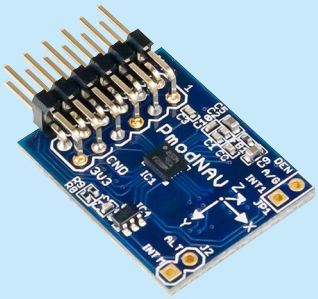
Digilent Pmod i STM32 (9)

W dziewiątym, przedostatnim już, odcinku cyklu poświęconego modułom Pmod firmy Digilent przedstawione zostaną: PmodNAV z akcelerometrem, magnetometrem, żyroskopem i barometrem, PmodPMON1 służący do monitorowania mocy oraz PmodTMP2 z czujnikiem temperatury.

Przykłady przedstawione w niniejszym artykule zostały przygotowane dla środowiska Atollic TrueSTUDIO i zestawu uruchomieniowego KAmeLeon (www.kameleonboard.org) z wykorzystaniem biblioteki STM32Cube_FW_L4.

Moduł PmodNAV (**fotografia 1**) umożliwia wykonywanie pomiarów ruchu w dziewięciu osiach (3-osiowy akcelerometr, 3-osiowy żyroskop, 3-osiowy magnetometr) wraz z ciśnieniem atmosferycznym. Jego sercem są dwa układy firmy STMicroelectronics: LSM9DS1 i LPS25HB.



Fotografia 1. Wygląd modułu PmodNAV

Zakresy pomiarowe poszczególnych wielkości są konfigurowalne i wynoszą:

- akcelerometr: ±2/±4/±8/±16 g,
- żyroskop: ±245/±500/±2000 dps,
- magnetometr: ±4/±8/±12/±16 Gauss,
- barometr: 260-1260 hPa.

Rozdzielczość pomiaru ciśnienia wynosi 4096 LSB/hPa, natomiast dla pozostałych wielkości jest zależna od wybranego zakresu, a jej maksymalna wartość (dla najmniejszego zakresu) wynosi odpowiednio: 0,061 mg/LSB, 8,75 mdps/LSB i 0,14 mGauss/LSB.

Układ LSM9DS1 umożliwia generację przerwań po przekroczeniu ustawionych progów mierzonych wielkości na każdej z osi, po wykryciu aktywności oraz wówczas, gdy dostępne są nowe dane pomiarowe. Dodatkowo akcelerometr i żyroskop mogą zostać skonfigurowane w trybie FIFO, w którym dane pomiarowe obu czujników mogą być buforowane w wewnętrznej kolejce. Do jej obsługi mogą być użyte przerwania informujące przekroczeniu zadanej liczby elementów. Czujnik ciśnienia LPS25HB charakteryzuje się podobną funkcjonalnością, pozwalającą na konfigurację progów do generacji przerwań. Posiada on także wewnętrzną kolejkę FIFO do buforowania danych pomiarowych.

Oba układy mogą być obsługiwane za pośrednictwem interfejsów SPI oraz I²C, jednak w module PmodNAV udostępniony jest tylko pierwszy z nich. Znajduje się on na złączu Pmod typu 2A (J1), w którym oprócz linii danych oraz zegara znajdują są trzy sygnały wybierające urządzenie (akcelerometr/żyroskop, magnetometr, barometr), wspólna linia przerwań i sygnał gotowości danych z magnetometru. Dwa ostatnie sygnały nie są używane w przykładzie, jednak ze względu na rozmieszczenie pozostałych sygnałów na złączu nie jest możliwe podłączenie modułu PmodNAV do złącza Pmod-SPI na płytce KAmeLeon. Z tego względu moduł został podłączony do złącza ARDUINO według **tabeli 1** i **fotografii 2**.

Pliki do projektu opisanego w artykule są dostępne pod adresem **http://kameleonboard.org**

Przykładowy kod do obsługi modułu PmodNAV umieszczono w plikach PmodNAV.c i PmodNAV.h. Pierwsza z funkcji – PmodNAV_Config, znajdująca się na **listingu 1**, jest odpowiedzialna za konfigurację interfejsu SPI w trybie 3 (CPOL=1, CPHA=1) z programową kontrolą sygnału CS. Dodatkowo uruchamiane są ciągłe pomiary we wszystkich czujnikach:

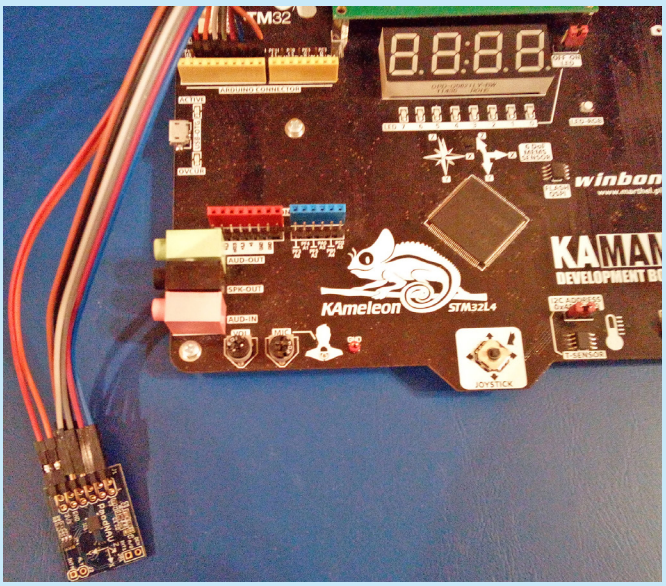
- akcelerometr: 10 Hz,
- żyroskop: 14,9 Hz,
- magnetometr: 10 Hz (wartość domyślna),
- barometr: 12,5 Hz.

Linie GPIO są ustawiane w funkcji HAL_SPI_MspInit, która jest wywoływana przez bibliotekę STM32Cube wewnątrz funkcji HAL_SPI_Init. Piny podłączone do linii danych (MISO oraz MOSI) i zegara (SCLK) są konfigurowane jako funkcja alternatywna dla SPI2, natomiast wszystkie linie CS są ustawiane jako zwykłe wyjścia i będą kontrolowane programowo. Funkcja ta została przedstawiona na **listingu 2**.

Do komunikacji z czujnikami modułu PmodNAV zostały przygotowane dwie funkcje pomocnicze: writeRegister i readRegister, pokazane na **listingu 3**. Realizują one zapis i odczyt rejestru o podanym

Tabela 1. Sygnały PmodNAV oraz odpowiadające im piny złącza ARDUINO i mikrokontrolera; w tabeli pominięto sygnały nieużywane w przykładzie (INT i DRDY) i linie zasilania występujące na złączu Pmod

Sygnał	Numer pinu PmodNAV	Numer pinu Kameleon ARDUINO CONNECTOR	Pin mikrokontrolera
CS_A/G	1	D10	PB12
MOSI	2	D11	PB15
MISO	3	D12	PB14
SCLK	4	D13	PB10
CS_M	9	D9	PB13
CS_ALT	10	D8	PD11



Fotografia 2. Moduł PmodNAV przyłączony do zestawu KAmeLeon

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

adresie. Wszystkie czujniki wymagają podania kierunku transmisji na najstarszym bicie adresu (1 – odczyt, 0 – zapis), a każda transakcja musi składać się z co najmniej dwóch bajtów. Opisywane funkcje operują na pojedynczych rejestrach, dlatego zawsze wykorzystywane są dwa bajty – adres oraz wartość rejestru.

Listing 1. Konfiguracja interfejsu SPI2 i konfiguracja modułu PmodNAV

```
void PmodNAV_Config(void)
{
// Configure the SPI connected to the Pmod module.
  pmodNavSpi.Instance = SPI2;
  pmodNavSpi.Init.Mode = SPI_MODE_MASTER;
  pmodNavSpi.Init.Direction = SPI_DIRECTION_2LINES;
  pmodNavSpi.Init.DataSize = SPI_DATASIZE_8BIT;
  pmodNavSpi.Init.CLKPolarity = SPI_POLARITY_HIGH;
  pmodNavSpi.Init.CLKPhase = SPI_PHASE_2EDGE;
  pmodNavSpi.Init.NSS = SPI_NSS_SOFT;
  pmodNavSpi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
  pmodNavSpi.Init.FirstBit = SPI_FIRSTBIT_MSB;
  pmodNavSpi.Init.TIMode = SPI_TIMODE_DISABLE;
  pmodNavSpi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  pmodNavSpi.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;

  HAL_SPI_Init(&pmodNavSpi);
// Enable the continuous conversion mode for the magnetometer.
  writeRegisterM(0x22, 0x00);
// Enable the gyroscope measurements with the  14.9Hz data rate.
  writeRegisterAG(0x10, 0x20);
// Enable the accelerometer measurements with the  10Hz data rate.
  writeRegisterAG(0x20, 0x20);
// Enable the pressure measurements with 12.5 Hz data rate.
  writeRegisterALT(0x20, 0xB0);
}
```

Listing 2. Konfiguracja linii GPIO dla modułu PmodNAV

```
void HAL_SPI_MspInit(SPI_HandleTypeDef *hspi)
{
  __HAL_RCC_SPI2_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_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_SPI2;
  GPIO_InitStruct.Pin = GPIO_PIN_10 | GPIO_PIN_14 | GPIO_PIN_15;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  // CS_A/G - PB12
  GPIO_InitStruct.Pin = GPIO_PIN_12;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
  // CS_M - PB13
  GPIO_InitStruct.Pin = GPIO_PIN_13;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_13, GPIO_PIN_SET);
  // CS_ALT - PD11
  GPIO_InitStruct.Pin = GPIO_PIN_11;
  HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_11, GPIO_PIN_SET);
}
```

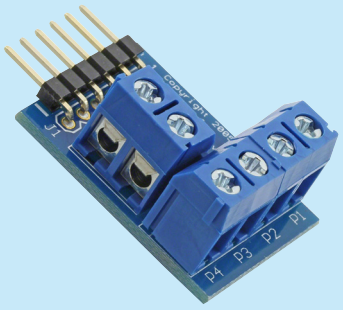
Listing 3. Zapis i odczyt rejestru układów LSM9DS1 i LPS25HB

```
#define SPI_READ_FLAG   0x80
#define SPI_WRITE_FLAG  0x00

static void writeRegister(uint8_t address, uint8_t data)
{
  uint8_t txbuf[2] = {address | SPI_WRITE_FLAG, data};
  HAL_SPI_Transmit(&pmodNavSpi, txbuf, 2, 100);
}

static uint8_t readRegister(uint8_t address)
{
  uint8_t txbuf[2] = {address | SPI_READ_FLAG, 0x00};
  uint8_t rxbuf[2] = {0x00};
  HAL_SPI_TransmitReceive(&pmodNavSpi, txbuf, rxbuf, 2, 100);
  return rxbuf[1];
}
```

Listing 4. Zapis i odczyt rejestru akcelerometru i żyroskopu

```
static void writeRegisterAG(uint8_t address, uint8_t data)
{
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
  writeRegister(address, data);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
}

static uint8_t readRegisterAG(uint8_t address)
{
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
  uint8_t data = readRegister(address);
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
  return data;
}
```

Opisane wyżej funkcje są używane przez zestaw innych funkcji, realizujących zapis i odczyt rejestrów poszczególnych układów. Obsługują one dodatkowo odpowiednie sygnały CS, co zostało przedstawione na **listingu 4**. Pozostałe zestawy funkcji obsługujące magnetometr i barometr wyglądają analogicznie.



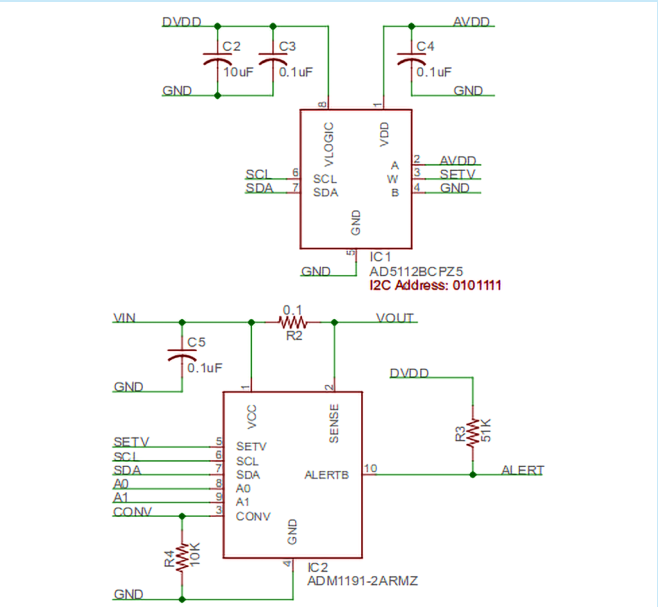
Fotografia 3. Wygląd modułu PmodPMON1

Pozostałe funkcje zaimplementowane w pliku PmodNAV.c są odpowiedzialne za konwersję odczytanych wartości z czujników na wartości wyrażone w odpowiednich jednostkach:

- akcelerometr [mg],
- żyroskop [mdps],
- magnetometr [mG],
- barometr [hPa].

Jako przykład, na **listingu 5** została przedstawiona funkcja konwertująca dane pomiarowe z akcelerometru, w której konwersja jest przeprowadzana dla zakresu ±2 g i rozdzielczości 0,061 mg/LSB. Funkcje obsługujące pozostałe czujniki zostały zaimplementowany w analogiczny sposób.

Główna funkcja przykładu – main, wykonuje konfigurację modułu PmodNAV, a następnie cyklicznie odczytuje wskazania wszystkich dostępnych czujników. Wyniki są wysyłane na port szeregowy LPUART1, dostępny jako wirtualny port szeregowy po podłączeniu KAmeLeona do komputera. Obsługa portu znajduje się w plikach serial.c i serial.h.



Rysunek 4. Połączenie układów ADM119 i AD512 w module PmodPMON1 (źródło: dokumentacja modułu PmodPMON1)

Listing 5. Konwersja wyników pomiaru z akcelerometru

```
void PmodNAV_ReadAcc(int32_t* x, int32_t* y, int32_t* z)
{
  *x = (int16_t)((readRegisterAG(0x29) << 8) |
readRegisterAG(0x28));
  *y = (int16_t)((readRegisterAG(0x2B) << 8) |
readRegisterAG(0x2A));
  *z = (int16_t)((readRegisterAG(0x2D) << 8) |
readRegisterAG(0x2C));

  *x = (*x * 1.0) * 0.061;
  *y = (*y * 1.0) * 0.061;
  *z = (*z * 1.0) * 0.061;
}
```

Tabela 2. Sygnały PmodPMON1 oraz odpowiadające im piny złącz ARDUINO, I²C Expander oraz wyprowadzenia mikrokontrolera; w tabeli pominięto linie zasilania występujące na złączu Pmod

Sygnał	Numer pinu PmodPMON1	Pin ARDUINO CONNECTOR	Pinu I²C Expander	Pin mikrokontrolera
SCL	1 (J2)	-	2	PF1
SDA	2 (J2)	-	3	PF0
CONV	1 (J1)	D15	-	PF14
ALERT	3 (J1)	D14	-	PF15

Listing 6. Fragment funkcji konfigurującej interfejs I2C

```
void PmodPMON1_Config(void)
{
  ...
  pmodPmon1I2c.Instance = I2C2;
  pmodPmon1I2c.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  pmodPmon1I2c.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  pmodPmon1I2c.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  pmodPmon1I2c.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  pmodPmon1I2c.Init.OwnAddress1 = 0x01;
  // Set the I2C_TIMINGR register.
  pmodPmon1I2c.Init.Timing = 0x10563046;
  HAL_I2C_Init(&pmodPmon1I2c);
  PmodMON1_ClearAlarm();

  uint8_t monCommand = 0x05;
  HAL_I2C_Master_Transmit(&pmodPmon1I2c, PMODPMON1_MON_ADDRES,
                          &monCommand, 1, 100);
  uint8_t potCommand[2] = {0x02, 0x01};
  HAL_I2C_Master_Transmit(&pmodPmon1I2c, PMODPMON1_POT_ADDRES,potCommand, 2, 100);
}
```

PmodPMON1

Moduł PmodPMON1 (**fotografia 3**) jest monitorem poboru mocy, składającym się z dwóch układów firmy Analog Devices: ADM119 i AD5112. Pierwszy z nich jest monitorem prądu i napięcia z możliwością generacji alarmu po przekroczeniu zadanej wartości natężenia prądu, ustalanej poprzez napięcie podawane na pin SETV. Do tego celu służy drugi z układów – AD5112, będący potencjometrem cyfrowym o całkowitej rezystancji 5 kΩ i 64 pozycjach ślizgacza. Sposób połączenia obu układów został przedstawiony na **rysunku 4**.

Oba układy są podłączone do jednej magistrali I²C. Potencjometr jest widoczny pod adresem 0x2F, natomiast monitor ma adres konfigurowany za pomocą zworek JP1 i JP2. W przykładzie obie zworki są ustawione w pozycji 1, dzięki czemu ustawiony jest adres 0x30.

Moduł PmodPMON1 posiada 8-pinowe złącze Pmod I²C (J2) służące do komunikacji z obydwoma układami. Oprócz tego dostępne są: złącze J1, zawierające sygnały CONV (wyzwalanie konwersji) i ALERT (powiadomienie o przekroczeniu progu) oraz J3 służące do podłączenie mierzonego układu. Do podłączenia modułu zostały wykorzystane dwa złącza na płytce KAmeLeon, oznaczone jako: I²C KAmod Expander i ARDUINO CONNECTOR, zgodnie z **tabelą 2**.

Kod do obsługi modułu PmodPMON1 znajduje się w plikach PmodPMON1.c oraz PmodPMON1.h i składa się z czterech funkcji. Pierwsze dwie zajmują się konfiguracją interfejsu I²C i wyprowadzeń mikrokontrolera. Fragment funkcji PmodPMON1_Config został przedstawiony na **listingu 6**. Jest w nim konfigurowany interfejs I²C oraz oba układy znajdujące się w module PmodPMON1. Komenda 0x05 przesyłana do układu ADM119 uruchamia ciągłą konwersję natężenia prądu oraz napięcia, z kolei bajty 0x02 i 0x01 przesłane do potencjometru AD5112 ustawiają wartość rezystancji zgodnie z zależnością:

$$R_{WB}(D) = \frac{D}{64} \times R_{AB} + R_W$$

W równaniu tym wartość D jest ustalana przez drugi z bajtów (0x01), a wartości rezystancji R_{AB} i R_W wynoszą odpowiednio 5 kΩ i 70 Ω. Takie ustawienie potencjometru ustala napięcie ~89 mV na wejściu SETV monitora ADM119, co z kolei powoduje generację alarmu po przekroczeniu natężenia prądu o wartości ~49 mA ze względu na wartość rezystora pomiarowego (R2) wynoszącą 0,1 Ω i wzmocnienie napięciowe układu pomiarowego wynoszące 18.

Funkcja PmodPMON1_Config konfiguruje także dwa piny mikrokontrolera podłączone do sygnałów CONV (PF14) oraz ALERT (PF15). Pierwszy z nich jest ustawiany jako wyjście w stanie wysokim, co wymusza ciągłą konwersję. Drugi jest wejściem z przerwaniem wyzwalanym na obu zboczach sygnału.

Druga z funkcji konfiguracyjnych – HAL_I2C_MspInit ustawia piny PF0 i PF1 w funkcji alternatywnej AF4 dla interfejsu I2C2. Zgodnie z konwencją biblioteki STM32Cube, jest ona wywoływana wewnątrz funkcji HAL_I2C_Init.

Kolejną funkcją zaimplementowaną w pliku PmodPMON1.c jest PmodPMON1_GetMeasurements. Odczytuje ona trzy rejestry przechowujące ostatnio zmierzone wartości napięcia i natężenia prądu. Odczyty te są następnie konwertowane według zależności:

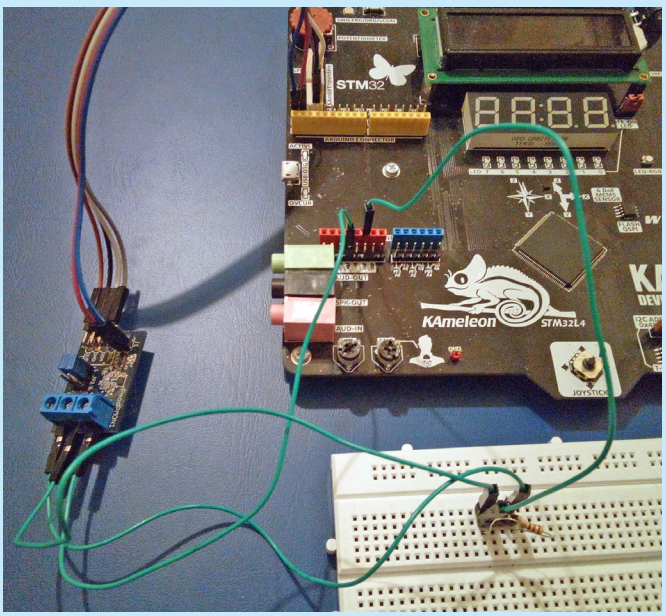
$$Voltage = \left(\frac{V_{FULLSCALE}}{4096}\right) \times Code$$

$$Current = \frac{\left(\left(\frac{I_{FULLSCALE}}{4096}\right) \times Code\right)}{Sense\ Resistor}$$

$V_{FULLSCALE}$, $I_{FULLSCALE}$ i *Sense Resistor* wynoszą odpowiednio 26,52 V, 105,84 mV i 0,1 Ω.

Ostatnia funkcja – PmodMON1_ClearAlarm, czyści flagę alarmu w układzie ADM119. Powoduje to zmianę stanu linii ALERT, chyba że próg alarmowy jest w dalszym ciągu przekroczony, wówczas flaga jest ponownie ustawiania, a linia ALERT pozostaje w stanie aktywnym, czyli niskim.

Plik main.c implementuje dwie funkcje. HAL_GPIO_EXTI_Callback obsługuje przerwania od linii ALERT. Jeżeli alarm jest aktywny (stan niski), to zapalana jest dioda LED1 (PC7) i ustawiana jest programowa flaga *alarmFlag*. W przeciwnym razie dioda jest gaszona a flaga zerowana. W funkcji main wykonywana jest najpierw konfiguracja modułu PmodNAV, a następnie w pętli sprawdzana jest flaga *alarmFlag* – jeżeli została ona ustawiona w przerwaniu, to jest wywoływana opisana wyżej funkcja zerująca alarm. Następnie odczytywane są wartości napięcia i natężenia prądu zmierzone przez układ ADM119 i wysyłane za pośrednictwem portu szeregowego LPUART1, widocznego na komputerze jako wirtualny port szeregowy. Przykładowy układ pomiarowy został przedstawiony na **fotografii 5**.

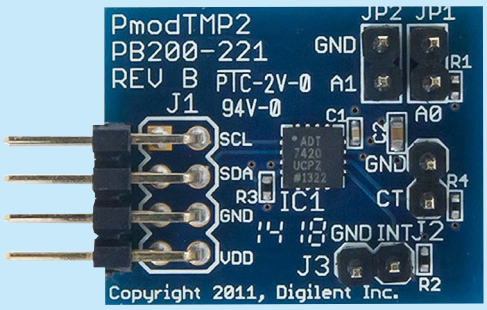


Fotografia 5. Moduł PmodPMON1 podłączony do zestawu KAmeLeon

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

PmodTMP2

Ostatnim omawianym modułem w tej części cyklu jest PmodTMP2 (**fotografia 6**) z układem Analog Devices ADT7420. Jest to cyfrowy czujnik temperatury o rozdzielczości pomiaru 16 bitów i dokładności:



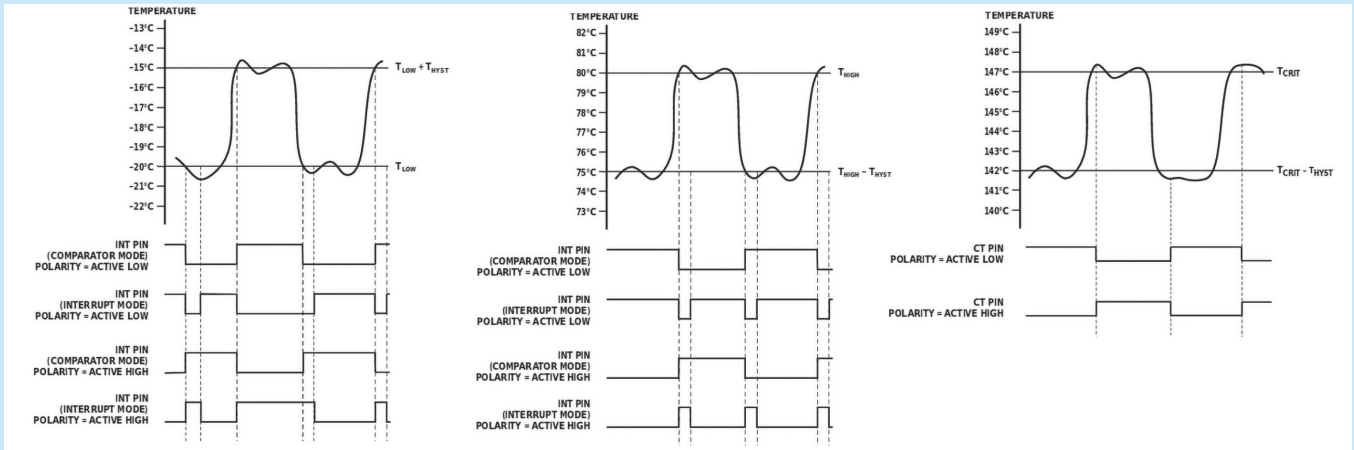
Fotografia 6. Wygląd modułu PmodTMP2

- ±0,20°C w zakresie od −10°C do +85°C przy zasilaniu 3,0 V,
- ±0,25°C w zakresie od −20°C to +105°C przy zasilaniu od 2,7 V do 3,3V.

Układ ADT7420 umożliwia zdefiniowanie trzech alarmów dla zdefiniowanych progów temperatury:

- niski,
- wysoki,
- krytyczny.

Po przekroczeniu progów niskiego i wysokiego, generowany jest sygnał na linii INT, natomiast przekroczenie poziomu krytycznego wyzwala przerwanie na linii CT. Oba sygnały mogą być użyte do obsługi zdarzenia po stronie mikrokontrolera. Moduł PmodTMP2 udostępnia oba sygnały na złączach J2(CT) i J3(INT). Wszystkie zdefiniowane alarmy mogą także używać wartości histerezy wpływającej na wyłączenie alarmu. Sposób generacji przerwania we wszystkich trzech przypadkach został przedstawiony na **rysunku 7**.



Rysunek 7. Generacja przerwań od poszczególnych progów temperatury (źródło: dokumentacja układu ADT7420)

Moduł PmodTMP2 został wyposażony w 8-pinowe złącze Pmod I²C (J1), które w przykładzie zostało podłączono do złącza I²C KAmod Expander zestawu KAmeLeon. Dodatkowo sygnał INT został podłączony do pinu PF15 (D10) na złączu ARDUINO. Sposób połączenia został przedstawiony w **tabeli 3** i na **fotografii 8**.

Kod do komunikacji z modułem PmodTMP2 znajduje się w plikach PmodTMP2.c i PmodTMP2.h. Funkcja służąca do konfiguracji interfejsu I²C i układu ADT7420 – PmodTMP2_Config, znajduje się na **listingu 7**. W pierwszej kolejności ustawiany jest pin PF15 jako wejście z przerwaniem wyzwalanym zboczem opadającym. Następnie konfigurowany jest interfejs I2C2, dla którego piny (PF0 i PF1) są ustawiane w funkcji HAL_I2C_MspInit wywoływanej wewnątrz HAL_I2C_Init. Na koniec konfigurowane są progi temperatury: niski i wysoki. Wartości progów temperatury przed wpisaniem do rejestrów muszą być przesunięcie o 7 bitów w lewo, co jest jednoznaczne z ich pomnożeniem przez 128. Histereza jest wpisywana bezpośrednio do rejestru i może przyjąć wartości od 0°C do 15°C. Podczas zapisu i odczytu rejestrów układ ADT7420 inkrementuje adres, dlatego jest możliwy zapis i odczyt wielu rejestrów podczas pojedynczej transakcji I²C.

Listing 7. Konfiguracja pinów i interfejsu I2C2 dla modułu PmodTMP2

```
void PmodTMP2_Config(int16_t maxTemp, int16_t minTemp, uint8_t hyst)
{
  __HAL_RCC_GPIOF_CLK_ENABLE();
  GPIO_InitTypeDef GPIO_InitStruct;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Pin = GPIO_PIN_15;
  HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);

  HAL_NVIC_SetPriority(EXTI15_10_IRQn, 2, 0);
  HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

  pmodTmp2I2c.Instance = I2C2;
  pmodTmp2I2c.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  pmodTmp2I2c.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  pmodTmp2I2c.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  pmodTmp2I2c.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  pmodTmp2I2c.Init.OwnAddress1 = 0x01;
  pmodTmp2I2c.Init.Timing = 0x10563046;

  HAL_I2C_Init(&pmodTmp2I2c);
  // The temperature thresholds are shifted left by 7 bits.
  maxTemp *= 128;
  minTemp *= 128;
  uint8_t tHigh[2] = {maxTemp >> 8, maxTemp & 0x0F};
  uint8_t tLow[2] = {minTemp >> 8, minTemp & 0x0F};
  uint8_t tHyst = hyst;
  HAL_I2C_Mem_Write(&pmodTmp2I2c, PMODTMP2_ADDRESS, 0x04, 1, tHigh, 2, 100);
  HAL_I2C_Mem_Write(&pmodTmp2I2c, PMODTMP2_ADDRESS, 0x06, 1, tLow, 2, 100);
  HAL_I2C_Mem_Write(&pmodTmp2I2c, PMODTMP2_ADDRESS, 0x0A, 1, &tHyst, 1, 100);
}
```

Listing 8. Odczyt i konwersja temperatury

```
int16_t PmodTMP2_GetTemperature()
{
  // Read the current temperature value (2 bytes).
  uint8_t data[2] = {0};
  HAL_I2C_Mem_Read(&pmodTmp2I2c, PMODTMP2_ADDRESS, 0x00, 1, data, 2, 100);
  int16_t temperature = (data[0] << 8) | data[1];
  // The temperature value has 0.0625°C resolution and is shifted left by 3 bits.
  // It is multiplied by 10 to keep the 0.1°C resolution of the result.
  temperature = (temperature * 10) / 128;
  return temperature;
}
```

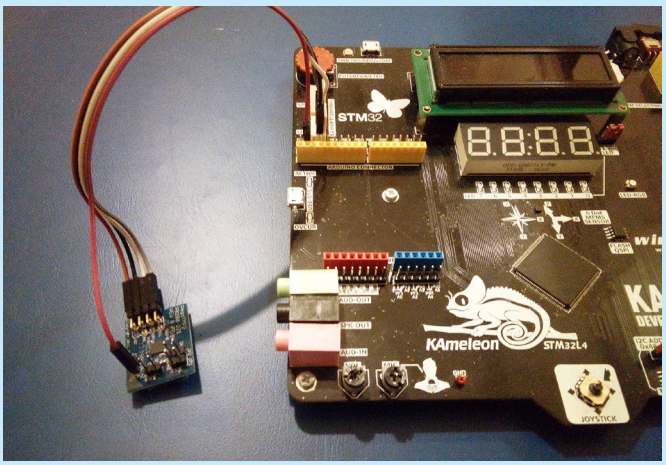
Drugą z funkcji potrzebnych do obsługi modułu PmodTMP2 jest PmodTMP2_GetTemperature odpowiedzialna za odczyt i konwersję zmierzonej temperatury. Podobnie jak w przypadku progów, zmierzona wartość temperatury jest w kodzie uzupełnień do dwójki. Wartość zwracana z funkcji ma dokładność 0,1°C, dlatego przed dzieleniem przez 128 wynik jest mnożony przez 10. Kod funkcji został przedstawiony na **listingu 8**.

Przykładowa aplikacja, której kod znajduje się w pliku main.c, konfiguruje moduł PmodTMP2 i ustawia początkowy stan diody LED 0 informującej o przekroczeniu jednego z ustawionych progów temperatury. Następnie w pętli głównej programu, co 1 s odczytywana jest wartość temperatury przesyłana następnie za pośrednictwem portu szeregowego do komputera. W pliku main.c znajduje się także funkcja obsługi przerwania od linii INT zmieniając stan diody LED 0 na płytce KAmeLeon.

Krzysztof Chojnowski

Tabela 3. Sygnały PmodTMP2 oraz odpowiadające im piny złącz ARDUINO, I²C Expander oraz wyprowadzenia mikrokontrolera; w tabeli pominięto linie zasilania występujące na złączu Pmod

Sygnał	Numer pinu PmodTMP2	Pin ARDUINO CONNECTOR	Pinu I²C Expander	Pin mikrokontrolera
SCL	1 (J2)	-	2	PF1
SDA	2 (J2)	-	3	PF0
INT	3 (J1)	D14	-	PF15



Fotografia 8. Moduł PmodTMP2 przyłączony do zestawu KAmeLeon

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