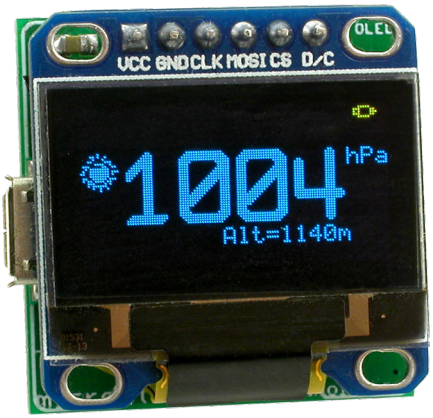
microStation – mała stacja pogodowa

Pogoda potrafi być bardzo kapryśna. Nie mając gwarancji dobrej pogody trudno planować dłuższe eskapady, ale jest pewien gadżet, który powinien poprawić nieco nasze szanse na dobrą zabawę na świeżym powietrzu. Mowa o podręcznej stacji pogodowej, która pokaże bieżący stan pogody oraz poinformuje o prognozie na podstawie bieżących parametrów pogodowych.



Schemat ideowy urządzenia nazwanego, z uwagi na miniaturowe wymiary, microStation, pokazano na **rysunku 1**. Jest to bardzo prosty system mikroprocesorowy, którego „sercem" jest niewielki mikrokontroler firmy Microchip (dawniej Atmel) typu ATtiny84 realizujący całą, założoną funkcjonalność urządzenia. Mikrokontroler ten steruje pracą niewielkiego graficznego wyświetlacza OLED o organizacji 128×64 pikseli, używając do tego celu programowej implementacji interfejsu SPI, w który wyposażony jest zastosowany wyświetlacz. Obsługuje także scalony czujnik temperatury, wilgotności i ciśnienia atmosferycznego pod postacią układu firmy Bosch Sensortec o oznaczeniu BME280, używając do tego celu programowej implementacji interfejsu I²C, w który wyposażony jest wspomniany czujnik.

Czujnik BME280

Układ BME280 stanowi główny element stacji pogodowej, dostarczając wszystkie interesujące nas dane pogodowe, w związku z czym warto przyjrzeć się bliżej temu ciekawemu peryferium.

Układ BME280 charakteryzuje się następującymi, wybranymi cechami:

- pomiar ciśnienia atmosferycznego (300…1100 hPa), temperatury otoczenia (-40…85°C) i wilgotności (0…100%),
- szeroki zakres napięć zasilania 1,7...3,6 V,
- niski pobór prądu rzędu 3,6 µA/pomiar,
- wysoka dokładność pomiaru oraz niski poziom szumów,
- szybka magistrala I²C z dopuszczalną prędkością sygnału zegarowego dochodzącą do 3,4 MHz,
- szybka magistrala SPI z dopuszczalną prędkością sygnału zegarowego dochodzącą do 10 MHz,
- małe wymiary obudowy typu LGA 2,5×2,5 mm.

Powyższe parametry idealnie wpisują się w założenia naszego projektu, a jedynym problemem jaki możemy napotkać stosując wspomniany element jest jego dość kłopotliwa, jeśli chodzi o montaż, obudowa LGA, wymagająca sporej wprawy w lutowaniu oraz odpowiedniego sprzętu lutowniczego. Obsługa peryferiów tego typu polega na zapisie/odczycie wielu, specjalnych rejestrów konfiguracyjnych lub też rejestrów danych, przy udziale których, po pierwsze możemy zainicjować proces pomiarowy, a po drugie, dokonać odczytu interesujących nas wartości ciśnienia atmosferycznego, wilgotności i temperatury otoczenia. Będą to wartości „surowe", wymagające odpowiedniego przeliczenia. Każdy element BME280 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 osiemnastu) specjalnych współczynników korekcyjnych (zapisanych w nieulotnej pamięci EEPROM elementu), dzięki którym możliwe staje się obliczenie skompensowanej wartości ciśnienia atmosferycznego, wilgotności i temperatury. Jest to dość typowe rozwiązanie stosowane przez wielu producentów w przypadku elementów tego rodzaju, które przechodzą proces kalibracji na ostatnim etapie produkcji. Zatem, pierwszą czynnością, jaką należy wykonać w przypadku obsługi układu BME280 jest odczyt osiemnastu, 16- i 8-bitowych rejestrów, które przechowują wartości współczynników korekcyjnych. Następnie należy wysyłać do układu BME280 rozkaz inicjujący pomiary, by po pewnym czasie odczytać nieskompensowane wartości interesujących nas parametrów. Dalej, na podstawie dość skomplikowanych wzorów dostarczonych przez producenta układu, można obliczyć skompensowane wartości ciśnienia atmosferycznego, wilgotności i temperatury otoczenia. Jest to dość niezrozumiałe podejście, ale stosowane przez większość producentów scalonych barometrów. Przecież korzystając z faktu, że współczynniki korekcyjne znajdują się w nieulotnej pamięci układu, można całą procedurę obliczeń pozostawić po stronie elementu, a użytkownikowi udostępnić gotowe wyniki, co znacznie ułatwiłoby obsługę takiego rodzaju peryferium. Na szczęście firma Bosch Sensortec dostarcza gotowy driver do obsługi swojego czujnika, co znacznie upraszcza proces implementacji własnego oprogramowania.

Program

Szczegóły implementacyjne mojej wersji oprogramowania zacznę od pliku

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

W ofercie AVT* AVT-5722

Podstawowe parametry:
- pomiar ciśnienia atmosferycznego (300…1100 hPa),
- pomiar temperatury otoczenia (-40…85°C),
- pomiar wilgotności (0…100%),
- prezentacja wyników na wyświetlaczu OLED,
- zasilanie 5 V (USB),
- pobór prądu (z akumulatora) ok. 7,5 mA, lub 0,7 mA w stanie Standby,
- niewielkie wymiary.

Projekty pokrewne na www.media.avt.pl:

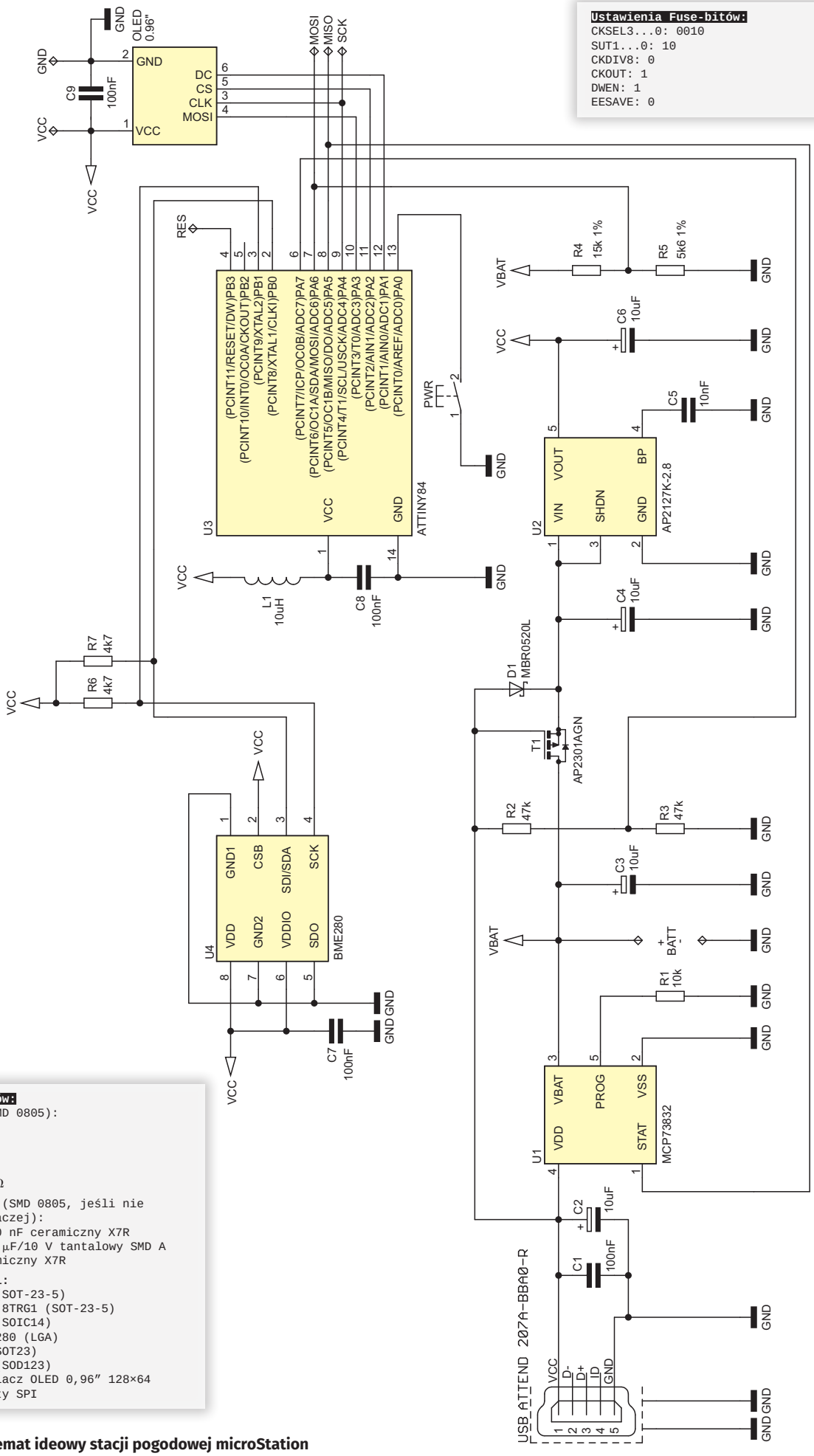
AVT-5668	Moduł czujnika temperatury, wilgotności, ciśnienia z interfejsem I²C (EP 3/2019)
AVT-5654	Amatorska stacja pogodowa (EP 12/2018)
AVT-5639	Bezprzewodowy czujnik warunków atmosferycznych (EP 10/2018)
AVT-5605	wiStation – domowa stacja pogodowa z prognozą pogody (EP 9/2017)

* **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.

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



Wykaz elementów:

Rezystory: (SMD 0805):
R1: 10 kΩ
R2, R3: 47 kΩ
R4: 15 kΩ 1%
R5: 5,6 kΩ 1%
R6, R7: 4,7 kΩ

Kondensatory: (SMD 0805, jeśli nie zaznaczono inaczej):
C1, C7…C9: 100 nF ceramiczny X7R
C2…C4, C6: 10 µF/10 V tantalowy SMD A
C5: 10 nFceramiczny X7R

Półprzewodniki:
U1: MCP73832 (SOT-23-5)
U2: AP2127K-2.8TRG1 (SOT-23-5)
U3: ATtiny84 (SOIC14)
U4: Bosch BME280 (LGA)
T1: LGE2301 (SOT23)
D1: MBR0520L (SOD123)
OLED: wyświetlacz OLED 0,96” 128×64 niebiesko/żółty SPI

Rysunek 1. Schemat ideowy stacji pogodowej microStation

nagłówkowego związanego z obsługą elementu BME280, którego zawartość pokazuje **listing 1**. Został napisany w taki sposób, by bez zagłębiania się w dokumentację czujnika można było zorientować się w funkcjonalności poszczególnych rejestrów konfiguracyjnych/danych.

Dalej, potrzebujemy kilka prostych funkcji narzędziowych, dzięki którym (i przy udziale

Listing 1. Plik nagłówkowy do obsługi układu BME280

```
//Adresy sprzętowe i2c układu BME280
#define BME280_WRITE_ADDR 0xEC
#define BME280_READ_ADDR 0xED

//Rejestry współczynników korekcyjnych
#define BME280_DIG_T1_REG 0x88
#define BME280_DIG_T2_REG 0x8A
#define BME280_DIG_T3_REG 0x8C

#define BME280_DIG_P1_REG 0x8E
#define BME280_DIG_P2_REG 0x90
#define BME280_DIG_P3_REG 0x92
#define BME280_DIG_P4_REG 0x94
#define BME280_DIG_P5_REG 0x96
#define BME280_DIG_P6_REG 0x98
#define BME280_DIG_P7_REG 0x9A
#define BME280_DIG_P8_REG 0x9C
#define BME280_DIG_P9_REG 0x9E

#define BME280_DIG_H1_REG 0xA1
#define BME280_DIG_H2_REG 0xE1
#define BME280_DIG_H3_REG 0xE3
#define BME280_DIG_H4_REG 0xE4
#define BME280_DIG_H5_REG 0xE5
#define BME280_DIG_H6_REG 0xE7

//Rejestry identyfikacyjne układu
#define BME280_CHIPID_REG 0xD0
#define BME280_VERSION_REG 0xD1

//Rejestr resetu software-owego
#define BME280_SOFTRESET_REG 0xE0
          #define DO_RESET 0xB6

//Rejestr ustawień oversamplingu dla
//pomiaru wilgotności
#define BME280_CONTROLHUMID_REG 0xF2
          #define H_OVERSMP_SKIPPED (0<<0)
          #define H_OVERSMP_X1 (1<<0)
          #define H_OVERSMP_X2 (2<<0)
          #define H_OVERSMP_X4 (3<<0)
          #define H_OVERSMP_X8 (4<<0)
          #define H_OVERSMP_X16 (5<<0)

//Rejestr statusowy
#define BME280_STATUS_REG 0XF3
          #define IS_MEASURING (1<<3)
          #define IM_UPDATE (1<<0)

//Rejestr ustawień oversamplingu dla
//pomiaru temperatury i ciśnienia oraz
//ustawień trybu pracy
#define BME280_CONTROL_MEASURE 0xF4
          #define T_OVERSMP_SKIPPED (0<<5)
          #define T_OVERSMP_X1 (1<<5)
          #define T_OVERSMP_X2 (2<<5)
          #define T_OVERSMP_X4 (3<<5)
          #define T_OVERSMP_X8 (4<<5)
          #define T_OVERSMP_X16 (5<<5)
          #define P_OVERSMP_SKIPPED (0<<2)
          #define P_OVERSMP_X1 (1<<2)
          #define P_OVERSMP_X2 (2<<2)
          #define P_OVERSMP_X4 (3<<2)
          #define P_OVERSMP_X8 (4<<2)
          #define P_OVERSMP_X16 (5<<2)
          #define MODE_SLEEP (0<<0)
          #define MODE_FORCED (1<<0)
          #define MODE_NORMAL (3<<0)

//Rejestr konfiguracyjny ustawień czasu
//Standby (istotny w normalnym trybie pracy)
//oraz ustawień filtra IIR
#define BME280_CONFIG_REG 0xF5
          #define STBY_TIME_0_5 0<<5)
          #define STBY_TIME_62_5 1<<5)
          #define STBY_TIME_125 (2<<5)
          #define STBY_TIME_250 (3<<5)
          #define STBY_TIME_500 (4<<5)
          #define STBY_TIME_1000 (5<<5)
          #define STBY_TIME_10 (6<<5)
          #define STBY_TIME_20 (7<<5)
          #define FILTER_OFF (0<<2)
          #define FILTER_2 (1<<2)
          #define FILTER_4 (2<<2)
          #define FILTER_8 (3<<2)
          #define FILTER_16 (4<<2)

//Rejestry danych: ciśnienia,
//temperatury i wilgotności
#define BME280_PRESSUREDATA_REG 0xF7
#define BME280_TEMPDATA_REG 0xFA
#define BME280_HUMIDDATA_REG 0xFD
```

Listing 2. Funkcje umożliwiające zapis i odczyt 8-bitowego rejestru

```
//Zapis wartości „regValue” do 8-bitowego rejestru
//znajdującego się pod adresem „regAddress”
void BME280writeRegister8(uint8_t regAddress, uint8_t regValue)
{
          i2cStart();
          i2cWriteByte(BME280_WRITE_ADDR);
          i2cWriteByte(regAddress);
          i2cWriteByte(regValue);
          i2cStop();
}
//Odczyt wartości 8-bitowego rejestru
//znajdującego się pod adresem „regAddress”
uint8_t BME280readRegister8(uint8_t regAddress)
{
          uint8_t readByte;
          i2cStart();
          i2cWriteByte(BME280_WRITE_ADDR);
          i2cWriteByte(regAddress);
          i2cRepeatStart();
          i2cWriteByte(BME280_READ_ADDR);
          readByte = i2cReadByte(NACK);
          i2cStop();
          return readByte;
}
```

Listing 3. Funkcja umożliwiająca odczyt wartości 16-bitowego rejestru

```
//Odczyt wartości 16-bitowego rejestru
//znajdującego się pod adresem „regAddress”
//(format Little endian)
uint16_t BME280readRegister16(uint8_t regAddress)
{
          uint16_t readWord;
          i2cStart();
          i2cWriteByte(BME280_WRITE_ADDR);
          i2cWriteByte(regAddress);
          i2cRepeatStart();
          i2cWriteByte(BME280_READ_ADDR);

          //Odczytany, młodszy bajt słowa
          readWord = i2cReadByte(ACK);

          //Odczytany, starszy bajt słowa
          readWord |= i2cReadByte(NACK)<<8;

          i2cStop();
          return readWord;
}
```

Listing 4. Funkcja umożliwiająca odczyt wartości 24-bitowego rejestru danych

```
//Odczyt wartości 24-bitowego rejestru
//znajdującego się pod adresem „regAddress”
//(format Big endian, z przesunięciem 4 bity w prawo)
uint32_t BME280readRegister24(uint8_t regAddress)
{
          uint32_t readLong;
          i2cStart();
          i2cWriteByte(BME280_WRITE_ADDR);
          i2cWriteByte(regAddress);
          i2cRepeatStart();
          i2cWriteByte(BME280_READ_ADDR);

          //Odczytany bajt słowa MSB
          readLong = (uint32_t) i2cReadByte(ACK)<<12;

          //Odczytany bajt słowa LSB
          readLong |= (uint32_t) i2cReadByte(ACK)<<4;

          //Odczytany bajt słowa XLSB
          readLong |= (uint32_t) i2cReadByte(NACK)>>4;

          i2cStop();
          return readLong;
}
```

Listing 5. Typ danych odpowiedzialny za przechowywanie wartości współczynników korekcyjnych

```
//Struktura globalna przechowująca
//współczynniki korekcyjne układu BME280
struct
{
          //Współczynniki temperatury
          uint16_t T1;
          int16_t T2;
          int16_t T3;
          //Współczynniki ciśnienia
          uint16_t P1;
          int16_t P2;
          int16_t P3;
          int16_t P4;
          int16_t P5;
          int16_t P6;
          int16_t P7;
          int16_t P8;
          int16_t P9;
          //Współczynniki wilgotności
          uint8_t H1;
          int16_t H2;
          uint8_t H3;
          int16_t H4;
          int16_t H5;
          int8_t H6;
} Cfs;
```

magistrali I²C) będziemy mogli operować na rejestrach konfiguracyjnych oraz danych naszego peryferium. Jednak tym razem nie możemy ograniczyć się wyłącznie do dwóch funkcji umożliwiających zapis i odczyt, gdyż rejestry układu BME280 występują w trzech konfiguracjach: jako 8-, 16- i 24-bitowe i to zarówno w notacji little endian, jak i big endian. Funkcje umożliwiające zapis i odczyt 8-bitowego rejestru układu BME280 prezentuje **listing 2**. Funkcję umożliwiającą odczyt wartości 16-bitowego rejestru układu BME280 zapisanego w notacji little endian (takimi rejestrami są niektóre z rejestrów przechowujących wartości współczynników korekcyjnych) prezentuje **listing 3**. Na koniec prosta funkcja narzędziowa umożliwiająca odczyt wartości 24-bitowego rejestru układu BME280 zapisanego w notacji big endian (takimi rejestrami są rejestry „surowych" danych). Kod funkcji pokazuje **listing 4**.

Mając funkcje narzędziowe operujące na rejestrach układu BME280 pora na przedstawienie podstawowych funkcji obsługi. Zanim do nich przejdę niezbędne jest wprowadzenie nowego, strukturalnego typu danych, dzięki któremu możliwe stanie się odczytanie i przechowywanie wszystkich współczynników korekcyjnych. Wspomniany typ danych pokazano na **listingu 5**. Funkcję umożliwiającą odczyt wszystkich współczynników korekcyjnych układu BME280 prezentuje **listing 6**. Na **listingu 7**, pokazano funkcję inicjalizacyjną czujnika, a **listingi 8** i **9** prezentują dwie proste funkcje sprawdzające czy czujnik jest w trakcie odczytywania współczynników korekcyjnych (a dokładnie w trakcie przepisywania wartości współczynników korekcyjnych z wbudowanej, nieulotnej pamięci NVM do rejestrów konfiguracyjnych układu) lub w trakcie wykonywania pomiarów (zainicjowanych odrębną funkcją). Kod funkcji, która inicjuje proces wykonania pomiarów przez układ BME280 wygląda jak na **listingu 10**. Konieczność wprowadzenia takiej funkcji wynika z faktu, że układ BME280 został wcześniej skonfigurowany do pracy w trybie forced, czyli takim, w którym pomiary wykonywane są na żądanie, zaś po ich wykonaniu układ automatycznie przechodzi do stanu uśpienia o bardzo niskim poborze mocy (<0,1 µA). Oczywiście może pracować również w normalnym trybie pracy podczas którego cyklicznie wykonuje pomiary bez potrzeby każdorazowej inicjacji procesu pomiarowego, jednak wtedy pobiera znacznie więcej energii, gdyż pomiędzy pomiarami nie przechodzi do stanu uśpienia (pozostaje w stanie Standby, którego czas konfigurujemy dzięki stosownym rejestrom konfiguracyjnym). Na koniec dość skomplikowane funkcje umożliwiające odczyt rzeczywistych (skompensowanych) wartości ciśnienia atmosferycznego, wilgotności i temperatury. Kod funkcji pokazano na **listingach 11**, **12** i **13**. Zostały one napisane na podstawie API dostarczanego

Listing 6. Funkcja umożliwiająca odczyt wszystkich współczynników korekcyjnych

```
void BME280readCoeffs(void)
{
        Cfs.T1 = (uint16_t) BME280readRegister16(BME280_DIG_T1_REG);
        Cfs.T2 = (int16_t) BME280readRegister16(BME280_DIG_T2_REG);
        Cfs.T3 = (int16_t) BME280readRegister16(BME280_DIG_T3_REG);

        Cfs.P1 = (uint16_t) BME280readRegister16(BME280_DIG_P1_REG);
        Cfs.P2 = (int16_t) BME280readRegister16(BME280_DIG_P2_REG);
        Cfs.P3 = (int16_t) BME280readRegister16(BME280_DIG_P3_REG);
        Cfs.P4 = (int16_t) BME280readRegister16(BME280_DIG_P4_REG);
        Cfs.P5 = (int16_t) BME280readRegister16(BME280_DIG_P5_REG);
        Cfs.P6 = (int16_t) BME280readRegister16(BME280_DIG_P6_REG);
        Cfs.P7 = (int16_t) BME280readRegister16(BME280_DIG_P7_REG);
        Cfs.P8 = (int16_t) BME280readRegister16(BME280_DIG_P8_REG);
        Cfs.P9 = (int16_t) BME280readRegister16(BME280_DIG_P9_REG);

        Cfs.H1 = (uint8_t) BME280readRegister8(BME280_DIG_H1_REG);
        Cfs.H2 = (int16_t) BME280readRegister16(BME280_DIG_H2_REG);
        Cfs.H3 = (uint8_t) BME280readRegister8(BME280_DIG_H3_REG);

        Cfs.H4 = (int16_t) ((BME280readRegister8(BME280_DIG_H4_REG) << 4)|
        (BME280readRegister8(BME280_DIG_H5_REG) & 0xF));

        Cfs.H5 = (int16_t) ((BME280readRegister8(BME280_DIG_H5_REG + 1) << 4)|
        (BME280readRegister8(BME280_DIG_H5_REG) >> 4));

        Cfs.H6 = (int8_t) BME280readRegister8(BME280_DIG_H6_REG);
}
```

Listing 7. Funkcja inicjalizacyjna układu BME280

```
void BME280init(void)
{
        //Inicjalizacja interfejsu i2c
        i2cInit();
        //Reset programowy układu BME280
        BME280writeRegister8(BME280_SOFTRESET_REG, DO_RESET);
        _delay_ms(10);
        //Oczekiwanie, aż układ BME280 przepisze wszystkie
        //współczynniki korekcyjne z nieulotnej pamięci NVM
        //do stosownych rejestrów konfiguracyjnych
        //(po włączeniu zasilania)
        while(BME280isReadingCalibData());
        _delay_ms(10);
        //Odczyt współczynników korekcyjnych układu BME280
        //do pamięci RAM mikrokontrolera
        BME280readCoeffs();
        //Ustawiamy oversampling x1 dla pomiaru wilgotności
        BME280writeRegister8(BME280_CONTROLHUMID_REG, H_OVERSMP_X1);
        //Ustawiamy oversampling x1 dla pomiaru
        //temperatury i ciśnienia. Włączamy tryb forced układu BME280.
        BME280writeRegister8(BME280_CONTROL_MEASURE, T_OVERSMP_X1|
        P_OVERSMP_X1|MODE_FORCED);
}
```

Listing 8. Funkcja sprawdzająca czy układ jest w trakcie odczytywania współczynników korekcyjnych

```
uint8_t BME280isReadingCalibData(void)
{
        uint8_t Status = BME280readRegister8(BME280_STATUS_REG);
        return (Status & IM_UPDATE) != 0;
}
```

Listing 9. Funkcja sprawdzająca czy układ jest w trakcie wykonywania pomiarów

```
uint8_t BME280isMeasuring(void)
{
        uint8_t Status = BME280readRegister8(BME280_STATUS_REG);
        return (Status & IS_MEASURING) != 0;
}
```

Listing 10. Funkcja inicjalizująca proces wykonywania pomiarów (w trybie forced)

```
void BME280takeMeasure(void)
{
        //Inicjalizacja nowego pomiaru (niezbędna w trybie FORCED)
        BME280writeRegister8(BME280_CONTROL_MEASURE, T_OVERSMP_X1|
        P_OVERSMP_X1|MODE_FORCED);
        //Oczekiwanie na zakończenie pomiaru
        while(BME280isMeasuring());
        _delay_ms(5);
}
```

przez firmę Bosch Sensortec. Korzystają z globalnej zmiennej *int32_t Tfine* przechowującej temperaturę otoczenia w wysokiej rozdzielczości (do kompensacji ciśnienia i wilgotności). Dość szczegółowy opis układu BME280 był podyktowany tym, że jest bardzo ciekawy element a oprócz drivera firmy Bosch nie ma w Internecie zbyt wielu sprawdzonych przykładów oprogramowania.

Prognozowanie pogody

Stacja pogodowa została wyposażona w pewną istotną funkcjonalność – umożliwia prognozowanie pogody. Aby zrozumieć zasadę działania zastosowanego mechanizmu należy omówić podstawowe zagadnienia związane 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. Zatem 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 pokazuje **rysunek 2**.

Wartość ciśnienia atmosferycznego dla standardowych warunków pogodowych dla wysokości h (n.p.m.) możemy wyznaczyć z uproszczonej zależności opisanej wzorem:

$$P_h = P_0 e^{-\frac{h}{7990}}$$

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. w metrach

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 wiadomo, ciśnienie atmosferyczne ulega ciągłym zmianom, które zależą 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 pogorszenie 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

Listing 11. Funkcja zwracająca wartość rzeczywistego ciśnienia atmosferycznego

```
//Funkcja zwracająca zmierzone ciśnienie
//w formie liczby całkowitej bez znaku.
//Jednostką jest hPa

uint16_t readPressure(void)
{
	int32_t Var1;
	int32_t Var2;
	int32_t Var3;
	int32_t Var4;
	uint32_t Var5;
	uint32_t Pressure;
	int32_t rawPressure = BME280readRegister24(BME280_PRESSUREDATA_REG);

	Var1 = (((int32_t)Tfine) / 2) - (int32_t)64000;
	Var2 = (((Var1 / 4) * (Var1 / 4)) / 2048) * ((int32_t)Cfs.P6);
	Var2 = Var2 + ((Var1 * ((int32_t)Cfs.P5)) * 2);
	Var2 = (Var2 / 4) + (((int32_t)Cfs.P4) * 65536);
	Var3 = (Cfs.P3 * (((Var1 / 4) * (Var1 / 4)) / 8192)) / 8;
	Var4 = (((int32_t)Cfs.P2) * Var1) / 2;
	Var1 = (Var3 + Var4) / 262144;
	Var1 = (((32768 + Var1)) * ((int32_t)Cfs.P1)) / 32768;

	//Uniknięcie dzielenia przez zero
	if(Var1)
	{
		Var5 = (uint32_t)((uint32_t)1048576) - rawPressure;
		Pressure = ((uint32_t)(Var5 - (uint32_t)(Var2 / 4096))) * 3125;
		if (Pressure < 0x80000000) Pressure = (Pressure << 1) / ((uint32_t)Var1);
		else Pressure = (Pressure / (uint32_t)Var1) * 2;

		Var1 = (((int32_t)Cfs.P9) * ((int32_t)(((Pressure / 8) *
		(Pressure / 8)) / 8192))) / 4096;

		Var2 = (((int32_t)(Pressure / 4)) * ((int32_t)Cfs.P8)) / 8192;
		Pressure = (uint32_t)((int32_t)Pressure + ((Var1 + Var2 + Cfs.P7) / 16));
		if(Pressure < 30000) Pressure = 30000;
		else if(Pressure > 110000) Pressure = 110000;
	}
	else Pressure = 30000;

	//Jednostką jest hPa
	return Pressure/100;
}
```

Listing 12. Funkcja zwracająca wartość temperatury otoczenia

```
//Funkcja zwracająca zmierzoną temperaturę
// w formie liczby całkowitej ze znakiem.
//Jednostką jest 0.01'C

int16_t readTemperature(void)
{
	int32_t Var1;
	int32_t Var2;
	int16_t Temperature;
	int32_t rawTemperature = BME280readRegister24(BME280_TEMPDATA_REG);

	Var1 = (int32_t)((rawTemperature / 8) - ((int32_t)Cfs.T1 * 2));
	Var1 = (Var1 * ((int32_t)Cfs.T2)) / 2048;
	Var2 = (int32_t)((rawTemperature / 16) - ((int32_t)Cfs.T1));
	Var2 = (((Var2 * Var2) / 4096) * ((int32_t)Cfs.T3)) / 16384;
	Tfine = Var1 + Var2;
	Temperature = (Tfine * 5 + 128) / 256;

	return Temperature;  //Jednostką jest 0.01'C
}
```

Listing 13. Funkcja zwracająca wartość wilgotności

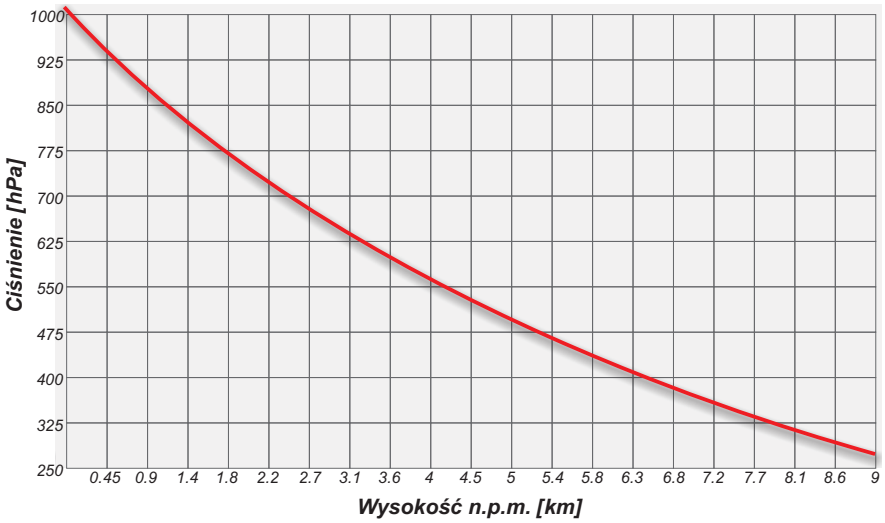
```
//Funkcja zwracająca zmierzoną wilgotność
//w formie liczby całkowitej bez znaku.
//Jednostką jest %

uint8_t readHumidity(void)
{
	int32_t Var1;
	int32_t Var2;
	int32_t Var3;
	int32_t Var4;
	int32_t Var5;
	uint32_t Humidity;
	int32_t rawHumidity = BME280readRegister16(BME280_HUMIDDATA_REG);

	//Zamiana z Little endian na Big endian,
	//bo wilgotność jest w formacie Big endian
	rawHumidity = (rawHumidity>>8)|(rawHumidity<<8);

	Var1 = Tfine - ((int32_t)76800);
	Var2 = (int32_t)(rawHumidity * 16384);
	Var3 = (int32_t)(((int32_t)Cfs.H4) * 1048576);
	Var4 = ((int32_t)Cfs.H5) * Var1;
	Var5 = (((Var2 - Var3) - Var4) + (int32_t)16384) / 32768;
	Var2 = (Var1 * ((int32_t)Cfs.H6)) / 1024;
	Var3 = (Var1 * ((int32_t)Cfs.H3)) / 2048;
	Var4 = ((Var2 * (Var3 + (int32_t)32768)) / 1024) + (int32_t)2097152;
	Var2 = ((Var4 * ((int32_t)Cfs.H2)) + 8192) / 16384;
	Var3 = Var5 * Var2;
	Var4 = ((Var3 / 32768) * (Var3 / 32768)) / 128;
	Var5 = Var3 - ((Var4 * ((int32_t)Cfs.H1)) / 16);
	Var5 = (Var5 < 0 ? 0 : Var5);
	Var5 = (Var5 > 419430400 ? 419430400 : Var5);
	Humidity = (uint32_t)(Var5 / 4096);
	if (Humidity > 102400) Humidity = 102400;

	return Humidity/1024;  //Jednostką jest %
}
```



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

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. Jednak dla prostych aplikacji możemy posiłkować się algorytmami, z jakich korzysta większość elektronicznych stacji pogodowych, a które oparte są tylko na śledzeniu zmian ciśnienia atmosferycznego. Z uwagi na charakter zachodzących zmian prognozowanie pogody jest znacznie utrudnione dla obszarów położonych wysoko n.p.m. Na koniec, warto podkreślić istotną różnicę pomiędzy wartością bezwzględnego ciśnienia atmosferycznego dla danej miejscowości (ciśnienia tam panującego) a wartością ciśnienia podawanego w prognozach pogody. Ostatnie jest ciśnieniem, jakie wystąpiłoby danego dnia w danej miejscowości, gdyby znajdowała się ona na poziomie morza, czyli 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.

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 ph) 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 zależności pokazane w **tabeli 1**. Aby zapewnić realizację funkcjonalności prognozowania pogody nasze urządzenie pozwala ustawić wysokość nad poziomem morza, na jakiej się znajduje. W tym celu przewidziano odpowiednią funkcjonalność systemu Menu.

Budowa

Kilka słów komentarza wymaga blok zasilający, zbudowany z zastosowaniem specjalizowanego kontrolera ładowania akumulatorów li-ion i li-pol pod postacią układu MCP73832 firmy Microchip. Integruje w sobie kompletny system ładowania, którego podstawowe cechy to:

- szeroki zakres napięć zasilania 3,75...6 V,
- programowany prąd szybkiego ładowania 15...500 mA (rezystor R1),
- możliwość wyboru wartości prądu ładowania wstępnego,
- możliwość wyboru poziomu naładowania akumulatora,
- wbudowany mechanizm wykrywania podłączonego akumulatora,
- trójstanowe wyjście statusu procesu ładowania STAT,
- automatyczne przejście do trybu power-down,

Stosując układ w aplikacji prostej ładowarki jedynym „zmartwieniem" użytkownika jest ustawienie prądu szybkiego ładowania, poprzez dobór rezystora podłączonego pomiędzy wyprowadzenie PROG a masę. Prąd ten dobieramy według poniższej zależności:

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

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

W naszym urządzeniu rezystor RPROG(R1) ma wartość 10 kΩ, co ustawia prąd ładowania szybkiego na wartość 100 mA. Taka wartość została celowo wybrana. Do zasilania naszego urządzenia przewidziano podłączenie go do portu USB komputera (lub popularnego zasilacza sieciowego z portem USB przeznaczonego do ładowania telefonów komórkowych), który zwykle pozwala na maksymalny pobór prądu rzędu 500 mA w trybie high-power, zaś typowo 100 mA.

Dodatkowego wyjaśnienia wymaga opcjonalny układ współdzielenia obciążenia zbudowany przy użyciu tranzystora T1 typu MOSFET z kanałem P, diody Schottky'ego D1 oraz rezystorów R2 i R3. Takie rozwiązanie było konieczne, aby nie zaburzać działania kontrolera ładowania, ponieważ odbiornik pobierając nieustannie prąd z ogniwa, a więc z układu nadzorującego, nie pozwoli na skuteczną detekcję końca procesu ładowania. W przypadku obecności napięcia USB (czyli stanu wysokiego na bramce tranzystora) tranzystor T1 przechodzi w stan wyłączenia, odłączając tym samym ładowany akumulator

Tabela 1. Zależności opisujące prognozowanie pogody, zastosowane w urządzeniu

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

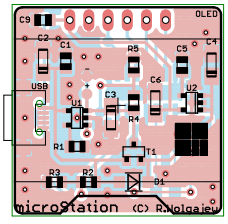
od obciążenia. W tym czasie obciążenie, jakim jest stacja pogodowa, zostaje zasilone bezpośrednio z napięcia portu USB poprzez diodę D1. W przypadku odłączenia napięcia zasilającego USB, bramka tranzystora T1 zostaje ściągnięta do masy powodując przewodzenie tranzystora, a tym samym zasilenie urządzenia z akumulatora. Wtedy dioda D1 zabezpiecza przed przepływem prądu wstecznego tj. z akumulatora w kierunku źródła napięcia zasilającego (USB). Wyjście z układu ładowania wprowadzono na wejście stabilizatora LDO typu AP2127K-2.8TRG1, który zapewnia stały poziom napięcia zasilającego system mikroprocesorowy (2,8 V) niezależnie od stanu układu ładowania.

Aplikacja urządzenia „microStation" korzysta z wbudowanego w strukturę mikrokontrolera przetwornika ADC (wejście ADC6), przy udziale którego (poprzez prosty dzielnik napięciowy R4, R5) mierzony jest poziom napięcia akumulatora zasilającego co pozwala na orientacyjne określenie stanu jego naładowania. Jest to oczywiście rozwiązanie dość proste, gdyż dokładne określenie stanu naładowania akumulatora wymagałoby kontrolowania wielkości gromadzonego i traconego ładunku, jednak w tak prostym systemie jest w zupełności wystarczające.

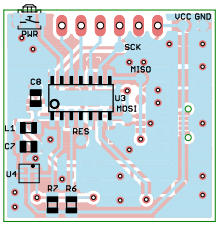
Montaż

Schemat montażowy urządzenia microStation pokazano na **rysunkach 3** i **4**.

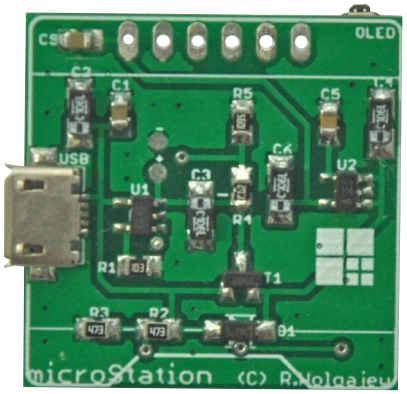
Niewielki, dwustronny obwód drukowany z przewagą elementów SMD wymiarami zbliżony jest do wymiarów płytki drukowanej zastosowanego modułu wyświetlacza OLED (28×28 mm). Montaż urządzenia rozpoczynamy od warstwy BOTTOM, gdzie przylutowujemy układ BME280. Proces ten najłatwiej wykonać przy użyciu stacji lutowniczej na gorące powietrze (tzw. Hot Air) i odpowiednich stopów lutowniczych, gdyż obudowa tego elementu nie pozwala na zastosowanie typowej stacji lutowniczej.



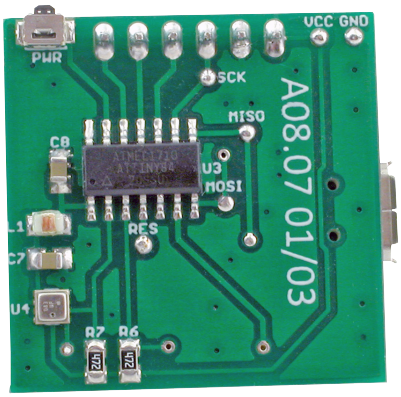
Rysunek 3. Schemat płytki PCB wraz z rozmieszczeniem elementów, strona TOP



Rysunek 4. Schemat płytki PCB wraz z rozmieszczeniem elementów, strona BOTTOM



Fotografia 5. Widok zmontowanej płytki od strony TOP, bez wlutowanego wyświetlacza OLED



Fotografia 6. Widok zmontowanej płytki od strony BOTTOM

Następnie lutujemy pozostałe elementy półprzewodnikowe, potem rezystory i kondensatory, dławik i na samym końcu wyłącznik PWR. Na warstwie TOP dokonujemy montażu w tej samej kolejności kończąc przylutowaniem gniazda USB. Z uwagi na zagęszczenie wyprowadzeń układów scalonych przed pierwszym podłączeniem urządzenia 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 topnika. Na samym końcu, do tak przygotowanej płytki, montujemy wyświetlacz OLED, zwyczajnie lutując jego wyprowadzenia w przeznaczone do tego celu pola lutownicze. Połączenia te zapewniają jednocześnie wystarczający montaż mechaniczny. Wybierając konkretny moduł wyświetlacza dostępny w handlu należy zakupić wersję wyposażoną w następujące sygnały sterujące: CLK (sygnał zegarowy magistrali SPI), MOSI (wejście danych magistrali SPI), CS (wejście wyboru sterownika SSD1306) oraz DC (wejście, decydujące o charakterze wysyłanych danych: 1→ dane pamięci obrazu, 0→ rozkaz sterujący). Nie mniej ważne jest rozmieszczenie sygnałów zasilających, jako że moduły dostępne w handlu mają często zamienione miejscami sygnały zasilania (VCC) i masy (GND). Akumulator zasilający podłączamy do wyprowadzeń +/– po stronie TOP i umieszczamy pomiędzy modułem wyświetlacza OLED a płytką naszego urządzenia. Poprawnie zmontowany układ powinien działać od razu po podłączeniu zasilania. Widok zmontowanego urządzenia microStation (od strony TOP) bez wlutowanego wyświetlacza OLED prezentuje **fotografia 5**, a od strony bottom **fotografia 6**.

Obsługa

Na **rysunku 7** pokazano wygląd poszczególnych ekranów Menu stacji pogodowej. Zmiana ekranu Menu następuje w sposób automatyczny co 2 sekundy. Długie naciśnięcie przycisku PWR powoduje wyłączenie urządzenia, zaś kolejne krótkie naciśnięcie, jego ponowne włączenie. Warto również podkreślić, iż podczas włączania urządzenia sprawdzany jest stan naładowania wbudowanego akumulatora i w przypadku jego rozładowania na ekranie pokazywany jest stosowny symbol (przekreślonej baterii) po czy urządzenie jest automatycznie wyłączane. Podłączeniu urządzenia do ładowania towarzyszy zmiana symbolu baterii (w prawym górnym rogu ekranu) na symbol wtyczki kabla USB, przy czym do czasu pełnego naładowania symbol ten pozostaje niewypełniony, zaś po pełnym naładowaniu zostaje wypełniony, co informuje o zakończeniu procesu ładowania. Po odłączeniu urządzenia od ładowarki ponownie pokazywany jest symbol baterii zasilającej, przy czym stopień jego wypełnienia zależy od stanu naładowania akumulatora (w 10 krokach). Krótkie naciśnięcie przycisku PWR obsługiwane jest wyłącznie w trybie pokazywania ciśnienia atmosferycznego i powoduje każdorazową zmianę wysokości nad poziomem morza (co 10 m) na jakiej znajduje się nasze urządzenie, co, jak powiedziano wcześniej, niezbędne jest z punktu widzenia funkcji prognozowania pogody.

Robert Wołgajew
robert.wolgajew@ep.com.pl

Rysunek 7. Wygląd poszczególnych ekranów Menu