteByte(TIME_START_REG);
    //I2C Restart
    i2cRestart();
    //Adres MCP79410 do odczytu
    i2cWriteByte(MCP79410_READ_ADDR);
    //Sekundy w zapisie BCD - maskujemy bit
    //pracującego oscylatora (bit7)
    readByte = i2cReadByte(ACK) & 0x7F;
    Time->Second = ((readByte>>4)*10) + (readByte&0x0F);
    //Minuty w zapisie BCD
    readByte = i2cReadByte(ACK);
    Time->Minute = ((readByte>>4)*10) + (readByte&0x0F);
    //Godziny w zapisie BCD (standardowo zapis 24-godzinny)
    readByte = i2cReadByte(NACK);
    Time->Hour = (((readByte&0x30)>>4)*10) + (readByte&0x0F);
    i2cStop();
}
```

Listing 4. Funkcje przeznaczone do odczytu i zapisu daty układu MCP-79410

```
void RTCwriteDate(dateType *Date){
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy rejestru daty
    //(w tym przypadku dni tygodnia)
    i2cWriteByte(DATE_START_REG);
    //Dzień tygodnia +
    //aktywacja podtrzymania bateryjnego
    i2cWriteByte(Date->weekDay|VBAT_ENABLE);
    //Dzień w zapisie BCD
    i2cWriteByte(((Date->Day/10)<<4)|(Date->Day%10));
    //Miesiąc w zapisie BCD
    i2cWriteByte(((Date->Month/10)<<4)|(Date->Month%10));
    //Rok w zapisie BCD
    i2cWriteByte(((Date->Year/10)<<4)|(Date->Year%10));
    i2cStop();
}

void RTCreadDate(dateType *Date){
    uint8_t readByte;
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy rejestru daty
    //(w tym przypadku dni tygodnia)
    i2cWriteByte(DATE_START_REG);
    //I2C Restart
    i2cRestart();
    //Adres MCP79410 do odczytu
    i2cWriteByte(MCP79410_READ_ADDR);
    //Dzień tygodnia
    Date->weekDay = i2cReadByte(ACK) & 0x07;
    //Dzień w zapisie BCD
    readByte = i2cReadByte(ACK);
    Date->Day = ((readByte>>4)*10) + (readByte&0x0F);
    //Miesiąc w zapisie BCD
    readByte = i2cReadByte(ACK);
    Date->Month = (((readByte & 0x10)>>4)*10) + (readByte&0x0F);
    //Rok w zapisie BCD
    readByte = i2cReadByte(NACK);
    Date->Year = ((readByte>>4)*10) + (readByte&0x0F);
    i2cStop();
}
```

Listing 5. Funkcje przeznaczone do odczytu i zapisu wbudowanej pamięci RAM układu MCP-79410

```
void RTCwriteRAM(uint8_t *dataToWrite, uint8_t Bytes){
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy wbudowanej pamięci RAM zegara RTC
    i2cWriteByte(RAM_START_REG);
    while(Bytes--) i2cWriteByte(*dataToWrite++);
    i2cStop();
}

void RTCreadRAM(uint8_t *dataToRead, uint8_t Bytes){
    i2cStart();
    //Adres MCP79410 do zapisu
    i2cWriteByte(MCP79410_WRITE_ADDR);
    //Adres startowy wbudowanej pamięci RAM zegara RTC
    i2cWriteByte(RAM_START_REG);
    //I2C Restart
    i2cRestart();
    //Adres MCP79410 do odczytu
    i2cWriteByte(MCP79410_READ_ADDR);
    while(Bytes--) *dataToRead++ = i2cReadByte(Bytes? ACK:NACK);
    i2cStop();
}
```

Listing 6. Funkcja obsługi przerwania INT0 odpowiedzialna za realizację programowego stopera

```
//Przerwanie zachodzi co 1s,
//gdyż wyzwalane jest sygnałem 1Hz z układu RTC
ISR(INT0_vect){
    //Obsługa stopera
    if(Timer.Activity) if(++Timer.Second == 60){
        Timer.Second = 0;
        if(++Timer.Minute == 100) Timer.Minute = 0;
    }
}
```

Listing 7. Deklaracja typu programowego stopera

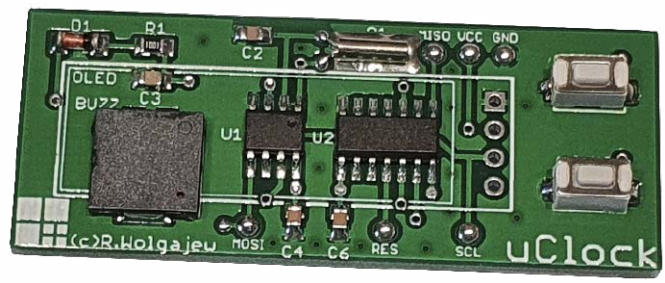
```
//Definicja typu strukturalnego Timera
typedef struct{
    uint8_t Minute;     //0...99
    uint8_t Second;     //0...59
    uint8_t Activity;   //0/1
}timerType;
```

przerwania INT0 mikrokontrolera wyzwalanego sygnałem 1 Hz z układu RTC. Kod tej prostej funkcji pokazano na **listingu 6**, korzysta ona z nowego typu strukturalnego, którego deklarację pokazano na **listingu 7.**

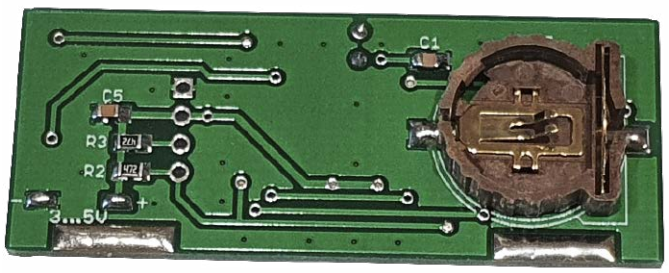
Montaż i uruchomienie

Schemat montażowy urządzenia został pokazany na **rysunkach 2 i 3**. Zaprojektowano niewielki, dwustronny obwód drukowany z zastosowaniem elementów o niezbyt wymagających obudowach, co w założeniu miało uprościć jego implementację, spełniając jedno z założeń konstrukcyjnych.

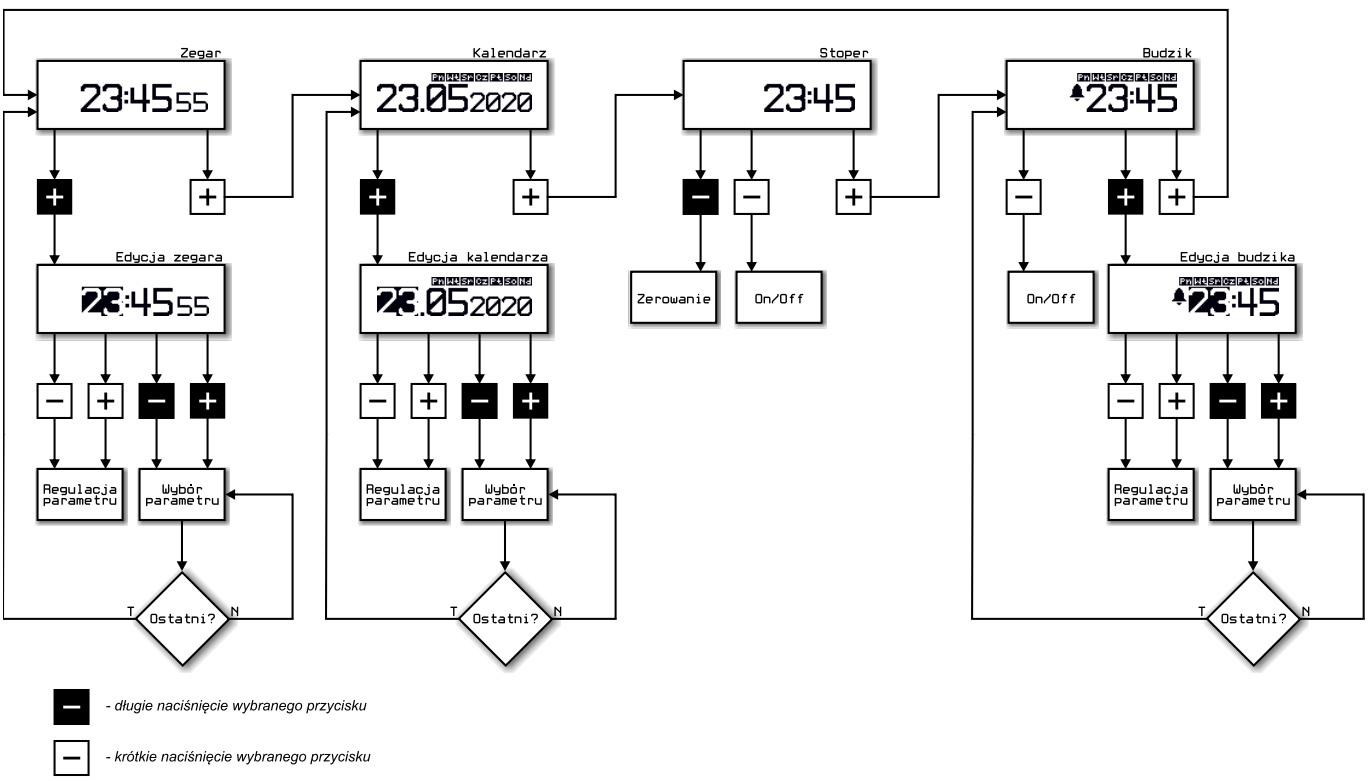
Montaż rozpoczynamy od warstwy TOP, gdzie w pierwszej kolejności montujemy wszelkiego rodzaju półprzewodniki. Następnie przylutowujemy elementy bierne



Fotografia 4. Zmontowane urządzenie tuż przed przylutowaniem wyświetlacza OLED, widok od strony warstwy TOP



Fotografia 5. Zmontowane urządzenie od strony warstwy BOTTOM



Rysunek 6. Diagram obrazujący sposób obsługi urządzenia µClock (w tym funkcjonalność przycisków funkcyjnych)

(w tym rezonator kwarcowy) a na końcu przyciski PLUS i MINUS oraz buzzer piezoelektryczny BUZZ. Dalej przechodzimy na warstwę BOTTOM, gdzie podobnie jak poprzednio, montujemy wszelkie elementy bierne, a na końcu koszyk baterii podtrzymującej zasilanie BATT.

Następnie wracamy na warstwę TOP, gdzie montujemy moduł wyświetlacza OLED, zwyczajnie lutując jego wyprowadzenia typu goldpin do stosownych otworów na obwodzie drukowanym, co jednocześnie zapewnia mu wystarczająco stabilny montaż mechaniczny. Wykonując to zadanie, musimy jednak zwrócić uwagę na wzajemne położenie płaszczyzny wyświetlacza w stosunku do zamontowanych wcześniej przycisków funkcyjnych.

Zasilanie urządzenia podłączamy za pomocą przewodów przylutowanych od strony warstwy BOTTOM w miejscach oznaczonych +/–. Należy pamiętać, by nie przekroczyć dopuszczalnej wartości napięcia zasilającego (5 V), gdyż mogłoby to uszkodzić nasze urządzenie. Poprawnie zmontowany układ nie wymaga żadnych regulacji i powinien działać tuż po włączeniu zasilania.

Na obwodzie drukowanym urządzenia, przy jego dolnej krawędzi, po stronie BOTTOM przewidziano dwa szerokie pola lutownicze (w kształcie prostokątów) niepokryte maską lutowniczą. Można do nich przylutować kawałek laminatu stanowiący niejako stopkę urządzenia. Na **fotografii 4** pokazano zmontowane urządzenie (od strony TOP) tuż przed przylutowaniem wyświetlacza OLED, zaś na **fotografii 5** pokazano to samo urządzenie od strony BOTTOM.

Obsługa

Konstruując program obsługi aplikacji, kierowałem się jednym z początkowych założeń, a mianowicie prostotą obsługi. Dlatego zdecydowałem się na zastosowanie wyłącznie dwóch przycisków sterujących, oznaczonych umownie PLUS i MINUS. Dzięki zaimplementowaniu obsługi krótkiego i długiego wciśnięcia przycisków możliwe stało się znaczne uproszczenie sposobu obsługi urządzenia, nie wpływając negatywnie na ergonomię lub, co gorsza, na funkcjonalność urządzenia.

Urządzenie µClock dysponuje czterema, w pełni konfigurowalnymi, funkcjonalnościami:

- zegara z sekundnikiem,
- kalendarza z dniem tygodnia,
- stopera,
- budzika z planem tygodniowym.

Wszystkie te funkcjonalności obsługuje się w bardzo intuicyjny sposób za pomocą wyłącznie dwóch klawiszy funkcyjnych. Klawisze przyjmują różne funkcjonalności zależne od trybu pracy, w jakim urządzenie się znajduje. Sposób obsługi wraz z zaznaczeniem funkcji każdego z klawiszy (PLUS i MINUS) pokazano na **rysunku 6**.

Warto podkreślić, że konstruując graficzny interfejs użytkownika (GUI) naszego urządzenia, kładłem bardzo duży nacisk na przejrzystość prezentowanych informacji i ich walory estetyczne. Właśnie dlatego posłużono się wyłącznie dwoma krojami czcionek ekranowych o prostej i eleganckiej formie, co w połączeniu z niezaprzeczalnymi zaletami zastosowanego wyświetlacza OLED postawiło na dość wysokim poziomie osiągnięty efekt końcowy. Całość prezentuję się po prostu ładnie, a ergonomia i łatwość obsługi urządzenia potęgują te walory.

Robert Wołgajew, EP