Watch

DODATKOWE MATERIAŁY NA FTP:
ftp://ep.com.pl
USER: 07643, PASS: 332wwppm

W ofercie AVT*
AVT-5525 A, UK

Podstawowe informacje:
- Napięcie zasilania: 5 V (USB).
- Maksymalny prąd obciążenia (wyświetlacz załączony/wyłączony): 20 mA/1 mA.
- Średni czas pracy na zasilaniu akumulatorowym: 7 dni.
- Zegar, kalendarz, stoper, budzik (z planem tygodniowym), barometr, termometr, prognoza pogody.
- Płytka przystosowana do zamocowania paska zegarka.

Projekty pokrewne na FTP:
(wymienione artykuły są w całości dostępne na FTP)

AVT-5522	Zegar ustawiany za pomocą GPS (EP 9/2015)
AVT-3132	Prosty zegar LED (EdW 7/2015)
AVT-5377	Mega stoper – wielofunkcyjny licznik, nie tylko czasu (EP 12/2012)
AVT-513	Zegar ze stuletnim kalendarzem i termometrem (EP 10-11/2011)
AVT-5281	„Inteligentny" zegar z wyświetlaczem LED (EP 3/2011)
AVT-5273	Zegar cyfrowy z analogowym sekundnikiem (EP 1/2011)
AVT-2849	Tiny clock (EdW 1/2008)
AVT-2721	Mikroprocesorowy zegar (EdW 4/2004)
AVT-2632	Gigantyczny zegar (EdW 5/2002)
AVT-5022	Programowany zegar z DCF77 (EP 6-7/2001)
AVT-5002	Zegar cyfrowy z wyświetlaczem analogowym (EP 3/2001)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). http://sklep.avt.pl

Zegarek naręczny (2)

Współczesny świat pędzi szybciej, niż kiedykolwiek wcześniej. Szybki rozwój technologii, elektroniki i najnowsze obszary ich zastosowań powodują, że nie wyobrażamy sobie już życia bez wielu urządzeń, które jeszcze do niedawna uważalibyśmy za zbędne gadżety. ***Rekomendacje:*** *własnoręcznie wykonany smartwatch przyciągnie spojrzenie wielu ciekawskich oczu.*

Jako ostatnią omówimy funkcję obsługi przerwania od porównania zawartości licznika Timer2 z zawartością rejestru OCR2A tegoż modułu, która to wywoływana co 10ms (czyli 100 razy na sekundę) realizuje całą, założoną funkcjonalność programowego zegara RTC. Funkcję pokazano na **listingu 12**. Wspomniana powyżej funkcja obsługi przerwania *TIMER2_COMPA_vect* korzysta ze zmiennych globalnych modułu, których definicje przedstawiają się następująco:*timerType Timer*; *clockType*Clock; *alarmType Alarm*;

To tyle, jeśli chodzi o realizację zegara czasu rzeczywistego z wykorzystaniem asynchronicznego trybu pracy układu czasowo-licznikowego Timer2. Prawda, że proste? Pora na przedstawienie ostatniego elementu naszego urządzenia, którym jest scalony barometr Bosch BMP180. Jego parametry użytkowe idealnie wpisują się w założenia projektu, a jedynym problemem, który możemy napotkać stosując wspomniany element, jest jego dość niewygodna w montażu ręcznym obudowa LGA7. Jak to zwykle bywa, obsługa elementów tego typu polega na zapisie/odczycie wielu, specjalnych rejestrów konfiguracyjnych lub też rejestrów danych, przy których udziale, po pierwsze, możemy zainicjować proces pomiarowy, a po drugie, dokonać odczytu wartości „surowego" ciśnienia atmosferycznego i temperatury otoczenia. Dlaczego użyłem słowa „surowego"? Otóż, każdy element BMP180 przechodzi na etapie produkcji proces kalibracji, który zapewnia osiągnięcie założonej dokładności pomiarów niezależnie od właściwości elementu piezo-rezystancyjnego, który stanowi w nim przetwornik ciśnienia na napięcie. Proces ten kończy się ustaleniem szeregu (dokładnie 11) specjalnych współczynników korekcyjnych (zapisanych w pamięci EEPROM elementu), dzięki którym możliwe staje się obliczenie skompensowanej wartości ciśnienia atmosferycznego i temperatury. Jest to dość typowe rozwiązanie stosowane przez wielu producentów w przypadku elementów tego rodzaju, które przechodzą proces kalibracji na ostatnim etapie produkcji. Biorąc to pod uwagę, pierwszą czynnością, jaką należy wykonać w przypadku obsługi

Listing 10. Funkcja odpowiedzialna za inicjalizację i uruchomienie zegara RTC

```
void megaRTCinit(void)
{
    ASSR |= (1<<AS2);  //Timer2 taktowany za pomocą kwarcu 40kHz podłączonego do wypr. TOSC1/TOSC2
    TCCR2B = (1<<CS21);  //Preskaler = 8
    TCCR2A = (1<<WGM21); //Tryb CTC
    OCR2A  = 49;  //Przerwanie co 10ms (40kHz/8/50)
    while(ASSR & ((1<<TCN2UB)|(1<<OCR2AUB)|(1<<OCR2BUB)|(1<<TCR2AUB)|(1<<TCR2BUB)));  //Oczekiwanie na aktualizację
rejestrów
    TIFR2 &= ~((1<<OCF2B)|(1<<OCF2A)|(1<<TOV2));  //Skasowanie ewentualnych flag przerwań
    TIMSK2 |= (1<<OCIE2A);  //Uruchomienie przerwania od porównania rejestru OCR2A
    //Uruchomienie Timera0 w trybie CTC generującego przebieg 3kHz na wyjściu OC0B dla sterowania głośniczkiem piezzo
    TCCR0A = (1<<COM0B0)|(1<<WGM01);  //Toggle OC0B on Compare Match, CTC Mode
    OCR0A  = 165;  //3000Hz
}
```

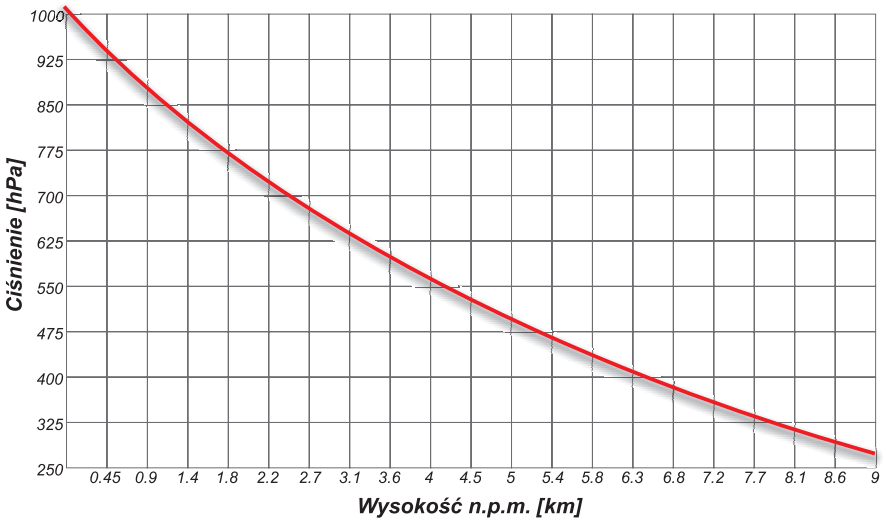
Listing 11. Funkcja odpowiedzialna za ustalenie liczby dni miesiąca

```
inline uint8_t getMonthDaysLimit(uint8_t Year, uint8_t Month)  //Year: 0...99, Month: 0..11
{
    //Ustalamy maksymalną liczbę dni dla miesiąca będącego argumentem funkcji
    registeruint8_tdayLimit = pgm_read_byte(&monthsDays[Month]);
    //Dla Lutego wyznaczamy liczbę dni 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 = 29; elsedayLimit = 28;  //
Rok przestępny lub nie
    }
    returndayLimit;
}
```

barometru BMP180 jest odczyt jedenastu, 16-bitowych rejestrów, które przechowują wartości współczynników korekcyjnych. Następnie wysyłamy do układu BMP180 rozkaz inicjujący pomiar ciśnienia lub temperatury by po pewnym czasie (zależnym od wartości, którą chcemy odczytać, jak i preferowanej dokładności pomiaru) odczytać „surową" (czyli nieskompensowaną) wartość interesującego nas parametru. Dalej, na podstawie dość skomplikowanych wzorów dostarczonych przez producenta układu, obliczamy wartość skompensowanego ciśnienia atmosferycznego lub temperatury otoczenia. Firma Bosch Sensortec dostarcza gotowy driver do obsługi swojego czujnika, co znacznie upraszcza proces implementacji własnego oprogramowania.

Pora na szczegóły implementacyjne. Zacznę od pliku nagłówkowego związanego z obsługą barometru, którego zawartość pokazano na **listingu 13**. Uważny Czytelnik zapewne zwróci uwagę na dość ciekawą konstrukcję struktury danych przechowującej współczynniki korekcyjne. Jest to połączenie unii z tak zwaną strukturą anonimową, dzięki czemu, co zobaczymy za chwilę, w dość prosty sposób odczytamy z pamięci układu BMP180 wszystkie współczynniki korekcyjne (korzystając z pola index wspomnianej unii). Zatem przejdźmy do wspomnianej funkcji, którą pokazano na **listingu 14**. Kolejne dwie funkcje, które przedstawiono na **listingu 15**, pozwalają na odczyt „surowych" wartości ciśnienia atmosferycznego i temperatury otoczenia (funkcje te inicjują stosowny pomiar, a następnie po niezbędnym czasie konwersji, odczytują żądane wielkości). I na sam koniec, na **listingu 16**, funkcja, dzięki której obliczymy rzeczywiste, skompensowane wartości interesujących nas wielkości fizycznych.

W tym miejscu warto wspomnieć jeszcze o jednej funkcjonalności naszego



Rysunek 2. Wykres zależności bezwzględnego ciśnienia powietrza od wysokości n.p.m.

urządzenia, którą jest możliwość prognozowania pogody. Aby jednak zrozumieć zasadę działania zastosowanego mechanizmu nie sposób choćby po krótce nie omówić podstawowych zagadnień związanych z pojęciem ciśnienia atmosferycznego. Z definicji, ciśnienie atmosferyczne jest stosunkiem wartości siły, z jaką słup powietrza atmosferycznego naciska na powierzchnię Ziemi, do powierzchni, na jaką ten słup naciska. Wynika z tego, że dla przykładu ciśnienie atmosferyczne w górach jest niższe a na nizinach wyższe, ponieważ słup powietrza ma w tych rejonach różne wysokości. Zależność ta ma w przybliżeniu charakter wykładniczy, co pokazano na **rysunku 2**. Wartość ciśnienia atmosferycznego dla standardowych warunków pogodowych dla wysokości h (n.p.m.) możemy, zatem wyznaczyć z uproszczonej zależności opisanej wzorem:

$$p_h = p_0 \bullet e^{\frac{-h}{7990}}$$

gdzie

p_h – ciśnienie na wysokości h n.p.m. dla standardowych warunków pogodowych

p_0 – ciśnienie ma poziomie morza równe 1013,25 hPa

h – wysokość n.p.m.

Właśnie na podstawie średniej wartości ciśnienia atmosferycznego na Ziemi na poziomie morza wprowadzono (dla pogodowych warunków standardowych), nieużywaną już, jednostkę atmosfery równą 1013,25 hPa. Jak jednak wiadomo, ciśnienie

atmosferyczne ulega ciągłym zmianom, które to zależne są od zmiany warunków pogodowych w związku z czym na podstawie zmian ciśnienia na założonej wysokości n.p.m. możemy z pewnym prawdopodobieństwem określić tendencję zmian pogody. Dla przykładu, w umiarkowanych szerokościach geograficznych powolne obniżanie się ciśnienia oznacza zbliżanie się niżu, czyli pogorszenia pogody (wystąpienie chmur, opadów, w lecie – ochłodzenia, w zimie – ocieplenia) zaś wzrost ciśnienia łączy się z poprawą pogody (ustąpieniem mgły/zachmurzenia, osłabieniem wiatru itp.). Ciśnienie wyższe o kilka hPa od ciśnienia p_h dla danego rejonu zwiastuje utrwalenie się ładnej pogody, jednak z drugiej strony, szybki wzrost ciśnienia może także zwiastować burzę. Co ciekawe, na podstawie gradientu zmian ciśnienia można dość dokładnie określić spodziewaną siłę wiatru. Pomimo tych, wydawałoby się prostych zależności, przewidywanie zmian pogody jest procesem bardzo skomplikowanym (zwłaszcza przygotowywanie prognoz długookresowych) i obarczonym pewnym błędem, o czym niejednokrotnie możemy się przekonać słuchając prognoz pogody nadawanych przez stacje radiowy czy telewizyjne. Jednak dla prostych aplikacji możemy posiłkować się algorytmami, z jakich korzysta większość elektronicznych stacji pogodowych, a które oparte są li tylko na śledzeniu zmian ciśnienia atmosferycznego. Z uwagi na charakter zachodzących zmian prognozowanie pogody jest znacznie utrudnione dla obszarów położonych wysoko n.p.m.

Listing 12. Funkcja odpowiedzialna za realizację programowego zegara czasu rzeczywistego (RTC)

```
ISR(TIMER2_COMPA_vect)
{
    register uint8_t dayLimit, Flag;
    static uint8_t Hundredths;
    //Obsługaprogramowegozegara RTC
    if(++Hundredths == 100)
    {
        Hundredths = 0;
        Flag = Clock.Flag | CLOCK_SECOND;  //Optymalizacja volatile
        if(++Clock.Second == 60)
        {
            Clock.Second = 0; Flag |= CLOCK_MINUTE;
            if(++Clock.Minute == 60)
            {
                Clock.Minute = 0; Flag |= CLOCK_HOUR;
                if(++Clock.Hour == 24)
                {
                    Clock.Hour = 0; Flag |= CLOCK_DAY|CLOCK_WEEKDAY;
                    if(++Clock.WeekDay == 7) Clock.WeekDay = 0;  //Dzieńtygodnia
                    //Teraz ustalimy maksymalną liczbę dni dla bieżącego miesiąca
                    dayLimit = getMonthDaysLimit(Clock.Year, Clock.Month);
                    if(++Clock.Day> dayLimit-1)  //To był ostatni dzień bieżącego miesiąca
                    {
                        Clock.Day = 0; Flag |= CLOCK_MONTH;
                        if(++Clock.Month == 12)
                        {
                            Clock.Month = 0; Flag |= CLOCK_YEAR;
                            if(++Clock.Year == 100) Clock.Year = 0;
                        }
                    }
                }
            }
        }
        Clock.Flag = Flag;  //Optymalizacja volatile
    }
    //Obsługaprogramowegotimera
    if(Timer.Activity)
    {
        Timer.Hundredth++;
        Flag = Timer.Flag | TIMER_HUNDREDTH;  //Optymalizacja volatile
        if(Timer.Hundredth%10 == 0)
        {
            Flag |= TIMER_HUNDREDTH10;
            if(Timer.Hundredth == 100)
            {
                Timer.Hundredth = 0; Timer.Second++; Flag |= TIMER_SECOND;
                if(Timer.Second%10 == 0)
                {
                    Flag |= TIMER_SECOND10;
                    if(Timer.Second == 60)
                    {
                        Timer.Second = 0; Timer.Minute++; Flag |= TIMER_MINUTE;
                        if(Timer.Minute%10 == 0)
                        {
                            Flag |= TIMER_MINUTE10;
                            if(Timer.Minute == 100) {Timer.Minute = Timer.Second = Timer.Hundredth = 0; Flag = TIMER_
ALL_VALUES;}
                        }
                    }
                }
            }
        }
        Timer.Flag = Flag;  //Optymalizacja volatile
    }
    //Obsługa programowego budzika
    if(Alarm.Activity)
    {
        //Sprawdzenie, czy wśród aktywnych dni alarmowania znajduje się dzień bieżący
        for(uint8_t Day=0; Day<8; ++Day) if(Alarm.WeekDays& (1<<Day))
        {
            //Sprawdzenie dnia, godziny i minuty alarmowania
            if(Clock.WeekDay == Day &&Alarm.Hour == Clock.Hour&&Alarm.Minute == Clock.Minute)
            {
                //Generowanie dźwięku głośniczka piezzo, w przypadku wystąpienia alarmu
                if(Hundredths%30 == 0) START_SOUND;
                if(Hundredths%50 == 0) STOP_SOUND;
                break;
            }
            else STOP_SOUND;
        }
    } else STOP_SOUND;
}
```

Listing 13. Treść pliku nagłówkowego związanego z obsługą barometru scalonego BMP180

```
//Structure that keeps all the BMP180 factory coefficients
typedef union
{
    struct
    {
        int16_t AC1, AC2, AC3;
        uint16_t AC4, AC5, AC6;
        int16_t B1, B2, MB, MC, MD;
    };
    uint16_t index[11];
} coeffsType;

//Structure that keeps output data
typedef struct
{
    uint16_tTemperature;  //in 0.1'C
    uint16_tPressure;  //in hPa
} paramsType;

//Global structure that keeps output data
extern paramsTypeParams;
//Function prototypes
void BMP180readCoefficients(void);
void BMP180readParams(uint8_tOverSmpl);

#define BMP180_WRITE_ADDR 0xEE
#define BMP180_READ_ADDR 0xEF
//Registers and they addresses
#define BMP180_COEFF_AC1_REG 0xAA
#define BMP180_COEFF_AC2_REG 0xAC
#define BMP180_COEFF_AC3_REG 0xAE
#define BMP180_COEFF_AC4_REG 0xB0
#define BMP180_COEFF_AC5_REG 0xB2
#define BMP180_COEFF_AC6_REG 0xB4
#define BMP180_COEFF_B1_REG 0xB6
#define BMP180_COEFF_B2_REG 0xB8
#define BMP180_COEFF_MB_REG 0xBA
#define BMP180_COEFF_MC_REG 0xBC
#define BMP180_COEFF_MD_REG 0xBE
#define BMP180_CONTROL_REG 0xF4
    #define BMP180_OVERSAMPLING_X1 0x00
    #define BMP180_OVERSAMPLING_X2 0x01
    #define BMP180_OVERSAMPLING_X4 0x02
    #define BMP180_OVERSAMPLING_X8 0x03
#define BMP180_TEMP_START_REG 0xF6  //MSB
#define BMP180_PRESSURE_START_REG 0xF6  //MSB
#define BMP180_ID_REG 0xD0  //Should be 0x55 if BMP180 is present
//Commands
#define BMP180_READ_TEMP_CMD 0x2E
#define BMP180_READ_PRESSURE_CMD 0x34
```

Jak łatwo się domyślić, urządzenie nasze udostępnia możliwość określenia wysokości nad poziomem morza, na jakiej to się znajdujemy by zapewnić realizację funkcjonalności prognozowania pogody. W tym celu przewidziano odpowiednią funkcjonalność systemu Menu.

Znając już wszystkie, programowe szczegóły implementacyjne, powróćmy jeszcze na chwilę do schematu naszego urządzenia, gdyż kilka słów uwagi należy się blokowi zasilającemu, który to zbudowano z wykorzystaniem specjalizowanego układu ładowania akumulatorów litowo-jonowych i litowo-polimerowych MCP73832 produkcji firmy Microchip. Układ ten integruje w sobie kompletny system ładowani. Idealnie nadaje się on do zastosowania w prostych aplikacjach ładowarek ogniw litowo-jonowych i litowo-polimerowych, gdyż w pełni automatycznie nadzoruje proces ładowania takiego akumulatora wybierając odpowiedni tryb ładowania oraz mechanizm kontroli, zaś jedynym „zmartwieniem" użytkownika jest wybór prądu ładowania szybkiego, którego dokonujemy dobierając wartość rezystora podłączonego pomiędzy wyprowadzenie PROG a masę. Prąd ten dobieramy według poniższej zależności:

$$I_{REG} = 1000\ V/R_{PROG}$$

gdzie wielkości: I_{REG} wyrażono w mA, zaś R_{PROG} w kΩ.

Listing 14. Funkcja odpowiedzialna za odczyt wszystkich współczynników korekcyjnych układu BMP180

```
void BMP180readCoefficients(void)
{
    TWIstart();
    TWIwriteByte(BMP180_WRITE_ADDR);  //BMP180 i2c address for write operation
    TWIwriteByte(BMP180_COEFF_AC1_REG);  //Coefficient AC1 start register (MSB)
    TWIstart();  //i2c restart
    TWIwriteByte(BMP180_READ_ADDR);  //BMP180 i2c address for read operation
    //Now, we read all the BMP180 factory coefficients (AC1...MD) into special structure
    for (uint8_tidx=0; idx<11; ++idx) Cfs.index[idx] = TWIreadInteger(idx == 10? NACK:ACK);
    TWIstop();
}
```

Na koniec, warto podkreślić istotną różnicę pomiędzy wartością bezwzględnego ciśnienia atmosferycznego dla danej miejscowości (ciśnienia tam panującego) a wartością ciśnienia podawanego w prognozach pogody. To ostatnie jest ciśnieniem, jakie wystąpiłoby danego dnia w danej miejscowości, gdyby znajdowała się ona na poziomie morza, czyli de facto jest to bezwzględne ciśnienie atmosferyczne dla tego rejonu przeliczone dla poziomu morza. Tego typu konwersja ułatwia zorientowanie się, co do warunków pogodowych dla różnych wysokości n.p.m.

Pora na przedstawienie prostego algorytmu prognozowania pogody. Generalnie, w nieskomplikowanych konstrukcjach stacji pogodowych, które do prognozowania pogody wykorzystują wyłącznie wartość mierzonego ciśnienia atmosferycznego, korzysta się z dwóch rodzajów algorytmów. Pierwszy z nich śledzi zmiany ciśnienia atmosferycznego i po upływie pewnego czasu (zwykle 6 do 12 godzin) na podstawie gradientu zmian jest w stanie określić oczekiwany stan pogody. Drugi z algorytmów, z którego korzysta nasze urządzenie, oblicza wzorcowe ciśnienie dla danej wysokości n.p.m. (na której to się znajdujemy) dla dobrych warunków pogodowych (wspomniany wcześniej wzór na p_h) i na podstawie różnicy ciśnienia mierzonego i wzorcowego określa prognozę pogody niwelując potrzebę oczekiwania na zmiany gradientu ciśnienia. Oczywiście, prognozy takie mogą być obarczone dużym błędem, lecz wykonanie bardziej wiarygodnej prognozy pogody wymagałoby dysponowania większa liczbą wielkości wejściowych. Dla naszego urządzenia przyjęto następujące wartości kryterialne:

Różnica pomiędzy Ph a Pcomp	Symbol pogody
Diff ≥9	Słoneczko
–9 <Diff< 9	Zachmurzenie
Diff ≤ –9	Deszcz/Śnieg

W tym urządzeniu rezystor R_{PROG} (R1) ma wartość 10 kΩ, co ustawia prąd ładowania szybkiego na 100 mA. Nie bez powodu ustawiono tego typu prąd ładowania, wszak pamiętać należy, iż dla wygody do zasilania naszego urządzenia przewidziano podłączenie go do portu USB komputera (lub

Listing 15. Funkcje odpowiedzialne za odczyt „surowych" wartości ciśnienia atmosferycznego i temperatury otoczenia

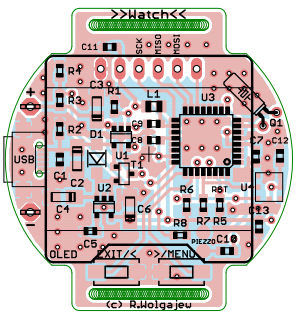
```
uint16_t BMP180readRawTemperature(void)
{
        register uint16_trawT;
        TWIstart();
        TWIwriteByte(BMP180_WRITE_ADDR);  //BMP180 i2c address for write operation
        TWIwriteByte(BMP180_CONTROL_REG);  //BMP180 control register
        TWIwriteByte(BMP180_READ_TEMP_CMD);  //Read temperature command that starts T conversion
        TWIstop();
        _delay_ms(5);  //We need to wait while conversion ends
        TWIstart();
        TWIwriteByte(BMP180_WRITE_ADDR);  //BMP180 i2c address for write operation
        TWIwriteByte(BMP180_TEMP_START_REG);  //Temperature register (MSB)
        TWIstart();  //i2c restart
        TWIwriteByte(BMP180_READ_ADDR);  //BMP180 i2c address for read operation
        rawT = TWIreadInteger(NACK);  //We read raw temperature value
        TWIstop();
        returnrawT;
}

uint32_t BMP180readRawPressure(uint8_t OverSmpl)
{
    register uint32_trawP;
    TWIstart();
    TWIwriteByte(BMP180_WRITE_ADDR);  //BMP180 i2c address for write operation
    TWIwriteByte(BMP180_CONTROL_REG);  //BMP180 control register
    TWIwriteByte(BMP180_READ_PRESSURE_CMD + (OverSmpl<<6));  //Read pressure command that starts conversion
    TWIstop();
    //We need to wait while conversion ends. The time depends on OverSmpl setting
    if(OverSmpl == BMP180_OVERSAMPLING_X1) _delay_ms(5);
    else if(OverSmpl == BMP180_OVERSAMPLING_X2) _delay_ms(8);
    else if(OverSmpl == BMP180_OVERSAMPLING_X4) _delay_ms(14);
    else_delay_ms(26);
    TWIstart();
    TWIwriteByte(BMP180_WRITE_ADDR);  //BMP180 i2c address for write operation
    TWIwriteByte(BMP180_PRESSURE_START_REG);  //Pressure register (MSB)
    TWIstart();  //i2c restart
    TWIwriteByte(BMP180_READ_ADDR);  //BMP180 i2c address for read operation
    rawP = (uint32_t) TWIreadInteger(ACK)<<8;  //MSB i LSB
    rawP += TWIreadByte(NACK);  //XLSB
    TWIstop();
    rawP>>= (8-OverSmpl);  //Bosch datasheet, page 15
    returnrawP;
}
```

Listing 16. Funkcja odpowiedzialna za obliczenie rzeczywistych wartości ciśnienia atmosferycznego i temperatury otoczenia

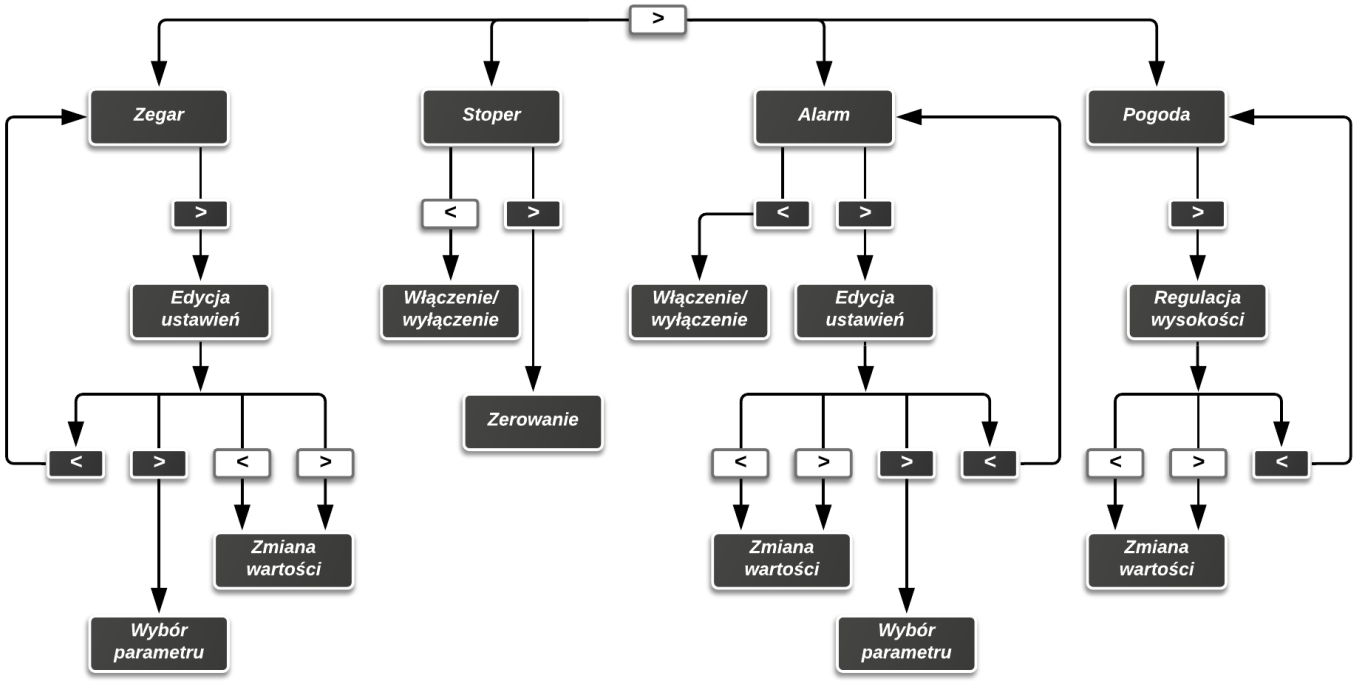
```
void BMP180readParams(uint8_t OverSmpl)
{
    int32_t UT, UP, X1, X2, X3, B3, B5, B6, Pressure;  //Following the BMP180 datasheet
    uint32_t B4, B7;  //Following the BMP180 datasheet
    UT = BMP180readRawTemperature();  //We read raw temperature
    UP = BMP180readRawPressure(OverSmpl);  //We read raw pressure
    //Now we do some temperature calculations
    X1 = ((UT - (int32_t)Cfs.AC6) * (int32_t)Cfs.AC5) >>15;
    X2 = ((int32_t)Cfs.MC<<11) / (X1+(int32_t)Cfs.MD);
    B5 = X1 + X2;
    Params.Temperature = (B5+8) >>4;  //Output in 0.1'C
    //Next we do some pressure calculations
    B6 = B5 - 4000;
    X1 = ((int32_t)Cfs.B2 * ((B6 * B6)>>12)) >>11;
    X2 = ((int32_t)Cfs.AC2 * B6) >>11;
    X3 = X1 + X2;
    B3 = ((((int32_t)Cfs.AC1*4 + X3) <<OverSmpl) + 2) >>2;
    X1 = ((int32_t)Cfs.AC3 * B6) >>13;
    X2 = ((int32_t)Cfs.B1 * ((B6 * B6) >>12)) >>16;
    X3 = ((X1 + X2) + 2) >>2;
    B4 = ((uint32_t)Cfs.AC4 * (uint32_t)(X3 + 32768)) >>15;
    B7 = ((uint32_t)UP - B3) * (50000UL >>OverSmpl);
    Pressure = B7<0x80000000? (B7<<1)/B4 : (B7/B4)<<1;
    X1 = (Pressure >>8) * (Pressure >>8);
    X1 = (X1 * 3038) >>16;
    X2 = (-7357 * Pressure) >>16;
    Params.Pressure = (Pressure + ((X1 + X2 + 3791)>>4)) / 100;  //Output in hPa
}
```

popularnego zasilacza sieciowego z portem USB przeznaczonego do ładowania telefonów komórkowych) zaś ten pozwala na maksymalny pobór prądu rzędu 500 mA w trybie high-power, zaś typowo ok. 100 mA. Kilka dodatkowych słów uwagi wymaga także opcjonalny układ współdzielenia obciążenia zbudowany przy użyciu tranzystora T1 typu MOSFET z kanałem P, diody Schottky D1 oraz rezystora R2.

Dlaczego niezbędna była implementacja tego rodzaju rozwiązania? Otóż, układ MCP73832 nie posiada w swojej strukturze odpowiednich bloków funkcjonalnych odpowiedzialnych za współdzielenie obciążenia, to znaczy odpowiedzialnych za uwzględnienie w procesie ładowania faktu, iż w czasie, gdy układ nadzoruje proces ładowania akumulatora tenże akumulator zasila urządzenie, które pobiera z niego prąd. Taka sytuacja, po pierwsze powoduje w najlepszym wypadku wydłużenie samego procesu ładowania, zaś w skrajnych wypadkach może go zaburzyć czy też spowodować, iż proces ładowania nigdy nie dobiegnie końca (gdyż odbiornik pobierając nieustannie prąd z ogniwa, a więc de facto z układu nadzorującego nie pozwoli tym samym na skuteczną detekcję końca procesu ładowania). Aby temu zapobiec zastosowano prosty „przełącznik" w postaci tranzystora MOSFET, którego bramkę podłączono bezpośrednio do napięcia USB zasilającego ładowarkę. W przypadku obecności napięcia USB (czyli de facto stanu wysokiego



Rysunek 3. Schemat montażowy urządzenia Watch



Rysunek 4. Diagram obrazujący sposób obsługi urządzenia Watch

na bramce tranzystora) tranzystor T1 przechodzi w stan wyłączenia (brak przewodzenia pomiędzy drenem a źródłem) odłączając tym samym ładowany akumulator od obciążenia. W tym samym czasie obciążenie, jakim jest w naszym wypadku urządzenie „Watch", zostaje zasilone bezpośrednio z napięcia portu USB, a dokładnie rzecz ujmując, poprzez diodę D1. W przypadku odłączenia naszego urządzenia od napięcia zasilającego USB, bramka tranzystora T1 zostaje ściągnięta do masy (poprzez rezystor R2) powodując przewodzenie tegoż tranzystora a więc tym samym zasilenie naszego urządzenia z akumulatora ACCU. W tym przypadku dioda D1 pełni nieco inną funkcję, a mianowicie zabezpiecza przed przepływem prądu wstecznego tj. z akumulatora w kierunku źródła napięcia zasilającego (USB). W ten prosty sposób zbudowano prosty i w pełni funkcjonalny układ współdzielenia obciążenia, który czasami występuje w innych typach scalonych kontrolerów ładowania produkcji firmy Microchip. Dalej, wyjście z układu ładowania wprowadzono na wejście stabilizatora LDO typu APE8865Y5-27-HF-3, który zapewnia stały poziom napięcia zasilającego system mikroprocesorowy (2,7 V) niezależnie od stanu układu ładowania.

Już zupełnie dla porządku dodam, że aplikacja układu „Watch" korzysta również z wbudowanego w strukturę mikrokontrolera przetwornika ADC (wejście ADC1), przy którego udziale (za pomocą dzielnika rezystorowego R3/R4) jest mierzony poziom napięcia akumulatora zasilającego, co pozwala na „zgrubne" określenie stanu naładowania tegoż elementu. Jest to oczywiście rozwiązanie dość proste, gdyż dokładne określenie stanu naładowania akumulatora wymagałoby zastosowania specjalizowanego kontrolera wielkości gromadzonego i traconego ładunku, jednak w tak prostych systemach wydaje się w zupełności wystarczające.

Rysunek 5. Widok dostępnych funkcji (ekranów Menu) urządzenia Watch

Montaż

Schemat montażowy pokazano na **rysunku 3**. Jak widać, zaprojektowano „zgrabną", niewielką płytkę drukowaną, która kształtem przypomina zegarek naręczny umożliwiając też zamocowanie typowego paska utrzymującego ją na nadgarstku (przewidziano odpowiednie, podłużne otwory montażowe).

Montaż urządzenia rozpoczynamy od przylutowania scalonego barometru BMP180. Proces ten najłatwiej wykonać przy użyciu stacji lutowniczej na gorące powietrze (tzw. Hot Air) i odpowiednich stopów lutowniczych. Następnie lutujemy pozostałe elementy półprzewodnikowe, potem

rezystory i kondensatory jak i inne elementy bierne, a na końcu przyciski, gniazdo USB i rezonator kwarcowy. Z uwagi na zagęszczenie wyprowadzeń układów scalonych przed pierwszym podłączeniem układu do zasilania należy jeszcze raz sprawdzić jakość wykonanych połączeń, aby nie dopuścić do ewentualnych zwarć. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej. Na samym końcu, do tak przygotowanej płytki, montujemy wyświetlacz OLED, zwyczajnie lutując jego wyprowadzenia w przeznaczone do tego celu pola lutownicze (należy sprawdzić polaryzację zasilania), gdyż połączenia te zapewniają mu jednocześnie wystarczający montaż mechaniczny. Akumulator zasilający podłączamy do wyprowadzeń ACCU (zachowując, co oczywiste odpowiednią polaryzację) i podklejamy do płytki urządzenia od strony BOTTOM. Podobnie postępujemy z sygnalizatorem piezoelektrycznym, który za pomocą krótkich przewodów podłączamy do przeznaczonych do tego celu wyprowadzeń (oznaczonych „PIEZZO"), zaś sam element umieszczamy pomiędzy płytką urządzenia a płytką zastosowanego modułu OLED. Poprawnie zmontowany układ powinien działać od razu po podłączeniu zasilania.

Obsługa

Jako, że urządzenie „Watch" jest z założenia medium mobilnym, które to może być obsługiwane w nieoptymalnych warunkach rzeczywistych, ergonomia i prostota obsługi układu jak i czytelność interfejsu użytkownika była podstawowym kryterium przy konstruowaniu stosownych procedur sterujących. Zgodnie z tymi podstawowymi założeniami, na płytce sterownika przewidziano wyłącznie 2 przyciski sterujące (*Menu* i *Exit*), przy czym każdy z nich obsługuje zdarzenie krótkiego (ok.100 ms) jak i długiego (powyżej 500 ms) naciśnięcia. Diagram obrazujący sposób obsługi urządzenia pokazano na **rysunku 4** (symbole przycisków wypełnione kolorem czarnym symbolizują długie naciśnięcie wybranego elementu), zaś widok dostępnych funkcji Menu urządzenia na **rysunku 5**.

Już zupełnie dla porządku dodam, iż urządzenie „Watch" wyposażono w prosty mechanizm redukcji poboru mocy, którego działanie polega na automatycznym wyłączeniu wyświetlacza OLED po czasie około 20 sekund bezczynności (braku działań po stronie użytkownika). Aktywacja wyświetlacza następuję z chwilą przyciśnięcia dowolnego z elementów sterujących. Ten prosty mechanizm jest na tyle skuteczny, iż pozwala na ponad tygodniową pracę systemu bez potrzeby ponownego ładowania wbudowanego akumulatora zasilającego.

Robert Wołgajew, EP