Podstawowe parametry:

Węzeł pomiarowy:
- źródło napięcia zasilania: bateria CR2032,
- maksymalny prąd obciążenia (tryb uśpienia/nadawanie): 10 µA/22,6 mA,
- liczba adresów: 8,
- zakres mierzonych temperatur: –30...50°C,
- rozdzielczość mierzonych temperatur: 0,5°C,
- dokładność pomiaru temperatury (typowa): 0,4°C,
- częstość transmisji danych: co 64 s,
- częstotliwość pracy transceivera: 868 MHz,
- zasięg w terenie otwartym: ok. 100 m.

Moduł odbiornika:
- napięcie zasilania: 4...9 V,
- maksymalny prąd obciążenia: 30 mA,
- częstotliwość pracy transceivera: 868 MHz,
- zasięg w terenie otwartym: ok. 100 m.

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

AVT5949	Energooszczędny termometr LED (EP 08/2022)
AVT5892	Energooszczędny termometr z kalibracją (EP 10/2021)
AVT5635	Bezprzewodowy, energooszczędny system pomiaru temperatury (EP 8–9/2018)
AVT5623	4-kanałowy termometr z interfejsem Wi-Fi (EP 4/2018)
AVT5566	THPStation – rozbudowany termometr z Wi-Fi (EP 1/2017)
AVT5535	Termometr 2-kanałowy z interfejsem Bluetooth (EP 4/2016)
AVT5518	Termometr bezprzewodowy (EP 11/2015)
AVT1863	Termometr z interfejsem Bluetooth (EP 8/2015)
AVT1790	Termometr XXL (EP 2/2014)
AVT5489	8-kanałowy termometr z alarmem i wyświetlaczem LCD (EP 11/2013)
AVT5420	Wielopunktowy termometr z rejestracją (EP 10/2013)
AVT1734	Termometr do wędzarni (EP 4/2013)
AVT5373	Tlogger – rejestrator temperatury (EP 12/2012)
AVT1705	Moduł do pomiaru temperatury z interfejsem RS485 (EP 9/2012)
AVT1697	Wielogabarytowy termometr LED (EP 8/2012)
AVT5389	4-kanałowy termometr z wyświetlaczem LED (EP 5/1012)
AVT5330	Termometr PC (EP 2/2012)
AVT5301	Wskaźnik komfortu cieplnego z wbudowanym kalendarzem sezonowym (EP 7/2011)
AVT1582	Domowy termometr RGB (EP 8/2010)
AVT5230	Rejestrator temperatury z interfejsem USB (EP 4/2010)
AVT5205	System pomiaru temperatury z termoparą typu K (EP 10/2009)
AVT5117	Termometr USB (EP 11/2007)
AVT5108	2-kanałowy termometr z dwukolorowym wyświetlaczem LED (EP 8/2007)
AVT5041	Termometr MIN-MAX (EP 11/2001)
AVT3122	Termometr z wyświetlaczem LED

W ofercie AVT*

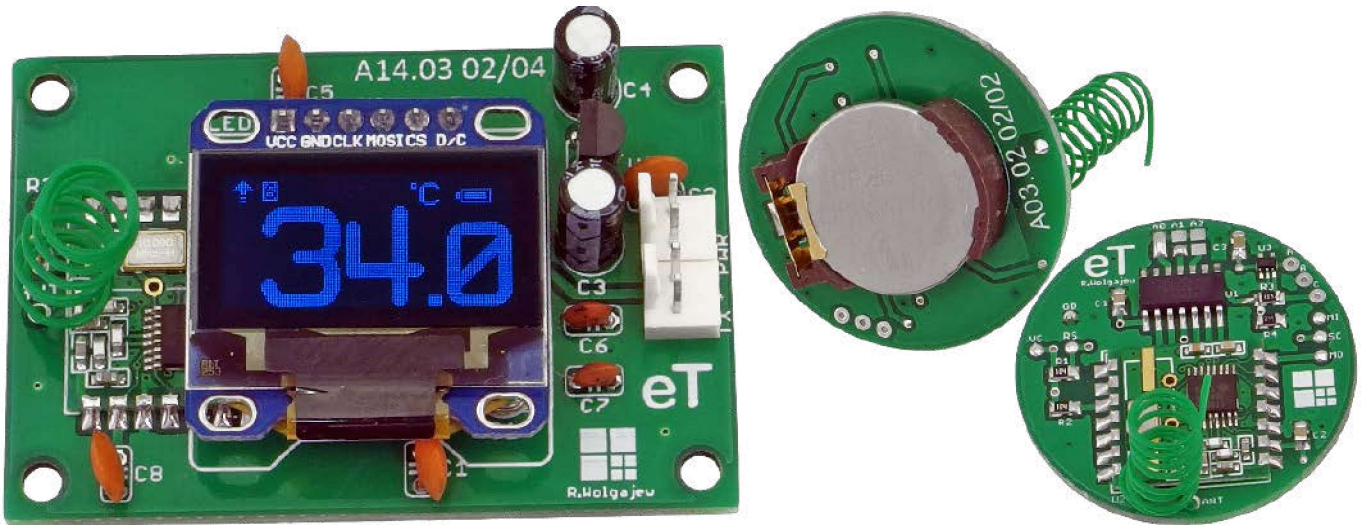
AVT5952

* 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.

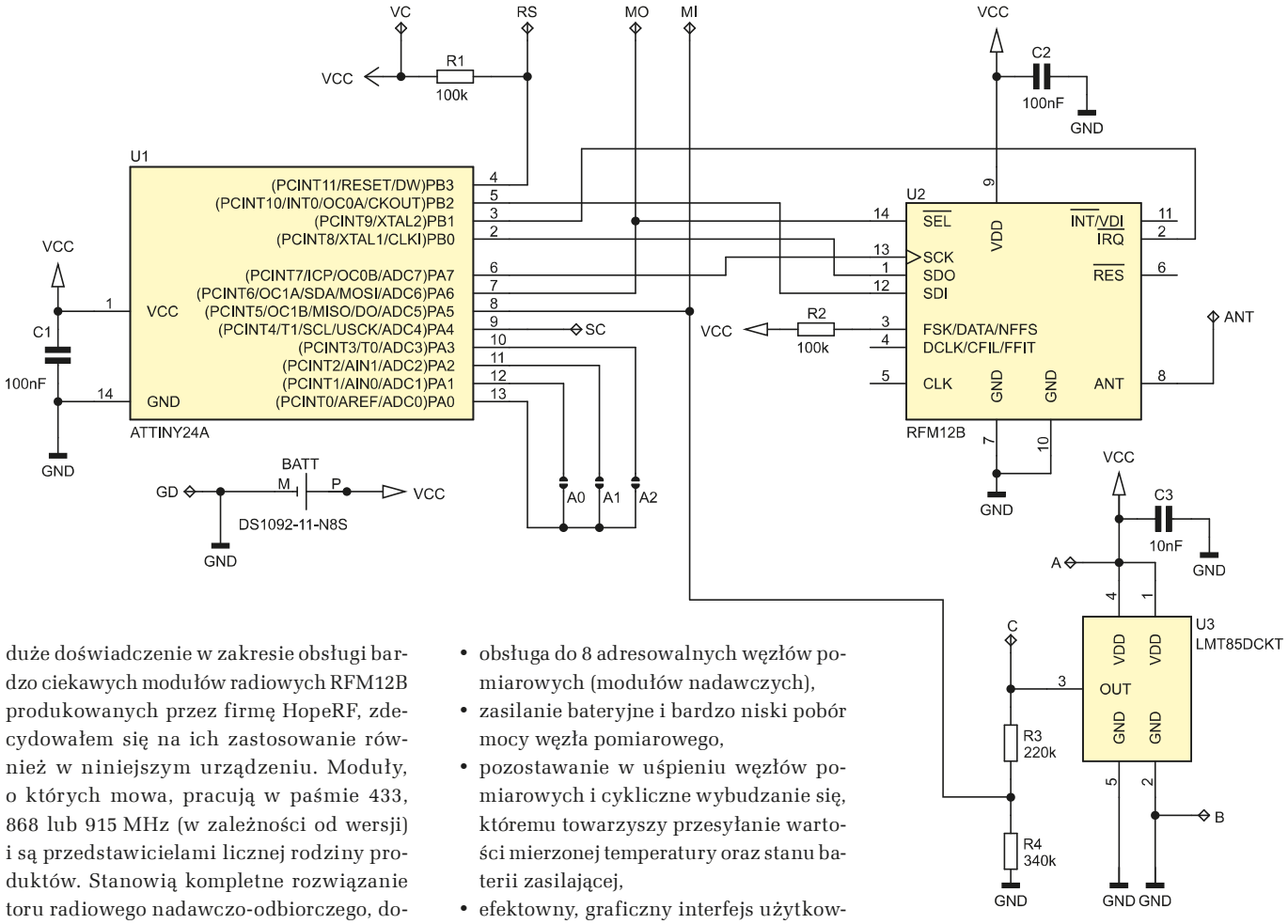


eT – wielokanałowy, bezprzewodowy system pomiaru temperatury

W swojej praktyce inżynierskiej kilkakrotnie podejmowałem wyzwanie skonstruowania systemu bezprzewodowej akwizycji danych. Tym razem stanąłem przed wyzwaniem skonstruowania prostego systemu bezprzewodowego pomiaru temperatury, który, po pierwsze, nie wymagałby żadnej konfiguracji, zaś po drugie, odznaczał się walorem w postaci niskiego poboru energii przez węzły pomiarowe. Co więcej, zależało mi na użyciu łatwo dostępnych i tanich modułów radiowych wyposażonych w tryb niskiego poboru mocy.

Do tej pory opracowałem zarówno dość proste konstrukcje zawierające nieskomplikowane moduły radiowe pracujące w paśmie 433 MHz i pozbawione jakiegokolwiek stosu komunikacyjnego, jak i zaawansowane układy pracujące na bazie technologii ZigBee, a więc korzystające z wszelkich dobrodziejstw tego medium komunikacyjnego. Za każdym razem był to jednak pewien kompromis pomiędzy funkcjonalnością tak zbudowanego systemu, kosztem jego implementacji a energooszczędnością.

Tym razem stanąłem przed wyzwaniem skonstruowania prostego systemu bezprzewodowego, który będzie odznaczał się niskim poborem energii przez węzły pomiarowe. Jako że w moim poprzednim projekcie energooszczędnego systemu pomiaru temperatury z EP 8/18, EP 9/18 zdobyłem



Rysunek 1. Schemat ideowy węzła pomiarowego

duże doświadczenie w zakresie obsługi bardzo ciekawych modułów radiowych RFM12B produkowanych przez firmę HopeRF, zdecydowałem się na ich zastosowanie również w niniejszym urządzeniu. Moduły, o których mowa, pracują w paśmie 433, 868 lub 915 MHz (w zależności od wersji) i są przedstawicielami licznej rodziny produktów. Stanowią kompletne rozwiązanie toru radiowego nadawczo-odbiorczego, dostarczając wygodny interfejs komunikacyjny SPI pozwalający na przeprowadzenie pełnej konfiguracji elementu w ramach dostępnej szerokiej palety ustawień i sterowanie komunikacją radiową. W tym miejscu nie będę powtarzał informacji dotyczących specyfikacji i obsługi tych peryferiów, gdyż takowe zamieściłem w ramach wspomnianych wcześniej artykułów, w związku z czym zainteresowanych tymi szczegółami Czytelników odsyłam do wskazanych wcześniej wydań EP.

Budowa i działanie

Przejdźmy od razu do szczegółów konstrukcyjnych przedmiotu niniejszego artykułu, a mianowicie bezprzewodowego, energooszczędnego systemu wielopunktowego pomiaru temperatury, który w założeniach charakteryzować się ma następującymi cechami funkcjonalnymi:

- obsługa do 8 adresowalnych węzłów pomiarowych (modułów nadawczych),
- zasilanie bateryjne i bardzo niski pobór mocy węzła pomiarowego,
- pozostawanie w uśpieniu węzłów pomiarowych i cykliczne wybudzanie się, któremu towarzyszy przesyłanie wartości mierzonej temperatury oraz stanu baterii zasilającej,
- efektowny, graficzny interfejs użytkownika po stronie układu nadrzędnego (modułu odbiorczego),
- wysoka ergonomia obsługi całego systemu i brak konieczności konfiguracji,
- kontrola aktywności węzłów pomiarowych przez układ nadrzędny.

Kilka niezbędnych słów uwagi należy się zasilaniu węzłów pomiarowych. Założono, że moduł nadawczy będzie pracował z zasilaniem bateryjnym w postaci pastylki CR2032 i większość swojego czasu będzie pozostawał w uśpieniu (dla ograniczenia poboru mocy), czekając na wybudzenie przez odpowiednio skonfigurowany podsystem mikrokontrolera nazywany watchdogiem. Wspomnianemu wybudzeniu (mikrokontrolera i modułu RF) towarzyszyć będzie wysłanie komunikatu do adresowalnego układu nadrzędnego i ponowne uśpienie urządzenia. W ten prosty sposób ograniczamy do minimum pobór energii ze źródła zasilania, pozwalając na wieloletnią pracę urządzenia. Uważny Czytelnik dostrzeże pewne ograniczenia i sformułuje związane z nimi zapytania. Otóż bateria CR2032 przeznaczona jest do zasilania urządzeń (3 V) cechujących się bardzo niskim poborem prądu rzędu ułamków mA do pojedynczych mA. Nasz układ po wybudzeniu aktywuje nadajnik modułu RFM12B, który w czasie transmisji pobiera prąd rzędu 22,6 mA (maksymalnie), co stanowi bardzo duże obciążenie dla baterii zasilającej.

Na szczęście transmisja trwa około 12 ms, w związku z czym pobrana energia jest niewielka, niemniej jednak takie obciążenie skromnej baterii niesie za sobą pewne reperkusje. Po pierwsze, z czasem spada jej znamionowa pojemność, napięcie znamionowe oraz wzrasta rezystancja wewnętrzna. Spadek

WYKAZ ELEMENTÓW, które możesz zamówić w sklepie AVT na stronie sklep.avt.pl lub bezpośrednio (ul. Leszczynowa 11, 03-197 Warszawa, tel. 48222578451, e-mail: handlowy@avt.pl):

Węzeł pomiarowy
Rezystory: (SMD 0805)
R1, R2: 100 kΩ
R3: 220 kΩ 1%
R4: 340 kΩ 1%

Kondensatory: (SMD 0805)
C1, C2: 100 nF
C3: 10 nF

Półprzewodniki:
U1: ATtiny24A (SOIC14)
U2: RFM12B–866MHz (SMD)
U3: LMT85DCKT (SC-70 lub TO-92)

Pozostałe:
BATT: koszyczek baterii CR2032 typu CONNFLY
DS1092-11-N8S

Moduł odbiornika
Rezystory: (miniaturowe 1/8 W)
R1, R2: 100 kΩ

Kondensatory:
C1, C2, C5, C8: 100 nF
C3, C4: 100 µF/16 V
C6, C7:22 pF

Półprzewodniki:
U1: ATtiny84 (DIL14)
U2: LP2950ACZ-3.0/NOPB (TO-92)
U3: RFM12B–866MHz (SMD)
OLED: wyświetlacz OLED 0,96" 128×64, niebieski, SPI

Pozostałe:
PWR, TX: gniazdo męskie 2 piny (NSL25-2W)
Q1: rezonator kwarcowy 4 MHz niski

pojemności nie jest jakiś drastycznie wielki, ale można go szacować na 25%, przy spadku napięcia baterii do 2,2 V. Zagadnienie jest naprawdę bardzo ciekawe, w związku z czym zachęcam ambitnych Czytelników do zgłębienia tematu pod adresem internetowym [1], gdzie inżynierowie firmy Energizer i Nordic Semiconductor bardzo drobiazgowo zaprezentowali ten interesujący temat w dokumencie o nazwie *High pulse drain impact on CR2032 coin cell battery capacity*.

Na szczęście główne podzespoły nadajnika pracują już przy niewielkich napięciach zasilających (moduł RF: 2,2 V, mikrokontroler: 1,8 V) i nawet przy 25% spadku pojemności tego typu aplikacja powinna zapewnić długą pracę urządzenia. Aby ocenić, jak długo węzeł pomiarowy pracował będzie na pojedynczej baterii CR2032, należy zastanowić się, z jakich etapów składa się praca węzła pomiarowego i jakie są wtedy prądy pobierane ze źródła napięcia zasilającego. Przystępując do obliczeń, przyjąłem następujący podział cyklu pracy urządzenia:

- etap trybu *power-down* (uśpienia), który trwa z dużym przybliżeniem 24 h/dobę i podczas którego pobierany jest prąd rzędu 10 µA,
- etap pomiarów przetwornika ADC (po wybudzeniu), który trwa średnio 3 ms i podczas którego pobierany jest prąd rzędu 0,4 mA,
- etap transmisji modułu RFM12B (po wybudzeniu), który trwa średnio 12 ms i podczas którego pobierany jest prąd rzędu 22,6 mA.

Założono ponadto, że pomiary przetwornika ADC i następująca po nich transmisja wykonywane są co 64 s, a więc 1350 razy na dobę. Przy tych założeniach otrzymano ponad 17 miesięcy pracy węzła pomiarowego na pojedynczej baterii CR2032 (o zredukowanej o 25% pojemności), co wydaje się wartością co najmniej zadowalającą.

Przejdźmy zatem do schematu układu podrzędnego, czyli węzła pomiarowego, który został pokazany na **rysunku 1**. Jak widać, zaprojektowano bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler ATtiny24A (niskonapięciowy) taktowany wewnętrznym oscylatorem 1 MHz sterujący pracą modułu transceivera dzięki realizacji programowej obsługi interfejsu SPI oraz obsłudze przerwania zewnętrznego *Pin Change Interrupt 1* (wyprowadzenie PCINT9) odpowiedzialnego za mechanizm wysyłania danych. Ponadto, dzięki użyciu przetwornika ADC, wbudowanego w mikrokontroler sterujący, możliwy stał się pomiar temperatury przetwornika temperatura/napięcie pod postacią układu LMT85 oraz pomiar napięcia baterii zasilającej.

Co ciekawe, na pierwszy rzut oka nie wydaje się, aby nasz sterownik w jakikolwiek sposób używał przetwornika ADC do pomiaru napięcia baterii zasilającej, gdyż żaden z jego kanałów wejściowych nie jest przez niego używany w tym celu. To prawda, patrząc na schemat układu i nie mając do dyspozycji listingu programu, można by wysnuć taki wniosek. Jest jednak zgoła inaczej. Nasz przetwornik ADC mierzy w takim przypadku specjalne, wewnętrzne napięcie odniesienia V_{BG}=1,1 V dzięki temu, że wewnętrzny, analogowy multiplekser przetwornika może zostać właśnie w ten sposób ustawiony. Napięciem odniesienia jest w takim wypadku napięcie zasilające mikrokontroler, czyli napięcie dostarczane na wyprowadzenie VCC. Skoro mierzone napięcie ma wartość stałą (1,1 V), zaś napięcie odniesienia wartość zmienną, w prosty sposób możemy ustalić jego wartość. Prawda, że ciekawe? Dzięki temu prostemu trikowi możliwy stał się pomiar napięcia baterii zasilającej bez angażowania dodatkowego pinu mikrokontrolera.

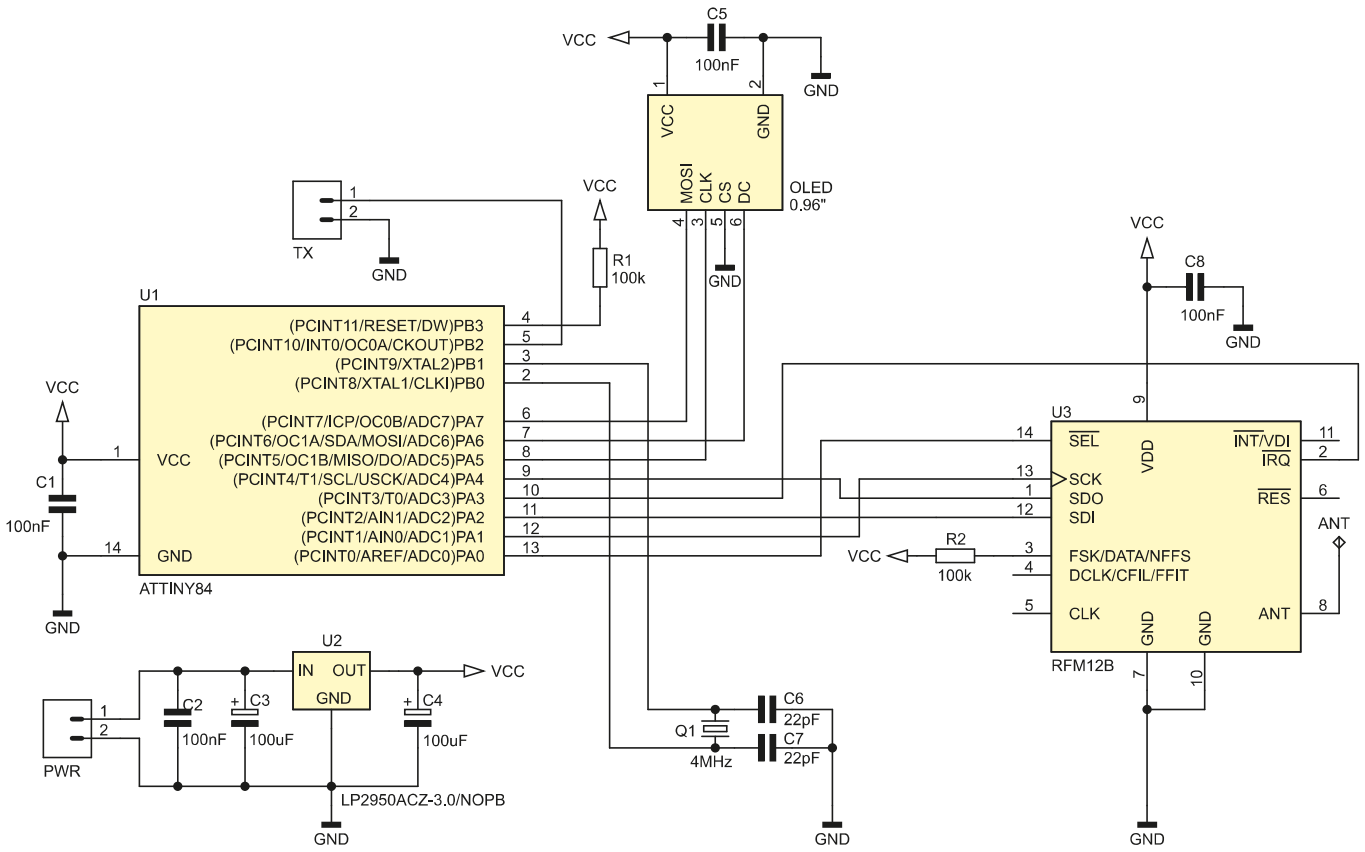
Dodatkowo, mikrokontroler obsługuje 3 wyprowadzenia adresowe A0…A2 (pola lutownicze), pozwalające na ustawienie adresu sprzętowego węzła pomiarowego.

Zgodnie z tym, co napisano już wcześniej, nasz węzeł pomiarowy powinien charakteryzować się minimalnym zapotrzebowaniem na energię elektryczną, jako że jest zasilany niewielką baterią CR2032. W związku z powyższym zastosowano poniższe mechanizmy programowo-sprzętowe:

- wyłączono wszystkie nieużywane peryferia mikrokontrolera (komparator analogowy, TIMER1, TIMER0, USI),
- wprowadzono mikrokontroler w tryb niskiego poboru mocy

Listing 1. Ciało funkcji main węzła pomiarowego

```
int main(void) {
    char Data[3];
    uint8_t timer8s = 0, Address;

    _delay_ms(1000);

    //Nieużywane, jako wyjściowe ze stanem 0
    DDRA |= (1<<PA4);
    //Port sztucznej masy, jako wyjściowy ze stanem 0
    GND_DDR |= (1<<GND_NR);
    //Podciągnięcie portu adresu do VCC
    ADDRESS_PORT |= (1<<PA3)|(1<<PA2)|(1<<PA1);
    _delay_ms(1);
    //Odczytanie adresu urządzenia
    Address = (ADDRESS_PIN & 0b1110) >> 1;
    //Port sztucznej masy, jako wyjściowy ze stanem 1
    GND_PORT |= (1<<GND_NR);
    //Redukcja poboru mocy przez wyłączenie modułów
    //(lub ich zegarów): TIMER1, TIMER0, USI
    PRR = (1<<PRTIM1)|(1<<PRTIM0)|(1<<PRUSI);
    //Wyłączenie komparatora analogowego
     dla zmniejszenia poboru mocy
    ACSR = (1<<ACD);
    //Uruchomienie i konfiguracja watchdoga:
    //Watchdog Timeout Interrupt Enable, Timeout: 1024K cycles -> 8.0 s
    WDTCSR = (1<<WDIE)|(1<<WDP3)|(1<<WDP0);
    //Uruchomienie i konfiguracja RFM12B, w tym interfejsu SPI
    RFM12bInit(MASTER_ID);
    //Wprowadzmy moduł RFM12B w tryb power-down
    RFM12bPowerDown();


    while(1) {
        cli();
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
        sleep_enable();
        sei();
        sleep_cpu();
        //-------------------------------------------------------------
        //W tym miejscu śpimy i czekamy na wybudzenie
        //przez zmianę stanu przycisku SW
        //-------------------------------------------------------------
        sleep_disable();

        //Co 64 sekundy wysyłamy dane do mastera
        if(timer8s == 0)
        {
            //Wykonanie niezbędnych pomiarów
            Data[0] = Address;  //Adres sprzętowy naszego urządzenia
            //Mierzona temperatura
            Data[1] = ADCmeasure(ADC_CHANNEL_TEMP);
            //Napięcie zasilania
            Data[2] = ADCmeasure(ADC_CHANNEL_VCC);
            //Wyłączenie ADC dla oszczędzania energii
            ADC_STOP_MEASURE;

            //Wychodzimy z trybu power-down modułu RFM12b
            RFM12bPowerUp();

            //Uruchomienie przerwania odpowiedzialnego za nadawanie
            GIFR |= (1<<PCIF1);
            GIMSK |= (1<<PCIE1);

            //Inicjujemy wysłanie ramki danych
            RFM12bStartTx(Data, 3, MASTER_ID);
            //Czekamy na zakończenie transmisji
            while(RFM12B.Status != PACKET_SENT);

            //Wyłączenie przerwania odpowiedzialnego za nadawanie
            GIMSK &= ~(1<<PCIE1);

            //Wprowadzmy moduł RFM12B w tryb power-down
            RFM12bPowerDown();
        }
        timer8s = (timer8s+1) & 0b111;
    }
}
```



Rysunek 2. Schemat ideowy układu nadrzędnego systemu pomiarowego

power-down, z którego wybudzany jest cyklicznie co 8 s poprzez odpowiednio skonfigurowany układ watchdog, którego zadziałanie nie powoduje zresetowania mikrokontrolera, tylko wywołanie stosownego przerwania systemowego mającego możliwość wybudzania mikrokontrolera. Wybudzony mikrokontroler raz na 8 wybudzeń (czyli co 64 s) inicjuje transmisję danych, po czym przechodzi ponownie w tryb *power-down*,

- nieużywany transceiver RFM12B wprowadzany jest każdorazowo w tryb niskiego poboru mocy.

Prawda, że proste? Nasz węzeł pomiarowy, pracując wyłącznie w trybie nadajnika, wysyła każdorazowo 3 bajty danych użytecznych: swój adres sprzętowy, wartość zmierzonej temperatury zewnętrznej (ze znakiem) oraz wartość napięcia zasilania. Nie jest sprawdzana obecność częstotliwości nośnej, a więc zajętość pasma transmisji, ale przeprowadzone testy praktyczne wykazały, że ryzyko kolizji danych jest naprawdę pomijalne, zwłaszcza przy tak skonstruowanym systemie pomiarowym.

Na **listingu 1** pokazano wygląd funkcji main programu obsługi węzła pomiarowego, realizującą całą, założoną funkcjonalność.

Tyle w kwestii konstrukcji oprogramowania węzła pomiarowego, a zatem czas na omówienie schematu ideowego jednostki nadrzędnej, której zadaniem jest obsługa węzłów pomiarowych i wizualizacja przesyłanych danych.

Schemat ideowy układu nadrzędnego pokazano na **rysunku 2**. Tak jak poprzednio, zaprojektowano bardzo prosty system mikroprocesorowy, którego sercem jest niewielki mikrokontroler ATtiny84 taktowany rezonatorem kwarcowym o częstotliwości 4 MHz sterujący pracą modułu transceivera dzięki realizacji programowej obsługi interfejsu SPI oraz obsłudze przerwania zewnętrznego *Pin Change Interrupt 0* (wyprowadzenie PCINT3) odpowiedzialnego za mechanizm odbierania danych. Ponadto mikrokontroler obsługuje graficzny wyświetlacz OLED o rozdzielczości 128×64 stanowiący interfejs użytkownika (także dzięki realizacji programowej obsługi interfejsu SPI) oraz programowy interfejs USART (wyjście TX) pozwalający na transmisję pomiarów do innych systemów akwizycji danych.

Zastosowanie programowej implementacji wyjściowego interfejsu USART wynikało z faktu, że zastosowany rodzaj mikrokontrolera nie ma sprzętowej instancji tego rodzaju peryferium. Dodatkowo, dla maksymalnego uproszczenia implementacji, przyjęto następujące założenia dotyczące parametrów transmisji: prędkość transmisji równa 9600 bitów na sekundę, 1 bit startu, 1 bit stopu, bez bitu kontroli parzystości.

Przejdźmy zatem do zagadnień implementacyjnych odpowiedzialnych za to zadanie. Do realizacji programowego nadajnika

Listing 2. Plik nagłówkowy modułu programowego USART-a

```
#define USART_TX_PORT PORTB
#define USART_TX_DDR DDRB
#define USART_TX_NR PB2
#define USART_MAX_BYTES 8
#define STOP_TRANSMISSION TCCR0B = 0x00
//Preskaler=8
#define START_TRANSMISSION TCCR0B = (1<<CS01)

typedef struct {
    //Dane przeznaczone do wysłania
    uint8_t Data[USART_MAX_BYTES];
    //Wskaźnik liczby danych do wysłania
    uint8_t Bytes;
}usartType;

extern volatile usartType USART;
```

Listing 3. Funkcja inicjalizacyjna nadajnika programowego USART-a

```
void initUsart(void){
    //Port USART TX, jako wyjściowy ze stanem 1 – IDLE
    USART_TX_PORT |= (1<<USART_TX_NR);
    USART_TX_DDR |= (1<<USART_TX_NR);
    //Konfiguracja Timer0,
    //wywołującego przerwanie 9600 razy na sekundę
    //CTC
    TCCR0A = (1<<WGM01);
    //Przerwanie 9600 razy na sekundę
    OCR0A = 51;
    //Timer/Counter0, Output Compare A Match Interrupt Enable
    TIMSK0 = (1<<OCIE0A);
}
```

interfejsu USART o podanych wyżej parametrach zastosowano układ czasowo-licznikowy Timer0, skonfigurowany do pracy w trybie CTC w taki sposób, by jego przerwanie od porównania wywoływane było 9600 razy na sekundę, a więc co 104 µs. To właśnie we wspomnianym przerwaniu, zupełnie w tle programu głównego, odbywa się przesyłanie danych.

Pora na trochę kodu. Na początek plik nagłówkowy modułu programowego USART-a, który pokazano na **listingu 2**. Jak widać, wprowadzono zmienną strukturalną USART typu *usartType*, która nieco porządkuje kod modułu. Dalej, na **listingu 3** pokazano funkcję inicjalizacyjną nadajnika programowego USART-a. Dalej funkcja obsługi przerwania od porównania układu czasowo-licznikowego Timer0 realizująca właściwe wysyłanie danych, którą pokazano na **listingu 4**. Prawda, że proste? Oczywiście nie jest to rozwiązanie uniwersalne, a szyte na miarę, niemniej jednak myślę, że na tej podstawie każdy Czytelnik może dostosować je do swoich potrzeb.

Na sam koniec omówię funkcję *main* programu obsługi układu nadrzędnego, realizującą całą, założoną funkcjonalność. Ciało funkcji pokazano na **listingu 5**. Jak widać, funkcja *main* wywołuje 3 inne funkcje narzędziowe o nazwach: *handleNode()*, *handleNodeTimer()* i *redrawScreen()*. Dwie pierwsze odpowiadają za obsługę węzłów pomiarowych, zaś ostatnia generuje zawartość ekranu graficznego interfejsu użytkownika. Na **listingu 6** pokazano ciało funkcji *handleNode()* odpowiedzialnej za obsługę nadchodzących ramek wysyłanych przez węzły pomiarowe oraz wysyłanie odebranych danych za pośrednictwem interfejsu USART, zaś na **listingu 7** pokazano ciało funkcji *handleNodeTimer()* odpowiedzialnej za obsługę timera węzłów pomiarowych odliczającego czas ich aktywności.

Montaż i uruchomienie

Tyle w kwestiach implementacyjnych. Przejdźmy zatem do schematu montażowego węzła pomiarowego, który pokazano na **rysunku 3**. Zaprojektowano niewielki, dwustronny obwód drukowany (średnica jedynie 35 mm) z przewagą elementów SMD. Montaż węzła pomiarowego rozpoczynamy

Ustawienia Fuse-bitów węzła pomiarowego:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1

Ustawienia Fuse-bitów odbiornika:
CKSEL3...0: 1101
SUT1...0: 11
CKDIV8: 1
CKOUT: 1

Listing 4. Funkcja obsługi przerwania realizująca wysyłanie danych

```
//Przerwanie 9600 razy na sekundę
ISR(TIM0_COMPA_vect) {
    static uint8_t Bit, Byte;

    //Jeśli są jakieś dane do przesłania
    if(USART.Bytes) {
        //START BIT
        if(Bit == 0) {
            //0 na port TX - START BIT
            USART_TX_PORT &= ~(1<<USART_TX_NR);
            Bit++;
        //Dane
        } else if(Bit>0 && Bit<9) {
            if(USART.Data[Byte] & 0x01) USART_TX_PORT |= (1<<USART_TX_NR);
            else USART_TX_PORT &= ~(1<<USART_TX_NR);
            USART.Data[Byte] >>= 1;
            Bit++;
        //STOP BIT
        } else {
            //1 na port TX - STOP BIT
            USART_TX_PORT |= (1<<USART_TX_NR);
            //Zerujemy wskaźnik bitów,
            //bo będziemy przesyłać kolejny bajt danych
            Bit = 0;
            //Zmieniamy wskaźnik bajtów na kolejny bajt danych
            Byte++;
            //Sprawdzamy, czy wysłaliśmy już wszystkie bajty danych
            if(--USART.Bytes == 0) {
                Byte = 0;
                STOP_TRANSMISSION;
            }
        }
    }
}
```

Listing 5. Ciało funkcji main układu nadrzędnego

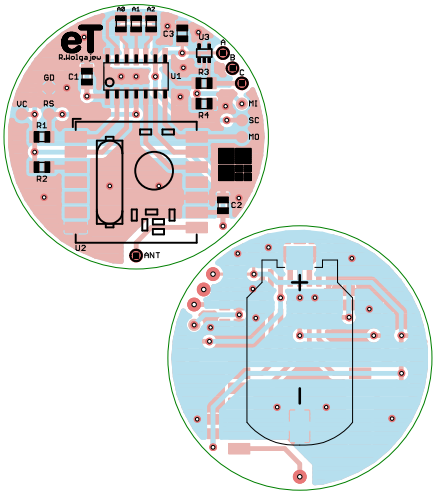
```
int main(void) {
    //Inicjalizacja OLED-a, ustawienie czcionki i pokazanie logo
    OLEDinit();
    OLEDdrawBitmap(40, 1, Logo);
    OLEDsetFont(&whiteRabbit35x48);
    _delay_ms(1000);
    OLEDcls();
    //Inicjalizacja Timera1 ustawiającego flagi co 1s i 2s
    initTimer();
    //Inicjalizacja software'owego USART-a
    initUsart();
    //Inicjalizacja RFM12B w trybie odbiornika
    RFM12bInit(MASTER_ID);

    sei();
    while(1) {
        //Obsługa nadchodzących ramek,
        //w tym wysyłanie danych USART-em
        handleNode();
        //Obsługa timera węzłów
        //odliczającego czas aktywności węzła
        handleNodeTimer();
        //Cykliczne wyświetlanie węzła sieci (co 2s)
        redrawScreen();
    }
}
```

Listing 6. Funkcja odpowiedzialna za obsługę nadchodzących ramek wysyłanych przez węzły pomiarowe oraz wysyłanie odebranych danych za pośrednictwem interfejsu USART

```
void handleNode(void) {
    //Odebrano poprawną ramkę danych: Size(=3)|Adres|Temp|Batt|CRC8
    if(dataReady) {
        dataReady = 0;

        Node[Address].lastTemp = Node[Address].currTemp;
        Node[Address].currTemp = Temp;
        Node[Address].Batt =  Batt;
        Node[Address].Timer = NODE_TIMEOUT;

        //Czekamy na zakończenie poprzedniej transmisji,
        //jeśli jeszcze się nie zakończyła
        while(USART.Bytes);
        //Przygotowanie danych
        /do wysłania danych USART-em (w ISR)
        USART.Data[0] = Address;  //Adres
        USART.Data[1] = Temp;  //Temperatura
        USART.Data[2] = Batt;  //Stan baterii
        USART.Bytes = 3;
        START_TRANSMISSION;
    }
}
```

Listing 7. Funkcja odpowiedzialna za obsługę timera węzłów pomiarowych odliczającego czas ich aktywności

```
void handleNodeTimer(void) {
    if(timer1sSet) {
        timer1sSet = 0;
        for(uint8_t i=0; i<8; ++i) if(Node[i].Timer) Node[i].Timer--;
    }
}
```



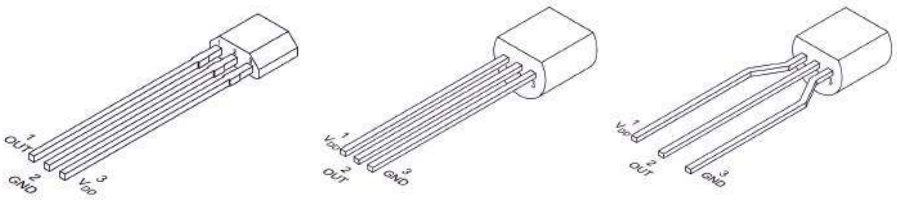
Rysunek 3. Schemat płytki PCB węzła pomiarowego

od przylutowania mikrokontrolera, następnie lutujemy czujnik LMT85, moduł RFM12B a na końcu elementy bierne. Dalej, przechodzimy na warstwę BOTTOM, gdzie przylutowujemy koszyczek baterii zasilającej. Do tak przygotowanej płytki przylutowujemy antenę nadawczą w postaci kawałka przewodu o długości około 17 cm (może być odpowiednio zwinięty). Na samym końcu wybieramy adres sprzętowy modułu, posługując się polami lutowniczymi A0…A2, które umożliwiają ustawienie 8 adresów z przedziału 0…7. Kropla cyny na wybranym polu lutowniczym oznacza stan logiczny 0 na danym pinie mikrokontrolera. Widok zmontowanego węzła pomiarowego od strony TOP pokazano na fotografii tytułowej.

Warto również podkreślić, że moduł węzła pomiarowego ma specjalne pola lutownicze oznaczone jako A…C, do których możemy przylutować przetwornik LMT85 w odbudowie THT typu TO-92 (zamiast standardowej SMD typu SC-70), przez co przetwornik taki możemy umieścić poza płytką węzła pomiarowego. Znaczenie poszczególnych pól lutowniczych jest następujące:

- A to napięcie zasilania,
- B to masa,
- C to wyjście czujnika.

Na **rysunku 4** pokazano wygląd obudów THT układu LMT85 z opisem wyprowadzeń. Co ważne i potwierdzone w praktyce, zdobycie układu LMT85 u dystrybutorów elementów elektronicznych może okazać się dość kłopotliwe, gdyż terminy dostaw sięgają 9...12 miesięcy. Sam znalazłem się w tej sytuacji i zastanawiałem się długo, jak podołać tym ograniczeniom. Rozwiązanie okazało się banalne. Układy LMT85 zamówiłem na stronie producenta (*ti.com*) taniej niż u dystrybutorów (poniżej

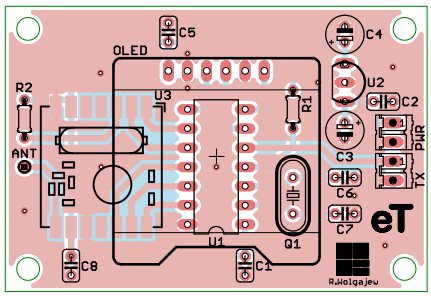


Rysunek 4. Wygląd obudów THT układu LMT85 z opisem wyprowadzeń

1 $ za sztukę) i niezbędne 2 sztuki miałem na biurku już po niespełna tygodniu, tak więc szczerze polecam to rozwiązanie.

Przejdźmy do montażu modułu nadrzędnego, którego obwód drukowany pokazano na **rysunku 5**. Tym razem zaprojektowano nieco większy obwód drukowany, który integruje w sobie kompletny zasilacz i zbudowany jest w zdecydowanej większości z elementów przewlekanych. Montaż obwodu drukowanego odbiornika rozpoczynamy od przylutowania modułu RFM12B, następnie lutujemy pozostałe elementy półprzewodnikowe, dalej elementy bierne oraz elementy mechaniczne.

Na samym końcu przylutowujemy wyświetlacz OLED, korzystając z wbudowanego złącza typu goldpin, które zapewnia mu zarówno niezbędne połączenie elektryczne, jak i montaż mechaniczny. Można również skorzystać z gniazda typu goldpin, przez co połączenie urządzenia z wyświetlaczem będzie łatwo rozłączalne. 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). Widok zmontowanego odbiornika od strony warstwy TOP z przylutowanym wyświetlaczem OLED pokazano na fotografii tytułowej.



Rysunek 5. Schemat płytki PCB modułu nadrzędnego

Rysunek 6. Graficzny interfejs użytkownika modułu nadrzędnego

Obsługa systemu

Kilka słów uwagi należy się obsłudze systemu. Jeśli chodzi o węzły pomiarowe, to ich obsługa ogranicza się w zasadzie do nadania adresu sprzętowego, o czym napisano już wcześniej. Z kolei w przypadku układu nadrzędnego jedyną czynnością, jaką należy wykonać, jest wyłącznie podłączenie źródła napięcia zasilania. Po wykonaniu tej czynności układ nadrzędny zaczyna oczekiwać na dane przesyłane z węzłów pomiarowych. Do czasu, gdy żaden węzeł pomiarowy nie przesłał danych, stosowna informacja pojawia się na wyświetlaczu w postaci napisu „No sensors!”. W chwili, gdy co najmniej jeden węzeł pomiarowy przesłał oczekiwane dane, zostaną one wyświetlone w ramach graficznego interfejsu użytkownika, który pokazano na **rysunku 6**. Jak widać, pokazywana jest temperatura ze znakiem (rozdzielczość 0,5°C), numer sprzętowy węzła pomiarowego (0…7), stan baterii zasilającej (graficznie – 10 poziomów) oraz tendencja zmian temperatury (strzałka symbolizująca, czy temperatura spada, czy też rośnie). Wskazania prezentowane są cyklicznie dla każdego aktywnego węzła przez 2 sekundy.

W przypadku, gdy węzeł pomiarowy przestanie wysyłać dane przez okres co najmniej 192 s (3 cykle pomiarowe), zostanie on automatycznie usunięty z listy aktywnych węzłów pomiarowych. Każdemu odbiorowi danych nadesłanych przez węzeł pomiarowy towarzyszy transmisja danych przez interfejs USART. Wysyłane są 3 bajty danych:

- adres węzła pomiarowego (0…7),
- temperatura (ze znakiem) węzła pomiarowego (–60...10, przy czym jednostką jest 0,5°C),
- napięcie baterii zasilającej węzła pomiarowego (0…10).

Robert Wołgajew, EP

[1] https://bit.ly/3Ol42zY