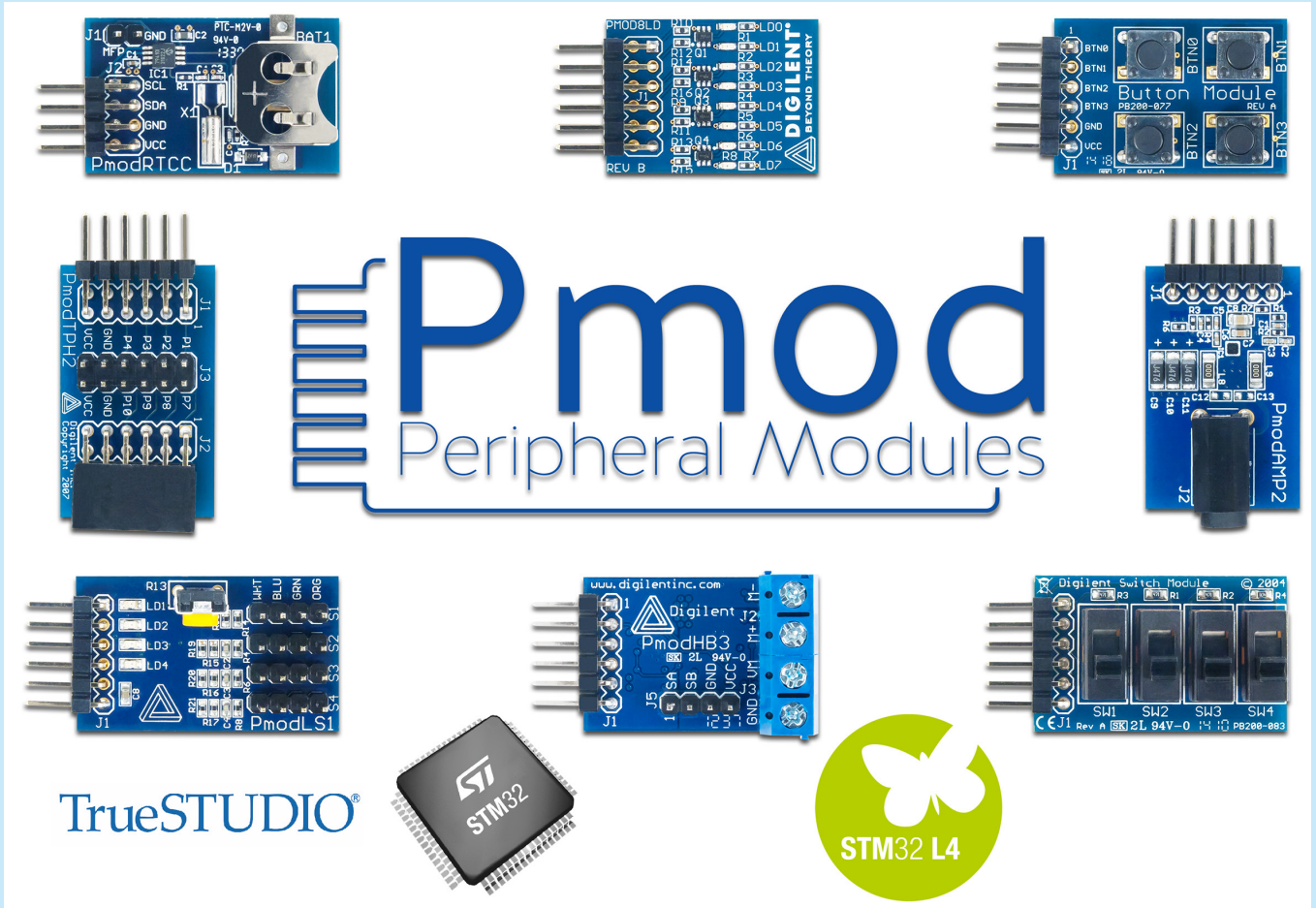


Digilent PMOD i STM32 (2)

Biblioteki do obsługi modułów peryferyjnych

W drugiej części cyklu poświęconego modułom peryferyjnym Pmod przedstawimy dwa kolejne moduły z rodziny Pmod: PmodHYGRO (czujnik temperatury i wilgotności) i PmodOLEDrgb (kolorowy wyświetlacz graficzny OLED).

Przedstawione przykłady kodów wykorzystują bibliotekę STM32Cube_FW_L4 i mogą być uruchomione w środowisku Atollic TrueSTUDIO na zestawie deweloperskim KAmeLeon (*www.kameleonboard.org*).

Biblioteki opisane w artykule są dostępne bezpłatnie do pobrania na stronie ***KAMAMI.pl***. W oknie wyszukiwarki trzeba wpisać nazwę modułu Pmod, na stronie wyrobu jest dostępny link do biblioteki.

PmodHYGRO

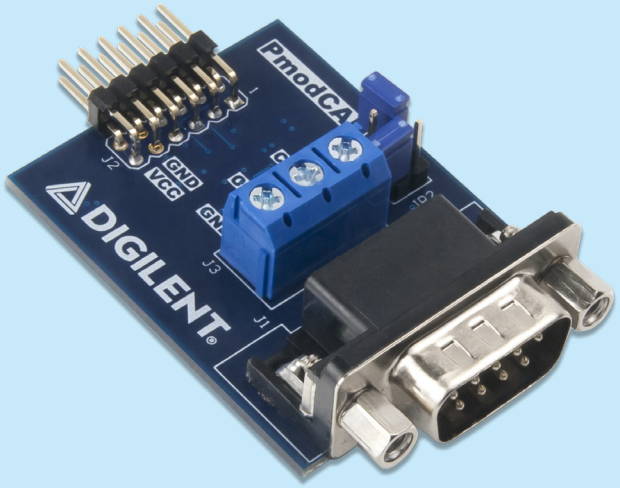
Moduł PmodHYGRO (**fotografia 1**) jest czujnikiem temperatury i wilgotności względnej opartym o układ Texas Instruments HDC1080. Zakres mierzonych temperatur wynosi od –40°C do +125°C, a rozdzielczość wyniku pomiaru może być skonfigurowana na 11 lub 14 bitów. Dokładność pomiaru zależy od mierzonej wartości, co pokazano na **rysunku 2**.

Zakres pomiarowy wilgotności wynosi 0…100%, a wynik pomiaru może być podawany z rozdzielczością 8, 11 lub 14 bitów. Podobnie jak w wypadku temperatury, dokładność pomiaru zależy również

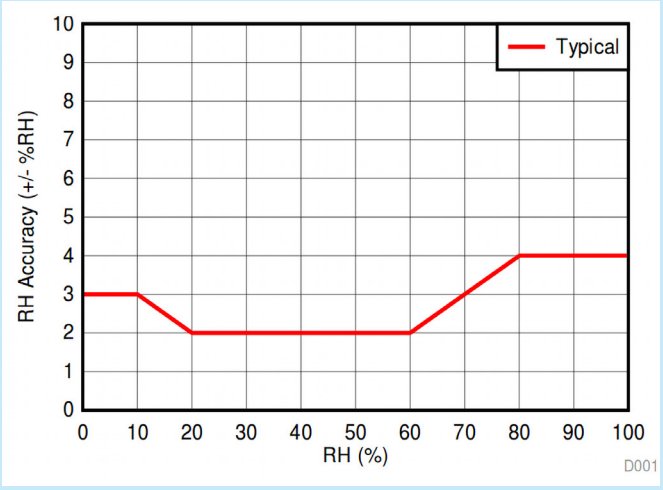
NAME	Bits			DESCRIPTION
RST	[15]	Software reset bit	0	Normal Operation, this bit self clears
			1	Software Reset
Reserved	[14]	Reserved	0	Reserved, must be 0
HEAT	[13]	Heater	0	Heater Disabled
			1	Heater Enabled
MODE	[12]	Mode of acquisition	0	Temperature or Humidity is acquired.
			1	Temperature and Humidity are acquired in sequence, Temperature first.
BTST	[11]	Battery Status	0	Battery voltage > 2.8V (read only)
			1	Battery voltage < 2.8V (read only)
TRES	[10]	Temperature Measurement Resolution	0	14 bit
			1	11 bit
HRES	[9:8]	Humidity Measurement Resolution	00	14 bit
			01	11 bit
			10	8 bit
Reserved	[7:0]	Reserved	0	Reserved, must be 0

Rysunek 1. Rejestr konfiguracyjny (0x02) (źródło: dokumentacja układu HDC1080)

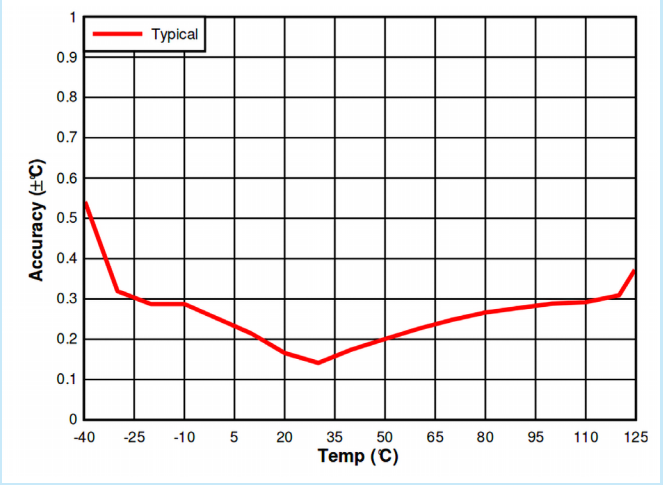
Poprzednie części kursu i dodatkowe materiały dostępne są na stronie www.media.avt.pl



Fotografia 2.Wygląd modułu PmodHYGRO



Rysunek 3. Zależność dokładności pomiaru temperatury od mierzonej wartości (źródło: dokumentacja układu HDC1080)



Rysunek 4. Zależność dokładności pomiaru wilgotności względnej od mierzonej wartości (źródło: dokumentacja układu HDC1080)

od mierzonej wartości. Tę zależność pokazano na **rysunku 3**. Czas pomiaru obu wielkości jest zależny od skonfigurowanej rozdzielczości i dla 14-bitów nie przekracza 6,5 ms.

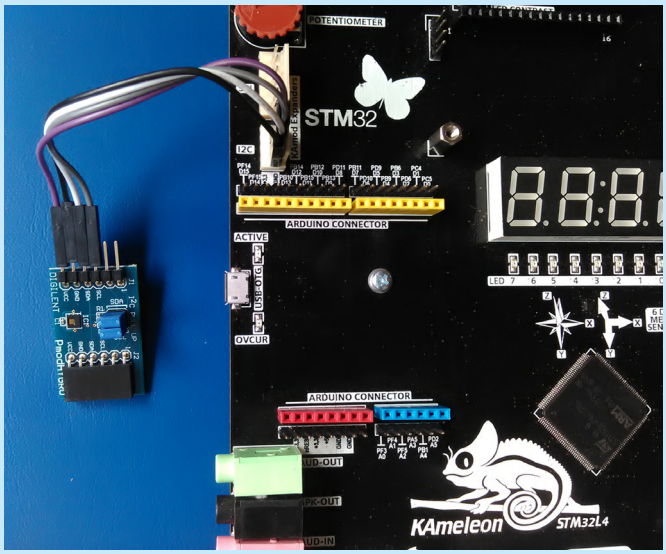
Moduł PmodHYGRO wyposażono w 6-pinowe złącze Pmod dla interfejsu I²C. Na potrzeby przykładu moduł należy dołączyć do złącza **I2C KAmod Expander**, jak pokazano na **fotografii 4**. Lista pinów na złączu Pmod wraz z odpowiadającymi im pinami mikrokontrolera znajduje się w **tabeli 1**.

Tabela 1. Sygnały PmodHYGRO oraz odpowiadające im piny złącza KAmod Expander I²C i piny mikrokontrolera; N/C oznacza sygnały niepodłączone

Sygnał	Numer pinu PmodHYGRO	Numer pinu Kameleon Kamod Expander I²C	Pin mikrokontrolera
N/C	1	-	-
N/C	2	-	-
SCL	3	2	PF1
SDA	4	3	PF0
GND	5	4	-
VCC	6	1	-

Listing 1. Konfigurowanie I²C

```
void PmodHYGRO_Config()
{
    pmodHygroI2c.Instance = I2C2;
    pmodHygroI2c.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
    pmodHygroI2c.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
    pmodHygroI2c.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
    pmodHygroI2c.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
    pmodHygroI2c.Init.OwnAddress1 = 0x01;
    // Set the I2C_TIMINGR register to fulfill the HDC1080 timing
requirements (max 400kHz).
    pmodHygroI2c.Init.Timing = 0x10563046;
    HAL_I2C_Init(&pmodHygroI2c);
    // The HDC1080 requires at most 15 ms start-up delay.
    HAL_Delay(15);
}
```



Fotografia 5. PmodHYGRO podłączony do zestawu KAmeLeon

Komunikacja z HDC1080 odbywa się za pośrednictwem 16-bitowych rejestrów zapisywanych i odczytywanych za pomocą interfejsu I²C. Rejestr konfiguracyjny znajduje się pod adresem 0x02 i zawiera bity ustawiające rozdzielczość i tryb pomiaru. Rozdzielczość jest ustawiana dla każdej z wielkości osobno, natomiast tryb pomiaru dotyczy pomiaru jednej z dwóch wielkości, lub obu w sekwencji. Strukturę rejestru konfiguracyjnego pokazano na rysunku 1. Warto dodać, że w rejestrze znajduje się także możliwość włączenia, lub wyłączenia grzałki służącej do testowania pomiaru temperatury i usuwania wilgoci zgromadzonej wewnątrz urządzenia. Domyślna wartość rejestru po resecie wynosi 0x1000 i oznacza pomiar temperatury i wilgotności w sekwencji oraz maksymalną rozdzielczość – 14 bitów.

Obsługa modułu PmodHYGRO w przykładzie znajduje się w pliku PmodHYGRO.c. Funkcja PmodHYGRO_Config(), przedstawiona na **listingu 1**, jest odpowiedzialna za konfigurację interfejsu I²C. Ustawia ona m.in. wartość rejestru I2C_TIMINGR, odpowiedzialnego za konfigurację przebiegów czasowych na liniach SDA i SCL. Zawartość tego rejestru pokazano na **rysunku 5**. Pole PRESC jest preskalerem głównego przebiegu zegarowego taktującego moduł I²C mikrokontrolera,

Listing 2. Konfigurowanie GPIO

```
void HAL_I2C_MspInit(I2C_HandleTypeDef *hi2c)
{
    __HAL_RCC_I2C2_CLK_ENABLE();
    __HAL_RCC_GPIOF_CLK_ENABLE();
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C2;
    GPIO_InitStruct.Pin = GPIO_PIN_0 | GPIO_PIN_1;
    HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
}
```

Listing 3. Pomiar i odczyt wartości temperatury i wilgotności

```
void PmodHYGRO_GetMeasurements(int16_t* temperature, int16_t* humidity)
{
    // Write the address 0x00 pointing to temperature register in order to execute the measurements.
    uint8_t data[4] = {0};
    HAL_I2C_Master_Transmit(&pmodHygroI2c, PMODHYGRO_ADDRESS, data, 1, 100);
    // Wait for the measurements to complete.
    HAL_Delay(13);
    // Read the four bytes starting from address 0. First two bytes contain the 14-bit temperature
    // value, the last two bytes - humidity value.
    HAL_I2C_Master_Receive(&pmodHygroI2c, PMODHYGRO_ADDRESS, data, 4, 100);
    uint16_t temperatureRaw = (data[0] << 8) | data[1];
    uint16_t humidityRaw = (data[2] << 8) | data[3];
    // Calculate the real temperature and humidity values according to the formula described on the
    // website: https://reference.digilentinc.com/reference/pmod/pmodhygro/reference-manual
    float temp = (temperatureRaw * 1.0 / (1 << 16) * 165) - 40;
    *temperature = temp * 10;
    temp = (humidityRaw * 1.0 / (1 << 16)) * 100;
    *humidity = temp;
}
```

natomiast SCLL i SCLH ustalają czas trwania poziomu niskiego i wysokiego sygnału zegarowego. Pozostałe dwa pola dotyczą sygnału na linii danych SDA i oznaczają czas pomiędzy ustaleniem stanu na linii danych, a narastającym zboczem zegara (SCLDEL) oraz czas pomiędzy opadającym zboczem zegara, a zmianą stanu na linii danych (SDADEL). Faktyczne wartości wszystkich wymienionych parametrów poza SDADEL mają wartość o 1 większą niż ustawiona w rejestrze. Konfiguracja kończy się wywołaniem funkcji opóźniającej – HAL_Delay realizującej wymagane przez układ HDC1080 opóźnienie 15 ms w celu inicjalizacji urządzenia (*start-up time*).

Konfiguracja GPIO (piny PF0 i PF1) i włączenie zegara dla I²C znajduje się w funkcji HAL_I2C_MspInit, zamieszczonej na **listingu 2**. Jest ona wołana wewnątrz biblioteki HAL, w funkcji HAL_I2C_Init. Wartość rejestru konfiguracyjnego nie jest modyfikowana podczas konfiguracji, więc obowiązuje wartość domyślna omówiona wyżej.

Ostatnia z funkcji w pliku PmodHYGRO.c, pokazna na **listingu 3**, jest odpowiedzialna za wyzwalanie pomiaru i odczyt wartości. Pierwszym krokiem jest zapisanie adresu rejestru zawierającego zmierzoną wartość temperatury (adres 0x00). Jest to jednoznaczne z wyzwoleniem sekwencji dwóch pomiarów: temperatury i wilgotności, ze względu na ustawiony tryb pracy w rejestrze konfiguracyjnym.

Czas konwersji wynosi 6,5 ms dla pomiaru wilgotności i 6,35 ms dla pomiaru temperatury przy rozdzielczości 14 bitów. Z tego powodu, po wysłaniu adresu 0x00 jest wprowadzone opóźnienie 13 ms przed wykonaniem odczytu. Odczyt danych obejmuje 4 bajty, które można uzyskać w jednej transakcji I²C. Konwersja odczytanych danych na wartości temperatury i wilgotności jest zgodna z zależnościami przedstawionymi w dokumentacji układu HDC1080:

$$Temperatura[°C] = \left(\frac{Temperatura[15:0']}{2^{16}}\right)\cdot 165° - 40°C$$

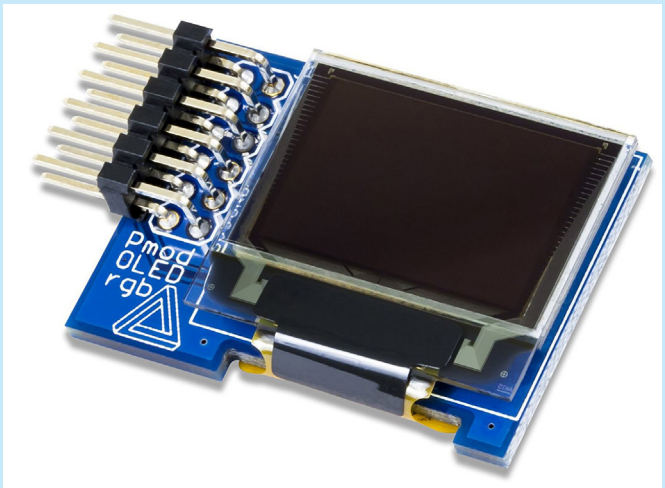
$$\text{Wilgotnosc wzgledna} = \left(\frac{\text{Wilgotnosc}[15:00]}{2^{16}}\right)\cdot 100\%RH$$

Funkcja PmodHYGRO_GetMeasurements jest wywoływana raz na sekundę w pętli głównej programu, a wartości temperatury i wilgotności są wypisywane za pośrednictwem portu szeregowego LPUART1, którego obsługa znajduje się w pliku serial.c.

PmodOLEDrgb

Moduł PmodOLEDrgb (**fotografia 6**) zawiera kolorowy wyświetlacz graficzny typu OLED o przekątnej 0.95 cala ze sterownikiem SSD1331. Wyświetlacz ma rozdzielczość 96x64 pikseli i 16-bitową głębię kolorów RGB (5:6:5). Zastosowany sterownik SSD1331 ma wewnętrzną pamięć mieszczącą (96×64×6) bitów i wsparcie sprzętowe, między innymi dla operacji czyszczenia ekranu, rysowania linii i prostokątów. Dzięki takiemu rozwiązaniu rysowanie podstawowych kształtów wymaga zdecydowanie mniejszej ilości danych przesyłanych do wyświetlacza w porównaniu do standardowego podejścia polegającego na rysowaniu piksel po pikselu.

PmodOLEDrgb zawiera 12-pinowe złącze z interfejsem SPI i dodatkowymi pinami sterującymi pracą modułu. Odpowiada ono złączu typu 2A z niepołączonym pinem MISO i pinami INT oraz RESET zamienionymi na sygnały kontrolne wyświetlacza. Ze względu na wykorzystanie pinów 9 i 10 na złączu, modułu nie można dołączyć do gniazda Pmod-SPI w zestawie KAmeLeon, dlatego na potrzeby przykładu



Fotografia 6. Wygląd modułu PmodOLEDrgb

PmodOLEDrgb został podłączony do złącza oznaczonego jako ARDUINO CONNECTOR. Rozmieszczenie pinów na złączu oraz sposób podłączenia do zestawu KAmeLeon umieszczono w **tabeli 2**.

Tabela 2. Podłączenie modułu PmodOLEDrgb do złącza ARDUINO zestawu KAmeLeon

Sygnał	Numer pinu PmodOLEDrgb	Numer pinu KAmeLeon ARDUINO CONNECTOR	Pin mikrokontrolera
CS	1	D10	PB12
MOSI	2	D11	PB15
N/C	3	-	-
SCK	4	D13	PB10
GND	5	GND	-
VCC	6	+3,3	-
D/C	7	D6	PD10
RES	8	D7	PB11
VCCEN	9	D8	PD11
PMODEN	10	D9	PB13
GND	11	-	-
VCC	12	-	-

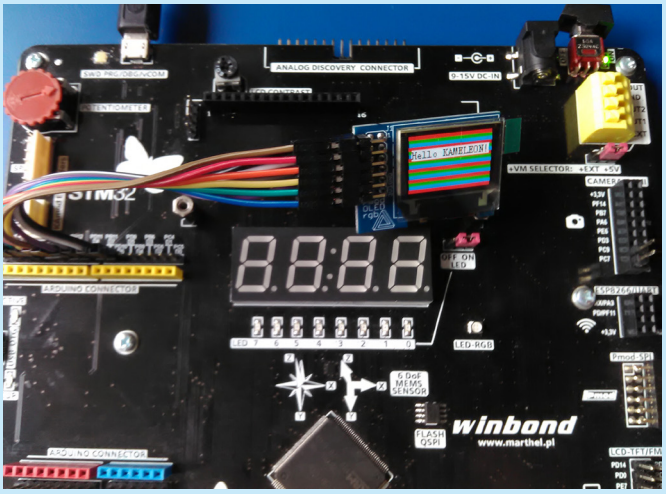
Poprzednie części kursu i dodatkowe materiały dostępne są na stronie www.media.avt.pl

Listing 4. Funkcja wyświetlająca znaki

```
void Oledrgb::DrawChar(uint8_t col, uint8_t row, char character, uint16_t color, uint16_t bgColor)
{
    uint8_t bitmap[FONT_WIDTH * FONT_HEIGHT * 2] = {0};
    for(int i = 0; i < FONT_WIDTH * FONT_HEIGHT; i++)
    {
        bitmap[2 * i] = (bgColor >> 8) & 0xFF;
        bitmap[2 * i + 1] = bgColor & 0xFF;
    }

    uint8_t charIndex = character - ' ';

    for(int i = 0; i < 6; i++) {
        for(int j = 0; j < FONT_HEIGHT; j++)
            if ((font[charIndex][i] << j) & 0x8000) {
                bitmap[2 * (i + (j * FONT_WIDTH))] = (color >> 8) & 0xFF;
                bitmap[2 * (i + (j * FONT_WIDTH)) + 1] = (color) & 0xFF;
            }
    }
    this->DrawBitmap(col, row, col + FONT_WIDTH - 1, row + FONT_HEIGHT - 1, bitmap);
}
```



Fotografia 7. Efekt uruchomienia przykładowego programu dla PmodOLEDrgb

Do obsługi wyświetlacza została wykorzystana biblioteka udostępniona przez firmę Digilent na stronie modułu (***http://bit.ly/2MGijYD***).

Znajdująca się na stronie wersja sterownika PmodOLEDrgb dla środowiska MPIDE została na potrzeby przykładu zmodyfikowana tak, aby współpracowała z biblioteką STM32Cube_FW_L4 i środowiskiem Atollic TrueSTUDIO. Źródła sterownika, znajdujące się w katalogu Drivers/OLEDrgb zostały napisane w języku C++, dlatego też przykładowy projekt został skonfigurowany dla tego języka.

W celu uruchomienia wyświetlacza należy utworzyć obiekt klasy Oledrgb, której interfejs znajduje się w pliku Drivers/OLEDrgb/OLEDrgb.h. Za konfigurację mikrokontrolera oraz inicjalizację wyświetlacza jest odpowiedzialna metoda Oledrgb::begin, wywołująca kolejno dwie inne metody: Oledrgb::OledrgbHostInit oraz Oledrgb::OledrgbDevInit. Pierwsza z nich konfiguruje SPI oraz wymienione w **tabeli 3** piny mikrokontrolera. Sygnał zegarowy SPI jest ustawiony w trybie 3 (Mode = 3, CPOL = 1, CPHA = 1), dane są 8-bitowe, a sygnał CS jest kontrolowany programowo. Druga z wymienionych metod konfiguruje wyświetlacz zgodnie z sekwencją opisaną w dokumentacji modułu dostępnej na stronie ***http://bit.ly/2KaHxQE***.

Tabela 3. Lista metod do obsługi wyświetlacza PmodOLEDrgb

Metoda	Opis
Oledrgb::DrawRectangle	Rysowanie prostokąta (wsparcie sprzętowe).
Oledrgb::DrawLine	Rysowania linii (wsparcie sprzętowe).
Oledrgb::DrawPixel	Rysowanie piksela.
Oledrgb::DrawBitmap	Rysowanie bitmapy.
Oledrgb::DrawString	Rysowanie napisu (dodane do oryginalnej biblioteki).
Oledrgb::DrawChar	Rysowanie znaku (dodane do oryginalnej biblioteki).

Po zakończonej konfiguracji dostępnych jest kilka metod umożliwiających pracę z wyświetlaczem. Są one wymienione w tab. 3. Do oryginalnej biblioteki zostały dodane dwie metody pozwalające na wyświetlanie pojedynczych znaków i całych napisów. Funkcja wyświetlająca znaki, pokazana na **listingu 4**, generuje lokalną bitmapę zawierającą znak na tle o kolorze zdefiniowanym w argumencie. Jest to optymalizacja zapobiegająca ustawianiu na wyświetlaczu pojedynczych pikseli, co znacząco wpływa na szybkość wyświetlania. Funkcja obsługuje czcionkę o nazwie *font* zdefiniowaną w pliku Drivers/OLEDrgb/Font.c.

W funkcji main przykładu generowana jest bitmapa zawierająca powtarzające się poziome paski w trzech głównych kolorach: czerwonym, zielonym i niebieskim. Na bitmapie wyświetlany jest napis „Hello KAMELEON!". Do generacji kodów kolorów wykorzystywana jest metoda Oledrgb::BuildRGB, która konwertuje trzy 8-bitowe wartości (RGB) na jedną liczbę 16-bitową. Efekt działania programu zaprezentowano na **fotografii 7**.

Krzysztof Chojnowski