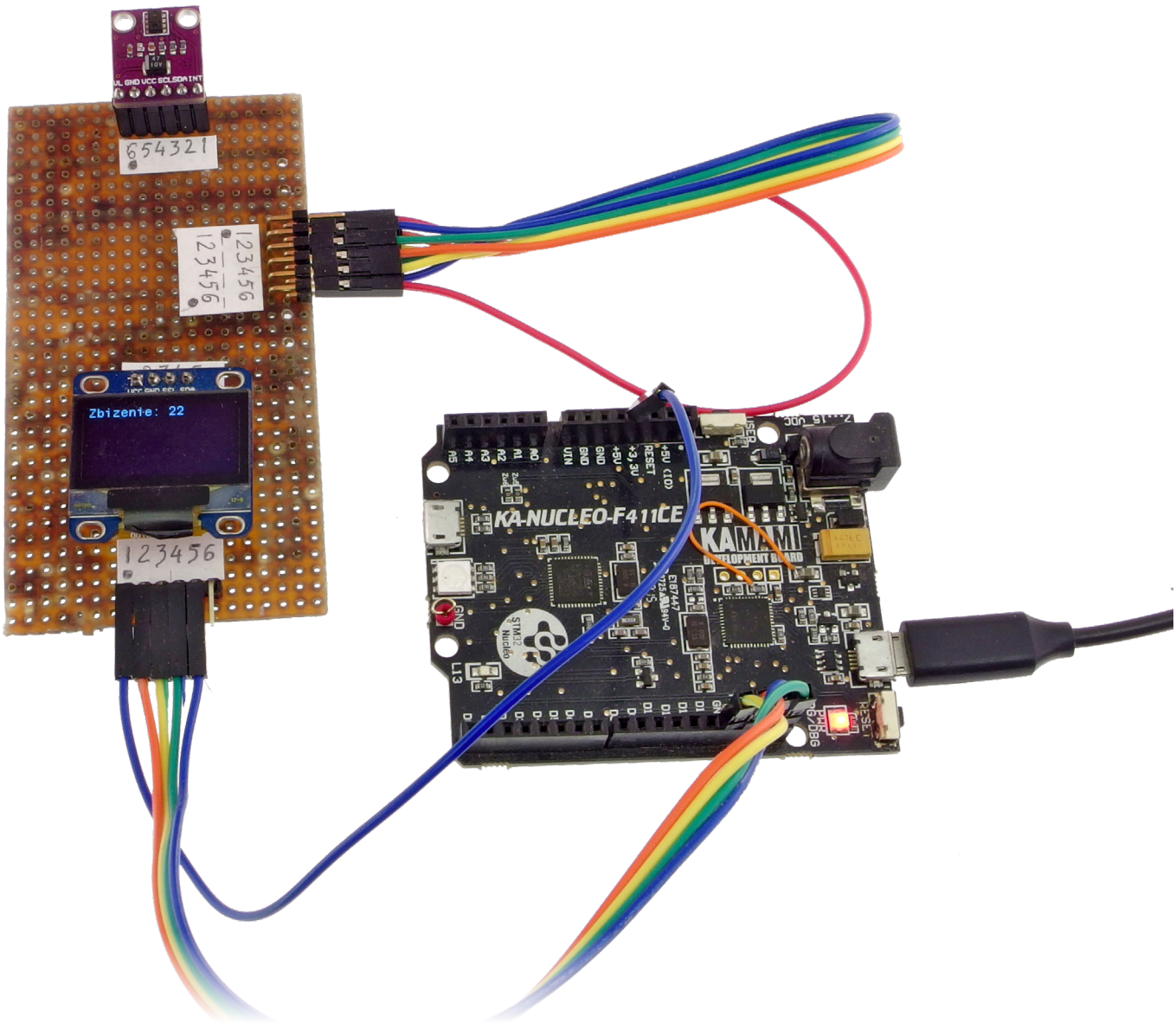


Czujnik światła z układem APDS9930

Miniaturowy czujnik APDS9930 realizuje dwie funkcje: pomiar natężenia oświetlenia i detekcję zbliżania obiektu na dystansie do 10 cm. Małe wymiary, wysoka czułość i niski pobór prądu sugerują, że skonstruowano go z przeznaczeniem do użycia w aparatach telefonii komórkowej. Montaż w formie modułu ułatwia podłączenie i pozwala skorzystać z funkcji czujnika. Sześć styków szpilkowych daje dostęp do magistrali sterującej oraz do sygnału przerwania.

Fotografia 1. Wygląd modułu APDS9930

Parametry modułu

Widok płytki modułu został pokazany na **fotografii 1**. Główne parametry zamontowanego na module czujnika są następujące:

- wykrywanie natężenia oświetlenia (funkcja ALS) już od poziomu 0,01 luxa,
- wykrywanie obecności obiektu na dystansie od 0 do około 10 cm przy pomocy impulsów IR,
- komunikacja za pośrednictwem magistrali I²C o szybkości transmisji do 400 kHz,

- wyjście sygnalizujące przerwanie wywoływane poziomem oświetlenia albo wykryciem obiektu,
- rozdzielone zasilanie czujnika 2,2...3,6 V/250 µA od zintegrowanej we wspólnej obudowie diody IR 3...4,5 V/100 mA generującej impulsy używane do wykrycia obiektu,
- fabrycznie skalibrowana detekcja dystansu,
- wewnętrzny regulator prądu diody IR o programowalnej wydajności.

Funkcje wyprowadzeń modułu:

- VL – podłączenie zewnętrznego zasilania diody IR czujnika. Nie ma potrzeby stosowania dodatkowego opornika ograniczającego prąd diody,
- VCC, GND – podłączenie zasilania czujnika,
- SCL, SDA – linie magistrali sterującej I²C,
- INT – wyjście sygnału przerwania.

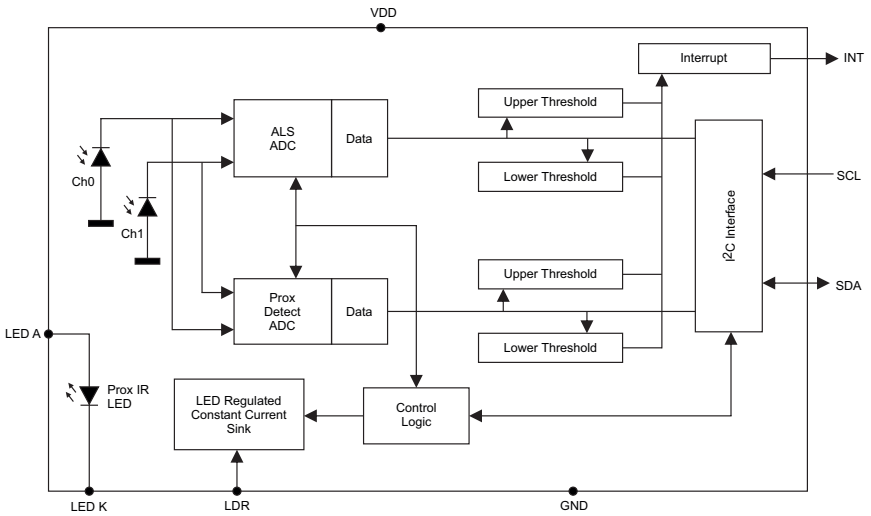
Budowa czujnika

Na **rysunku 1** pokazano schemat blokowy czujnika APDS9930. W części odbiorczej pracują dwie fotodiody CH0 i CH1. Pierwsza z nich jest czuła na światło w szerokim zakresie od promieniowania widzialnego do podczerwieni, druga jest ograniczona do zakresu IR. Dzięki temu przy pomiarze natężenia oświetlenia można wyeliminować wpływ składowej podczerwonej i uzyskać charakterystykę czułości zbliżoną do tej jaką ma oko człowieka. Diody są podłączone zarówno do bloku pomiaru natężenia oświetlenia ALS (*ambient light sensing*) jak i do bloku detekcji zbliżenia (*Proxy Detect*).

Detekcja działa na zasadzie odbicia impulsów promieniowania podczerwonego od zbliżającego się obiektu. Krótkie impulsy emituje umieszczona wewnątrz obudowy czujnika dioda Prox IR LED. Prąd diody i intensywność impulsów ustala regulator, którego wyprowadzenie LDR jest zewnętrznie łączone z wyprowadzeniem LED K diody. Do anody LED A podłącza się zewnętrzne napięcie zasilania, które może być oddzielone od zasilania pozostałych bloków czujnika.

O tym, który blok jest aktywny: pomiar oświetlenia czy detekcja zbliżania, decyduje użytkownik przez ustawienia rejestrów sterujących czujnika. Użytkownik może włączyć przerwanie podłączone do wyprowadzenia INT. A także ustawić progi jego zadziałania zarówno dla pomiaru oświetlenia jak i detekcji zbliżania.

Dostęp do rejestrów czujnika i wymiana danych odbywa się za pośrednictwem magistrali I²C. Czujnik APDS9930 reaguje na adres 0x39 (konwencja 7 bitowego adresu). W trybie zapisu dostęp do rejestru następuje po wysłaniu adresu (Slave Address) i numeru rejestru (Command Code). W trybie odczytu należy wysłać adres, numer rejestru, sekwencję I²C Start i ponownie adres (dodatkowy bit kierunku powinien być ustawiony) – czujnik prześle wtedy zawartość rejestru.



Rysunek 1. Schemat blokowy

Biblioteka

Istnieją biblioteki do obsługi czujnika APDS9930. Pod internetowym adresem ***https://github.com/Depau/APDS9930*** udostępniona jest biblioteka dla Arduino wraz z kilkoma przykładami zastosowania, a także opisem sposobu podłączenia modułu do płytki Arduino. Na bibliotekę składa się szereg metod pozwalających kontrolować działanie czujnika APDS9930 w różnych trybach pracy:

- Procedura *init()* ustawia czujnik do pracy z wartościami domyślnymi zapisanymi w pliku nagłówkowym *APDS9930.h* w sekcji Default values.
- Procedura *enableLightSensor(bool interrupts)* włącza pomiar natężenia oświetlenia z uruchomionym (*true*) lub nie (*false*) trybem przerwania.
- Procedura *enableProximitySensor(bool interrupts)* włącza tryb detekcji zbliżenia obiektu, który może być dodatkowo sygnalizowany sygnałem przerwania.
- Do odczytu pomiaru natężenia oświetlenia w luksach służy procedura *readAmbientLightLux(float &val)*. Z kolei procedura *readProximity(uint16_t &val)* pozwala odczytać wartość liczbową proporcjonalną do bliskości obiektu gdy czujnik pracuje w trybie detekcji zbliżenia. Wartość 0 oznacza brak detekcji, wartość 1023 maksymalne zbliżenie.

Listing 1. Uproszczony przykład programu, kiedy czujnik pracuje w trybie wykrywania zbliżenia obiektu

```
#include <APDS9930.h>

APDS9930 apds = APDS9930();
uint16_t proximity_data = 0;

void setup(){
    Serial.begin(9600);
    apds.init()

    //Start APDS-9930 tryb zbliżeniowy (bez przerwania)
    if ( apds.enableProximitySensor(false) ) {
        Serial.println(F("Sensor pracuje"));
    } else {
        Serial.println(F("Błąd inicjacji!"));
    }

    //parametr PPULSE
    apds.wireWriteDataByte(APDS9930_PPULSE, 4);
    //parametr POFFSET
    apds.wireWriteDataByte(APDS9930_POFFSET, -150);
}

void loop() {
    //Odczyt wartości trybu zbliżenia
    if ( !apds.readProximity(proximity_data) ) {
        Serial.println("Błąd odczytu");
    } else {
        Serial.print("Zbliżenie: ");
        Serial.println(proximity_data);
    }

    //Pauza 250 ms przed następnym odczytem
    delay(250);
}
```

Istnieją także procedury dostępu do określonych rejestrów w tym rejestrów określających progi zadziałania wyjścia przerwania:

- setProximityIntLowThreshold(uint16_t threshold),
- setProximityIntHighThreshold(uint16_t threshold),
- *setLightIntLowThreshold(uint16_t threshold)*,
- *setLightIntHighThreshold(uint16_t threshold)*.

Na **listingu 1** został pokazany uproszczony przykład programu, kiedy czujnik pracuje w trybie wykrywania zbliżenia obiektu. Po inicjacji następuje włączenie trybu zbliżeniowego. Dobierając wartość współczynnika PPULSE w zakresie 1...255 określa się sposób reakcji czujnika. Na **rysunku 2** pokazano jak różne wartości współczynnika wpływają na wynik pomiaru w funkcji rzeczywistej odległości obiektu od czujnika. Dobranie wartości POFFSET

eliminuje szumy odczytu gdy obiekt znajduje się poza zakresem. Następnie w pętli odczytywany jest współczynnik, którego wartość rosnąca od 0 do 1023 oznacza stopniowe zbliżanie się obserwowanego obiektu do czujnika.

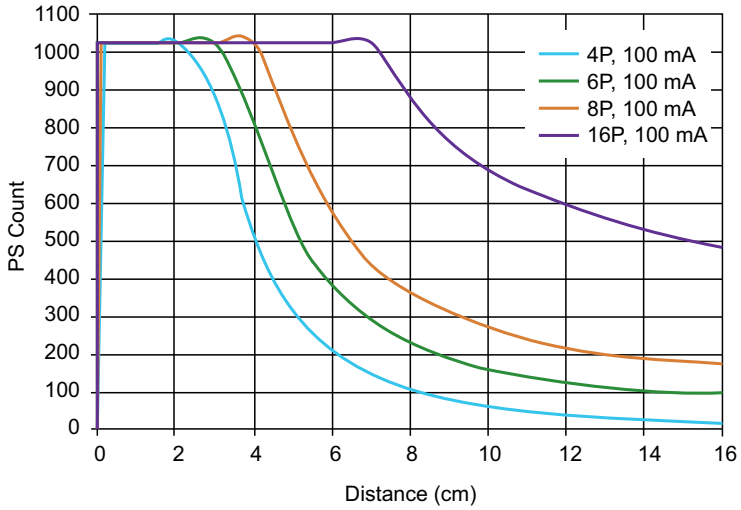
APDS9930 i Nucleo

Czujnik można podłączyć do dowolnej platformy sprzętowej o ile pracuje z poziomami logicznymi 0/3,3 V, potrafi obsłużyć transmisję I²C i jest w stanie dostarczyć zasilanie do czujnika. Warunki te spełnia dowolna płytka Nucleo z kontrolerem STM32. W **tabeli 1** zestawiono potrzebne połączenia dla trybu pracy bez wykorzystania wyjścia przerwania.

Niezbędnym krokiem jest dostosowanie biblioteki czujnika APDS9930 do współpracy z płytką Nucleo. Jedna z możliwości to skorzystanie z projektu STM32Duino. Są to biblioteki „tłumaczące" oprogramowanie pisane dla Arduino na kod wykonywalny dla płyt z kontrolerami rodziny STM32. Inną możliwością jest użycie środowiska programistycznego przeznaczonego dla STM-ów i dostosowanie biblioteki czujnika. W tym drugim podejściu trzeba dostosować procedury odwołujące się bezpośrednio do warstwy sprzętowej komunikującej się z interfejsem I²C. W bibliotece APDS9930 do komunikacji z magistralą I²C zastosowano kilka procedur biblioteki Arduino – *Wire.c*. Są to:

- *begin()* – ustawienia inicjujące bibliotekę Wire,
- *requestFrom()* – odczyt do buforu z układu o podanym adresie żądanej liczby bajtów,
- *beginTransmission()* – przygotowanie do wysłania określonej liczby bajtów z bufora do układu,
- *endTransmission()* – zakończenie wysyłania bajtów do układu,
- *write()* – realizacja wysłania bajtów,
- *available()* – zwraca ilość odebranych bajtów oczekujących w buforze,
- *read()* – odczyt z bufora odbiorczego pojedynczego bajtu.

Należy pamiętać, że w bibliotece Wire operacje przesyłania danych pomiędzy układem nadrzędnym a podporządkowanym (w tym przypadku modułem APDS9930) odbywają się za pośrednictwem 32-bitowych buforów: nadawczego i odbiorczego. Należy zatem napisać własne procedury, które będą realizowały wymienione funkcje, korzystając z dostępu do sprzętowego interfejsu kontrolera STM32 zamontowanego na użytej płytce Nucleo.



Rysunek 2. Charakterystyki detekcji obiektu

Listing 2. Kod procedury odczytu bajtów z układu podrzędnego

```
extern I2C_HandleTypeDef hi2c1;

//odczytuje ilość (quantity) bajtów z układu podrzędnego o adresie (slave_adr)
//zwraca ilość odebranych bajtów w buforze
uint8_t PWire::requestFrom(uint8_t slave_adr, uint8_t quantity) {
    HAL_StatusTypeDef status;

    status =HAL_I2C_Master_Receive
        (&hi2c1, (uint16_t)slave_adr, (uint8_t *)buforRec_Wire,
        (uint16_t)quantity, 1000);

    if (stat !=HAL_OK) quantity = 0;

    ile_w_buforRec_Wire = quantity;
    poz_odczytu_w_buforRec_Wire = 0;

    return quantity;
}
```

Na **listingu 2** można zobaczyć kod procedury odczytu bajtów z układu podrzędnego realizującej funkcję *requestFrom()*. Procedura jest jedną z funkcji klasy PWire, która zastąpi arduinową klasę Wire. Jeżeli moduł podłączony zostanie do wyprowadzeń tak jak podano w tabeli, to obsługą magistrali I²C zajmie się sprzętowy interfejs I2C1. Wykorzystywana jest standardowa procedura HAL-a *HAL_I2C_Master_Receive()*, a także statyczny bufor odbiorczy buforRec_Wire[] o rozmiarze 32 bajtów.

W samej bibliotece APDS9930 potrzebne będą niewielkie zmiany. Dotyczą głównie jej początkowej części i włączanych plików nagłówkowych co pokazano na **listingu 3**.

Listing 3. Niewielkie zmiany potrzebne w bibliotece APDS9930

```
// #include <Arduino.h>
#include <PWire.h>      //zmienione
#include "APDS9930.h"
#include "algorithm"    //dodane
using namespace std;    //dodane
#include <stdio.h>      //dodane

PWire Wire;             //dodane
```

Przede wszystkim należy usunąć odwołanie do biblioteki *Arduino.h* i zastąpić plik *Wire.h* włączeniem stworzonej własnej biblioteki *PWire.h*. Kolejne pliki nagłówkowe i deklaracja przestrzeni nazw są potrzebne do realizacji niektórych funkcji biblioteki APDS9930. Należy także pozbyć się kilku odwołań do nieobsługiwanej teraz procedury *Serial.println()*.

Ponieważ biblioteka APDS9930 napisana jest w C++, najwygodniej używać ją w projektach dla tego samego języka. Co prawda popularne narzędzie STM32CubeMX generuje projekty w C ale przekształcenie do wersji w C++ nie powinno stanowić wielkiego problemu. W wersji środowiska programistycznego SW4STM32 po wczytaniu utworzonego projektu należy wybrać opcję Project Explorer → prawy przycisk myszy → Convert to C++. Należy także zmienić nazwę pliku *main.c* na *main.cpp*.

Ryszard Szymaniak
biuro@ars.info.pl

Tabela 1. Połączenia potrzebne dla trybu pracy bez wykorzystania wyjścia przerwania

Czujnik APDS9930	Nucleo	Opis
SDA	PB9	linia SDA magistrali I2C1
SCL	PB8	linia SCL magistrali I2C1
Vcc	3,3 V	zasilanie modułu
GND	GND	masa
VL	3,3 V/100 mA	zasilanie diody IR modułu