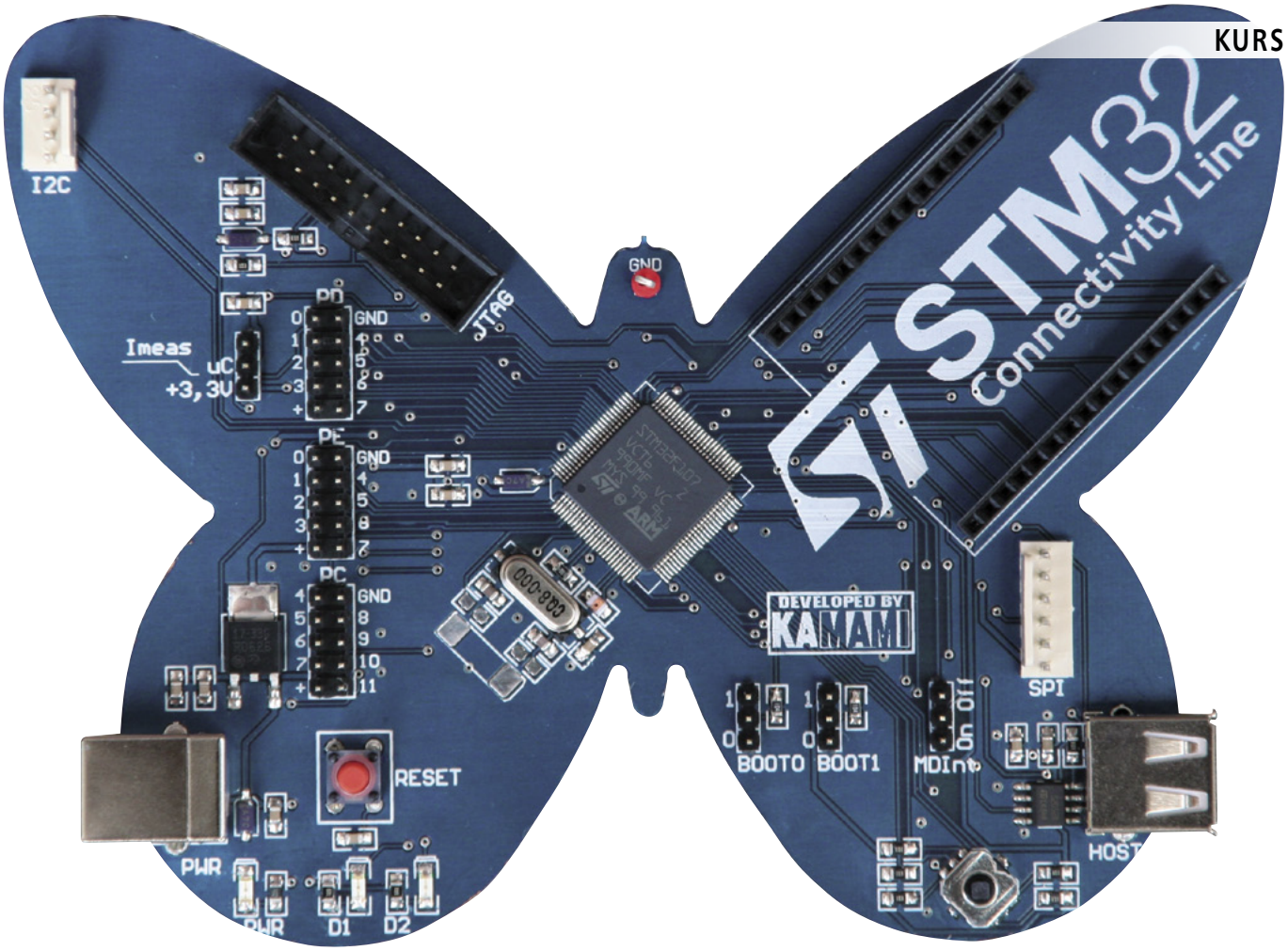


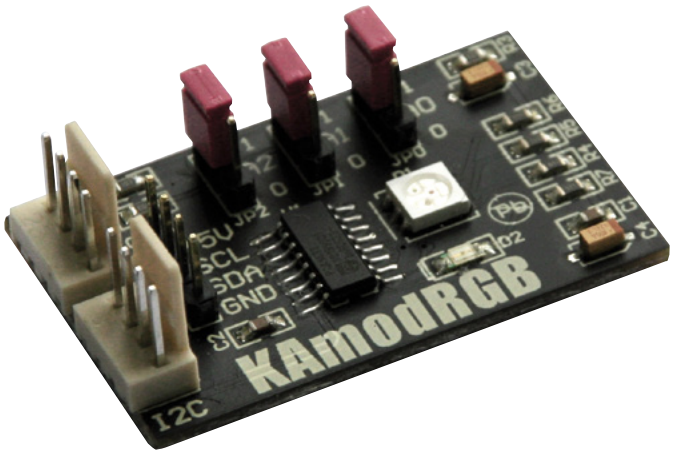
ISIX-RTOS
Przykłady w języku C

Miniaturowy system operacyjny ISIX przedstawiliśmy Czytelnikom w EP 7/2010. Przykłady napisane dla niego w języku C++ cieszyły się dużym zainteresowaniem, ale wielu Czytelników EP sygnalizowało chęć zapoznania się z podobnymi aplikacjami napisanymi w C. Odpowiedź na te postulaty przedstawiamy w artykule. Wszystkie przykłady przygotowano dla mikrokontrolera STM32F107 zastosowanego w zestawie STM32Butterfly.

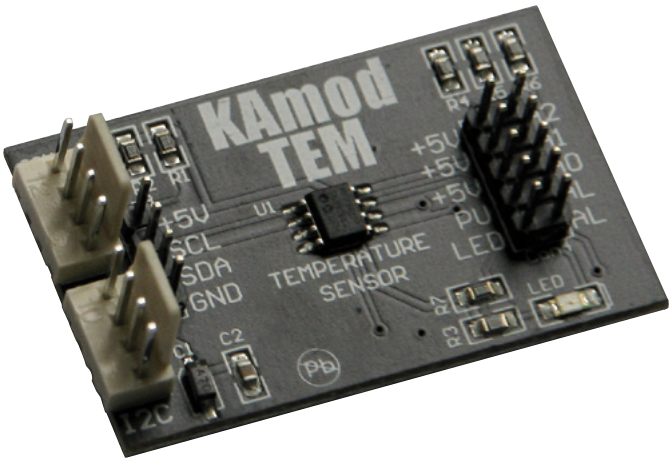
Przykład 3: termometr RGB

W ramach utrwalenia wiadomości na temat obsługi magistrali I²C w systemie ISIX oraz pokazania możliwości ciekawego modułu z diodą LED RGB, pokażemy w jaki sposób na podstawie poznanych wcześniej wiadomości stworzyć nietypowy termometr. Prezentacja wyników pomiaru odbywa się za pomocą diody RGB. Ciepło jest sygnalizowane kolorem czerwonym, zimno – kolorem niebieskim, a temperatury pośrednie są sygnalizowane przez inne kolory.

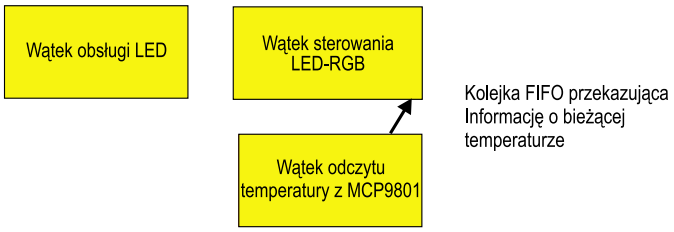
Sterowanie diodą LED RGB może być zrealizowane za pomocą układów PWM mikrokontrolera lub za pomocą specjalizowanych układów przeznaczonych do tego celu. W naszym przypadku postanowiono skorzystać z gotowego rozwiązania modułu KAmodRGB (firmy *Kamami* – **fotografia 1**), który wyposażono w 4-kanałowy sterownik LED z interfejsem I²C. Dzięki dodatkowemu sterownikowi, mikrokontroler jedynie przesyła za pomocą magistrali I²C dane o nasyceniu każdego z kolorów (RGB), a generowaniem odpowiednich sygnałów zajmuje się wyspecjalizowany układ. Rolę czujnika temperatury spełnia w przykładzie moduł KAmodTEM (**fotografia 2**), w którym zastosowano nowoczesny, cyfrowy czujnik temperatury MCP9801, również komunikujący się z mikrokontrolerem poprzez interfejs I²C.



Fotografia 1.



Fotografia 2.



Rysunek 3.

Po połączeniu obu modułów z zestawem STM32Butterfly należy w module KAmodRGB przestawić wszystkie zwory konfiguracji adresu w pozycje 0, w efekcie czego kontroler LED RGB będzie dostępny na magistrali pod adresem 0x00, natomiast moduł KAmodTEM (podobnie jak w poprzednim przykładzie) pod adresem 0x90.

Sposób działania aplikacji, z uwzględnieniem podziału na wątki pokazano na **rysunku 3**. Zasada działania aplikacji jest w zasadzie identyczna jak w poprzednim przykładzie. Różnica polega jedynie na zmianie funkcji wątku wyświetlania. Program rozpoczyna działanie od funkcji *main()* (**listing 1**).

Najpierw jest tworzony niezależny wątek migania diodą LED D1, następnie tworzona jest kolejka FIFO, której maksymalną pojemność ustawiono na 10 elementów zdefiniowanych jako struktura *msg*. Następnie inicjalizowana jest biblioteka obsługi interfejsu I²C, a w przypadku pomyślnego utworzenia kolejki FIFO, tworzone jest zadanie odczytu temperatury *temp_read_task* oraz wątek wyświetlania temperatury *display_srv_task*. Po utworzeniu wszystkich wątków , uruchamiany jest planista zadań systemu ISIX. Strukturę wiadomości nieco odmiennie niż w poprzednim przypadku zdefiniowano w sposób pokazany na **listingu 2**.

W tym przypadku celowo zdecydowano się na zdefiniowanie temperatury w postaci liczby zmiennoprzecinkowej typu float, aby pokazać że, operacje zmiennoprzecinkowe są wykonywane przez rdzeń *CORTEX-M3* bardzo sprawnie. Ten typ zostanie również użyty w przypadku obliczenia nasycenia barw w zależności od temperatury, o czym napiszemy w dalszej części.

Za odczytywanie temperatury z czujnika temperatury odpowiada wątek *temp_read task*, który działa identycznie jak w poprzednim przykładzie, jedyna różnica występuje w funkcji *tempsensor_get()* (**listing 3**).

Odczytana temperatura jest przeliczana na liczbę zmiennoprzecinkową. Tak wyliczona wartość jest przekazywana do struktury *msg* i przesyłana do kolejki FIFO. Za wizualizację temperatury odpowiedzialny jest wątek *display_srv_task*, który odbiera informację o aktualnej temperaturze z kolejki FIFO i prezentuje ją za pomocą diody RGB (**listing 4**).

Wątek wizualizacji rozpoczyna pracę od inicjalizacji kontrolera modułu KAmodRGB poprzez wywołanie funkcji *rgb_init()* (**listing 5**). W przypadku niepowodzenia (rezultat <0) , funkcja *isix_task_delete(NULL)* kasuje bieżący wątek który ją wywołał. Jeżeli inicjalizacja przebiegła poprawnie zadanie przechodzi do pętli realizującej cykl sterowania modułem.

Inicjalizacja modułu KAmodRGB sprowadza się do przesłania parametrów konfiguracyjnych do rejestrów układu PCA9633. Do rejestru konfiguracyjnego MODE1 *(0x00)* przesyłana jest wartość 0 co oznacza, że układ PCA będzie reagował tylko na adres własny (istnieje również możliwość reakcji na adres grupowy, co umożliwia równoczesne sterowanie kilku układów na magistrali.) Do rejestru konfiguracyjnego MODE2 (0x01) wpisywana jest również wartość 0, co powoduje skonfigurowanie wszystkich wyjść LED jako typ *OpenDrain*. Na koniec inicjalizacji do rejestru LEDOUT (0x08), wpisywana jest wartość 255, co powoduje uruchomienie wszystkich 4 kanałów PWM.

Jeżeli proces inicjalizacji przebiegł pomyślnie, to program wchodzi do pętli głównej, gdzie cyklicznie odczytywane są

List. 1. Program główny termometru RGB
```
/** Main func  */
int main(void)
{
        //Create ISIX blinking task
        isix_task_create( blinking_task, NULL,ISIX_PORT_SCHED_MIN_STACK_DEPTH, TASK_PRIO_LED);
        //Create fifo msgs
        fifo_t *temp_fifo = isix_fifo_create( 10, sizeof(struct msg) );
        //Initialize i2c bus
        i2cm_init(I2C_SPEED);
        if(temp_fifo)
        {
                //Create isix tasks (temp and disp)
                isix_task_create(temp_read_task,temp_fifo,TASK_STK_SIZE,TASK_PRIO_TEMP);
                isix_task_create(display_srv_task,temp_fifo,TASK_STK_SIZE,TASK_PRIO_TEMP);
        }
        //Start the scheduler
        isix_start_scheduler();
}
```

List. 2. Struktura wiadomości
```
//Message structure
struct msg
{
        float t;                        //Current temperature
        int errno;                      //Error code
};
```

List. 3. Odczyt temperatury z czujnika
```
static int tempsensor_get(float *t)
{
        static const unsigned char temp_reg = MCP9800_TEMP_REG;
        static char temp[2];
        int ecode;
        //Read the temperature
        ecode = i2cm_transfer_7bit(TEMPSENSOR_I2CADDR,&temp_reg,sizeof(temp_reg),temp,sizeof(temp));
        //Convert to integer
        if(ecode>=0)
        {
                *t = (float)temp[0] + (temp[1]>>4)/16.0f;
        }
        return ecode;
}
```

List. 4. Wyświetlanie temperatury za pomocą diody RGB

```
#define COLD_VALUE 24.0f        /* Cold value */
#define HOT_VALUE 30.0f         /* Hot Value */
#define PWM_MAX 255                     /* Maximum pwm value */
#define PWM_MAX_F 255.5
#define A_VALUE -0.1f           /* A value of quadratic equation */
#define FAIL_AMBER_VAL 200      /* Fail amber 4 PWM ratio */

/** Display server task */
static ISIX_TASK_FUNC(display_srv_task, entry_params)
{
        fifo_t *temp_fifo = (fifo_t*)entry_params;
        struct msg msg;                 //Message structure
        //If init rgb fail terminate task
        if(rgb_init()<0)
        {
                isix_task_delete(NULL);
        }
        unsigned pcolor = 0;
        for(;;)
        {
                //Read data from fifo
                if(isix_fifo_read( temp_fifo, &msg,ISIX_TIME_INFINITE )==ISIX_EOK)
                {
                        //Display temp or error
                        if(msg.errno>=0)
                        {
                                //Calculate saturation
                                int b = PWM_MAX_F * (HOT_VALUE - msg.t)/(HOT_VALUE-COLD_VALUE);
                                int g = PWM_MAX_F * (A_VALUE * (msg.t-COLD_VALUE)*(msg.t-HOT_VALUE));
                                int r = PWM_MAX_F * (msg.t-COLD_VALUE)/(HOT_VALUE-COLD_VALUE);
                                if(b<0) b=0; else if(b>PWM_MAX) b=PWM_MAX;
                                if(r<0) r=0; else if(r>PWM_MAX) r=PWM_MAX;
                                if(g<0) g=0; else if(g>PWM_MAX) g=PWM_MAX;
                                //Calculate color
                                unsigned color = RGB(r,g,b);
                                //Update britness only if color changed value
                                if(color!=pcolor)
                                        rgb_set_color(color);
                                pcolor = color;
                        }
                        else    // If fail enable only amber
                                rgb_set_color(RGBA(0,0,0,FAIL_AMBER_VAL));
                }
        }
}
```

List. 5. Inicjalizacja kontrolera modułu KAmodRGB

```
//Initialize rgb controller
static int rgb_init(void)
{
        int errcode;
        static const uint8_t init_reg[RGB_INIT_NUMREG][RGB_INIT_NSEQ] =
        { { 0, 0 }, { 1, 0 }, { 8, 255 } };
        for(int i=0;i<RGB_INIT_NUMREG;i++)
        {
        if(
                (errcode=i2cm_transfer_7bit(RGBCTRL_I2CADDR,init_reg[i],RGB_INIT_NSEQ ,NULL,0))
                < 0
        )
                        return errcode;
        }
        return errcode;
}
```

wiadomości otrzymywane od wątku odczytującego temperaturę z czujnika. Jeżeli w strukturze wiadomości nie został ustawiony kod błędu, wówczas wyznaczany jest poziom nasycenia poszczególnych barw *R, G, B* na podstawie aktualnej temperatury. Wartości skrajne temperatury *COLD_VALUE* (kolor niebieski) i *HOT_VALUE* (kolor czerwony) zdefiniowano odpowiednio na wartości 24 i 30°C, dzięki czemu jedynie poprzez dotknięcie ręką czujnika, temperatury będziemy mogli zobaczyć zmieniające się barwy od niebieskiego do czerwonego. Ponieważ układ PCA9633 zapewnia 8-bitowy PWM, wartość 0 odpowiada odpowiednio wygaszeniu diody dla danego kanału, a 255 pełną jasność diody. Wyznaczenie wartości współczynników PWM dla nasycenia kolorów R i B została zrealizowana jako funkcja liniowa. Natomiast nasycenie koloru zielonego zrealizowano jako funkcję kwadratową z miejscami zerowymi występującymi dla skrajnych wartości temperatury (zimno, gorąco) oraz ekstremum przypadające mniej więcej w połowie zakresu temperatur. Po wyliczeniu nasycenia poszczególnych składowych, wywoływane jest makro, grupujące poszczególne kolory w jedną 32-bitową zmienną reprezentującą wszystkie składowe kolorów. Funkcja *rgb_set_color()* (**listing 6**), przesyłająca wartości kolorów do układu PCA, wywoływana jest jedynie wtedy, gdy nastąpiła zmiana nasycenia w stosunku do poprzedniego cyklu.

Działanie funkcji ustawiającej nasycenie jest bardzo proste, i sprowadza się do przesłania wartości nasycenia kolorów do rejestrów (PWM0...PWM3), w wyniku czego następuje ustawienie odpowiedniego koloru diody LED.

W rzeczywistej aplikacji termometru należy poszerzyć zakres temperatur ciepło-zimno oraz można pomyśleć o wyznaczeniu różnych wartości ciepło-zimno w zależności od pory roku itp.

Lucjan Bryndza, EP
lucjan.bryndza@ep.com.pl

List. 6. Funkcja przesyłająca wartości kolorów do PCA

```
static int rgb_set_color (unsigned c)
{
        uint8_t txbytes[] =
        {
                0x82,       //Color register
                (c)>>24,    //A
                (c)>>16,    //B
                (c),        //R
                (c)>>8      //G
        };
        return i2cm_transfer_7bit(RGBCTRL_I2CADDR,txbytes,sizeof(txbytes),NULL,0);
```