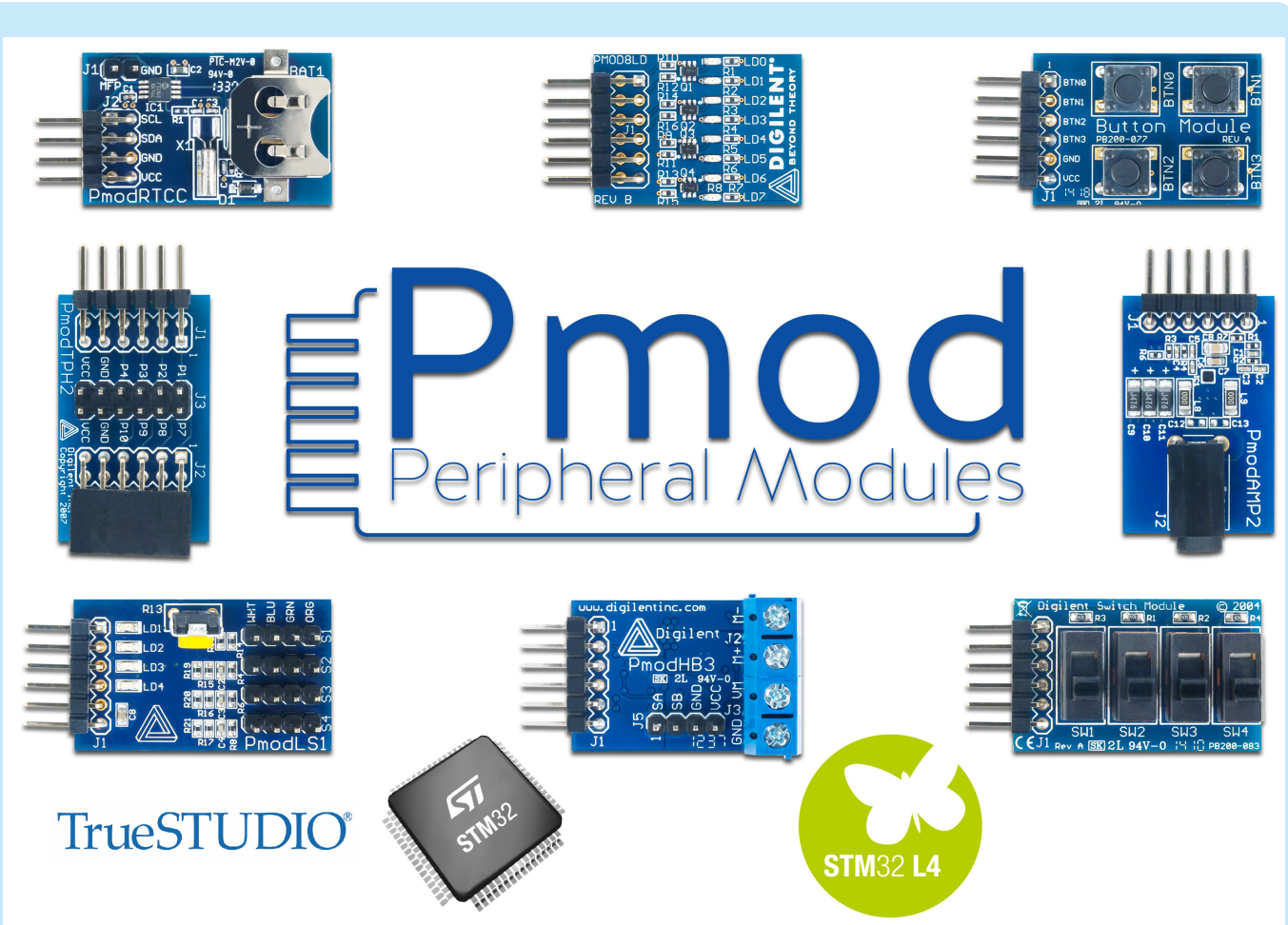


Digilent Pmod i STM32 (4)

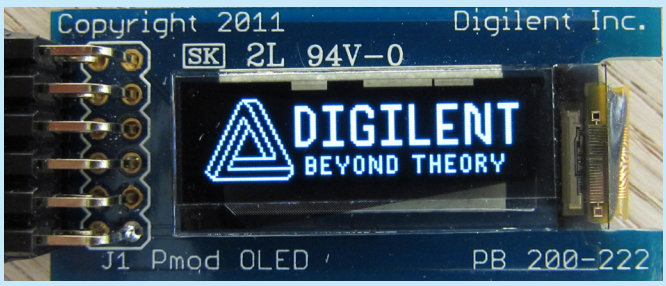
Biblioteki do obsługi modułów peryferyjnych

W kolejnym artykule z cyklu poświęconego modułom Pmod zostaną opisane moduły PmodOLED, PmodGPS oraz Pmod8LD. Przykłady dla wymienionych modułów zostały przygotowane dla środowiska Atollic TrueSTUDIO i zestawu uruchomieniowego KAmeLeon (*www.kameleonboard.org*) z wykorzystaniem biblioteki STM32Cube_FW_L4.

PmodOLED

Moduł PmodOLED (**fotografia 1**) zawiera monochromatyczny wyświetlacz graficzny typu OLED o rozdzielczości 128×32 pikseli. Wyświetlacz jest sterowany za pomocą kontrolera SSD1306 posiadającego wewnętrzną pamięć RAM o rozmiarze 1 kB, dzięki czemu może on obsługiwać wyświetlacze o maksymalnym rozmiarze 128×64 piksele. Pamięć RAM kontrolera została podzielona na 8 stron, do których dostęp jest możliwy za pośrednictwem interfejsów SPI, I²C i równoległego z 8-bitową szyną danych.

PmodOLED został wyposażony w 12-pinowe złącze SPI typu 2A zawierające oprócz sygnałów SPI, cztery sygnały sterujące pracą wyświetlacza. Ze względu na liczbę sygnałów niezbędnych do obsługi modułu nie może być on podłączony do złącza Pmod-SPI Kameleona, dlatego na potrzeby przykładu zostało wykorzystane złącze Arduino. Lista wszystkich sygnałów oraz sposób ich podłączenia do zestawu KAmeLeon znajduje się w **tabeli 1**.



Fotografia 1. Moduł PmodOLED

Do obsługi wyświetlacza została wykorzystana biblioteka pobrana z oficjalnej strony modułu PmodOLED i przeznaczona dla środowiska MPIDE ***http://bit.ly/2xPZ7BT***. Znajdujący się w niej plik sterownika (Drivers/OLED/OledDriver.cpp) został zmodyfikowany tak, aby korzystał z biblioteki STM32Cube_FW_L4 do obsługi interfejsu

SPI i linii GPIO. Biblioteka została napisana w języku C++, dlatego projekt przykładu został skonfigurowany również dla tego języka.

Obsługa wyświetlacza rozpoczyna się od utworzenia obiektu klasy OledClass i wywołaniu metody OledClass::begin(). Jest ona odpowiedzialna za konfigurację interfejsów SPI i GPIO, inicjalizację wewnętrznych pól sterownika, przeprowadzenie procedury inicjalizacji wyświetlacza i wyzerowanie wewnętrznego bufora danych oraz pamięci kontrolera SSD1306. Fragment kodu, realizujący konfigurację SPI został przedstawiony na listingu 4. Ustawia on polaryzację i fazę zegara w trybie 3. (CPOL=1, CPHA=1), rozmiar danych na 8 bitów i programową kontrolę sygnału CS (NSS). Pełna procedura inicjalizacji wyświetlacza została opisana w dokumentacji modułu dostępnej na stronie: ***http://bit.ly/2RdpVnV***.

Operacje rysowania i pisania są dostępne poprzez API biblioteki OLED, którego najważniejsze metody zostały wymienione w **tabeli 2**. Ze względu na obecność wewnętrznego bufora obrazu w bibliotece OLED, wszystkie operacje są wykonywane w pamięci mikrokontrolera, a aktualizacja wyświetlacza jest wykonywana na żądanie. Jest to optymalizacja mająca na celu ograniczenie liczby transferów SPI i umożliwienie nakładania na siebie wyświetlanych elementów. W przypadku tekstu opcja buforowania może zostać włączona lub wyłączona. W drugim przypadku napisy są wysyłane bezpośrednia do pamięci kontrolera wyświetlacza. Na uwagę zasługują także dwa tryby pisania. Pierwszy z nich (metody OledClass::drawChar i OledClass::drawString) umieszcza znaki w dowolnym miejscu wyświetlacza ustawianym metodą OledClass::moveTo. W drugim trybie, obsługiwanym przez metody OledClass::putChar i OledClass::putString, wyświetlacz jest podzielony na 4 linie i 16 kolumn, do których wyrównany jest napis. Linia i kolumna wybierana jest za pomocą metody OledClass::setCursor.

W opisywanym przykładzie wykonywana jest inicjalizacja, a następnie cały wyświetlacz jest pokrywany wybranym wzorcem. Na środku umieszczane są dwa napisy: „Hello PmodOLED" i „KAmeLeon board". Efekt działania programu został przedstawiony na **fotografii 2**.

Tabela 1. Podłączenie modułu PmodOLED do złącza ARDUINO zestawu KAmeLeon

Sygnał	Numer pinu PmodOLED	Numer pinu Kameleon ARDUINO CONNECTOR	Pin mikrokontrolera
CS	1	D10	PB12
MOSI	2	D11	PB15
MISO	3	D12	PB14
SCLK	4	D13	PB10
GND	5	GND	-
VCC	6	+3,3	-
D/C	7	D9	PB13
Reset	8	D8	PD11
VBATC	9	D7	PB11
VDDC	10	D6	PD10
GND	11	-	-
VCC	12	-	-

Tabela 2. Wybrane metody klasy OledClass

Metoda	Opis
OledClass::begin	Włączenie zasilania kontrolera i inicjalizacja wyświetlacza
OledClass::end	Wyłączenie zasilania kontrolera i wyświetlacza
OledClass::displayOn	Włączenie wyświetlania
OledClass::displayOff	Wyłączenie wyświetlania bez utraty stanu pamięci obrazu
OledClass::clear	Czyszczenie wewnętrznego bufora biblioteki i pamięci kontrolera
OledClass::clearBuffer	Czyszczenie wewnętrznego bufora biblioteki
OledClass::updateDisplay	Wysłanie bufora obrazu do pamięci kontrolera
OledClass::setDrawColor	Ustawienie koloru do rysowania piksela i linii (0 lub 1)
OledClass::setDrawMode	Ustawienie trybu nakładania (nadpisanie, AND, OR, XOR)
OledClass::setFillPattern	Ustawienie wzorca wypełnienia
OledClass::getStdPattern	Pobranie wskaźnika na jeden ze wzorców wypełnienia (0 - 7)
OledClass::moveTo	Ustawienie punktu początkowego do rysowania
OledClass::drawPixel	Rysowanie piksela
OledClass::drawLine	Rysowanie linii
OledClass::drawRect	Rysowanie niewypełnionego prostokąta
OledClass::drawFillRect	Rysowanie prostokąta wypełnionego wybranym wzorcem
OledClass::putBmp	Rysowanie bitmapy
OledClass::drawChar	Rysowanie znaku
OledClass::drawString	Rysowanie napisu
OledClass::setCursor	Ustawienie kursora do pisania z wyrównaniem do linii i kolumny
OledClass::setCharUpdate	Włączenie lub wyłączenie buforowania znaków i napisów
OledClass::putChar	Rysowanie znaku z wyrównaniem do linii i kolumny
OledClass::putString	Rysowanie napisu z wyrównaniem do linii i kolumny

PmodGPS

Moduł PmodGPS (**fotografia 3**) został wyposażony w moduł Gms-u1LP z wbudowaną anteną. Moduł ten zapewnia czułość na poziomie –165 dBm i dokładność pozycjonowania 3 m. Domyślnie moduł Gms-u1LP wysyła dane w postaci zdań zgodnych z protokołem NMEA za pośrednictwem interfejsu UART. Typy wysyłanych zdań zostały zamieszczone w **tabeli 3**. Dodatkowo dostępne są dwa sygnały: 3DF i 1PPS. Pierwszy z nich informuje o stanie pozycjonowania zmieniając stan co sekundę podczas ustalania pozycji i trzymając stan niski jeżeli pozycja (2D lub 3D) została ustalona. Drugi sygnał – 1PPS zapewnia synchronizację z wewnętrznym czasem uzyskanym z GPS dostarczając impulsów o długości 100 ms na początku każdej sekundy.

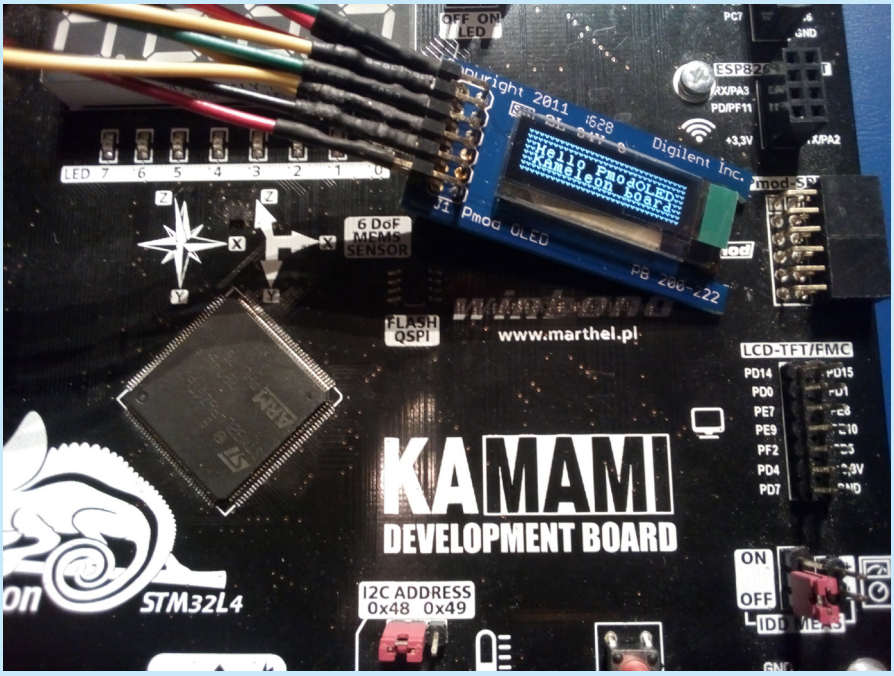
Zdania w protokole NMEA wysyłane są w postaci tekstowej. Zaczynają się od prefiksu $GP, po którym występuje typ zdania i lista wartości oddzielonych od siebie przecinkami. Najważniejszym zadaniem, z punktu widzenia danych niezbędnych do nawigacji, jest

Tabela 3. Zdania NMEA wysyłane przez PmodGPS

Typ	Opis
GGA	Czas i pozycja
GSA	Aktywne satelity GNSS i rozmycie precyzji (DOP)
GSV	Widoczne satelity GNSS
RMC	Zalecane minimalne dane GNSS
VTG	Kurs nad ziemią i prędkość względem ziemi

RMC (*Recommended Minimum Navigation Information*) zawierające takie informacje jak czas, współrzędne geograficzne, prędkość oraz kurs. Szczegółowa struktura zdania RMC została przedstawiona w **tabeli 4**. Opis pozostałych zdań zawierającym m.in. wysokość nad poziomem morza (GGA), widoczne satelity (GSV) można znaleźć w dokumentacji modułu PmodGPS: ***http://bit.ly/2QiNYQL***.

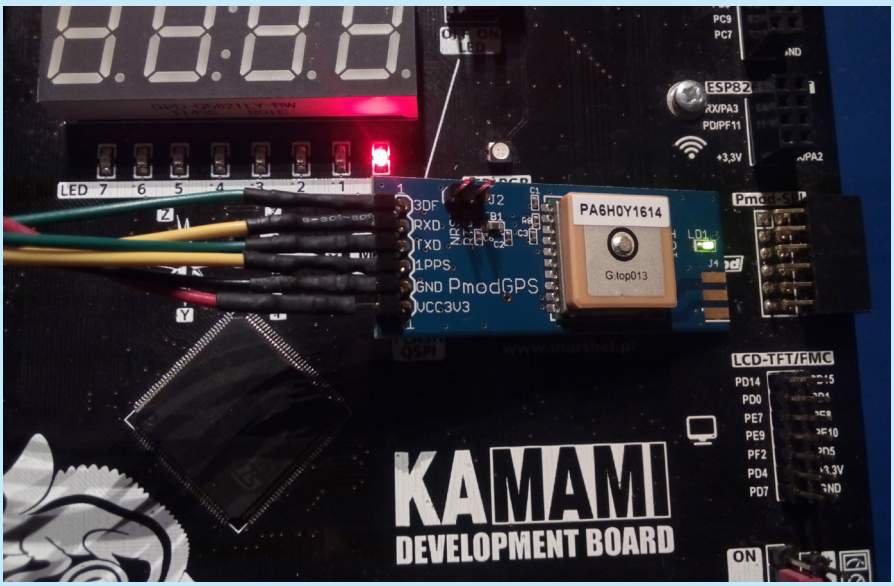
Poprzednie części kursu i dodatkowe materiały dostępne są na stronie www.media.avt.pl



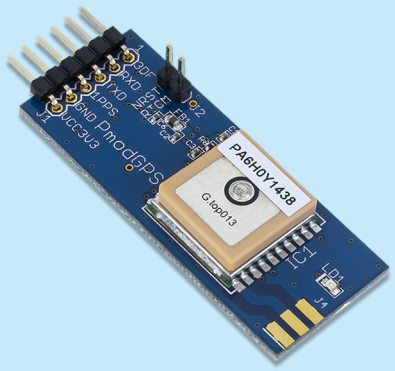
Fotografia 2. PmodOLED przyłączony do zestawu KAmeLeon

Tabela 4. Struktura zdania RMC

Pole	Opis
Message ID	Nagłówek zdania: $GPRMC
UTC Time	Aktualna godzina w postaci hhmmss.sss
Status	Stan danych – ważne (A), lub nie (V)
Latitude	Szerokość geograficzna w stopniach (d) i minutach (m) : ddmm.mmmm
N/S indicator	Szerokość geograficzna północna (N) lub południowa (S)
Longitude	Długość geograficzna w stopniach (d) i minutach (m) : ddmm.mmmm
E/W indicator	Długość geograficzna wschodnia (E) lub zachodnia (W)
Speed	Prędkość względem ziemi w węzłach
Course	Kurs rzeczywisty w stopniach
Date	Aktualna data w postaci ddmmyy
Magnetic variation	Deklinacja magnetyczna w stopniach
E/W indicator	Deklinacja magnetyczna wschodnia (E) lub zachodnia (W)
Mode	Tryb autonomiczny (A), różnicowy (D), szacowany (E)
Checksum	Suma kontrolna



Fotografia 4. PmodGPS przyłączony do zestawu KAmeLeon



Fotografia 3. Moduł PmodGPS

Do modułu Gms-u1LP można także wysyłać komendy NMEA konfigurujące urządzenie. Mają one określoną strukturę, przedstawioną w **tabeli 5**. Jedyną komendą używaną w przykładzie jest odczyt wersji firmware'u znajdującego się w module: "$PMTK605*31\r\n". W odpowiedzi na jej wysłanie, moduł PmodGPS odpowiada pakietem „$PMTK705,AXN_2.31_3339_13082100,5458,PA6H,1.0*69\r\n", zawierającym nazwę i wersję oprogramowania (AXN_2.31_3339_13082100).

PmodGPS jest wyposażony w złącze typu UART typu 4 z pinami 3DF i 1PPS w miejsce CTS i RTS. W przykładzie, moduł został podłączony do złącza Arduino udostępniającego interfejs USART3 i linie GPIO. Lista pinów złącza J1 modułu PmodGPS i odpowiadające im piny złącza Arduino zostały przedstawione w **tabeli 6**.

Konfiguracja modułu PmodGPS znajduje się w dwóch funkcjach w pliku src/PmodGPS.c. Pierwsza z nich – PmodGPS_Config – przedstawiona na **listingu 2**, konfiguruje piny 1PPS i 3DF jako przerwania zewnętrzne oraz interfejs USART3 według domyślnych ustawień modułu (baudrate 9600, 8-bitowe dane, 1 bit stopu, brak parzystości).

Piny dla interfejsu UART konfigurowane są w drugiej funkcji – PmodGPS_HAL_UART_MspInit, pokazanej na **listingu 3**. Funkcja ta jest wywoływana podczas inicjalizacji interfejsu UART przez bibliotekę STM32Cube za pośrednictwem innej funkcji – HAL_UART_MspInit, znajdującej się w pliku main.c.

Ze względu na to, że w przykładzie używane są dwa interfejsy UART, funkcja HAL_UART_MspInit decyduje o tym, który interfejs powinien być skonfigurowany. Plik src/PmodGPS.c zawiera ponadto funkcje odpowiedzialne za wymianę danych z modułem PmodGPS: PmodGPS_Write oraz PmodGPS_Read, jednak są one jedynie wrapperami na nieblokujące funkcje transmisji i odczytu danych korzystających z przerwań: HAL_UART_Transmit_IT i HAL_UART_Receive_IT.

Odbiór danych z modułu PmodGPS odbywa się za pośrednictwem szeregu zmiennych globalnych przedstawionych na **listingu 4**. Do przechowywania danych używane są bufory typu DataBuffer zawierające maksymalnie po 128 bajtów danych oraz index wskazujący zajętość bufora. W przykładzie zostały użyte dwa bufory – jeden zapisywany przychodzącymi danymi, a drugi zawierający gotowy pakiet danych do przetworzenia. Indeks aktualnie zapisywanego bufora znajduje się w zmiennej currentDataBuffer, natomiast flaga informująca o zakończeniu odbioru pakietu przechowywana jest w zmiennej dataReady. Wymienione zmienne są współdzielone przez funkcje main oraz HAL_UART_RxCpltCallback, w pliku main.c. Funkcja main, po wykonaniu podstawowej inicjalizacji procesora i wyczyszczeniu buforów wysyła opisaną wcześniej komendę odczytu wersji firmware'u, po czym uruchamia asynchroniczny odczyt danych do pierwszego bufora i przechodzi w stan oczekiwania na odebranie pakietu. Odebranie pakietu jest sygnalizowane przez flagę dataReady, po otrzymaniu której cały pakiet wypisywany jest na port szeregowy. Cała procedura konfiguracji i wypisywania danych została przedstawiona na **listingu 5**.

Listing 1. Konfiguracja interfejsu SPI dla PmodOLED

```
pmodOledSpi.Instance = SPI2;
pmodOledSpi.Init.Mode = SPI_MODE_MASTER;
pmodOledSpi.Init.Direction = SPI_DIRECTION_2LINES;
pmodOledSpi.Init.DataSize = SPI_DATASIZE_8BIT;
pmodOledSpi.Init.CLKPolarity = SPI_POLARITY_HIGH;
pmodOledSpi.Init.CLKPhase = SPI_PHASE_2EDGE;
pmodOledSpi.Init.NSS = SPI_NSS_SOFT;
pmodOledSpi.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
pmodOledSpi.Init.FirstBit = SPI_FIRSTBIT_MSB;
pmodOledSpi.Init.TIMode = SPI_TIMODE_DISABLE;
pmodOledSpi.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
pmodOledSpi.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
HAL_SPI_Init(&pmodOledSpi);
```

Listing 2. Konfiguracja interfejsu UART dla PmodGPS

```
void PmodGPS_Config(void)
{
    // Enable clock for GPIOD port required for 3DF and 1PPS pins.
    __HAL_RCC_GPIOD_CLK_ENABLE();
    // Configure the interrupts on PD10 (1PPS) and PD6 (3DF). The interrupts are active on both edges
    // to indicate the connection and disconnection events.
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Pin = GPIO_PIN_6 | GPIO_PIN_10;

    HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
    HAL_NVIC_SetPriority(EXTI9_5_IRQn, 2, 0);
    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 2, 1);
    HAL_NVIC_EnableIRQ(EXTI9_5_IRQn);
    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);

    pmodGpsUart.Instance = USART3;
    pmodGpsUart.Init.BaudRate = 9600;
    pmodGpsUart.Init.WordLength = UART_WORDLENGTH_8B;
    pmodGpsUart.Init.StopBits = UART_STOPBITS_1;
    pmodGpsUart.Init.Parity = UART_PARITY_NONE;
    pmodGpsUart.Init.Mode = UART_MODE_TX_RX;
    pmodGpsUart.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    pmodGpsUart.Init.OverSampling = UART_OVERSAMPLING_16;
    pmodGpsUart.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    HAL_UART_Init(&pmodGpsUart);
}
```

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 bezpośredni link do biblioteki.

Listing 3. Konfiguracja GPIO dla interfejsu UART

```
void PmodGPS_HAL_UART_MspInit(UART_HandleTypeDef *huart)
{
    // Enable the clocks for GPIO pins used by the UART3 port (PC4, PC5).
    __HAL_RCC_USART3_CLK_ENABLE();
    __HAL_RCC_GPIOC_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_AF7_USART3;
    GPIO_InitStruct.Pin = GPIO_PIN_4 | GPIO_PIN_5;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

    // Enable the interrupts generated by USART3 port.
    HAL_NVIC_SetPriority(USART3_IRQn, 1, 0);
    HAL_NVIC_EnableIRQ(USART3_IRQn);
}
```

Dane z modułu PmodGPS odbierane są w przerwaniu od portu USART3, którego obsługa znajduje się na **listingu 6**. W pierwszej kolejności sprawdzane jest czy został odebrany cały pakiet, co jest jednoznaczne z otrzymaniem znaku nowej linii: '\n'. Jeżeli tak, to ustawiana jest flaga dataReady i zmieniany jest bufor do odczytu, w przeciwnym razie przesuwany jest jedynie indeks obecnego bufora. Po zakończonym odbiorze inicjalizowany jest odczyt kolejnego znaku do aktualnego bufora.

Listing 4. Zmienne używane do odbioru danych z modułu PmodGPS

```
#define DATA_BUFFERS_COUNT 2

typedef struct
{
    uint8_t buffer[128];
    uint32_t index;
} DataBuffer;

DataBuffer dataBuffers[DATA_BUFFERS_COUNT];
uint32_t currentDataBuffer = 0;
uint8_t dataReady = 0;
```

Funkcja obsługi przerwania od dwóch sygnałów modułu: 1PPS i 3DF została zaimplementowana w funkcji HAL_GPIO_EXTI_Callback – w zależności od tego, który sygnał wywołał przerwanie zmieniany jest stan jednej z diod znajdujących się na płytce KAmeLeon, dzięki czemu możliwe jest obserwowanie stanu obu linii. Moduł PmodGPS dołączony do płyty KAmeLeon został przedstawiony na **fotografii 4**.

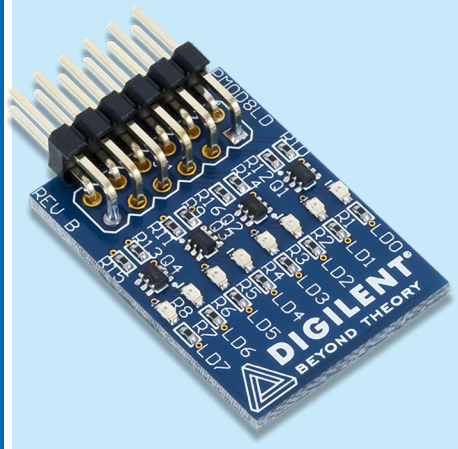
Pmod8LD

Jako ostatni zostanie przedstawiony moduł Pmod8LD (**fotografia 5**). Zawiera on 8 LEDów sterowanych niezależnie za pomocą linii GPIO – zapalanych stanem wysokim i gaszonych stanem niskim. Sygnały wejściowe są podłączone do bramek tranzystorów bipolarnych, dzięki czemu piny sterujące nie są obciążane prądowo, gdy diody są zapalone. Pmod8LD posiada 12-pinowy interfejs GPIO, którego sposób podłączenia do złącza Arduino zestawu KAmeLeon został przedstawiony w **tabeli 7**.

Tabela 5. Struktura komendy NMEA

Pole	Opis
Preamble	Preambuła „$”
Talker ID	Identyfikator komendy „PMTK”
Packet Type	Typ pakietu o wartościach od „000” do „999”
Data Field	Opcjonalne pola danych rozdzielone przecinkami
Checksum	Suma kontrolna poprzedzona znakiem „*”

Obsługa modułu Pmod8LD znajduje się w plikach src/Pmod8LD.h i src/Pmod8LD.c. Pierwszy z nich zawiera enumerację Pmod8LD_Led, z listą wszystkich dostępnych LEDów oraz deklaracje funkcji do konfiguracji, włączania i wyłączania, zdefiniowanych w drugim



Fotografia 5. Moduł Pmod8LD

Listing 5. Funkcja main przykładu

```
int main(void)
{
    HAL_Init();
    SystemClock_Config();
    Led_Config();
    Serial_Config();
    for (uint32_t i = 0; i < DATA_BUFFERS_COUNT; i++) dataBuffers[i].index = 0;
    PmodGPS_Config();
    PmodGPS_Write("$PMTK605*31\r\n", 13);
    PmodGPS_Read(dataBuffers[0].buffer, 1);
    while(1)
    {
        while(dataReady == 0)
        __NOP();
        dataReady = 0;
    // The buffer before the current one contains the data received from PmodGPS.
        uint32_t readyDataBuffer =
            (currentDataBuffer > 0) ? (currentDataBuffer - 1) : (DATA_BUFFERS_COUNT -
1);
    // Write the data from the buffer to the serial port and clear the buffer.
        Serial_Write((char*)dataBuffers[readyDataBuffer].buffer,
dataBuffers[readyDataBuffer].index + 1);
        dataBuffers[readyDataBuffer].index = 0;
    }
}
```

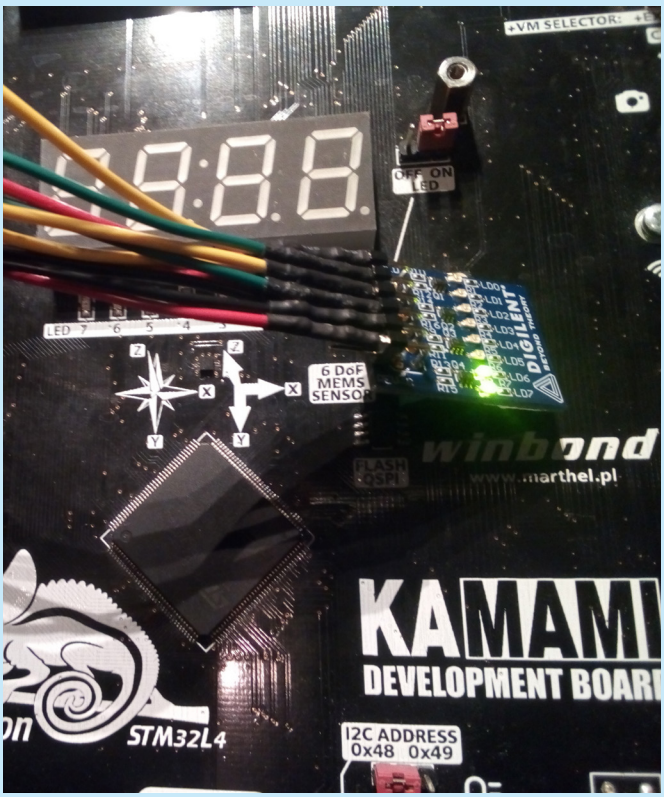
Listing 6. Obsługa przerwania interfejsu UART

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(dataBuffers[currentDataBuffer].buffer[dataBuffers[currentDataBuffer].index] == '\n')
    {
        dataReady = 1;
        currentDataBuffer++;
        if(currentDataBuffer == DATA_BUFFERS_COUNT) currentDataBuffer = 0;
    } else {
        dataBuffers[currentDataBuffer].index++;
    }
    PmodGPS_Read(&dataBuffers[currentDataBuffer].buffer[dataBuffers[currentDataBuffer].index], 1);
}
```

Listing 7. Fragmenty funkcji obsługujących diody Pmod8LD

```
void Pmod8LD_SetLed(Pmod8LD_Led led)
{
    switch(led)
    {
        case Pmod8LD_Led0:
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_SET);
        break;

void Pmod8LD_ResetLed(Pmod8LD_Led led)
{
    switch(led)
    {
        case Pmod8LD_Led0:
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_10, GPIO_PIN_RESET);
        break;
```



Fotografia 6. Pmod8LD przyłączony do zestawu KAmeLeon

z plików. Funkcja konfiguracyjna Pmod8LD_Config włącza zegary dla portów GPIOB oraz GPIOD, po czym ustawia wszystkie piny podłączone do modułu Pmod8LD jako wyjścia. Do ustawania stanu diody służą funkcje Pmod8LD_SetLed i Pmod8LD_ResetLed, których fragmenty przedstawione zostały na **listingu 7**. Wszystkie wymienione funkcje wykorzystują funkcje do obsługi GPIO znajdujące się w bibliotece STM32Cube.

Główna pętla przykładowej aplikacji wykonuje konfigurację pinów, po czym zapala i gasi kolejno wszystkie diody modułu Pmod8LD, co zostało pokazane na **fotografii 6**.

Krzysztof Chojnowski

Tabela 6. Podłączenie modułu PmodGPS do złącza ARDUINO zestawu KAmeLeon

Sygnał	Numer pinu PmodGPS	Numer pinu Kameleon ARDUINO CONNECTOR	Pin mikrokontrolera
3DF	1	D2	PD6
RX	2	D1	PC4
TX	3	D0	PC5
1PPS	4	D6	PD10
GND	5	GND	-
VCC	6	+3,3	-

Tabela 7. Podłączenie modułu PmodLD8 do złącza ARDUINO zestawu KAmeLeon

Sygnał	Numer pinu Pmod8LD	Numer pinu Kameleon ARDUINO CONNECTOR	Pin mikrokontrolera
LD0	1	D6	PD10
LD1	2	D7	PB11
LD2	3	D8	PD11
LD3	4	D9	PB13
GND	5	GND	-
VCC	6	+3,3	-
LD4	7	D10	PB12
LD5	8	D11	PB15
LD6	9	D12	PB14
LD7	10	D13	PB10
GND	11	-	-
VCC	12	-	-