

**Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie**

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki



PRACA INŻYNIERSKA

MICHAŁ MARCHEWKA

AUTONOMICZNY CZUJNIK PRZYSPIESZENIA

PROMOTOR:
dr inż. Adam Piłat

Kraków 2013

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIESZCZONE W PRACY.

.....

PODPIS

AGH
University of Science and Technology in Krakow

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics



ENGINEER'S THESIS

MICHAŁ MARCHEWKA

AUTONOMOUS ACCELEROMETER

SUPERVISOR:
Adam Piłat Ph. D

Krakow 2013

Serdecznie dziękuję mojej rodzinie i przyjaciołom za nieocenione wsparcie i pomoc. Dziękuję promotorowi tej pracy za profesjonalny nadzór nad jej wykonaniem. Mateuszowi Furdynie i Krzysztofowi Kapuścicie za udostępnienie sprzętu i cenne porady techniczne.

Spis treści

1. Wstęp.....	9
1.1. Motywacja	9
1.2. Kryteria projektowe	9
2. Kosztorys i opis budowy urządzeń	11
2.1. Kosztorys	11
2.2. Opis wykorzystanych modułów.....	11
2.2.1. Mikrokontroler MSP430G2553	11
2.2.2. Moduł czujnika ruchu	11
2.2.3. Moduł radiowy CC110L	12
2.3. Budowa systemu pomiarowego	13
2.3.1. Urządzenie pomiarowe.....	14
2.3.2. Urządzenie pośredniczące	14
2.4. Przepustowość transmisji danych	15
2.4.1. Magistrala I ² C.....	15
2.4.2. Magistrala SPI - urządzenie pomiarowe	16
2.4.3. Magistrala SPI - urządzenie pośredniczące	16
2.4.4. Komunikacja radiowa.....	17
2.4.5. Pakiet przesyłany przez port szeregowy	17
3. Sprzętowe moduły komunikacji cyfrowej.....	19
3.1. Moduły komunikacji szeregowej w MSP430a	19
3.2. Biblioteka obsługi portu szeregowego z wykorzystaniem modułu USCI	20
3.2.1. Ogólny opis systemu UART	20
3.2.2. Zasada działania	20
3.2.3. Wykorzystanie	20
3.2.4. Inicjalizacja	20
3.2.5. Wysłanie bajtu	23
3.3. Biblioteka obsługi protokołu I ² C z wykorzystaniem modułu USCI - funkcje niskiego poziomu	23
3.3.1. Opis protokołu.....	23
3.3.2. Zasada działania	23
3.3.3. Opis biblioteki.....	24
3.3.4. Bajt stanu.....	24
3.3.5. Inicjalizacja	25
3.3.6. Transmisja danych.....	25
3.3.7. Odbiór danych.....	27
3.3.8. Obsługa stanu NACK.....	27

3.4. Biblioteka obsługi protokołu I ² C z wykorzystaniem modułu USCI - funkcje wysokiego poziomu	28
3.4.1. Opis biblioteki.....	28
3.4.2. Odczyt rejestru	28
3.4.3. Odczyt sekwencji bajtów	28
3.4.4. Odczyt sekwencji bajtów z oczekiwaniem na zakończenie	29
3.4.5. Zapis rejestru.....	29
3.5. Programowa obsługa protokołu SPI dla urządzenia CC110L z wykorzystaniem sprzętowego modułu USCI	29
3.5.1. Ogólny opis protokołu.....	29
3.5.2. Zasada działania.....	30
3.5.3. Opis biblioteki.....	30
3.5.4. Inicjalizacja	31
3.5.5. Odczyt rejestru	31
3.5.6. Odczyt grupy rejestrów	33
3.5.7. Odczyt rejestru stanu.....	33
3.5.8. Zapis rejestru.....	34
3.5.9. Zapis grupy rejestrów.....	34
3.5.10. Wysłanie sygnału komendy.....	35
4. Urządzenia peryferyjne - opis sprzętu i bibliotek.....	37
4.1. Biblioteka obsługi czujnika MPU-6050	37
4.1.1. Inicjalizacja	37
4.1.2. Sprawdzenie poziomu zapełnienia kolejki FIFO	39
4.1.3. Procedura obsługi przerwań	41
4.2. Biblioteka obsługi modułu radiowego CC110L	41
4.2.1. Reset urządzenia	41
4.2.2. Inicjalizacja	42
4.2.3. Wysłanie pakietu	42
4.2.4. Odbiór pakietu.....	43
5. Oprogramowanie zarządzające systemem pomiarowym.....	45
5.1. Języki programowania	45
5.1.1. Oprogramowanie mikrokontrolera.....	45
5.1.2. Oprogramowanie komputera PC.....	45
5.2. Środowisko programistyczne Code Composer Studio	45
5.2.1. Opis ogólny	45
5.2.2. Marketing	46
5.2.3. Wolne oprogramowanie	46
5.3. Środowisko SmartRF Studio.....	46
5.3.1. Model warstwowy	47
5.4. Oprogramowanie urządzenia pośredniczącego.....	48
5.4.1. Ogólny opis działania.....	48
5.4.2. Programowy protokół szeregowej transmisji danych	48
5.4.3. Maszyna stanów transmisji szeregowej	49
5.4.4. Główny program urządzenia pośredniczącego	49
5.5. Oprogramowanie urządzenia pomiarowego	50

5.5.1. Ogólny opis działania.....	50
5.5.2. Główny program urządzenia pomiarowego	50
5.6. Aplikacja akwizycji danych dla komputera PC.....	51
5.6.1. Maszyna stanów obsługująca odbiór danych.....	51
6. Eksperyment porównawczy	53
6.1. Opis stanowiska	53
6.2. Porównanie jakości pomiarów prędkości kątowej.....	53
6.2.1. Konfiguracja sprzętu	53
6.2.2. Konfiguracja oprogramowania.....	54
6.2.3. Przeprowadzenie eksperymentu.....	54
6.2.4. Opracowanie danych.....	55
6.2.5. Zmiana formatu zapisu czasu.....	55
6.2.6. Interpolacja.....	55
6.2.7. Filtracja	56
6.2.8. Porównanie danych	60
6.3. Porównanie jakości pomiarów przyspieszenia liniowego	61
6.3.1. Konfiguracja sprzętu	61
6.3.2. Konfiguracja oprogramowania.....	61
6.3.3. Opracowanie danych.....	61
6.3.4. Zmiana formatu zapisu czasu.....	61
6.3.5. Filtracja	61
6.3.6. Porównanie danych	64
7. Podsumowanie	67
7.1. Ogólnie	67
7.2. Realizacja wyznaczonych celów	67
7.3. Rozwój pracy	67
Bibliografia	69

1. Wstęp

1.1. Motywacja

Celem niniejszej pracy jest zaprojektowanie oraz implementacja autonomicznego czujnika przyspieszenia. Czujnik ten powinien pracować korzystając z zasilania baterijnego oraz wysyłać pomiary drogą radiową do urządzenia odbiorczego. Rozwiązania tego typu są bardziej kosztowne od rozwiązań klasycznych - przewodowych, jednak często prowadzenie przewodów zasilających bądź telekomunikacyjnych jest kłopotliwe. Wszędzie tam, gdzie mamy do czynienia z elementami ruchomymi, przewody mogą łatwo zostać uszkodzone lub, co gorsza, mogą uszkodzić urządzenie na którym zostały zainstalowane.

Bezprzewodowa transmisja danych daje możliwość łatwego zbudowania bardziej skomplikowanego systemu pomiarowego. System taki składać się może z urządzenia zbierającego dane oraz zespołu czujników. W sytuacji, gdy wymagana jest znaczna ilość takich czujników na dużym obszarze, rozwiązanie bezprzewodowe okazuje się znacznie bardziej opłacalne[7].

1.2. Kryteria projektowe

Podczas projektowania omawianego urządzenia kierowano się następującymi kryteriami:

- Zachowanie minimalnej masy i rozmiaru układu.

To kryterium jest szczególnie ważne ze względu na wpływ, jaki masa czujnika może wywierać na zachowanie obiektu, na którym został on zainstalowany. Jeśli na przykład mamy do czynienia z ruchomym obiektem mechanicznym i znaczaco wpłyниemy na jego inertję, to nastawy regulatora sterującego tym obiektem mogą okazać się nieoptimalne lub nawet szkodliwe dla tego obiektu.

- Minimalny pobór mocy.

Minimalizacja poboru mocy wiąże się ściśle z poprzednim kryterium. W niewielkich urządzeniach elektronicznych baterie są często najcięższym elementem. Bywają one nawet cięższe od wszystkich pozostałych komponentów razem wziętych. Podczas doboru komponentów wchodzących w skład tego czujnika oraz podczas opracowywania oprogramowania położono szczególny nacisk na energooszczędność. Umożliwiło to zachowanie odpowiednio długiego czasu pracy przy zastosowaniu stosunkowo lekkich ogniw.

- Jak największa przepustowość łącza.

Większa przepustowość łącza oznacza możliwość zastosowania większej częstotliwości próbkowania, która jest istotna dla pomiarów ruchu.

- Niski koszt.

2. Kosztorys i opis budowy urządzeń

2.1. Kosztorys

Koszt zbudowania prototypu:

- 2x TI MSP430 Launchpad: 30zł
- RF Booster Pack (dwa urządzenia w komplecie): 60zł
- Czujnik ruchu: 50zł (po zakupie znaleziono identyczną płytę w cenie 10zł u innego dostawcy)
- Konwerter UART-USB: 20zł

Całkowita cena budowy urządzenia wynosiła 160zł.

2.2. Opis wykorzystanych modułów

2.2.1. Mikrokontroler MSP430G2553

Wersja obudowy wykorzystanego mikrokontrolera to PDIP20 o wymiarach ok 25mm x 10mm x 5mm. Seria mikrokontrolerów MSP430 oferuje bardzo niski pobór prądu przy sporej mocy obliczeniowej. Urządzenie bazuje na wydajnym 16-bitowym mikroprocesorze o architekturze RISC. Wyposażone jest w 16-bitowe rejesty oraz generator stałych. Zapewnia to maksymalną wydajność kodu. Cyfrowo sterowany oscylator (DCO) pozwala na przejście z trybu oszczędzania energii do trybu czuwania w czasie krótszym niż $1\mu\text{s}$. Urządzenie wyposażane jest w liczne moduły sprzętowej obsługi sygnałów cyfrowych i analogowych[9].

Pobór prądu:

- aktywna praca: poniżej $420\mu\text{A}$;
- czuwanie: $0,5\mu\text{A}$;
- wyłączony mikroprocesor z potrzymaniem pamięci RAM: $0,1\mu\text{A}$.

2.2.2. Moduł czujnika ruchu

Moduł czujnika ruchu o nazwie MPU-6050, który został wykorzystany w niniejszej pracy wyposażony jest w 3-osiowy czujnik przyspieszenia oraz 3-osiowy żyroskop. Ponadto, urządzenie to posiada wiele dodatkowych funkcjonalności takich jak filtr dolnoprzepustowy, kolejkę FIFO, cyfrowy procesor ruchu (DMP) oraz funkcję detekcji spadku swobodnego[5]. Układ scalony czujnika zainstalowany jest na niewielkiej płycie typu „breakout board” o nazwie GY-521. Jej wymiary to 20x15mm. Na płycie znajdują się:

- właściwe urządzenie pomiarowe MPU-6050 w obudowie typu QFN o wymiarach 5x5mm;

- rezystory typu pull-up zainstalowane na liniach SDA, SCL, XDA oraz XCL;
- regulator napięcia;
- inne elementy pasywne.

Dokumentacja urządzenia podaje maksymalne napięcie, którym może być zasilany układ. Wynosi ono 3,46V[5]. Dzięki regulatorowi można zastosować zasilanie wyższym napięciem.

Pobór prądu:

- Tryb normalny - działa akcelerometr i żyroskop: 4,3mA.
- Tryb low-power - działa jedynie akcelerometr.

Pobór prądu zależny od częstotliwości próbkowania:

- 1,25 Hz: $10\mu\text{A}$;
- 5 Hz: $20\mu\text{A}$;
- 20 Hz: $70\mu\text{A}$;
- 40 Hz: $140\mu\text{A}$.

Opis poszczególnych wyprowadzeń w konfiguracji użytej w tej pracy:

- VCC - zasilanie;
- GND - uziemienie.
- Główna magistrala I²C. Łączy urządzenie w trybie slave z mikrokontrolerem.

Wykorzystuje linie:

- SCL - linia zegara głównej magistrali I²C;
- SDA - linia danych głównej magistrali I²C.

- Pomocnicza magistrala I²C służy do połączenia z urządzeniem zewnętrznym w trybie master.

Wykorzystuje linie:

- XDA - linia danych pomocniczej magistrali I²C;
- XCL - linia zegara pomocniczej magistrali I²C.

- AD0 - wejście decydujące o adresie urządzenia na głównej magistrali I²C.

Stan wysoki - adres 0x69, stan niski - domyślny adres 0x68. W tej pracy pin ten podłączony jest do uziemienia.

- INT - pin, generujący przerwania dla mikrokontrolera.

Skonfigurowany na generowanie stanu wysokiego przez krótki czas po pojawienniu się nowych danych w kolejce FIFO.

2.2.3. Moduł radiowy CC110L

W niniejszej pracy został wykorzystany moduł bezprzewodowy o nazwie CC110L RF BoosterPack. Moduł ten dostarczony jest w formie niewielkiej płytki drukowanej z możliwością dołączenia do zestawu Value Line Launchpad MSP-EXP430G2. Dzięki temu, możliwe jest szybkie i bezproblemowe prototypowanie bez konieczności lutowania. Płytki ta - po dokonaniu odpowiednich modyfikacji - umożliwia również bezpośrednie zainstalowanie mikrokontrolera na swojej powierzchni. Dostępny jest także obszar przeznaczony do prototypowania, co ułatwia podłączenie zewnętrznych komponentów.

Na płytce zainstalowany jest moduł A110LR09A wyprodukowany przez firmę Anaren na podstawie urządzenia CC110L firmy Texas Instruments. Znajduje się na niej także kilka elementów pasywnych. Moduł A110LR09A wyposażony jest w zintegrowaną antenę. Częstotliwość, na której pracuje moduł radiowy w niniejszym projekcie to 868MHz. Jest to częstotliwość, na której, na terenie Europy dozwolona jest transmisja pod warunkiem ograniczenia zajętości medium do 0,1% lub zastosowania procedury „Listen Before Transmit” (LBT). Ponieważ zaprojektowany system pomiarowy transmituje dane z dużą częstotliwością, zaimplementowana została procedura LBT[16].

Użycie prądu:

- tryb uśpienia: $0,4 \mu\text{A}$;
- tryb czuwania: 1,7 mA;
- transmisja z prędkością 1,2 kBaud: 14,7 mA;

Wymiary:

- moduł A110LR09A : 9mm x 16mm x 2,5mm;
- cała płytka drukowana: 3cm x 7cm;

Konfiguracja wyprowadzeń:

- SCLK - zegar magistrali SPI.
- MISO (GDO1)- wyjście danych magistrali SPI.

Gdy linia CSL jest nieaktywna, linia MISO pełni funkcję linii ogólnego przeznaczenia GDO1. Linia GDO1 została skonfigurowana tak, aby przyjmowała stan wysokiej impedancji, gdy linia CSL jest nieaktywna, ponieważ magistrala SPI zarządzana jest przez moduł USCI.

- MOSI - wejście danych magistrali SPI.
- CSL - linia aktywująca urządzenie podrzędne na magistrali SPI.
- GDO0 - linia ogólnego przeznaczenia - pełni funkcję zależną od konfiguracji.

W opracowanym systemie pomiarowym, na tym wyprowadzeniu pojawia się logiczna jedynka, gdy zostało wysłane/odebrane słowo synchronizacji. Na końcu procedury przesyłania pakietu drogą radiową, linia wraca do stanu logicznego zera. W trybie transmisji, stan logiczny również przyjmuje wartość 0, jeśli pakiet zostaje odrzucony ze względu na przekroczenie dopuszczalnej długości, lub gdy moduł wejdzie w stan RXFIFO_OVERFLOW. W trybie transmisji poziom logiczny linii wraca do wartości 0, gdy w kolejce TX FIFO wystąpi niedobór danych.

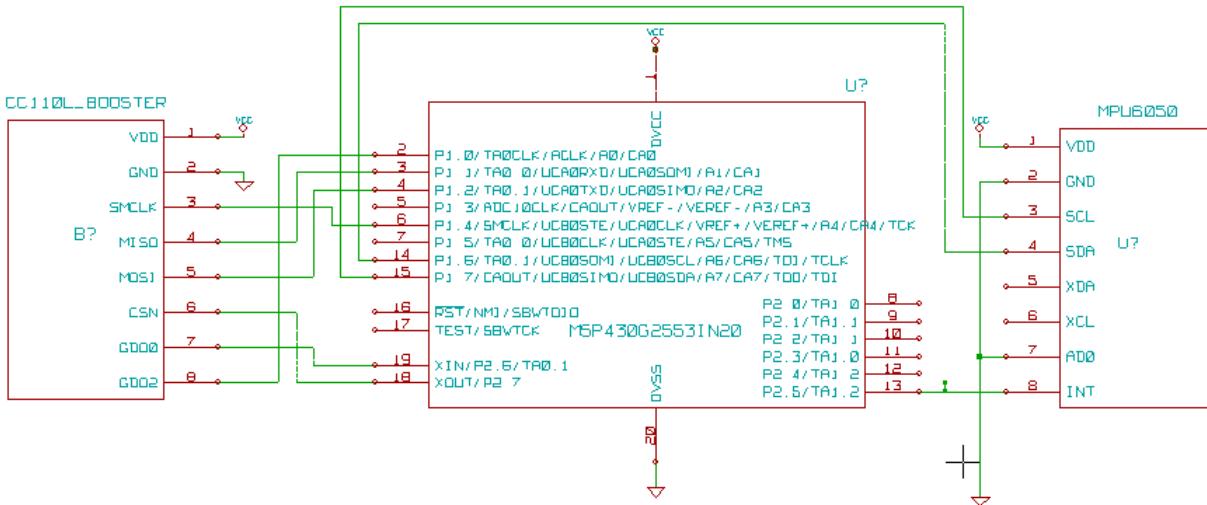
- GDO2 - linia ogólnego przeznaczenia - pełni funkcję zależną od konfiguracji.

Do linii tej przypisany jest sygnał CHIP_RDYn. Sygnalizuje on gotowość urządzenia. Gdy urządzenie jest gotowe, dozwolone jest sterowanie nim przy użyciu magistrali SPI.

2.3. Budowa systemu pomiarowego

Skonstruowany system pomiarowy składa się z dwóch oddzielnych części, komunikujących się drogą radiową:

- urządzenia pomiarowego;
- Urządzenia pośredniczącego.



Rysunek 2.1: Schemat połączeń urządzenia pomiarowego.

Zadaniem urządzenia pomiarowego jest zbieranie danych z akcelerometru i żyroskopu, a następnie przesyłanie ich do urządzenia pośredniczącego. Urządzenie pośredniczące ma za zadanie odebrać dane oraz przesyłać je dalej, do komputera klasy PC, przy pomocy portu szeregowego. Potrzeba wykorzystania urządzenia pośredniczącego wynika z wykorzystania modułu radiowego przystosowanego do energooszczędnej pracy, nie posiadającego możliwości bezpośredniej komunikacji z komputerem. Wykorzystany moduł, wyprodukowany przez firmę Anaren, jest jednym z najbardziej energooszczędnnych urządzeń tego typu dostępnych na rynku. Użycie go jest korzystne wszędzie tam, gdzie pożądany jest długi czas działania przy zasilaniu baterijnym. Znajduje on zastosowanie m. in. w zewnętrznych czujnikach, systemach alarmowych itp.

2.3.1. Urządzenie pomiarowe

Schemat urządzenia pomiarowego przedstawiony został na rysunku 2.1. Składa się ono z:

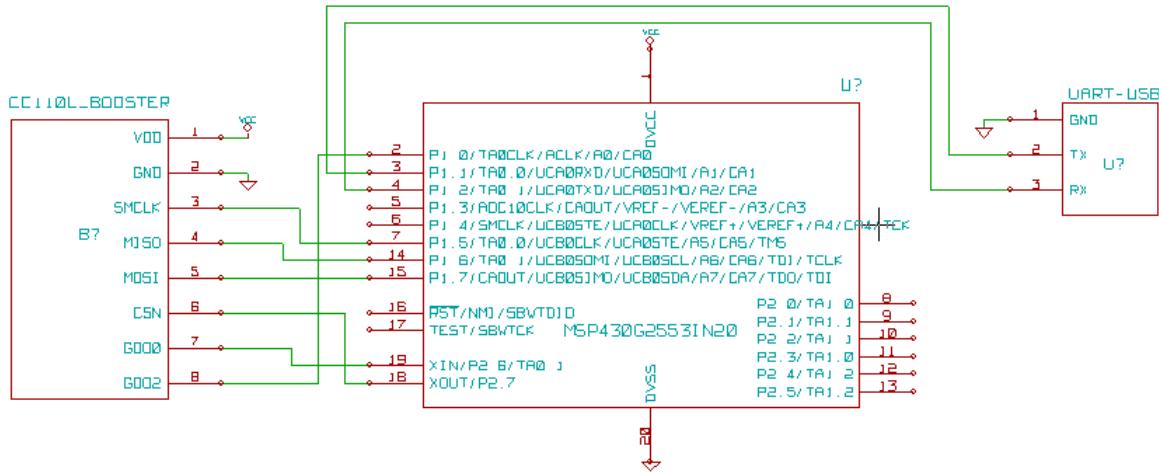
- mikrokontrolera MSP430G2553;
- modułu czujnika ruchu MPU6050;
- modułu radiowego A110RLR09A.

2.3.2. Urządzenie pośredniczące

Schemat urządzenia pośredniczącego przedstawiony został na rysunku 2.2. Składa się ono z:

- mikrokontrolera MSP430G2553
- modułu radiowego A110RLR09A
- konwertera UART-USB

Waga zbudowanego urządzenia to około 16g bez baterii. Jego wymiary to 70x30x20 mm.



Rysunek 2.2: Schemat połączeń urządzenia pośredniczącego.

2.4. Przepustowość transmisji danych

Poniżej przedstawione są możliwości łącz cyfrowych obecnych w urządzeniach. Sam czujnik ruchu pozwala na próbkowanie danych z częstotliwością do 8kHz. Co istotne, akcelerometr dostarcza pomiarów z częstotliwością nie większą niż 1kHz, zatem powyżej tej częstotliwości, ten sam pomiar jest próbkowany kilkukrotnie. Żydroskopy mogą pracować poprawnie w pełnym zakresie częstotliwości próbkowania[5].

Częstotliwość próbkowania całego systemu pomiarowego jest jednak ograniczona przez wykorzystane w nim media transmisji cyfrowej. Możliwości każdego z nich opisane są poniżej.

2.4.1. Magistrala I²C

Magistrala I²C wykorzystywana jest do przesyłu danych z modułu czujnika ruchu do mikrokontrolera zainstalowanego w urządzeniu pomiarowym.

Taktowanie zegara magistrali I²C między tymi urządzeniami zostało ustawione na 387,10 kHz. Maksymalna dopuszczalna wartość to 400kHz[5].

Funkcja odczytu pojedynczego rejestru opisana w podsekcji 3.4.5 na stronie 29 dokonuje opisanych niżej transmisji na magistrali I²C:

- wysłanie sekwencji startu;
- zaadresowanie urządzenia (1 bit R/W, 7 bitów adresu, 1 bit ACK), przesłanie adresu rejestru (8 bitów danych, 1 bit ACK);
- wysłanie sekwencji powtózonego startu;
- zaadresowanie urządzenia (1 bit R/W, 7 bitów adresu, 1 bit ACK), odebranie żądanego bajtu (8 bitów danych, 1 bit ACK);
- wysłanie sekwencji stopu;

Funkcja odczytu sekwencji 12 bajtów, zawierających pomiary z czujników modułu składa się z sygnałów:

- sekwencja startu;
- zaadresowanie urządzenia (1 bit R/W, 7 bitów adresu, 1 bit ACK), przesłanie adresu pierwszego rejestru (8 bitów danych, 1 bit ACK);

- sekwencja powtórzonego startu;
- zaadresowanie urządzenia (1 bit R/W, 7 bitów adresu, 1 bit ACK), odebranie 12 bajtów (12 transmisji, z których każda składa się z 8 bitów danych i 1 bitu ACK);
- sekwencja stopu.

Podczas pojedynczej obsługi przerwania informującego o obecności nowych danych w kolejce FIFO, funkcja odczytu pojedynczego rejestru wywoływana jest trzykrotnie (odczyt dwóch bajtów zajętości kolejki oraz jednego bajtu zwierającego flagi przerwań). Następnie wywoływana jest funkcja odczytu sekwencji 12 bajtów z kolejki FIFO. Sumaryczny czas trwania całej operacji został odczytany z analizatora stanów logicznych i wynosi $690,4\mu s$. Zakładamy istnienie $50\mu s$ przerwy między kolejnymi transmisjami. Pozwala to uniknąć przepełnienia kolejki transmisji. Otrzymujemy ograniczenie częstotliwości próbkowania o wartości 1350Hz. Jest to najsilniejsze z ograniczeń mediów transmisji cyfrowej wykorzystanych w tej pracy. Stanowi ono zatem górne ograniczenie częstotliwości próbkowania całego systemu pomiarowego.

2.4.2. Magistrala SPI - urządzenie pomiarowe

Dane przesyłane są z mikrokontrolera do modułu radiowego w urządzeniu pomiarowym przez magistralę SPI. Taktowana jest ona częstotliwością zegara wynoszącą 2MHz. Największe możliwe taktowanie zegara dla tego urządzenia to 10 MHz[11]. Przesłanie jednej ramki drogą radiową składa się z:

- zaadresowania rejestru kolejki TXFIFO: (1 bajt);
- przesłania ramki składającej się z bajtu długości oraz 12 bajtów danych (w sumie 13 bajtów);
- przesłania sygnału komendy aktywującej transmisję radiową (1 bajt).

Łącznie, w jednej ramce przesyłanych jest 15 bajtów. Czas trwania tej operacji to $61,2\mu s$. Zatem to medium, dla zastosowanych ustawień przepustowości, przy założeniu $15\mu s$ przerwy między kolejnymi transmisjami, nakłada na system ograniczenie częstotliwości próbkowania rzędu 13kHz.

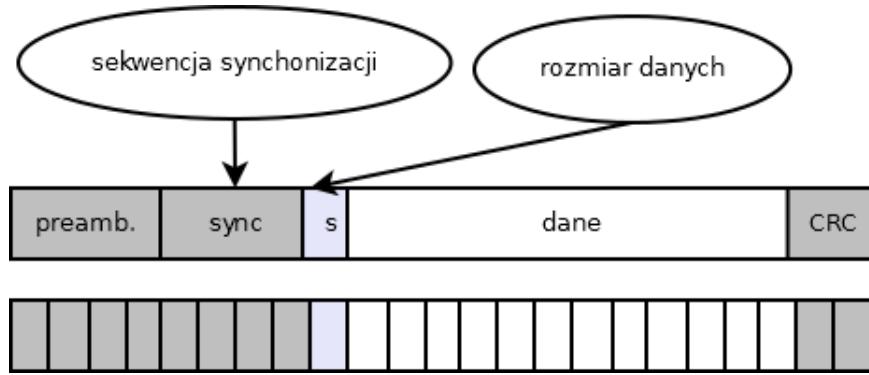
2.4.3. Magistrala SPI - urządzenie pośredniczące

Dane przesyłane są w tym przypadku po magistrali SPI taktowanej zegarem 2MHz. Odebranie pojedynczej ramki przesłanej drogą radiową do urządzenia CC110L przez mikrokontroler składa się z:

- zaadresowania rejestru RXBYTES oraz odczytu jego wartości (2 bajty);
- zaadresowania rejestru kolejki TXFIFO oraz odczytu bajtu długości (2 bajty);
- zaadresowania rejestru kolejki TXFIFO oraz odczytu 12 kolejnych bajtów (13 bajtów);
- zaadresowania rejestru kolejki TXFIFO oraz odczytu 2 bajtów CRC (3 bajty).

Łącznie, w jednej ramce przesyłanych jest 20 bajtów.

Czas trwania tej operacji to $81,6\mu s$. Zatem to medium, dla zastosowanych ustawień przepustowości, przy założeniu $15\mu s$ przerwy między kolejnymi transmisjami, nakłada na system ograniczenie częstotliwości próbkowania rzędu 10kHz.



Rysunek 2.3: Ramka przesyłana drogą radiową.

2.4.4. Komunikacja radiowa

Ramka przesyłana drogą radiową przy zastosowanej konfiguracji urządzenia składa się z:

- preambuły (4 bajty);
- słowa synchronizacji (4 bajty);
- bajtu długości danych (1 bajt);
- bajtu adresu - nie używany (0 bajtów);
- danych (12 bajtów);
- sumy kontrolnej CRC16 (2 bajty).

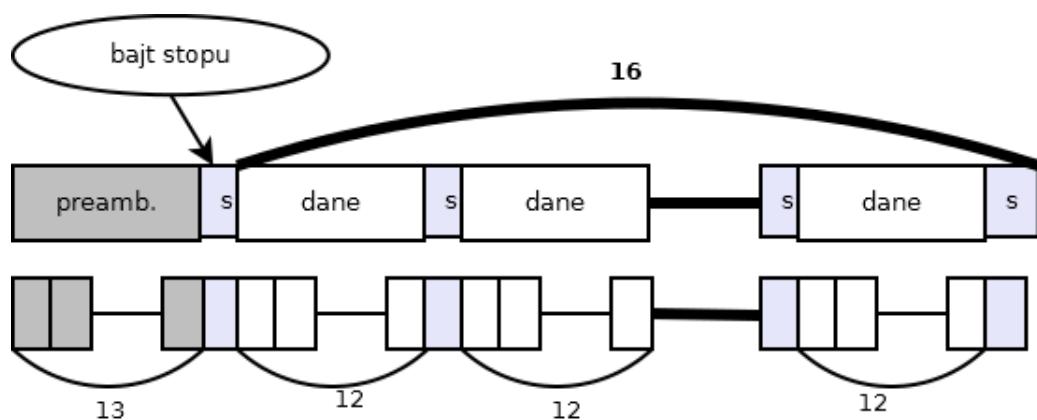
Łączna ilość bajtów przesyłanych w jednej ramce wynosi 23. Szybkość transmisji radiowej została skonfigurowana do 500kbaud. Maksymalna przepustowość osiągalna dla tego urządzenia wynosi 600kbaud, jednak wymaga ona innej konfiguracji parametrów radia[11]. Czas transmisji pojedynczej ramki dla 500kbaud wynosi około $368\mu\text{s}$. Zakładając odstęp między ramkami rzędu $50\mu\text{s}$, maksymalna osiągalna częstotliwość próbkowania dla tego medium jest ograniczona do około 2390 Hz.

2.4.5. Pakiet przesyłany przez port szeregowy

Pędkość transmisji portu szeregowego ustawiona została na 2Mbaud. Pojedynczy pakiet składa się z:

- preambuły (13 bajtów);
- 1 bajtu stopu;
- 16 ramek o następującej budowie:
 - dane (12 bajtów);
 - 1 bajt stopu.

Łączny rozmiar pakietu to 212 bajtów, przy czym każdy z nich zawiera 16 ramek z pomiarami. Czas wysyłania jednego pakietu to $848\mu\text{s}$. Ograniczenie częstotliwości próbkowania nakładane przez to medium przy $50\mu\text{s}$ przerwy między pakietami to 17,8kHz.



Rysunek 2.4: Pakiet przesyłany przez UART.

3. Sprzętowe moduły komunikacji cyfrowej

3.1. Moduły komunikacji szeregowej w MSP430a

Mikrokontrolery serii MSP430 wyposażone są w moduły, które częściowo lub w pełni sprzętowo wspierają transmisję szeregową. Każdy moduł, w zależności od konfiguracji, może obsługiwać różne protokoły transmisji. Moduły, które realizują komunikację po części sprzętowo noszą nazwę USI (Universal Serial Interface). USI składa się z następujących funkcjonalności ułatwiających implementację komunikacji szeregowej[13]:

- 8- lub 16-bitowy rejestr przesywny do transmisji/odbioru strumieni danych;
- sprzętowe funkcjonalności ułatwiające implementację komunikacji po magistrali SPI oraz I²C;
- możliwość generowania przerwań;
- generator zegara;
- detekcję stanu łącza - sygnałów START, STOP, utraty arbitrażu;
- możliwość działania jako urządzenie podzielone w trybie oszczędzania energii, bez udziału wewnętrznego zegara.

Pomimo tych udogodnień, obsługa komunikacji musi być zrealizowana w większości przez oprogramowanie.

Całkowite wsparcie sprzętowe zapewniane jest przez moduły USCI (Universal Serial Communication Interface) oraz USART (Universal Synchronous/Asynchronous Recieve/Transmit). W tej pracy wykorzystywane są wyłącznie moduły USCI.

Istnieje kilka rodzajów USCI. Każdy z nich oznaczony jest inną literą (np. USCI_A, USCI_B). Różnią się one wspieranymi protokołami komunikacji. Jeśli w mikrokontrolerze występuje kilka modułów tego samego typu, są one kolejno numerowane (np. USCI_A0, USCI_A1)[13].

Mikrokontroler MSP430G2553, który został użyty zarówno w urządzeniu pomiarowym, jak i w urządzeniu pośredniczącym wyposażony jest w dwa moduły: USCI: USCI_A0 oraz USCI_B0.

USCI_A0 obsługuje następujące tryby komunikacji[12]:

- SPI (3- oraz 4-pin);
- UART;
- enhanced UART;
- IrDA.

USCI_B0 obsługuje:

- SPI (3- oraz 4-pin);
- I²C.

Mikrokontroler w urządzeniu pomiarowym wykorzystuje moduł USCI_A0 do komunikacji z modułem radiowym przez SPI oraz USCI_B0 do odczytu danych z czujnika ruchu MPU-6050 oraz jego konfiguracji po magistrali I²C.

W urządzeniu pośredniczącym USCI_A0 obsługuje port szeregowy, natomiast USCI_B0 odpowiada za obsługę magistrali SPI pomiędzy mikrokontrolerem a modułem radiowym.

3.2. Biblioteka obsługi portu szeregowego z wykorzystaniem modułu USCI

3.2.1. Ogólny opis systemu UART

UART (ang. Universal Asynchronous Reciever/Transmitter) to szeroko wykorzystywany system asynchronicznej transmisji szeregowej. UART często wykorzystywany jest pod standardami warstwy fizycznej takimi jak: RS-232, RS-422 czy RS-485[17].

3.2.2. Zasada działania

Transmisja po magistrali UART zazwyczaj inicjowana jest przesłaniem bitu START (stan niski) trwającego jeden cykl zegara. Następnie przesyłane są kolejne bity danych począwszy od bitu najmłodszego, opcjonalny bit parzystości oraz sekwencja STOP (stan wysoki) o długości 1, 1,5 lub 2 bitów. Możliwy jest zarówno tryb pracy half duplex jak i full duplex. Transmisja przez UART jest asynchroniczna, zatem do poprawnej komunikacji niezbędna jest identyczna częstotliwość zegara na każdym urządzeniu końcowym[17].

3.2.3. Wykorzystanie

W tej pracy UART jest wykorzystywany do komunikacji pomiędzy mikrokontrolerem urządzenia pośredniczącego a komputerem klasy PC, który jest finalnym odbiorcą danych pomiarowych. Urządzenie pośredniczące wykorzystuje do komunikacji przez UART moduł USCI_A0 mikrokontrolera. Komputer wyposażony jest w konwerter UART-USB operujący napięciami 0-3.3V.

3.2.4. Inicjalizacja

Inicjalizacja modułu USCI do transmisji przez I²C odbywa się przez funkcję wykonującą następujące czynności:

- Konfigurację rejestrów USCI Control Register 0 oraz USCI Control Register 1.
Sposób konfiguracji został opisany w tabelach 3.1 i 3.2.
- Ustawienie odpowiedniego trybu na pinach SDA oraz SCL.
Odbiera się to przez ustawienie (rozumiane w tej pracy jako nadanie wartości jedynki logicznej danemu bitowi) bitu 1 oraz bitu 2 w rejestrze P1SEL (bity numerowane od 0).

- Konfigurację zegara modułu USCI.

Polega ona na wpisaniu odpowiednich wartości do rejestrów UCB0BR0 i UCB0BR1. Stanowią one odpowiednio młodszy i starszy bajt wartości, przez którą dzielony jest sygnał zegara źródłowego. Zegarem źródłowym jest SMCLK. Konfiguracja SMCLK jest domyślna - ma on wartość identyczną do zegara taktującego mikroprocesor[13], który ustawiony jest na 16MHz. Niniejsza biblioteka wykorzystuje przepustowość portu szeregowego o wartości 2Mbaud. Wartości rejestrów UCB0BR0 i UCB0BR1 wynoszą odpowiednio 8 i 0.

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCPEN	7	Aktywacja trybu parzystości 0 - parzystość wyłączona; 1 - parzystość włączona.	0
UCPAR	6	Wybór pomiędzy parzystością a nieparzystością 0 - nieparzystość; 1 - parzystość.	0
UCMSB	5	Wybór pierwszeństwa MSB 0 - pierwszeństwo LSB; 1 - pierwszeństwo MSB.	0
UC7BIT	4	Długość znaku 0 - 8-bit; 1 - 7-bit.	0
UCSPB	3	Wybór trybu 0 - Slave 1 - Master	0
UCMODEx	1-2	Tryb USCI 00 - tryb UART; 01 - tryb zwolnionej magistrali; 10 - tryb magistrali adresowanej bitowo; 11 - tryb UART z automatyczną detekcją prędkości transmisji.	00
UCSYNC	0	Wybór trybu synchronicznego/asynchronicznego 0 - tryb asynchroniczny; 1 - tryb synchroniczny.	0

Tablica 3.1: USCI Control Register 0 - ustawienia dla UART

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCSSELx	7-6	Wybór źródła zegara 00 - NA; 01 - ACLE; 10 - SMCLK; 11 - SMCLK.	11
UCRXEIE	5	Tryb odbioru znaków zawierających błędy 0 - tryb odrzucania błędnych znaków; 1 - tryb przyjmowania błędnych znaków;	0
UCBRKIE	4	Aktywacja przerwania od znaku przerwy 0 - wyłączone; 1 - włączone.	0
UCDORM	3	Tryb uśpenia 0 - wyłączony; 1 - włączony.	0
UCTXADDR	2	Aktywacja trybu adresacji 0 - następna wysłana ramka zawiera dane; 1 - następna wysłana ramka zawiera adres.	0
UCTXBRK	1	Aktywacja transmisji przerwy 0 - następna ramka to nie przerwa; 1 - następna ramka będzie przerwą.	0
UCSWRST	0	Reset 0 - tryb normalnej pracy; 1 - stan resetu.	0

Tablica 3.2: USCI Control Register 1 - ustawienia dla UART

3.2.5. Wysłanie bajtu

Wysłanie bajtu przez port szeregowy odbywa się z wykorzystaniem funkcji, która jako argument pobiera wartość bajtu przeznaczonego do wysłania. Przebieg wywołania funkcji:

- Program oczekuje aż bufor przesyłu danych będzie wolny (flaga UCA0TXIFG w rejestrze IFG2 ma wartość 0).
- Następuje wpisanie żądanej wartości do rejestru UCA0TBUF.

3.3. Biblioteka obsługi protokołu I²C z wykorzystaniem modułu USCI - funkcje niskiego poziomu

3.3.1. Opis protokołu

I²C (Inter-Integrated Circuit) to protokół komunikacji szeregowej korzystający jedynie z dwóch linii sygnałowych. Protokół został opracowany przez firmę Phillips w 1982 roku. Od tego czasu specyfikacja protokołu była regularnie rozwijana. Najnowszą specyfikacją jest wersja 5 wprowadzona w 2012 roku.

Komunikacja możliwa jest jedynie w trybie half-duplex. Protokół nie wymaga linii wyboru urządzenia podrzędnego (Chip Select). Zamiast tego, urządzenia są adresowane przy pomocy 7-bitowego lub 10-bitowego adresu, który jest unikalny dla każdego z nich. Dopuszczalna jest teoretycznie dowolna ilość urządzeń typu master i slave podpiętych do jednej magistrali, o ile spełniona zostanie unikalność adresu i wymagane parametry fizyczne łącza. Transmitowane dane podzielone są na 8-bitowe bajty[8].

3.3.2. Zasada działania

Fizycznie, magistrala składa się z linii:

- SCL - linia zegara kontrolowana przez urządzenie nadzorujące;
- SDA - linia danych.

Obie linie podłączone są do napięcia zasilania przez rezystor typu pull-up. Każde urządzenie podłączone jest do magistrali za pomocą wejść/wyjść typu otwarty-dren. Taka konfiguracja pozwala na sprawną komunikację w różnych kierunkach, po tych samych liniach, bez występowania konfliktów napięć. Ponadto, pozwala ona na jednoczesną transmisję danych oraz nasłuchiwanie łącza pod kątem wystąpienia ewentualnych kolizji. Podanie logicznego zera na jedną z linii odbywa się przez zwarcie jej z uziemieniem. Wystawienie logicznej jedynki to po prostu brak zwarcia w danym momencie.

Komunikacja po magistrali I²C inicjowana jest przez urządzenie nadzorujące przez wysłanie sekwencji START. Wszystkie urządzenia podrzędne podpięte do magistrali przechodzą wtedy w tryb nasłuchiwanego. Urządzenie nadzorujące wysyła następnie bajt dostępu składający się z bitu R/W (decydującego o poleceniu transmisji lub odbioru) oraz 7-bitowego adresu urządzenia, z którym ma zostać nawiązana komunikacja. Jeśli któreś urządzenie podrzędne posiada przesłany adres, wysyła ono potwierdzenie w postaci sygnału ACK (stan niski).

Następnie, jeśli przesłany bit R/W był w stanie niskim, urządzenie nadzorujące wysyła dane do urządzenia podrzędnego. Po każdym przesłanym bajcie urządzenie podrzędne wysyła potwierdzenie (w postaci bitu o nazwie ACK i wartości logicznej 0).

Jeśli przesłany bit R/W był w stanie wysokim, to urządzenie podrzędne dokonuje transmisji. Urządzenie nadzorujące potwierdza wtedy odbiór bajtu danych przesłaniem bitu ACK, poza ostatnim bajtem, kiedy sygnalizowany jest koniec odbioru sygnałem NACK (stan wysoki)[8].



Rysunek 3.1: Przykładowy przebieg sygnałów na magistrali I²C

3.3.3. Opis biblioteki

Biblioteka obsługująca protokół I²C została napisana bazując na bibliotece udostępnianej przez firmę Texas Instruments[10].

Wykorzystanie modułu USCI pozwala na oszczędzenie czasu, który mikroprocesor poświęca na przesyłanie danych. Jest to możliwe dzięki wykorzystaniu przerwań informujących o pojawienniu się nowych danych w buforze odbiorczym modułu USCI lub o zwolnieniu bufora transmisyjnego.

Dzięki temu, podczas trwania transmisji/odbioru, między obsługą kolejnych przerwań mikroprocesor jest zwolniony i może wykonywać inne operacje. Oprogramowanie może sprawdzić czy transmisja została już zakończona przez odczyt flagi zajętości modułu USCI (bit UCBBUSY rejestru UCB0STAT).

Funkcje niskiego poziomu obsługi I²C; służą do realizacji trzech prostych operacji: inicjalizacji modułu USCI, transmisji danych do urządzenia podległego oraz odbioru danych z urządzenia podległego. Przesyłane lub odbierane dane nie są w tym przypadku przypisane do żadnych rejestrów urządzenia podległego, które mają zostać odczytane/zapisane. Inicjowany jest jedynie przesył/odbiór bajtów do/z urządzenia podległego o określonym adresie.

3.3.4. Bajt stanu

Biblioteka przechowuje w pamięci o zakresie globalnym swój bajt stanu o nazwie SW_STATE. Funkcje jego poszczególnych bitów zostały określone przy pomocy dyrektywy #define języka C w następujący sposób:

- BIT1 - SW_TX_END - flaga końca transmisji;
- BIT2 - SW_RX_END - flaga końca odbioru;
- BIT7 - SW_RSTT - opcja „nie wysyłaj STOP po transmisji” - po transmisji zamiast sygnału STOP przesłany zostanie sygnał REPEATED START;
- BIT5 - SW_NACKFG - flaga informująca, że wystąpił stan NACK;

Przy starcie programu bajt ten jest inicjalizowany wartością 0.

3.3.5. Inicjalizacja

Inicjalizacja modułu USCI do transmisji po magistrali I²C odbywa się przez funkcję wykonującą następujące czynności:

- Konfiguracja rejestrów zarządzających modułem.

Dla urządzenia pomiarowego konfiguracja modułu USCI do komunikacji po magistrali I²C odbywa się przez odpowiednie ustawienie wartości rejestrów USCI_B0 Control Register 0 oraz USCI_B0 Control Register 1. Tabele 3.3 oraz 3.4 przedstawiają funkcje poszczególnych bitów tych rejestrów jak również wartości wybrane dla omawianego projektu.

- Ustawienie trybu USCI na pinach SDA oraz SCL.

Odbiera się to przez ustawienie dwóch najstarszych bitów w rejestrach P1SEL oraz P2SEL.

- Konfiguracja zegara modułu USCI.

Dla wykorzystanego modułu czujnika, taktowanie zegara magistrali powinno być mniejsze od 400kHz. Wartość podzielnika zegara 16MHz ustawiono na 39. Wartości rejestrów UCB0BR0 i UCB0BR1 wynoszą odpowiednio 39 i 0 (w zapisie dziesiętnym).

- Konfiguracja adresu urządzenia podrzędnego.

Do rejestrów UCB0I2CSA zostaje wpisany adres modułu czujnika ruchu. Ponieważ w module MPU-6050 pin AD0 czujnika jest połączony do uziemienia, adres ten to 0x68 (w zapisie heksadecymalnym).

- Konfiguracja przerwań mikrokontrolera.

Ustawienie bitu UCNACKIE w rejestrze UCB0I2CIE - aktywacja przerwania w przypadku wystąpienia stanu NACK.

3.3.6. Transmisja danych

Transmisja danych rozpoczynana się wywołaniem funkcji przyjmującej jako argumenty wskaźnik na tablicę danych przeznaczonych do wysłania oraz zmienną określającą ilość bajtów przeznaczonych do transmisji. Przebieg funkcji:

- Flaga SW_TX_END jest zerowana.
- Zerowane są wszystkie bity rejestrów UCB0CTL1 oprócz bitów decydujących o trybie pracy (wyborze źródła zegara).
- Następuje ustawienie bitu UCB0TXIE w rejestrze IE2 mikrokontrolera - aktywacja przerwania od zakończenia transmisji bajtu danych.
- Funkcja zapisuje w odpowiednim miejscu pamięci o zakresie globalnym wskaźnik na tablicę danych, które mają zostać wysłane oraz informację o jej rozmiarze. Wartości te są później wykorzystywane w obsłudze przerwań.
- Na końcu ustawiane są flagi UCTR i UCTXSTT rejestrów UCB0CTL1, co inicjuje rozpoczęcie transmisji.

Dalsza transmisja odbywa się wyłącznie podczas obsługi przerwań otrzymanych od modułu USCI.

Gdy zostanie otrzymany sygnał przerwania od modułu, informujący o wolnym buforze transmisji, kod obsługujący przerwanie przekazuje kolejny bajt do modułu USCI. Inkrementowany jest wskaźnik na tablicę danych, a licznik bajtów pozostałych do wysłania jest dekrementowany. W przypadku, gdy licznik bajtów pozostałych do przesłania jest równy zero, nie jest już przesyłany kolejny bajt. Zamiast

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCA10	7	Tryb własnej adresacji 0 - adresacja 7-bitowa; 1 - adresacja 10-bitowa.	1
UCSLA10	6	Tryb adresacji urządzeń podrzędnych 0 - adresacja 7-bitowa; 1 - adresacja 10-bitowa.	0
UCMM	5	Konfiguracja środowiska multi-master 0 - tryb single-master - jedno urządzenie nadziedne; 1 - tryb multi-master - wiele urządzeń nadziednych.	0
Unused	4	Nie wykorzystywane	-
UCMST	3	Wybór trybu 0 - Slave; 1 - Master.	1
UCMODEx	1-2	Tryb USCI 00 - 3-pin SPI; 01 - 4-pin SPI z aktywnym stanem UCxSTE wysokim; 10 - 4-pin SPI z aktywnym stanem UCxSTE niskim; 11 - I ² C.	11
UCSYNC	0	Wybór trybu synchronicznego/asynchronicznego 0 - tryb asynchroniczny; 1 - tryb synchroniczny.	1

Tablica 3.3: USCI Control Register 0 - ustawienia dla I²C

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCSSELx	7-6	Wybór źródła zegara 00 - NA; 01 - ACLE;urządzeniu 10 - SMCLK; 11 - SMCLK.	10
Unused	5	Nie wykorzystywane	-
UCTR	4	Tryb transmisji/odbioru 0 - odbiór; 1 - transmisja.	0/1
UCTXNACK	3	Wysłanie stanu NACK w trybie master 0 - brak oczekującego wysyłania NACK; 1 - zaplanowane i oczekujące wysłanie NACK.	0/1
UCTXSTP	2	Wysłanie stanu STOP w trybie master 0 - brak oczekującego wysyłania STOP; 1 - zaplanowane i oczekujące wysłanie STOP.	0/1
UCTXSTT	1	Wysłanie stanu START w trybie master 0 - brak oczekującego wysyłania START; 1 - zaplanowane wysłanie START;.	0/1
UCSWRST	0	Reset 1 - stan resetu; 0 - tryb normalnej pracy.	0/1

Tablica 3.4: USCI Control Register 1 - ustawienia dla I²C

tego, w rejestrze UCB0CTL1 ustawiany jest bit UCTXSTP, co powoduje, że moduł USCI wysyła na linii I²C sekwencję STOP. Gdy flaga SW_RSTT jest ustawiona bit UCTXSTP nie jest ustawiany, ponieważ oznacza to, że planowane jest wysłanie sekwencji REPEATED START poprzedzającej dalszą transmisję lub odbiór na magistrali. Na końcu, ustawiana jest flaga SW_TX_END.

Podeczas trwania transmisji, główna część programu może sprawdzać czy transmisja została już zakończona przez odczyt bitu UCBBUSY z rejestrów UCB0STAT lub przez sprawdzanie flagi SW_TX_END.

3.3.7. Odbiór danych

Odbiór danych odbywa się przez wywołanie funkcji, która jako argumenty pobiera wskaźnik na tablicę przeznaczoną do zapisu danych oraz zmienną opisującą ilość bajtów, jaka ma zostać odebrana. Działanie funkcji:

- Flaga SW_RX_END jest zerowana;
- Zerowane są wszystkie bity rejestrów UCB0CTL1 oprócz bitów decydujących o trybie pracy (wyborze źródła zegara).
- Następuje ustawienie bitu UCB0RXIE w rejestrze IE2 oraz wyzerowanie wszystkich innych jego bitów - aktywacja przerwania od zakończenia odbioru bajtu.
- Funkcja zapisuje w odpowiednim miejscu pamięci o zakresie globalnym wskaźnik do tablicy przeznaczonej na dane oraz informację o jej rozmiarze. Zmienne te są później wykorzystywane w obsłudze przerwań.
- Jeśli do przesłania jest tylko jeden bajt, to licznik bajtów do wysłania zostaje ustawiony na wartość 0. Dalej wysyłana jest sekwencja START rozpoczęta ustawieniem bitu UCTXSTT w rejestrze UCB0CTL1. Program oczekuje na wyzerowanie tego bitu przez sprzęt. Sygnalizuje ono zakończenie wysyłania sekwencji. Zgodnie z wymaganiami maszyny stanów modułu USCI[10] przed odebraniem ostatniego bajtu (czyli w tym przypadku również pierwszego) ustawiany jest bit UCTXSTP w celu zapewnienia poprawnego wysłania sekwencji STOP.
- Jeśli do przesłania jest więcej niż jeden bajt, licznik bajtów zostaje zmniejszony o 2.
- Na końcu ustawiana jest flaga UCTXSTT rejestrów UCB0CTL1, co inicjuje rozpoczęcie transmisji;
- Dalsza transmisja odbywa się wyłącznie podczas obsługi przerwań otrzymanych od modułu USCI.

Gdy zostanie otrzymany sygnał przerwania od modułu USCI informujący o otrzymaniu kolejnego bajtu, kod obsługujący przerwanie zapisuje ten bajt z bufora oraz zapisuje go do pamięci. Następnie, inkrementowany jest wskaźnik na tablicę, do której zapisywane są dane. Licznik bajtów pozostałych do odebrania jest dekrementowany. Następnie:

- Jeżeli licznik bajtów do odbioru ma wartość 1, oznacza to że do odebrania został ostatni bajt. Wtedy, w UCB0CTL1 ustawiany jest bit UCTXSTP.
- Jeżeli licznik bajtów do odbioru ma wartość 1, flaga SW_RX_END jest ustawiana.

Podeczas trwania transmisji, główna część programu może sprawdzać czy transmisja została już zakończona przez odczyt bitu UCBBUSY z rejestrów UCB0STAT lub przez sprawdzanie flagi SW_RX_END.

3.3.8. Obsługa stanu NACK

Gdy zostanie odebrany sygnał NACK, moduł USCI generuje przerwanie oraz ustawia flagę UCNACKIFG w rejestrze UCB0STAT. Procedura obsługi przerwania przebiega w następujący sposób:

- Na magistrali wysyłana jest sekwencja STOP;
- Zerowana jest flaga UCNACKIFG w rejestrze UCB0STAT;
- Ustawiana jest flaga informująca, że pojawił się stan NACK (SW_NACKFG) w bajcie SW_STATE.

3.4. Biblioteka obsługi protokołu I²C z wykorzystaniem modułu USCI - funkcje wysokiego poziomu

3.4.1. Opis biblioteki

Funkcje wysokiego poziomu w odpowiedni sposób wykorzystują funkcje poziomu niskiego. W tej warstwie komunikacji istnieje pojęcie rejestrów urządzenia podległego. Funkcje niskiego poziomu są wywoływane naprzemiennie, aby uzyskać zapis/odczyt pojedynczego lub sekwencji bajtów pod określonym adresem rejestrów urządzenia podległego.

3.4.2. Odczyt rejestru

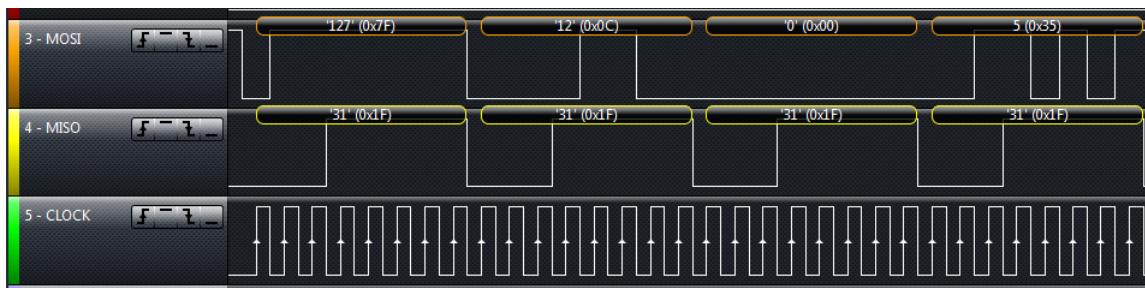
Funkcja odczytu rejestru jako argument pobiera adres rejestru, który ma zostać odczytany. Zwraca odczytaną wartość. Działa ona w następujący sposób:

- Ustawiany jest bit SW_RSTT w bajcie SW_STATE, aby między transmisją a odbiorem pojawił się sygnał REPEATED START zamiast kolejno STOP i START.
- Adres rejestru przeznaczonego do odczytu kopiowany jest do pomocniczej tablicy.
- Funkcja czeka, aż moduł USCI będzie wolny.
- Przy pomocy funkcji transmisji poziomu niskiego do urządzenia podległego wysłany zostaje adres rejestru (w formie jednego bajtu), który ma zostać odczytany.
- Funkcja czeka aż moduł USCI będzie wolny czyli aż transmisja zakończy się;
- Flaga SW_RSTT zostaje wyzerowana;
- Przy pomocy funkcji odbioru niskiego poziomu, zostaje rozpoczęty odbiór jednego bajtu z urządzenia podległego;
- Następuje oczekiwanie na zakończenie odbioru oraz zwrócenie odebranej wartości.

3.4.3. Odczyt sekwencji bajtów

Funkcja odczytu grupy rejestrów jako argumenty przyjmuje wskaźnik na tablicę, do której mają zostać zapisane dane, zmienną informującą o ilości bajtów przeznaczonych do odczytu oraz adres rejestru, który ma zostać przesłany do urządzenia podległego. Należy pamiętać, aby tablica przekazana do funkcji miała wielkość pozwalającą na pomieszczenie odebranych danych. Funkcja działa w sposób podobny do funkcji odczytu pojedynczego rejestru:

- Ustawiany jest bit SW_RSTT w bajcie SW_STATE, aby między transmisją a odbiorem pojawił się sygnał REPEATED START zamiast kolejno STOP i START.
- Adres rejestru przeznaczonego do odczytu kopiowany jest do pomocniczej tablicy.
- Funkcja czeka, aż moduł USCI będzie wolny.



Rysunek 3.2: Przykładowy przebieg sygnałów na magistrali SPI

- Przy pomocy funkcji transmisji poziomu niskiego, do urządzenia podległego wysłany zostaje adres rejestrów (w formie jednego bajtu), który ma zostać odczytany.
- Funkcja czeka aż moduł USCI będzie wolny czyli aż transmisja zakończy się;
- Flaga SW_RSTT zostaje wyzerowana;
- Przy pomocy funkcji odbioru niskiego poziomu zostaje rozpoczęty odbiór zadanej ilości bajtów z urządzenia podległego;
- Na końcu wywołania tej funkcji nie występuje oczekiwanie na zakończenie odbioru. Pozwala to mikroprocesorowi na wykonywanie innych poleceń w czasie trwania transmisji.

3.4.4. Odczyt sekwencji bajtów z oczekiwaniem na zakończenie

Funkcja ta wywołuje funkcję zwykłego odczytu sekwencji bajtów przekazując do niej swoje argumenty. Następnie program biernie oczekuje na zakończenie transmisji.

3.4.5. Zapis rejestru

Funkcja zapisu do rejestrów jako argumenty pobiera adres rejestrów oraz wartość, jaka ma zostać do niego wpisana.

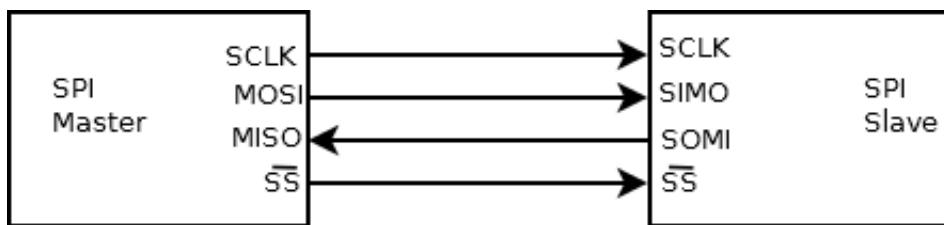
Zapis do rejestrów urządzenia podległego odbywa się w następujący sposób:

- Do pomocniczej tablicy, do pola o indeksie 0 zostaje wpisany bajt adresu.
- Do pola o indeksie 1 zostaje wpisana wartość, która ma zostać ustawiona na tym rejestrze.
- Następuje oczekiwanie na wyzerowanie bitu zajętości modułu USCI.
- Odbywa się transmisja dwóch bajtów z tablicy pomocniczej przy wykorzystaniu funkcji transmisji niskiego poziomu.
- Następuje oczekiwanie na zakończenie transmisji oraz zakończenie wywołania.

3.5. Programowa obsługa protokołu SPI dla urządzenia CC110L z wykorzystaniem sprzętowego modułu USCI

3.5.1. Ogólny opis protokołu

SPI (ang. Serial Peripheral Interface) jest protokołem szeregowej transmisji danych wykorzystywanym do komunikacji pomiędzy mikroprocesorem a urządzeniami periferyjnymi. Protokół SPI został wprowadzony wraz z mikroprocesorem Motorola 68000 w 1979 r.



Rysunek 3.3: Schemat magistrali SPI

Obecnie jest to jeden z najszerzej wykorzystywanych tego typu protokołów. Urządzenia wykorzystujące SPI mogą działać w jednym z dwóch trybów: w trybie urządzenia nadzawanego „Master” lub w trybie urządzenia podzewanego „Slave”. Dopuszczalne jest tylko jedno urządzenie nadzawne typu „Master” oraz teoretycznie dowolna ilość urządzeń podzewanego „Slave” podłączonych do jednej magistrali SPI.

Komunikacja odbywa się w trybie full-duplex, asynchronicznie, z sygnałem zegara kontrolowanym przez urządzenie nadzawne[8].

3.5.2. Zasada działania

Urządzenie nadzawne chcąc nawiązać komunikację z urządzeniem podzewanym w pierwszej kolejności wybiera je przy pomocy linii Slave Select. Urządzenie nadzawne ustala częstotliwość zegara, jego fazę i polaryzację odpowiednio do wymagań urządzenia Slave. Zazwyczaj, urządzenia wykorzystujące SPI wyposażone są w rejestr przesuwny pomiędzy liniami MOSI i MISO. Sparowane urządzenia tworzą zatem między sobą bufor cykliczny.

Dane wymieniane są równocześnie na liniach MOSI i MISO w kolejnych taktach zegara generowanych przez urządzenie nadzawne. W każdym cyklu zegara urządzenia dokonują jednocześnie transmisji oraz odbioru jednego bitu, nawet gdy zachodzi potrzeba komunikacji tylko w jedną stronę[8]. Magistrala składa się z wymienionych niżej linii:

- SCLK - sygnał zegara;
- MOSI (SIMO) - Master Output Slave Input - wyjście urządzenia nadzawanego, wejście urządzenia podzewanego;
- MISO (SOMI) - Master Input Slave Output - wejście urządzenia nadzawanego, wyjście urządzenia podzewanego;
- SS - Slave Select - stan niski aktywuje wybrane urządzenie podzewanego.

3.5.3. Opis biblioteki

Biblioteka obsługująca magistralę SPI została napisana w nieco odmienny sposób do biblioteki magistrali I²C. Podstawową różnicą jest to, że transmisja i odbiór danych nie odbywają się w przerwaniach. Podczas transmisji lub odbioru wywołana funkcja biernie oczekuje na zakończenie transmisji. Takie rozwiązanie jest prostsze w implementacji oraz oszczędza kłopotów związanych z konfliktami przerwań (patrz: podsekcja 4.1.3 na stronie 41). W tym przypadku nie jest jednak możliwe zwrócenie kontroli wykonania do nadzawanego fragmentu programu przed zakończeniem transmisji/odbioru. Ta utrata optymalności kodu jest rekompensowana przez fizyczne możliwości magistrali SPI. Maksymalna dopuszczalna przepustowość łącza dla urządzenia CC110L to aż 10MHz. Czasy transmisji są więc wielokrotnie krótsze od tych występujących na magistrali I²C.

Istnieją dwie wersje tej biblioteki, z których jedna przeznaczona jest dla urządzenia pomiarowego, a druga dla urządzenia pośredniczącego. Różnice w nich są minimalne i wynikają z wykorzystania różnych modułów USCI. Obejmują one nazwy rejestrów oraz numery pinów.

Podczas korzystania z zestawu BoosterPack przewidziane jest wykorzystanie modułu USCI_B0. Pozwala to na bezpośrednie podłączenie mikrokontrolera MSP430G2553 do płytki, bez wprowadzania żadnych modyfikacji w jej połączeniach. W urządzeniu pośredniczącym wykorzystany jest ten właśnie moduł.

Ponieważ USCI_A0 nie obsługuje protokołu I²C, w urządzeniu pomiarowym należało wykorzystać USCI_B0 do obsługi tego protokołu. Do obsługi magistrali SPI wykorzystano w tym urządzeniu moduł USCI_A0, co prowadziło do konieczności modyfikacji fizycznych połączeń płytki.

Poniżej, ze względu na minimalne różnice została opisana jedyńce wersja biblioteki stworzona dla urządzenia pośredniczącego.

3.5.4. Inicjalizacja

Inicjalizacja modułu USCI do transmisji po magistrali SPI odbywa się przez funkcję wykonującą następujące czynności:

- Skonfigurowanie rejestrów modułu USCI.

Dla urządzenia pomiarowego konfiguracja modułu USCI do komunikacji po magistrali SPI odbywa się przez odpowiednie ustawienie wartości rejestrów USCI_A0 Control Register 0 oraz USCI_A0 Control Register 1. Dla urządzenia pośredniczącego są to odpowiednio rejesty USCI_B0 Control Register 0 oraz USCI_B0 Control Register 1.

Tabele 3.5 i 3.6 przedstawiają funkcje poszczególnych bitów tych rejestrów oraz wartości wybrane dla omawianego projektu.

- Ustawienie odpowiedniego trybu na pinach MOSI, MISO oraz UCLK.

Odbywa się to przez ustawienie trzech najstarszych bitów w rejestrach P1SEL oraz P2SEL.

- Konfigurację zegara modułu USCI.

Polega ona na wpisaniu odpowiednich wartości do rejestrów UCB0BR0 i UCB0BR1. Wartość podzielnika ustawiono na 32 uzyskując taktowanie 500kHz. Wartości rejestrów UCB0BR0 i UCB0BR1 to odpowiednio 32 i 0.

3.5.5. Odczyt rejestru

Odczyt pojedynczego rejestru odbywa się przez wywołanie odpowiedniej funkcji, która jako argument pobiera adres rejestru urządzenia CC110L oraz zwraca odczytaną jego wartość. Działa ona w następujący sposób:

- Aktywowany jest pin Chip Select urządzenia CC110L.
- Program oczekuje aż urządzenie będzie gotowe (pin GDO_2 w stanie wysokim), a następnie aż bufor transmisji będzie wolny (flaga UCB0TXIFG w rejestrze IFG2 posiada wartość 0).
- Transmitowany jest jeden bajt dostępu składający się z najstarszego bitu (R/W) o wartości 1 (oznacza to rozkaz odczytu), kolejnego bitu („Burst access”) o wartości 0 (dostęp do pojedynczego rejestru) oraz 6-bitowego adresu rejestru, który ma zostać odczytany[11]. Transmisja inicjowana jest poprzez zapis do bufora UCB0TXBUF.
- Program oczekuje aż bufor transmisji zostanie zwolniony.
- Ze względu na specyfikację protokołu SPI[8] i maszyny stanów USCI[14], aby możliwy był odbiór danych, należy dokonać pustej transmisji przez wpisanie do bufora UCB0TXBUF dowolnej wartości.

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCCKPH	7	Faza zegara. 0 - zmiana stanu przy pierwszym zboczu UCLK, odczyt przy następnym; 1 - odczyt przy pierwszym zboczu zegara, zmiana stanu przy następnym.	1
UCCKPL	6	Polaryzacja zegara. 0 - stan nieaktywny niski; 1 - stan nieaktywny wysoki.	0
UCMSB	5	Pierwszeństwo MSB 0 - LSB pierwsze; 1 - MSB pierwsze.	1
UC7BIT	4	Długość znaku 0 - 8 bitów; 1 - 7-bitów.	0
UCMST	3	Wybór trybu 0 - slave; 1 - master.	1
UCMODEx	1-2	Tryb USCI 00 - 3-pin SPI; 01 - 4-pin SPI z aktywnym stanem UCxSTE wysokim; 10 - 4-pin SPI z aktywnym stanem UCxSTE niskim; 11 - I ² C.	00
UCSYNC	0	Wybór trybu synchronicznego/asynchronicznego 0 - tryb asynchroniczny; 1 - tryb synchroniczny.	1

Tablica 3.5: USCI Control Register 0 - ustawienia dla SPI

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
UCSELx	7-6	Wybór źródła zegara 00 - NA; 01 - ACLE; 10 - SMCLK; 11 - SMCLK.	10
Unused	5-1	Nie wykorzystywane.	-
UCSWRST	0	Reset 0 - tryb normalnej pracy; 1 - stan resetu.	0/1

Tablica 3.6: USCI Control Register 1 - ustawienia dla SPI

- Po jej zakończeniu program oczekuje, aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0).
- Z bufora UCB0RXBUF odczytana zostaje wartość zwrócona przez urządzenie. Pin Chip Select jest dezaktywowany stanem wysokim oraz następuje zwrócenie odczytanej wartości rejestru przez funkcję.

3.5.6. Odczyt grupy rejestrów

Funkcja służąca do odczytu grupy rejestrów pobiera następujące parametry: adres rejestru, od którego ma rozpocząć odczyt, adres tablicy, do której należy zapisać dane oraz ilość bajtów, jaką funkcja ma odczytać. Przebieg działania:

- Aktywowany jest pin Chip Select urządzenia CC110L.
- Program oczekuje aż urządzenie będzie gotowe (pin GDO_2 w stanie wysokim), a następnie aż bufor transmisji będzie wolny (flaga UCB0TXIFG w rejestrze IFG2 posiada wartość 0).
- Transmitowany jest jeden bajt dostępu składający się z najstarszego bitu (R/W) o wartości 1 (oznacza to rozkaz odczytu), kolejnego bitu („Burst access”) o wartości 1 (dostęp do grupy rejestrów) oraz 6-bitowego adresu pierwszego rejestru, który ma zostać odczytany. Transmisja inicjowana jest poprzez zapis do bufora UCB0TXBUF.
- Program oczekuje aż bufor transmisji zostanie zwolniony.
- Ze względu na specyfikację protokołu SPI i maszyny stanów USCI, aby możliwy był odbiór danych, należy dokonać pustej transmisji przez wpisanie do bufora UCB0TXBUF dowolnej wartości. Następująca procedura wykonywana jest w pętli, n razy, gdzie n to liczba bajtów, które mają zostać odczytane.
 - do bufora UCB0TXBUF wpisana zostaje dowolna wartość (w tym wypadku jest to 1);
 - Program oczekuje aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0).
 - Program oczekuje aż bufor transmisji zostanie zwolniony.
 - Do odpowiedniego miejsca tablicy wyjściowej zapisana zostaje wartość pobrana z rejestru UCB0RXBUF.
- Program oczekuje, aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0).
- Pin Chip Select jest dezaktywowany stanem wysokim oraz następuje zwrócenie odczytanej wartości rejestru przez funkcję.

3.5.7. Odczyt rejestru stanu

Aby uzyskać dostęp do rejestrów w zakresie 0x30 - 0x3D, w bajcie dostępu bit „burst access” decydujący o dostępie do pojedynczego lub grupy rejestrów zmienia swoją funkcję. Wykorzystywany jest do wyboru pomiędzy dostępem do rejestrów stanu (dla wartości 1), a sygnałami komend (dla wartości 0). Z tego względu możliwy jest jedynie pojedynczy dostęp do rejestrów stanu[11].

Funkcja odczytu rejestru stanu jest identyczna do funkcji odczytu zwykłego rejestru, poza miejscem gdzie wysyłany jest bajt dostępu. Bit „burst access” jest w nim ustawiony na wartość 1, a nie 0.

3.5.8. Zapis rejestru

Zapis rejestru obsługuje funkcja pobierająca dwa argumenty: adres rejestru oraz wartość, jaka ma zostać do niego zapisana. Przebieg działania:

- Aktywowany jest pin Chip Select urządzenia CC110L.
- Program oczekuje aż urządzenie będzie gotowe (pin GDO_2 w stanie wysokim), a następnie aż bufor transmisji będzie wolny (flaga UCB0TXIFG w rejestrze IFG2 posiada wartość 0).
- Transmitowany jest jeden bajt składający się z najstarszego bitu (R/W) o wartości 0 (oznacza to rozkaz zapisu), kolejnego bitu („Burst access”) o wartości 0 (dostęp do pojedynczego rejestru) oraz 6-bitowego adresu rejestru, który ma zostać zapisany. Transmisja inicjowana jest poprzez zapis tego bajtu do bufora UCB0TBUF.
- Program oczekuje aż bufor transmisji zostanie zwolniony.
- Zadana wartość rejestru wpisywana jest do bufora UCB0TBUF.
- Program oczekuje, aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0).
- Pin Chip Select jest dezaktywowany stanem wysokim oraz następuje zakończenie działania funkcji.

3.5.9. Zapis grupy rejestrów

Funkcja służąca do zapisu grupy rejestrów pobiera następujące parametry: adres rejestru, od którego ma rozpocząć zapis, adres tablicy zawierającej dane przeznaczone do zapisu oraz ilość bajtów, jaką funkcja ma zapisać. Przebieg działania:

- Aktywowany jest pin Chip Select urządzenia CC110L.
- Program oczekuje aż urządzenie będzie gotowe (pin GDO_2 w stanie wysokim), a następnie aż bufor transmisji będzie wolny (flaga UCB0TXIFG w rejestrze IFG2 posiada wartość 0).
- Transmitowany jest jeden bajt składający się z najstarszego bitu (R/W) o wartości 0 (oznacza to rozkaz zapisu), kolejnego bitu („Burst access”) o wartości 1 (dostęp do grupy rejestrów) oraz 6-bitowego adresu pierwszego rejestru, który ma zostać zapisany.
- Następująca procedura wykonywana jest w pętli, n razy, gdzie n to liczba bajtów, które mają zostać odczytane:
 - Program oczekuje aż bufor transmisji zostanie zwolniony.
 - Do bufora transmisji wpisywany jest kolejny bajt z tablicy bajtów przeznaczonych do transmisji.
- Program oczekuje aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0);
- Pin Chip Select jest dezaktywowany stanem wysokim oraz następuje zakończenie działania funkcji.

3.5.10. Wysłanie sygnału komendy

Sygnał komendy to instrukcja o długości jednego bajtu wysyłana do modułu CC110L. Przez zaadresowanie wewnętrznego rejestru odpowiadającego za sygnał komendy, zainicjowane zostanąewnętrzne procedury[11].

Funkcja wysyłająca sygnał komendy pobiera tylko jeden argument - wartość bajtu (komendy), który ma zostać wysłany. Dopuszczalne wartości bajtów komend zawierają się w zakresie 0x30-0x3D. Przebieg działania:

- Aktywowany jest pin Chip Select urządzenia CC110L.
- Program oczekuje aż urządzenie będzie gotowe (pin GDO_2 w stanie wysokim), a następnie aż bufor transmisji będzie wolny (flaga UCB0TXIFG w rejestrze IFG2 posiada wartość 0).
- Transmitowany jest jeden bajt reprezentujący komendę.
- Program oczekuje, aż moduł USCI przestanie być zajęty (flaga UCBUSY w rejestrze UCB0STAT przyjmuje wtedy wartość 0).
- Pin Chip Select jest dezaktywowany stanem wysokim oraz następuje zakończenie działania funkcji.

4. Urządzenia peryferyjne - opis sprzętu i bibliotek

4.1. Biblioteka obsługi czujnika MPU-6050

4.1.1. Inicjalizacja

Inicjalizacja odbywa się przez konfigurację rejestrów w sposób opisany w odpowiednich tabelach. W tym celu wykorzystywana jest biblioteka magistrali I²C. Rejestry, które nie są tu wymienione, nie są nadpisywane przez program i mają wartości domyślne równe 0.

Rejestr nr 107 MGMT_1 - Power Management 1

Rejestr służący do konfiguracji trybu zasilania oraz źródła zegara. Posiada on również możliwość resetowania całego urządzenia oraz wyłączenia czujnika temperatury[6]. Jest to pierwszy i najważniejszy rejestr, który musi zostać skonfigurowany, przed konfiguracją innych rejestrów. Domyślna wartość dla tego rejestru to 0x40, co oznacza, że po włączeniu zasilania urządzenie znajduje się w trybie uśpienia.

Rejestr nr 27 GYRO_CONFIG - Gyroscope Configuration

Rejestr kontrolujący aktywację procedury testu żyroskopu oraz jego rozdzielczość.

Rejestr nr 27 SMPLRT_DIV - Gyroscope Configuration

Rejestr, którego wartość decyduje o dzielниku częstotliwości próbkowania żyroskopów urządzenia MPU-6050. Częstość aktualizacji pomiarów z poszczególnych czujników urządzenia, kolejki FIFO oraz procesora ruchu (DMP) również bazują na wartości tego rejestru. Częstotliwość próbkowania generowana jest przez podzielenie częstotliwości wybranego w rejestrze MGMT_1 zegara przez wartość liczbową tego rejestru.

$$f_s = \frac{f_g}{d} \quad (4.1)$$

, gdzie:

- f_s - uzyskana częstotliwość próbkowania;
- f_g - częstotliwość wyjścia żyroskopu, która wynosi 8kHz, gdy filtr dolnoprzepustowy (DLPF) jest wyłączony (DLPF_CFG-0 lub 7) lub 1kHz gdy DLPF jest włączony;
- d - podzielnik częstotliwości, tj. wartość tego rejestru;

Częstotliwość próbkowania czujnika przyspieszenia to 1kHz. Oznacza to, że w przypadku częstotliwości próbkowania większej od 1kHz, ta sama próbka wyjścia czujnika może wystąpić więcej niż raz w kolejce FIFO, DMP lub rejestrach czujnika. Wartość tego rejestru ustawiana podczas inicjalizacji to 0x08, co skutkuje próbkowaniem z częstotliwością 1kHz.

Rejestr nr 106 USER_CTRL - User Control

Rejestr ten pozwala na włączenie, wyłączenie lub zresetowanie kolejki FIFO, trybu I²C master oraz głównego interfejsu I²C. Kolejka FIFO, moduł I²C master.

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
DEVICE_RESET	7	Po ustawieniu tego bitu wszystkie rejestyry urządzenia zostają ustawione na swoje domyślne wartości. Bit ten zostaje automatycznie wyzerowany po zakończeniu procedury resetu.	0
SLEEP	6	Ustawienie tego bitu przełącza czujnik w tryb uśpienia.	0
CYCLE	5	Ustawienie tego bitu blokuje tryb SLEEP i przestawia urządzenie w tryb cyklicznego przechodzenia pomiędzy stanami uśpienia i czuwania, gdzie dane z sensorów odczytywane są z częstotliwością określoną przez LP_WAKE_CTRL (rejestr 108).	1
-	4	Bit zarezerwowany.	-
TEMP_DIS	3	Ustawienie tego bitu wyłącza czujnik temperatury.	1
CLKSEL[2:0]	0-2	3-bitowa wartość bez znaku. Decyduje o źródle zegara: 0 - wewnętrzny oscylator o taktowaniu 8MHz; 1 - PLL z osią X żyroskopu jako użytką jako odniesienie; 2 - PLL z osią Y żyroskopu jako użytką jako odniesienie; 3 - PLL z osią Z żyroskopu jako użytką jako odniesienie; 4 - PLL z zewnętrznym odniesieniem 32,768 kHz; 5 - PLL z zewnętrznym odniesieniem 19,2 MHz; 6 - zarezerowane; 7 - zatrzymanie zegara, generator w stanie reset.	2

Tablica 4.1: Rejestr nr 107 MGMT_1 - Power Management 1

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
XG_ST	7	Ustawienie tego bitu rozpoczyna procedurę testu osi żyroskopu w osi X.	0
YG_ST	6	Ustawienie tego bitu rozpoczyna procedurę testu osi żyroskopu osi Y.	0
ZG_ST	5	Ustawienie tego bitu rozpoczyna procedurę testu osi żyroskopu w osi Z.	0
FS_SEL[1:0]	3-4	Wybór pełnego zakresu wyjścia żyroskopu 0 - $\pm 250 \text{ } ^\circ/\text{s}$; 1 - $\pm 500 \text{ } ^\circ/\text{s}$; 2 - $\pm 1000 \text{ } ^\circ/\text{s}$; 3 - $\pm 2000 \text{ } ^\circ/\text{s}$.	1
-	2	Bit zarezerwowany.	-
-	1	Bit zarezerwowany.	-
-	0	Bit zarezerwowany.	-

Tablica 4.2: Rejestr nr 27 GYRO_CONFIG

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
-	7	Bit zarezerwowany.	-
FIFO_EN	6	Ustawienie tego bitu aktywuje kolejkę FIFO. Wyzerowanie go wyłącza ją. Żadne dane nie mogą zostać wpisane do kolejki gdy jest ona wyłączona. Stan kolejki pozostaje niezmieniony do resetu zasilania.	1
I2C_MST_EN	5	Ustawienie tego bitu aktywuje tryb I ² C Master. Gdy jest on wyzerowany wartości logiczne na pomocniczych liniach I ² C (AUX_DA oraz AUX_CL) odpowiadają wartościami na głównej linii I ² C (ADA oraz SCL).	0
I2C_IF_DIS	4	Bit ten, gdy ustawiony, blokuje główny interfejs I ² C i aktywuje interfejs SPI. Dla MPU-6050, wyposażonego tylko w interfejs I ² C powinien on być zawsze wyzerowany.	0
-	3	Bit zarezerwowany.	-
FIFO_RESET	2	Ustawienie tego bitu resetuje kolejkę FIFO. Po zakończonej operacji bit ponownie się zeruje.	1
I2C_MST_RESET	1	Ustawienie tego bitu resetuje I ² C Master. Po zakończonej operacji bit ponownie się zeruje.	0
SIG_COND_RESET	0	Ustawienie tego bitu resetuje ścieżki sygnałów dla wszystkich czujników (żydroskopów, czujników przyspieszenia i czujnika temperatury). Wyzerowane zostają również rejestrzy czujników. Po zakończonej operacji bit ponownie wraca do wartości 0.	0

Tablica 4.3: Rejestr nr 106 USER_CTRL - User Control

Rejestr nr 35 FIFO_EN - FIFO Enable

Rejestr FIFO_EN służy do wyboru odczytów czujników, które wpisywane będą do kolejki FIFO. Dane wpisywane są do kolejki z częstotliwością konfigurowaną przy pomocy rejestrów MGMT_1 i SMPLRT_DIV.

Rejestr nr 56 INT_ENABLE - Interrupt Enable

Rejestr ten służy do wyboru zdarzeń wywołujących sygnał przerwania na pinie INT.

4.1.2. Sprawdzenie poziomu zapełnienia kolejki FIFO

Sprawdzenie wypełnienia kolejki FIFO urządzenia MPU-6050 odbywa się przez:

- odczyt rejestrów MPU6050_RA_FIFO_COUNTL;
- odczyt rejestrów MPU6050_RA_FIFO_COUNTH;
- złożenie odczytanych bajtów do postaci zmiennej typu unsigned integer i zwrócenie wartości przez funkcję.

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
TEMP FIFO_EN	7	Ustawienie tego bitu aktywuje zapis rejestrów TEMP_OUT_H i TEMP_OUT_L (rejestry 65 i 66) do kolejki FIFO.	0
XG FIFO_EN	6	Ustawienie tego bitu aktywuje zapis rejestrów GYRO_XOUT_H i GYRO_XOUT_L (Rejestry 67 i 68) do kolejki FIFO.	1
YG FIFO_EN	5	Ustawienie tego bitu aktywuje zapis rejestrów GYRO_YOUT_H i GYRO_YOUT_L (Rejestry 69 i 70) do kolejki FIFO.	1
ZG FIFO_EN	4	Ustawienie tego bitu aktywuje zapis rejestrów GYRO_ZOUT_H i GYRO_ZOUT_L (Rejestry 71 i 72) do kolejki FIFO.	1
ACCEL FIFO_EN	3	Ustawienie tego bitu aktywuje zapis rejestrów ACCEL_XOUT_H, ACCEL_XOUT_L, ACCEL_YOUT_H, ACCEL_YOUT_L, ACCEL_ZOUT_H, ACCEL_ZOUT_L, (rejestry 59 do 64) do kolejki FIFO.	1
SLV2 FIFO_EN	2	Ustawienie tego bitu aktywuje zapis rejestrów EXT_SENS_DATA (Rejestry 73 do 96) związanych z Urządzeniem podrzędnym Slave 2 do kolejki FIFO.	1
SLV1 FIFO_EN	1	Ustawienie tego bitu aktywuje zapis rejestrów EXT_SENS_DATA (Rejestry 73 do 96) związanych z Urządzeniem podrzędnym Slave 1 do kolejki FIFO.	0
SLV0 FIFO_EN	0	Ustawienie tego bitu aktywuje zapis rejestrów EXT_SENS_DATA (Rejestry 73 do 96) związanych z Urządzeniem podrzędnym Slave 0 do kolejki FIFO.	0

Tablica 4.4: Rejestr nr 35 FIFO_EN - FIFO Enable

Nazwa bitu	Numer bitu	Funkcja	Wybrana wartość
-	7	Bit zarezerwowany.	-
-	6	Bit zarezerwowany.	-
-	5	Bit zarezerwowany.	-
ZG FIFO_EN	4	Ustawienie tego bitu aktywuje generowanie przerwania w przypadku przepelnienia kolejki FIFO.	1
ACCEL FIFO_EN	3	Ustawienie tego bitu aktywuje generowanie przerwania w przypadku wystąpienia przerwania od modułu I ² C Master.	0
-	2	Bit zarezerwowany.	-
-	1	Bit zarezerwowany.	-
SLV0 FIFO_EN	0	Ustawienie tego bitu aktywuje generowanie przerwania podczas pojawienia się nowych danych we wszystkich rejestrach czujników.	1

Tablica 4.5: Rejestr nr 56 INT_ENABLE - Interrupt Enable

4.1.3. Procedura obsługi przerwań

Zgodnie ze specyfikacją opracowanej dla potrzeb tej pracy biblioteki, właściwa część przesyłu bajtów po magistrali I²C odbywa się podczas obsługi przerwań. Wywołanie funkcji inicjującej odczyt lub zapis rejestrów urządzenia w procedurze obsługi innego przerwania uniemożliwiłoby jej poprawne wykonanie. Z tego względu, w obsłudze przerwania pochodzącego od pinu INT urządzenia MPU-6050 ustawiana jest jedynie flaga informująca o tym, że takie przerwanie wystąpiło. Nie są wykonywane żadne inne czynności. Właściwa obsługa przerwań pochodzących z tego pinu powinna odbywać się w pętli głównej programu, gdzie umieszczona zostaje funkcja obsługi przerwania.

Funkcja ta jako argument pobiera wskaźnik na tablicę bajtów, do której powinny zostać zapisane dane pobrane z kolejki FIFO. Zwraca bajt o wartości 0 - w przypadku gdy przerwanie nie wystąpiło lub 1 - gdy wystąpiło i zostało obsłużone. Działanie tej funkcji jest następujące:

- Na czas wywołania zostają wyłączone przerwania od pinu P2.5 mikrokontrolera - do tego pinu podpięty jest pin INT czujnika.
- Z urządzenia MPU-6050 odczytany zostaje rejestr INT_STATUS oraz sprawdzona zostaje ilość bajtów znajdujących się w kolejce. Dane te zostają zapisane do zmiennych lokalnych.
- Sprawdzona jest flaga informująca o tym, czy wystąpiło przerwanie od MPU-6050 (bit MPU_PENDING_INT w bajcie MPU_SW_STATE). Jeśli przerwanie nie wystąpiło funkcja kończy działanie zwracając wartość 0.
- Jeśli przerwanie wystąpiło, sprawdzane jest źródło przerwania przez odczytanie zapisanej wcześniej wartości rejestru INT_STATUS urządzenia MPU-6050.
- W przypadku, gdy w rejestrze INT_STATUS ustawiony jest bit FIFO_OFLOW_INT, który oznacza przepełnienie kolejki FIFO, kolejka ta jest czyszczona przez ustawienie bitu FIFO_RESET w rejestrze USER_CTRL. Ponadto, w bajcie MPU_SW_STATE zostaje ustawiony bit MPU_OFLOW_OCCURED sygnalizujący, że wystąpiło przepełnienie oraz opróżnienie kolejki FIFO. Następnie funkcja kończy działanie zwracając wartość 1
- Jeśli bit FIFO_OFLOW_INT ma wartość 0, sprawdzany jest bit DATA_RDY_INT rejestru INT_STATUS. Bit ten informuje o dostępności danych w kolejce FIFO urządzenia MPU-6050.
- Jeśli DATA_RDY_INT ma wartość 0 funkcja kończy działanie zwracając 0
- Jeśli DATA_RDY_INT ma wartość 1 sprawdzana jest zapisana wcześniej zmienna przechowująca informacje o ilości bajtów znajdujących się w kolejce
- Jeśli wartość ta przekracza 800, program dokonuje natychmiastowego odczytu danych o długości 10 ramek (120 bajtów) z kolejki FIFO. Nadmiarowe pomiary zapisywane są w tablicy specjalnie przeznaczonej do tego celu. Zapobiega to przepełnianiu się kolejki oraz chroni przed utratą tych danych. W przeciwnieństwie do normalnego odczytu, funkcja nie zwraca kontroli do nadzędnej części programu, lecz biernie oczekuje na moment zakończenie transmisji.
- Jeśli wartość ta jest mniejsza od 800, inicjowany jest normalny odczyt danych z kolejki oraz przekazanie kontroli do programu nadzędnego ze zwróceniem wartości 1. Program nadzędny może sprawdzić czy odczyt został zakończony w sposób opisany w sekcji 3.3 na stronie 23.

4.2. Biblioteka obsługi modułu radiowego CC110L

4.2.1. Reset urządzenia

Zgodnie z dokumentacją[11], po podłączeniu zasilania system musi zostać poddany procedurze resetu. Może ona zostać wykonana na dwa sposoby: automatyczny lub ręczny. Niniejsza biblioteka wykorzystuje ręczną procedurę resetu. Odbywa się ona w następujący sposób:

- Podana zostaje następująca sekwencja stanów na pin Chip Select urządzenia: 1010, oczekując między każdą zmianą stanu 600 cykli mikroprocesora, czyli około 37,5us.
- W stanie niskim pinu Chip Select przesłana zostaje komenda SRES (wywołująca reset urządzenia).
- Pin Chip Select zostaje ponownie ustawiany w stan wysoki.

4.2.2. Inicjalizacja

Inicjalizacja urządzenia odbywa się przez ustawienie odpowiednich wartości rejestrów konfiguracyjnych przy pomocy funkcji zapisu rejestru opisanej w podsekcji 3.5.8. Ustawienia dotyczą między innymi:

- wejść/wyjść;
- obsługi pakietów;
- częstotliwości transmisji;
- modemu;
- maszyny stanowej radia.

Ustalenie odpowiednich wartości, które powinny zostać przesłane do rejestrów urządzenia może zostać dokonane przez:

- ręczną analizę dokumentacji z wykorzystaniem odpowiedniej wiedzy z zakresu transmisji radiowej;
- wykorzystanie gotowych ustawień udostępnionych przez inny podmiot;
- wygenerowanie ustawień przy pomocy programu SmartRF studio.

Dla tej biblioteki zostały wykorzystane nieznacznie zmodyfikowane ustawienia znajdujące się w dokumentacji produktu CC110L BoosterPack[4].

4.2.3. Wysłanie pakietu

Funkcja wysyłająca pakiet pobiera tablicę bajtów, które powinny zostać wysłane oraz zmienną informującą o długości tej tablicy. Ze względu na konfigurację automatycznej obsługi ramek, która została zastosowana dla tej biblioteki, pierwszy wysłany bajt powinien oznaczać długość danych przesyłanych w ramce (payload). Długość danych nie obejmuje samego bajtu długości. Powinna ona być zatem o 1 bajt mniejsza od długości tablicy przekazanej do funkcji. Działanie funkcji:

- Wyłączenie przerwań od pinu GDO0 urządzenia. Odbywa się to przez wyzerowanie szóstego bitu w rejestrze P2IE, to jest bitu odpowiadającego za aktywację przerwań na 6 pinie portu 2 mikrokontrolera, do którego podpięty jest GDO0. Procedura ta jest konieczna, ponieważ GDO0 sygnalizuje również odebranie nowej ramki przez urządzenie, co normalnie wywołuje przerwanie.
- Wysłanie danych z wejściowej tablicy bajtów do urządzenia radiowego przy pomocy funkcji zapisu grupy rejestrów znajdującej się w bibliotece SPI opisanej w podsekcji 3.5.9 na stronie 34. Jako adres pierwszego rejestru podany zostaje rejestr TXFIFO.
- Wysłanie sygnału komendy STX, co inicjuje transmisję danych drogą radiową.
- Oprogramowanie oczekuje, aż pin GDO0 osiągnie stan niski - oznacza to zakończenie wysyłania sekwencji synchronizacji.

- Oprogramowanie oczekuje aż pin GDO0 osiągnie stan wysoki - oznacza to zakończenie transmisji radiowej.
- Ze względu na wystąpienie zbocza na pinie podłączonym do GDO0 odpowiadająca mu flaga przerwania zostaje ustawiona. Program usuwa ją przez wyzerowanie 6 bitu w rejestrze P2IFG.
- Następuje ponowna aktywacja przerwań zablokowanych na początku wywołania oraz zakończenie działania funkcji.

4.2.4. Odbiór pakietu

Wywołanie funkcji odbioru pakietu odbywa się w przerwaniu związanym z pinem podłączonym do linii GDO0 modułu radiowego. Obsługa przerwania jest włączona zarówno w urządzeniu pośredniczącym jak i w urządzeniu pomiarowym, mimo, że urządzenie pomiarowe ma za zadanie jedynie wysyłanie danych. Powodem tego są przepisy prawne związane z transmisją na częstotliwości 868MHz na terenie Europy, które wymagają implementacji procedury LBT[16]. Odbiór danych wykonywany jest w procedurze obsługi przerwania, która zajmuje mikroprocesor do momentu odczytu wszystkich danych. Można zatem powiedzieć, że poza momentem, gdzie doszło już do rozpoczęcia transmisji, urządzenie znajduje się cały czas w trybie nasłuchiwanego, co spełnia wymagania procedury LBT.

Funkcja odbioru pakietu pobiera wskaźnik na tablicę przeznaczoną do zapisu odebranych danych oraz wskaźnik na zmienną informującą o jej długości. Zwraca zmienną typu char o wartości 1, gdy odbiór pakietu zakończył się powodzeniem lub 0, gdy odbiór zakończył się porażką. Przebieg funkcji:

- Przy pomocy funkcji odczytu rejestru stanu, odczytany zostaje rejestr RXBYTES, który zawiera informację o ilości bajtów znajdujących się w kolejce FIFO przechowującej odebrane dane w urządzeniu CC110L.
- Jeśli ich ilość jest równa 0, funkcja kończy działanie zwracając wartość 0, w przeciwnym wypadku wykonywane są poniższe operacje:
- Z kolejki znajdującej się pod adresem RXFIFO odczytany zostaje pierwszy bajt, który powinien nieść informację o długości danych w odebranej ramce.
- Jeśli długość tablicy przeznaczonej na dane jest wystarczająca aby je pomieścić, zostają one odczytane z rejestru RXFIFO urządzenia przez wywołanie funkcji odczytu grupy rejestrów. Następnie, odczytane zostają dwa kolejne bajty, które niosą informacje o poprawności sumy kontrolnej CRC. Gdy suma kontrolna jest poprawna, funkcja zwraca wartość 1, w przeciwnym wypadku 0.
- Jeśli długość bufora nie jest wystarczająca, wejściowy parametr długości zostaje zastąpiony rzeczywistą długością oraz następuje wyczyszczenie kolejki odbiorczej przy użyciu sygnału komendy SFRX. Następnie funkcja zwraca wartość 0.

5. Oprogramowanie zarządzające systemem pomiarowym

5.1. Języki programowania

5.1.1. Oprogramowanie mikrokontrolera

Mikrokontrolery zainstalowane w urządzeniach mogą być programowane z użyciem następujących języków programowania:

- Assembler;
- C;
- C++.

W niniejszej pracy mikrokontrolery znajdujące się w urządzeniu pomiarowym i w urządzeniu pośrednim zostały zaprogramowane z użyciem języka C.

5.1.2. Oprogramowanie komputera PC

Aplikacja służąca do gromadzenia pomiarów została napisana w języku skryptowym Python z wykorzystaniem następujących bibliotek:

- biblioteka graficzna PyQt4;
- biblioteka obsługi portu szeregowego PySerial.

5.2. Środowisko programistyczne Code Composer Studio

Podczas realizacji tego projektu, do zaprogramowania mikrokontrolerów MSP430 zostało wykorzystane środowisko programistyczne Code Composer Studio v5 (CCSv5) dostarczone przez firmę Texas Instruments.

5.2.1. Opis ogólny

CCSv5 jest rozwiązaniem bardzo wygodnym, ponieważ nie wymaga praktycznie żadnej konfiguracji. Wszystkie niezbędne czynności dokonywane są automatycznie, podczas instalacji. Do dyspozycji użytkownika są liczne narzędzia takie jak debugger, graficzna konfiguracja peryferiów urządzeń, oraz narzędzia Power Advice i Optimization Advice. Power Advice opierając się na informacjach o budowie i sposobie działania konkretnego mikrokontrolera wskazuje miejsca w kodzie i sugeruje zmiany, które mogłyby zwiększyć oszczędność energii. Optimization Advice w podobny sposób podpowiada jak potencjalnie przyspieszyć obliczenia wykonywane przez mikroprocesor.

5.2.2. Marketing

Code Composer Studio to środowisko płatne, można jednak skorzystać z wersji darmowej, która ograniczona jest do 16kB kodu. Istnieje również możliwość skorzystania z licencji testowej, która zapewnia pełną funkcjonalność pakietu przez 90 dni z możliwością przedłużenia o kolejne 90 dni. Umożliwia to realizację semestralnych projektów studenckich[3]. Firma Texas Instruments posiada również program wsparcia dla studentów[1]. Takie podejście nie tylko ułatwia studentom naukę ale jest również korzystne dla producenta sprzętu, ponieważ zwiększa popularność i znajomość produktu na rynku.

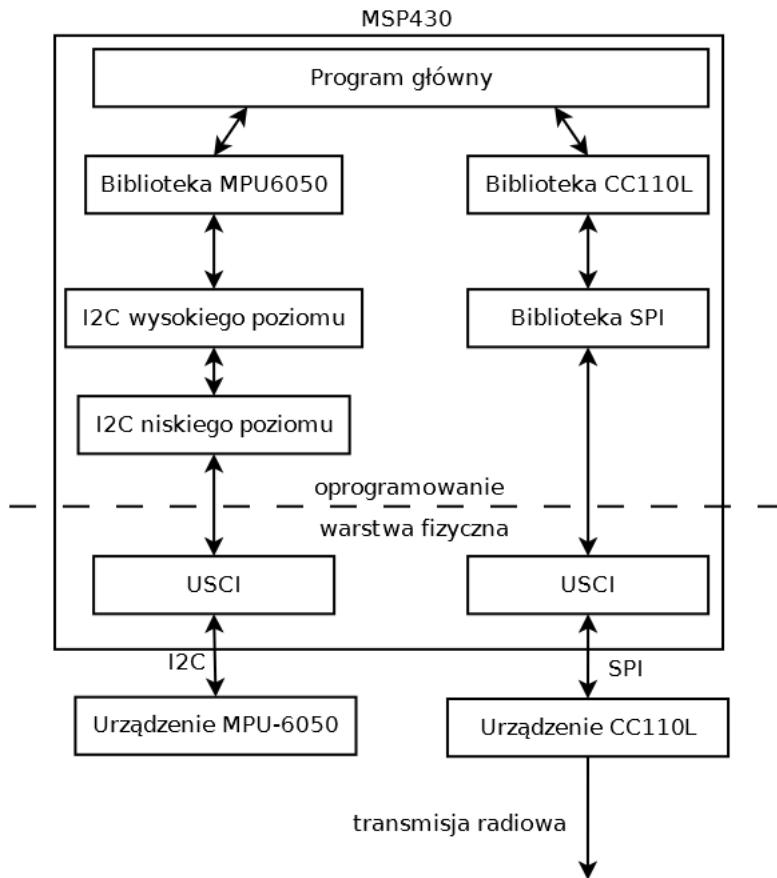
5.2.3. Wolne oprogramowanie

Pomimo otwartości na projekty studenckie i uczelniane firmy Texas Instruments, środowisko CCS nie wspiera w pełni wolnego oprogramowania. Koszt zakupu pełnej wersji środowiska CCS jest spory, co więcej przeznaczone jest ono głównie dla komercyjnych systemów operacyjnych. Istnieje co prawda wersja CCS dla systemu operacyjnego Linux, jednak nie posiada ona wszystkich funkcjonalności dostępnych w wersji pełnej i nie wspiera wszystkich dostępnych urządzeń. Przykładowo programator wykorzystany w tej pracy - Value Line Launchpad MSP-EXP430G2 nie jest obsługiwany przez tę wersję.

Istnieje jednak otwarty-źródłowy kompilator o nazwie MSPGCC stworzony przez społeczność użytkowników. Rozwój tego kompilatora jest od ponad roku wspierany przez samego producenta[2]. Ponieważ projekt jest otwarty-źródłowy, jest on zgodny ze wszystkimi systemami operacyjnymi. Warto jednak wspomnieć, nie ma pełnej zgodności interpretacji kodu pomiędzy MSPGCC i kompilatorem zawartym w CCS. Chcąc skompilować kod napisany dla jednego z tych kompilatorów przy pomocy drugiego konieczne jest dokonanie kilku zmian, między innymi zmiana składni funkcji obsługujących przerwania. Niektórzy programiści wykorzystują w tym celu gotowe fragmenty kodu zawierające odpowiednie makra.

5.3. Środowisko SmartRF Studio

Do poprawnej transmisji/odbioru danych przy pomocy modułu CC110L konieczna jest konfiguracja dużej ilości jego wewnętrznych rejestrów. Ręczna konfiguracja byłaby bardzo kłopotliwa. Przy takiej ilości rejestrów łatwo o pomyłkę. Ponadto, do przeprowadzenia takiej konfiguracji wymagana jest odpowiednia wiedza z zakresu komunikacji radiowej. Firma Texas Instruments dostarcza za darmo oprogramowanie pozwalające na automatyczne wygenerowanie ustawień modułu. Ponadto, oprogramowanie to pozwala na wygenerowanie kodu źródłowego służącego do ustawiania rejestrów. Generowanie kodu odbywa się przez stworzenie odpowiedniego szablonu, który jest następnie wypełniany wygenerowanymi przez program wartościami i nazwami wybranych rejestrów. Pozwala to na szybkie testowanie ustawień bez konieczności samodzielnej analizy bazującej na dokumentacji.



Rysunek 5.1: Warstwowy model urządzenia pomiarowego

5.3.1. Model warstwowy

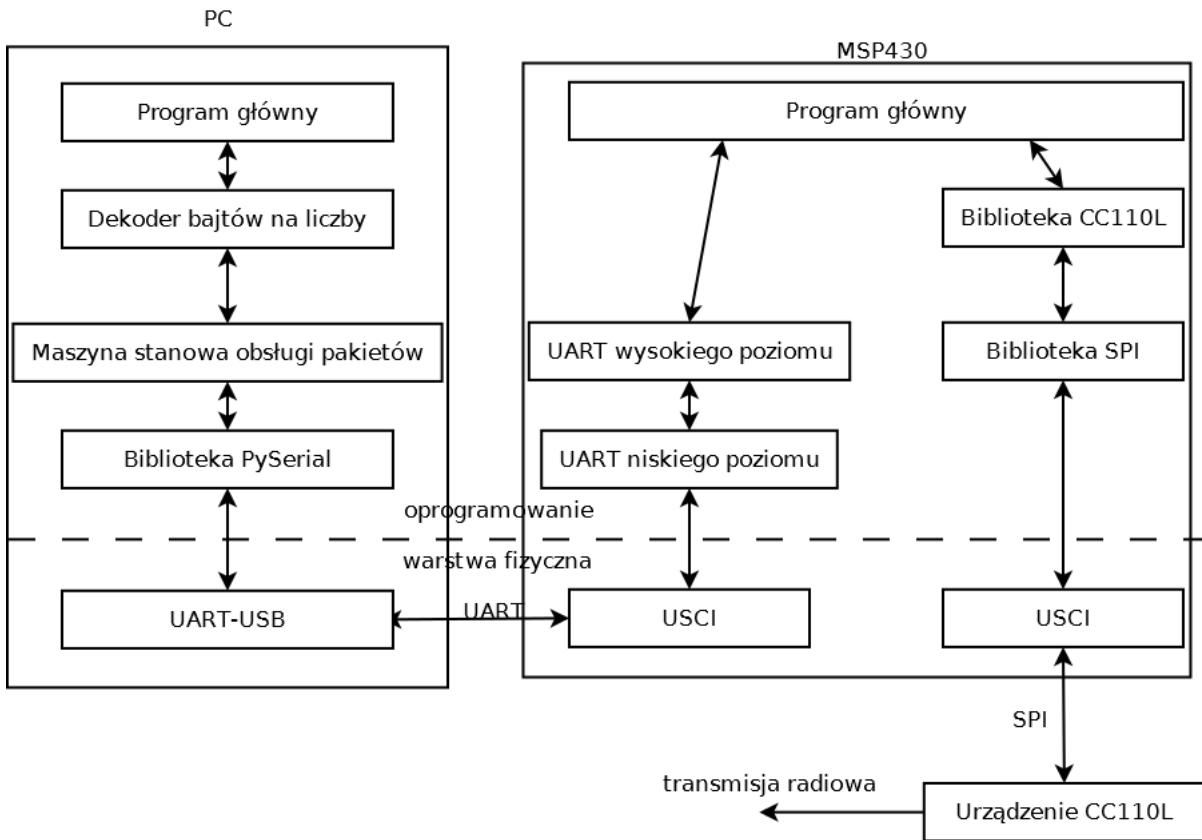
Rysunki 5.1 i 3.4 przedstawiają model warstwowy komunikacji poszczególnych elementów systemu bazując na opisanych w tej pracy bibliotekach.

Wykorzystanie modelu warstwowego umożliwia łatwą modyfikację lub nawet całkowitą podmianę poszczególnych warstw. Biblioteki przygotowane dla potrzeb tej pracy mogą łatwo zostać użyte w innych projektach.

Za komunikację z modułem MPU-6050 na poziomie fizycznym odpowiada sprzętowy moduł USCI mikrokontrolera. Modułem tym zarządza biblioteka protokołu I²C niskiego poziomu. Biblioteka wysokiego poziomu została dopasowana do specyfiki czujnika i pozwala na zapis/odczyt jego rejestrów, zarówno w sposób pojedynczy jak i grupowy. Korzysta z niej biblioteka urządzenia MPU-6050, dająca możliwość bezpośredniego wykorzystania jego funkcjonalności w programie głównym, bez wiedzy na temat specyfiki magistrali.

Komunikacja mikrokontrolera z urządzeniem CC110L na poziomie fizycznym jest obsługiwana przez moduł USCI. Moduł USCI zarządzany jest z wykorzystaniem biblioteki protokołu SPI. Korzysta z niej biblioteka urządzenia CC110L, która dzięki zapisowi, odczytowi rejestrów oraz wysyłaniu sygnałów komend umożliwia łatwą obsługę urządzenia w programie głównym.

Magistrala portu szeregowego również kontrolowana jest przez moduł USCI. Do sterowania nim wykorzystano prostą bibliotekę pozwalającą na jego inicjalizację oraz przesyłanie danych. Na jej podstawie wykonano bibliotekę wysokiego poziomu umożliwiającą efektywne przesyłanie danych w postaci binarnych ramek i pakietów. Zalety takiej organizacji transmisji to unikanie błędów oraz minimalizacja czasu przesyłu.



Rysunek 5.2: Warstwowy model urządzenia pośredniczącego.

5.4. Oprogramowanie urządzenia pośredniczącego

5.4.1. Ogólny opis działania

Moduł radiowy wysyła ramkę do urządzenia pośredniczącego, gdzie przekazywana jest ona z kolejnymi portem szeregowym komputera klasy PC. Dzięki uniwersalności portu szeregowego, dalsza akwizycja wartości pomiarowych może zostać wykonana przez dowolną aplikację.

5.4.2. Programowy protokół szeregowej transmisji danych

Pomiary przesyłane są z urządzenia pośredniczącego do komputera klasy PC przez port szeregowy z zastosowaniem protokołu umożliwiającego identyfikację znaczenia poszczególnych bajtów. Komputer składa z otrzymanych bajtów 16-bitowe liczby typu signed integer. Zastosowany protokół posiada również mechanizmy pozwalające zminimalizować długość pakietu, a zatem także czas jego przesyłania.

Pojedyncza ramka z zestawem pomiarów ma rozmiar 12 bajtów. Aby wprowadzić możliwość rozpoznawania początku i końca ramki rozpatrzone zostały następujące scenariusze:

- **Wykorzystanie zależności czasowych.**

Ponieważ zastosowana przepustowość portu szeregowego to 2Mbaud, transmisja jednej ramki zajmuje jedynie $48\mu\text{s}$, podczas gdy dla uzyskania częstotliwości próbkowania danych o wartości 1kHz, ramki powinny być przesyłane jedynie co 1ms. Przerwy między kolejnymi ramkami pozwalają na łatwą identyfikację końca i początku ramki bez stosowania dodatkowych protokołów. W takim przypadku można w programie obsługującym odbiór danych na komputerze PC ustawić minimalny czas, po którym stwierdzony zostaje początek nowej ramki. Rozwiążanie to w ocenie autora jest mało uniwersalne. Przy większym stopniu wypełnienia łącza (na przykład gdyby ustaliona została mniejsza szybkość transmisji portu szeregowego) mogą pojawić się nieprawidłowości.

ści w działaniu systemu. Ponadto, wymagałoby to wymuszania odpowiedniego odstępu między kolejnymi ramkami. Należałoby również pilnować, by nie otrzymać zbyt dużego odstępu między wewnętrznymi bajtami ramki.

- **Zastosowanie odpowiednio długiej preambuły przed ramką oraz dwóch bajtów stopu.**

Bajt stopu i preambuła muszą być tak dobrane, aby preambuła była unikalna, to znaczy na łączu nie może pojawić się identyczna kombinacja bajtów nie będąca preambułą. W zależności od implementacji maszyny stanów odbierającej dane, taka sytuacja mogłaby doprowadzić do niepoprawnego odczytu pomiarów, lub nawet do zawieszenia się programu. Dodając do protokołu preambułę przed każdą ramką oraz bajt stopu po ramce, narzut protokołu jest ponad dwa razy większy od samych danych. Według autora, taka sytuacja nie powinna mieć miejsca, nawet gdy dysponujemy dużą przepustowością łącza. Należy podkreślić, że zmniejszając ilość przesyłanych danych odciążone zostaje nie tylko samo łącze, ale również mikrokontroler. Mikrokontroler, który zajęty jest wysyłaniem ciągu bajtów przez port szeregowy może obsługiwać przerwanie informujące o nowych danych z modułu radiowego z pewnym opóźnieniem, co negatywnie wpływa na płynność pracy systemu.

- **Zastosowanie pakietu składającego się z preambuły oraz pewnej ilości ramek oddzielonych bajtami stopu.**

Takie rozwiązanie pozwala zaoszczędzić przepustowość łącza. Większa ilość ramek w pakiecie oznacza większą oszczędność łącza.[15]. Z drugiej strony, im mniej ramek zawiera się w pakiecie, tym większa częstość synchronizacji. Częstsza synchronizacja gwarantuje większą stabilność systemu. W przypadku błędu lub niepowodzenia transmisji, po określonym czasie poprawny odbiór danych zostanie przywrócony. W urządzeniu pośredniczącym, będącym tematem tej pracy zastosowano właśnie to rozwiązanie. Długość pakietu ustalono domyślnie na 16 ramek.

5.4.3. Maszyna stanów transmisji szeregowej

Maszyna stanów transmisji danych z urządzenia pośredniczącego do komputera została zaimplementowana w formie funkcji języka C z polem "unsigned static char cnt". Pole to pełni funkcję licznika, który po osiągnięciu określonej wartości (16) zeruje się. Argumentem tej funkcji jest ramka danych przeznaczonych do przesłania w formie tablicy zmiennych typu char. Funkcja wysyła tę ramkę przez port szeregowy. W przypadku gdy licznik ma wartość 0, dodaje też 13-bajtową preambułę składającą się z samych jedynek logicznych przed danymi. Po każdej ramce oraz po preambule dodawany jest też bajt stopu/przerwy składający się z samych zer. Na końcu wywołania licznik jest inkrementowany. Co istotne, do przesłania pakietu nie jest wymagane uprzednie złożenie go z odpowiedniej ilości ramek. Ramki wysyłane są na bieżąco, maszyna stanów jedynie dodaje preambułę oraz bajty stopu/przerwy w odpowiednich miejscach. Implementacja tej maszyny stanów została oznaczona w diagramie 5.2 jako „UART wysokiego poziomu”.

5.4.4. Główny program urządzenia pośredniczącego

Program przy starcie wykonuje inicjalizację poszczególnych modułów. Następnie przechodzi do pętli głównej gdzie obsługiwany jest przepływ danych.

Przebieg inicjalizacji modułów:

- ustawienie głównego zegara mikrokontrolera do wartości 16MHz, blokada modułu Watchdog;
- inicjalizacja tablicy pełniącej funkcję bufora danych o długości 12 bajtów;
- inicjalizacja modułu USCI_A0 dla UART;
- inicjalizacja modułu USCI_B0 dla SPI;

- reset oraz inicjalizacja urządzenia CC110L;

Przebieg głównej pętli programu:

- sprawdzenie, czy moduł radiowy odebrał nowe dane. Jeśli tak to wykonywane są następne kroki:
- wyczyszczenie flagi informującej o pojawieniu się nowych danych;
- skopiowanie nowych danych do bufora;
- wysłanie danych z bufora przez port szeregowy.

5.5. Oprogramowanie urządzenia pomiarowego

5.5.1. Ogólny opis działania

Po otrzymaniu zasilania, mikrokontroler zainstalowany w urządzeniu pomiarowym, korzystając z biblioteki protokołu I²C oraz biblioteki służącej do obsługi czujnika konfiguruje podstawowe, niezbędne do działania rejesty modułu MPU-6050. Zostają ustawione domyślne wartości dla rejestrów odpowiadających m.in. za tryb pracy, częstotliwość próbkowania oraz zasilanie.

Tak skonfigurowany czujnik dostarcza pomiary z żyroskopu i akcelerometru przez wpisywanie ich do swojej wewnętrznej kolejki FIFO oraz wysyłanie przerwania do mikrokontrolera. Przerwanie to informuje o obecności nowych danych w kolejce. Mikrokontroler, otrzymawszy przerwanie pobiera pomiary z kolejki. Z odebranych danych zostaje utworzona ramka, która następnie przekazywana jest do bufora modułu radiowego.

5.5.2. Główny program urządzenia pomiarowego

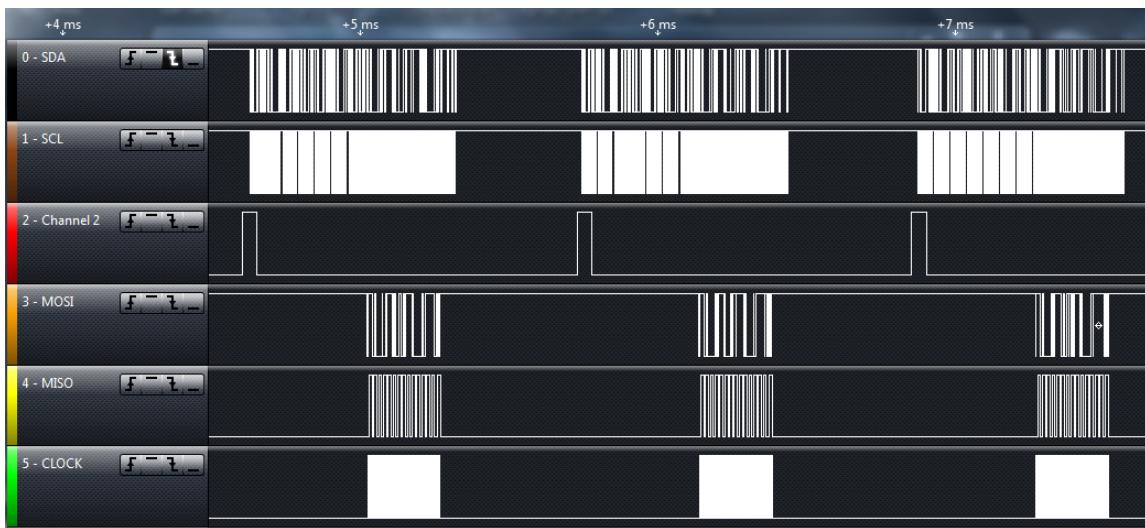
Program przy starcie wykonuje inicjalizacji poszczególnych modułów. Następnie przechodzi do pętli głównej gdzie obsługiwany jest przepływ danych.

Przebieg inicjalizacji modułów:

- Ustawienie głównego zegara mikrokontrolera do częstotliwości 16MHz, blokada modułu Watch-dog.
- Inicjalizacja dwóch tablic pełniących funkcje buforów danych.
Pierwsza z tablic ma pojemność 12 bajtów, druga 13 bajtów. Pierwszy bajt drugiej tablicy ma zawsze wartość 12. Jest to bajt oznaczający długość ramki, niezbędny przy transmisji radiowej.
- Inicjalizacja modułu USCI_A0 dla protokołu SPI.
- Reset oraz inicjalizacja urządzenia CC110L.
- Inicjalizacja modułu USCI_B0 dla protokołu I²C.
- Inicjalizacja urządzenia MPU-6050.

Przebieg głównej pętli programu:

- Skopiowanie 12 ostatnich bajtów bufora 1 do bufora 2.
- Oczekiwanie aż magistrala I²C będzie gotowa.
- Wykonanie funkcji obsługi przerwań urządzenia MPU.
- Na końcu, jeśli funkcja obsługi przerwań zgłosiła rozpoczęcie odbioru nowych danych, to dane znajdujące się w buforze 2 (poprzednia próbka) są wysyłane drogą radiową do urządzenia pośredniczącego.



Rysunek 5.3: Przebiegi sygnałów urządzenia pomiarowego.

Dzięki wykorzystaniu podwójnego buforowania oraz modułów USCI, możliwa jest jednocześnie transmisja i odbiór danych przez mikrokontroler. Przebiegi sygnałów na rysunku 5.3 pozyskane z analizatora stanów logicznych pokazują jednoczesną transmisję przy pomocy magistral SPI oraz I²C. Sygnał „Channel 2” to dodatkowo wyprowadzony sygnał informujący o tym, że mikroprocesor zajęty jest inicjowaniem transmisji po magistrali I²C.

5.6. Aplikacja akwizycji danych dla komputera PC

Oprogramowanie służące do zbierania danych z autonomicznego czujnika przyspieszenia zostało napisane w języku skryptowym Python. Posiada ona prosty interfejs graficzny składający się z pola, w którym pojawiają się informacje dla użytkownika w formie tekstu oraz przycisków służących do sterowania programem.

Chcąc rozpoczęć odczyt danych przesyłanych z urządzenia przez UART, użytkownik najpierw nawiązuje połączenie z samym portem szeregowym. Dokonuj się tego przez naciśnięcie przycisku „Connect”. W polu informacyjnym pojawia się informacja o udanym bądź nieudanym połączeniu. Rozpoczęcie/zakończenie zbierania danych kontrolowane są odpowiednio przyciskami Start/Stop. Przycisk Save zapisuje dane zebrane między ostatnimi użyciami przycisków Start i Stop do pliku o nazwie „out.txt” znajdującego się w głównym katalogu programu. Format pliku składa się z kolumn zawierających:

- 6 liczb reprezentujących pomiary z kolejnych czujników urządzenia pomiarowego;
- Dokładnej daty i czasu zapisanych w formacie UTC.

Kolejne wartości oddzielone są od siebie przy pomocy znaku tabulacji.

5.6.1. Maszyna stanów obsługująca odbiór danych

Po poprawnym połączaniu z portem szeregowym oraz naciśnięciu przycisku Start maszyna stanów rozpoczyna działanie i przechodzi do stanu nasłuchiwanego.

Opis stanów:

- Stan nasłuchiwanego.

W tym stanie aplikacja czeka na odebranie pełnej preambuły. Na bieżąco czytane są kolejne odbierane bajty. Jeśli zostanie odebrana ilość bajtów o wartości 0xFF przekraczająca długość ramki,

to stwierdzane jest odebranie początku ramki. Zostaje wtedy odczytany jeszcze jeden bajt, który interpretowany jest jako bajt stopu (jego wartość nie jest sprawdzana) a następnie maszyna przechodzi do stanu odbioru danych.

- Stan odbioru danych.

Podczas przebywania w tym stanie zbierane jest kolejno n ramek, gdzie n oznacza długość pakietu. W systemie stworzonym na potrzeby tej pracy, wartość ta ustawiona została domyślnie na 16 ramek. Po zebraniu tej ilości ramek maszyna wraca do stanu nasłuchiwanego.

Procedura odbioru każdej ramki składa się z:

- Odczytu kolejnych 12 bajtów (długość ramki), zapisania odczytanych danych do pamięci programu.
 - Odczytu pojedynczego bajtu, o którym program zakłada, że jest bajtem stopu (nie jest sprawdzana jego wartość).
- Po odczytaniu n ramek maszyna przechodzi ponownie w stan nasłuchiwanego.

6. Eksperiment porównawczy

6.1. Opis stanowiska

Stanowisko, na którym przeprowadzono eksperiment składało się z:

- układu wahadła odwróconego;
- komputera klasy PC służącego do sterowania układem wahadła;
- urządzenia pomiarowego i pośredniczącego, które są tematem tej pracy;
- komputera klasy PC służącego do zbierania danych z autonomicznego czujnika przyspieszenia;
- pary baterii AAA służących do zasilenia urządzenia pomiarowego.

Fizyczna konstrukcja obejmuje ramę z napędzanym wózkiem, do którego swobodnie przymocowane jest wahadło fizyczne o jednym stopniu swobody. Urządzenie posiada enkodery pozwalające określić przesunięcie wózka oraz kąt odchylenia wahadła.

Komputer wyposażony jest w kartę pozwalającą na zbieranie odczytów z enkoderów oraz sterowanie silnikami napędzającymi wózek. Oprogramowanie zarządzające układem napisane jest w środowisku Matlab/Simulink.

Przeliczenie fal prostokątnych będących wyjściami z enkoderów na kąt wychylenia wahadła oraz kąt obrotu silnika sterującego wózkiem odbywa się na karcie pomiarowej. Również pozycja wózka obliczana jest na karcie, bazując na kącie obrotu silnika. Wobec tego, dane wejściowe spływające do komputera to kąt wychylenia wahadła oraz pozycja wózka. Jako, że zainstalowane w układzie enkodery są inkrementalne, to wartości przez nie dostarczane są względne.

Kalibracji odczytów można dokonać przy pomocy oprogramowania, ustawiając wózek i wahadło w pozycji, w której wejścia powinny mieć wartości zerowe. Następnie należy ustawić przełącznik „Reset Encoders” w pozycji „Reset” (Patrz rys. 6.1).

Znajdujący się na schemacie blok Sensors wylicza przy pomocy operacji różniczkowania prędkość kątową wahadła oraz prędkość liniową wózka.

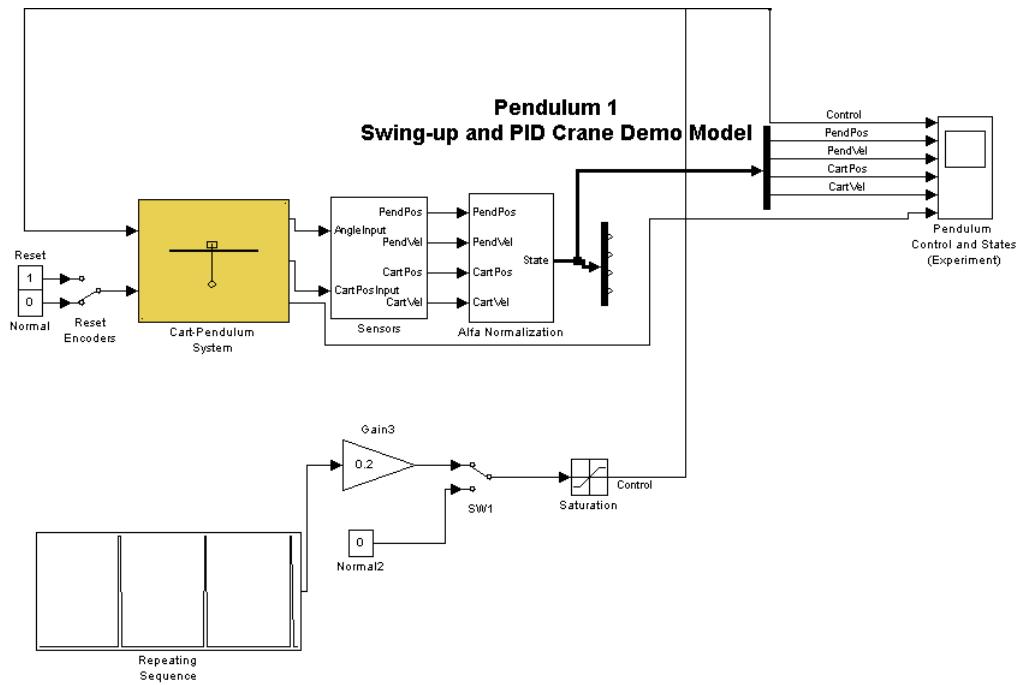
Zdjęcia z instalacji urządzenia znajdują się w dodatku A.

6.2. Porównanie jakości pomiarów prędkości kątowej

6.2.1. Konfiguracja sprzętu

Urządzenie pomiarowe przyklejono taśmą na osi wahadła fizycznego wyposażonego w enkoder. W ten sposób dokonano weryfikacji jakości pomiarów systemu będącego tematem tej pracy. Urządzenie zainstalowano blisko osi wahadła. Baterie zostały umieszczone powyżej czujnika. Celem uniemożliwienia ruchu wózka przywiązanego do konstrukcji urządzenia; dzięki temu możliwy był wyłącznie ruch obrotowy samego wahadła.

Należy wyjaśnić, że mocowanie urządzenia za pomocą taśmy było podeykowane ograniczonymi zasobami laboratoryjnymi, a umieszczenie bezprzewodowego urządzenia pomiarowego na osi wahadła służyło zmniejszeniu wpływu jego masy na fizykę wahadła.



Rysunek 6.1: Program zarządzający układem wahadła

6.2.2. Konfiguracja oprogramowania.

Oprogramowanie urządzenia pomiarowego oraz urządzenia pośredniczącego zostało skonfigurowane zgodnie z opisem zawartym w rozdziale 5 na stronie 45. Wyjątek stanowi długość pakietu dla portu szeregowego, która została zmniejszona do jednej ramki celem zapewnienia stabilniejszej synchronizacji. Najważniejsze ustawienia to:

- częstotliwość pomiarów czujników: 1kHz;
- przepustowość transmisji radiowej: 500kbaud;
- przepustowość portu szeregowego: 2Mbaud;
- rozmiar pakietu transmitowanego przez port szeregowy: 1 ramka;
- wyłączony cyfrowy filtr dolnoprzepustowy (DLPF) czujnika.

Dane z autonomicznego czujnika zebrano przy pomocy oprogramowania opisanego w rozdziale 5. Dane z enkoderów zebrano korzystając z odpowiednio zmodyfikowanego oprogramowania zarządzającego wahadłem. W oprogramowaniu wyłączono wszelkie sterowania podawane do urządzenia. Czas symulacji typu „real time” ustalono na 120 sekund. Długość kroku skonfigurowano do stałej wartości 0.001 sekundy.

6.2.3. Przeprowadzenie eksperymentu

Procedura przeprowadzenia pomiarów była następująca:

- odchylenie wahadła o niewielki kąt w prawo;
- aktywacja zbierania pomiarów z bezprzewodowego czujnika;
- rozpoczęcie symulacji w programie Simulink;
- wypuszczenie wahadła, obserwacja oscylacji periodycznych tłumionych;

- oczekiwanie na wygaśnięcie oscylacji;
- wyłączenie oprogramowania zbierającego dane.

6.2.4. Opracowanie danych

Dane z autonomicznego czujnika zapisane zostały w formie plików składających się z wierszy zawierających kolejne próbki. Każdy wiersz składa się z sześciu liczb oddzielonych znakami tabulacji. Liczby te reprezentują pomiary z czujników urządzenia w następującej kolejności:

- czujnik przyspieszenia dla osi X;
- czujnik przyspieszenia dla osi Y;
- czujnik przyspieszenia dla osi Z;
- żyroskop dla osi X;
- żyroskop dla osi Y;
- żyroskop dla osi Z.

Ostatnim elementem każdego wiersza jest dokładna data i godzina zapisane w standardowym formacie UTC. Czas zapisano z dokładnością do 0,001s.

Wartości, które poddano analizie podczas tego eksperymentu to pomiary żyroskopu pracującego w osi Z oraz wartości przyspieszenia kątowego obliczane w bloku Signals oprogramowania wahadła.

6.2.5. Zmiana formatu zapisu czasu

Aby znormalizować sposób zapisu czasu przeprowadzono następującą procedurę:

- Zignorowano wszystkie daty.
- Czas rejestracji każdej próbki przeliczono na sekundy (otrzymano więc ilość sekund od godziny 0:00).
- Od obliczonych wartości odjęto wartość pierwszej próbki, otrzymując tym samym czas wyrażony w sekundach i liczony od rozpoczęcia pomiarów.

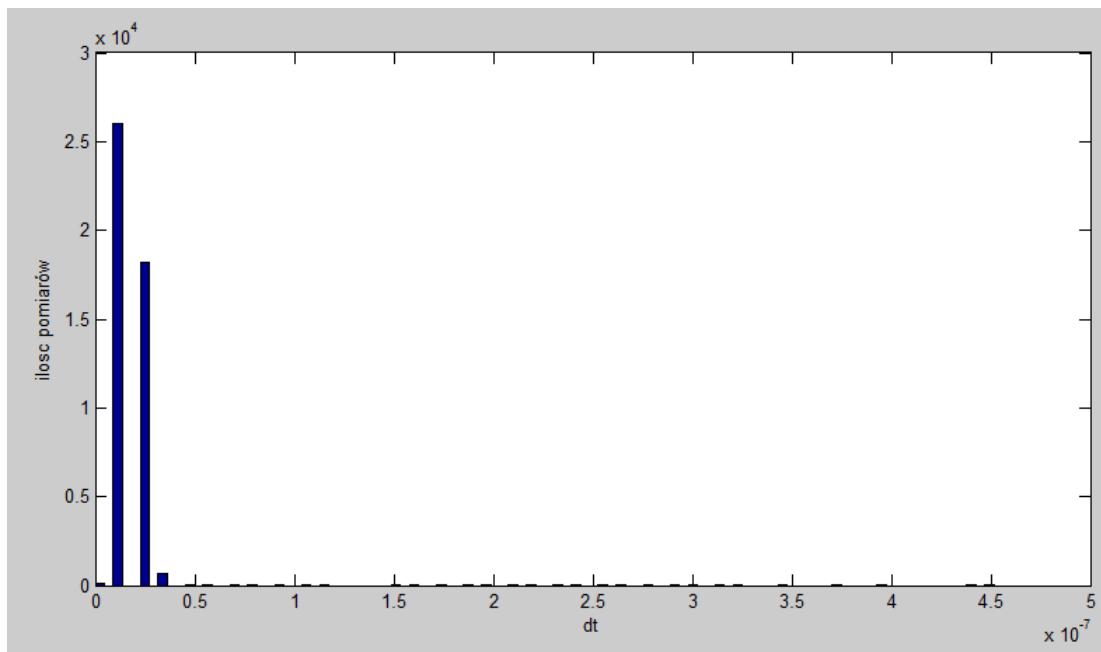
6.2.6. Interpolacja

Zebrane próbki okazały się występować ze zmienną częstotliwością, mniejszą od 1kHz. Rysunek 6.2 przedstawia histogram chwilowych okresów próbkowania danych otrzymywanych przez komputer.

Dokonano interpolacji danych zgromadzonych z czujnika, aby możliwa była efektywna praca na nich w programie Matlab. Uzyskano dzięki temu dane o stałym okresie próbkowania. Wektor czasu, na który dokonano interpolacji ustalono w następujący sposób:

- Początek wektora ma wartość 0.
- Każdy kolejny element jest większy od poprzedniego o 0,001 [sekundy].
- Ostatni element ma wartość ostatniego elementu oryginalnego wektora czasu.

Interpolacji dokonano wykorzystując wbudowaną funkcję pakietu Matlab "interp1". Rodzaj interpolacji to wielomiany sklejane pierwszego stopnia, zatem wykres po interpolacji praktycznie nie różni się wizualnie od wykresu z oryginalnym próbkowaniem.



Rysunek 6.2: Histogram chwilowych okresów próbkowania danych z czujnika.

6.2.7. Filtracja

Wykres 6.3 przedstawia surowe dane, które zostały odebrane z bezprzewodowego czujnika dla żyroskopu pracującego w osi Z. Wykres 6.4 przedstawia dane poddane filtracji dolnoprzepustowej.

Filtracja została wykonana w dwóch etapach:

- usunięcie wartości znacznie odbiegających od przeciętnych wartości pomiarów;
- zastosowanie cyfrowego filtra dolnoprzepustowego przy pomocy funkcji „butter” (komenda: butter(2, 0.005)) oraz „filter” pakietu Matlab. .

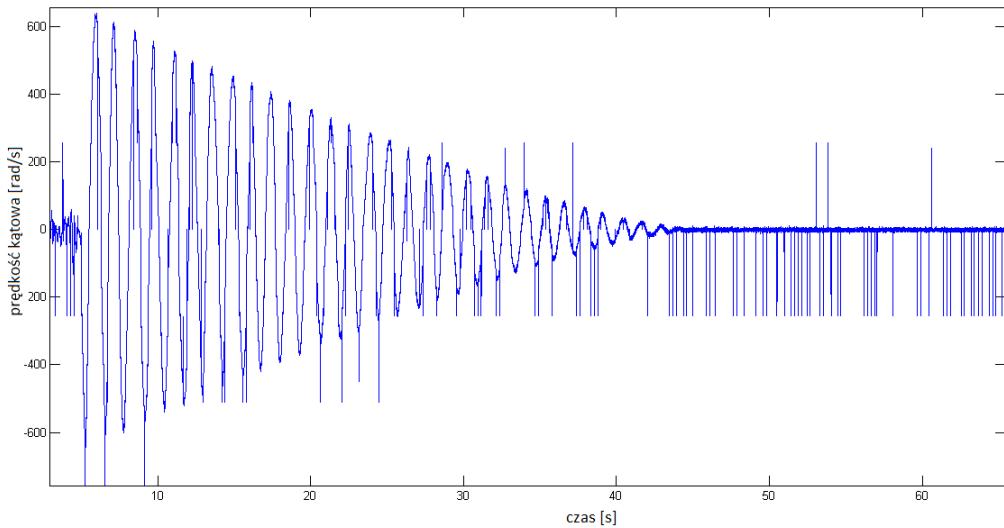
Na wykresie surowych danych widoczne są dwa rodzaje zakłóceń:

- zakłócenia/błedy transmisji cyfrowej;
- typowy dla czujników szum biały.

Użycie takiej filtracji spowodowało znaczne wygładzenie wykresu, zgodnie z oczekiwaniami. Niestety wyraźnie widać też negatywne efekty filtra - przesunięcie w fazie oraz tłumienie. Lepsze efekty można było uzyskać wykorzystując inne metody filtracji, jednak sam moduł MPU-6050 posiada możliwośćłączenia wewnętrznej filtracji dolnoprzepustowej. Dzięki temu można ocenić przydatność tej funkcji.

Błędy transmisji cyfrowej są wyraźnie widoczne w postaci pojawiających się na wykresie „szpilek”. Obserwowane szpilki mają zazwyczaj wartości okrągłe w systemie binarnym, np. -256, 255, a ich długość to jedna próbka. Przeprowadzono analizę, która pozwoliła zidentyfikować źródło tych zakłóceń: jest nim transmisja przez port szeregowy.

Zdecydowana większość szpilek pojawiających się na wykresie ma wartość -256. Wartości przesyłane są przez UART w formacie uzupełnień do dwóch. Binarna wartość starszego bajtu dla liczb ujemnych większych od -256 to 11111111. Liczba -256 zapisana binarnie ma z reprezentacją binarną 11111111 00000000. Stąd wniosek, że drugi bajt liczby odpowiadającej odczytom z żyroskopu działającego na osi Z bywał czasami transmitowany niepoprawnie i odbiorca go odrzucał. Następny w kolejności występuje zawsze bajt stopu, który był interpretowany przez oprogramowanie jako młodszy bajt pomiaru. Stąd pojawiająca się często wartość -256.



Rysunek 6.3: Surowe dane z żyroskopu

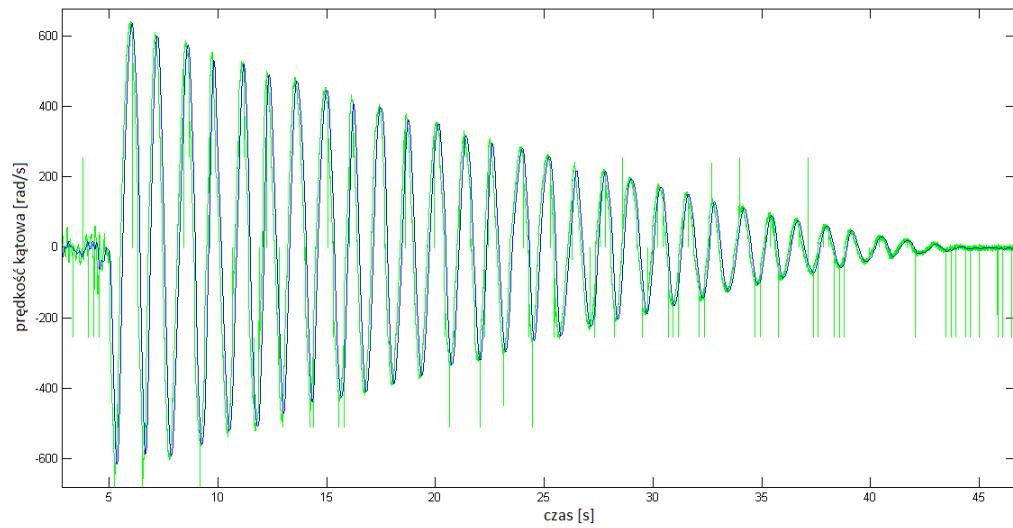
Należy zauważyć, że uzyskana zmienna częstotliwość próbkowania również miała swoją przyczynę w transmisji szeregowej. Tłumaczy to sposób działania maszyny stanów odbierającej dane z portu szeregowego, która przed rozpoczęciem zapisywania kolejnych bajtów zawierających wartości pomiarów, najpierw zlicza bajty preambuły. Jeśli któryś z nich nie zostanie otrzymany lub będzie miał wartość inną niż 0xFF, cała ramka zostaje pominięta. Z tego powodu część ramek nie była odbierana.

Problemy z transmisją z wykorzystaniem portu szeregowego mogły być spowodowane ustawieniem zbyt dużej prędkości transmisji (taktowanie zegara UART 2MHz). Problem ten można rozwiązać poprzez:

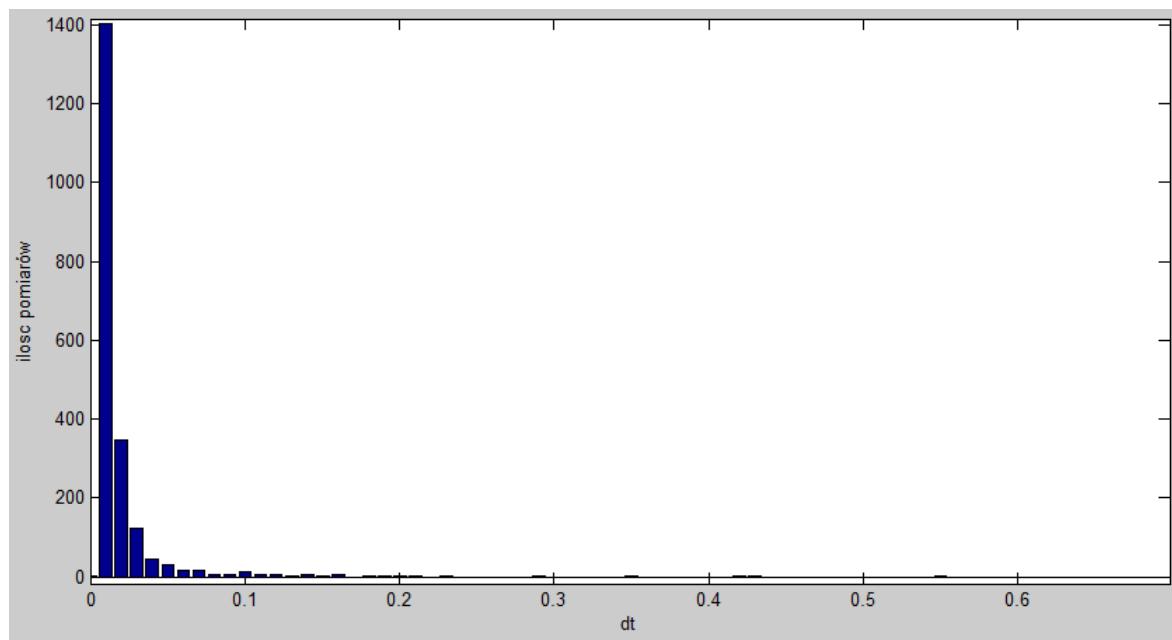
- zmniejszenie prędkości transmisji;
- wprowadzenie systemu weryfikacji otrzymania danych.

Wdrożenie systemu weryfikacji wymagałoby zastosowania odpowiedniego protokołu, który korzystając z dodatkowego bufora umożliwiłby retransmisję ramki w przypadku braku potwierdzenia jej otrzymania.

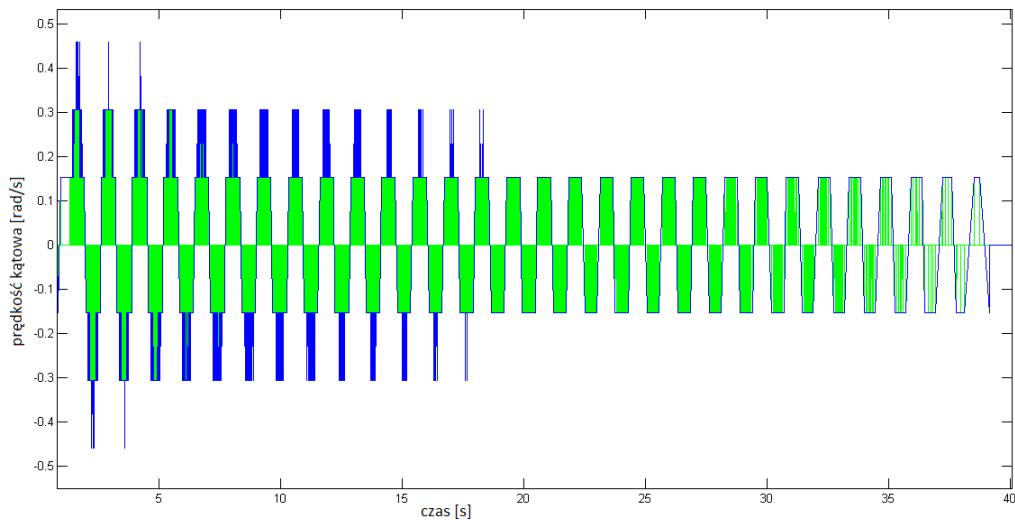
Wartości prędkości kątowej obliczone metodą ilorazów różnicowych w bloku „Sensors”, wymagały pewnej obróbki przed zaprezentowaniem ich na wykresie porównawczym. Długość kroku symulacji, która podczas eksperymentu wynosiła 0,001s okazała się być krótsza od okresu zmian wartości odczytywanych z enkoderów podczas obserwowanych drgań gasących. Skutkiem tego wykres prędkości kątowych jest postrzępiony. W miejscach, gdzie ze względu na ograniczoną rozdzielcość przez jakiś czas występuje stała wartość położenia enkoderów, mimo ruchu wahadła, wartość ilorazu różnicowego powraca do zera. Dla czytelniejszej analizy, usunięto próbki, których iloraz różnicowy miał zerową wartość przed wygaśnięciem oscylacji (rys. 6.6 na stronie 59 i 6.7 na stronie 59). Rysunek 6.5 przedstawia histogram, w którym oś pozioma to różnica czasu między zmianami odczytów z enkoderów.



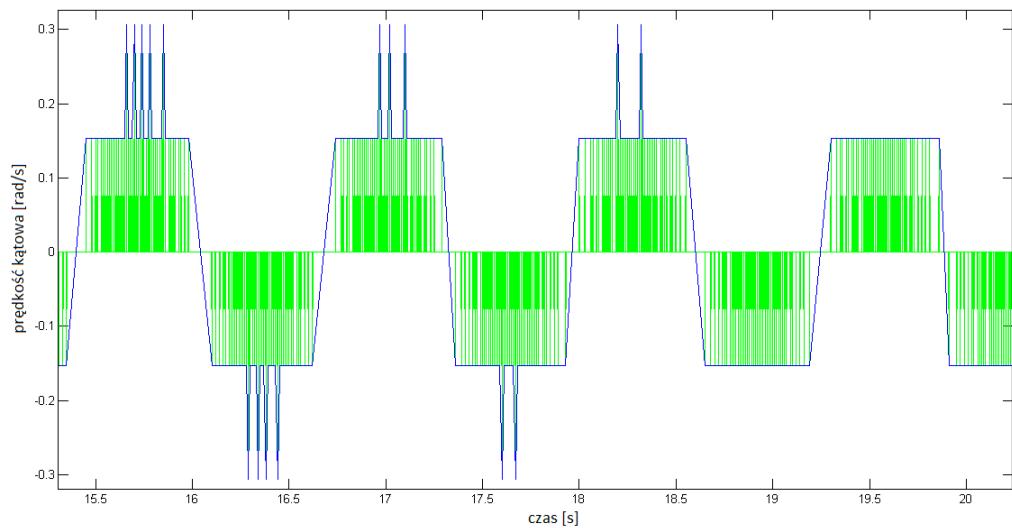
Rysunek 6.4: Zielony - surowe dane z żyroskopu. Niebieski - dane poddane filtracji dolnoprzepustowej.



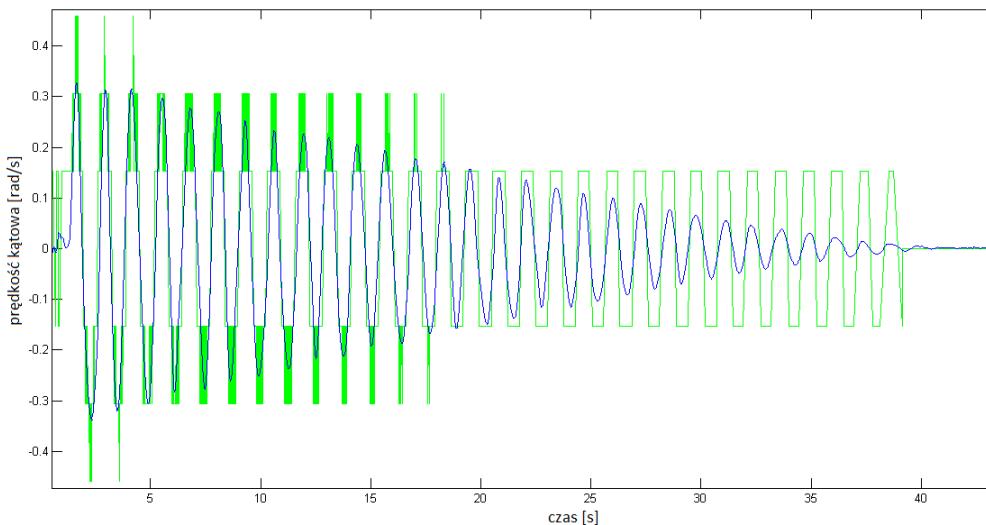
Rysunek 6.5: Histogram zmian odczytów enkoderów w czasie.



Rysunek 6.6: Zielony - surowe dane z enkoderów. Niebieski - przetworzone dane z enkoderów.



Rysunek 6.7: Zielony - surowe dane z enkoderów. Niebieski - przetworzone dane z enkoderów.



Rysunek 6.8: Zielony - prędkość kątowa obliczona z enkoderów. Niebieski - prędkość kątowa odczytana z żyroskopu.

6.2.8. Porównanie danych

Wykres 6.8 przedstawia porównanie próbek prędkości kątowej otrzymanych z bezprzewodowego czujnika po filtracji dolnoprzepustowej (wykres niebieski) z prędkością kątową obliczoną metodą ilorazów różnicowych na podstawie odczytów z enkoderów (wykres zielony). Podczas prób ilustracji tych danych na wykresie pojawił się problem braku synchronizacji czasowej dwóch serii pomiarów, ponieważ aktualny czas symulacji zwracany przez program Simulink jest względny i liczony jest od jej rozpoczęcia.

Rozpoczęcie zbierania pomiarów z obu czujników aktywowane zostało jeszcze przed puszczeniem obiektu w ruch. Dzięki temu na podstawie przebiegu prędkości kątowej można było łatwo znaleźć w obu przebiegach wspólny moment - początek ruchu harmonicznego wahadła. Dokonano tego przy pomocy skryptu wykrywającego w każdej serii próbki o wartości przekraczającej pewną ustaloną wartość progową. Następnie odpowiednio przesunięto w czasie jedną z serii.

Pomiary generowane przez symulację dostarczane są w jednostkach układu SI. Dane z urządzenia pomiarowego to 16-bitowe liczby typu signed integer, których pełny zakres jest skonfigurowany do pewnej, określonej wartości. Podczas wykonywania eksperymentu zakres ten ustalony był na $\pm 1000^{\circ}/\text{s}$. Dokonano przeskalowania jednostek do układu SI.

Łatwo można zaobserwować znaczną przewagę pomiarów pochodzących z żyroskopu. Pomiary z enkoderów są ograniczone ze względu na swoją rozdzielczość. Amplituda może zostać odczytana jedynie z dokładnością jednego kwantu. Same drgania mogą być obserwowane tylko do momentu, w którym ich amplituda jest na tyle duża, aby powodować zmianę ustawienia enkoderów.

Podjęto próbę otrzymania przesunięcia kątowego wahadła z informacji o prędkości kątowej uzyskanych z autonomicznego czujnika. Obliczono całkę z pomiarów prędkości kątowej metodą trapezów, jednak rezultat tej operacji nie był satysfakcyjny. Wynik obarczony był kumulującymi się błędami.

Ze względu na brak odpowiednich narzędzi, pozycja instalacji nie była dokładna, co niewątpliwie doprowadziło do zaburzenia pomiarów.

Sposób instalacji pozwalał na pewien, ograniczony ruch urządzenia względem wahadła. Jako iż żyroskop dokonuje pomiaru prędkości kątowej, samo przesunięcie urządzenia w stosunku do osi wahadła nie powinno wpływać na pomiary. Również rotacja czujnika względem osi wahadła jest bez znaczenia. Jednakże, obrót względem osi nie będących równoległymi do osi obrotu wahadła w znacznym stopniu zaburzył pomiary. Zaburzenie to dotyczy w głównej mierze kształtu przebiegu oscylacji i nie dało się

go uniknąć bez zastosowania odpowiedniego sposobu montażu. Należy zauważyć, że okres drgań został zachowany, a przekłamanie amplitudy jest prawdopodobnie minimalne.

6.3. Porównanie jakości pomiarów przyspieszenia liniowego

6.3.1. Konfiguracja sprzętu

Urządzenie pomiarowe zainstalowano bezpośrednio na wózku, na którym wisły wahadło. Wahadło ani wózek nie zostały unieruchomione.

6.3.2. Konfiguracja oprogramowania

Konfiguracja oprogramowania bezprzewodowego czujnika była identyczna z konfiguracją opisaną w sekcji 2.

Przełącznik SW1 w schemacie blokowym został ustawiony w górnej pozycji, aby umożliwić sterowanie wózkiem w pętli otwartej. Sterowanie odbywało się z wykorzystaniem bloku Repeated Sequence. Blok ten został zaprogramowany na wysyłanie serii krótkich, oddalonych od siebie impulsów. Wózek poddany takiemu sterowaniu okresowo przemieszczał się na niewielką odległość.

Konfiguracja bloku Repeating Sequence:

- time values: [0 1 1.01 1.03 1.05 1.1]*10;
- output values: [0 0 1 1 0 0].

6.3.3. Opracowanie danych

Dla tego eksperymentu istotnym parametrem była wartość przyspieszenia liniowego osi Y autonomicznego czujnika oraz pochodna prędkości liniowej wózka obliczana przez blok Signals.

6.3.4. Zmiana formatu zapisu czasu

Dokonano przetworzenia danych w sposób identyczny do sposobu wykorzystanego w eksperymencie opisany w sekcji 2.

6.3.5. Filtracja

Na wykresie surowych pomiarów z bezprzewodowego czujnika widoczne są trzy rodzaje zakłóceń:

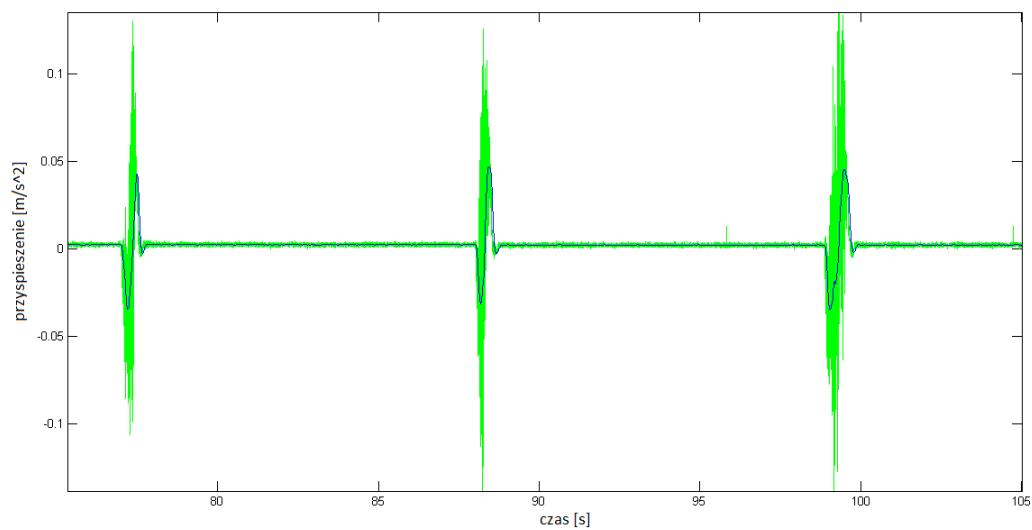
- zakłócenia/błędy transmisji cyfrowej;
- typowy dla czujników szum biały;
- zakłócenia pochodzące od swobodnie wiszącego wahadła.

W przypadku pomiarów przyspieszenia dla osi Y, ilość zakłóceń transmisji cyfrowej okazała się być znacznie mniejsza niż w przypadku pomiarów dla osi Z żyroskopu analizowanych w eksperymencie dotyczącym prędkości kątowej. Wynika to z faktu, że dane te znajdują się bliżej początku ramki przesyłanej przez magistralę UART.

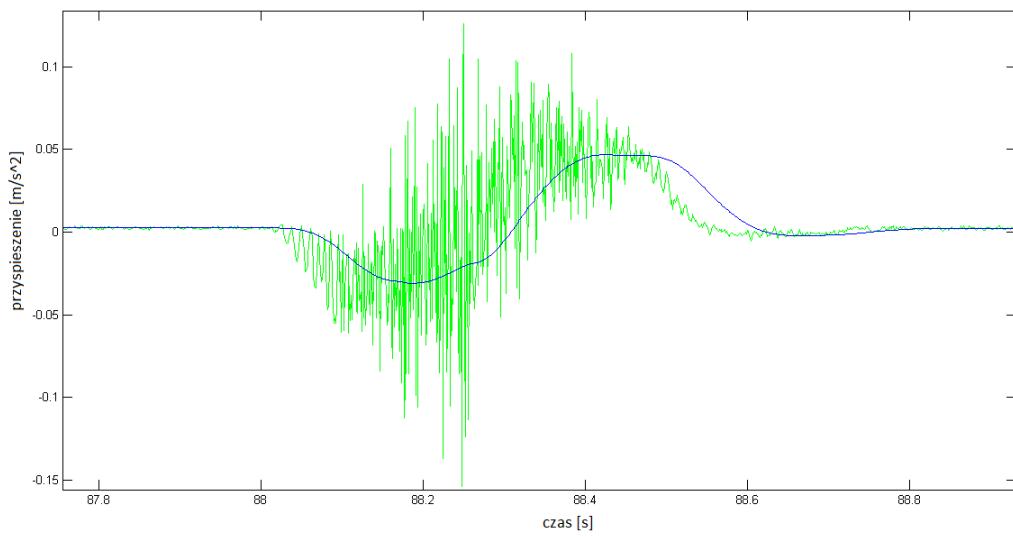
Dla danych z autonomicznego czujnika zastosowano filtr dolnoprzepustowy sporządzony na podstawie komendy butter(2,0,01);

Użycie takiej filtracji spowodowało znaczne wygładzenie wykresu, zgodnie z oczekiwaniemi. Niestety wyraźnie widać też negatywne efekty filtra - przesunięcie w fazie oraz tłumienie.

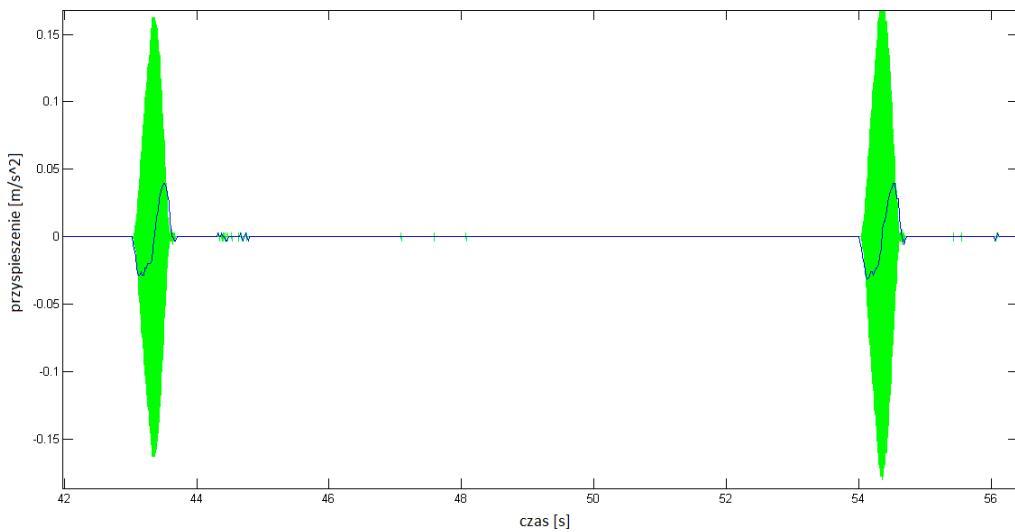
Dokonano normalizacji pomiarów, przez dodanie do każdej próbki stałej o wartości przeciwniej do średniej pomiarów. W ten sposób wyeliminowano błąd ustawienia czujnika.



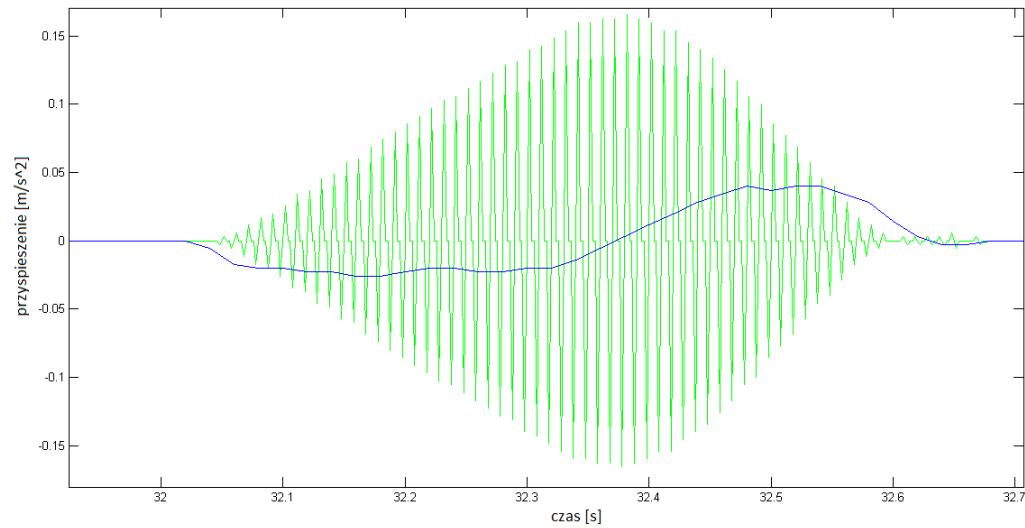
Rysunek 6.9: Zielony - surowe dane z akcelerometru. Niebieski - dane poddane filtracji.



Rysunek 6.10: Zielony - surowe dane z akcelerometru. Niebieski - dane poddane filtracji.

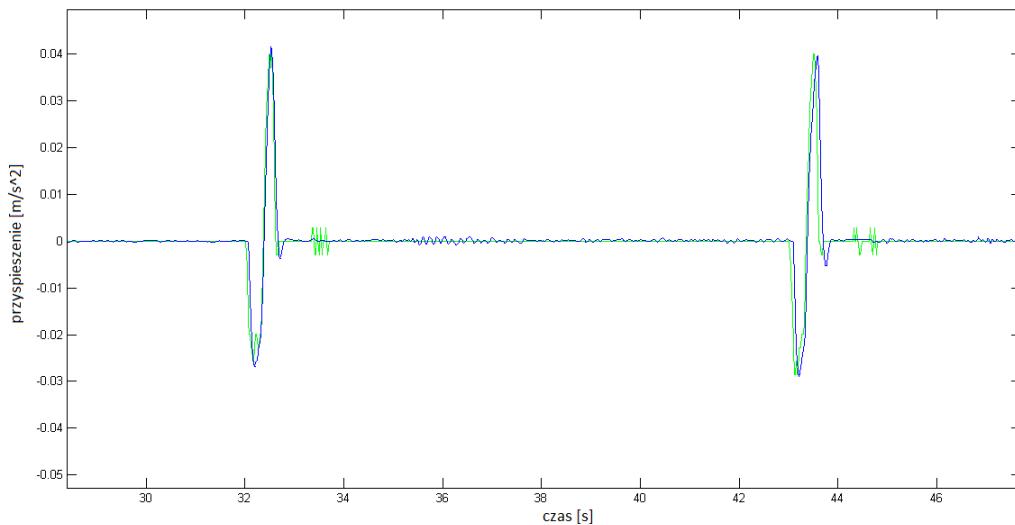


Rysunek 6.11: Zielony - surowe dane z enkoderów. Niebieski - przetworzone dane z enkoderów.



Rysunek 6.12: Zielony - surowe dane z enkoderów. Niebieski - przetworzone dane z enkoderów.

Podobnie jak w poprzednim eksperymencie, pojawił się problem zbyt dużej częstotliwości próbkowania pomiarów z enkoderów. Dla zwiększenia czytelności wykresów, do obliczenia pochodnej prędkości, wykorzystano co dziesiątą próbkę.



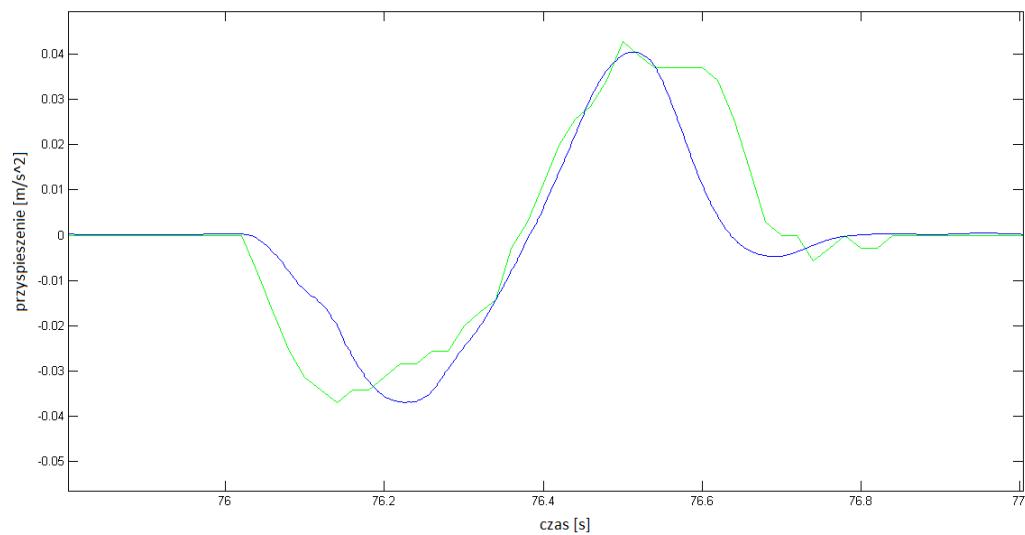
Rysunek 6.13: Zielony - przyspieszenie liniowe obliczone z enkoderów. Niebieski - przyspieszenie liniowe odczytane z akcelerometru.

6.3.6. Porównanie danych

Wykresy przyspieszeń pochodzących od danych z enkoderów oraz z autonomicznego czujnika przyspieszenia są podobne. Wykres danych z autonomicznego czujnika jest gładzszy, co spowodowane jest filtracją dolnoprzepustową. Różnica w gładkości wynika również ze względnie niewielkiej rozdzielczości enkoderów.

Dane generowane przez symulację dostarczane są w jednostkach układu SI. Podczas wykonywania eksperymentu zakres pracy żyroskopów urządzenia pomiarowego ustalony był na $\pm 8g$. Dokonano odpowiedniego przeskalowania pomiarów.

Próba obliczenia prędkości liniowej oraz położenia przy pomocy całki dla tego eksperymentu również zakończyła się porażką. Błędy kumulacyjne były znaczące nawet po normalizacji pomiarów. Prędkość wyliczona w ten sposób nie była zerowa dla odcinków czasu, gdy wózek znajdował się w rzeczywistości w spoczynku. Miała ona wartość losową, większą lub mniejszą od zera.



Rysunek 6.14: Zielony - przyspieszenie liniowe obliczone z enkoderów. Niebieski - przyspieszenie liniowe odczytane z akcelerometru.

7. Podsumowanie

7.1. Ogólnie

Podstawowy cel pracy polegający na budowie autonomicznego czujnika przyspieszenia został osiągnięty. Zbudowany czujnik zbiera nie tylko informacje o przyspieszeniu, dodatkowo dokonuje on pomiaru przyspieszeń kątowych obiektu, na którym jest zainstalowany. Udało się wykonać eksperyment potwierdzający działanie systemu oraz jego przydatność.

7.2. Realizacja wyznaczonych celów

- Zachowanie minimalnej masy i rozmiaru układu.

Waga samego urządzenia pomiarowego to 16g. Jego wymiary to 70x30x20mm. Waga 2 baterii AAA wykorzystanych podczas eksperymentu to 28g. Podczas eksperymentu wykorzystano stosunkowo ciężkie baterie, ponieważ system pracował z bardzo dużą częstotliwością próbkowania oraz nieprzewidywalny był czas trwania całego badania. Urządzenie jest również w stanie pracować na znacznie lżejszych bateriach, np. CR2032. Czas działania jest wtedy istotnie krótszy.

- Minimalny pobór mocy.

Urządzenie było w stanie pracować z częstotliwością próbkowania 1kHz, zasilane dwoma bateriami typu AAA, nieprzerwanie przez około godziny. Po tym czasie baterie nadal były naładowane. Najbardziej energochłonnym modułem jest moduł radiowy.

- Jak największa przepustowość łącza.

Zrealizowany system pomiarowy był w stanie dostarczać pomiary z częstotliwością zbliżoną do 1kHz, co jest maksymalną częstotliwością próbkowania dla akcelerometru znajdującego się w module MPU-6050. Najbardziej ograniczającym medium transmisji danych okazała się magistrala I²C.

- Niski koszt.

Całkowita cena budowy urządzenia wynosiła 160zł.

7.3. Rozwój pracy

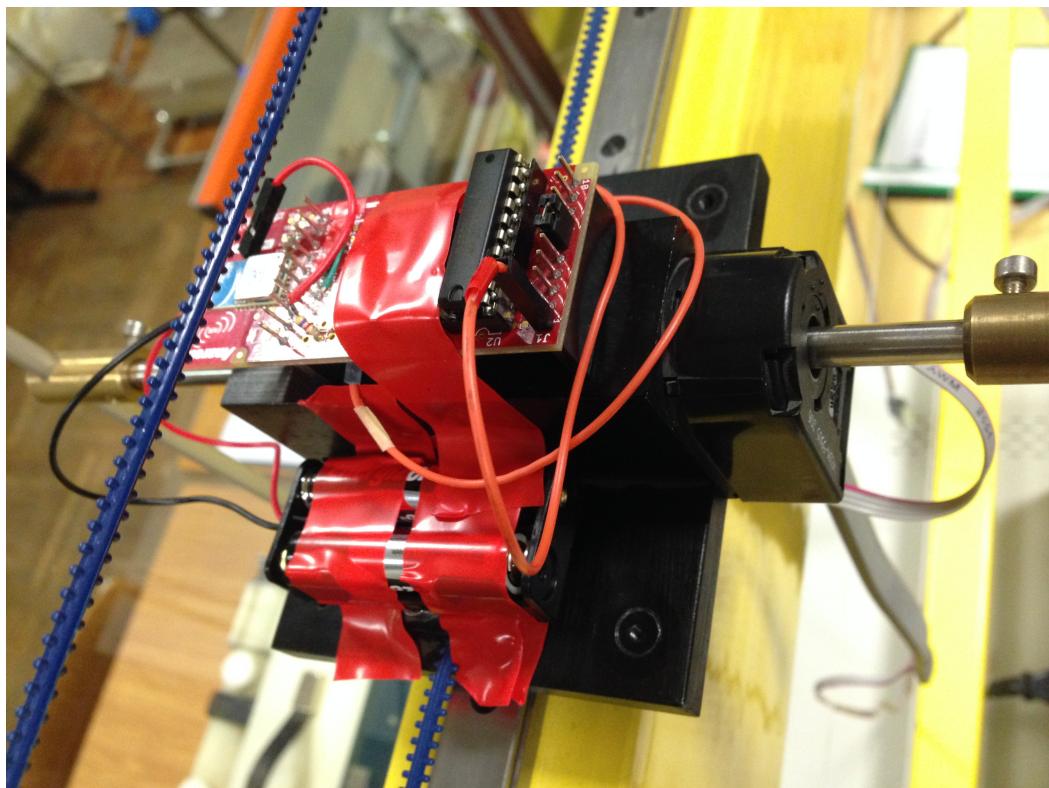
- Zużycie energii może zostać zmniejszone przez zastąpienie niektórych pętli w programie trybami niskiego poboru mocy. Również stworzenie innej biblioteki dla protokołu SPI pracującej z wykorzystaniem przerwań pozwoliłoby wydłużyć czas pracy na zasilaniu baterijnym. Możliwa jest także optymalizacja kodu z wykorzystaniem narzędzi Power Advice i Optimization Advice.
- Waga i rozmiar urządzenia mogą zostać znacznie zmniejszone przez opracowanie odpowiedniej płytki drukowanej i zastosowanie modułów w mniejszych obudowach.

- Na rynku dostępna jest inna wersja wykorzystanego w tej pracy czujnika o nazwie MPU-6000, która wyposażona jest w magistralę SPI. Maksymalne taktowanie tej magistrali dla MPU-6000 to aż 20MHz. Ze względu na warstwową budowę przygotowanych dla tej pracy bibliotek, możliwe jest łatwe dostosowanie oprogramowania do tej wersji urządzenia.
- Kolejną możliwością rozwoju tej pracy jest wykorzystanie dodatkowych funkcji modułu czujnika ruchu. Czujnik posiada m.in. wbudowany cyfrowy procesor ruchu (DMP), którego wykorzystanie byłoby bardzo ciekawym projektem. Niestety dokumentacja tego modułu nie jest publicznie dostępna, a jej uzyskanie wymaga m.in. podpisania umowy o poufności.
- Użyteczne byłoby zaprojektowanie systemu zdalnego sterowania urządzeniem. Przy zastosowaniu odpowiedniego protokołu i aplikacji graficznej umożliwiłoby to wygodne zarządzanie funkcjami i parametrami urządzenia z komputera PC.

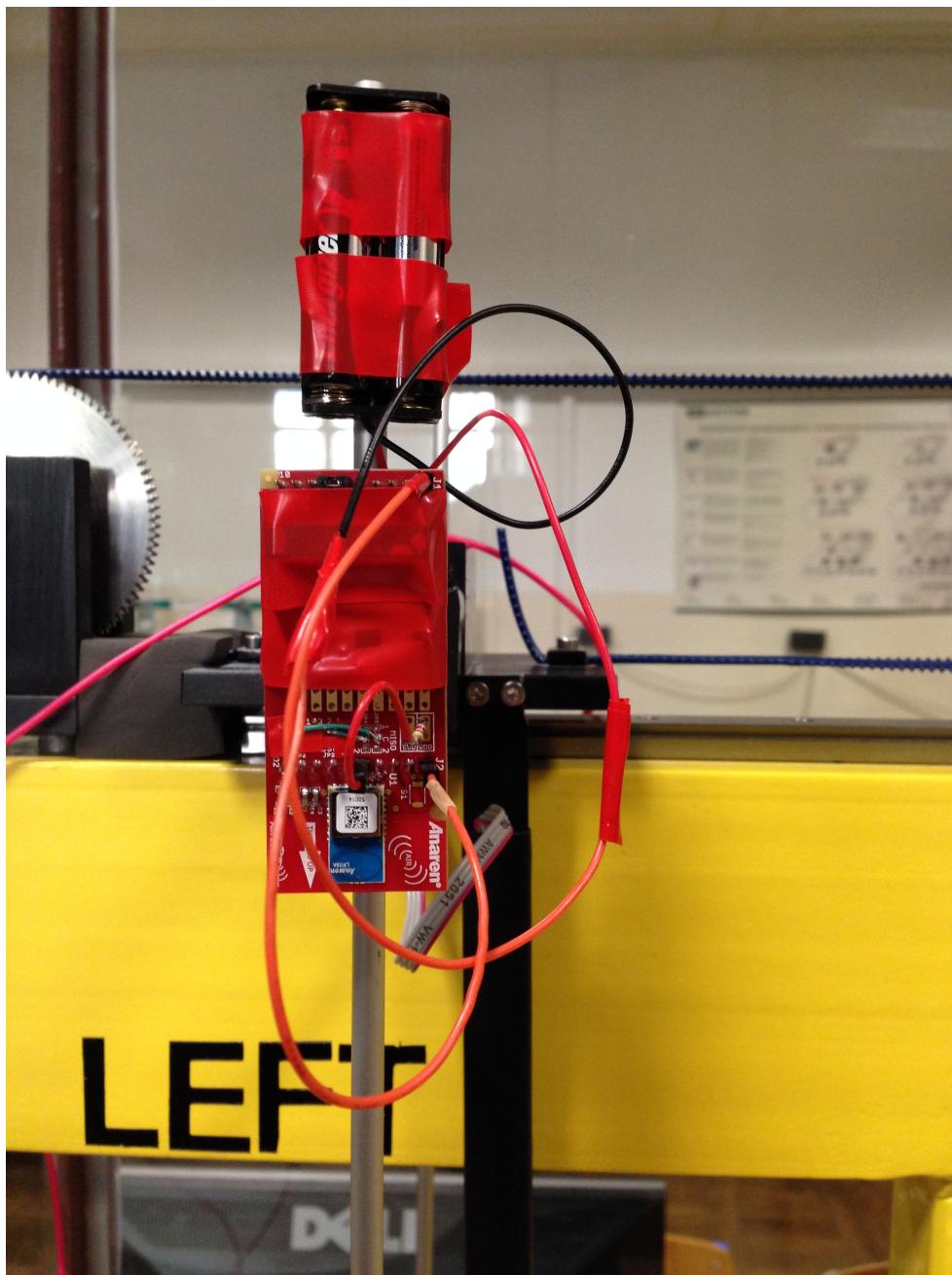
Bibliografia

- [1] <http://e2e.ti.com/group/universityprogram/educators/f/777/t/165038.aspx>.
- [2] <http://www.mail-archive.com/mspgcc-users@lists.sourceforge.net/msg11305.html>.
- [3] <http://www.ti.com/tool/ccstudio>.
- [4] Anaren. *AIR BoosterPack Users Manual*.
- [5] InvenSense. *MPU-6000 and MPU-6050 Product Specification*, revision 3.4 edition, 2013.
- [6] InvenSense. *MPU-6000 and MPU-6050 Register Map and Descriptions*, revision 4.2 edition, 2013.
- [7] J. Kowalski. Bezprzewodowy system akwizycji danych. *Elektrotechnika i Elektronika*, 29:1–7, 2010.
- [8] F. Leens. An introduction to i2c and spi protocols. 2009.
- [9] Texas Instruments. *MSP430 Product Brochure*.
- [10] Texas Instruments. *Using the USCI I2C Master*, 2007.
- [11] Texas Instruments. *CC110L Value Line Transciever Datasheet*, 2013.
- [12] Texas Instruments. *MSP430 G2x53 and G2x13 Datasheet*, 2013.
- [13] Texas Instruments. *MSP430x2xx Family User's Guide*, 2013.
- [14] Texas Instruments. *Universal Serial Communication Interface SPI Mode*, 2013.
- [15] A. Turnau. Laboratorium teorii automatów - "górnik".
- [16] Urzad Komunikacji Elektronicznej. *Wykaz systemów radiowych - zasady korzystania z częstotliwości ogólnodostępnych*.
- [17] K. S. Y. Wang. A new approach to realize uart. In *International Conference on Electronic & Mechanical Engineering and Information Technology*, 2011.

Dodatek A



Rysunek 1: Montaż urządzenia pomiarowego na wózku.



Rysunek 2: Montaż urządzenia pomiarowego na wahadle.