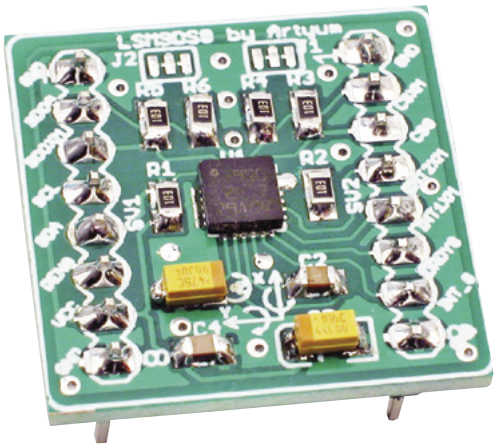
Czujnik inercyjny LSM9DS0 oraz jego zastosowanie praktyczne

AVT 5491



W artykule opisano projekt modułu płytki PCB z układem czujnika inercyjnego LSM9DS0. Dzięki zastosowaniu złącza goldpin jest możliwe wielokrotne zastosowanie jednego układu IMU w wielu urządzeniach. W dalszej kolejności omówiono przykłady praktycznej komunikacji z czujnikiem inercyjnym LSM9DS0 za pomocą interfejsu I²C. Oprogramowanie wykonano w języku C dla mikrokontrolera ATmega8.

***Rekomendacje:** tekst jest przeznaczony dla osób, które chcą w krótkim czasie rozpocząć pracę z układem IMU i zastosować go we własnych projektach.*

Zaprezentowane w dalszej części artykułu kody źródłowe napisano w języku C dla mikrokontrolera ATmega8. Należy mieć na uwadze, iż pominięto niektóre funkcjonalności, dlatego zawarte tu informacje należy traktować jako wstęp i ułatwienie do dalszej analizy dokumentacji producenta. Opis jest jednak wystarczający, aby uruchomić LSM9DS0 na 8-bitowej platformie wyposażonej w interfejs I²C.

Możliwości LSM9DS0

IMU (*Inertial Measurement Unit*) LSM9DS0 jest inercyjnym urządzeniem pomiarowym firmy STMicroelectronics wykonanym w technologii MEMS. W jednym układzie scalonym znajdują się sensory: 3-osiowy żyroskop, 3-osiowy akcelerometr, 3-osiowy magnetometr. Taki zestaw czujników jest nazywany 9DOF (9 stopni swobody). Dodatkowo, w strukturę układu wbudowano również termometr.

Układ umożliwia pomiar sił przyspieszenia, prędkości kątowych oraz wartości pól magnetycznych w przestrzeni 3D. Wysoka, 16-bitowa rozdzielczość odczytów odzwierciedla:

- Przyśpieszenia liniowe w zakresach ±2g, ±4g, ±6g, ±8g, ±16g.
- Wartości indukcji pola magnetycznego w zakresach: ±2, ±4, ±8, ±12 Gs.
- Prędkości kątowe w zakresach ±245/±500/±2000 stopni na sekundę.

Każdy z 3 czujników może zostać wprowadzony w tryb uśpienia, dzięki czemu możliwe jest wykorzystywanie tylko wybranych sensorów oraz oszczędność w poborze prądu. Duże możliwości i niewielkie wymiary umożliwiają zastosowanie w wielu projektach opartych na procesorach 8 lub 32-bitowych. Komunikacja z układem możliwa jest poprzez magistralę SPI lub I²C. Układ jest kompatybilny z I²C o standardowej szybkości linii danych 100 kHz, jak również 400 kHz w trybie szybkim. Wyposażony jest w konfigurowalne generatory przerwań na określonych pinach, które umożliwiają programowe wykrywanie ruchu lub reakcje na zmianę położenia w przestrzeni 3D. Układ zasilany jest napięciem od 2,4 V do 3,6 V.

Konfiguracja sprzętowa modułu LSM9DS0

Na **rysunku 1** zamieszczono schemat płytki drukowanej z układem LSM9DS0. Projekt uwzględnia zalecenia producenta dotyczące filtrowania zasilania, a także przewiduje łatwą zmianę parametrów użytkowych. Płytkę wyposażono w dwie zworki lutownicze (J1, J2) sterujące doprowadzeniami CS_G i CS_XM oraz SDO_G i SDO_XM. Pierwsza para umożliwia wybór interfejsu komunikacyjnego dla G oraz XM zgodnie z **tabelą 1**. Druga para spełnia dwa zadania – umożliwia ustawienie adresu lub wyprowadzenie danych za pomocą odpowiedniego interfejsu komunikacyjnego (**tabela 2**).

Zmiana poziomów na wyprowadzeniach jest wykonywana poprzez rozwarcie lub zwarcie za pomocą zworek ścieżek łączących rezystory R3…R6 z linią Vcc. Domyślnie układ został przygotowany do pracy poprzez magistralę I²C z adresami 0x6B (G) i 0x1D (XM).

Na schemacie ideowym można zauważyć rozmieszczenie wyprowadzeń płytki drukowanej. Ich funkcje opisano w **tabeli 3**.

W ofercie AVT*	
AVT-5491 A	
Podstawowe informacje:	
• Wymiary: 31 mm×17 mm.	
Dodatkowe materiały na FTP:	
ftp://ep.com.pl, user: 54721, pass: qn2jbq4t	
• wzory płytek PCB	
Projekty pokrewne na FTP:	
(wymienione artykuły są w całości dostępne na FTP)	
AVT-1806	Detektor drgań (EP 7/2014)
AVT-5393	ASO – Automatyczny system ostrzegania (EP 4/2013)
AVT-5387	gLogger – 3-osiowy rejestrator przyśpieszenia (EP 3/2013)
AVT-5223	Kieszonkowy akcelerometr (EP 2/2010)

* Uwaga:
Zestawy AVT mogą występować w następujących wersjach:
AVT xxxx UK – to zaprogramowany układ. Tylko i wyłącznie. Bez elementów dodatkowych.
AVT xxxx A – płytka drukowana PCB (lub płytki drukowane, jeśli w opisie wyraźnie zaznaczono), bez elementów dodatkowych.
AVT xxxx A+ – płytka drukowana i zaprogramowany układ (czyli połączenie wersji A i wersji UK) bez elementów dodatkowych.
AVT xxxx B – płytka drukowana (lub płytki) oraz komplet elementów wymienionych w załączniku pdf
AVT xxxx C – to nic innego jak zmontowany zestaw B, czyli elementy wlutowane w PCB. Należy mieć na uwadze, że o ile nie zaznaczono wyraźnie w opisie, zestaw ten nie ma obudowy ani elementów dodatkowych, które nie zostały wymienione w załączniku pdf
AVT xxxx CD – oprogramowanie (nieczęsto spotykana wersja, lecz jeśli występuje, to niezbędne oprogramowanie można ściągnąć, klikając w link umieszczony w opisie kitu)
Nie każdy zestaw AVT występuje we wszystkich wersjach! Każda wersja ma załączony ten sam plik pdf! Podczas składania zamówienia upewnij się, którą wersję zamawiasz! (UK, A, A+, B lub C). *http://sklep.avt.pl*

Schemat montażowy pokazano na **rysunku 2**. Sensor zamontowano na płytce dwustronnej, a wszystkie komponenty umieszczono na warstwie górnej. W pierwszej kolejności należy zamontować układ IMU (LSM9DS0), którego polaryzacja musi być zgodna z nadrukowanym schematem wektorów pomiarowych. W następnej kolejności należy zamontować elementy pasywne: kondensatory i rezystory. Przed montażem listwy goldpin należy zwrócić uwagę na umiejscowienie układu w docelowym projekcie, gdyż zmiana orientacji IMU, pomimo iż ta jest dowolna, niesie za sobą określone konsekwencje, jeśli chodzi o interpretację wyników pomiarów.

Niestety, montaż układu LSM9DS0 na PCB metodą „domową” jest bardzo trudny ze względu na małe i trudno dostępne

pady. Z tego powodu może sprawić znaczne trudności osobom niemającym wprawy w montażu układów elektronicznych.

Komunikacja I²C

Komunikacja za pomocą I²C jest dwukierunkowa i przebiega w konfiguracji master – slave. Możliwe są 4 tryby komunikacji: wysłanie bajtu do układu *slave*, wysłanie wielu bajtów do układów *slave*, odczyt bajtu z układu *slave*, odczyt wielu bajtów z układów *slave*. Inicjalizacja połączenia jest wykonywana po stronie układu *master*, którym z reguły jest mikrokontroler. Dla uproszczenia kodu przyjęto definicje pokazane na **listingu 1**.

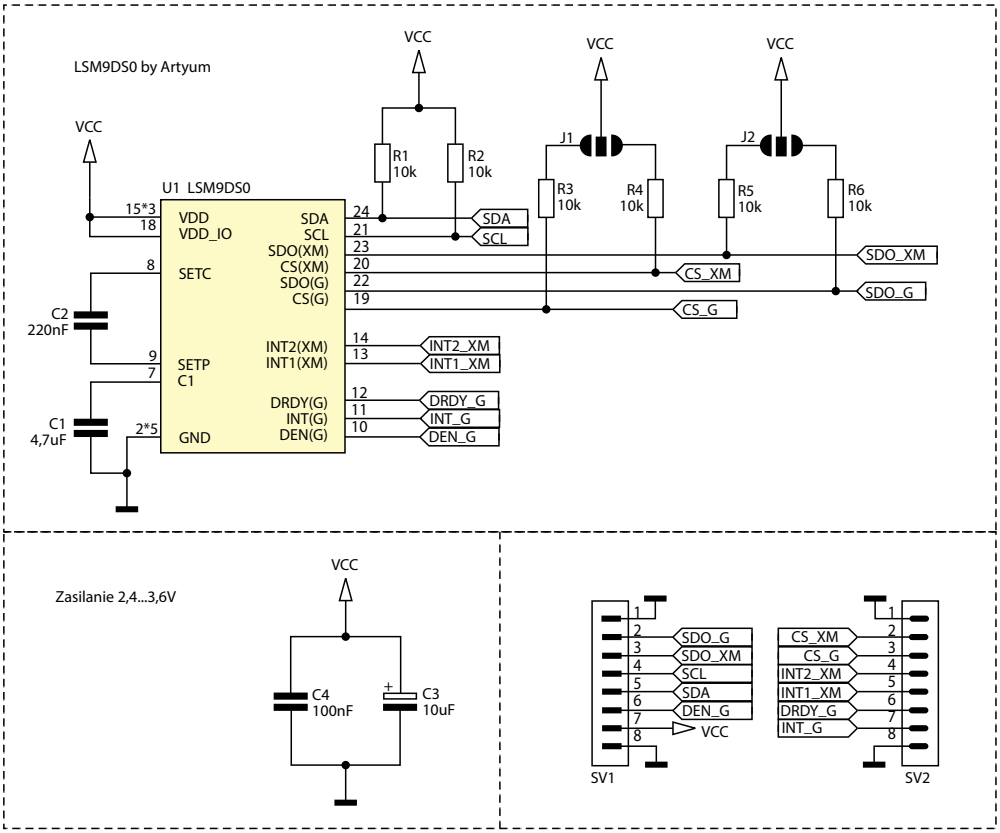
Zmienną I2C_ILOOP utworzono w celu zabezpieczenia przez zawieszaniem się programu w wypadku wystąpienia problemu w trakcie komunikacji.

Inicjalizacja interfejsu TWI (I²C) mikrokontrolera ATmega8 sprowadza się do określenia częstotliwości pracy zgodnie ze wzorem znajdującym się w dokumentacji. Zależnie od trybów pracy interfejsów układów dołączonych do I²C, maksymalna częstotliwość przebiegu zegarowego może wynosić 100 kHz lub 400 kHz, co odpowiada prędkości transmisji 100 kb/s lub 400 kb/s. Oczywiście, trzeba przy tym uwzględnić częstotliwość taktowania mikrokontrolera. W wypadku taktowania za pomocą przebiegu o częstotliwości 8 MHz do rejestru TWBR należy zapisać liczbę dziesiętną 8 (**listing 2**). Odpowiednie dobranie częstotliwości zmniejsza prawdopodobieństwo wystąpienia błędów w transmisji danych.

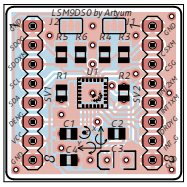
Na **listingu 3** pokazano funkcję służącą do inicjalizacji interfejsu TWI. Status TWI zwracany jest w rejestrze TWSR w bitach o numerach 7…3, dlatego pozostałe 3 bity (2…0) zostały zamaskowane liczbą 0xF8 (0b11111000). Kody statusów TWI są szczegółowo opisane w dokumentacji CPU. Działanie funkcji przesyłającej 1 bajt danych (**listing 4**) polega na zapisaniu bajtu w rejestrze TWDR, a następnie włączeniu interfejsu sprzętowego.

Przy odczycie danych wyróżnia się 2 tryby: z potwierdzeniem (ACK) lub bez potwierdzenia (NACK). Sterowanie opcją ACK w interfejsie sprzętowym TWI zaimplementowanym przez Atmel odbywa się za pomocą bit TWEA. Jeśli bit jest ustawiony 1, wówczas zostanie wygenerowany sygnał ACK. Odczyt danych z opcjami ACK oraz NACK pokazano na **listingu 5**.

Transmisję danych kończy wysłanie sygnału *Stop*, który zwalnia linie transmisyjne i umożliwia przesyłanie danych przez inne urządzenia dołączone do I²C lub nawiązanie nowego połączenia. Funkcję wysyłającą sekwencję *Stop* pokazano na **listingu 6**. Kolejne funkcje służące do transmisji danych wykonano na podstawie dokumentacji układu LSM9DS0



Rysunek 1. Schemat ideowy płytki z sensorem LSM9DS0



Rysunek 2. Schemat montażowy płytki z sensorem LSM9DS0

Tabela 1. Wybór interfejsu komunikacyjnego za pomocą J1

CS_G / CS_XM	Interfejs G	Interfejs XM
0	SPI	SPI
1	I²C	I²C

Tabela 2. Wybór funkcji za pomocą zwory J2

SDO_G/ SDO_XM	SPI	I²C
SDO_XM	Seria Data Output (SDO) dla XM	Konfiguracja adresu XM
SDO_G	Seria Data Output (SDO) dla G	Konfiguracja adresu G

Tabela 3. Opis wyprowadzeń modułu akcelerometru

Nazwa	Opis funkcji
SDA	Linia SDA dla I²C
SCL	I²C – Linia SCL SPI – Linia SPC
SDO_G	I²C – Wybór adresu SPI – Linia SDO dla żyroskopu
SDO_XM	I²C – Wybór adresu SPI – Linia SDO dla akcelerometru i magnetometru
INT2_XM	Linia przerwania nr 2 dla akcelerometru i magnetometru
INT1_XM	Linia przerwania nr 1 dla akcelerometru i magnetometru
DRDY_G	Linia informująca o gotowości danych do odczytu z żyroskopu
CS_XM	Wybór interfejsu komunikacyjnego dla akcelerometru i magnetometru
CS_G	Wybór interfejsu komunikacyjnego dla żyroskopu
INT_G	Linia programowalnego przerwania dla żyroskopu
DEN_G	Włącznik danych żyroskopu
GND	GND
ACC	Zasilanie 2,4 – 3,6 V

oraz ATmega8. Pokazano jest na **listingu 7**. Odczyt rozpoczyna się od wysłania na poprzez interfejs TWI (I²C) adresu urządzenia slave (*sla*) wraz z ustawioną flagą *Write*. Po tej operacji układ slave oczekuje na otrzymanie od master kolejnego bajtu, tym razem z adresem komórki pamięci do odczytu (*adr*). Następnie, jest wykonywane odwrócenie kierunku komunikacji poprzez wykonanie rozkazu *Repeated Start* oraz ponownie przesłanie poprzez TWI adresu układu slave, lecz tym razem z ustawioną flagą *Read*. Należy zwrócić uwagę na to, że na każdym etapie komunikacji następuje weryfikacja kodów kontrolnych opisanych w dokumentacji mikrokontrolera. W przypadku wystąpienia błędu na dowolnym etapie wymiany danych odczyt jest dyskwalifikowany, a funkcja zwraca wartość 0. Operacja zapisu bajtu do urządzenia slave jest analogiczna do powyższej, lecz tym razem nie ma konieczności zmiany kierunku komunikacji. Dzięki ustawieniu siódmego bitu w chwili przekazywania adresu (*adr*) zostaje uruchomiona automatyczna inkrementacja adresów do odczytu. Za pomocą tego mechanizmu jest możliwe odczytywanie wielu następujących po sobie rejestrów, dopóki potwierdzenie ACK jest włączone. Nie ma wówczas konieczności przesyłania adresów w kolejnych odczytach (**listing 8**).

Dla ułatwienia analizy, na **listingu 9** umieszczono deklaracje nagłówków opisywanych funkcji.

bit nr 7	6	5	4	3	2	1	0
DR1	DR0	BW1	BW0	PD	Zen	Yen	Xen

DR1..DR0	Wybór częstotliwości pracy
BW1..BW0	Ustawienie szerokości pasma
PD	Włącznik żyroskopu (0: wyłączony, 1: praca normalna lub uśpienie)
Zen	Włącznik osi Z (0: oś Z wyłączona; 1: oś Z włączona)
Yen	Włącznik osi Y (0: oś Y wyłączona; 1: oś Y włączona)
Xen	Włącznik osi X (0: oś X wyłączona; 1: oś X włączona)

Rysunek 3. Rejestr kontrolny akcelerometru CTRL_REG1_G

Listing 1. Definicje ułatwiające zrozumienie działania programu
```
#define I2C_READ 0x01
#define I2C_WRITE 0x00
#define I2C_TWINTSET (TWCR & (1<<TWINT))
#define I2C_RETTWSR (TWSR & 0xF8)
#define I2C_ILOOP 255
```

Listing 2. Ustawienie częstotliwości sygnału SCL
```
//Inicjalizacja Master dla CPU 8 MHz
void i2c_init(void)
{
    //SCL freq= F_CPU/(16+2(TWBR)*4^TWPS)
    TWBR=8; //8:100KHz
    TWSR=(0<<TWPS1)|(0<<TWPS0); //Prescaler: 1
}
```

Listing 3. Inicjalizacja interfejsu TWI mikrokontrolera ATmega8
```
uint8_t i2c_start(void)
{
    uint8_t i=I2C_ILOOP;
    //Wyczyszczenie flagi TWI interrupt,
    //wysłanie sygnału start, włączenie TWI
    TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
    //Oczekiwanie na przesłanie sygnału start
    while ((i) && (!I2C_TWINTSET)) i--;
    //TWSR & 0xF8 - zamaskowanie preskalera w TWSR
    if (I2C_TWINTSET) return I2C_RETTWSR; else return 0;
}
```

Listing 4. Funkcja przesyłająca pojedynczy bajt za pomocą TWI
```
uint8_t i2c_write_data(uint8_t data)
{
    uint8_t i=I2C_ILOOP;
    TWDR=data; // put data in TWDR
    TWCR=(1<<TWINT)|(1<<TWEN);
    while ((i) && (!I2C_TWINTSET)) i--;
    if (I2C_TWINTSET) return I2C_RETTWSR; else return 0;
}
```

Listing 5. Odczyt danych z potwierdzeniem (ACK)
```
uint8_t i2c_read_data_ack(uint8_t *data)
{
    uint8_t i=I2C_ILOOP;
    TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWEA);
    while ((i) && (!I2C_TWINTSET)) i--;
    if (I2C_TWINTSET) {
        *data = TWDR;
        return I2C_RETTWSR;
    }
    else return 0;
}

uint8_t i2c_read_data_nack(uint8_t *data) {
    uint8_t i=I2C_ILOOP;
    TWCR=(1<<TWINT)|(1<<TWEN);
    while ((i) && (!I2C_TWINTSET)) i--;
    if (I2C_TWINTSET) {
        *data = TWDR;
        return I2C_RETTWSR;
    }
    else return 0;
}
```

Adresowanie LSM9DS0

Każde urządzenie typu slave dołączone do I²C musi mieć unikalny adres. Układ LSM9DS0 funkcjonalnie podzielono na dwa urządzenia. Pierwsze z nich udostępnia dane z żyroskopu i oznaczono G. Drugie – oznaczone XM – dostarcza dane z akcelerometru oraz magnetometru. Producent przewidział możliwość zmiany domyślnych adresów. Można to zrobić za pomocą odpowiednich poziomów na wejściach SDO_G i SDO_XM. W tabeli 4 podano adresy interfejsów sensorów wbudowanych w strukturę LSM9DS0.

Konfigurowanie LSM9DS0

Konfigurowanie układu sensora jest wykonywane poprzez zapisanie rejestrów konfiguracyjnych. Zostały one podzielone na część

Tabela 4. Zależność adresu sensorów wbudowanych w LSM9DS0 od poziomu na doprowadzeniach SDO_G, SDO_XM

SDO_G / SDO_XM	G	XM
0	0x6A	0x1E
1	0x6B	0x1D

Wykaz elementów

Rezystory:
R1…R6: 10 kΩ (SMD 0805)
Kondensatory:
C1: 4,7 µF (SMD 0805)
C2: 220 nF (SMD 0805)
C3: 10 µF/16 V (SMD „B")
C4: 100 nF (SMD 0805)
Półprzewodniki:
IMU: LSM9DS0
Inne:
JP1…JP3: listwa goldpin (11 wypr.)

odpowiedzialną za funkcjonowanie żyroskopu (G) oraz część zawierającą nastawy akcelerometru i magnetometru (XM). Sposób konfigurowania układu żyroskopu zademonstrowano na przykładzie rejestru żyroskopu CTRL_REG1_G.

Zawartość rejestru CTRL_REG1_G pokazano na **rysunku 3**.

Możliwe kombinacje bitów DR1..BW0 wymienione zostały w kolejnej tabeli w dokumentacji. Zgodnie z tymi informacjami, aby włączyć żyroskop i ustawić jego częstotliwość pracy należy zapisać do rejestru CTRL_REG1_G przykładową wartość: 0b10111111 (0xBF), co oznacza:

10 11	1	1 1 1
380Hz / 100 Cutoff	Żyroskop włączony	Osie Z, Y i X włączone

Wykorzystując wcześniej opisane funkcje, konfigurowanie sensora G sprowadza się do zapisania liczby 0xBF do rejestru CTRL_REG1_G w urządzeniu G:

i2c_write_byte(0x6B, 0x20, 0xBF);

Liczba 0x6B jest adresem slave sensora G (pin SDO_G jest ustawiony), natomiast 0x20 to adres rejestru CTRL_REG1_G.

Odczytywanie danych z LSM9DS0

Dane z sensorów udostępniane są do odczytu z pamięci wewnętrznej IMU w 3 grupach. Każda grupa składa się z sześciu 8-bitowych rejestrów, które zawierają dane dla osi X, Y i Z w parach komplementarnych. Przykładowo, mierzone prędkości kątowe z żyroskopu udostępniane są w 6 następujących po sobie komórkach pamięci o adresach:

Dla osi X: 28h, 29h.
Dla osi Y: 2Ah, 2Bh.

Listing 6. Funkcja przesyłająca sekwencję *Stop*

```
uint8_t i2c_stop(void) {
    uint8_t i=I2C_ILOOP;
    TWCR=(1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
    while ((i) && (!I2C_TWINTSET)) i--;
    if (I2C_TWINTSET) return I2C_RETTWSR; else return 0;
}
```

Listing 7. Funkcje przesyłające dane za pomocą TWI (I^2C)

```
uint8_t i2c_read_byte(uint8_t sla, uint8_t adr, uint8_t *data)
{
    //Start
    if (i2c_start() != 0x08) { i2c_stop(); return 0; }
    //Wysłanie adresu urządzenia SLAVE + WRITE
    if (i2c_write_data((sla<<1) | I2C_WRITE) != 0x18)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie subadresu
    if (i2c_write_data(adr) != 0x28 )
    {
        i2c_stop();
        return 0;
    }
    //Repeated Start
    if (i2c_start() != 0x10)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie adresu urządzenia SLAVE + READ
    if (i2c_write_data((sla<<1) | I2C_READ) != 0x40)
    {
        i2c_stop();
        return 0;
    }
    //Odczyt danych
    if (i2c_read_data_nack(data) != 0x58)
    {
        i2c_stop();
        return 0;
    }
    i2c_stop();
    return 1;
}

uint8_t i2c_write_byte(const uint8_t sla, const uint8_t adr, uint8_t data)
{
    //Start
    if (i2c_start() != 0x08)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie adresu urządzenia SLAVE + WRITE
    if (i2c_write_data((sla<<1) | I2C_WRITE) != 0x18)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie subadresu
    if (i2c_write_data(adr) != 0x28 )
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie bajtu
    if(i2c_write_data(data) != 0x28 )
    {
        i2c_stop();
        return 0;
    }
    i2c_stop();
    return 1;
}
```

Dla osi Z: 2Ch, 2Dh.

Do odczytania wskazań danej osi konieczne jest pobranie obu komplementarnych komórek. Przykładowy odczyt pomiarów prędkości kątowych z żyroskopu polega na odczycie

Tabela 5. Opis rejestrów konfiguracyjnych G

Adres	Nazwa	Przeznaczenie
0x20	CTRL_REG1_G	Za pomocą pierwszego z rejestrów ustala się częstotliwość wykonywania pomiarów: 95 Hz, 190 Hz, 380 Hz, 760 Hz. Bity 3…0 są odpowiedzialne za włączanie i wyłączanie sensora G.
0x21	CTRL_REG2_G	Konfigurowanie filtru górnoprzepustowego.
0x22	CTRL_REG3_G	Rejestr umożliwia sterowanie sygnałami na doprowadzeniach układu scalonego w zależności od zdarzeń. Bit I2_DRDY powoduje ustawienie wyprowadzenia DRDY_G, jeśli sensor G jest gotowy do odczytu.
0x23	CTRL_REG4_G	W tym rejestrze konfigurowana jest czułość żyroskopu w zakresach: 245, 500, 2000 dps (dps = stopnie na sekundę) oraz jest ustalana kolejność bajtów 16-bitowego wyniku pomiaru.
0x24	CTRL_REG5_G	Rejestr 5 zawiera między innymi włącznik kolejkowania FIFO, filtrowania górnoprzepustowego. Umożliwia także wyczyszczenie pamięci IMU (reboot).

sześciu następujących po sobie rejestrów poprzez wywołanie funkcji odczytu sekwencyjnego:

```
uint8_t dane[6];
i2c_read_sequence(0x6B, 0x28, dane, 6);
```

Liczba 0x28 jest adresem rejestru OUT_X_L_G – pierwszego z serii rejestrów zawierających dane X, Y oraz Z. Dane zostają zwrócone w formie 8-bitowych liczb w tabeli *dane[]*. Finalne 16-bitowe odczyty otrzymywane są po przekształceniu komplementarnych par za pomocą operacji bitowych:

```
int16_t GX = (dane[1]<<8) | dane[0];
int16_t GY = (dane[3]<<8) | dane[2];
int16_t GZ = (dane[5]<<8) | dane[4];
```

Należy pamiętać, iż zwracane wartości są **typu całkowitego ze znakiem**. Dla uzupełnienia przykładu w **tabeli 5** zamieszczono opis rejestrów konfiguracyjnych G, natomiast w **tabeli 6** rejestrów konfiguracyjnych rejestru XM.

Listing 8. Sekwencyjny odczyt i sekwencyjny zapis danych

```
uint8_t i2c_read_sequence(uint8_t sla, uint8_t adr, uint8_t dest[], uint8_t
cnt)
{
    //Start
    if (i2c_start() != 0x08) { i2c_stop(); return 0; }
    //Wysłanie adresu urządzenia SLAVE + WRITE
    if (i2c_write_data((sla<<1) | I2C_WRITE) != 0x18)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie subadresu + autoincrement
    if (i2c_write_data(adr | (1<<7)) != 0x28)
    {
        i2c_stop();
        return 0;
    }
    //Repeated Start
    if (i2c_start() != 0x10)
    {
        i2c_stop();
        return 0;
    }
    //Wysłanie adresu urządzenia SLAVE + READ
    if (i2c_write_data((sla<<1) | I2C_READ) != 0x40)
    {
        i2c_stop();
        return 0;
    }
    //Odczyt danych
    for (uint8_t i=0; i<cnt; i++)
    {
        if (i < cnt-1)
        {
            //Odczyt z ACK
            if (i2c_read_data_ack(&dest[i]) != 0x50)
            {
                i2c_stop();
                return 0;
            }
        }
        else
        {
            //Odczyt ostatniego bajtu z NACK
            if (i2c_read_data_nack(&dest[i]) != 0x58)
            {
                i2c_stop();
                return 0;
            }
        }
    }
    i2c_stop();
    return 1;
}
```

Listing 9. Deklaracje nagłówków funkcji obsługi TWI

```
void i2c_init(void);
uint8_t i2c_start(void);
uint8_t i2c_write_data(uint8_t data);
uint8_t i2c_read_data_ack(uint8_t *data);
uint8_t i2c_read_data_nack(uint8_t *data);
uint8_t i2c_stop(void);
uint8_t i2c_read_byte(uint8_t sla, uint8_t adr, uint8_t *data);
uint8_t i2c_write_byte(uint8_t sla, uint8_t adr, uint8_t data);
uint8_t i2c_read_sequence(uint8_t sla, uint8_t adr, uint8_t dest[], uint8_t
cnt);
```

Kolejkowanie FIFO

Układ LSM9DS0 umożliwia odczytywanie danych z buforów kolejkowych FIFO o pojemności 32 wyników pomiarów (slotów) z żyroskopu i akcelerometru dla osi X, Y i Z. Bufory mogą działać w 5 różnych trybach.

FIFO Bypass. W tym trybie bufor przechowuje tylko dane tylko z ostatniego odczytu z sensorów, które są bezpowrotnie tracone w chwili nadpisania przez kolejne odczyty.

FIFO. Dane sensorów X, Y i Z zapisywane są w kolejnych komórkach kolejki do chwili, gdy wszystkie 32 sloty zostaną wykorzystane. Wówczas następuje zatrzymanie odczytów do czasu przywrócenia trybu Bypass.

Stream. W tym trybie dane zapisywane są w kolejce, dopóki wszystkie 32 sloty zostaną zapełnione. Gdy to się stanie następuje nadpisywanie najstarszych danych przez nowe odczyty.

Stream-to-FIFO. Dane kolekcjonowane są w trybie Stream do czasu wystąpienia zdarzenia zdefiniowanego w rejestrze INT1_CFG_G.

Bypass-to-Stream. Ten tryb zakłada przejście ze stanu Bypass do Stream po zajściu zdarzenia kontrolowanego poprzez rejestr INT1_CFG_G.

Podsumowanie

Mam nadzieję, że podane informacje oraz projekt modułu ułatwią jego samodzielne wykorzystanie. Wkrótce w miesięczniku EP opublikuję praktyczny przykład zastosowania układu LSM9DS0 do zdalnego sterowania kamerą.

Arkadiusz Witczak

Tabela 6. Opis rejestrów konfiguracyjnych XM

Adres	Nazwa	Przeznaczenie
0x1F	CTRL_REG0_XM	Uruchomienie procesu czyszczenia pamięci XM (reboot). Rejestr umożliwia także włączenie kolejki FIFO oraz filtrów górnoprzepustowych.
0x20	CTRL_REG1_XM	Włączenie sensora XM oraz konfigurowanie częstotliwości wykonywania pomiarów: 3,125 Hz, 6,25 Hz, 12,5 Hz, 25 Hz, 50 Hz, 100 Hz, 200 Hz, 400 Hz, 800 Hz, 1600 Hz.
0x21	CTRL_REG2_XM	Bity 5..3 odpowiedzialne są za wybór skali odczytów z akcelerometru spośród następujących zakresów: ±2g, ±4g, ±6g, ±8g, ±16g. Dostępna jest także opcja antyaliasingu.
0x22	CTRL_REG3_XM	Bity sterujące generatorami przerwań. Wśród nich są bity P1_DRDYA i P1_DRDYM odpowiedzialne za generowanie przerwania od akcelerometru lub magnetometru (INT1_XM).
0x23	CTRL_REG4_XM	Bity sterujące przerwaniami: P2_DRDYA i P2_DRDYM. Załączają przerwanie od sensora XM (INT2_XM).
0x24	CTRL_REG5_XM	Rejestr umożliwia włączenie lub wyłączenie termometru oraz steruje częstotliwością pomiarów wykonywanych przez magnetometr: 3,125 Hz, 6,25 Hz, 12,5 Hz, 50 Hz, 100 Hz. Częstotliwość 100 Hz jest możliwa do uzyskania przy ustawieniu wartości powyżej 50 Hz w rejestrze CTRL_REG1_XM
0x25	CTRL_REG6_XM	Wybór zakresu magnetometru: ±2/±4/±8/±12 Gs.
0x26	CTRL_REG7_XM	Wybór trybu pracy magnetometru.