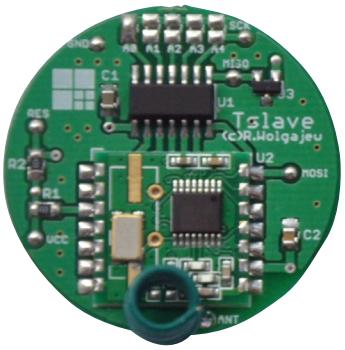


Autor składa podziękowania panu Sławomirowi Szwedzie z firmy Unisystem za dostarczenie wyświetlacza OLED znajdującego zastosowanie w niniejszym projekcie.

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

W ofercie AVT* AVT-5635

Projekty pokrewne na www.media.avt.pl:

AVT-5623	4-kanałowy termometr z interfejsem Wi-Fi (EP 4/2018
AVT-5566	THPStation – rozbudowany termometr z Wi-Fi (EP 1/2017
AVT-1941	8-kanałowy termometr z I²C (EP 1/2017
AVT-5573	Nieskomplikowany termometr-rejestrator (EP 11/2016
AVT-5535	Termometr 2-kanałowy z interfejsem Bluetooth (EP 4/2016
AVT-5518	Termometr bezprzewodowy (EP 11/2015
AVT-1863	Termometr z interfejsem Bluetooth (EP 8/2015
AVT-1790	Termometr XXL (EP 2/2014
AVT-5489	8-kanałowy termometr z alarmem i wyświetlaczem LCD (EP 11/2013
AVT-5420	Wielopunktowy termometr z rejestracją (EP 10/2013
AVT-1734	Termometr do wędzarni (EP 4/2013
AVT-5373	Tlogger – rejestrator temperatury (EP 12/2012
AVT-1705	Moduł do pomiaru temperatury z interfejsem RS485 (EP 9/2012
AVT-1697	Wielogabarytowy termometr LED (EP 8/2012
AVT-5389	4-kanałowy termometr z wyświetlaczem LED (EP 5/1012
AVT-5330	Termometr PC (EP 2/2012
AVT-5301	Wskaźnik komfortu cieplnego z wbudowanym kalendarzem sezonowym (EP 7/2011
AVT-1582	Domowy termometr RGB (EP 8/2010
AVT-5230	Rejestrator temperatury z interfejsem USB (EP 4/2010
AVT-5205	System pomiaru temperatury z termoparą typu K (EP 10/2009
AVT-5117	Termometr USB (EP 11/2007
AVT-5108	2-kanałowy termometr z dwukolorowym wyświetlaczem LED (EP 8/2007
AVT-957	Moduł pomiaru temperatury (EP 11/2006
AVT-2787	PC – Termometr – termometr internetowy (EdW 5/2006
AVT-918	Termometr z termoparami J albo K (EP 2/2006
AVT-5041	Termometr MIN-MAX (EP 11/2001

* Uwaga! Elektroniczne zestawy do samodzielnego montażu. Wymagana umiejętność lutowania!
Podstawową wersją zestawu jest wersja [B] nazywana potocznie KITem (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 dokumentacja

Kity w których występuje układ scalony wymagający zaprogramowania, posiadają 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 email: kity@avt.pl.

Bezprzewodowy, energooszczędny system pomiaru temperatury (2)

Stanąłem przed wyzwaniem skonstruowania prostego systemu bezprzewodowego pomiaru temperatury, który, po pierwsze, nie wymagałby żadnej konfiguracji, a po drugie, odznaczałby się walorem w postaci małego poboru energii przez węzły pomiarowe. Zależało mi też na użyciu łatwo dostępnych, tanich modułów radiowych mających tryb oszczędzania energii. Prezentowane urządzenie opracowałem w efekcie tak sformułowanych wymagań.

W pierwszej części artykułu opisano oprogramowanie systemu do pomiaru temperatury. W części drugiej zostanie zaprezentowany schemat ideowy oraz będzie opisana budowa urządzenia.

Kończąc opis oprogramowania, opiszę funkcję inicjującą odbieranie danych – pokazano ją na **listingu 9**. Ta funkcja, poza przygotowaniem zmiennych używanych w mechanizmie odbioru danych, uruchamia odbiornik transceivera RFM-12B, co powoduje wygenerowanie szeregu przerwań, w ramach obsługi których następuje faktyczne odebranie ramki danych. Jej odebraniu (z poprawną sumą CRC8) towarzyszy zmiana wartości globalnej zmiennej statusowej **RFM12B.Status** na predefiniowaną wartość **NEW_PACKET**. W tym momencie odebrane dane umieszczone zostają w globalnej zmiennej **RFM12B.Buffer**.

Pora na „wisienkę na torcie”, czyli funkcję obsługi przerwania zewnętrznego (od wyprowadzenia nIRQ modułu), która to jest „silnikiem” całego mechanizmu realizując rzeczywiste nadawanie lub odbieranie danych. Tę funkcję przedstawiono na **listingu 10**. Zaopatrzono ją w bardzo bogate komentarze, więc nie wymaga ona dodatkowego opisywania. Warto jednak zauważyć, że cały proces wysyłania czy odbierania danych odbywa się w tle nie wstrzymując działania pętli głównej aplikacji. Bieżący stan realizowanego procesu możemy każdorazowo sprawdzić testując wartość zmiennej globalnej **RFM12B.Status**.

Uff, to tyle, jeśli chodzi o nasze ciekawe peryferium, w związku z czym pora na przedstawienie szczegółów konstrukcyjnych

Listing 9. Funkcja inicjująca proces odbierania danych

```
void RFM12bStartRx(void)
{
    RFM12B.Status = RECEIVING;  //Zmiana statusu transceivera
    RFM12B.Idx = 0;  //Zerujemy indeks odbieranego znaku
    ATOMIC_BLOCK(ATOMIC_RESTORESTATE)
    {
        //Zresetowanie bufora FIFO (konieczne, by uruchomić mechanizm detekcji słowa synchro) i włączenie toru radiowego odbiornika
        SPIsendWord(POWER_MANGMNT_REG|ENABLE_RECEIVER|ENABLE_BASEBAND|ENABLE_CRYSTAL_OSC|DISABLE_CLOCK_OUTPUT);
        SPIsendWord(RX_FIFO_SETTINGS_REG|FIFO_DEPTH_8BITS|SYNCHRO_PATT_2DD4|FIFO_START_SYNCHRO|FIFO_FILL_DISABLE|RESET_NON_SENSITIVE);
        SPIsendWord(RX_FIFO_SETTINGS_REG|FIFO_DEPTH_8BITS|SYNCHRO_PATT_2DD4|FIFO_START_SYNCHRO|FIFO_FILL_ENABLE|RESET_NON_SENSITIVE);
    }
}
```

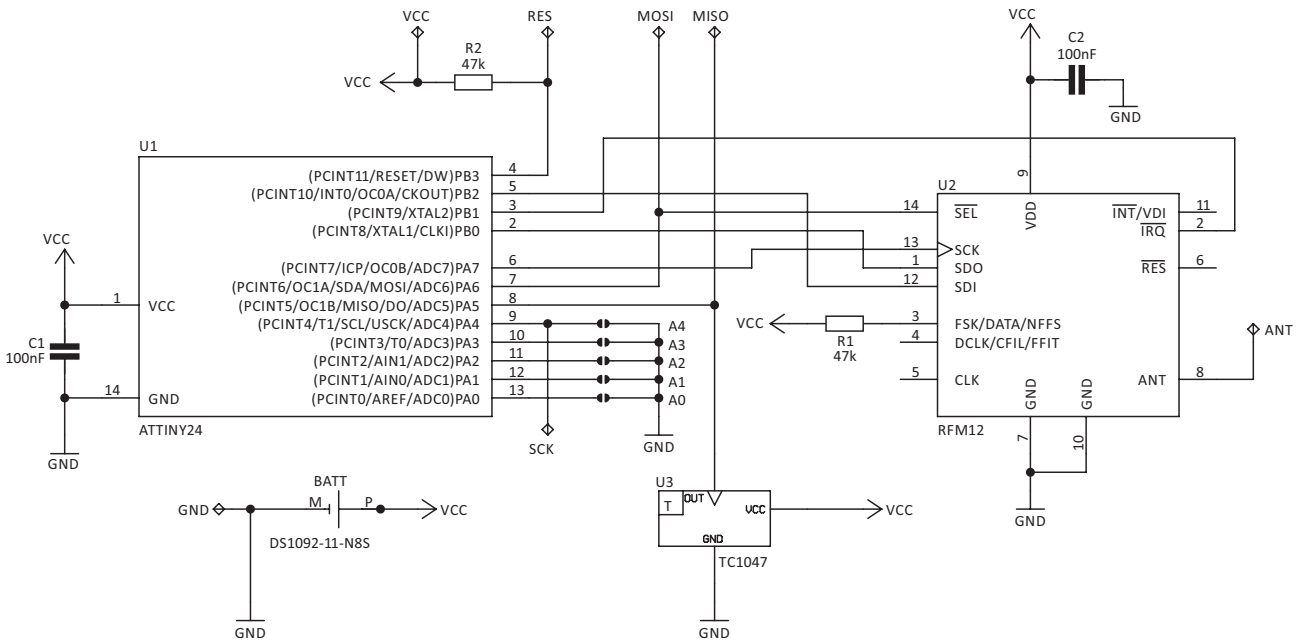
Listing 10. Funkcja obsługi przerwania zewnętrznego realizująca rzeczywiste nadawanie lub odbieranie danych

```
ISR(RFM12_IRQ_NAME)
{
    uint8_t frameSize, CRC8 = 0;
//Reagujemy tylko na zbocze opadające przerwania Pin Change Interrupt 1
    if(!(RFM12_IRQ_PIN & (1<<RFM12_IRQ_NR)))
    {
//Transceiver w trybie odbiornika
        if(RFM12B.Status == RECEIVING)
        {
//Odczyt odebranego bajta danych - sprawdzamy, przy okazji czy nie przekroczono maksymalnej długości ramki
//Jeśli długość przekroczono to odrzucamy taką ramkę danych (jako błędną) i wyłączamy tor radiowy odbiornika
            if(RFM12B.Idx < BUFFER_SIZE) RFM12B.Buffer[RFM12B.Idx++] = SPIsendWord(RX_FIFO_READ_REG) & 0xFF;
            else
            {
                SPIsendWord(POWER_MANGMNT_REG|ENABLE_CRYSTAL_OSC|DISABLE_CLOCK_OUTPUT);
                RFM12B.Status = FAULTY_PACKET;  //Zmiana statusu transceivera
            }

//Sprawdzamy, czy otrzymano już kompletną ramkę danych o długości Size (bajt 0. ramki) plus 2. Dodatkowe
//bajty, które muszą zostać odebrane to: rozmiar ramki oraz suma CRC8 (bajty postambuły: 0xAA pomijamy)
            frameSize = RFM12B.Buffer[0];  //Rozmiar przesłanej ramki danych (tylko dane użyteczne)
            if(RFM12B.Idx == frameSize+2)
            {
//Wyłączenie toru radiowego odbiornika
                SPIsendWord(POWER_MANGMNT_REG|ENABLE_CRYSTAL_OSC|DISABLE_CLOCK_OUTPUT);
//Obliczenie CRC8 z odebranych bajtów danych (w tym bajta Size - nr 0.) i porównanie z przesłanym CRC8 (bajt Size+1)
                for(uint8_t i=0; i<= frameSize; ++i) CRC8 = _crc_ibutton_update(CRC8, RFM12B.Buffer[i]);
//Zmiana statusu transceivera i sygnalizacja nadejścia nowej ramki danych w przypadku zgodnej sumy CRC8
                if(RFM12B.Buffer[frameSize+1] == CRC8) RFM12B.Status = NEW_PACKET; else RFM12B.Status = FAULTY_PACKET;
            }
        }
        else if(RFM12B.Status == BROADCASTING)  //Transceiver w trybie nadajnika
        {
//Wysyłamy bajt danych
            SPIsendWord(TX_WRITE_REG|RFM12B.Buffer[RFM12B.Idx]);
//Sprawdzamy, czy pozostało coś jeszcze do wysłania i jeśli nie to wyłączamy tor radiowy nadajnika
            if(RFM12B.Idx == 0)
            {
                SPIsendWord(POWER_MANGMNT_REG|ENABLE_CRYSTAL_OSC|DISABLE_CLOCK_OUTPUT);
                RFM12B.Status = PACKET_SENT;  //Zmiana statusu transceivera
            }
            else RFM12B.Idx--;
        }
    }
}
```

przedmiotu niniejszego artykułu, a mianowicie bezprzewodowego, energooszczędnego systemu wielopunktowego pomiaru temperatury, który w założeniach charakteryzować się ma następującymi cechami funkcjonalnymi:

- Obsługa do 32 adresowalnych węzłów pomiarowych (układów Slave).
- Zasilanie bateryjne i niski pobór mocy węzła pomiarowego.
- Pozostawanie w uśpieniu węzłów pomiarowych i cykliczne wybudzanie



Rysunek 6. Schemat ideowy węzła pomiarowego

Listing 11. Funkcja *main* węzła pomiarowego

```
int main(void)
{
    char Data[3];
    //Podciągnięcie portu adresu do VCC
    ADDRESS_PORT |= (1<<PA4)|(1<<PA3)|(1<<PA2)|(1<<PA1)|(1<<PA0);
    //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(869000, 8000, ADDRESS_PIN & 0b11111);
    sei();
    while(1)
    {
        set_sleep_mode(SLEEP_MODE_PWR_DOWN);
        sleep_enable();
        sleep_cpu();
        //W tym miejscu CPU oczekuje na wybudzenie przez Watchdog
        sleep_disable();
        //Wykonanie niezbędnych pomiarów
        Data[0] = ADDRESS_PIN & 0b11111;  //Adres sprzętowy naszego urządzenia
        Data[1] = ADCmeasure(ADC_CHANNEL_TEMP);  //Wykonanie pomiaru temperatury
        Data[2] = ADCmeasure(ADC_CHANNEL_VCC);  //Wykonanie pomiaru napięcia zasilania
        ADC_STOP_MEASURE;  //Wyłączenie ADC dla oszczędzania energii
        //Wychodzimy z trybu power-down modułu RFM12b
        RFM12bPowerUp();
        //Inicjujemy wysłanie ramki danych
        RFM12bStartTx(Data, 3, MASTER_ID);
        //Czekamy na zakończenie transmisji
        while(RFM12B.Status != PACKET_SENT);
        //Wprowadzamy moduł RFM12B w tryb power-down
        RFM12bPowerDown();
    }
}
```

się, któremu towarzyszy przesyłanie wartości mierzonej temperatury oraz stanu baterii zasilającej.

- Efektowny, graficzny interfejs użytkownika po stronie układu nadrzędnego (układu Master).
- Wysoka ergonomia obsługi całego systemu i brak konieczności konfiguracji.
- Kontrola aktywności węzłów pomiarowych przez układ nadrzędny.

Zacznijmy zatem od układu podrzędnego, czyli węzła pomiarowego, którego schemat pokazano na **rysunku 6**. Jego „sercem” jest niewielki mikrokontroler ATtiny24 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 wykorzystaniu przetwornika A/C wbudowanego w mikrokontroler, możliwy stał się pomiar temperatury przetwornika temperatura/napięcie pod postacią układu TC1047 oraz pomiar napięcia baterii zasilającej.

Na pierwszy „rzut oka” nie wydaje się, aby mikrokontroler w jakikolwiek sposób używał przetwornika A/C do pomiaru napięcia baterii zasilającej, ponieważ żadne z jego wejść zewnętrznych nie jest używane w tym celu. To prawda. Patrząc na schemat układu i nie mając do dyspozycji listingu programu można by wysnuć taki wniosek, jednak przetwornik A/C mierzy w takim przypadku specjalne, wewnętrzne napięcie odniesienia V_{BG}=1,1 V, dzięki temu, iż 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.

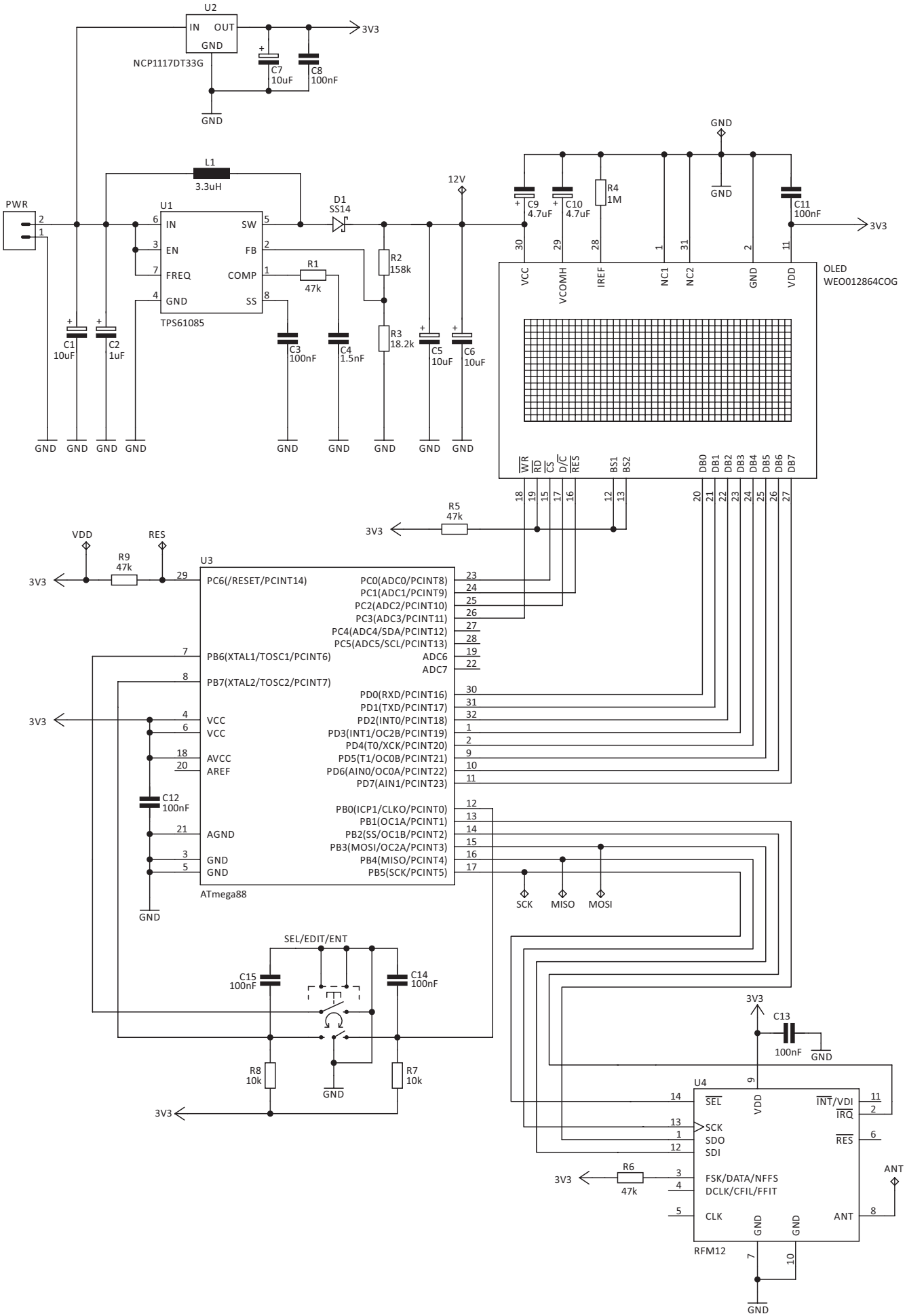
Listing 12. Plik nagłówkowy obsługi wyświetlacza OLED ze sterownikiem SSD1309

```
//Ustawienia konfiguracji połączeń
#define DATA_PORT PORTD
#define DATA_DDR DDRD
#define CONTROL_PORT PORTC
#define CONTROL_DDR DDRC
#define WR_NR PC3  //Write (0)
#define CS_NR PC0  //Chip select (0)
#define DC_NR PC2  //Data (1)/Command (0)
#define RST_NR PC1  //Reset (0)
#define RESET_WR CONTROL_PORT &= ~(1<<WR_NR)
#define SET_WR CONTROL_PORT |= (1<<WR_NR)
#define RESET_CS CONTROL_PORT &= ~(1<<CS_NR)
#define SET_CS CONTROL_PORT |= (1<<CS_NR)
#define RESET_DC CONTROL_PORT &= ~(1<<DC_NR)
#define SET_DC CONTROL_PORT |= (1<<DC_NR)
#define RESET_RST CONTROL_PORT &= ~(1<<RST_NR)
#define SET_RST CONTROL_PORT |= (1<<RST_NR)
//Definicje rozkazów sterujących sterownika SSD1309
#define SET_CONTRAST_CTRL 0x81
#define OUTPUT_FOLLOWS_RAM 0xA4
#define OUTPUT_IGNORES_RAM 0xA5
#define SET_NORMAL_DISPLAY 0xA6
#define SET_INVERSE_DISPLAY 0xA7
#define SET_DISPLAY_ON 0xAF
#define SET_DISPLAY_OFF 0xAE
#define SET_MEM_ADDR_MODE 0x20
#define HORIZONTAL_ADDRESSING 0x00
#define VERTICAL_ADDRESSING 0x01
#define PAGE_ADDRESSING 0x02
#define SET_COLUMN_ADDRESS 0x21
#define SET_PAGE_ADDRESS 0x22
#define SET_DISPLAY_START_LINE 0x40
#define SET_SEGMENT_REMAP_NORMAL 0xA0
#define SET_SEGMENT_REMAP_INVERSED 0xA1
#define SET_MULTIPLEX_RATIO 0xA8
#define MASTER_CONFIGURATION 0xAD
#define SET_COM_SCAN_DIR_NORMAL 0xC0
#define SET_COM_SCAN_DIR_INVERSED 0xC8
#define SET_DISPLAY_OFFSET 0xD3
#define SET_COM_PINS_HARDWARE_CONFIG 0xDA
#define SET_DISPLAY_CLOCK 0xD5
#define SET_PRECHARGE_PERIOD 0xD9
#define SET_VCOMH_DESELECT_LVL 0xDB
```

Skoro mierzone napięcie ma wartość stałą (1,1 V), zaś napięcie odniesienia wartość zmienną, łatwo możemy ustalić jego wartość. Dzięki temu „trickowi” stał się możliwy pomiar napięcia baterii zasilającej bez angażowania dodatkowego pinu mikrokontrolera, którego de facto już nie mamy. Dodatkowo, mikrokontroler obsługuje 5 wyprowadzeń adresowych A0…A4, pozwalających na ustawienie adresu sprzętowego węzła pomiarowego.

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

- Wyłączono wszystkie nieużywane peryferia mikrokontrolera (komparator analogowy, TIMER1, TIMER0, USI).
- Wprowadzono mikrokontroler w tryb obniżonego poboru energii Power-down, z którego jest wybudzany cyklicznie co 8 sekund przez odpowiednio skonfigurowany układ Watchdog, którego zadziałanie nie powoduje zresetowania mikrokontrolera, tylko wywołanie stosownego



Rysunek 7. Schemat ideowy układu nadrzędnego systemu pomiarowego

przerwania systemowego mającego możliwość wybudzania mikrokontrolera. Wybudzony mikrokontroler inicjuje transmisję danych, po czym przechodzi ponownie w tryb Power-down.

- Nieużywany transceiver RFM-12B wprowadzany jest każdorazowo w tryb obniżonego poboru energii.

Dzięki opisanym mechanizmom zapotrzebowanie na prąd zmniejszono do około 40 μA w trybie obniżonego poboru energii oraz 20 mA w czasie transmisji danych trwającej 10 ms (przy pełnej mocy nadawania). Osiągnięte wyniki zapewniają zadowalający czas pracy węzła pomiarowego na jednaj baterii zasilającej.

Na **listingu 11** pokazano funkcję **main** programu obsługi węzła pomiarowego, realizującą całą, założoną funkcjonalność. Transceiver pracując wyłącznie w trybie nadajnika wysyła każdorazowo 3 bajty danych użytecznych: swój adres sprzętowy, wartość zmierzonej temperatury zewnętrznej 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 zbudowanym systemie pomiarowym. Oczywiście, wysyłanie mierzonej temperatury co 8 sekund może okazać się zbyt częste jak i niepotrzebne w kontekście „drenowania" zasobów baterii zasilającej, lecz dzięki nieskomplikowanej korekcie możemy ten czas łatwo modyfikować.

Tyle w kwestii konstrukcji oprogramowania węzła pomiarowego, a zatem czas na przedstawienie schematu ideowego jednostki nadrzędnej, której zadaniem jest obsługa węzłów pomiarowych i wizualizacja przesyłanych danych. Schemat ideowy układu nadrzędnego pokazano na **rysunku 7**.

Tym razem do czynienia mamy z nieco bardziej rozbudowanym systemem mikroprocesorowym, którego „sercem" jest popularny mikrokontroler ATmega88 sterujący pracą modułu transceivera dzięki realizacji programowej obsługi interfejsu SPI oraz obsłudze przerwania zewnętrznego **Pin Change Interrupt 0** (wyprowadzenie PCINT2) odpowiedzialnego za mechanizm odbierania danych. Ponadto, mikrokontroler zajmuje się też obsługą enkodera obrotowego ze zintegrowanym przyciskiem, stanowiącego element interfejsu użytkownika oraz, co ważniejsze, realizuje obsługę doskonałej jakości wyświetlacza graficznego OLED o przekątnej 2,7" o grubości zaledwie 2 mm (!) i rozdzielczości 128×64 piksele, wyposażonego w sterownik ekranu SSD1309. Ten element potrzebuje do działania napięcia 12 V, więc wykonano przetwornicę step-up z użyciem układu TPS61085. Warto również podkreślić, iż nie bez powodu wybrano ten rodzaj wyświetlacza – jego właściwości użytkowe w porównaniu z typowym elementem LCD są nie do przecenienia. Co więcej, cena tego modułu jest zbliżona do ceny analogicznego elementu wykonanego w technologii LCD, co nie pozostawia nam właściwie wyboru.

W związku z powyższym oraz z uwagi na fakt, iż wyświetlacz z tego rodzaju sterownikiem nie był dotychczas przedmiotem żadnego opracowania, postanowiłem przedstawić podstawowe funkcje obsługi tego peryferium. Zacznijmy standardowo od przedstawienia krótkiego pliku nagłówkowego, w którym zdefiniowano adresy rejestrów sterujących, a który to pokazano na **listingu 12**. Aby jednak zacząć współpracę z tym arcyciekawym wyświetlaczem niezbędne są dwie podstawowe funkcje narzędziowe, które pozwolą na wysłanie do panelu OLED rozkazów sterujących, dzięki którym jest możliwa konfiguracja sterownika SSD1309 lub też danych obrazu. Interpretacja rodzaju danych, które mają zostać odebrane przez sterownik ekranu jest zdeterminowana stanem wyprowadzenia DC. Jest to typowe rozwiązanie w sterowaniu wyświetlaczami wszelkiego typu. Poziom niski na tym wyprowadzeniu decyduje o tym, iż przesyłana wartość zostanie zinterpretowana jako komenda sterująca zaś poziom wysoki spowoduje, iż dana zinterpretowana zostanie jako dana pamięci ekranu. Wspomniane, podstawowe funkcje narzędziowe zamieszczono na **listingu 13**.

Dysponując podstawowymi funkcjami narzędziowymi z list. 13 możemy przystąpić

Listing 13. Podstawowe funkcje narzędziowe odpowiedzialne za wysyłanie rozkazów sterujących lub danych obrazu

```
void writeCmnd(uint8_t Command)
{
    RESET_CS;
    RESET_DC;
    RESET_WR;
    DATA_PORT = Command;
    SET_WR;
    SET_DC;
    SET_CS;
}

void writeData(uint8_t Data)
{
    RESET_CS;
    RESET_WR;
    DATA_PORT = Data;
    SET_WR;
    SET_CS;
}
```

Listing 14. Funkcja inicjująca sterownik ekranu SSD1309 wyświetlacza OLED z rodziny WEO012864

```
void OLEDinit(void)
{
    //Port danych OLEDa jako wyjscia ze stanem "0"
    DATA_DDR = 0xFF;
    //Porty sterujące OLEDa jako wyjścia ze stanem "1"
    CONTROL_PORT |= (1<<WR_NR)|(1<<CS_NR)|(1<<DC_NR)|(1<<RST_NR);
    CONTROL_DDR |= (1<<WR_NR)|(1<<CS_NR)|(1<<DC_NR)|(1<<RST_NR);
    //Zerowanie sprzętowe sterownika SSD1309
    RESET_RST;
    _delay_ms(1);
    SET_RST;
    _delay_ms(1);
    //Konfiguracja wszystkich parametrów sprzętowych sterownika SSD1309
    writeCmnd(SET_DISPLAY_OFF);
    writeCmnd(SET_MEM_ADDR_MODE);
    writeCmnd(HORIZONTAL_ADDRESSING);
    writeCmnd(SET_CONTRAST_CTRL);
    writeCmnd(0x80);
    writeCmnd(SET_DISPLAY_START_LINE|0);  //Start Line = 0
    writeCmnd(SET_SEGMENT_REMAP_INVERSED);
    writeCmnd(OUTPUT_FOLLOWS_RAM);  //Entire Display On
    writeCmnd(SET_NORMAL_DISPLAY);
    writeCmnd(SET_MULTIPLEX_RATIO);
    writeCmnd(0x3F);  //1/64 Duty
    writeCmnd(MASTER_CONFIGURATION);
    writeCmnd(0x8E);  //Select external VCC supply
    writeCmnd(SET_COM_SCAN_DIR_INVERSED);
    writeCmnd(SET_DISPLAY_OFFSET);//Set Display Offset
    writeCmnd(0x00); //Default: 0
    writeCmnd(SET_DISPLAY_CLOCK);
    writeCmnd(0xF0); //ew. 0x00: 105HZ
    writeCmnd(SET_PRECHARGE_PERIOD);//Set Pre-charge Period
    writeCmnd(0xF1);  //ewentualnie FF
    writeCmnd(SET_COM_PINS_HARDWARE_CONFIG);
    writeCmnd(0x12);  //Alternative COM Pin
    writeCmnd(SET_VCOMH_DESELECT_LVL);
    writeCmnd(0x40);  //ew. 0xFF: ~0.84xVCC
    OLEDclearArea(0, 0, 127, 7);  //CLS
    writeCmnd(SET_DISPLAY_ON);
}
```

Listing 15. Funkcja narzędziowa odpowiedzialna za ustawienie aktywnego obszaru ekranu, w ramach którego przeprowadzany jest zapis do pamięci ekranu sterownika SSD1309

```
//Column: 0...127, Page: 0...7
void OLEDsetActiveWindow(uint8_t startColumn, uint8_t startPage, uint8_t endColumn, uint8_t endPage)
{
    writeCmnd(SET_COLUMN_ADDRESS);
    writeCmnd(startColumn);
    writeCmnd(endColumn);
    writeCmnd(SET_PAGE_ADDRESS);
    writeCmnd(startPage);
    writeCmnd(endPage);
}
```

Listing 16. Funkcja narzędziowa odpowiedzialna za wyświetlenie obrazka

```
void OLEDdrawBitmap(uint8_t Column, uint8_t Page, const uint8_t *Bitmap)
{
    register uint8_t Width, Height;
    register uint16_t bytesToSend;
    Width = pgm_read_byte(Bitmap++);  //Pierwszy bajt tablicy Bitmap to szerokość: 0...127
    Height = pgm_read_byte(Bitmap++)>>3;  //Drugi bajt tablicy Bitmap to wysokość: 8, 16, 24...64 -> przeliczamy na bajty
    bytesToSend = Width*Height;  //Liczba bajtów przeznaczonych do wysłania do OLEDa
    OLEDsetActiveWindow(Column, Page, Column+Width-1, Page+Height-1);
    while(bytesToSend--) writeData(pgm_read_byte(Bitmap++));
}
```

Listing 17. Definicja nowego typu danych odpowiedzialnego za przechowywanie parametrów bieżącej czcionki ekranowej

```
typedef struct
{
    uint8_t Width;  //Rzeczywista szerokosc znaku (px)
    uint8_t Height;  //Rzeczywista wysokosc znaku (bajty)
    uint8_t Interspace;  //Odstęp pomiędzy znakami (px)
    uint8_t BytesPerChar;  //Liczba bajtów danych tablicy wzorców na 1 znak
    uint8_t FirstCharCode;  //Kod ASCII pierwszego znaku
    const uint8_t *Bitmap;  //Wskaźnik to tablicy zawierającej wzorce poszczególnych znaków
} fontDescription;
```

Listing 18. Funkcja odpowiedzialna za ustawienie bieżącej czcionki ekranowe

```
void setFont(const fontDescription *Font)
{
    CurrentFont.Width = pgm_read_byte(&Font->Width);  //Rzeczywista szerokość czcionki
    CurrentFont.Height = pgm_read_byte(&Font->Height);  //Rzeczywista wysokość czcionki
    CurrentFont.Interspace = pgm_read_byte(&Font->Interspace);  //Odstęp pomiędzy znakami
    CurrentFont.BytesPerChar = pgm_read_byte(&Font->BytesPerChar);  //Liczba bajtów na definicje pojedyn. znaku
    CurrentFont.FirstCharCode = pgm_read_byte(&Font->FirstCharCode);  //Kod ASCII definicji pierwszego znaku
    CurrentFont.Bitmap = (uint8_t*)pgm_read_word(&Font->Bitmap);  //Wskaźnik do tablicy wzorców tej czcionki
}
```

Listing 19. Funkcja odpowiedzialna za rysowanie znaków, przy użyciu bieżącej czcionki ekranowej

```
void OLEDdrawChar(char Character, uint8_t Column, uint8_t Page, uint8_t Inverted)
{
    register uint8_t readByte;
    const uint8_t *dataPointer;
    register uint8_t bytesToSend;
    //Ustalamy adres początku wzorca znaku ASCII, który zamierzamy wyświetlić
    dataPointer = &CurrentFont.Bitmap[(CurrentFont.BytesPerChar*(Character-CurrentFont.FirstCharCode))];
    //Określamy okno zapisu by uprościć samą procedurę zapisu
    OLEDsetActiveWindow(Column, Page, Column+CurrentFont.Width-1, Page+CurrentFont.Height-1);
    //Określamy liczbę bajtów do wysłania
    bytesToSend = CurrentFont.BytesPerChar;
    while(bytesToSend--)
    {
        readByte = Inverted? ~pgm_read_byte(dataPointer++):pgm_read_byte(dataPointer++);
        writeData(readByte);
    }
}
```

Wykaz elementów:

Węzeł pomiarowy

Rezystory:
R1, R2: 47 kΩ (SMD 0805)

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

Półprzewodniki:
U1: ATtiny24 (SOIC14)
U3: TC1047A (SOT23)

Inne:
U2: RFM12B-866MHz (SMD)
BATT: koszyczek baterii CR2032 typu CONNFLY DS1092-11-N8S

Moduł nadrzędny

Rezystory: (SMD 0805)
R1, R5, R6, R9: 47 kΩ
R2: 158 kΩ/1%
R3: 18,2 kΩ/1%
R4: 1 MΩ
R7, R8: 10 kΩ

Kondensatory: (SMD 0805)
C1, C5…C7: 10 µF/16 V (SMD „A”/3216-18R)
C2: 1 µF/16 V (SMD „A”/3216-18R)
C3, C8, C11…C15: 100 nF
C4: 1,5 nF
C9, C10 : 4,7 µF/20 V (SMD „A”/3216-18R)

Półprzewodniki:
U1: TPS61085 (TSSOP8)
U2: NCP1117DT33G (TO252)
U3: ATmega88 (TQFP32)
D1: SS14 (SMA)

Inne:
U4: RFM12B-866 MHz (SMD)
L1 – dławik mocy 3,3 µH typu DLG-0504-3R3
SELECT/ENTER – enkoder ze zintegrowanym przyciskiem
PWR – gniazdo męskie kątowe 90˚ 2pin (NSL25-2W)
ZIF – złącze typu ZIF do montażu powierzchniowego (raster 0,5 mm, 31-pin, górny kontakt)
OLED – wyświetlacz OLED Winstar WEO012864KSPP3N00000 (128×64 px)

do inicjalizacji naszego wyświetlacza, ponieważ jak każde peryferium tego typu wymaga on ustawienia szeregu rejestrów konfiguracyjnych sterownika ekranu, których to wartości zależne są od właściwości wbudowanego panelu OLED, jak i specyfikacji producenta modułu. Stosowną funkcję inicjalizacyjną pokazano na **listingu 14**. Brak przeprowadzenia stosownej inicjalizacji w zasadzie uniemożliwia poprawne funkcjonowanie modułu OLED, ponieważ domyślne ustawienia rejestrów sterujących układu SSD1309 zwykle odbiegają od tych, wymaganych przez producenta panelu. Dalej, na **listingu 15** pokazano kolejną funkcję narzędziową odpowiedzialną za ustawienie aktywnego obszaru ekranu, w ramach którego przeprowadzany jest zapis do pamięci ekranu sterownika SSD1309. Jest ona bardzo użyteczna, gdyż jej użycie upraszcza a zarazem znacznie przyśpiesza wyświetlanie obrazów, których rozmiar jest inny aniżeli całkowity, fizyczny rozmiar ekranu. Warto w tym miejscu zauważyć, iż jeden bajt pamięci obrazu odpowiada ośmiu pikselom obrazu ułożonym w pionie, stąd całą wysokość obrazu równą 64 piksele podzielono na 8 tzw. stron (Pages).

Dalej, na **listingu 16** pokazano funkcję umożliwiającą wyświetlenie obrazka, którego treść znajduje się w tablicy, będącej argumentem wywołania, przy czym dwa pierwsze bajty tej tablicy określają szerokość i wysokość obrazka, zaś pozostałe bajty przechowują treść obrazka.

W tej chwili przyszedł czas na obsługę ostatniego elementu interfejsów graficznych, czcionek ekranowych! Aby jednak umożliwić wygodną obsługę wielu czcionek ekranowych, konieczne było wprowadzenie nowego typu danych, którego definicję pokazano na **listingu 17**. Bazując na zdefiniowanej strukturze wykonano funkcję, która korzystając z globalnej zmiennej

```
Ustawienia fusebitów modułu nadrzędnego:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1
EESAVE: 0
Ustawienia fusebitów węzła pomiarowego:
CKSEL3...0: 0010
SUT1...0: 10
CKDIV8: 0
CKOUT: 1
DWEN: 1
```

Rysunek 8. Przykładowy wygląd interfejsu użytkownika

static *fontDescription* CurrentFont pozwala na ustawienie bieżącej czcionki ekranowej – **listing 18**. Funkcję umożliwiającą rysowanie znaków przy użyciu bieżącej czcionki ekranowej zamieszczono na **listingu 19.** Jak zwykle, do wygenerowania plików zawierających wzorce czcionek, oparte na czcionkach systemu Windows, polecam doskonały program **PixeLab** autorstwa **Marcina Popławskiego**, którego szczegółowy opis znalazł się w Elektronice Praktycznej 06/2015. Tyle w kwestii obsługi naszego ciekawego wyświetlacza graficznego, którego zastosowanie, mam nadzieję, znacznie uatrakcyjni prezentowane urządzenie.

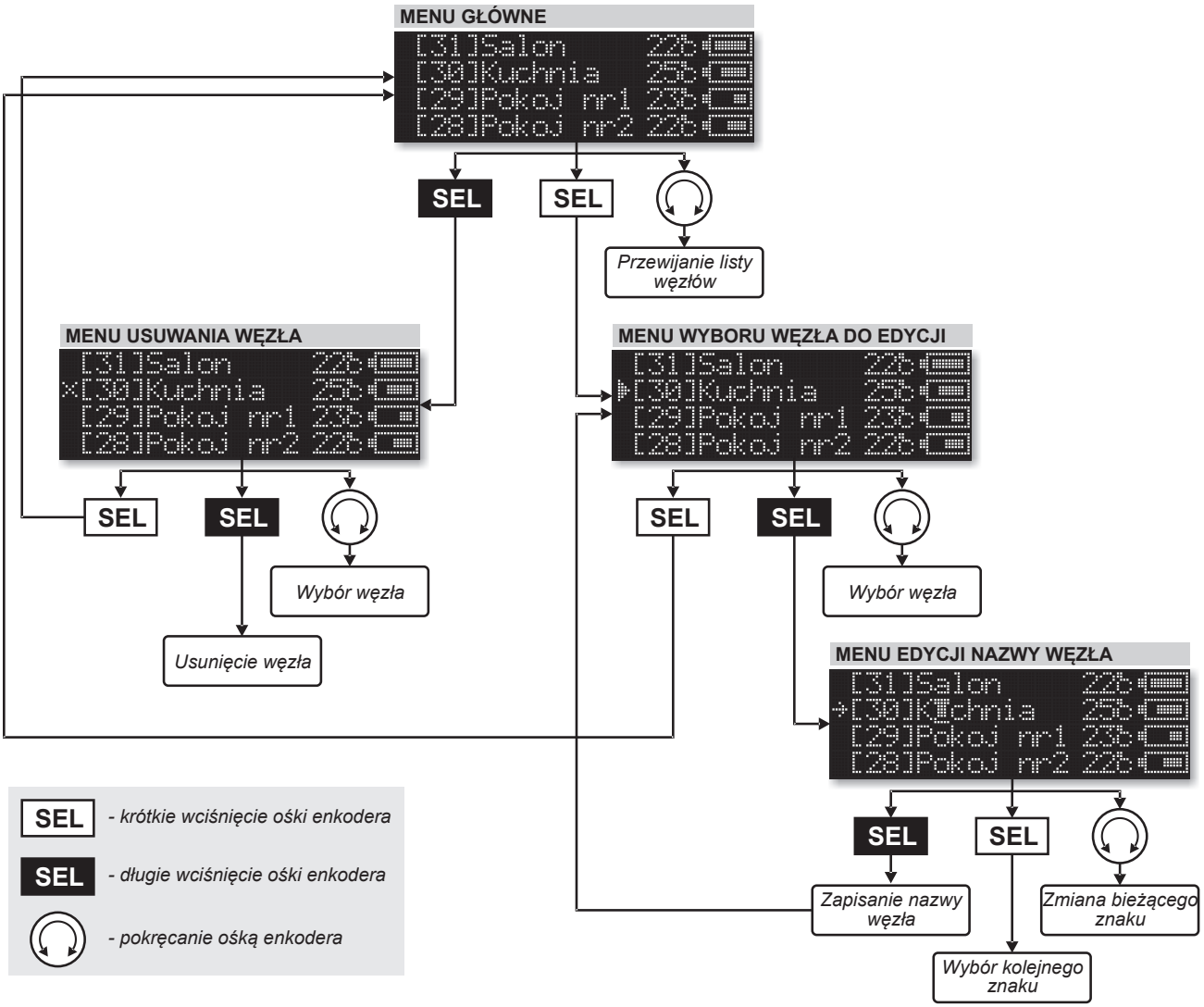
Obsługa

Projektując system bezprzewodowego pomiaru temperatury przyjąłem jako główne założenie maksymalną ergonomię obsługi, jak i brak potrzeby konfiguracji. W związku z powyższym, wszystko, co musimy zrobić, to nadać każdemu węzłowi sieci niepowtarzalny adres sprzętowy, posługując się zworkami na polach lutowniczych A0…A4, po czym włączyć go umieszczając baterię zasilającą. Po krótkiej chwili moduł taki zostanie wyświetlony na liście modułów w ramach graficznego interfejsu użytkownika modułu nadrzędnego i jeśli nie jest jeszcze znany (ma nowy, nieznany adres sprzętowy) zostanie mu nadana domyślna nazwa „No name", którą następnie możemy poddać edycji. Przykładowy wygląd interfejsu użytkownika pokazano na **rysunku 8**. Na pierwszym miejscu jest pokazywany adres sprzętowy każdego węzła sieci, jego nazwa nadana przez użytkownika, następnie zmierzona temperatura (lub symbol '—', w wypadku, gdy węzeł sieci przestanie wysyłać dane przez czas dłuższy niż 1 minuta) oraz stan baterii zasilającej w formie symbolu graficznego.

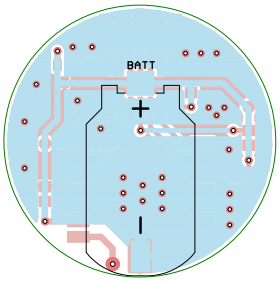
W ramach graficznego interfejsu użytkownika jednorazowo można wyświetlić dane 8 węzłów sieci. Ponadto, korzystając z enkodera z wbudowanym przyciskiem, możliwe jest przesuwanie listy węzłów (gdy jest ich więcej, niż 8), edycja ich nazw lub usuwanie węzłów. A wszystkie te czynności w wygodny sposób wykonujemy wyłącznie przy użyciu wspomnianego enkodera obrotowego. Wygląd wszystkich ekranów Menu wraz z funkcjonalnością enkodera obrotowego (w tym wbudowanego przycisku) pokazano na **rysunku 9.**

Montaż

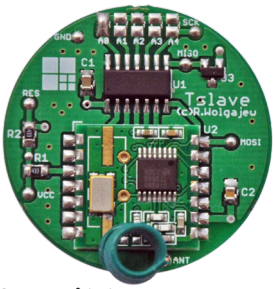
Opis montażu systemu rozpoczniemy od montażu obwodu drukowanego węzła pomiarowego, którego schemat montażowy pokazano na **rysunku 10**. Montaż rozpoczynamy od przylutowania mikrokontrolera, następnie lutujemy czujnik TC1047, moduł



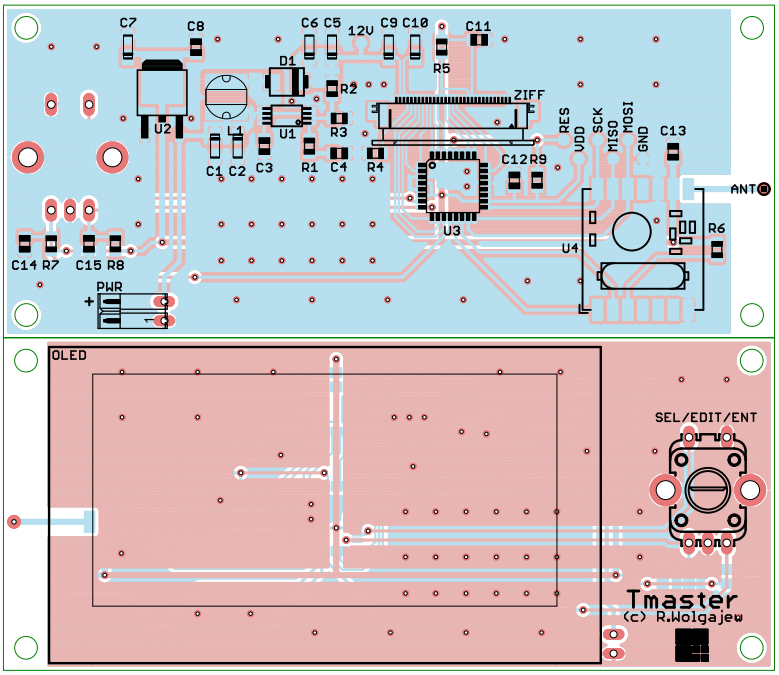
Rysunek 9. Wygląd i sposób obsługi Menu systemu pomiaru temperatury



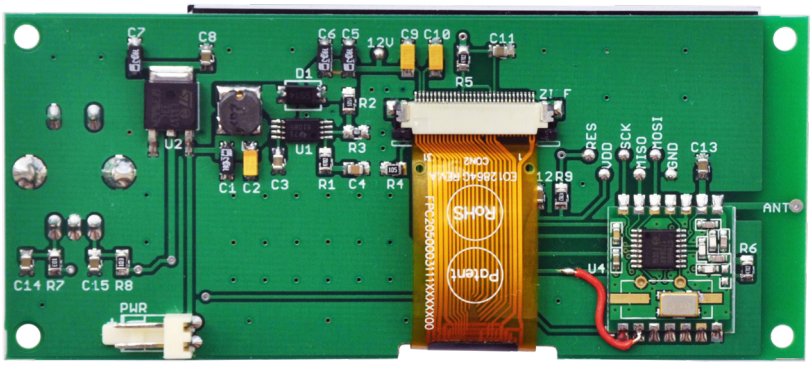
Rysunek 10. Schemat montażowy węzła pomiarowego



Fotografia 11. Widok zmontowanego węzła pomiarowego od strony TOP



Rysunek 12. Schemat montażowy modułu nadrzędnego



Fotografia 13. Wygląd zmontowanego obwodu drukowanego układu nadrzędnego od strony BOTTOM

RFM-12B 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 164 mm. Widok zmontowanego węzła pomiarowego od strony TOP pokazano na **fotografii 11**.

W tym momencie przejdźmy do montażu modułu nadrzędnego. Jego schemat montażowy pokazano na **rysunku 12**. Z uwagi na fakt, iż moduł wyświetlacza OLED jest dołączony do płytki naszego urządzenia z użyciem gniazda ZIF o bardzo gęstym rastrze wyprowadzeń (0,5 mm), montaż modułu nadrzędnego rozpoczynamy właśnie od przylutowania wspomnianego gniazda. Najprostszym sposobem montażu elementów o tak dużym zagęszczeniu wyprowadzeń, niewymagającym jednocześnie posiadania specjalistycznego sprzętu, jest użycie typowej stacji lutowniczej, dobrej jakości cyny z odpowiednią ilością topnika oraz dość cienkiej plecionki rozlutowniczej, która umożliwi usunięcie nadmiaru cyny spomiędzy wyprowadzeń złącza. Należy przy tym uważać by nie uszkodzić termicznie tegoż elementu. Jakość tak wykonanego połączenia sprawdzamy pod lupą korzystając z najprostszego miernika pozwalającego sprawdzić ciągłość połączeń. Wspomniana kontrola będzie znacznie łatwiejsza, jeśli zmontowaną płytkę przemyjemy alkoholem izopropylowym w celu wypłukania nadmiaru kalafonii lutowniczej. Następnie lutujemy mikrokontroler, moduł RFM-12B, pozostałe półprzewodniki a na samym końcu elementy bierne oraz złącze zasilające. Dalej, przechodzimy na warstwę TOP, gdzie przylutowujemy nieliczne elementy bierne oraz enkoder obrotowy. Na samym końcu podłączamy wyświetlacz OLED do złącza ZIFF po stronie BOTTOM, zaś sam element przyklejamy po stronie TOP (w miejscu wyznaczonym obrysem) korzystając z dwustronnej taśmy klejącej.

Podobnie, jak poprzednio, do tak przygotowanej płytki przylutowujemy antenę nadawczą w postaci kawałka przewodu o długości 164 mm. Wygląd zmontowanego obwodu drukowanego układu nadrzędnego od strony BOTTOM pokazano na **fotografii 13**.

Robert Wołgajew, EP