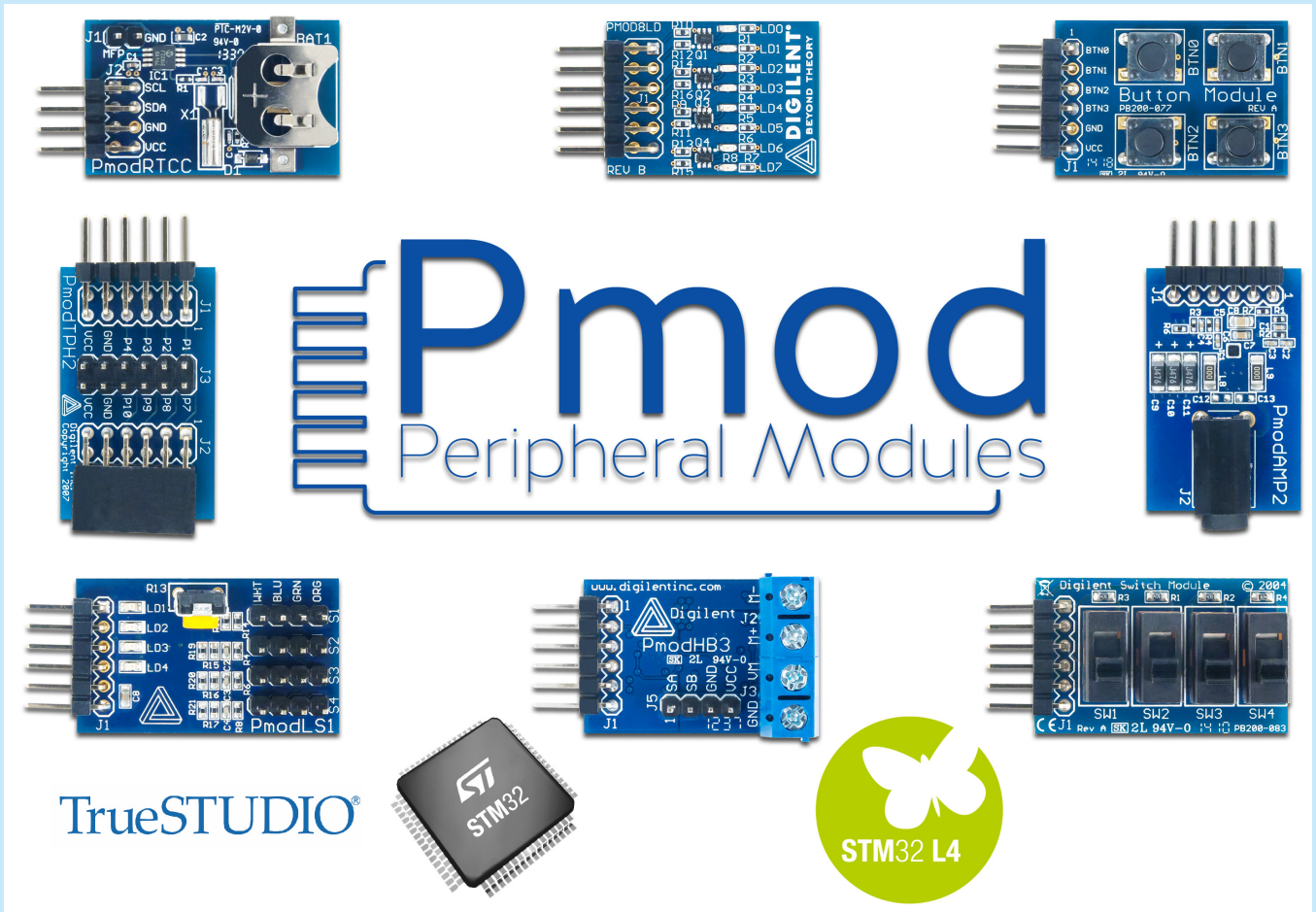
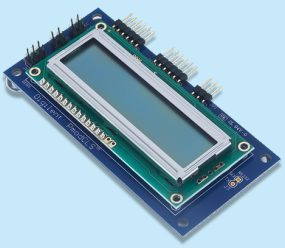


Digilent Pmod i STM32 (5)

*W kolejnej części cyklu poświęconego modułom peryferyjnym Pmod przedstawione zostaną trzy moduły z interfejsem SPI: PmodCLS z wyświetlaczem znakowym, PmodDPG1 z różnicowym czujnikiem ciśnienia i PmodISNS20 zawierający czujnik Halla do pomiaru natężenia prądu. Przykłady dla wymienionych modułów zostały przygotowane dla środowiska Atollic TrueSTUDIO i zestawu uruchomieniowego KAmeLeon (**www.kameleonboard.org**) z wykorzystaniem biblioteki STM32Cube_FW_L4.*

PmodCLS

Moduł PmodCLS (**fotografia 1**) ma wyświetlacz znakowy LCD o rozdzielczości 2 linie po 16 znaków. Do jego obsługi służy znajdujący się na płytce, zaprogramowany mikrokontroler ATmega48, przyjmujący komendy za pośrednictwem jednego z interfejsów szeregowych: SPI, I²C lub UART. Aktywny interfejs do komunikacji z modułem PmodCLS jest wybierany za pomocą zworek na złączu JP2 (MD0…MD2). W przykładzie użyto interfejsu SPI, dlatego zworka MD0 powinna być zamknięta, natomiast MD1 i MD2 należy zostawić otwarte. Trzeba również pamiętać o odpowiednim ustawieniu zworki na złączu JP1, aby sygnał SS (Slave Select) złącza J1 był doprowadzony do mikrokontrolera sterującego wyświetlaczem – piny 1 (SS) i 2 (środkowy) złącza JP1 powinny być zwarte. Po takiej konfiguracji modułu, złącze J1 odpowiadające złączu Pmod SPI typu 2. może być dołączone do gniazda Pmod-SPI zestawu Kameleon. Opis połączeń z pinami mikrokontrolera STM32L496ZG umieszczono w **tabeli 1**.



Fotografia 1. Wygląd modułu PmodCLS

Moduł PmodCLS jest kontrolowany za pomocą komend wysyłanych przez wybrany interfejs. Każda komenda rozpoczyna się sekwencją znaków 0x1b i 0x5b (ESC i '[' w kodzie ASCII), po której następują argumenty i kod komendy. Część z dostępnych komend, wykorzystanych w przykładzie znajduje się w **tabeli 2**. Komendy zostały poprzedzone sekwencją '\x1b['. Pełną listę komend można znaleźć na stronie z dokumentacją modułu: ***http://bit.ly/2CLDJ3W***.

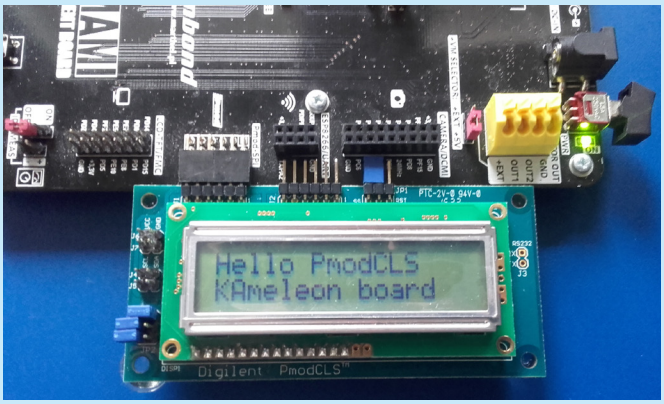
Tabela 1. Podłączenie modułu PmodCLS do złącza Pmod-SPI zestawu KAmeLeon

Sygnał	Numer pinu złącza Pmod	Pin mikrokontrolera
CS	1	PB0
MOSI	2	PA7
MISO	3	PE14
SCK	4	PA1
GND	5	-
VCC	6	-

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

Tabela 2. Przykładowe komendy modułu PmodCLS

Komenda	Opis
\x1b[j	Wyczyszczenie wyświetlacza i ustawienie kursora w pozycji zerowej
\x1b[0h	Ustawienie trybu wyświetlacza (0 – 16 znaków na linię, 1 – 40 znaków na linię)
\x1b[0c	Ustawienie trybu kursora (0 – niewidoczny, 1 – widoczny, 2 – mrugający)
\x1b[0;0H	Ustawienie pozycji kursora w pozycji 0;0 (<wiersz>;<kolumna>)



Fotografia 2. Moduł PmodCLS podłączony do zestawu KAmeLeon

Po ustawieniu żądanej konfiguracji i określeniu pozycji kursora można umieścić na wyświetlaczu napis. Łańcuch znaków należy wysłać bezpośrednio przez wybrany interfejs, bez sekwencji poprzedzającej, jak w wypadku komend. Kontroler rozpoznaje znaki w kodzie ASCII i umieszcza je na wyświetlaczu.

Obsługę wyświetlacza zaimplementowano w plikach **src/PmodCLS.h i src/PmodCLS.c** w postaci dwóch funkcji: konfiguracyjnej **PmodCLS_Config** i wypisującej tekst na wyświetlaczu **PmodCLS_Write**. Pierwsza z nich jest odpowiedzialna za inicjalizację modułów peryferyjnych i modułu PmodCLS. Do komunikacji z wyświetlaczem jest konfigurowany interfejs SPI w trybie 0 (polaryzacja i faza zegara równe 0) ze sprzętową obsługą pinu CS. Pełną konfigurację pokazano na **listingu 1**.

Piny GPIO są konfigurowane w funkcji HAL_SPI_MspInit, która jest wywoływana – zgodnie z konwencją biblioteki STM32Cube – podczas konfigurowania interfejsu SPI w funkcji HAL_SPI_Init. Wszystkie potrzebne piny mikrokontrolera są obsługiwane sprzętowo, zgodnie z konfiguracją zaprezentowaną na **listingu 2**. Na końcu funkcji PmodCLS_Config jest wysyłana sekwencja komend do kontrolera wyświetlacza odpowiedzialna za początkową inicjalizację modułu: wyczyszczenie wyświetlacza i ustawienie trybu kursora. Tę sekwencję zamieszczono na **listingu 3**. Użyta funkcja write jest wyłącznie wrapperem na biblioteczną funkcję HAL_SPI_Transmit realizującą blokującą transmisję SPI. Została ona także użyta w funkcji PmodCLS_Write przedstawionej na **listingu 4**. Przyjmuje ona numer linii (0 lub 1), łańcuch znaków i jego długość, ustawia kursor na początku wybranej linii oraz wysyła żądany napis do kontrolera wyświetlacza. W głównej funkcji programu – main – jest wywoływana funkcja konfigurująca wyświetlacz, a następnie wyświetlane są na nim komunikaty:

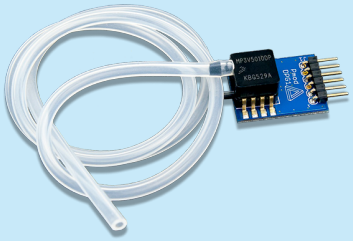
```
Hello PmodCLS
KAmeleon board
```

Efekt działania programu pokazano na **fotografii 2**.

PmodDPG1

Moduł PmodDPG1 (**fotografia 3**) składa się z trzech głównych elementów:

1. Różnicowego czujnika ciśnienia MP3V5010.
2. Przetwornika analogowo-cyfrowego ADCS7476.
3. Układu zapewniającego napięcie referencyjne do pomiarów REF3030.



Fotografia 3. Wygląd modułu PmodDPG1

Pierwszy z układów, MP3V5010, służy do pomiaru różnicy ciśnienia pomiędzy swoimi wejściami. Maksymalna różnica mierzonego ciśnienia to 10 kPa, natomiast rozdzielczość pomiarowa wynosi 270 mV/kPa. Czujnik ma wyjście analogowe, z którego można odczytać napięcie proporcjonalne do zmierzonej różnicy ciśnienia na wejściach układu.

Za pomiar napięcia na wyjściu czujnika ciśnienia jest odpowiedzialny 12-bitowy przetwornik analogowo-cyfrowy z interfejsem SPI typu ADCS7476. Rozpoczęcie konwersji jest inicjalizowane sygnałem CS, który musi być utrzymywany na poziomie niskim aż do końca konwersji następującego na szesnastym, opadającym zboczu sygnału zegarowego SCLK o maksymalnej częstotliwości

Listing 1. Konfiguracja interfejsu SPI do komunikacji z modułem PmodCLS

```
// Enable clock for all GPIO ports required for Pmod connector configuration.
__HAL_RCC_SPI1_CLK_ENABLE();
// Configure the SPI peripheral for 2 data lines and Chip Select pin managed by hardware.
pmodClsSpi.Instance = SPI1;
pmodClsSpi.Init.Mode = SPI_MODE_MASTER;
pmodClsSpi.Init.Direction = SPI_DIRECTION_2LINES;
pmodClsSpi.Init.DataSize = SPI_DATASIZE_8BIT;
pmodClsSpi.Init.CLKPolarity = SPI_POLARITY_LOW;
pmodClsSpi.Init.CLKPhase = SPI_PHASE_1EDGE;
pmodClsSpi.Init.NSS = SPI_NSS_HARD_OUTPUT;
pmodClsSpi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
pmodClsSpi.Init.FirstBit = SPI_FIRSTBIT_MSB;
pmodClsSpi.Init.TIMode = SPI_TIMODE_DISABLE;
pmodClsSpi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
pmodClsSpi.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
HAL_SPI_Init(&pmodClsSpi);
```

Listing 2. Konfiuracja pinów GPIO do komunikacji SPI z modułem PmodCLS

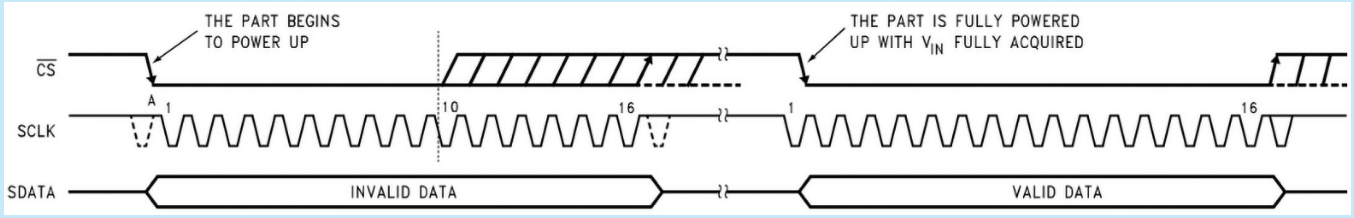
```
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_PULLDOWN;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI1;
GPIO_InitStruct.Pin = GPIO_PIN_1 | GPIO_PIN_7;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
GPIO_InitStruct.Pin = GPIO_PIN_14;
HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Pin = GPIO_PIN_0;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
```

Listing 3. Komendy odpowiedzialne na inicjalizację wyświetlacza

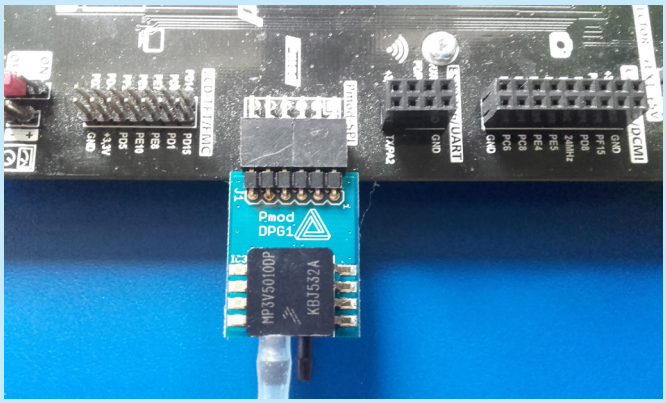
```
write("\x1b[j", 3);      // Clear the display and home cursor.
write("\x1b[0h", 4);     // Set the display mode (16 characters wrap)
write("\x1b[0c", 4);     // Set cursor mode (cursor off).
write("\x1b[0;0H", 6);   // Set the cursor position to 0,0.
```

Listing 4. Funkcja wypisujaca znaki na wyświetlaczu

```
void PmodCLS_Write(uint8_t line, char* text, uint32_t len)
{
    // Verify the arguments according to the defined limits.
    if(line >= MAX_LINES || len > MAX_LINE_CHARACTERS) return;
    // Set the cursor position to the beginning of the selected
line..
    if(line == 0) write("\x1b[0;0H", 6);
    else write("\x1b[1;0H", 6);
    write(text, len);
}
```



Rysunek 4. Transfery SPI potrzebne do pomiaru i odczytu danych (źródło: dokumentacja ADCS7476)



Fotografia 5. Moduł PmodDPG1 dołączony do płytki KAmeLeon

20 MHz. Zmiana poziomu sygnału CS na wysoki pomiędzy dziesiątym a szesnastym zboczem sygnału zegarowego powoduje, że układ pozostaje w tym samym trybie, jednak wówczas dane są nieważne. Układ ADCS7476 ma także tryb obniżonego poboru prądu, do którego przechodzi się zmieniając poziom sygnału CS na wysoki przed dziesiątym opadającym zboczem sygnału zegarowego SCLK. Do pomiaru i odczytu wartości napięcia potrzebne są dwa 16-bitowe transfery SPI – pierwszy do przeprowadzenia konwersji, a drugi do transmisji danych, które są przesunięte do prawej strony (najmniej znaczące 12 bitów w słowie 16-bitowym), co pokazano na **rysunku 4**.

Moduł PmodDPG1 należy dołączyć do złącza Pmod-SPI zestawu Kameleon, jak pokazano na **fotografii 5**. Połączenia z pinami mikrokontrolera są identyczne, jak w wypadku PmodCLS w tabeli 1.

W przykładzie kod do obsługi modułu PmodDPG1 znajduje się w pliku src/PmodDPG1.c. Pierwsza z funkcji – PmodDPG1_Config, przedstawiona na **listingu 5**, konfiguruje interfejs SPI do komunikacji z układem. SPI jest w trybie 2 (CPOL=1, CPHA=0) z 16-bitowymi danymi. Interfejs jest skonfigurowany dla transmisji w jedną stronę – bez linii MOSI, ze sprzętową kontrolą sygnału CS. Włączanie zegarów i pinów znajduje się w funkcji HAL_SPI_MspInit, która jest wywoływana w funkcji HAL_SPI_Init. Funkcja HAL_SPI_MspInit jest przedstawiona na **listingu 6**. Wszystkie piny są kontrolowane sprzętowo i konfigurowane w funkcji alternatywnej dla SPI1. Ostatnia z funkcji obsługi modułu PmodDPG1 zajmuje się odczytem i przeliczaniem wartości mierzonej. Pokazano ją na **listingu 7**. Konwersja jest przeprowadzona zgodnie z zależnością zamieszczoną w dokumentacji modułu PmodDPG1 i liczoną wg wzoru:

```
P [kPa] = (wartość zmierzona ADC/4096-0,08)/0,09
```

W przykładzie różnica ciśnienia jest odczytywana cyklicznie co 0,5 sekundy w pętli głównej programu, w funkcji main. Po odczycie wyniki są wysyłane przez interfejs szeregowy LPUART1, którego obsługa znajduje się w pliku src/serial.c.

PmodISNS20

Ostatnim z prezentowanych w artykule modułów jest PmodISNS20 (**fotografia 6**), czyli moduł do pomiaru prądu oparty o czujnik ACS722. Czujnik ten jest w stanie mierzyć prąd stały lub zmienny o maksymalnym natężeniu 20 A. Układ ma wyjście analogowe, z którego można odczytać napięcie proporcjonalne do mierzonego prądu. Czułość pomiaru wynosi 66 mV/A, a maksymalny błąd pomiaru 3%. Układ ACS722 umożliwia wybór pasma sygnału wyjściowego: 80 kHz lub 20 kHz. Jego zmiana jest możliwa za pomocą zworki JP2. Dodatkowo, moduł PmodISNS20 pozwala na włączenie filtra analogowego ograniczającego pasmo do 120 Hz – służy do tego zworka JP1. Napięcie wyjściowe jest mierzone przez 12-bitowy przetwornik analogowo-cyfrowy typu ADCS7476 z interfejsem SPI, który omówiono przy okazji opisu modułu PmodDPG1.

Podobnie jak w wypadku dwóch poprzednich modułów omówionych w artykule, PmodISNS20 powinien zostać przyłączony do złącza Pmod-SPI znajdującego się na płytce KAmeLeon, jak

Listing 5. Inicjalizacja interfejsu SPI dla PmodDPG1

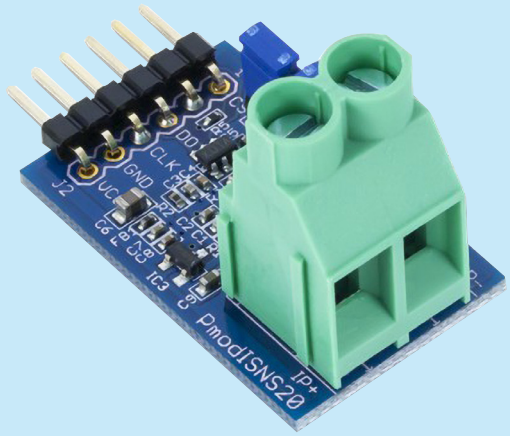
```
SPI_HandleTypeDef pmodDpg1Spi;
void PmodDPG1_Config(void)
{
    // Configure the SPI connected to the Pmod module. Only the MISO line is required.
    // The CS line is controlled by the hardware.
    pmodDpg1Spi.Instance = SPI1;
    pmodDpg1Spi.Init.Mode = SPI_MODE_MASTER;
    pmodDpg1Spi.Init.Direction = SPI_DIRECTION_2LINES_RXONLY;
    pmodDpg1Spi.Init.DataSize = SPI_DATASIZE_16BIT;
    pmodDpg1Spi.Init.CLKPolarity = SPI_POLARITY_HIGH;
    pmodDpg1Spi.Init.CLKPhase = SPI_PHASE_1EDGE;
    pmodDpg1Spi.Init.NSS = SPI_NSS_HARD_OUTPUT;
    pmodDpg1Spi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
    pmodDpg1Spi.Init.FirstBit = SPI_FIRSTBIT_MSB;
    pmodDpg1Spi.Init.TIMode = SPI_TIMODE_DISABLE;
    pmodDpg1Spi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    pmodDpg1Spi.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
    HAL_SPI_Init(&pmodDpg1Spi);
}
```

Listing 6. Inicjalizacja pinów interfejsu SPI dla PmodDPG1

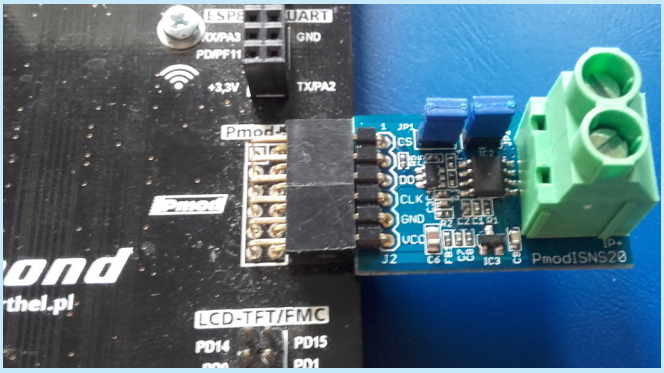
```
void HAL_SPI_MspInit(SPI_HandleTypeDef *hspi)
{
    // Initialize GPIO used by the SPI2 peripheral.
    __HAL_RCC_SPI1_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOE_CLK_ENABLE();
    GPIO_InitTypeDef GPIO_InitStruct;
    // Configure all required GPIO lines in alternate mode.
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull = GPIO_PULLUP;
    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.Pin = GPIO_PIN_0;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

Listing 7. Odczyt i konwersja wartości różnicy ciśnienia

```
uint32_t PmodDPG1_GetValue(void)
{
    /* Receive the 12-bit value from the ADC and convert it to the pressure difference in [Pa]. The conversion equation is taken from the PmodDPG1 reference manual. */
    uint16_t value[2] = {0};
    HAL_SPI_Receive(&pmodDpg1Spi, (uint8_t*)value, 2, 100);
    value[1] = ((value[1] * 1000 / 4096) - 80) / 0.09;
    return value[1];
}
```



Fotografia 6. Wygląd modułu PmodISNS20



Fotografia 7. Moduł PmodISNS20 dołączony do płytki KAmeLeon

na **fotografii 7**. Połączenia pinów mikrokontrolera z pinami modułu są zgodne z wymienionymi w tabeli 1.

Konfiguracja SPI dla ADCS7476, przedstawiona na **listingu 8** i zaimplementowana w pliku src/PmodISNS20.c, wygląda podobnie, jak w wypadku modułu PmodDPG1, jednak ten listing zawiera również programową kontrolę linii CS. Ze względu na różnicę w konfiguracji SPI, zmianie musi też ulec konfiguracja GPIO, w której pin PB0, odpowiadający linii CS, jest konfigurowany jako wyjście z rezystorem podciągającym. Jest to widoczne na **listingu 9**.

Ostatnia z funkcji, pokazana na **listingu 10**, odczytuje wartość napięcia z przetwornika analogowo-cyfrowego oraz dokonuje jego konwersji na natężenie prądu mierzone przez czujnik ACS722. Transmisja przez SPI musi być wykonana dwukrotnie, ponieważ za pierwszym razem przetwornik dokonuje konwersji, a za drugim jest odczytywana zmierzona wartość. Konwersja na natężenie prądu została zaimplementowana zgodne z zależnością ze wzoru:

Listing 8. Inicjalizacja interfejsu SPI dla PmodISNS20
```
SPI_HandleTypeDef pmodIsns20Spi;
void PmodISNS20_Config(void)
{
    /* Configure the SPI connected to the Pmod module. Only the
    MISO line is required. */
    pmodIsns20Spi.Instance = SPI1;
    pmodIsns20Spi.Init.Mode = SPI_MODE_MASTER;
    pmodIsns20Spi.Init.Direction = SPI_DIRECTION_2LINES_RXONLY;
    pmodIsns20Spi.Init.DataSize = SPI_DATASIZE_16BIT;
    pmodIsns20Spi.Init.CLKPolarity = SPI_POLARITY_HIGH;
    pmodIsns20Spi.Init.CLKPhase = SPI_PHASE_1EDGE;
    pmodIsns20Spi.Init.NSS = SPI_NSS_SOFT;
    pmodIsns20Spi.Init.BaudRatePrescaler = SPI_
BAUDRATEPRESCALER_64;
    pmodIsns20Spi.Init.FirstBit = SPI_FIRSTBIT_MSB;
    pmodIsns20Spi.Init.TIMode = SPI_TIMODE_DISABLE;
    pmodIsns20Spi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
    pmodIsns20Spi.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
    HAL_SPI_Init(&pmodIsns20Spi);
}
```

Listing 9. Inicjalizacja pinów interfejsu SPI dla PmodISNS20
```
/* Initialize GPIO used by the SPI2 peripheral. The CS is
configured
   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_PULLUP;
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.Pin = GPIO_PIN_0;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
```

Listing 10. Odczyt i konwersja wartości różnicy ciśnienia
```
int32_t PmodISNS20_GetValue(void)
{
    uint16_t adcValue = 0;

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_SPI_Receive(&pmodIsns20Spi, (uint8_t*)&adcValue, 1, 100);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET);
    HAL_SPI_Receive(&pmodIsns20Spi, (uint8_t*)&adcValue, 1, 100);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);
    /* The current value in [mA] is calculated according to the
formula
    given in the PmodISNS20 reference manual. */
    //return (int32_t)(((int16_t)adcValue - 2048) * 1000.0 /
89.95);
}
```

`I [mA] = 1000/89.95*(wartość zmierzona przez ADC -2048)`

Przykładowy program dokonuje pomiaru natężenia prądu co 0,5 sekundy, a zmierzoną wartość przesyła przez interfejs szeregowy LPUART1 obsługiwany przez funkcje zaimplementowane w pliku src/serial.c.

Krzysztof Chojnowski