Sterowniki robotów

Laboratorium – SPI i I2C

Interfejsy komunikacyjne w praktycznym wykorzystaniu

Wojciech Domski

Dokument `sr_lab05.pdf`, Wersja 1.3.0

Spis treści

1 Wprowadzenie **2**

2 Opis ćwiczenia **2**

3 Interfejsy komunikacyjne **3**
3.1 SPI 3
3.2 I2C 4

4 Zadania do wykonania **6**
4.1 X-NUCLEO-IKS01A2 – LSM6DSL 3-axis accelerometer + 3-axis gyroscope, LSM303AGR 3-axis magnetometer + 3-axis accelerometer, HTS221 humidity and temperature, LPS22HB pressure 6
4.2 KAmodBAR – Moduł czujnika ciśnienia z układem MPL115A1 (SPI) 7
4.3 KAmodEXP1 – Adresowalny ekspander GPIO z interfejsem SPI 7
4.4 Uporządkowanie stanowiska 8

5 Podsumowanie **9**

Literatura **10**

1 Wprowadzenie

Jedną z istotnych części mikrokontrolera są interfejsy komunikacyjne, które to pozwalają na komunikację pomiędzy urządzeniami zewnętrznymi takimi, jak czujniki MEMS, sterowniki silników, czy pamięci zewnętrzne. W instrukcji [1] został przedstawiony jeden z najpopularniejszych interfejsów komunikacyjnych – port szeregowy. Natomiast niniejszy dokument przedstawi dwa najbardziej popularne interfejsy najczęściej wykorzystywane w systemach wbudowanych – SPI oraz I2C.

2 Opis ćwiczenia

Uwaga! Przykłady implementacji różnych peryferiów mikrokontrolera można znaleźć w plikach biblioteki HAL dla danej rodziny układów. Pliki te znajdują się zazwyczaj w ustaloenj ścieżce, np. *C:/Users/Wojciech Domski/STM32Cube/Repository/*. Jednakże położenie tych plików może być różne w zależności od systemu operacyjnego, jak i miejsca instalacji biblioteki. Aby zweryfikować katalog przechowywania repozytorium należy uruchomić program `STM32CubeMX`, a następnie wybrać *Help* → *Updater settings ...* i odczytać zawartość pola *Repository folder*. Przykłady znajdują się w podkatalogu *STM32Cube_FW_L4_V1.7.0/Projects/STM32L476RG-Nucleo/Examples/* podzielonym na podfoldery ze względu na wykorzystywane peryferium.

W ramach laboratorium należy zrealizować jedno z ćwiczeń wskazane przez prowadzącego. Niemalże wszystkie ćwiczenia realizowane są w oparciu o płytki rozszerzeń X-NUCLEO [18] kompatybilne z zestawem ewaluacyjnym NUCLEO-L476RG [11], [12]. W ramach ćwiczenia wymagane jest czytanie oraz analizowanie schematów płytek rozszerzeń, a także czytanie dokumentacji technicznej układów wykonawczych. Ponadto, do badania poprawności komunikacji wykorzystywany będzie analizator stanów logicznych.

W trakcie laboratorium zrealizowane będą ćwiczenia oparte o następujące zestawy rozszerzeń:

X-NUCLEO-IKS01A2 – LSM6DSL 3-axis accelerometer + 3-axis gyroscope, LSM303AGR 3-axis magnetometer + 3-axis accelerometer, HTS221 humidity and temperature, LPS22HB pressure, [13] – obsługa akcelerometru, magnetometru, żyroskopu i barometru; wykorzystanie czujnika wilgotności i temperatury, obsługa interfejsu I2C.

KAmodBAR – Moduł czujnika ciśnienia z układem MPL115A1 (SPI), [4] – obsługa czujnika ciśnienia i zbieranie pomiarów z układu za pomocą interfejsu SPI,

KAmodEXP1 – Adresowalny ekspander GPIO z interfejsem SPI, [3] – obsługa modułu rozszerzającego liczbę konfigurowalnych cyfrowych wejść/wyjść i komunikacja za pomocą magistrali SPI.

3 Interfejsy komunikacyjne

Jednymi z najbardziej popularnych interfejsów komunikacyjnych oprócz USARTa (portu szeregowego) są SPI oraz I2C. Są one powszechnie wykorzystywane w różnego typu aplikacjach, a większość czujników, jak i sterowników posiada wbudowany kontroler takiego interfejsu. W niektórych przypadkach występują układy, które posiadają oba wspomniane interfejsy, które wykorzystują te same linie układu, jak np. układ akcelerometru i magnetometru LSM303C [17].

3.1 SPI

SPI (Serial Peripheral Interface) jest synchronicznym interfejsem szeregowym [16]. Posiada on 4 linie sygnałowe:

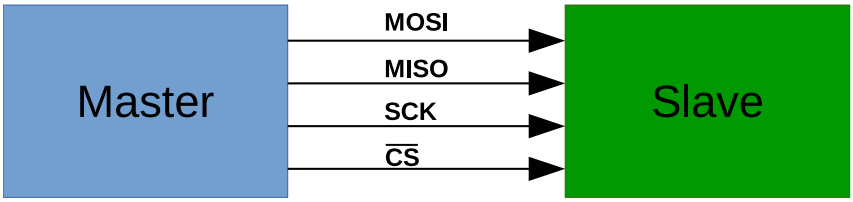
SCK – Serial Clock linia taktująca,

MOSI – Master-Output Slave-Input , linia danych od układu nadrzędnego (ang. *master*) do układu podrzędnego (ang. *slave*),

MISO – Master-Input Slave-Output , linia danych od układu podrzędnego do układu nadrzędnego,

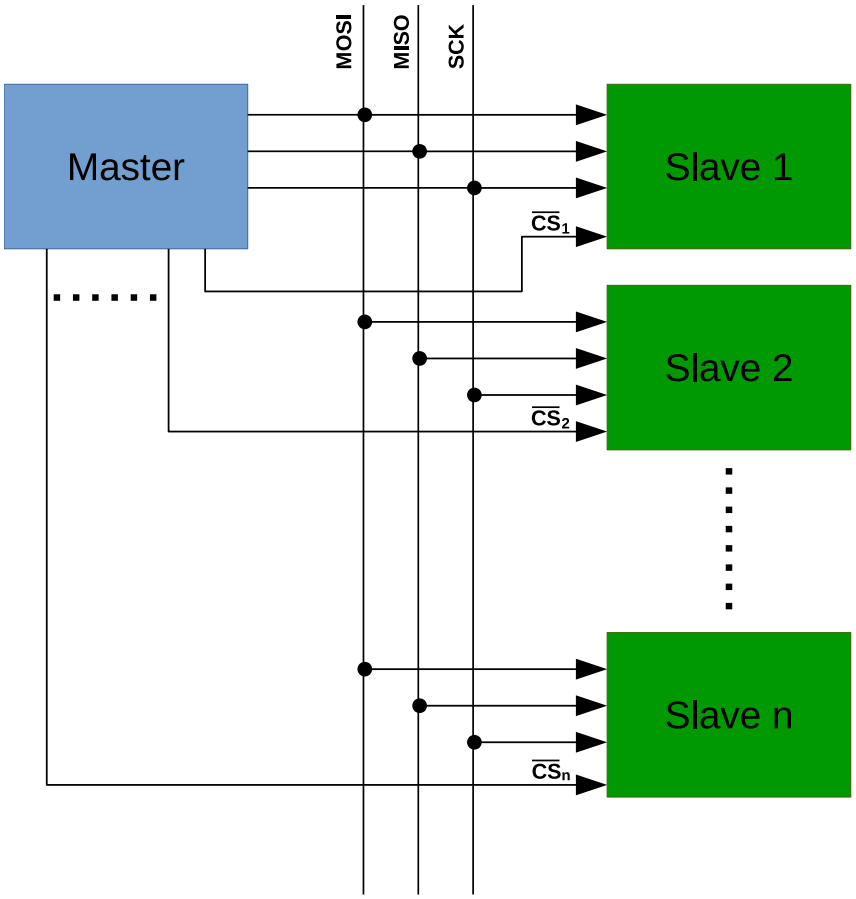
SS – Slave Select , linia wyboru układu podrzędnego, czasem określana jako CS (ang. *Chip Select*).

Niektóre linie nie muszą być wykorzystywane, zależy to od wyboru trybu pracy oraz konfiguracji. Jednakże zawsze wymagane jest, aby linie SCK oraz MOSI były aktywne. Przykładowe połączenie mastera oraz slavea zostało zaprezentowane na rysunku 1.



Rysunek 1: Interfejs SPI z jednym układem podrzędnym

Na dodatkową uwagę zasługuje linia CS, którą zarządza master. Aby uaktywnić transmisję ustawia on tą linię w stan niski, dopiero wówczas slave będzie uczestniczył w transmisji. Fakt ten pozwala na implementację topologii gwiazdy, gdzie w centrum znajduje się master, a w gałęziach układy slave. W takiej topologii wszystkie układy podrzędne podłączone są do linii SCK, MOSI, MISO, natomiast układ master posiada oddzielne linie CS dla każdego z urządzeń podrzędnych (rysunek 2).



Rysunek 2: Interfejs SPI z wieloma układami podrzędnymi

Sam interfejs SPI może pracować w trzech trybach:

Full–duplex Transfer danych odbywa się przy wykorzystaniu trzech linii,

Half–duplex Transfer danych odbywa się przy wykorzystaniu dwóch linii, gdzie linia danych (MOSI) wykorzystywana jest zarówno do nadawania, jak i odbierania danych od układu podrzędnego,

Simplex Podobnie, jak w przypadku trybu Half-duplex wykorzystywane są tylko dwie linie, jednakże linia danych jest jednokierunkowa.

W zależności od wykorzystywanego zewnętrznego układu jeden z trzech trybów może być użyty.

Biblioteka HAL [7] dostarcza API (ang. *Application Programming Interface*), które pozwala na łatwą konfigurację oraz wymianę danych przy użyciu interfejsu SPI. Do najbardziej podstawowych zalicza się: `HAL_SPI_Transmit()`, `HAL_SPI_TransmitReceive` oraz `HAL_SPI_Receive()`. Pierwsza z nich pozwala na wysłanie paczki danych przekazanej w buforze, druga na jednoczesne odbieranie danych i ich wysyłanie, natomiast ostatnia umożliwia odbiór danych od urządzenia podrzędnego.

```
1 HAL_StatusTypeDef HAL_SPI_Transmit (SPI_HandleTypeDef * hspi,
2   uint8_t * pData, uint16_t Size, uint32_t Timeout);
3
4 HAL_StatusTypeDef HAL_SPI_Receive (SPI_HandleTypeDef * hspi,
5   uint8_t * pData, uint16_t Size, uint32_t Timeout);
6
7 HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef * hspi,
8 uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout);
```

3.2 I2C

Drugim bardzo popularnym, a tym samym szeroko wykorzystywanym interfejsem komunikacyjnym obok SPI jest I2C (ang. *Inter–Integrated Circuit*). Czasem określany jako I^2C lub IIC. W odróżnieniu do SPI, I2C posiada tylko dwie linie:

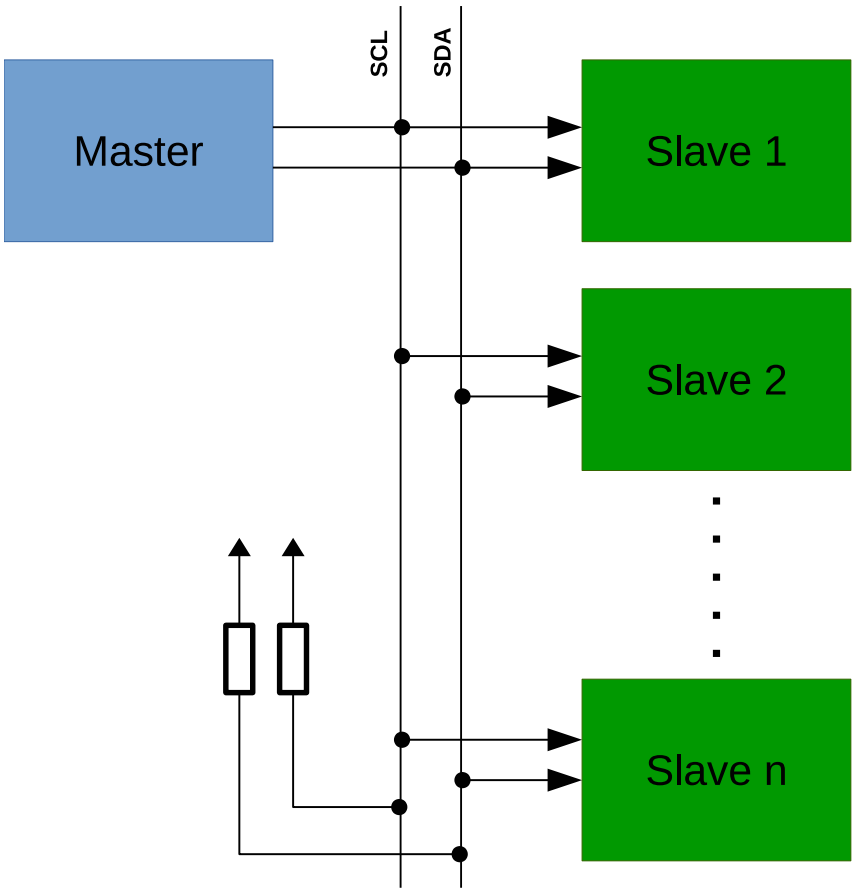
SCL sygnał zegarowy,

SDA linia danych.

Jest to interfejs, który wykorzystuje transmisję synchroniczną. Przykładową topologię zaprezentowano na rysunku 3. Master jest urządzeniem, który inicjalizuje transmisję do wybranego urządzenia podrzędnego. Każde z tych urządzeń posiada unikatowy adres dzięki, któremu możliwe jest jednoznaczne określenie, które urządzenie jest aktywne. Operacja ta określana jest jako adresowanie, podczas tej operacji wszystkie urządzenia typu slave nasłuchują transmisji. W momencie, w którym jedno z urządzeń posiada adres identyczny z tym, który jest rozgłaszany na szynie I2C przechodzi ono w stan aktywny. Transmisja odbywa się w trybie Half-Duplex (dane przepływają w obu kierunkach na jednej linii).

Adres urządzenia jest zazwyczaj 7-bitowy, jednakże każdorazowo wysyłany jest jeden bajt. Dzieje się tak, ponieważ adres urządzenia jest dopełniany jednym bitem. Posiada on specjalne znaczenie, które określa tryb do jakiego powinno przełączyć się urządzenie podrzędne. Jeśli wartość tego bitu jest równa 1 to slave przełącza się w tryb odbiornika (master będzie zapisywał dane do urządzenia), bądź nadajnika (urządzenie będzie wysyłać dane do mastera).

Ważnym elementem szyny I2C jest występowanie rezystorów podciągających widocznych na diagramie 3. Dzięki nim zarówno master, jak i urządzenia typu slave wysyłają sygnały na szynę I2C poprzez ściąganie jej do stanu niskiego.



Rysunek 3: Interfejs I2C z wieloma układami podrzędnymi

Biblioteka HAL [7] dostarcza API pozwalające na łatwą konfigurację oraz wymianę danych przy użyciu interfejsu I2C. Do najbardziej podstawowych zalicza się: `HAL_I2C_Master_Transmit()` oraz `HAL_I2C_Master_Receive()`. Pierwsza z nich pozwala na wysłanie paczki danych przekazanej w buforze, natomiast druga umożliwia odbiór danych od urządzenia podrzędnego. Istnieją również funkcję dedykowane dla przypadku implementacji urządzenia typu slave. Posiadają one analogiczne nazwy: `HAL_I2C_Slave_Transmit()` oraz `HAL_I2C_Slave_Receive()`.

```
1 HAL_StatusTypeDef HAL_I2C_Master_Transmit (I2C_HandleTypeDef * hi2c,
2   uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout);
3
```

```
4 HAL_StatusTypeDef HAL_I2C_Master_Receive(I2C_HandleTypeDef * hi2c,
5   uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout);
```

Jednakże najczęściej wykorzystywany jest inny zestaw instrukcji, a mianowicie `HAL_I2C_Mem_Write()` i `HAL_I2C_Mem_Read()`. W odróżnieniu do wcześniej przytoczonych funkcji funkcje `HAL_I2C_Mem_xxxx()` pozwalają na oddzielenie części adresowej od części danych. Jest to szczególnie przydatne podczas komunikacji z urządzeniami przechowujących dane, jak EEPROM, a także czujnikami MEMS.

```
1 HAL_StatusTypeDef HAL_I2C_Mem_Write (I2C_HandleTypeDef * hi2c,
2   uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize,
3   uint8_t * pData, uint16_t Size, uint32_t Timeout);
4
5 HAL_StatusTypeDef HAL_I2C_Mem_Read (I2C_HandleTypeDef * hi2c,
6   uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize,
7   uint8_t * pData, uint16_t Size, uint32_t Timeout);
```

Jedną z często wykorzystywanych funkcji jest `HAL_I2C_IsDeviceReady()` pozwalająca na stwierdzenie, czy urządzenie jest gotowe do transmisji.

```
1 HAL_StatusTypeDef HAL_I2C_IsDeviceReady (I2C_HandleTypeDef * hi2c,
2   uint16_t DevAddress, uint32_t Trials, uint32_t Timeout);
```

Jak już to zostało wcześniej opisane interfejs jest interfejsem typu Half–Duplex – tylko jedno z urządzeń może aktywnie prowadzić komunikację. Z uwagi na to wymagane jest istnienie mechanizmu, który spowodowałby jednoznaczne określenie kierunku transmisji. Takim mechanizmem jest ostatni bit adresujący urządzenie, który określa, czy będzie przeprowadzona operacja opisu, czy zapisu, a tym samym kierunek transmisji. Nasuwa to jednak pewne konsekwencje. Zazwyczaj w dokumentacjach technicznych podawany jest adres 7-bitowy, przykładowo `0x22` (`0b00100010`). Aby przeprowadzić udaną komunikację z takim urządzeniem należy owy adres przesunąć bitowo w lewo o jeden co w konsekwencji dla powyższego przykładu daje `0x44` (`0b0100010x`), gdzie najmłodszy bit `x` określa typ transmisji (zapis/odczyt). Dopiero tak uzyskany adres urządzenia należy przekazać do odpowiedniej funkcji.

Adresy urządzeń są wpisane na stałe do pamięci co w konsekwencji prowadzi do konfliktu podczas, gdy na szynie I2C znajdują się dwa identyczne układy. Większość układów dostępnych w sprzedaży pozwala na częściowe obejście tego problemu przez możliwość konfiguracji wybranych bitów adresu przez dedykowane porty wejściowe urządzenia. Wówczas najczęściej w dokumentacji technicznej urządzenia podawane są dwa adresy urządzenia, przykładowo `0x22` lub `0x23`, bądź w zapisie binarnym (`0b0010001x`), gdzie `x` jest stanem dedykowanego portu czujnika, czy pamięci. Pozwala to na poprawną pracę dwóch takich samych układów. Spotykane są również urządzenia, gdzie adres jest zależny od stanu na dwóch pinach. Umożliwia to podłączenie czterech tych samych układów do magistrali I2C. Ponadto, istnieją również czujniki, które posiadają programowalny adres co w efekcie przekłada się na możliwość wykorzystania wielu identycznych czujników na jednej szynie I2C.

4 Zadania do wykonania

Uwaga! Pamiętaj, aby wyłączyć optymalizację kodu (-O0), a także wykorzystać kompilację równoległą (*parallel*) np. -j8 [2]. Ponadto, ustaw automatyczny zapis przed wykonaniem kompilacji, aby uniknąć problemów związanych z rozbieżnością kodu, a plikiem wynikowym wgranym na mikrokontroler. Pamiętaj również, że gdy powtórnie generujesz projekt może okazać się konieczne jego wyczyszczenie jak i przebudowanie indeksu. W tym celu rozwiń menu kontekstowe dla projektu i wybierz *Clean project*, a nastepnie powtórz operację oraz wybierz *Index* → *Rebuild*.

Do każdego zadania stwórz projekt w programie STMStudio [10] i zwizualizuj najważniejsze parametry przy użyciu tego oprogramowania.

4.1 X-NUCLEO-IKS01A2 – LSM6DSL 3-axis accelerometer + 3-axis gyroscope, LSM303AGR 3-axis magnetometer + 3-axis accelerometer, HTS221 humidity and temperature, LPS22HB pressure

Odczytaj cztery wielkości fizyczne wskazane przez prowadzącego:

- przyspieszenie liniowe LSM6DSL [15] lub LSM303AGR [9],
- prędkość kątową LSM6DSL,
- natężenie pola magnetycznego LSM303AGR,

- wilgotność HTS221 [8],
- temperaturę HTS221,
- ciśnienie LPS22HB [14],

a następnie prześlij je do komputera za pomocą portu szeregowego. Do pomiaru wymienionych wielkości fizycznych wykorzystaj czujniki znajdujące się na module X-NUCLEO-IKS01A2.

Rozpocznij od sprawdzenia poprawności konfiguracji magistrali I2C. Do tego celu można wykorzystać rejestr *WHO_AM_I* jednego z czujników. Niektóre układy posiadają fizycznie definiowany adres urządzenia. W oparciu o noty katalogowe czujników uruchom układy pomiarowe oraz przeprowadź akwizycję danych wskazanych wielkości fizycznych.

4.2 KAmodBAR – Moduł czujnika ciśnienia z układem MPL115A1 (SPI)

Płytka KAmodBAR wyposażona jest w czujnik ciśnienia MPL115A1 [6]. Pozwala on na pomiar ciśnienia atmosferycznego w zakresie od 50 do 115 kPa. Czujnik komunikuje się po magistrali SPI.

W oparciu o dokumentację techniczną sensora [6] oraz dokumentację modułu KAmodBAR [4] wykonaj program, który cyklicznie będzie pobierał pomiary ciśnienia z układu MPL115A1.

4.3 KAmodEXP1 – Adresowalny ekspander GPIO z interfejsem SPI

KAmodEXP1 jest modułem, który jest tzw. ekspanderem portów. Pozwala on w prosty sposób za pomocą magistrali SPI wzbogacić mikrokontroler o dodatkowe wejścia/wyjścia cyfrowe. Ponadto, układ [5] oferuje możliwość konfiguracji przerwań dla wszystkich wejść/wyjść cyfrowych.

W oparciu o dokumentację techniczną układu [5] oraz dokumentację modułu KAmodEXP1 [3] wykonaj program, który skonfiguruje porty GP0, GP1, GP2 oraz GP3 układu MCP23S08 jako cyfrowe wyjścia i cyklicznie będzie zmieniał ich stany według poniższych wzorców:

$$\begin{aligned}
[GP0, GP1, GP2, GP3]_1 &= [0, 1, 0, 1], \\
[GP0, GP1, GP2, GP3]_2 &= [1, 0, 1, 0], \\
[GP0, GP1, GP2, GP3]_3 &= [1, 1, 1, 1], \\
[GP0, GP1, GP2, GP3]_4 &= [0, 0, 0, 0],
\end{aligned}$$

Wyprowadzenia GP0, GP1, GP2 oraz GP3 należy podłączyć do mikrokontrolera STM za pomocą przewodów. **Uwaga!** Przed podłączeniem modułu i płytki Nucleo należy ustawić porty mikrokontrolera jako wejścia, wgrać program, a następnie podłączyć moduł i płytkę.

4.4 Uporządkowanie stanowiska

Odłóż płytkę i kabel na miejsce. Usuń projekt z Atollic TrueSTUDIO. Można to zrobić przez kliknięcie prawym przyciskiem myszki na projekt i wybranie opcji Usuń (Delete) z menu kontekstowego.

5 Podsumowanie

Laboratorium dotyczy interfejsów komunikacyjnych w praktycznym ujęciu. Zaproponowano szereg ćwiczeń, które pozwalają na zapoznanie się z peryferiami mikrokontrolera służącymi do komunikacji. Interfejsy takie, jak SPI, czy I2C są wykorzystywane powszechnie w różnych aplikacjach, między innymi do komunikacji z czujnikami. Prawie każde z ćwiczeń opiera się o wykorzystanie płytki deweloperskiej NUCLEO-L476RG oraz modułu rozszerzającego X-NUCLEO, które są częścią ekosystemu zaproponowanego przez firmę ST do prototypowania urządzeń.

Literatura

[1] W. Domski. Sterowniki robotów, Laboratorium – Debugowanie, Zaawansowane techniki debugowania. Marzec, 2017.

[2] W. Domski. Sterowniki robotów, Laboratorium – Wprowadzenie, Wykorzystanie narzędzi STM32CubeMX oraz SW4STM32 do budowy programu mrugającej diody z obsługą przycisku. Marzec, 2017.

[3] Kamami. *KAmodEXP1 – Adresowalny ekspander GPIO z interfejsem SPI.*, Wrzesień, 2010. ver. 1.0, `http://dl.btc.pl/kamami_wa/kamodexp1.pdf`.

[4] Kamami. *KAmodBAR-SPI – Moduł czujnika ciśnienia z układem MPL115A1 (SPI).*, Czerwiec, 2012. ver. 1.0, `http://dl.btc.pl/kamami_wa/kamodbar_spi.pdf`.

[5] Microchip Technology Inc. *MCP23008/MCP23S08 – 8-Bit I/O Expander with Serial Interface.*, Sierpień, 2007. Revision E, `http://ww1.microchip.com/downloads/en/DeviceDoc/21919e.pdf`.

[6] NXP Semiconductors. *MPL115A1 – Miniature SPI digital barometer, 50 to 115 kPa.*, Październik, 2017. Rev. 8, `https://www.nxp.com/docs/en/data-sheet/MPL115A1.pdf`.

[7] ST. *Description of STM32L4 HAL and Low-layer drivers.*, Luty, 2016.

[8] ST. *HTS221 – Capacitive digital sensor for relative humidity and temperature.*, Sierpień, 2016. Rev 4, `http://www.st.com/resource/en/datasheet/hts221.pdf`.

[9] ST. *LSM303AGR – Ultra-compact high-performance eCompass module: ultra-low-power 3D accelerometer and 3D magnetometer.*, Wrzesień, 2016. Rev 9, `http://www.st.com/resource/en/datasheet/lsm303agr.pdf`.

[10] ST. *STM Studio run-time variables monitoring and visualization tool for STM32 microcontrollers.*, Marzec, 2016.

[11] ST. *STM32 Nucleo-64 board.*, Listopad, 2016.

[12] ST. *STM32 Nucleo-64 board, User manual.*, Listopad, 2016.

[13] ST. *Getting started with the X-NUCLEO-IKS01A2 motion MEMS and environmental sensor expansion board for STM32 Nucleo, User manual.*, Styczeń, 2017.

[14] ST. *LPS22HB – MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer.*, Czerwiec, 2017. Rev 6, `http://www.st.com/resource/en/datasheet/DM00140895.pdf`.

[15] ST. *LSM6DSL – iNEMO inertial module: always-on 3D accelerometer and 3D gyroscope.*, Wrzesień, 2017. Rev 7, `http://www.st.com/resource/en/datasheet/lsm6dsl.pdf`.

[16] ST. *STM32L4x5 and STM32L4x6 advanced ARM®-based 32-bit MCUs, Reference Manual.*, Marzec, 2017.

[17] ST. *LSM303C, Ultra-compact high-performance eCompass module: 3D accelerometer and 3D magnetometer.*, Czerwiec, 2014.

[18] ST. STM32 Open Development Environment, Wrzesień, 2016.

