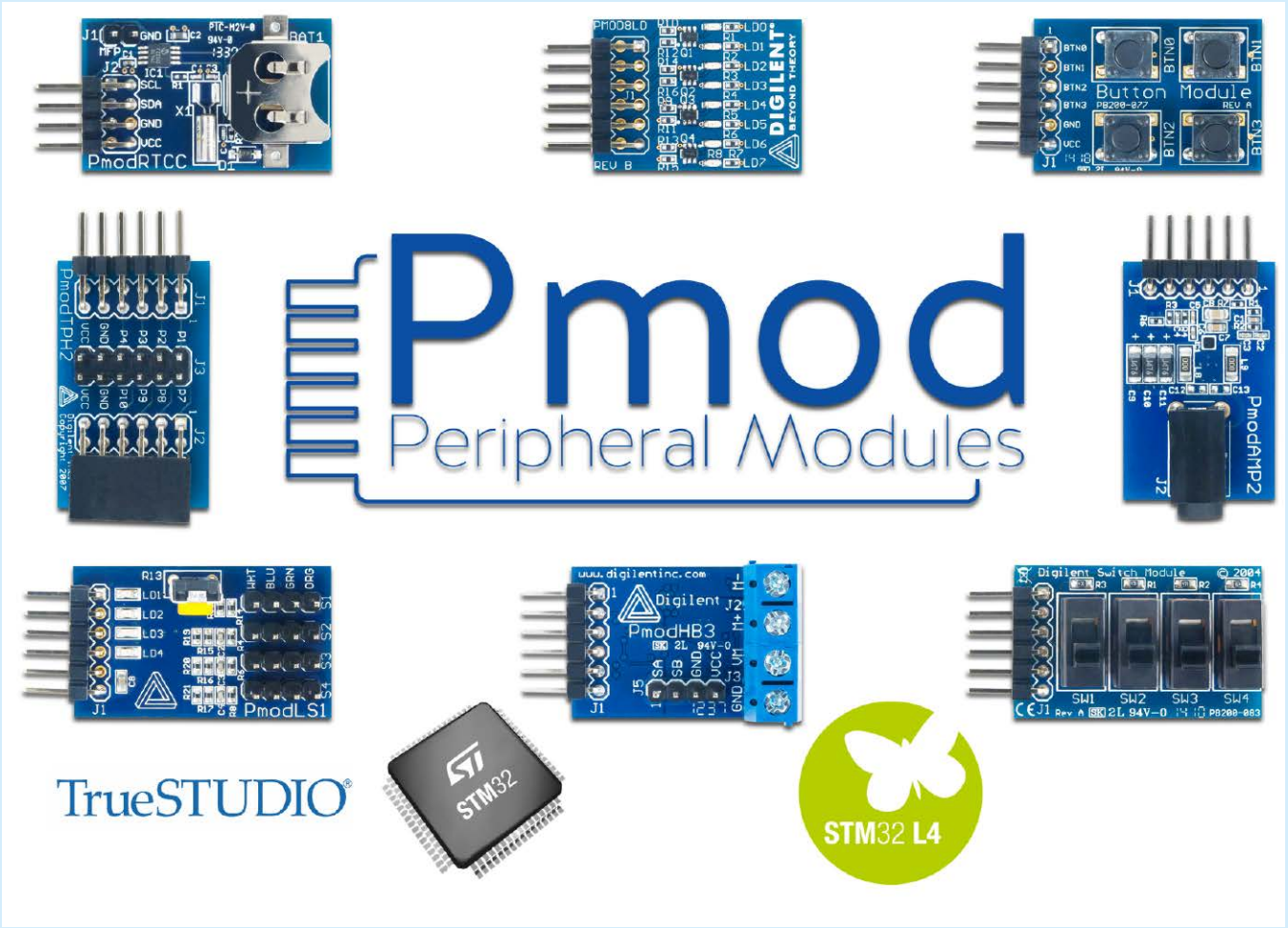


Digilent Pmod i STM32

Biblioteki do obsługi modułów peryferyjnych (1)

Standard interfejsów peryferyjnych firmy Digilent o nazwie Pmod zdobywa coraz szersze rynki, złącza z nim zgodne można spotkać na coraz większej liczbie płytek ewaluacyjnych z mikrokontrolerami, mikroprocesorami oraz układami FPGA. W cyklu artykułów przedstawimy biblioteki napisane w języku C dla środowiska Atollic TrueSTUDIO, które spełniają rolę „driverów" ułatwiających korzystanie z kilkunastu typów modułów Pmod z mikrokontrolerami STM32. Biblioteki były testowane na płytce KAmeLeon (www.kameleonboard.org) z użyciem bibliotek STM32Cube_FW_L4.

Pmod (***Peripheral module***) jest standardem złącza zaproponowanym przez firmę Digilent Inc. do komunikacji z układami peryferyjnymi. Występuje on w kilku odmianach zawierających linie GPIO i takie interfejsy komunikacyjne jak SPI, I²C, UART. W niniejszym artykule, będącym początkiem serii dotyczącej przykładów użycia modułów Pmod Digilent wraz z zestawem uruchomieniowym Kameleon, zostaną przedstawione wybrane konfiguracje złącza oraz PmodLED – moduł wykorzystujący linie GPIO do sterowania diodami.

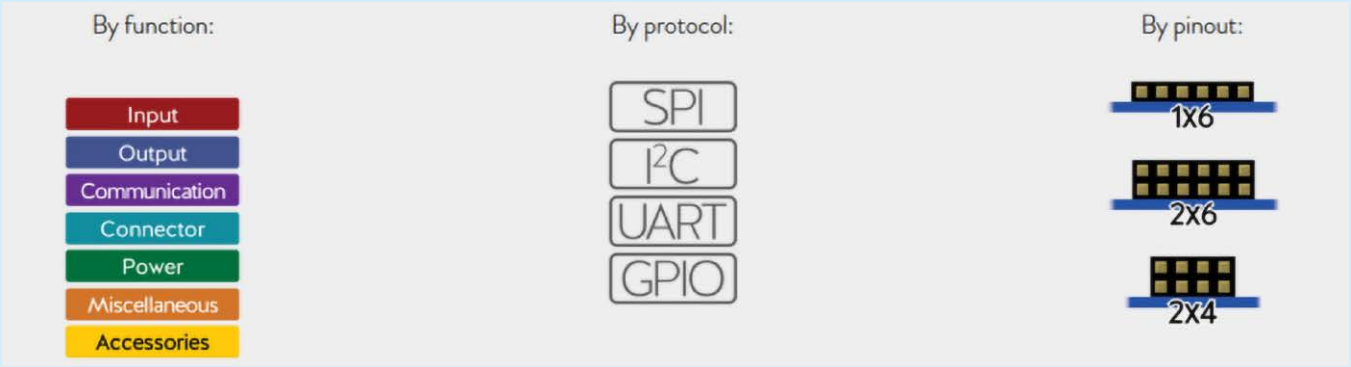
Standard Pmod

Standard interfejsu Pmod, opracowany i wdrożony na rynek przez firmę Digilent, opisuje złącza urządzeń peryferyjnych. Występują one w kilku wersjach różniących się liczbą linii sygnałowych i zasilania oraz typem interfejsu użytym do komunikacji (**rysunek 1**).

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

Wybrane wersje złączy zostały przedstawione w **tabelach 1...8**. Rola sygnałów oznaczonych N/S jest zależna od konkretnego modułu. Sygnały te mogą być wykorzystane jako dodatkowe linie sterujące lub źródła przerwań. Ponadto niektóre z sygnałów, np. CTS i RTS, mogą pełnić inne funkcje, opisane w dokumentacjach poszczególnych modułów. W przypadku interfejsu UART, wszystkie sygnały oznaczone są z punktu widzenia mikrokontrolera – TX

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



Rysunek 1. Najpopularniejsze typy złącz Pmod

Tabela 1. Interfejs I²C (6 pinów)

Sygnał	Numer pinu
N/S	1
N/S	2
SCL	3
SDA	4
GND	5
VCC	6

Tabela 2. Interfejs I²C (8 pinów)

Sygnał	Numer pinu		Sygnał
SCL	1	5	SCL
SDA	2	6	SDA
GND	3	7	GND
VCC	4	8	VCC

Tabela 3. Interfejs SPI (6 pinów) – Typ 2

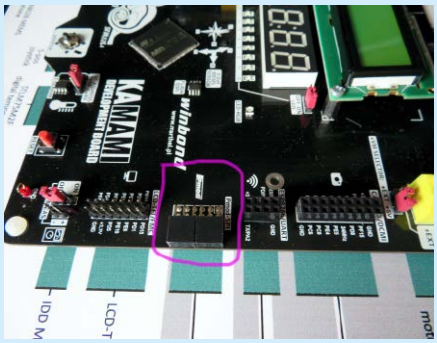
Sygnał	Numer pinu
SS	1
MOSI	2
MISO	3
SCK	4
GND	5
VCC	6

Tabela 4. Interfejs SPI (12 pinów) – Typ 2A

Sygnał	Numer pinu		Sygnał
SS	1	7	INT
MOSI	2	8	RESET
MISO	3	9	N/S
SCK	4	10	N/S
GND	5	11	GND
VCC	6	12	VCC

Tabela 5. Interfejs UART (6 pinów) – Typ 4

Sygnał	Numer pinu
CTS	1
TXD	2
RXD	3
RTS	4
GND	5
VCC	6



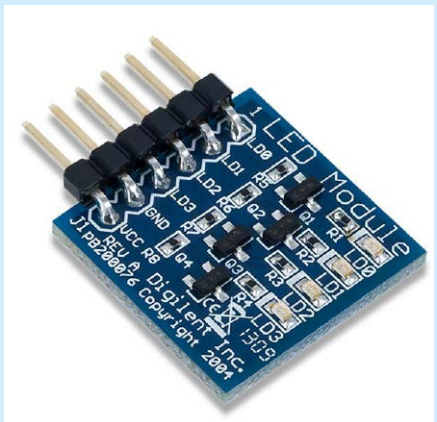
Fotografia 2. Złącze Pmod zestawu KAmeLeon, który jest platformą testową dla bibliotek opisanych w artykule

oznacza transmisję danych do modułu, a RX odbiór danych przez mikrokontroler.

W zestawie uruchomieniowym KAmeLeon znajduje się złącze Pmod z interfejsem SPI i sygnałami INT oraz Reset (**fotografia 2**). Konfiguracja ta odpowiada tabeli 4 bez połączonych pinów N/S (9 i 10). Listę wyprowadzeń mikrokontrolera STM32L496 z zestawu KAmeLeon podłączonych do złącza Pmod-SPI przedstawiono w **tabeli 9**.

Moduł PmodLED

Jako pierwszy przykład zostanie przedstawiony moduł PmodLED (**fotografia 3**). Zawiera on cztery LED kontrolowane za pośrednictwem sześciopinowego złącza typu 1 i włączane stanem wysokim na liniach GPIO. Na **fotografii 4** pokazano płytkę zestawu PmodLED dołączoną do KAmeLeona.



Fotografia 3. Wygląd modułu PmodLED

Tabela 6. Interfejs UART (12 pinów) – Typ 4A

Sygnał	Numer pinu		Sygnał
CTS	1	7	INT
TXD	2	8	RESET
RXD	3	9	N/S
RTS	4	10	N/S
GND	5	11	GND
VCC	6	12	VCC

Tab. 7. Interfejs GPIO (6 pinów) – Typ 1

Sygnał	Numer pinu
IO1	1
IO2	2
IO3	3
IO4	4
GND	5
VCC	6

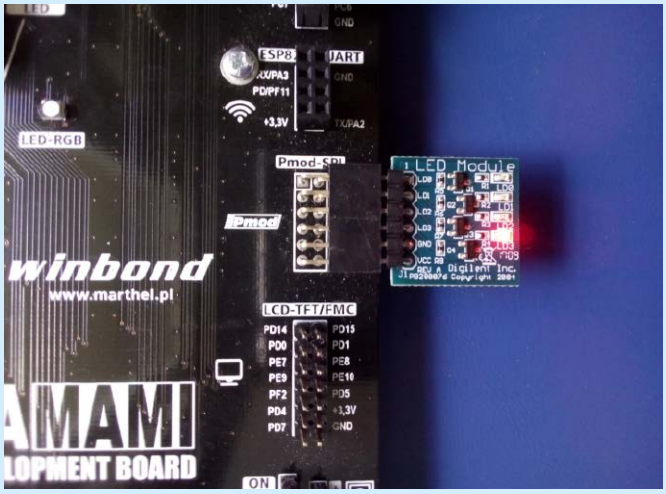
Aplikacja dla mikrokontrolera STM32L496ZG została napisana w oparciu o bibliotekę STM32Cube w środowisku

Tabela 8. Interfejs GPIO (12 pinów)

Sygnał	Numer pinu		Sygnał
IO1	1	7	IO5
IO2	2	8	IO6
IO3	3	9	IO7
IO4	4	10	IO8
GND	5	11	GND
VCC	6	12	VCC

Tabela 9. Złącze Pmod-SPI w zestawie KAmeLeon

Sygnał	Numer pinu		Sygnał
SS (PB0)	1	7	INT (PE12)
MOSI (PA7)	2	8	RESET (PE13)
MISO (PE14)	3	9	-
SCK (PA1)	4	10	-
GND	5	11	GND
VCC	6	12	VCC



Fotografia 4. Zestaw PmodLED dołączona do zestawu KAmeLeon

Atollic TrueSTUDIO. Obsługa modułu PmodLED znajduje się w pliku **PmodLED.c**. Znajduje się tam funkcja **PmodLED_Config** odpowiedzialna za konfigurację GPIO oraz dwie funkcje odpowiedzialne za zapalanie i gaszenie poszczególnych LEDów: **PmodLED_SetLed** i **PmodLED_ResetLed**. Funkcja **PmodLED_Config** została przedstawiona na **listingu 1**. Jej zadaniem jest konfiguracja wszystkich wymaganych przez moduł PmodLED linii GPIO.

Listing 1. Konfiguracja GPIO

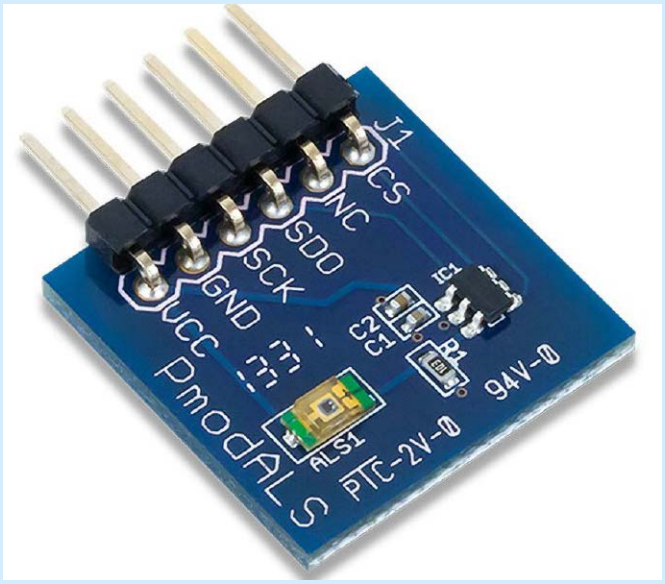
```
void PmodLED_Config(void)
{
    // Enable clock for all required GPIO ports.
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOE_CLK_ENABLE();
    // Initialize all GPIO as outputs.
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Mode  = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull  = GPIO_PULLUP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    GPIO_InitStruct.Pin = GPIO_PIN_1 | GPIO_PIN_7;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
    GPIO_InitStruct.Pin = GPIO_PIN_14;
    HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
    // Turn off all LEDs
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_14, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
}
```

Listing 2. Fragment funkcji PmodLED_ResetLed

```
void PmodLED_ResetLed(PmodLED_Led led)
{
    switch(led) {
    case PmodLED_Led0:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
        break;
    case PmodLED_Led1:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_RESET);
        break;
    case PmodLED_Led2:
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_14, GPIO_PIN_RESET);
        break;
    case PmodLED_Led3:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);
        break;
   }
}
```

Listing 3. Fragment funkcji PmodLED_SetLed

```
void PmodLED_SetLed(PmodLED_Led led)
{
    switch(led) {
    case PmodLED_Led0:
        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
        break;
    case PmodLED_Led1:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, GPIO_PIN_SET);
        break;
    case PmodLED_Led2:
        HAL_GPIO_WritePin(GPIOE, GPIO_PIN_14, GPIO_PIN_SET);
        break;
    case PmodLED_Led3:
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        break;
    }
}
```

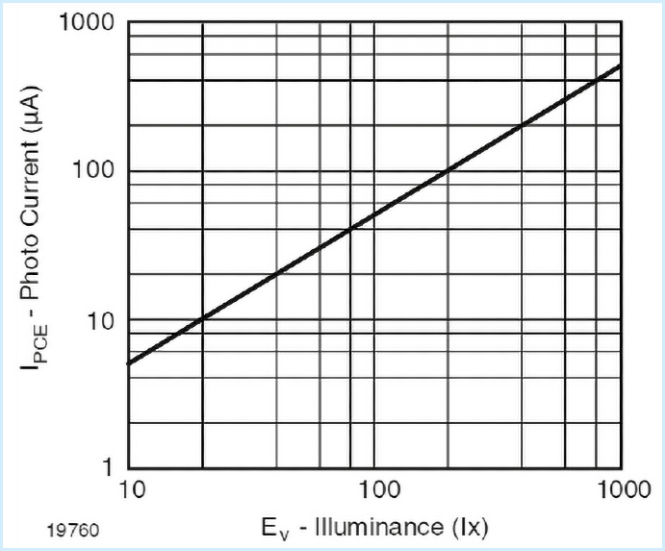


Fotografia 5. Wygląd zestawu PmodALS

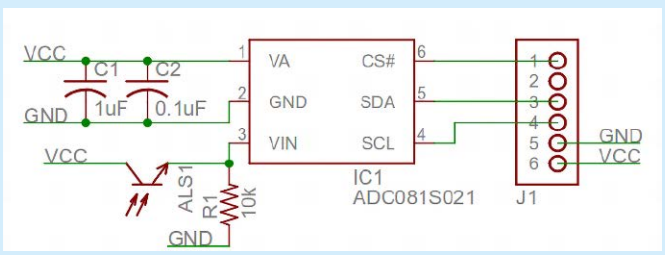
Funkcje PmodLED_SetLed i PmodLED_ResetLed, znajdujące się na **listingach 2** i **3**, pokazują zmianę stanu wyprowadzeń GPIO. Pokazane zostały tylko fragmenty obsługi jednej linii – PmodLED_Led0 (PB0) – pozostałe linie obsługiwane są w analogiczny sposób. Główna pętla aplikacji kolejno zaświeca i gasi wszystkie diody używając wyżej opisanych funkcji.

Moduł PmodALS

Moduł PmodALS (**fotografia 5**) zawiera dwa elementy umożliwiające mu pomiar natężenia światła widzialnego. Pierwszym z nich jest fototranzystor Vishay Semiconductor's TEMT6000X01 dokonujący konwersji natężenia światła na odpowiednią wartość



Rysunek 6. Charakterystyka konwersji światło/prąd kolektora fototranzystora w zestawie PmodALS



Rysunek 7. Układ pomiarowy zestawu PmodALS

Tabela 10. Interfejs SPI (6 pinów) – Typ 2

Sygnał	Numer pinu
CS (PB0)	1
MOSI	2
MISO (PE14)	3
SCK (PA1)	4
GND	5
VCC	6

prądu kolektora (**rysunek 6**). Fototranzystor działa w zakresie światła widzialnego (440 nm – 800 nm) z maksimum czułości przypadającym na długość fali wynoszącą 570 nm. Charakterystyka ta odpowiada czułości ludzkiego oka, dlatego czujnik ten dobrze nadaje się na przykład do kontrolowania jasności wyświetlaczy w zależności od poziomu światła otoczenia.

Drugim elementem wchodzącym w skład zestawu jest 8-bitowy przetwornik analogowo-cyfrowy Texas Instruments ADC081S021 umożliwiający odczyt napięcia proporcjonalnego do prądu kolektora fototranzystora, które odkłada się na rezystorze R1 (10 kW). Układ pomiarowy przedstawiono na **rysunku 7**.

Przetwornik ADC081S021 posiada interfejs SPI umożliwiający odczyt mierzonego napięcia. Posiada on dwa tryby działania – normalny (*Normal*) i wyłączony (*Shutdown*). Pierwszy z nich przeznaczony jest do wykonywania ciągłych konwersji bez oczekiwania na inicjalizację urządzenia przez każdym pomiarem. Aby urządzenie przełączyło się w tryb normalny, lub pozostało w nim po odczycie danych, sygnał *CS#*, którego zmiana na stan niski jest jednocześnie początkiem konwersji oraz odczytu danych, musi pozostać w stanie niskim po odczycie 16 bitów danych przez SPI, lub zmienić się na stan wysoki – drugi z tych warunków jest spełniony w przypadku zwykłego odczytu danych przez SPI. Urządzenie można wyłączyć zmieniając stan linii *CS#* na wysoki pomiędzy drugim i dziesiątym opadającym zboczem linii zegara *SCL*.

Aby odebrać dane z przetwornika należy odczytać 16 bitów danych. 8-bitowy wynik konwersji jest przesunięty o 4 bity w lewo. Należy pamiętać, że odczyt dotyczy zawsze konwersji rozpoczętej przez poprzednią zmianę sygnału *CS#*, dlatego do odczytania aktualnej wartości napięcia wymagane są dwie transakcje SPI, pomiędzy którymi musi upłynąć czas potrzebny do wykonania kolejnej konwersji (350 ns od trzynastego narastającego i 50 ns od szesnastego opadającego zbocza zegara).

Moduł PmodALS posiada złącze Pmod SPI typu 2 – jego opis znajduje się w **tabeli 10**. W nawiasach podano piny mikrokontrolera podłączone do poszczególnych sygnałów złącza (linia MISO nie jest używana).

Obsługa modułu PmodALS w przykładzie dla mikrokontrolera STM32L496ZG i zestawu KAmeLeon znajduje się w pliku *src/PmodALS.c*. Konfiguracja SPI wywoływana jest za pomocą funkcji *PmodALS_Config*, przedstawionej na **listingu 4**. Funkcja ta konfiguruje SPI1 w trybie tylko do odczytu (linia MISO) dla danych 16-bitowych. Funkcja biblioteczna *HAL_SPI_Init* wywołuje z kolei funkcję *HAL_SPI_MspInit*, gdzie konfigurowane są zegary peryferiów i linie GPIO. Funkcja ta pokazana została na **listingu 5**. Linie danych (MISO) i zegara SPI (SCK) są obsługiwane sprzętowo, natomiast linia CS jest sterowana programowo. Przetwornik ADC081S021 pracuje cały czas w trybie *Normal*, a odczyt danych zrealizowany jest jako zwykła transakcja SPI zaimplementowana w funkcji *PmodALS_GetValue* (**listing 6**). Dzięki temu sygnał CS jest zmieniany na stan wysoki po zakończeniu odczytu danych i przetwornik nie przechodzi do stanu *Shutdown*. Odczytana 16-bitowa wartość jest przesuwana w prawo o 4 bity, żeby otrzymać poziom natężenia światła z zakresu od 0 do 255.

W przykładzie dodatkowo konfigurowany jest Timer4 w trybie PWM, którego wyjście podłączone jest do pinu PD12, który z kolei steruje niebieską diodą LED-RGB (w zestawie KAmeLeon). Wypełnienie sygnału PWM jest zmieniane proporcjonalnie do wartości odczytanej przez przetwornik. Wartość ta jest także wypisywana na port szeregowy LPUART1 podłączony do programatora ST-Link i widoczny po podłączeniu do komputera jako wirtualny port szeregowy. Obsługa PWM i portu szeregowego jest zaimplementowana w plikach src/pwm.c i src/serial.c.

Krzysztof Chojnowski

Listing 4. Konfiguracja SPI

```
void PmodALS_Config(void)
{
  // Configure the SPI connected to the Pmod module. Only the MISO line is required.
    pmodAlsSpi.Instance = SPI1;
    pmodAlsSpi.Init.Mode = SPI_MODE_MASTER;
    pmodAlsSpi.Init.Direction = SPI_DIRECTION_2LINES_RXONLY;
    pmodAlsSpi.Init.DataSize = SPI_DATASIZE_16BIT;
    pmodAlsSpi.Init.CLKPolarity = SPI_POLARITY_HIGH;
    pmodAlsSpi.Init.CLKPhase = SPI_PHASE_1EDGE;
    pmodAlsSpi.Init.NSS = SPI_NSS_SOFT;
    pmodAlsSpi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
    pmodAlsSpi.Init.FirstBit = SPI_FIRSTBIT_MSB;
    pmodAlsSpi.Init.TIMode = SPI_TIMODE_DISABLE;
    pmodAlsSpi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    pmodAlsSpi.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
    HAL_SPI_Init(&pmodAlsSpi);
}
```

Listing 5. Konfiguracja zegarów peryferiów i linii GPIO

```
void HAL_SPI_MspInit(SPI_HandleTypeDef *hspi)
{
    // Initialize GPIO used by the SPI1 peripheral. The CS is controlled by the software (PB0 pin).
    __HAL_RCC_SPI1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOE_CLK_ENABLE();

    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
    GPIO_InitStruct.Pin = GPIO_PIN_1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    GPIO_InitStruct.Pin = GPIO_PIN_14;
    HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_PULLDOWN;
    GPIO_InitStruct.Pin = GPIO_PIN_0;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
}
```

Listing 6. Fragment funkcji PmodLED_SetLed

```
uint8_t PmodALS_GetValue(void)
{
    uint16_t value = 0;

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_SPI_Receive(&pmodAlsSpi, (uint8_t*)&value, 1, 100);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
    // The data obtained from the ADC is 16 bit, but the light
level value is only 8 bit.
    // It is shifted left by 4 bits.
    return (value >> 4);
}
```