Jerzy JAKUBIEC, Roman ŻURKOWSKI
INSTYTUT METROLOGII, ELEKTRONIKI I AUTOMATYKI POLITECHNIKI ŚLĄSKIEJ
ul. Akademicka 10, 44-100 Gliwice

Wielozadaniowy system operacyjny komunikatora systemu diagnostyki cieplnej budynków

Prof. dr hab. inż. Jerzy JAKUBIEC

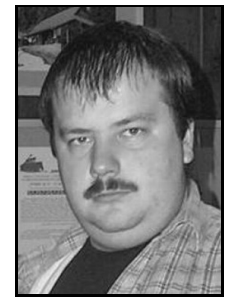
Pracownik Instytutu Metrologii, Elektroniki i Automatyki Politechniki Śląskiej. Studia ukończył na Wydziale Elektrycznym Politechniki Śląskiej w 1971 r., w 1978 uzyskał stopień doktora, w 1989 – doktora habilitowanego, a w 2004 – tytuł profesora. Zainteresowania naukowe: analiza propagacji błędów w systemach pomiarowo-sterujących, synteza modeli niepewności algorytmów przetwarzania danych pomiarowych, modelowanie metrologicznych właściwości torów przetwarzania analogowo-cyfrowego.

e-mail: jerzy.jakubiec@polsl.pl

Mgr inż. Roman ŻURKOWSKI

Absolwent Wydziału Elektrycznego Politechniki Śląskiej w Gliwicach – specjalność Automatyka i metrologia elektryczna (1999). Zajmuje się zagadnieniami budowy i analizy metrologicznej komputerowych systemów pomiarowych.

e-mail: roman.zurkowski@polsl.pl



Streszczenie

W artykule scharakteryzowano system operacyjny opracowany w celu uzyskania współbieżnej realizacji zadań przez mikrokontroler zarządzający pracą komunikatora będącego podstawowym urządzeniem systemu pomiarowego zbudowanego do prowadzenia diagnostyki cieplnej budynków. System operacyjny działa na zasadzie podziału czasu i nadzoruje wykonywanie zadań w liczbie równej liczbie portów szeregowych mikrokontrolera, przy czym zadania aktualnie wykonywane są wybierane z ogólnej puli zadań w trakcie konfiguracji systemu diagnostyki. Komunikator realizuje zadania związane obsługą zarządzanych przez niego przyrządów pomiarowych oraz komunikacją w sieci bezprzewodowej w standardzie ZigBee.

Słowa kluczowe: system pomiarowy do diagnostyki cieplnej budynków, system operacyjny mikrokontrolera, praca z podziałem czasu.

The multitasking operating system for a communicator in a building thermal diagnostic system

Abstract

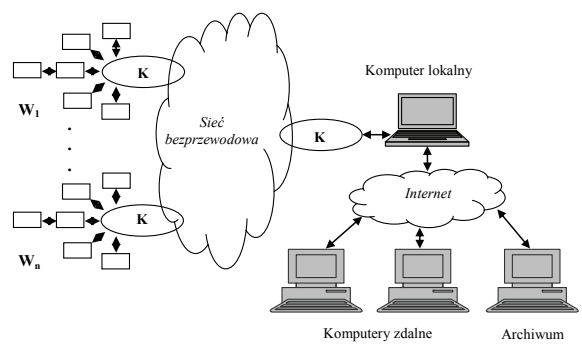
A communicator (Fig. 3) is a basic element of the measuring system constructed for performing in-situ thermal diagnostic of buildings. The structure of this system is shown in Fig. 1. The main role of the communicator is to manage a group of measuring instruments organized in nodes (Fig. 2), which consists in sending the control data and receiving the measured results. Moreover, it realizes wireless data transmission in a diagnostic system. All these functions should be performed parallel in real-time by a multitasking operating system managing a microcontroller which controls the communicator. To achieve this aim, an RTOS (Reduced Task Operating System) system, working on the time-sharing rule, has been developed. It has been constructed under the assumption that the number of tasks is constant and equal to the number of serial ports of the microcontroller as it results from the block-diagram of the communicator shown in. Fig. 4. The tasks activated in the selected working state of the system are chosen from the tasks which have been developed for the measurement instruments ready to be used in the diagnostic system. The main principles of the built RTOS functioning are described by the bock-diagrams shown in Figs. 5 and 6. The main loop program (Fig. 5) enables assignment of the constant time quantum of the microcontroller processor to the succeeding realized tasks while the interrupt service subroutines (Fig. 6) allows performing data transmission in the node managed by a communicator for both measurement instruments and the ZigBee module used in the diagnostic system for wireless communication. All these programs are realized independently and in real-time. Some exemplary solutions written in C, basic for the properties of the developed RTOS, are presented in Section 4.

Keywords: measurement system for thermal diagnostic of buildings, operating system of a microcontroller, multiprocessing.

1. Struktura i działanie systemu

Zadaniem omawianego systemu diagnostyki jest dostarczanie zbiorów wyników pomiarowych charakteryzujących stan cieplny obiektu, którym może być budynek lub zespół budynków. Zatem system ten ma służyć do wykonywania pomiarów in-situ, przy czym uzyskiwane wyniki mogą być wykorzystywane zarówno dla celów badawczych, jak i w celu przeprowadzenia audytu energetycznego budynku. Analiza aktualnego stanu rozwoju tego rodzaju systemów doprowadziła do wniosku, że nie ma na rynku rozwiązań, które mogłyby posłużyć jako źródło inspiracji w konstruowaniu systemu o wymaganych właściwościach. W związku z tym opisywany tu system został zbudowany od podstaw, na podstawie przyjętych założeń [1].

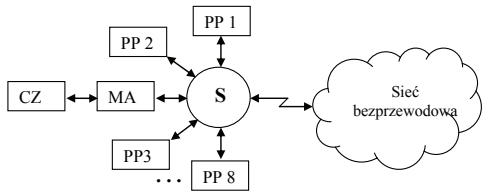
Ogólną strukturę systemu pokazano na rys. 1. Pomiary wykonywane są za pomocą zespołów wyspecjalizowanych przyrządów, zorganizowanych w węzły pomiarowe oznaczone jako $W_1,\ldots,W_n$. Każdy z węzłów zarządzany jest przez sterownik węzła nadzorujący przepływ informacji w jego obrębie przy użyciu interfejsów szeregowych RS 232 i RS 485 [2]. Węzły są integrowane w systemie przy użyciu sieci bezprzewodowej zbudowanej w standardzie ZigBee [3].



Rys. 1. Ogólna struktura systemu diagnostyki cieplnej budynków, $W_1,\ldots,W_n$ – węzły pomiarowe, K – komunikatory
Fig. 1. General structure of the system of building thermal diagnostic, $W_1,\ldots,W_n$ – measurement nodes, K - communicators

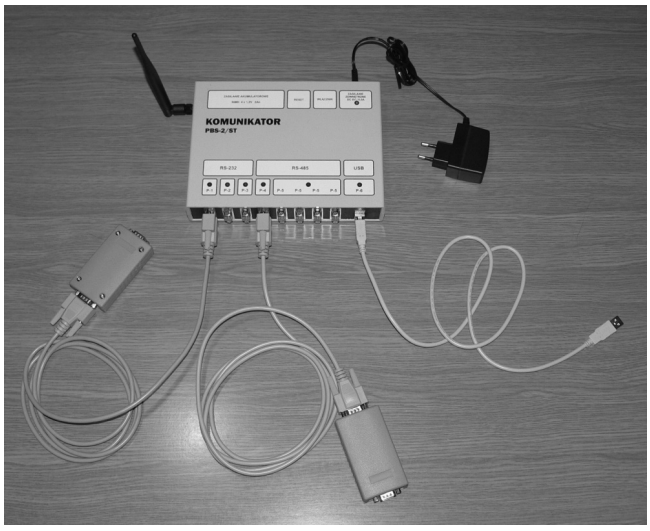
Funkcję sterownika spełnia wielofunkcyjne urządzenie nazywane komunikatorem i oznaczone na rys. 1 symbolem K, zaprojektowane w ramach projektu dla celów realizacji wymiany informacji w systemie, która odbywa się między lokalnymi bazami danych tworzonymi w sterownikach węzłów, a centralną bazą danych działającą na komputerze lokalnym [4]. Prócz funkcji sterownika komunikator może spełniać w systemie zadania koordynatora sieci bezprzewodowej oraz rutera sieci bezprzewodowej służącego do wzmacniania sygnału radiowego.

Z punktu widzenia struktury systemu jego podstawowym podzespołem jest węzeł pomiarowy organizujący pracę grupy przyrządów pomiarowych w sposób pokazany na rys. 2. Węzłem zarządza sterownik S nadzorujący komunikację z przyrządami pomiarowymi, której celem jest przekazywanie poleceń sterujących i odbiór danych pomiarowych za pomocą interfejsów szeregowych. Ponadto sterownik przeprowadza archiwizację danych ze wszystkich przyrządów za okres trwania eksperymentu pomiarowego, a także komunikuje się z koordynatorem sieci bezprzewodowej.



Rys. 2. Struktura węzła pomiarowego systemu, S oznacza sterownik węzła, PP – przyrząd pomiarowy, MA – moduł analogowy, CZ – czujnik
Fig. 2. Structure of the system measurement node, S denotes a node controller, PP – measurement instrument, MA – analog module, CZ – sensor

Do jednego sterownika można przyłączyć za pomocą złącz szufladowych do 8 przyrządów pomiarowych, przy czym 4 z nich mogą być modułami analogowymi zbudowanymi w celu umożliwienia wykorzystania w systemie czujników o wyjściu napięciowym, prądowym lub rezystancyjnym. Przyrządy używane w aktualnej wersji systemu scharakteryzowano w artykule [5]. Widok komunikatora z modułami analogowymi pokazano na rys. 3.



Rys. 3. Fotografia pokazująca komunikator połączony kablami z 2 modułami analogowymi
Fig. 3. Photograph showing the communicator connected with 2 analog modules by cables

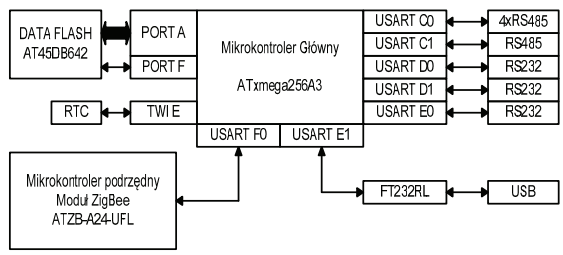
Dane pomiarowe gromadzone na bieżąco w sterowniku węzła są zarazem przesyłane za pomocą sieci bezprzewodowej do komunikatora pracującego jako koordynator sieci bezprzewodowej, który w tej roli jest połączony z komputerem lokalnym przy użyciu łącza USB. Ogólnie zadaniem tego komputera jest zarządzanie systemem z poziomu obiektu, co w szczególności polega na konfigurowaniu systemu przez przekazywaniu do węzłów informacji z jakimi przyrządami mają współpracować oraz określeniu parametrów pomiarów (rodzaju mierzonej wielkości, częstotliwości próbkowania itp.). Ponadto komputer lokalny odbiera na bieżąco dane pomiarowe z węzłów, przeprowadza ich archiwizację w bazie danych oraz służy do komunikowania się operatora z systemem.

Komputer lokalny jest instalowany na obiekcie, przy czym możliwe jest jego połączenie z komputerami zdalnymi za pośrednictwem Internetu. Komputery te wykorzystywane są jako zdalne konsole komputera lokalnego, dzięki czemu umożliwiają realizację wszystkich jego zadań na odległość. W przypadku braku bezpośredniego dostępu do Internetu można wykorzystać w tym celu sieć komórkową.

System może być wykorzystywany w różnych konfiguracjach, między innymi komunikatory mogą zostać użyte jako rejestratory danych pomiarowych. Konfiguracje te opisano w artykule [5].

2. Budowa komunikatora

Komunikator jest urządzeniem mikroprocesorowym o schemacie pokazanym na rys. 4. Jest on wyposażony w mikrokontroler ATXMEGA128-A3 [6] zawierający pamięć programu typu flash o pojemności 120 KB z dodatkowym obszarami: 8 KB, w którym mogą być lokowane dane oraz 8 KB z wpisanym programem ładującym. Wewnętrzna pamięć danych obejmuje 8 KB pamięci trwałej EEPROM oraz 2 KB pamięci statycznej SRAM.



Rys. 4. Schemat blokowy komunikatora
Fig. 4. Block-scheme of the communicator

Mikrokontroler komunikatora współpracuje z trwałą pamięcią zewnętrzną DataFlash AT45DB642 firmy Atmel o pojemności 64 Mb [7], w której tworzona jest lokalna baza danych służąca do przechowywania wyników pomiaru. Pojedynczy wynik stanowi 16 bajtowy rekord będący strukturą danych obejmującą 32 bitowy identyfikator, 32 bitowy znacznik czasu, 32 bitowy wynik pomiaru w formacie zmiennoprzecinkowym IEEE-754, 8 bitowy numer kanału oraz znacznik końca rekordu. Pamięć ta jest zorganizowana w 8192 strony o rozmiarze 1024 bajty każda, co pozwala na przechowanie ponad 500 000 wyników pomiaru. Dzięki temu możliwa jest realizacja pomiarów nawet przy długotrwałym zerwaniu transmisji bezprzewodowej, a także wykorzystanie komunikatora do pracy autonomicznej (bez komunikacji z komputerem lokalnym) jako rejestratora wyników pomiaru. W takim przypadku wyniki zapamiętane w lokalnej bazie danych są przekazywane do systemu po zakończeniu pomiarów i po podłączeniu komunikatora do dowolnego komputera w systemie. Wyniki pomiaru są znakowane czasowo za pomocą czasu astronomicznego określanego z rozdzielczością 1 sekundy. Do jego odmierzania zastosowano układ DS1307 firmy MAXIM [8].

Do komunikacji bezprzewodowej między komunikatorami wykorzystano moduł ATZB-A24-UFL firmy Atmel pracujący w standardzie ZigBee [3], który sprzężony jest z mikrokontrolerem przy użyciu portu szeregowego F0. Moduł ten przy pełnej mocy nadajnika zapewnia transmisję na odległość do 1 km przy braku przeszkód. Może być skonfigurowany jako urządzenie końcowe sieci bezprzewodowej i w takiej roli pracuje w sterowniku węzła. W przypadku, gdy moduł spełnia funkcję rutera, komunikator wykorzystywany jest do wzmacniania sygnałów radiowych i wówczas mikrokontroler wprowadzany jest w stan uśpienia. Natomiast użycie tego modułu jako koordynatora sieci powoduje, że komunikator spełnia rolę elementu sprzęgającego sieć bezprzewodową z komputerem lokalnym za pomocą interfejsu USB połączonego z portem E1 za pośrednictwem adaptera FT232RL (patrz rys. 4).

Gdy komunikator pracuje jako sterownik, do wymiany danych z przyrządami pomiarowymi wykorzystywane jest 5 portów szeregowych C0, C1, D0, D1 i F0. Port C0 połączony jest z adapterem interfejsu RS 485, którego sygnały wyprowadzone są równolegle na 4 złącza sterownika. W ten sposób uzyskuje się 4 kanały transmisyjne wykorzystywane do komunikacji sterownika z modułami analogowymi. Port ten można użyć do przyłączenia jednego przyrządu firmowego, gdyż takim przypadku pozostałe kanały nie mogą być wykorzystywane (przyrządy firmowe nie mogą być identyfikowane za pomocą adresu). Pozostałe 4 porty mikrokontrolera służą do podłączania przyrządów firmowych, przy czym port C1 pracuje w standardzie RS 485, a porty D0, D1 i F0 w standardzie RS 232C.Wszystkie kanały transmisyjne wyposażone są w optoizolację.

3. Ogólny opis oprogramowania komunikatora

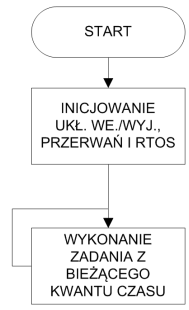
Oprogramowanie komunikatora napisano w języku C. Można je ogólnie podzielić na dwie części:

- system operacyjny organizujący wykonywanie programów użytkowych,
- programy użytkowe realizujące zadania związane z obsługą przyrządów pomiarowych, komunikacją bezprzewodową w systemie oraz komunikacją z komputerem lokalnym.

Komunikator pracujący jako sterownik węzła musi realizować współbieżnie wiele zadań związanych z obsługą przyrządów pomiarowych oraz komunikacją w systemie. W tym celu opracowano system operacyjny typu RTOS (ang. Reduce Task Operating System) [9] działający na zasadzie podziału czasu procesora na kwanty [10] o stałej wartości przydzielane zadaniom, których liczba i kolejność wykonywania jest z góry ustalona. Liczba zadań użytkowych jest równa liczbie portów szeregowych sterownika, gdyż przyjęto zasadę, że każde z zadań wykorzystuje jeden określony port sterownika do komunikacji z urządzeniem przyłączonym do złącza przydzielonego do tego portu. Łącznie wyodrębniono 9 zadań, z których Zadanie 0 realizuje tzw. pętlę główną, spełniającą funkcję planisty zadań systemu operacyjnego, 7 kolejnych zadań stanowi zadania użytkowe przypisane do określonych portów szeregowych, a zadanie 8 o nazwie **IDLE** jest zadaniem pustym, które jest aktywowane, gdy jakiś port szeregowy nie jest wykorzystywany (brak jest zadania użytkowego korzystającego z tego portu).

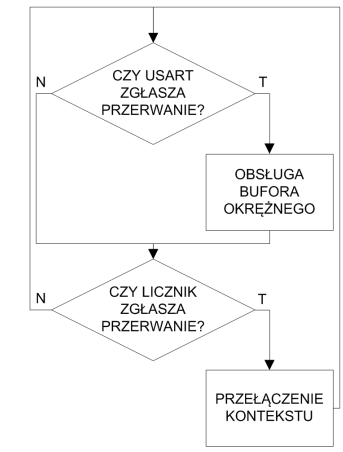
Opracowany system operacyjny wykorzystuje karuzelowy algorytmem realizacji zadań. Przełączanie zadań następuje po upływie kwantu czasu wynoszącego 1,25 ms, przy czym samo zadanie jest wykonywane przez 1,24 ms, a pozostały czas równy 1 ms przeznaczony jest na przełączanie kontekstu, polegające na zapisie na stosie stanu kończonego zadania i odczytaniu ze stosu stanu zadania aktywizowanego jako kolejne. Stan zadania określony jest przez zawartości wszystkich rejestrów roboczych mikrokontrolera oraz dane istotne dla reazlizacji zadania. Łączny czas wykonywania 8 zadań wynosi 10 ms, a zatem każdemu zadaniu procesor jest przydzielany z częstością 100 razy na sekundę.

W zbudowanym systemie operacyjnym można wyodrębnić dwie niezależne składowe, z których jedna realizuje pętlę główną, a druga obsługę przerwań. Pętla główna organizuje współbieżną realizację zadań na zasadzie podziału czasu i działa w sposób pokazany ogólnie na rys. 5. Po rozpoczęciu pracy komunikatora następuje zainicjowanie układów wejścia/wyjścia mikrokontrolera oraz obsługi przerwań, a także przygotowanie środowiska do działania samego systemu operacyjnego, co polega m.in. na zainicjowaniu stosu. Po tych czynnościach rozpoczyna się wykonywanie zadań użytkowych w niekończącej się pętli w kolejności zgodnej z numeracją portów szeregowych mikrokontrolera (porty C0, C1, D0, D1 i E0, E1 i F0 – patrz rys. 4), a jako ostatnie realizowane jest Zadanie 0**.** Wybór zadań użytkowych, wykonywanych w poszczególnych kwantach czasu, zależy od konfiguracji systemu diagnostyki określanej podczas jego instalacji w badanym obiekcie.



Rys. 5. Sieć działań programu realizującego pętlę główną
Fig. 5. Block-diagram of the program realizing the main loop

Obsługa przerwań realizuje podstawowy mechanizm reakcji systemu operacyjnego na zdarzenia zewnętrzne i jest wykonywana w tle programu realizującego pętlę główną, w sposób pokazany ogólnie na rys. 6.



Rys. 6. Sieć działań obsługi przerwań
Fig. 6. Block-diagram of the microcontroller interrupts realization

Sygnały przerwań odbierane są z portów szeregowych i licznika odmierzającego kwanty czasu. Podczas obsługi przerwania następuje zawieszenie realizacji aktualnie wykonywanego zadania, co powoduje, że czas niezbędny do wykonania podprogramu obsługi przerwania jest zabierany z kwantu przydzielonego aktywnemu zadaniu. Nie powoduje to istotnego skrócenia efektywnego czasu przydzielonego zadaniu, gdyż wykonywanie podprogramów obsługi przerwań trwa stosunkowo krótko w relacji do kwantu czasu.

Komunikator do transmisji danych wykorzystuje 7 portów szeregowych, które działają równocześnie i niezależnie od siebie. Aby umożliwić ciągłość transmisji danych przez porty we współpracy z zadaniami użytkowymi przełączanymi co 10 ms, dla każdego z portów zbudowano programy realizujące bufor okrężny [9] – zarówno dla danych odbieranych, jak i wysyłanych. Sposób budowy bufora okrężnego opisano w kolejnym punkcie.

Scharakteryzowana struktura oprogramowania pozwala na pracę komunikatora w jednym z trzech trybów: jako sterownika węzła, jako koordynatora sieci bezprzewodowej oraz jako rutera wzmacniającego sygnał radiowy. W pierwszym przypadku moduł ZigBee komunikatora pełni funkcję urządzenia końcowego sieci bezprzewodowej. Wówczas oprogramowanie mikrokontrolera wykonuje następujące działania:

- sprawdza bieżącą konfigurację przyrządów pomiarowych w węźle i na tej podstawie uruchamia poszczególne zadania pomiarowe oraz realizuje cykliczną aktywację wykonywania pomiarów przez przyrządy,
- aktywizuje podprogramy (zadania pomiarowe) służące do obsługi przyrządów pomiarowych przyłączonych do portów szeregowych,

- realizuje transmisję radiową wyników pomiarów do centralnej bazy danych organizowanej na komputerze lokalnym,
- przeprowadza archiwizację wyników pomiarów w lokalnej bazie danych mieszczącej się w pamięci trwałej komunikatora.

Komunikator zostaje wprowadzony w tryb koordynatora w przypadku, gdy jest połączony z komputerem lokalnym, na którym działa oprogramowanie nadzorujące działanie całego systemu [4], w tym program sprzęgający centralną bazę danych z bazami lokalnymi organizowanymi w sterownikach węzłów. Program zarządzający m.in. odczytuje z centralnej bazy danych konfigurację systemu i wysyła ustawienia portów do odpowiednich sterowników węzłów i odbiera od sterowników węzłów wyniki pomiarów, przechowywane w ich lokalnych bazach danych, po czym przesyła je do centralnej bazy danych. W trybie koordynatora komunikator nie wykonuje obsługi sygnałów przerwań z portów szeregowych C0, C1, D0, D1 i E0, a jedynie obsługuje sygnały przerwań z portów szeregowych E1 (połączenie z interfejsem USB) i F0 (połączenie z modułem ZigBee). Działanie programu sterownika w tym trybie polega ogólnie na kopiowaniu bajtów nadchodzących z portu E1 do portu F0 i z portu F0 do portu E1, przy czym obsługa pozostałych portów szeregowych jest wyłączona (odpowiednie przerwania są nieaktywne).

Komunikator pracuje w trybie rutera, gdy podczas konfiguracji systemu w module ZigBee zostanie ustawiony tryb Router. Do portów komunikatora nie są wówczas dołączane żadne przyrządy pomiarowe, a zatem mikrokontroler jest nieaktywny (przy zasilaniu bateryjnym komunikatora może być on wprowadzony w stan uśpienia). W stanie aktywności znajduje się wyłącznie moduł ZigBee, który retransmituje odbierane dane.

4. Wybrane rozwiązania systemu operacyjnego

Podstawowym zadaniem systemu operacyjnego jest współbieżne wykonywanie zadań użytkowych. Wybór zadań do realizacji w poszczególnych kwantach czasu odbywa się na podstawie tablicy konfiguracji komunikatora, przesyłanej z komputera lokalnego podczas instalacji systemu. Tablica ta umożliwia przyporządkowanie określonego portu zadaniu użytkowemu realizującego obsługę przyrządu podłączonego do tego portu i składa się części zorganizowanych jako struktura. Przykładowa struktura ma postać:

```
typedef struct {
 unsigned long id;  // unikatowy identyfikator przyrządu
 unsigned long ma;   // numery aktywnych kanałów – maska 32 bitów
 unsigned long in;  // odstęp pomiędzy pomiarami w s
 unsigned char pr;  // numer programu
 unsigned char sp;  // szybkość transmisji 0-9600; 1-19200;
  // 2-38400; 3-57600; 4-115200;
 unsigned char ga[2]; // dopełnienie 0xff
} PORT;
```

Każde z zadań wykonywane jest przez kwant czasu, po upływie którego następuje przełączanie kontekstu polegające na zapisaniu stanu zawieszanego zadania na stosie i odczytaniu ze stosu stanu zadania, któremu został przydzielony następny kwant. W zbudowanym systemie operacyjnym stan zadania jest zdefiniowany jako struktura:

```
typedef struct {
 volatile unsigned long ss; // SYSTEM_SIGNAL
 volatile unsigned char *s; // wskaźnik stosu
 volatile unsigned char rs; // numer portu szeregowego
 volatile unsigned char en; // stan zadania
} RTOS_context;
```

Obszar stosu przydzielony do zapis stanu zadania jest definiowany jako:

```
volatile unsigned char RTOS_STAKS[8][RTOS_STACK_SIZE];
```

gdzie RTOS_STACK_SIZE określa rozmiar obszaru stosu przeznaczony dla jednego zadania. Przyjęcie takiego rozwiązania powoduje, że kwanty czasu muszą być ponumerowane. Przypisanie konkretnego zadania do kwantu o określonym numerze następuje przy użyciu funkcji RTOS_SetTask:

```
volatile RTOS_context RTOS_TASKS[9]; // tablica kontekstów zadań
void RTOS_SetTask( unsigned char slot, unsigned char port, void (* fun)())
{
 union { void (* back)();
      unsigned char c[4]; } fun2char; // unia do konwersji adresu funkcji na pojedyncze bajty
 fun2char.back=fun;
 cli(); // blokada przerwań na czas ustawiania zadania
     RTOS_TASKS[slot].s=(unsigned char *)&RTOS_STAKS[slot-1][RTOS_STACK_SIZE-43]; // ustawienie wskaźnika stosu
 RTOS_TASKS[slot].s[38]=fun2char.c[3]; // przepisanie adresu funkcji
 RTOS_TASKS[slot].s[39]=fun2char.c[2]; // przepisanie adresu funkcji
 RTOS_TASKS[slot].s[40]=fun2char.c[1]; // przepisanie adresu funkcji
 RTOS_TASKS[slot].s[41]=fun2char.c[0]; // przepisanie adresu funkcji
 RTOS_TASKS[slot].en=1; // uaktywnienie zadania
 RTOS_TASKS[slot].rs=port; // przepisanie numeru portu
 sei(); // odblokowanie przerwań
}
```

Komunikator wyposażony jest w 7 portów szeregowych, które realizują transmisję równocześnie i niezależnie od siebie. W celu umożliwienia koordynacji transmisji i zadań użytkowych, przełączanych co kwant czasu, zbudowane zostały funkcje realizujące tzw. bufor okrężny [3] zarówno dla danych odbieranych, jak i wysyłanych przez port szeregowy. Bufor okrężny jest to struktura danych, do której dostęp realizowany jest przez wskaźniki zapisu i odczytu w taki sposób, że po osiągnięciu końca bufora kolejny zapis lub odczyt następuje od początku bufora. W tym celu zdefiniowano zmienne RS_ilen oraz RS_olen, określające liczbę bajtów znajdujących się w danej chwili w buforach. Przykładowe funkcje bufor okrężnego dla portu mają postać:

```
/* BUFOR ODBIORU */
volatile unsigned char RSC0_ibuf[256]; // bufor na dane przychodzące
volatile unsigned char RSC0_iin,RSC0_iout,RSC0_ilen; // dane bufora we
ISR(USARTC0_RXC_vect) // przerwanie RX
{
 RSC0_ilen++; // zwiększenie liczby bajtów
 RSC0_ibuf[RSC0_iin]=USARTC0_DATA; // zapisanie danych w buforze
 RSC0_iin++;    // zwiększenie indeksu zapisu
 RSC0_iin&=0xFF; // maska na 256 bajtów w buforze
}
unsigned char RSC0_getchar(void)
{
 while(!RSC0_ilen); // oczekiwanie na pojawienie się bajt w buforze
```

```
 RSC0_ilen--; // zmniejszenie liczby bajtów w buforze
 RSC0_iout++; // zwiększenie indeksu odczytu
 RSC0_iout&=0xFF; // maska na 256 bajtów w buforze
 return(RSC0_ibuf[RSC0_iout]); // odczyt bajtu z bufora
}
/* BUFOR NADAWANIA */
volatile unsigned char RSC0_obuf[64]; // bufor na dane wychodzące
volatile unsigned char RSC0_oin,RSC0_oout,RSC0_olen; // dane bufora wy
ISR(USARTC0_DRE_vect) // podprogram obsługi przerwania pustego bufora
{
 if(RSC0_olen) // czy są bajty do wysłania?
 {
  PORTC_OUTSET=0x02; // włączenie nadawania (tylko rs485)
  RSC0_olen--;      // zmniejszenie liczby bajtów
  USARTC0_DATA=RSC0_obuf[RSC0_oout]; // zapis danych do bufora nadajnika
  RSC0_oout++;  // zwiększenie indeksu
  RSC0_oout&=0x3F; // maska rozmiaru bufora 64 bajty
 }
 else
 {
  USARTC0_CTRLA=0x28; // wyłączenie przerwania gdy nie ma już danych w buforze
 }
}
void RSC0_putchar(unsigned char dat)
{
 while(RSC0_olen>62); // ograniczenie liczby bajtów w buforze wyj.
 RSC0_obuf[RSC0_oin]=dat; // zapis danych do bufora
 cli(); // blokada przerwania
 RSC0_olen++; // modyfikacja liczby bajtów w buforze
 sei(); // odblokowanie przerwań
 RSC0_oin++; // zwiększenie indeksu zapisu
 RSC0_oin&=0x3F; // maska indeksu zapisu
 USARTC0_CTRLA=0x22; // aktywacja przerwania
}
ISR(USARTC0_TXC_vect)
{
 PORTC_OUTCLR=0x02; // wyłączenie nadawania (tylko rs485)
}
```

Inicjowanie każdego z portów szeregowych polega na określeniu wartości początkowych zmiennych opisujących bufory okrężne portu oraz ustawieniu rejestrów stanu portu zgodnie z parametrami transmisji właściwymi dla wykorzystania portu do realizacji przyporządkowanego mu zadania użytkowego.

5. Uwagi końcowe

Scharakteryzowany system operacyjny został zbudowany w celu współbieżnej realizacji zadań wykonywanymi przez mikrokontroler sterownika węzła zarządzającego zespołem przyrządów pomiarowych w systemie diagnostyki cieplnej budynków. System ten został sprawdzony w warunkach eksploatacyjnych, a uzyskane wyniki testów skłaniają do wniosku, że zbudowany system operacyjny dobrze realizuje w praktyce założenia przyjęte jako podstawa jego budowy. Właściwości tego systemu pozwalają na stwierdzenie, że zastosowane w nim rozwiązania mają w dużym stopniu cechy uniwersalne i mogą być wykorzystane również w innych konstrukcjach sterowników zbudowanych z wykorzystaniem mikrokontrolerów i zarządzających wielozadaniowym procesem pomiarowym.

Zbudowany system ma prostą strukturę, co jest związane z przyjęciem założenia, że liczba zadań jest stała (nie wszystkie muszą być aktualnie realizowane) i są one wykonywane w niezmiennej kolejności sukcesywnie przez kwanty czasu o stałej wartości. Należy jednak podkreślić, że rodzaje zadań w puli wykonywanej aktualnie przez sterownik mogą się zmieniać w zależności od dołączonych do niego przyrządów pomiarowych, o czym decyduje operator systemu diagnostyki w trakcie jego konfiguracji. Można powiedzieć, że pula potencjalnych zadań sterownika jest z punktu widzenia systemu diagnostyki praktycznie nieograniczona (ograniczeniem jest pojemność pamięci programu mikrokontrolera, która jest relatywnie duża) i zależna od rodzajów przyrządów aktualnie użytkowanych w systemie.

Prostota systemu operacyjnego umożliwia pełną kontrolę nad realizacją zadań użytkowych przez mikrokontroler, co jest ułatwione przez zastosowanie sztywnych reguł planowania zadań. W razie potrzeby można dość prostymi środkami uzyskać bardziej elastyczne działanie planisty polegające głównie na przydzielaniu większych porcji czasu procesora zadaniom o wyższym priorytecie. Można to zrobić przez wydłużanie kwantu czasu przyznawanego takim zadaniom, a także przez blokadę realizacji zadań o niskim priorytecie (przykładowo powtarzanych z niewielką częstotliwością) na określoną liczbę kwantów.

Istotną cechą systemu jest obsługiwanie komunikacji przez porty szeregowe w tle realizacji programów użytkowych. Jest to możliwe dzięki zastosowaniu równoczesnej i autonomicznej transmisji danych przez porty z wykorzystaniem przerwań i przy użyciu buforów okrężnych. Zapewnia to bezstratne przesyłanie danych między sterownikiem węzła a przyrządami pomiarowymi w sposób niezależny od realizacji zadań użytkowych, a w szczególności umożliwia współbieżną obsługę przyrządów i komunikację z koordynatorem sieci bezprzewodowej ZigBee.

Publikacja wykonana w ramach w ramach projektu strategicznego: Zintegrowany system zmniejszania eksploatacyjnej energochłonności budynków, zadanie badawcze: Rozwój diagnostyki cieplnej budynków.

6. Literatura

[1] Jakubiec J., Konopka K., Żurkowski R.: System pomiarowy do diagnostyki cieplnej budynków. Metrologia dziś i jutro – 2011. Monografia. Oficyna Wydawnicza Politechniki Białostockiej, Białystok, 2011.
[2] Mielczarek W.: Szeregowe interfejsy cyfrowe. Helion, Gliwice 1994.
[3] ZigBit™ 2.4 GHz Amplified Wireless Modules, atmel.com, czerwiec 2009.
[4] Konopka K. Program wizualizacji i zarządzania pomiarami w systemie diagnostyki cieplnej budynków. Materiały IX Konferencji Naukowo-Technicznej PPM'12, ss. 266-269.
[5] Jakubiec J.: System diagnostyki cieplnej budynków. Pomiary Automatyka Kontrola, 2013, nr 7, s. 712-716.
[6] Dokumentacja mikrokontrolera ATXMEGA128-A3, http://www.atmel.com/
[7] Dokumentacja pamięci DataFlash AT45DB642, http://www.atmel.com/
[8] Dokumentacja zegara DS1307, http://www.maxim-ic.com/
[9] Barry R.: Multitasking on an AVR – example C implementation of a multitasking kernel for the AVR, avrfreaks.net, marzec 2004.
[10] Silberschatz A., Peterson J.L., Galvin P. B.: Podstawy systemów operacyjnych. Wyd. V, WNT Warszawa, 2005.

otrzymano / received: 06.08.2013
przyjęto do druku / accepted: 03.03.2014 *artykuł recenzowany / revised paper*