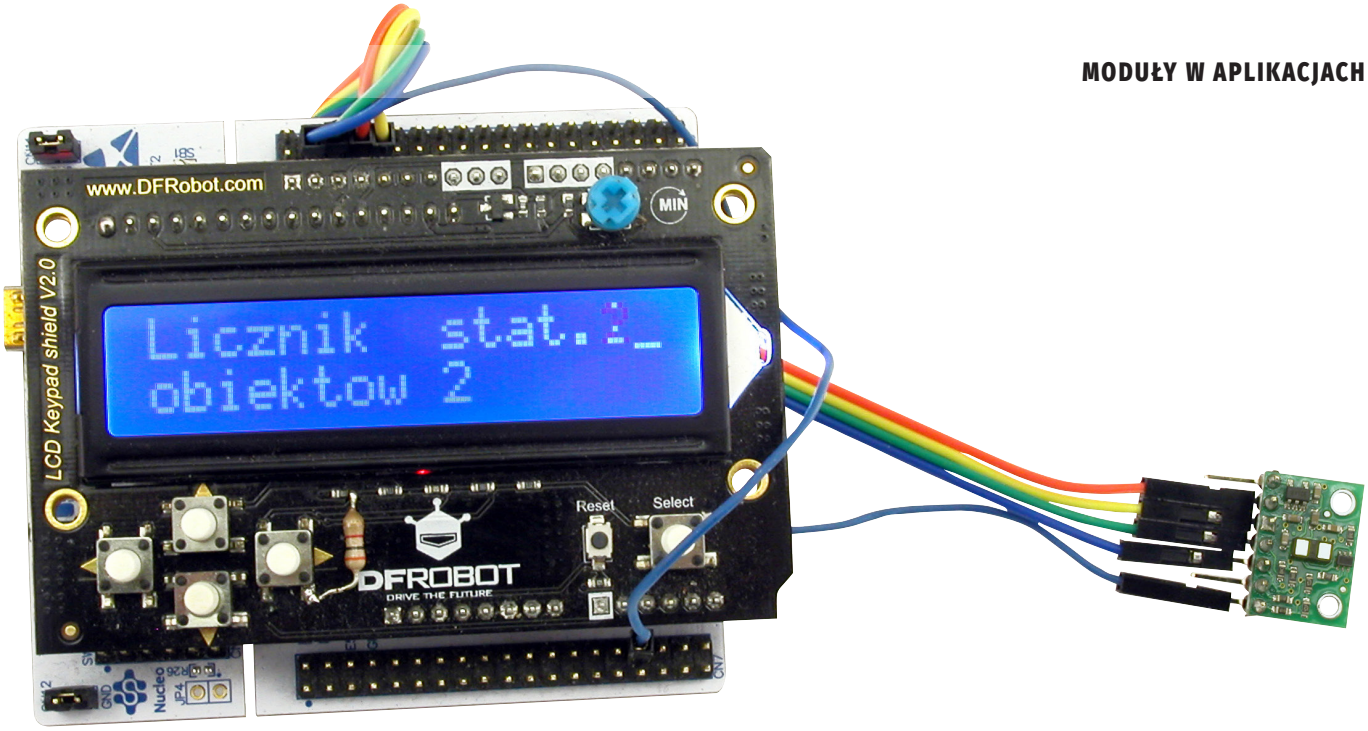


VL53L1X – laserowy sensor

Czujnik odległości działający w technologii lidar

VL53L1X to miniaturowy laserowy dalmierz o zasięgu do 4 m i wysokiej powtarzalności pomiarów. Może znaleźć zastosowanie do pomiarów odległości, zliczania obiektów czy jako prosty detektor gestów. W artykule omówiono sposób działania czujnika oraz możliwości wykorzystania bibliotek i przykładowych programów.

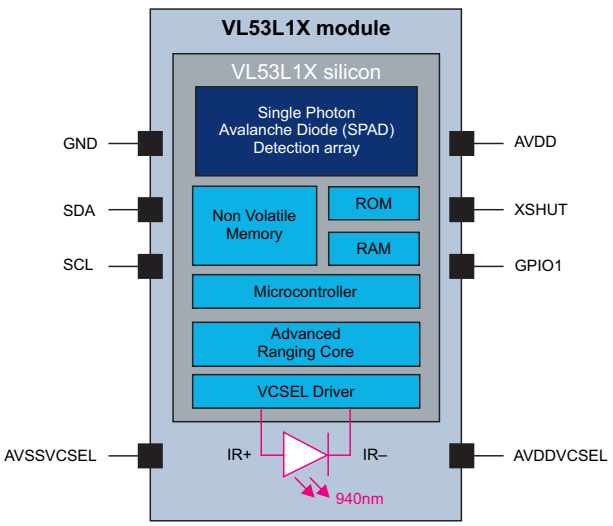
Obudowa sensora VL53L1X ma wymiary 4,9×2,5×1,56 mm, a jej wygląd został pokazany na **fotografii 1**. Wewnątrz mieści się kompletna struktura sensora. Najważniejsze bloki funkcjonalne są zaprezentowane na **rysunku 2**. Dioda LED symbolizuje półprzewodnikowy laser podczerwony małej mocy, wysyłający impulsy pomiarowe. Po odbiciu od obiektu światło lasera pada na matrycę diodową (SPAD). Częścią czujnika jest mikrokontroler, który na podstawie różnicy czasu pomiędzy wyemitowaniem impulsu światła a jego powrotem do matrycy, oblicza odległość sensora od obiektu. Mikrokontroler reaguje na polecenia sterujące przesyłane magistralą **I²C**.

Fotografia 1. Wygląd sensora VL53L1X

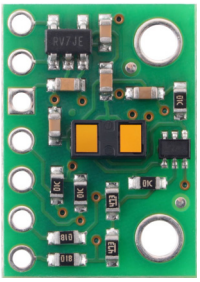
Budowa matrycy odbiorczej umożliwia użycie sensora nie tylko do pomiaru odległości. Możliwe jest śledzenie ruchu obiektu i wykrywanie jego kierunku. Pozwala to wykorzystać sensor do zliczania przesuwających się w polu widzenia przedmiotów lub osób. Na tej samej zasadzie można zbudować prosty detektor gestów reagujący na ruch dłonią w prawą lub lewą stronę.

Najważniejsze parametry sensora VL53L1X:

- trzy zakresy pomiarowe dla odległości od 4 cm do 4 m
- maksymalne pole widzenia (FoV) wynosi 27°
- dioda laserowa klasy 1 (bezpieczna dla wzroku) emituje światło podczerwone o długości fali 940 nm



Rysunek 2. Wewnętrzna struktura sensora



Rysunek 3. Wyprowadzenia modułu VL53L1X Pololu

- do detekcji użyto matrycy 16×16 czułych fotodiod. Obszar odbiorczy matrycy można zmniejszyć (minimum 4×4 fotodiody) co zawęża pole widzenia FoV sensora
- komunikacja poprzez magistralę I2C
- napięcie zasilania od 2,6...3,5 V

Budowa

Do testów i eksperymentów najwygodniej posłużyć się modułem z zamontowanym sensorem VL53L1X, stabilizatorem i układami przesuwania poziomów. Na potrzeby artykułu skorzystano z modułu firmy Pololu (**rysunek 3**), oraz z płytki NUCLEO-F411RE. W **tabeli 1** zestawiono wszystkie konieczne połączenia.

Dokumentacja i biblioteki

Ze względu na budowę sensora, w którym pracuje wewnętrzny procesor, do sterowania VL53L1X nie wykorzystuje się rejestrów ale procedury zebrane w specjalnej bibliotece. Pliki biblioteki producent, firma ST, zamieściła w spakowanym pakiecie STSW-IMG007 dostępnym do pobrania pod adresem [1]. Pakiet oprócz biblioteki zawiera dokument będący opisem (API) jej stosowania.

Dokumentację techniczną samego czujnika można pobrać pod adresem [2]. Przykładowe oprogramowanie wykorzystujące bibliotekę czujnika VL53L1X jest dostępne tu [3].

W pakiecie STSW-IMG009 znajdują się pliki prostego programu do pomiaru odległości, jakości oświetlenia i odczytania statusu. Oprócz tego pakiet zawiera „okrojoną" na potrzeby projektu wersję biblioteki wraz z dokumentacją. Pakiet STSW-IMG010 zawiera przykładowy projekt pokazujący jak zastosować czujnik VL53L1X do zliczania obiektów albo ludzi. Program pozwala nie tylko zliczać obiekty, ale także określać kierunek ich ruchu. Na tej samej zasadzie można zbudować oprogramowanie do detekcji gestów, reagujące na ruch dłoni. Pliki biblioteki napisane są w języku C i można je zaadoptować dla różnych platform programistycznych i kontrolerów. Przykładowe programy jako bazę sprzętową wykorzystują zestaw firmowy ST do testowania czujnika czyli płytę VL53L1X Expansion Board (X-NUCLEO-53L1A1) oraz NUCLEO-F401. W dalszej części artykułu pokażę jak przystosować udostępnione przykłady oprogramowania do pracy z modułem sensora VL53L1X Pololu oraz dowolną płytką NUCLEO (w opisie użyję NUCLEO-F411).

Tworzenie szkieletu oprogramowania oraz dodanie biblioteki VL53L1XM

Szkielet oprogramowania zostanie utworzony przy pomocy konfiguratora STM32CubeMX dla środowiska programistycznego SW4STM32. Dalszy opis dotyczyć będzie płytki NUCLEO-F411, która komunikuje się z modułem VL53L1X Pololu za pośrednictwem magistrali I²C oraz zasila moduł napięciem stabilizowanym 3,3 V. Wykorzystanie innego typu płytki NUCLEO lub DISCOVERY wymaga tylko zmiany ustawienia STM32CubeMX. Konieczne do wykonania połączenia wraz z oznaczeniami portów i numerami złącz zebrano w tabeli 1.

Kolejne kroki z STM32CubeMX będą wyglądać następująco:

- Wybrać docelową płytkę, dla której ma być generowany kod, np. NUCLEO-F411 z zachowaniem predefiniowanych ustawień.
- Ustawienia zegarów: taktowanie z wewnętrznego generatora HSI RC 16 MHz, HCLK 84 MHz
- Na zakładce Pinout & Configuration → Connectivity zaznaczyć jako aktywną magistralę I²C1, pozostawiając domyślne ustawienia parametrów.
- Ustawić funkcję portu PB9 jako linię I2C1_SDA a portu PB8 jako linię I2C1_SCL.
- Ustawić funkcję portu PA4 jako GPIO_EXTI4 z etykietą VL53L1X_INT. Port można połączyć z wyjściem przerwań sensora. Jest to opcjonalna możliwość nie wykorzystywana w dalszych przykładach.
- Na zakładce Pinout & Configuration → Connectivity włączyć port UART2 z szybkością transmisji 115200 bodów. Tym portem będą wysyłane komunikaty tekstowe o pracy oprogramowania.
- Korzystając z zakładki Project Manager wybrać nazwę i katalog docelowy do zapisu wygenerowanych plików projektu oraz ustawić środowisko programistyczne np. SW4STM32.

Po wygenerowaniu szkieletu kodu otwieramy nasze środowisko programistyczne, w tym opisie będzie to SW4STM32. W kolejnych krokach do wygenerowanego kodu trzeba dodać pliki biblioteki sensora VL53L1X. Można przekopiować pliki biblioteczne z jednego z udostępnionych przez firmę ST przykładów. Ja posłużyłem się przykładem STSW-IMG009. W dalszym opisie będą podawane położenia plików do przekopiowania z wewnętrznej struktury tego pakietu.

Kolejne kroki, które trzeba wykonać dodając pliki biblioteki VL53L1X:

- w drzewie projektu i katalogu Src tworzymy podkatalog dla biblioteki. Ja nadałem mu nazwę VL53L1X_Lib. Do utworzonego podkatalogu będą przekopiowywane kolejne pliki,
- z STSW-IMG009\Example\Src\ plik vl53l1_platform.c,
- z STSW-IMG009\Example\Inc\ pliki: vl53l1_platform.h, vl53l1_error_codes.h, vl53l1_types.h, vl53l1_ platform_log.h, vl53l1_ platform_user_config.h,
- z STSW-IMG009\API\core\ pliki: vl53l1_api.c, vl53l1_api.h.

Należy dokonać zmian w zaimportowanym pliku vl53l1_platform.c. Zależnie od użytej płytki trzeba dostosować deklarację typu bibliotek HAL. W przypadku NUCLEO-F411 zmiana będzie wyglądała tak:

```
z: #include „stm32xxx_hal.h" na:
#include „stm32f4xx_hal.h"
```

Usunąć lub zakomentować:

```
//#include „X-NUCLEO-53L1A1.h"
```

Ostatnia konieczna zmiana polega na wskazaniu I²C1 jako magistrali przypisanej do komunikacji z sensorem VL53L1X. Należy zastąpić:

```
#define VL53L0X_pI2cHandle
(&hi2c1)
```

na:

```
extern I2C_HandleTypeDef hi2c1;
#define XNUCLEO53L1A1_hi2c
hi2c1
```

Na koniec należy podać w ustawieniach kompilacji ścieżkę dostępu do utworzonego katalogu biblioteki VL53L1X_Lib. W środowisku SW4STM32 robi to się poprzez Properties → Settings → MCU GCC Compiler → Includes → Include Paths.

W dalszym kroku możemy do tak utworzonego projektu dodawać kod pobrany np. z przykładowych projektów.

Odczyt danych z sensora

Korzystając z przykładu STSW-IMG009 można zastosować sensor VL53L1X do pracy

Tabela 1. Opis połączeń pomiędzy czujnikiem i płytką Nucleo

VL53L1X Pololu	Nucleo	Opis
VIN	AVDD (3,3V) CN10-7	Zasilanie +3,3 V
GND	GND CN10-9	Masa
SDA	PB9 CN10-5	Linia SDA magistrali I²C
SCL	PB8 CN10-3	Linia SCL magistrali I²C
GPIO1	PA4 CN7-32	Wyjście przerwania z czujnika (do opcjonalnego wykorzystania)

Listing 1. Deklaracja niezbędnych zmiennych i prototyp funkcji *printf*

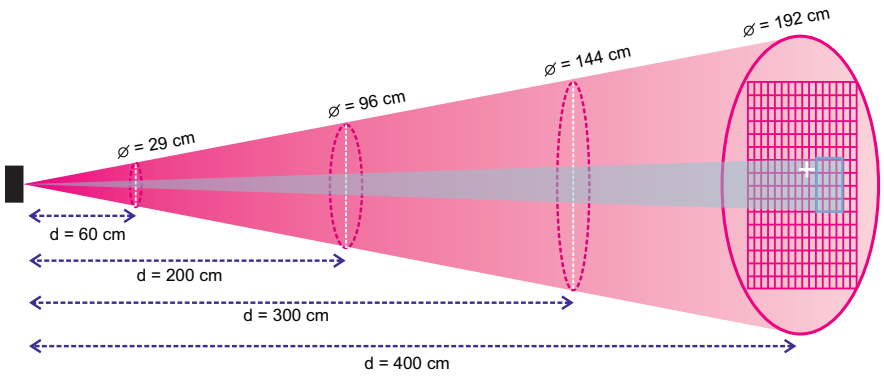
```
/* USER CODE BEGIN PV */
uint16_t dev=0x52;
int status=0;
uint8_t byteData;
uint16_t wordData, Distance,
uint16_t SignalRate, AmbientRate, SpadNum;
uint8_t RangeStatus,dataReady;
/* USER CODE END PV */
/* USER CODE BEGIN 0 */
//int fputc(int ch, FILE *f)
#define PUTCHAR_PROTOTYPE int __io_putchar (int ch)

PUTCHAR_PROTOTYPE
{
          HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, 0xFFFF);
          return ch;
}
/* USER CODE END 0 */
```

Listing 2. Inicjalizacja czujnika

```
/* Those basic I2C read functions can be used */
/* to check your own I2C functions */
status = VL53L1_RdByte(dev, 0x010F, &byteData);
printf("VL53L1X Model_ID: %X\n", byteData);
status = VL53L1_RdByte(dev, 0x0110, &byteData);
printf("VL53L1X Module_Type: %X\n", byteData);
status = VL53L1_RdWord(dev, 0x010F, &wordData);
printf("VL53L1X: %X\n", wordData);
while(sensorState==0)
{
          status = VL53L1X_BootState(dev, &sensorState);
          HAL_Delay(2);
}
printf("Chip booted\n");
/* This function must to be called */
/* to initialize the sensor with the default setting  */
status = VL53L1X_SensorInit(dev);
/* Optional functions to be used */
/* to change the main ranging parameters */
/* according the application requirements */
/* to get the best ranging performances */
/* 1=short, 2=long */
status = VL53L1X_SetDistanceMode(dev, 2);
/* in ms possible values [20, 50, 100, 200, 500] */
status = VL53L1X_SetTimingBudgetInMs(dev, 100);
/* in ms, IM must be > = TB */
status = VL53L1X_SetInterMeasurementInMs(dev, 100);
/* offset compensation in mm */
// status = VL53L1X_SetOffset(dev,20);
/* minimum ROI 4,4 */
// status = VL53L1X_SetROI(dev, 16, 16);
/* may take few second to perform the offset cal*/
// status = VL53L1X_CalibrateOffset(dev, 140, &offset);
/* may take few second to perform the xtalk cal */
// status = VL53L1X_CalibrateXtalk(dev, 1000, &xtalk);
printf("VL53L1X Ultra Lite Driver Example running ...\n");
/* This function has to be called to enable the ranging */
status = VL53L1X_StartRanging(dev);
```

Listing 3. Odczyt danych z czujnika

```
/* USER CODE BEGIN WHILE */
while (1)
{
          while (dataReady == 0){
                    status = VL53L1X_CheckForDataReady(dev, &dataReady);
                    HAL_Delay(2);
          }
          dataReady = 0;
          status = VL53L1X_GetRangeStatus(dev, &RangeStatus);
          status = VL53L1X_GetDistance(dev, &Distance);
          status = VL53L1X_GetSignalRate(dev, &SignalRate);
          status = VL53L1X_GetAmbientRate(dev, &AmbientRate);
          status = VL53L1X_GetSpadNb(dev, &SpadNum);
          /* clear interrupt has to be called to enable next interrupt*/
          status = VL53L1X_ClearInterrupt(dev);
          printf("%u, %u, %u, %u, %u\n", RangeStatus, Distance,
          SignalRate, AmbientRate,SpadNum);
          /* USER CODE END WHILE */
}
```



Rysunek 4. Pole widzenia sensora

jako dalmierz i poprzez port UART2 wysyłać sformatowane jako tekst dane pomiarowe. Na **listingu 1** pokazana została deklaracja niezbędnych zmiennych i prototyp funkcji printf do wysyłania danych portem UART2. **Listing 2** rozpoczyna się testem podłączenia sensora realizowanym poprzez wysłanie zapytania o jego typ. Następne komendy służą do inicjacji sensora w trybie długiego zasięgu (dystans do 4 metrów), na koniec włączana jest funkcja cyklicznych pomiarów. Funkcje kalibracji zostały zakomentowane ponieważ sensor powinien być skalibrowany na etapie produkcji i bez zmiany warunków jego użytkowania, powtórna kalibracja nie jest potrzebna. Na **listingu 3** w pętli głównej programu odbywa się ciągły odczyt z sensora. Najpierw program oczekuje, aż sensor będzie gotów przesłać nowe dane. Potem w serii kolejnych pobrań odczytywane są kolejne parametry: statusu pomiarów, zmierzona odległość w mm, parametr obrazujący jakość sygnału, poziom oświetlenia, ilość aktywnych diod matrycy SPAD. Na koniec odczytane dane są formatowane i wysyłane portem UART2. Następnie cały cykl rozpoczyna się na nowo.

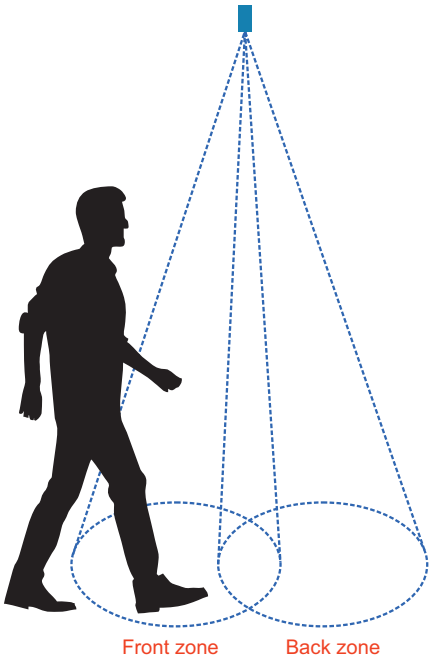
Dwie strefy odczytu

Przykład zawarty w pakiecie STSW-IMG010 pokazuje jak stworzyć oprogramowanie do zliczania obiektów lub do detekcji gestów – przesuwania dłoni. Działanie programu wiąże się z dwu strefowym odczytem matrycy SPAD, więc zanim przejdziemy do listingów warto napisać kilka słów o tej opcji działania sensora. Zagadnienie dokładnie opisuje dokument UM2555, który można znaleźć na stronach firmy ST.

Na **rysunku 4** pokazano jak w funkcji odległości zmienia się pole widzenia sensora. Jeżeli chcemy to pole zawęzić zmniejszamy ilość aktywnych fotodiod, przez określenie wymiarów zwartego prostokątnego obszaru, które mają utworzyć. Najmniejszy dopuszczalny obszar to 4×4. Określenie rozmiarów nowego aktywnego obszaru odbywa się poprzez wywołanie funkcji API VL53L1X_SetROI(). Pokazana na **rysunku 5** tabela odzwierciedla lokalizacje wszystkich fotodiod matrycy oraz ich numerację. Zaznaczony na żółto prostokąt oznacza aktywną strefę składającą się z 6×11 fotodiod. Teraz przy pomocy funkcji VL53L1X_SetROICenter() należy określić pozycję diody znajdującej się w centrum. Na rysunku 5 jest to fioletowy obszar wokół numeru fotodiody 223. Jeżeli geometryczny środek obszaru przypada pomiędzy rzędami, należy podać numer elementu znajdującego się bezpośrednio ponad i bezpośrednio z prawej strony pozycji środkowej. Od tej chwili sensor reaguje tylko na impulsy świetlne docierające do tej strefy. Jeżeli teraz przy pomocy funkcji VL53L1X_SetROICenter() zmienimy lokalizację punktu

```
/* Table of SPAD locations. Each SPAD has a number which is not obvious.
*
* 128,136,144,152,160,168,176,184,  192,200,208,216,224,232,240,248
* 129,137,145,153,161,169,177,185,  193,201,209,217,225,233,241,249
* 130,138,146,154,162,170,178,186,  194,202,210,218,226,234,242,250
* 131,139,147,155,163,171,179,187,  195,203,211,219,227,235,243,251
* 132,140,148,156,164,172,180,188,  196,204,212,220,228,236,244,252
* 133,141,149,157,165,173,181,189,  197,205,213,221,229,237,245,253
* 134,142,150,158,166,174,182,190,  198,206,214,222,230,238,246,254
* 135,143,151,159,167,175,183,191,  199,207,215,223,231,239,247,255

* 127,119,111,103, 95, 87, 79, 71,   63, 55, 47, 39, 31, 23, 15,  7
* 126,118,110,102, 94, 86, 78, 70,   62, 54, 46, 38, 30, 22, 14,  6
* 125,117,109,101, 93, 85, 77, 69,   61, 53, 45, 37, 29, 21, 13,  5
* 124,116,108,100, 92, 84, 76, 68,   60, 52, 44, 36, 28, 20, 12,  4
* 123,115,107, 99, 91, 83, 75, 67,   59, 51, 43, 35, 27, 19, 11,  3
* 122,114,106, 98, 90, 82, 74, 66,   58, 50, 42, 34, 26, 18, 10,  2
* 121,113,105, 97, 89, 81, 73, 65,   57, 49, 41, 33, 25, 17,  9,  1
* 120,112,104, 96, 88, 80, 72, 64,   56, 48, 40, 32, 24, 16,  8,  0 - Pin 1*/
```

Rysunek 5. Tabela SPAD pozycji fotodiod



Rysunek 6. Zliczanie osób

centralnego np. na 159, sensor będzie reagował na impulsy świetlne z obszaru 6×11 fotodiod wokół nowego punktu centralnego drugiej strefy. Dokonując przełączeń między dwiema lokalizacjami punktu centralnego, sensor naprzemiennie kontroluje sytuację w dwu strefach.

Użycie sensora do zliczania osób

Na **rysunku 6** pokazano jak użyć sensor VL53L1X do zliczania liczby osób. Sensor pracuje z dwiema strefami odczytu. Każdy przekraczający strefy w kolejności pierwsza-druga zostanie zliczony jako osoba wchodząca. Z kolei przekraczając strefy w kolejności druga-pierwsza, osoba zostanie zliczona jako wychodząca.

Inicjacja sensora jest podobna do tej z poprzedniego przykładu. Na **listingu 4** pokazano główne różnice: wprowadzenie tablicy center[2] do przechowania dwu lokalizacji punktów centralnych stref, zmiana parametrów czasowych oraz zaprogramowanie wymiarów stref.

Pętla główna programu pokazana jest na **listingu 5**. Tak jak w poprzednim przykładzie odczyt danych z sensora następuje, gdy zgłosi on gotowość. Po odczycie odległości następuje wyzerowanie przerwania i po krótkiej pauzie ustawiana jest nowa pozycja punktu środkowego drugiej strefy. Następnie wywoływana jest procedura ProcessPeopleCountingData(), której algorytm badania obecności obiektu w obu strefach określa, w którą stronę obiekt poruszał się i odpowiednio modyfikuje stan licznika PplCounter. Po uaktualnieniu licznika stref Zone i wysłaniu portem UART2 informacji o stanie operacji i zawartości licznika PplCounter, program wraca na początek pętli głównej.

W tym przykładzie UART został zaprogramowany do pracy z szybkością 460800 bodów. Program może reagować i odpowiednio zliczać ruchy dłoni o ile gesty nie są wykonywane na tle sylwetki. W czasie testów zauważyłem, że na działanie sensora negatywnie wpływa intensywne oświetlenie. Według dokumentacji sensor jest mniej podatny na zakłócenia gdy wybierze się mniejszy zasięg jego działania i ograniczy się intensywność oświetlenia.

Ryszard Szymaniak

Linki:
[1] *http://bit.ly/2mvNjSD*
[2] *http://bit.ly/2m9vGHR*
[3] *http://bit.ly/2kZIhgz*

Listing 4. Przygotowanie do odczytu danych z dwóch stref

```
/* USER CODE BEGIN PV */
/* these are the spad center of the 2 8*16 zones */
int center[2] = {167,231};
…
/* USER CODE END PV */
…
/* in ms possible values [20, 50, 100, 200, 500] */
status += VL53L1X_SetTimingBudgetInMs(dev, 20);
status += VL53L1X_SetInterMeasurementInMs(dev, 20);
/* minimum ROI 4,4 */
status += VL53L1X_SetROI(dev, 8, 16);
if (status != 0) {
          printf("Error in Initialization or configuration of the device\n");
          return (-1);
}
printf("Start counting people  ...\n");
```

Listing 5. Główna pętla programu realizującego funkcję zliczania osób

```
/* USER CODE BEGIN WHILE */
while (1){
          while (dataReady == 0) {
          status = VL53L1X_CheckForDataReady(dev, &dataReady);
          HAL_Delay(2);
          }

          dataReady = 0;
          status += VL53L1X_GetRangeStatus(dev, &RangeStatus);
          status += VL53L1X_GetDistance(dev, &Distance);
          /* clear interrupt has to be called to enable next interrupt*/
          status += VL53L1X_ClearInterrupt(dev);
          if (status != 0) {
          printf("Error in operating the device\n");
          return (-1);
}

HAL_Delay(10);
status = VL53L1X_SetROICenter(dev, center[Zone]);
if (status != 0) {
          printf("Error in chaning the center of the ROI\n");
          return (-1);
}
// inject the new ranged distance in the people counting algorithm
PplCounter = ProcessPeopleCountingData(Distance, Zone);
Zone++;
Zone = Zone%2;
printf("%d, %d, %d, %d\n", Zone, RangeStatus, Distance, PplCounter);
/* USER CODE END WHILE */
}
```