



Politechnika Wrocławska

Wydział Informatyki i Zarządzania

kierunek studiów: Inżynieria Systemów

specjalność: Systemy Sterowania

Praca dyplomowa – inżynierska

**Opracowanie platformy symulacyjnej do
analizowania zakłóceń pomiarowych**

Artem Melnyk

słowa kluczowe:
wariancja Allana
akcelerometr, żyroskop
szum, sygnał, zakłócenie

krótkie streszczenie:

Celem pracy było opracowanie platformy symulacyjnej do analizowania zakłóceń pomiarowych. Pomiary otrzymane od czujników zostały przeanalizowane na zawartość składników szumowych.

opiekun pracy
dyplomowej	Tytuł/stopień naukowy/imię i nazwisko	ocena	podpis
Ostateczna ocena za pracę dyplomową			
Przewodniczący Komisji egzaminu dyplomowego Tytuł/stopień naukowy/imię i nazwisko ocena podpis

Do celów archiwalnych pracę dyplomową zakwalifikowano do:*

- a) kategorii A (akta wieczyste)
- b) kategorii BE 50 (po 50 latach podlegające ekspertyzie)

* niepotrzebne skreślić

pieczęć wydziałowa

Wrocław
[2021]

Streszczenie

Celem niniejszej pracy dyplomowej było stworzenie platformy do analizy zakłóceń pomiarowych. Działanie platformy symulacyjnej zostało podzielone abstrakcyjnie na trzy etapy: generowanie pomiarów syntetycznych, analiza pomiarów rzeczywistych i prezentowanie wyników. Dane pomiarowe do analizy pomiarów rzeczywistych zostały otrzymane używając czujników, mianowicie akcelerometru i żyroskopu.

W części wstępnej została zawarta informacja o celach pracy oraz jej zakresie. Dodatkowo była przeprowadzona analiza stanu sztuki i opis narzędzi, które zostały wykorzystane w ramach pracy. Opis algorytmów generowania sygnałów syntetycznych oraz metody wariancji Allana są przedstawione w rozdziale drugim. Architektura strony i implementacja znajdują się w trzecim i czwartym rozdziałach.

Na końcu, w rozdziale „Prezentacja” został przedstawiony interfejs strony, zgodnie z analizą funkcjonalną użytkownika oraz omówienia rozwiązań nawiązujących do planu pracy.

Abstract

The purpose of this thesis was to create a platform for measurement disturbances analyses. The operation of the simulation platform has been abstractly divided into three stages: synthetic measurements generation, actual measurements analyze and results presentation. The necessary measurement data for the analysis of real measurements were obtained using the accelerometer and gyroscope.

The introductory part contains information about the goals and the scope of the work. Additionally, an analysis of the existing solutions and a description of the tools that were used in the work was carried out. The description of the algorithms for generating synthetic signals and the Allan variance method are presented in the second chapter. Site architecture and implementation are in the third and fourth chapters, accordingly.

Finally, in the "Presentation" chapter, the website interface was presented, in accordance with the users functional analysis and solutions related to the thesis.

Spis treści

Streszczenie	2.
Abstract	2.
Wstęp.....	5.
1.1 Geneza pracy.....	5.
1.2 Analiza stanu sztuki	6.
1.3 Stos technologiczny	10.
1.3.1 Technologie pomiarowe	10.
1.3.2 Technologie implementacyjne	13.
1.3.3 Technologie webowe	14.
1.4 Cel pracy	14.
1.5 Plan pracy	14.
2. Metody algorytmów i sygnałów pomiarowych.....	15.
2.1 Abstrakcyjny podział projektu.....	15.
2.2 Moduł do generowania syntetycznych sygnałów	17.
2.2.1 Ruch_idealny	17.
2.2.2 Szum_biały	19.
2.2.3 Szum różowy	20.
2.2.4 Szum_czerwony	21.
2.2.5 Szum fioletowy	22.
2.2.6 Dryft.....	23.
2.3 Platforma pomiarowa.....	24.
2.4 Wariancja Allana	25.
2.4.1 Szum ARW/VRW	26.
2.4.2 Szum kwantyzacji.....	27.
2.4.3 Szum BI	27.
2.4.4 Szum RRW	28.
2.4.5 Szum RR	29.
3. Projekt platformy symulacyjnej	31.
3.1 Analiza wymagań funkcjonalnych i нефункциональных.....	31.
3.1.1 Analiza wymagań funkcjonalnych.....	31.
3.1.2 Analiza wymagań нефункциональных.....	32.
3.2 Architektura platformy symulacyjnej	33.
3.2.1 Architektura modułu IMU	33.
3.2.2 Architektura modułu data_processing	35.

3.3	Architektura strony internetowej.....	37.
4	Implementacja platformy symulacyjnej.....	39.
4.1	Instrukcja uruchomienia programu.....	39.
4.2	Implementacja interfejsu	39.
4.3	Interakcja między Pythonem a PHP	41.
4.4	Implementacja IMU.....	41.
4.5	Implementacja Data_processing	42.
5	Prezentacja	44.
5.1	Interfejs	44.
5.2	Rozwiązania.....	49.
6	Podsumowanie.....	54.
6.1	Kierunki dalszego rozwoju	54.
7	Bibliografia.....	55.
	Spis rysunków	57.
	Spis tabel	59.

Wstęp

W tym rozdziale została omówiona geneza pracy, przeprowadzono analizę rozwiązań stanu sztuki, opis technologii użytych w trakcie implementacji platformy, cel pracy oraz plan realizacji.

1.1 Geneza pracy

W ostatnich latach obserwuje się wzrost popularności wśród czujników inercyjnych. Dane rodzaje czujników (akcelerometr i żyroskop) są wykorzystywane w wielu branżach, mianowicie w elektronice, przemyśle motoryzacyjnym, lotniczym itp. Rozwój współczesnych telefonów komórkowych, konsoli do gier, sprzętu medycznego oraz sportowego nie byłby możliwy bez wykorzystania czujników opartych na technologii MEMS. Przykładem mogą być smartfony wyposażone w czujnik żyroskopowy, który służy do określania kąta nachylenia telefonu lub czujnik przyspieszenia, używanego przez Google Maps, pozwalający sprawdzić aktualną prędkość jadącego samochodu. Oprócz telefonów tego typu czujniki znajdują się również w samolotach oraz w przemyśle stoczniowym.

W ramach tej pracy pomiary otrzymane od czujników opartych na technologii MEMS traktujemy jako sygnał. Sygnałem nazywamy pewną wielkość, która umożliwia przekazanie informacji oraz może zostać zmieniona w czasie. Przykładów sygnału istnieje nieskończenie wiele. Z tego powodu zostały one scharakteryzowane i podzielone na różne grupy. Zależność temperatury od czasu, zdjęcie, nagrany filmik – są to przykłady sygnałów. Każdy sygnał zawiera w sobie „pożądane” i „niepożądane” składniki. Część sygnału uznana za „pożądaną” jest wartością, którą chcemy przekazać. W rzeczywistości istnieją różne siły fizyczne, wpływające na każde ciało, obiekt itp. Na sygnał wpływają zakłócenia stałe oraz losowe. Stałe zakłócenia mogą być spowodowane z winy człowieka (urządzenia do zagłuszania sygnałów, stacje bazowe), natury (promieniowanie słoneczne) lub odbiornika (ograniczenie wewnętrzne: brak technologii, przy użyciu której jest wysyłany sygnał). Stały rodzaj zakłóceń jest do przewidzenia i można go zniwelować wieloma sposobami: poprzez użycie odpowiedniej technologii, konstrukcji, wzoru matematycznego lub zmiany dyslokacji urządzenia. Inaczej jest w przypadku losowych zakłóceń, nazywane także „szumami”. Szum ze swojej natury jest zjawiskiem chaotycznym i zawsze istnieje razem z sygnałem. Przez to całkowite usunięcie składników szumowych nie jest możliwe, natomiast używając odpowiednich filtrów, mianowicie falkową redukcję szumu (ang. *Wavelet denoising*) lub filtr medianowy (ang. *Median filter denoising*) można zniwelować wpływ składników szumowych do minimum. W ramach tej pracy zostały omówione i przedstawione metody do znalezienia oraz scharakteryzowania składników szumowych w sygnałach.

Poruszony temat jest aktualny nie tylko dla osób pracujących w branżach elektronicznych, ale również dla studentów. Czasami może się wydarzyć sytuacja, że skrypt napisany w edytorze jest poprawny, ale jednak urządzenia w warunkach domowych zachowują się niewłaściwie. Wyobraźmy sobie sytuację, gdy należy zaprojektować model szlabanu, do którego zostały użyte takie narzędzia: Arduino UNO, czujnik odległości HC-SR04, moduł RfID (służy do odczytu kart dotykowych) oraz silniczki odpowiedzialny za otwieranie lub zamykanie szlabanu. W jakimś momencie czujnik przestaje reagować na obiekt, który znajduje się w jego zasięgu lub przesyła błędny sygnał, czyli informuje o obecności obiektu przed szlabanem, gdy w rzeczywistości jest pusto. W takim przypadku mamy do czynienia z szumami systemowymi, wpływającymi na sygnał. Czasami do regulowania składników szumowych wystarczy użyć funkcji Delay, która zwalnia przesyłanie sygnału z czujnika do płytki Arduino. W rzeczywistości to wygląda w ten sposób, że czujnik musi przez czas ustalony z góry przekazywać sygnał potwierdzający obecność obiektu przed szlabanem. W bardziej

skomplikowanych warunkach, np. gdy zależy na tym, aby pomiary były dokładne, należy używać odpowiednich filtrów.

Ze wzrostem popularności oraz zwiększenia branż zastosowania czujników inercyjnych (akcelerometr oraz żyroskop) potrzeba w otrzymaniu dokładnego sygnału wzrosła. W celu analizowania składników „niepożądanych” stosuje się wiele różnych metod. Jedną z klasycznych metod, służącą do analizy szumowej sygnałów w dziedzinie czasowej to wariancja Allana. Istnieją również inne metody oparte na estymacji widmowej gęstości mocy (ang. *Power Spectral Density*, PSD) sygnałów w dziedzinie częstotliwościowej. W ramach tej pracy został wybrany jeden ze sposobów implementacji, w którym wyznaczono wariancję Allana na podstawie uśrednienia współczynnika wyjściowego próbki [10] (ang. *Calculate Allan variance using averages of output rate samples*). Na wykresie końcowym składniki szumowe reprezentowane w sygnale posiadają odpowiednie nachylenie.

Praca zawiera opracowanie odpowiednich technologii webowych, na których zostały umieszczone informacje bazowe o czujnikach, sygnałach, zakłóceniach pomiarowych, algorytmie Allana w prostej formie. Zostanie udostępniona możliwość generowania pomiarów losowych oraz przeprowadzenie analizy szumowej pomiarów otrzymanych w sposób rzeczywisty.

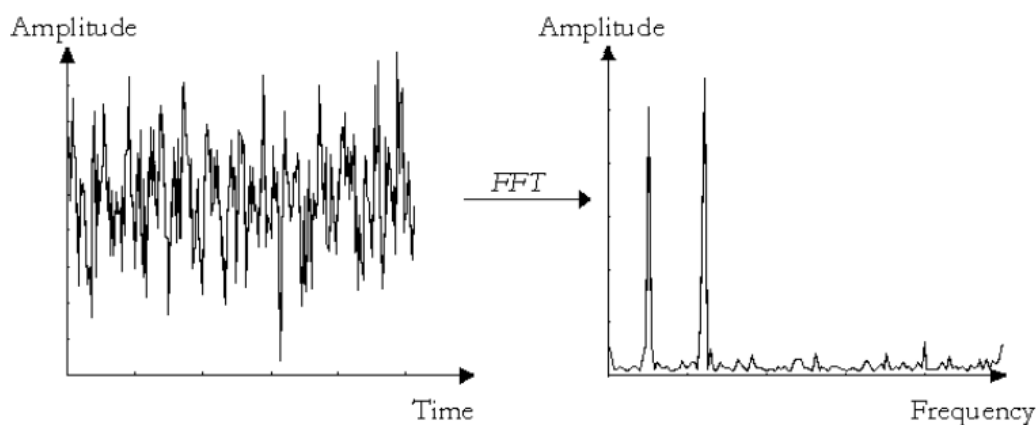
1.2 Analiza stanu sztuki

Główne kryteria oceny istniejących rozwiązań:

1. Wybór właściwej metody do analizy szumowej
2. Prezentacja rozwiązań
3. Pomiary rzeczywiste
4. Środowisko implementacyjne

Podstawowe zadanie w analizie szumowej rozpoczyna się od wyboru właściwej metody. Istnieje wiele metod analizy szumowej czujników inercyjnych, w tym widmowa gęstość mocy wykorzystująca funkcję autokorelacyjną, a także metody dyspersji. Szczegółowy opis został przedstawiony w książce „Zastosowanie przetwarzania sygnałów w fuzji danych strumieniowych” [5]. W tym opracowaniu przedstawiono zasadę działania, opisano analizę wykresów końcowych oraz sprowadzone porównania metod wariancji Allana i widmowej gęstości mocy.

Wariancja Allana bazuje na reprezentacji czasowej sygnału, a analiza widmowa gęstości mocy na częstotliwościowej. Reprezentacja czasowa sygnału pokazuje sposób zmiany sygnału w czasie, ilustruje jego bieżący kształt, natomiast częstotliwościowa przedstawia intensywność zmiany oraz w jaki sposób one zachodzą. Wykres w dziedzinie czasu i częstotliwości sygnału przedstawiono poniżej. Używając transformacji Fouriera można przekonwertować sygnał z dziedziny czasowej do częstotliwościowej. Metoda również działa w stronę odwrotną, wtedy stosuje się odwrotnej transformacji.



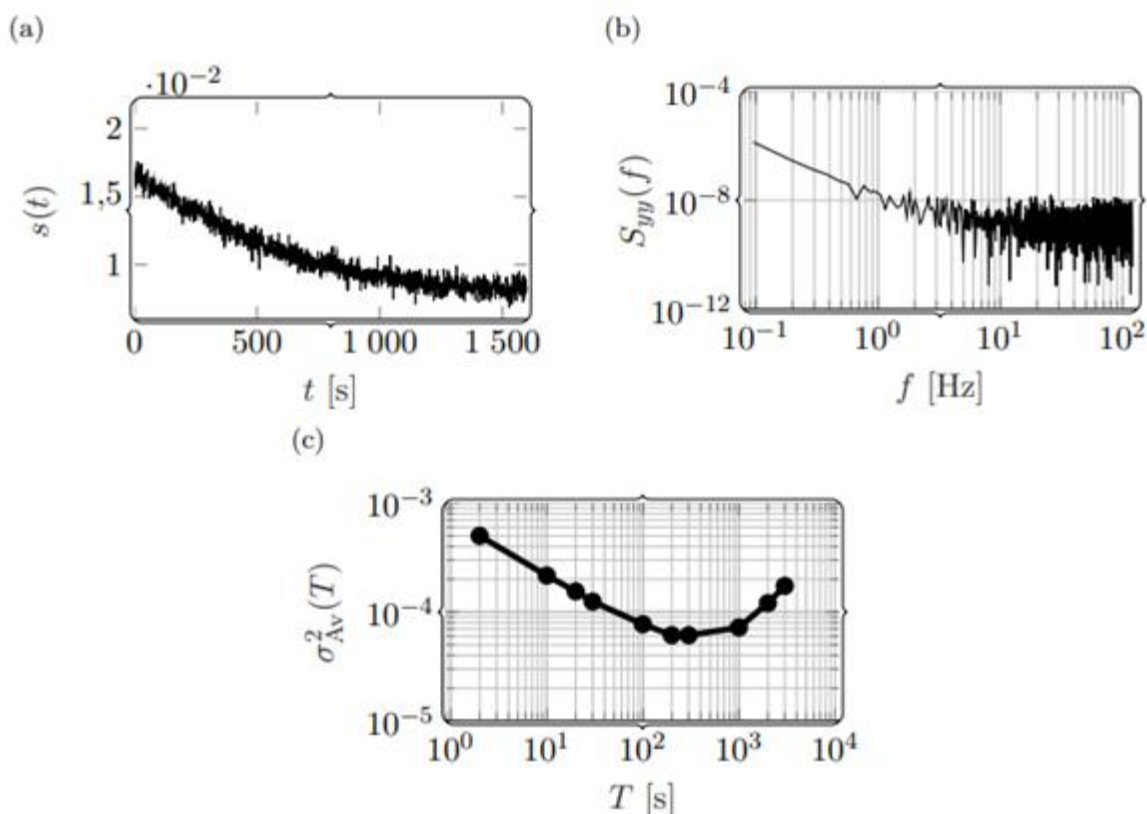
Rys. 1.1 Reprezentacja czasowa i częstotliwościowa sygnału

W porównaniu do wariancji Allana prezentowanie składników szumowych widmową gęstością mocy jest bardziej ograniczone. Dodatkowo implementacja metody widmowej jest o wiele trudniejszym zadaniem. Składniki szumowe na wykresie końcowym przedstawione w postaci linii prostych (zgodnie z częścią teoretyczną) z odpowiednim kątem nachylenia.

Tabela 1.1 Porównanie metod analizy szumowej

Typ szumu	Wariancja Allana	Nachylenie	Widmowa Gęstość Mocy	Nachylenie
Szum kwantyzacji	✓	-1	✓	1
Szum ARW/VRW	✓	-0,5	✓	-1
Szum skorelowany	✓	Lokalny maksimum +0.5 ... -0.5	✗	
Szum BI	✓	0	✓	-0,5
Szum ARRW/VRRW	✓	0,5	✓	0
Szum RR	✓	1	✗	

W pomiarach rzeczywistych analiza wykresów nie jest takim trywialnie prostym zadaniem. Czasami pomiary są mocno rozrzucone, co jednak bardziej komplikuje wyznaczenie odpowiedniego nachylenia. Wykresy na bazie rzeczywistych pomiarów prawdopodobnie zawierają z parę składników szumowych. Określenie ich typu polega na obliczeniu wyznaczników szumowych, które będą zbliżone do teoretycznych, przedstawionych w tabeli 1.1.



Analiza pomiaru z czujnika żyroskopowego dla osi x :
a) pomiar, b) widmowa gęstość mocy, c) wariancja Allana

Rys. 1.2 Ilustracja wykresów końcowych dla obu metod [5].

Na rysunku 1.2 wykres (a) reprezentuje zależność pomiarów czujnika od czasu, wykres (b) - zależność gęstości widmowej od częstotliwości, wykres (c) - zależność odchylenia Allana od czasu uśrednienia.

Wyznaczenie kąta nachylenia prostej w analizie widmowej jest trudniejsze w porównaniu do wykresu Allana. Z tego powodu wariancja Allana zostaje preferowaną metodą w analizie szumowej.

Na rynku publikacji naukowych istnieje sporo opracowań powiązanych z tematem w ramach niniejszej pracy. Jedno z nich jest dostępne w opracowaniu „Charakterystyka błędów i szumów w czujnikach inercyjnych MEMS metodą wariancji Allana”[9].

1. Wybór właściwej metody do analizy szumowej

Algorytm wybrany do analizy składników szumowych – wariancja Allana. Zasada działania, charakterystyki składników szumowych, dokładność szacowania, kąty nachylenia prostych na wykresie reprezentującym typ szumu, zostały omówione szczegółowo. Do implementacji został wykorzystany wzór polegający na obliczeniu wariancji Allana używając kątów wyjściowych θ .

$$\theta^2(\tau) = \frac{1}{2\tau^2(N-2m)} \sum_{k=1}^{N-2m} (\theta_{K+2m} - 2\theta_{K+m} + \theta_K)^2$$

Rys. 1.3 Wariancja Allana oparta na kontach wyjściowych [9].

Dodatkowo zostały przedstawione sposoby stosowane do odsumowania sygnału. Dyskretna transformacja falkowa (ang. *Discrete wavelet transform*) i filtr medianowy (ang. *Median filter*)..

2. Prezentacja rozwiązań

W dokumentacji zostały przedstawione i omówione wykresy pokazujące zależność otrzymanych pomiarów od liczby przypadków dla obu czujników. Analiza szumowa przed i po zastosowaniu filtrów. Nie był zrobiony interfejs reprezentujący wyniki pomiarów, wygodny dla użytkownika. Wyniki zostały omówione w ramach dokumentacji.

3. Pomiary rzeczywiste

W celu otrzymania pomiarów został użyty wysokowydajny czujnik IMU 3DM-GX3-25, wykorzystujący technologię czujników MEMS. Czujnik wyposażony w trzyosiowy akcelerometr i żyroskop. Pomiary dla akcelerometru wyrażone grawitacyjnie (g), natomiast dla czujnika żyroskopowego zostały wyrażone w radianach (ras/s). Do obliczenia wariancji Allana dane zostały przetworzone w m/s^2 dla akcelerometru oraz w stopnie dla czujnika żyroskopowego.

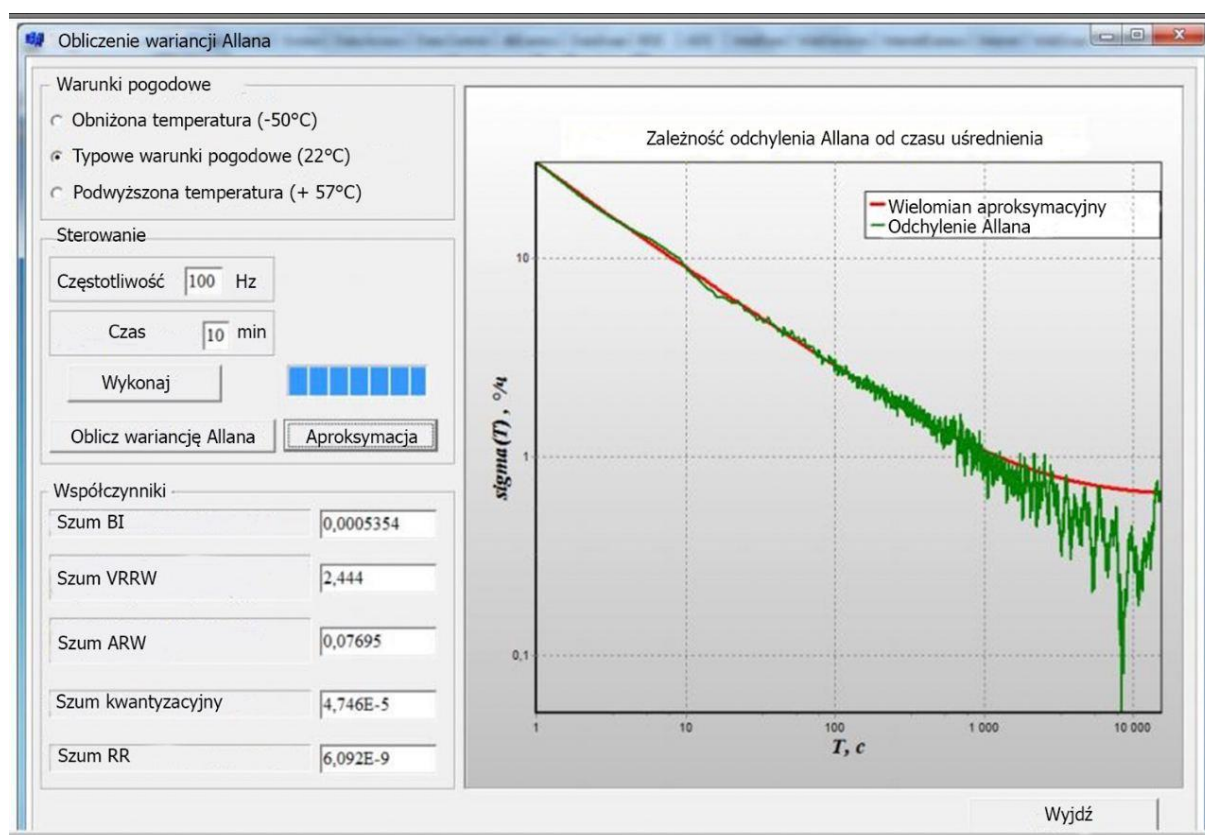
4. Implementacja

Algorytmy zostały zaimplementowane w Matlabie. Kod źródłowy jest przedstawiony w dodatku B.

W kolejnym opracowaniu „*Analiza zakłóceń pomiarowych dynamicznie dostrojonego żyroskopu metodą wariancja Allana*”[12] istotny jest interfejs programu. Opis kryteriów oceny istniejących rozwiązań w tym przypadku został pominięty, ponieważ struktura pracy, opis części teoretycznej, wyboru właściwej metody oraz pomiarów rzeczywistych są podobne do poprzedniego artykułu.

Idea pracy polega na stworzeniu i opracowaniu algorytmu do oceny sygnału otrzymanego w sposób eksperymentalny, używając dynamicznie dostrojonego żyroskopu (ang. *Dynamically tuned gyroscope*, DTG) w różnych warunkach pogodowych (do tego celu został on umieszczony w specjalnym pomieszczeniu, które umożliwia na regulowanie temperatury).

Po uruchomieniu programu pojawia się widok główny „*Obliczenie wariancji Allana*”. Użytkownik, stosując odpowiednie przyciski reguluje pomiary, na przykład temperaturę w specjalnym pomieszczeniu. Po odpowiedniej regulacji w wyniku został przedstawiony końcowy wykres odchylenia Allana, wykres aproksymacji oraz współczynniki reprezentujące odpowiedni szum w postaci liczbowej.



Rys. 1.4 Ilustracja środowiska prezentującego rozwiązania [12].

Różnica pomiędzy opracowaniem, utworzonym w ramach realizacji tej pracy dyplomowej a większością podobnych, polega na tym, iż został stworzony interfejs wraz z opisem rozwiązań w postaci strony webowej, na którym umieszczona informacja zgodnie ze standardami UX (user experience).

1.3 Stos technologiczny

Technologii wykorzystywane podczas realizacji niniejszej pracy zostały podzielone ze względu na ich rolę.

Do pierwszej grupy odnosimy narzędzia, rola których polegała na pobraniu pomiarów rzeczywistych. Grupa pierwsza zostanie określona jako technologie pomiarowe. Do tej grupy należy część elektroniczna projektu.

Druga grupa zawiera narzędzia służące do implementowania trudnych algorytmów, symulacji składników szumowych oraz sygnałów. Są to fundamentalne narzędzia, używając których napisano główne skrypty platformy. Druga grupa została scharakteryzowana jako technologie implementacyjne.

Z trzecią grupą łączymy narzędzia służące do stworzenia interfejsu. Wyniki pomiarów należy przedstawić w sposób czytelny, zrozumiały użytkownikowi. Narzędzia stosowane w ramach tej grupy są traktowane jako technologie webowe.

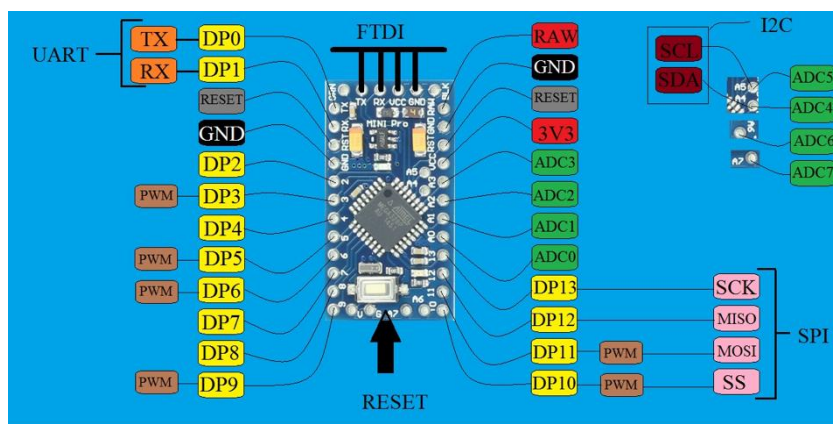
1.3.1 Technologie pomiarowe

Pierwsza grupa urządzeń, rola których polega na uzyskaniu pomiarów rzeczywistych i zapisywaniu ich w pliku **csv* lub **txt*. Abstrakcyjnie można określić, że działanie technologii

pomiarowych polega na otrzymaniu sygnału wejściowego do narzędzi, określanych jako grupa druga.

Do niniejszej grupy należą następujące urządzenia: platforma Arduino Pro Mini, czujnik MPU6050, płytki prototypowa i przewody służące do interakcji między czujnikiem a Arduino.

Arduino skonstruowane są na bazie mikrokontrolera Atmega328.



Rys. 1.5 Schemat Arduino Pro Mini [6].

Mikrokontroler składa się z 14 cyfrowych wejść/wyjść (6 z których wykorzystują jako wyjścia PWM), 8 wejść analogowych, rezonator kwarcowy oraz przycisk reset. Sześciostykowe złącze służy do zasilania i interakcji z płytką Arduino przez kabel USB.

Tabela 1.2 Specyfikacja techniczna Arduino Pro Mini [7].

Mikrokontroler	Atmega328
Napięcie zasilania VCC	5V
Napięcie zasilania RAW	5V – 12V
Cyfrowe wejścia/wyjścia	14 (6 z których wykorzystuje się jako wyjścia PWM)
Analogowe wejścia	8
Ciągłe napięcie przez wejście/wyjście	40 mikroAmper
Prąd maksymalny wyjścia 3.3 V	50 mikroAmper
Flash-pamięć	32 kB, gdzie 2 kB wykorzystuje się dla bootloadera
SRAM	2 kB
EEPROM	1 kB
Maksymalna częstotliwość zegara	8 MHz
Wbudowanych diod LED	16
Masa	2 g
Wymiary	33 x 18 mm

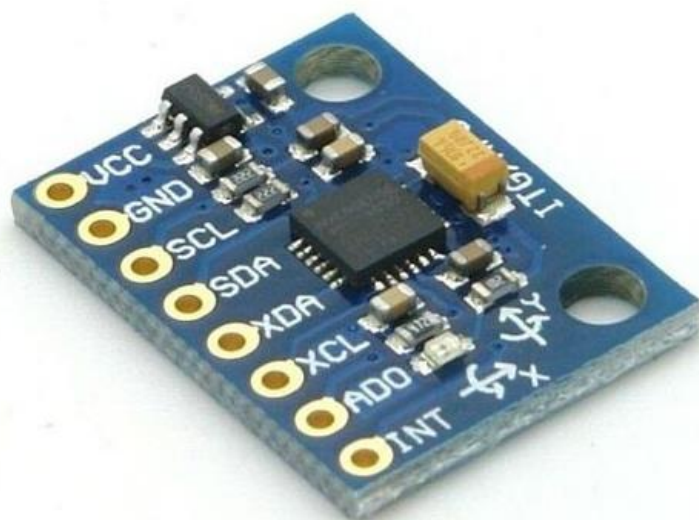
Na rysunku 1.5 można zauważyć następującą legendę:

- Wyjścia analogowe zaznaczone przez A0, A1, A2, A3, A4, A5, A6, A7.
- Wyjścia/Wejścia cyfrowe przez zwykłe liczby od 0 do 13. W sumie istnieje 14 cyfrowych pinów.
- Reset na schemacie zaznaczono jako RST i służy do restartu mikrokontrolera.

- RAW służy do podłączenia napięcia, którego nie możemy regulować.
- VCC służy do podłączenia regulowanego napięcia 3V lub 5V.
- GND – przewody uziemiające.

• I2C to szeregową dwukierunkową magistralę służącą do przesyłania danych pomiędzy układami elektronicznymi. W mikrokontrolerze składa się z dwóch pinów: SCL – przesyłanie sygnału zegarowego, w Arduino Pro Mini zaznaczony przez wejście analogowe A5; SDA – pin, przez który odbywa się przesyłanie danych, w kontrolerze jest zaznaczony przez A4. Czujnik łączy się z Arduino przez złącza męsko-żeńskie, męsko-męskie lub żeńsko-żeńskie.

Do pomiarów rzeczywistych położenia ciała w przestrzeni służy czujnik **MPU6050**. Moduł z żyroskopem, akcelerometrem oraz termometrem zbudowany na bazie MPU6050 oraz dość często wykorzystywany w amatorskiej robotyce.



Rys. 1.6 Czujnik MPU6050 [8].

Płytkę modułu również zawiera niezbędną wiązkę MPU6050, w tym rezystory podciągające interfejs I2C. Żyroskop wykorzystuje się do pomiaru liniowego przyspieszenia, a akcelerometr do pomiaru prędkości kątowej.

Tabela 1.3 Specyfikacja techniczna czujnika MPU6050 [8].

Zasilanie	3,5 – 6V
Pobór prądu	500 mikroAmper
Zakres pomiarowy akcelerometru	$\pm 2 \pm 4 \pm 8 \pm 16g$
Zakres pomiarowy żyroskopu	$\pm 250 \ 500 \ 1000 \ 2000 \ ^\circ / s$
Interfejs	I2C

Wartości wyjściowe otrzymane od czujnika MPU6050 na częstotliwości 100Hz zapisujemy w postaci pliku z rozszerzeniem *.csv lub *.txt.

1.3.2 Technologie implementacyjne

Technologie implementacyjne bazują się na środowisku **Pycharm** oraz bibliotekach, które zostały zainstalowane w ramach projektu. Pycharm to zintegrowane środowisko programistyczne (IDE) dla języka programowania Python firmy JetBrains [11].

Symulacja zakłóceń pomiarowych oraz analizowanie sygnałów rzeczywistych odbywa się w środowisku Pycharm, które jest bazowym narzędziem pozwalającym wykonanie różnych zadań programistycznych. Środowisko Pycharm w znacznym stopniu ułatwiło przebieg zadań pojawiających się w pracy dyplomowej, mianowicie napisanie kodu do symulacji szumu, ruchu żyroskopu i akcelerometru oraz pobieranie plików tekstowych.

Główne zalety Pycharm:

- Inteligentna edycja kodu z podświetlaniem składni języku Python
- Edycja oraz wsparcie dla HTML, CSS oraz JavaScript
- Możliwość instalacji dodatkowych pluginów itp.

Praca została zaimplementowana w środowisku Pycharm używając języka Python. Podczas realizacji pracy dyplomowej zostały użyte następujące biblioteki:

1) NumPy – służy do obsługi wielowymiarowych tablic oraz macierzy. Zawiera w sobie dużą ilość wielowymiarowych funkcji matematycznych.

2) Matplotlib – wykorzystywana do budowania wykresów. Dana biblioteka posiada szeroki spektrum zastosowania.

3) Math – przedstawia zestaw funkcji do wykonania operacji matematycznych, trygonometrycznych oraz logarytmicznych.

4) Allantools – przeznaczona do obliczania odchylenia Allana oraz powiązanych statystyk dotyczących czasu i częstotliwości.

5) SciPy – biblioteka uzależniona od NumPy i posiada zbiór narzędzi służących do wygodnego i szybkiego modyfikowania tablicą N-wymiarową. Została zaprojektowana do pracy z NumPy i zapewnia wiele wydajnych metod numerycznych, mianowicie procedury integracji numerycznej oraz optymalizacji. W pracy dyplomowej wykorzystuje się następujące moduły danej biblioteki:

- Linalg – procedury algebry liniowej.
- Constants – stałe fizyczne i matematyczne. Na przykład do symulacji ruchu żyroskopu lub akcelerometru potrzebujemy uwzględnić przyspieszenie Ziemi.

6) ABC – abstrakcyjna klasa bazowa. Wbudowana do nowych wersji Python, zaczynając od wersji 3.0. Wykorzystuje się do uwiarygodnienia kodu i uniknięcia konieczności wielokrotnego powtarzania tej samej nazwy lub metody.

Narzędzia wraz z bibliotekami opisane powyżej pozwalają w sposób wiarygodny dokonać symulacji ruchu czujnika inercyjnego oraz składników szumowych występujących w sygnale.

1.3.3 Technologie webowe

Do tego celu została stworzona **strona webowa** pobierająca wyniki od Pycharm i przedstawiająca je wraz z opisem.

Strona internetowa to dokument napisany w języku HTML, jest udostępniony w sieci Internet lub lokalnie na komputerze. Każda strona internetowa ma wspólne cechy: adres, logiczną strukturę, projektowanie itp. Dane pliki domyślnie ładują się na komputerze użytkownika, są przetwarzane przez wyszukiwarkę a następnie wyświetlane.

Podczas tworzenia strony internetowej zostały wykorzystane następujące narzędzia:

- Język HTML – używany do tworzenia struktury i wyświetlenia strony internetowej oraz jej zawartości; umożliwia przetwarzanie tekstu, wyodrębnianie elementów funkcjonalnych, tworzenie linków oraz dodawaniu obrazów, dźwięków i innych elementów multimedialnych na stronie.
- CSS – język stylów, odpowiada za ostylewanie dokumentów HTML, m.in. reguluje czcionkę, kolory, symbole i tło oraz szerokość i wysokość wyświetlanych elementów itp.
- JavaScript – jeden z najpopularniejszych języków programowania, obsługiwany przez wszystkie współczesne przeglądarki. Ma na celu „ożywienie” stron internetowych.
- jQuery – biblioteka JavaScript, zaprojektowana, aby uprościć pisanie skryptów podczas pracy z elementami HTML. Dana biblioteka posiada wiele gotowych funkcji, które można używać wraz z językiem JavaScript.

1.4 Cel Pracy

Cel pracy polega na analizie składników szumowych, które występują w sygnale otrzymanym na podstawie czujników akcelerometru i żyroskopu lub wygenerowanego syntetycznie.

1.5 Plan pracy

Do zrealizowania celu niniejszej pracy należało wykonać następujące działania:

1. Zapoznać się z materiałem teoretycznym
2. Dokonać symulacji ruchu żyroskopu i akcelerometru bez zakłóceń pomiarowych
3. Generowanie składników szumowych i połączenie ich z ruchem czujników
4. Zaprojektować i wykonać elektroniczne urządzenie pomiarowe
5. Wykonać serię eksperymentów i zebrać niezbędne dane pomiarowe
6. Zapoznać się z metodą odchylenia Allana i nauczyć się interpretować wyniki otrzymywane przy użyciu danej metody
7. Stworzenie wygodnego interfejsu, czyli strony internetowej używając najpopularniejsze technologii i narzędzie webowe.

2. Metody algorytmów i sygnałów pomiarowych

Platforma symulacyjna zorientowana jest na analizę błędów o charakterze losowym, które traktowane są jako zakłócenia sygnału pomiarowego. Na podstawie przeprowadzonej analizy można wyznaczyć typ szumu. Po ocenie charakterystyk szumu można określić ich model dla postaci zmiennych czasu. Znajac charakterystykę każdego szumu, można w prostszy sposób je zniwelować.

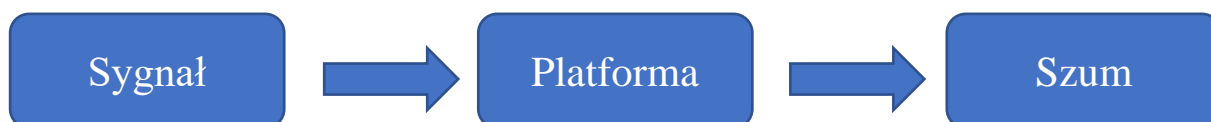
Poprawnie działająca platforma symulacyjna polega na zaprojektowanym algorytmie do reprezentacji szumu z wykorzystaniem metody wariancji Allana. Dana metoda jest najbardziej popularna w analizie typów zakłóceń sygnałów pomiarowych.

Platforma symulacyjna umożliwia sztuczną symulację ruchu czujnika żyroskopowego oraz akcelerometru. Symulację ruchu czujników inercyjnych MEMS można podzielić na następujące składniki:

- Ruch idealny żyroskopu i akcelerometru, bez zakłóceń pomiarowych
- Ruch czujników zakłócony wygenerowanym szumem białym, różowym, czerwonym, fioletowym oraz dryftem.

Ruch idealny żyroskopu lub akcelerometru jest możliwy w przypadku jego stanu spokoju (czujnik znajduje się w pozycji stacjonarnej). Warto uwzględnić, iż w pozycji stacjonarnej na czujnik oddziałują zakłócenie noszące charakter stały.

Zakłócony sygnał w środowisku idealnym otrzymujemy przez zaszumienie sygnału odpowiednim błędem pomiarowym. W przypadku symulacji uzyskanie odpowiednio dopasowanego szumu zgodnie z reprezentacją szumową według wariancji Allana, polega na wygenerowaniu szumów w sposób losowy. Najpierw generujemy szum biały, używając generatora liczb pseudolosowych. Szum biały służy tylko do reprezentacji szumu ARW /VRW. Pozostałe zakłócenia sygnału odpowiadają charakterystyce szumów kolorowych: czerwony, różowy, fioletowy. Kolorowe szumy uzyskujemy poprzez filtrację widma szumu białego. Również, platforma udostępnia możliwości do analizowania pomiarów rzeczywistych czujników inercyjnych.



Rys. 2.1 Wstępna koncepcja pracy

Elementem wejściowym pozostaje sygnał. Obiektem sterowania jest platforma symulacyjna, składająca się z algorytmów, które służą do analizy niepożądanych składników oraz symulacji. Elementem wyjściowym jest scharakteryzowanie składników szumowych występujących w sygnale.

2.1 Abstrakcyjny podział projektu

Projekt został podzielony na trzy podsystemy. W każdym podsystemie występują elementy wejściowe, wyjściowe oraz sterownik. Podział platformy symulacyjnej nosi charakter abstrakcyjny.

Tabela 2.1 Opis podsystemu symulacji

Element wejścia	Proces sterowania	Element wyjścia
Wybór odpowiedniego czujnika pomiarowego (akcelerometr lub żyroskop)	Platforma symulacyjna – algorytmy generowania szumów oraz ruchu czujnika	Ruch odpowiedniego czujnika bez zakłóceń pomiarowych
Wybór odpowiedniego szumu (biały, czerwony,...)		Ruch odpowiedniego czujnika wraz z wygenerowanymi zakłóceniami

Podsystem symulacji opiera się na wygenerowaniu idealnego środowiska i proponuje do wyboru dwie opcje: ruch idealny czujnika lub ruch zakłócony szumem wybieranym przez użytkownika. Praca tego podsystemu pozwala wygenerować zaszumiony sygnał.

Podstawowe założenia pracy podsystemu:

1. Użytkownicy mają możliwość, klikając na odpowiedni przycisk, wygenerować ruch idealnego lub zaszumionego czujnika pomiarowego
2. Użytkownik samodzielnie wprowadza częstotliwość i czas pomiaru próbki
3. Częstotliwość oraz czas pomiaru próbki domyślnie wynosi 100
4. Ilość wygenerowanych pomiarów to jest częstotliwość razy czas.

Tabela 2.2 Opis podsystemu identyfikacji

Element wejścia	Proces sterowania	Element wyjścia
Pomiary syntetyczne	Platforma symulacyjna – algorytm filtracji sygnału czujnika inercyjnego – wariancja Allana	Określenie typu szumu
Pomiary rzeczywiste		

Jak wynika z nazwy, podsystem opiera się na identyfikacji typu szumu. Praca tego podsystemu pozwala określić typy szumów występujące w sygnale. Podsystem identyfikacji szumu uruchamia się dwoma sposobami: automatycznie, gdy użytkownik używa podsystemu symulacji lub podczas analizy pomiarów rzeczywistych.

Podstawowe założenia pracy podsystemu:

1. Użytkownicy mają możliwość sprawdzić wygenerowany sygnał, z góry spodziewając się odpowiednich wyników

2. Część teoretyczna, odnośnie do wcześniej wygenerowanego losowego szumu, zostanie przedstawiona na stronie internetowej, reprezentującej platformę

3. Możliwość załadowania oraz filtracja rzeczywistych sygnałów pod względem zaszumionych składników.

Tabela 2.3 Opis podsystemu reprezentacji

Element wejścia	Proces sterowania	Element wyjścia
Odbiór informacji od środowiska Pycharm	Wyrażony przez stronę webową	Reprezentacja informacji w sposób wiarygodny dla użytkownika

Służy do wyświetlania materiału, w tym wykresy oraz ich opis. Opiera się na technologiach webowych i nie zawiera w sobie żadnych algorytmów filtracji sygnałów. Ten podsystem należy traktować jako jeden ze sposobów do prezentowania wyników.

Podstawowe założenia pracy podsystemu:

1. Możliwość zapoznania się z materiałem teoretycznym
2. Wyświetlanie wyników symulacji oraz identyfikacji
3. Prosty sposób na załadowanie pomiarów rzeczywistych czujników
4. Nadanie lub zmiana ścieżki źródłowej na komputerze użytkownika do pliku, gdzie zainstalowany jest Python „python.exe” oraz projektu „main.py”.

2.2 Moduł do generowania syntetycznych sygnałów

Pod tytułem „środowisko idealne” zostały określone pojęcia, charakteryzujące idealny ruch czujnika, czyli warunki, w których na pomiary nie wpływają żadne czynniki zewnętrzne lub wewnętrzne. W tym przypadku sygnał nie posiada żadnych niepożądanych składników. Teoretycznie w stanie nieruchomym sygnał podany od czujnika powinien pozostać niezmienny, czyli na wykresie musimy dostać linię prostą, a wykres Allana wtedy powinien być pusty. W praktyce na czujnik wciąż oddziałują różnego rodzaju zakłócenia, dlatego otrzymanie „czystego” sygnału jest niemożliwe. Został stworzony model środowiska idealnego oraz składniki niepożądane, które dodajemy pojedynczo: szum biały, różowy, czerwony, fioletowy oraz dryft czujników.

Modelowanie wykonujemy w celu porównania wykresów Allana dla symulowanych i rzeczywistych pomiarów. Każdy szum występujący w czujnikach został wygenerowany i dopasowany do ruchu idealnego. Abstrakcyjnie ten model został opisany w tabeli 2.1.

2.2.1 Ruch idealny

Przed początkiem symulacji ruchu akcelerometru lub żyroskopu w pozycji nieruchomej w idealnym środowisku należałoby uwzględnić zakłócenia stałe wpływające na ich sygnał.

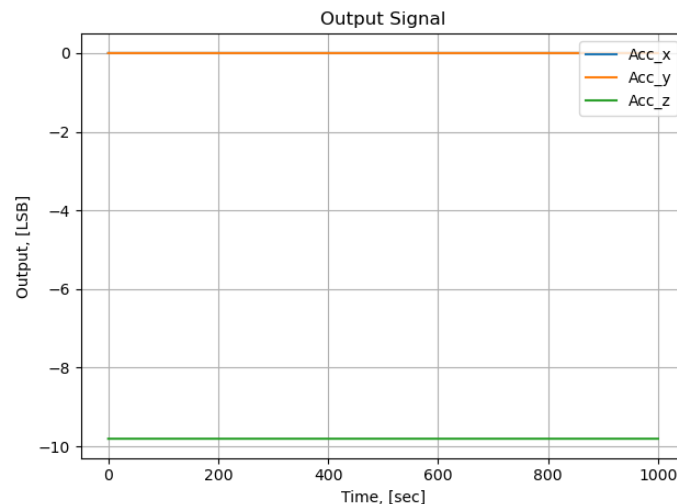
Zasada działania akcelerometru polega na wykrywaniu siły skierowanej w kierunku przeciwnym do wektora przyspieszenia. W przypadku Ziemi akcelerometr jest wrażliwy na grawitację. W trzyosiowym akcelerometrze oś Z zostanie zakłócona grawitacją. Nawet w stosunkowo stabilnym stanie akcelerometr wciąż pozostaje bardzo podatny na różne zakłócenia. Dlatego większość systemów IMU używa żyroskopu do wyrównania błędów występujących w pomiarach akcelerometru. Żyroskop jest mniej wrażliwy na czynniki zewnętrzne, ponieważ mierzy przyspieszenie kątowe, jednak tego typu czujniki posiadają problemy innego rodzaju. Na przykład brak powrotu do wartości zerowej po zatrzymaniu obrotów. Dlatego w środowisku idealnym zakładamy, że na akcelerometr wpływa tylko grawitacja, a żyroskop jest wolny od różnego rodzaju sił zewnętrznych.

Do opisanego ruchu idealnego czujnika, biorąc pod uwagę czynniki zewnętrzne występujące w sygnałach, służy wzór przedstawiony poniżej:

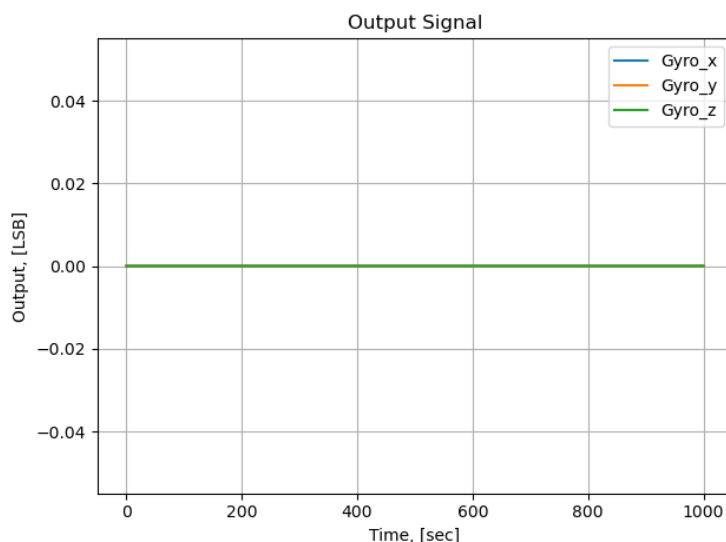
$$Out = Sensivity \times Input + Bias + Noise \quad (2.1)$$

gdzie *Out* – wektor wyjściowych sygnałów trójosiowego czujnika, *Input* – wektor mierzonej wartości, z góry ustawiony jako domyślny, *Sensitivity* – macierz 3x3 zawierająca współczynnik wrażliwości (na głównej przekątnej), *Bias* – wektor niestabilności czujnika do zakłóceń zewnętrznych, *Noise* – wektor losowych zakłóceń pomiaru. Wartość *Noise* podczas inicjalizacji posiada dwa główne znaczenia: *None* (wektor zerowy), gdy użytkownik ma na celu symulację idealnego środowiska lub wektor wartości odzwierciedlający odpowiedni szum. Wartości *Sensitivity*, *Bias* podawane są w postaci macierzy podczas inicjalizacji.

Wykres ruchu idealnego czujnika przedstawia zależność pomiarów od czasu, natomiast wykres wariancji Allana dla rysunków 2.2 i 2.3 będzie pusty.



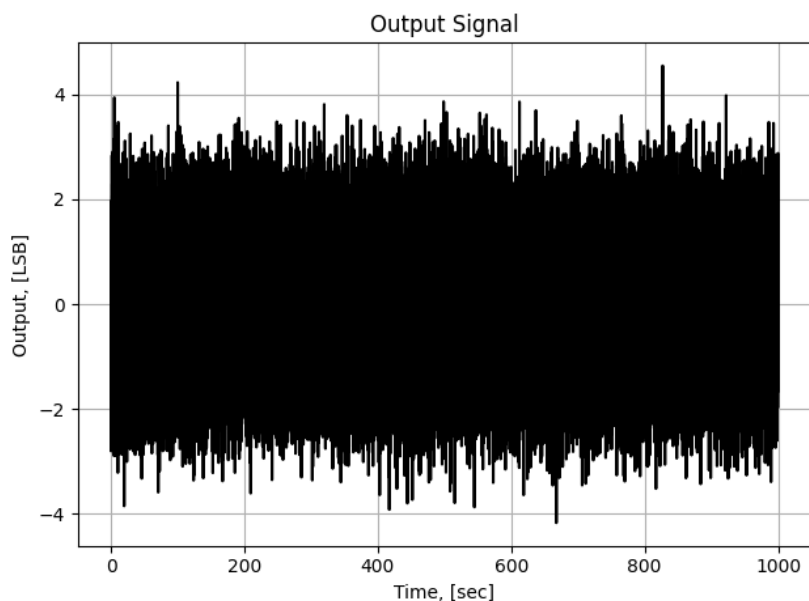
Rys. 2.2 Sygnał akcelerometru w środowisku idealnym



Rys. 2.3 Sygnał czujnika żyroskopowego w środowisku idealnym

2.2.2 Szum biały

Charakterystyka szumu białego opisuje się płaskim widmem oraz równomierną intensywnością po całym paśmie. Szczególnym przypadkiem szumu białego jest biały szum gaussowski, który całkowicie się nadaje do symulacji szumu ARW/VRW, występującego w czujnikach. Do wygenerowania tego typu szumu używamy jednego z najważniejszych rozkładów prawdopodobieństwa - rozkład Gaussa. Ilość pomiarów zależy od czasu oraz częstotliwości (krok). Używając wbudowanej funkcji pakietu NumPy **randn()**, generatora liczb pseudolosowych, otrzymujemy zakres liczb o rozkładzie normalnym. Ten typ rozkładu ze swojej charakterystyki najlepiej przedstawia biały szum, ponieważ liczby wygenerowane w taki sposób są równomiernie rozrzucone na całym paśmie i dopasowane do odchylenia oraz wariancji. Odchylenie standardowe wraz z wariancją w przypadku funkcji **randn()** ustawione domyślnie jako 1 i 0.



Rys. 2.4 Reprezentacja szumu białego

Widmo białego szumu analogicznie porównujemy do widma światła białego, które składa się z wielu innych kolorów. Dlatego na podstawie wcześniej wygenerowanego szumu białego można otrzymać szum innego koloru, używając filtrów przepustowych.

2.2.3 Szum różowy

Szum różowy (ang. *Pink noise*, *flicker noise*) charakteryzuje się procesem, w którym gęstość widmowa mocy jest odwrotnie proporcjonalna do częstotliwości. Ten rodzaj szumu wynika prawie we wszystkich materiałach i elementach stosowanych w elektronice. Cechą szumu różowego jest spadek częstotliwości w porównaniu do szumu białego, który istnieje na niskich częstotliwościach. Szum różowy charakteryzujemy jako przejściowy pomiędzy białym a czerwonym.

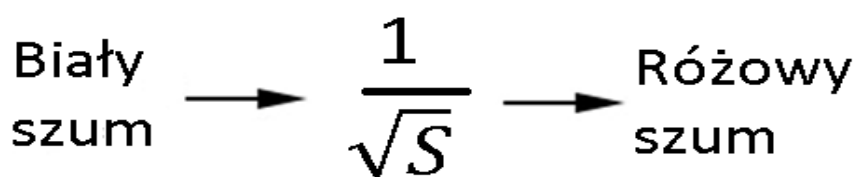
Idea stworzenia odpowiedniego filtra szumu czerwonego jest następująca: jeżeli biały szum $\omega(t)$ z widmową gęstością mocy $S(\omega)=S_0$ całkujemy m razy, wtedy powstanie proces losowy z gęstością widmową:

$$S_{\omega} = \frac{1}{\omega^{2m}} S_0 \quad (2.2)$$

Założymy że $2m = 1$:

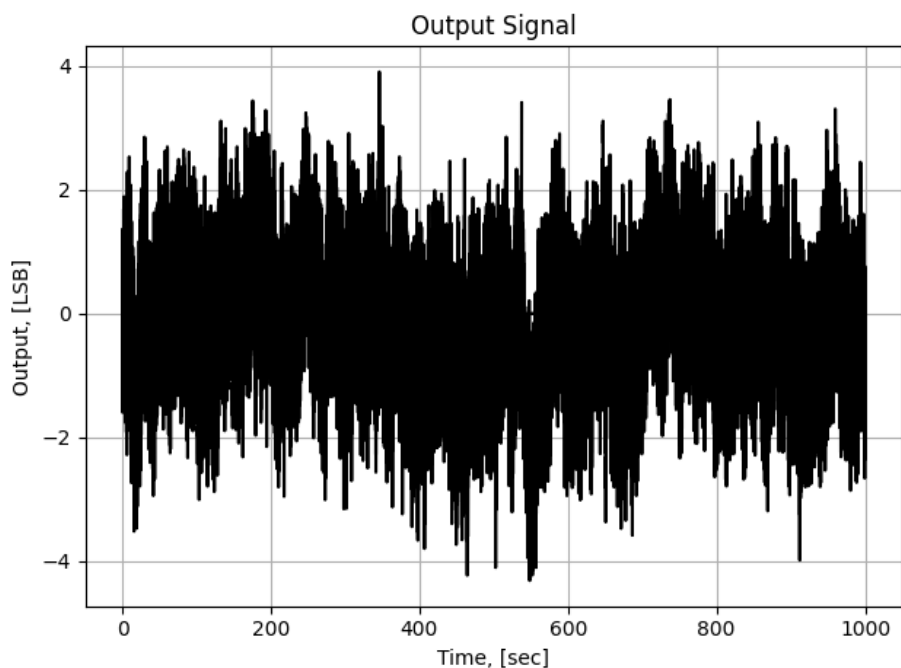
$$S\omega^{1/2} = \frac{1}{\omega} \quad (2.3)$$

wtedy otrzymamy wymagane widmo odwrotnie proporcjonalne do częstotliwości. Do otrzymanych wartości stosujemy transmitancji operatorowej i na wyjściu uzyskujemy szum o widmie $1/f$:



Rys. 2.5 Filtr szumu różowego [13].

Widmo szumu różowego dla czujników pomiarowych charakteryzuje się mniejszą częstotliwością oraz rozrzutem pomiarów.

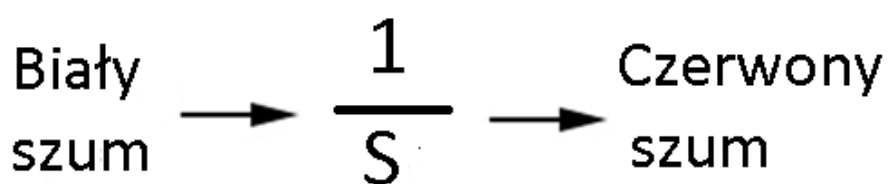


Rys. 2.6 Reprezentacja szumu różowego

W czujnikach inercyjnych szum różowy reprezentowany jest jako szum BI. W porównaniu do wykresów szumu białego intensywność tego szumu jest niższa.

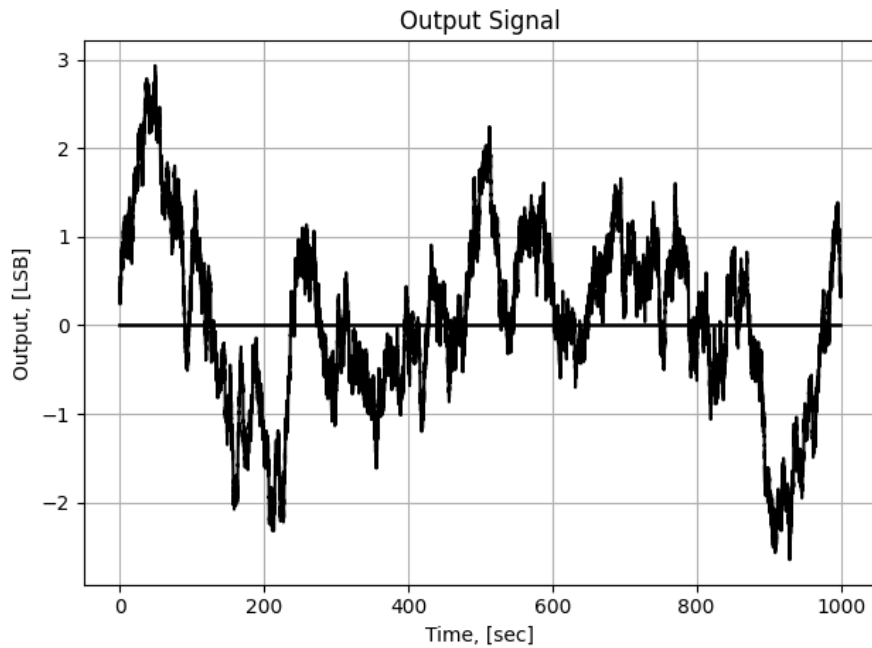
2.2.4 Szum czerwony

Znany również jako ruch Browna (and. *Red noise, brownian noise*). Charakteryzuje się procesem, w którym widmo jest w zakresie niskich częstotliwości, jeszcze niższych w porównaniu do szumu różowego. Widmo takiego szumu opada w miarę wzrostu częstotliwości i jest odwrotnie proporcjonalne do kwadratu częstotliwości $1/f^2$. Szum czerwony powstaje w wyniku całkowania szumu białego $1/s$, gdzie s – argument w transformacji Laplace’a:



Rys. 2.7 Filtr szumu czerwonego [13].

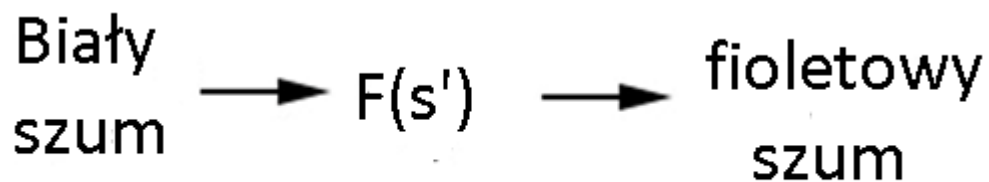
Wykres zależności czasu od wartości wyjściowej szumu czerwonego czujnika pomiarowego wygląda w sposób następujący:



Rys. 2.8 Reprezentacja szumu czerwonego

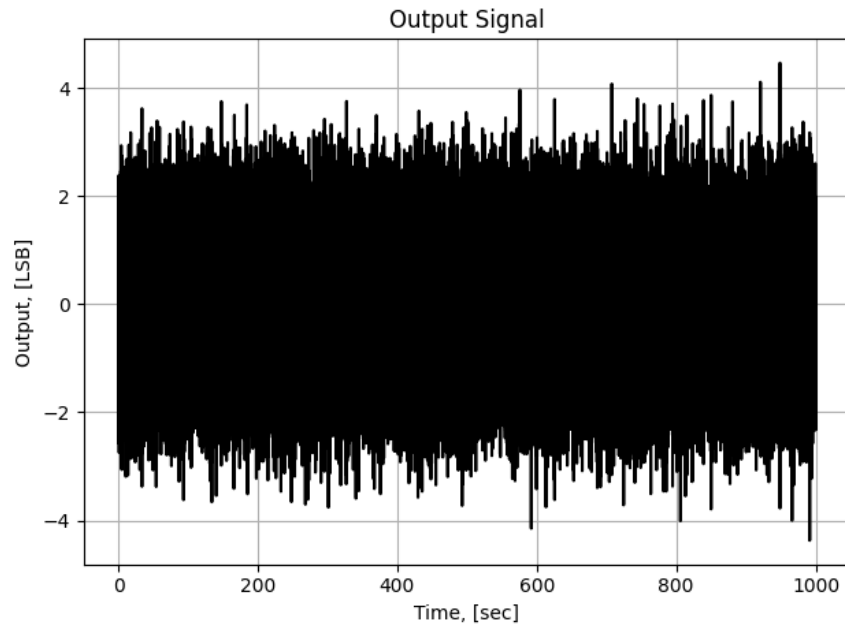
2.2.5 Szum fioletowy

Szum fioletowy (ang. *Violet noise*) – charakteryzuje się procesem, w którym widmo jest w zakresie wysokich częstotliwości. Wraz ze wzrostem częstotliwości gęstość widmowa również wzrasta. Szum fioletowy powstaje w wyniku różnicowania szumu białego.



Rys. 2.9 Filtr szumu fioletowego [13]

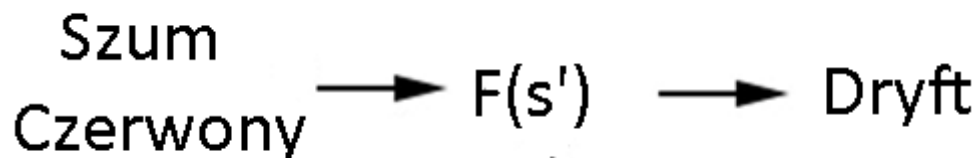
Ten typ szumu charakteryzuje się podobną intensywnością do szumu białego.



Rys. 2.10 Reprezentacja szumu fioletowego

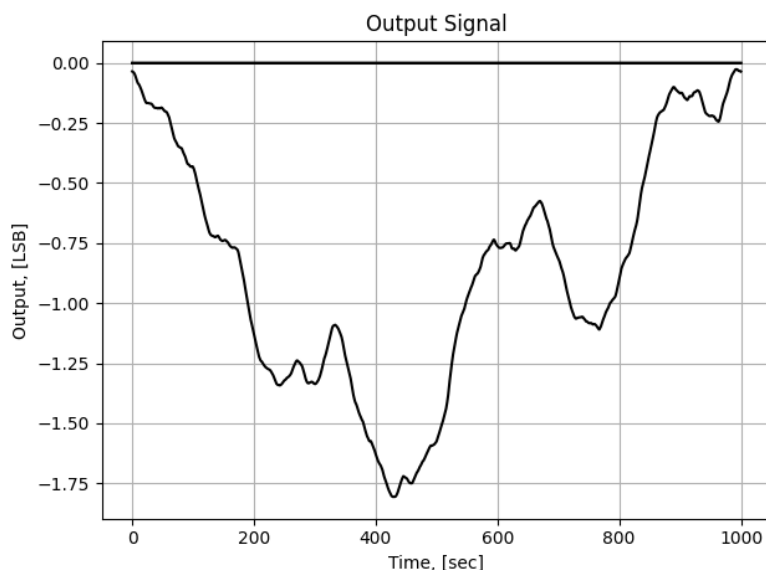
2.2.6 Dryft

W przypadku wariancji Allana dryft określa szum RR występujący w czujnikach. Ten rodzaj szumu pojawia się podczas monotonicznej zmiany sygnału na długim okresie czasowym. Aby wygenerować dryft czujników, należy zróżniczkować szum czerwony:



Rys 2.11 Filtr dryftu czujników pomiarowych [13].

Wykres zależności czasu od wartości wyjściowej dryftu czujników pomiarowych wygląda w sposób następujący:



Rys. 2.12 Reprezentacja dryftu

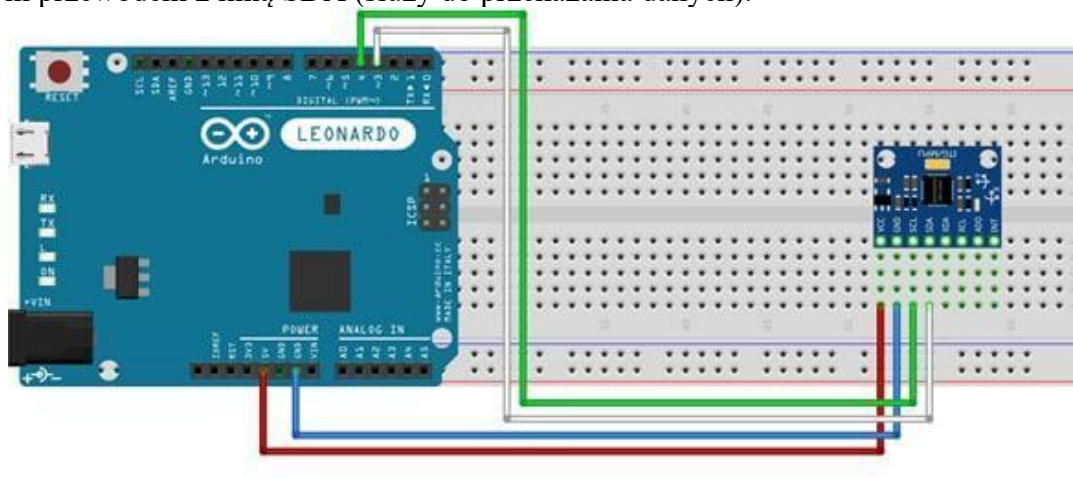
2.3 Platforma pomiarowa

Identyfikacja szumu zaczyna się od pomiarów syntetycznych lub rzeczywistych (tabela 2.2). Pomiary rzeczywiste zostały wygenerowane, wykorzystując następujące technologie pomiarowe: platforma Arduino oraz czujnik MPU6050, w który są wbudowane trzyosiowy czujnik przyspieszenia (akcelerometr) oraz trzyosiowy czujnik prędkości kątowej (żyroskop).

Żeby móc używać czujnika MPU6050, musimy posiadać:

- Platformę Arduino
- Płytkę prototypową
- Moduł GY-521(MPU6050)
- Przewody.

Podłączamy zgodnie ze schematem przedstawionym na rysunku poniżej. Czerwony przewód podaje napięcie 5V, a niebieski służy do uziemienia (GND). Zielony przewód łączy cyfrowy pin 4 platformy Arduino z linią SCL czujnika (linia czasowa) oraz pin 3 połączony białym przewodem z linią SDA (służy do przekazania danych).



Rys. 2.13 Schemat układu pomiarowego

Czujnik przyspieszenia działa w zakresach: $\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$. W przypadku żyroskopu będą to: $250^\circ/s$, $500^\circ/s$, $1000^\circ/s$, $2500^\circ/s$. Układ wyposażony w 16-bitowy przetwornik ADC (ang. *Analog to digital converter*). Napięcia podawane do czujnika to 3,3-5,0 V.

Różne wersje modułu GY-521 należą do kategorii cyfrowych lub analogowych. Cyfrowe moduły dostarczają informację za pomocą protokołu szeregowego jak I2C, SPI lub USART. Analogowe wprowadzają poziom napięcia z zakresu, który należy przekonwertować na wartość cyfrową za pomocą przetwornika analogowo-cyfrowego ADC. Zakres danych wyjściowych otrzymanych od czujnika zależy od modułu ADC. Na przykład 10-bitowy ADC wyprowadzi wartość z zakresu 0.....1023; 12-bitowy 0.....4095.

W niniejszej pracy został użyty czujnik o 16-bitowym ADC, a więc jego zakres wprowadzonych danych wynosi $[-32768, +32768]$ (2^{16}). Dane wyświetlane w formie „surowej” odzwierciedlają poziom napięć czujników, ADC przetwarza je na wartości bezwymiarowe. Do obliczenia przyspieszenia oraz prędkości kątowej służą specjalne wzory matematyczne, filtry oraz połączenia czujników akcelerometru i żyroskopu w celu wyrównania układów. Powyżej wymienione sposoby dają możliwość dokładnie wyznaczyć prędkość oraz przyspieszenie czujnika. Jednak do analizy szumowej początkowe „surowe” pomiary należy przekonwertować. Otrzymane wartości (optymalny czas trwania to 100s) zapisujemy w pliku *.csv, który później można załadować bezpośrednio do platformy.

2.4 Wariancja Allana

Wariancja Allana jest bazowym algorytmem, posługując się którym możemy analizować składniki szumowe. Po raz pierwszy został wprowadzony do badania stabilności zegarów i oscylatorów. Bazuje się na dwupróbkowej statystyce, która wyznaczana jest jako odchylenie standardowe różnic wartości średnich wielkości mierzonych w sąsiadujących ze sobą przedziałach czasowych [5].

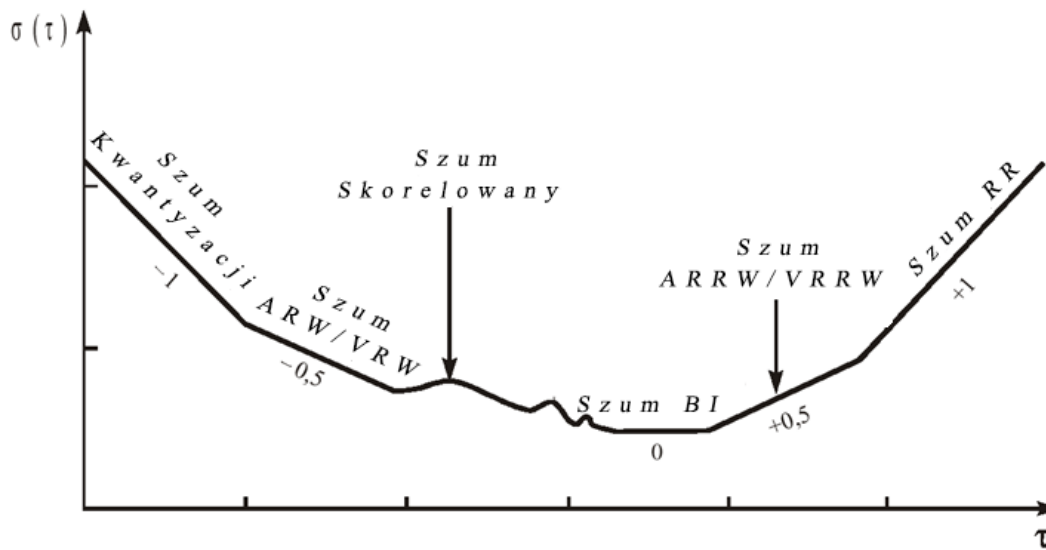
Żeby poprawnie użyć tej metody, obliczyć ją zgodnie z instrukcją oraz narysować wykres odchylenia Allana, musimy podzielić sygnał na równe próbki o jednakowej długości τ . Dla każdej próbki należy znaleźć wartość uśrednioną. Obliczanie składników szumowych odbywa się według wzoru:

$$\sigma_A^2(\tau) = \frac{1}{2\tau^2(L - 2l + 1)} \sum_{n=0}^{L-2l} \left(\sum_{i=1}^l (\Delta N(t_{n+i}) - \Delta N(t_{n+l+i})) \right)^2 \quad (2.4)$$

gdzie $\tau = l \cdot \Delta t$ – przedział uśrednienia, Δt - przedział sygnału wyjściowego czujnika, l – ilość wartości sygnału w przedziale uśrednienia, L – ogólna ilość wartości sygnału wyjściowego podczas jednostkowego uruchomienia, $\Delta N(t_i)$ – wzrost wielkości sygnału. W wyniku działania algorytmu otrzymujemy macierz zawierającą macierz τ – wielkość przedziału uśrednienia oraz $\sigma_A^2(\tau)$ macierz wartości wariancji Allana zgodnie z wartościami τ .

Do budowy wykresu należy obliczyć odchylenie Allana, czyli pierwiastek z wariancji. Otrzymany wykres w skali *log-log* przedstawia zależność odchylenia Allana od czasu uśrednienia. Dany typ wykresu wskazuje na rodzaj losowych procesów występujących na wyjściu sygnału.

Różne typy niepożądanych składników występujące na wykresie Allana powodują różne procesy. Szumy posiadają różne kąty nachylenia, co pozwala w łatwy sposób je zidentyfikować. W przypadku urządzeń MEMS ważnymi losowymi procesami są: szum kwantyzacji, szum ARW i VRW, szum BI, szum RRW oraz szum RR. Poniższy rysunek przedstawia każdy rodzaj zakłócenia oraz jego charakterystykę:



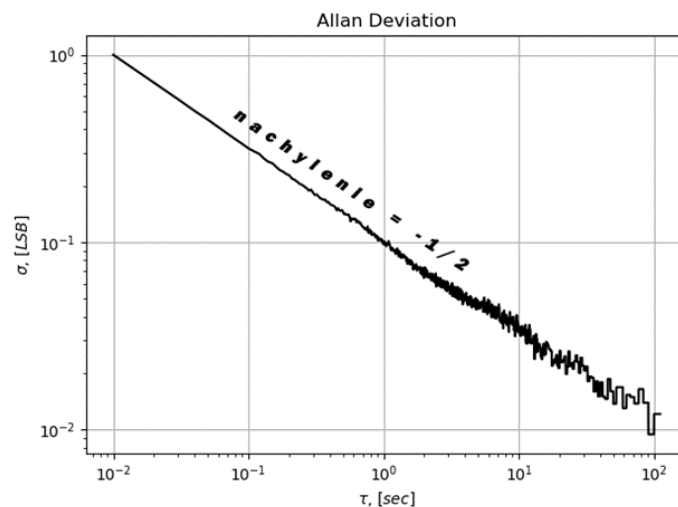
Rys. 2.14 Ilustracja błędów opisanych wariancją Allana

2.4.1 Szum ARW /VRW

Szum ARW (ang. *Angular Random Walk*) pojawia się w czujnikach żyroskopowych oraz szum VRW (ang. *Velocity Random Walk*) – akcelerometrach. Jest szumem o dominowaniu składników wysokoczęstotliwościowych, co oznacza, że czas korelacji tych szumów jest bardzo krótki. Ten typ szumu powoduje losowe zakłócenia kąta z rozkładem proporcjonalnym do pierwiastka kwadratowego z czasu pierwszego.

$$\sigma^2(\tau) = \frac{Q^2}{\tau} \quad (2.5)$$

Na wykresie logarymicznym odchylenia Allana szum ARW/VRW jest reprezentowany przez nachylenie prostej o współczynniku $-1/2$. Wartość współczynnika $\sigma(\tau)$ obliczamy w punkcie $\tau = 1$:



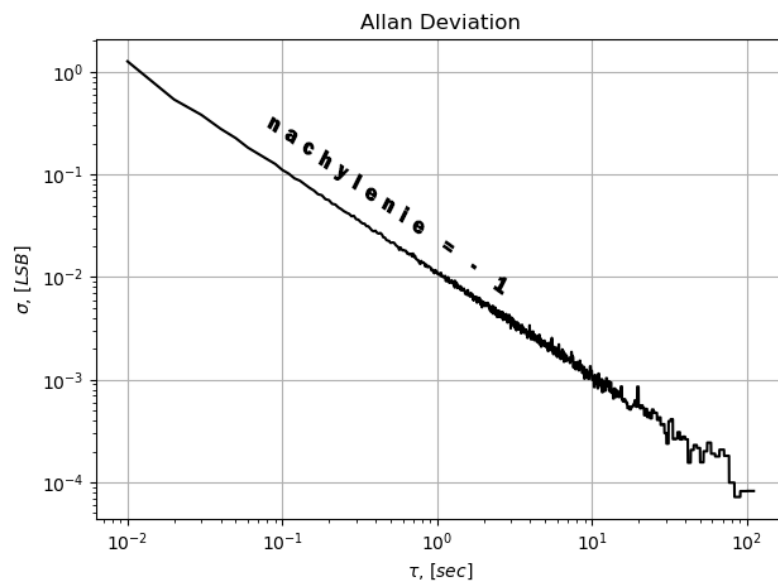
Rys. 2.15 Szum ARW/VRW

2.4.2 Szum kwantyzacji

Szum kwantyzacji (ang. *Quantization Noise*) spowodowany różnicą między rzeczywistymi amplitudami próbkowania punktów a rozdzielczością przetwornika analogowo-cyfrowego. Dany rodzaj szumu związany jest z operacją kwantyzacji sygnału pomiarowego przetwornika A/C czujników pomiarowych.

$$\sigma^2(\tau) = \frac{3Q^2}{\tau^2} \quad (2.6)$$

Na wykresie logarytmicznym odchylenia Allana szum kwantyzacji jest reprezentowany przez nachylenie prostej o współczynniku -1. Wartość współczynnika $\sigma(\tau)$ obliczamy w punkcie $\tau = \sqrt{3}$:

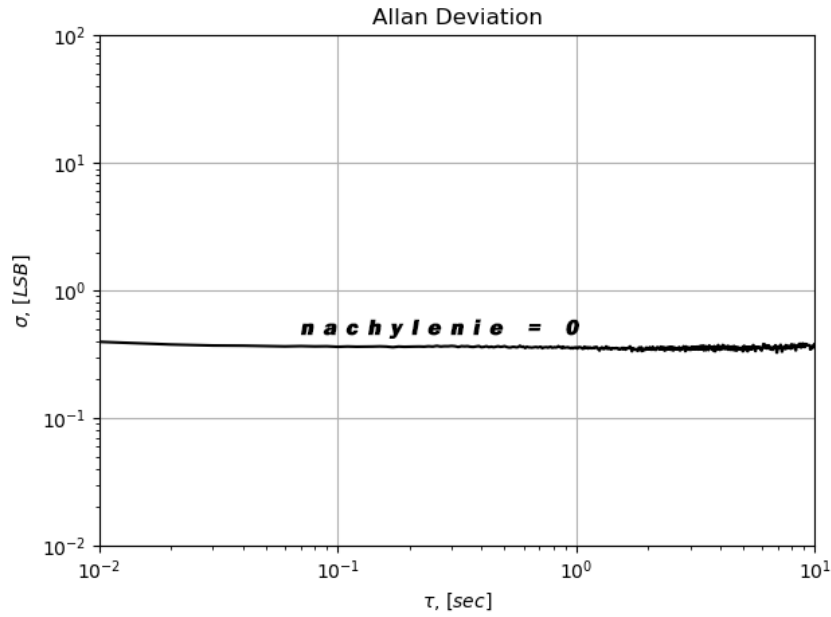


Rys. 2.16 Szum kwantyzacji

2.4.3 Szum BI

Szum BI (ang. *Bias Instability*) jest spowodowany przez szum części elektronicznej oraz drgania wywołane przez inne elementy wchodzące do czujników pomiarowych. Szum ten charakteryzuje się na niskich częstotliwościach, które przyczyniają się do powstania fluktuacji sygnału. W przybliżeniu wariancja szumu BI wynosi:

$$\sigma^2(\tau) = \frac{B^2}{0.6648^2}, \tau \gg 1/f_0 \quad (2.7)$$



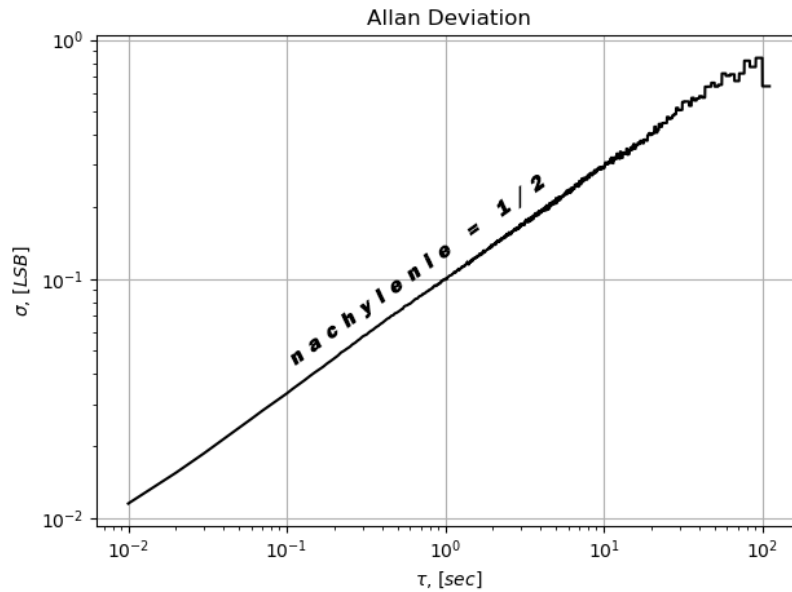
Rys. 2.17 Szum BI

2.4.4 Szum RRW

W szumie RRW (ang. *Rate Random Walk*) przeważają składowe o niskich częstotliwościach. W czujnikach inercyjnych wpływa on na pomiary przyspieszenia (ang. *Velocity Rate Random Walk*, VRRW) lub prędkości kątowej (ang. *Angular Rate Random Walk*, ARRW). Wariancja Allana dla tego wyznaczana jest według wzoru:

$$\sigma^2(\tau) = \frac{K^3}{3} \tau \quad (2.8)$$

gdzie K – współczynnik szumu RRW. Na wykresie logarytmicznym odchylenia Allana szum RRW jest reprezentowany przez nachylenie prostej o współczynniku $1/2$. Wartość współczynnika K obliczamy według wzoru (2.8) w punkcie $\tau = 3$.



Rys. 2.18 Szum RRW

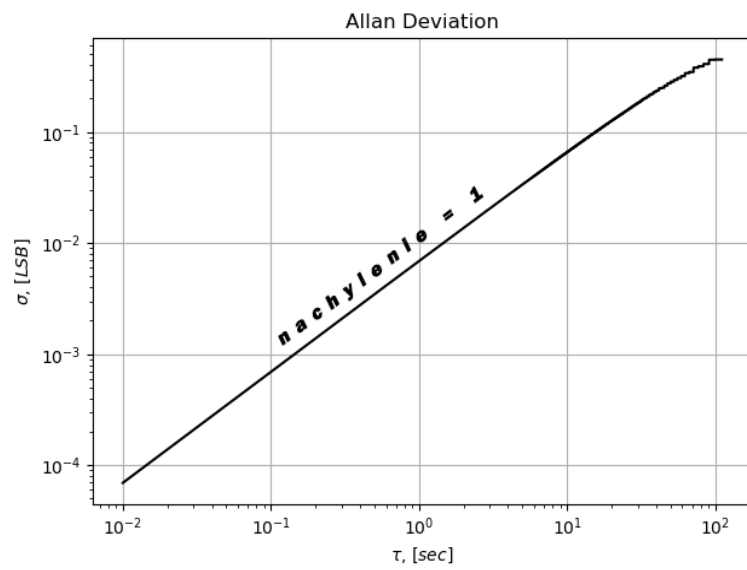
2.4.5 Szum RR

Szum RR (ang. *Rate Ramp*) opisuje zakłócenia sygnału czujnika pomiarowego związanego z błędem systematycznym wykonanych pomiarów. Jedną z przyczyn pojawienia tego typu szumu może być związana z niedokładnością pomiaru czujnika lub monotoniczną zmianą sygnału w czasie. Wariancję Allana obliczamy posługując się wzorem:

$$\sigma^2(\tau) = \frac{R^2 \tau^2}{2} \quad (2.9)$$

gdzie R to współczynnik dryftu.

Na wykresie logarytmicznym odchylenia Allana szum RR jest reprezentowany przez nachylenie prostej o współczynniku 1. Wartość współczynnika R obliczamy według wzoru w punkcie $\tau = \sqrt{2}$



Rys. 2.19 Szum RR

3. Projekt platformy symulacyjnej

Platforma symulacyjna została podzielona na dwie części. W środowisku Pycharm znajduje się główny skrypt służący do analizy szumowej oraz modelowania zakłóceń pomiarowych. Strona internetowa pełni funkcję reprezentacyjną. Przedstawia teorię, wykresy symulacji oraz wariancji Allana.

W tym rozdziale została przedstawiona analiza wymagań funkcjonalnych i niefunkcjonalnych oraz architektura.

3.1 Analiza wymagań funkcjonalnych i niefunkcjonalnych platformy symulacyjnej

Przeprowadzenie analizy wymagań jest niezbędne do określenia oczekiwań od rozwiązań i dokonanie właściwego wyboru. Sporządzenie odpowiedniej analizy polega na udokumentowaniu potrzeb określonego produktu lub usługi.

3.1.1 Analiza wymagań funkcjonalnych

Wymagania funkcjonalne definiują zbiór wszystkich wymagań, które do tej pory wymieniane były na etapie planowania i projektowania pracy. Opis wymagań został podany w formie zapytań użytkownika. Zapytania, rozumiane jako potrzeby użytkownika, wymagają odpowiadania w konkretny sposób i w prostej formie, według której musi działać projektowany system.

Pierwszym etapem jest wybór osób, które będą używać systemu. Głównie są to wykładowcy, inżynierowie lub ewentualnie mogą to być studenci. Jest to grupa docelowa zdefiniowana jako **użytkownik**. Dalej – zgodnie z potrzebami użytkownika, projektowany system powinien oferować następujące funkcje:

- | | |
|-------------|--|
| HU1 | Jako użytkownik <i>chcę</i> zapoznać się z podstawowym materiałem teoretycznym na stronie webowej. |
| HU2 | Jako użytkownik <i>chcę</i> dokładnie poczytać o czujnikach inercyjnych, a mianowicie o żyroskopie oraz akcelerometrze. |
| HU3 | Jako użytkownik <i>chcę</i> dokładnie zapoznać się z wariancją Allana. |
| HU4 | Jako użytkownik <i>chcę</i> mieć możliwość wygenerowania wykresu symulacji ruchu czujników w warunkach idealnych. |
| HU5 | Jako użytkownik <i>chcę</i> wygenerować szum biały w warunkach idealnych. |
| HU6 | Jako użytkownik <i>chcę</i> wygenerować szumy kolorowe w warunkach idealnych. |
| HU7 | Jako użytkownik <i>chcę</i> zobaczyć szczegółowy opis wykresów otrzymanych w wyniku symulacji w warunkach idealnych. |
| HU8 | Jako użytkownik <i>chcę</i> zbadać pomiary otrzymane w sposób eksperymentalny na zawartość składników szumowych. |
| HU9 | Jako użytkownik <i>chcę</i> pobrać wcześniej wygenerowane wykresy. |
| HU10 | Jako użytkownik <i>chcę</i> mieć możliwość oczyszczania wcześniej wygenerowanych wykresów podczas symulacji. |

Oprócz tego wymagania funkcjonalne opisują poniższe przypadki użycia, czyli podstawowy przebieg operacji, tzw. szczęśliwą ścieżkę wydarzeń. Przypadki użycia są dostosowane do zapotrzebowania użytkowników (HU1 – HU10).

Przypadek użycia (generowanie szumu białego w warunkach idealnych):

1. System domyślnie otwiera stronę tytułową
2. Użytkownik klika „*Symulacja*”
3. Wybiera szum biały, czas trwania oraz częstotliwość
4. System zwraca teoretyczny opis danej opcji oraz wykres.

Drugi scenariusz użycia (analizowanie pomiarów własnych):

1. System domyślnie otwiera stronę tytułową
2. Użytkownik wybiera opcję „*Dane rzeczywiste*”
3. W danych rzeczywistych ładujemy plik z danymi, klikając opcję „*Załaduj*”
4. Po przeprowadzeniu symulacji użytkownik otrzymuje wykresy.

Trzeci scenariusz użycia (pierwsze połączenie strony internetowej ze skryptami Python):

1. Domyślnie otwierana strona tytułowa
2. Użytkownik klika opcję „*Ustawienia*”
3. W ustawieniach należy podać ścieżkę do folderu z Pythonem plik **python.exe** oraz właściwego projektu „*imu_simulation_master*” plik **main.py**.

3.1.2 Analiza wymagań niefunkcjonalnych

Zbiór wymagań niefunkcjonalnych dotyczy dodatkowych wymagań, które domyślnie muszą być spełnione, oprócz wymagań funkcjonalnych. Analiza wymagań niefunkcjonalnych dotyczy następujących punktów: dostępności, poprawności, szybkości, bezpieczeństwa, wydajności oraz ergonomii oraz intuitywności:

I. Dostępność

Platforma symulacyjna ma charakter informacyjny i powinna być dostępna przez 24 godziny na dobę.

II. Poprawność

W celu sprawdzania poprawności wykonywanych obliczeń istnieje możliwość porównania pomiarów rzeczywistych z pomiarami otrzymanymi, używając funkcji „pythonowskich”. Główna wada platformy polega na tym, iż nie jest ona w stanie samodzielnie sporządzić analizy szumowej na otrzymanym wykresie pomiarów rzeczywistych. Analizę szumową należy przedstawić użytkownikowi na podstawie wcześniej przeczytanego materiału teoretycznego, umieszczonego na platformie symulacyjnej oraz otrzymanego wykresu. W przypadku pomiarów rzeczywistych platforma symulacyjna zwraca tylko końcowy wykres odchylenia Allana.

III. Szybkość

Szybkość obliczeń zależy od ogólnej liczby pomiarów. W przypadku tworzenia generatorem liczb pseudolosowych szumów, szybkość działania zależy od dwóch zmiennych: czasu oraz częstotliwości, ponieważ one definiują ogólną liczbę pomiarów, którą należy wylosować. W celu skrócenia czasu oczekiwania kod źródłowy został zoptymalizowany zgodnie z zasadami programowania obiektowego.

IV. Bezpieczeństwo

Funkcje użyte w platformie są dostępne bez konieczności logowania. Użytkownik nie musi podawać danych prywatnych, dlatego pytanie o bezpieczeństwie jest nieaktualne. W razie ewentualnego rozwoju danej platformy oraz umieszczenia jej na serwerze platforma zostanie zabezpieczona już od strony serwera (szyfrowana wersja protokołu http itp.).

V. Wydajność

Projekt platformy został zaprojektowany głównie dla pojedynczego użytkownika, chociaż istnieje możliwość zainstalowania go na serwerze i wtedy wydajność będzie zależała od możliwości serwera.

VI. Ergonomia i intuicyjność

Strona internetowa została zaprojektowana w sposób intuicyjny. Wykorzystanie każdej funkcji wymaga jak najmniejszej liczby działań ze strony użytkownika.

3.2 Architektura platformy symulacyjnej

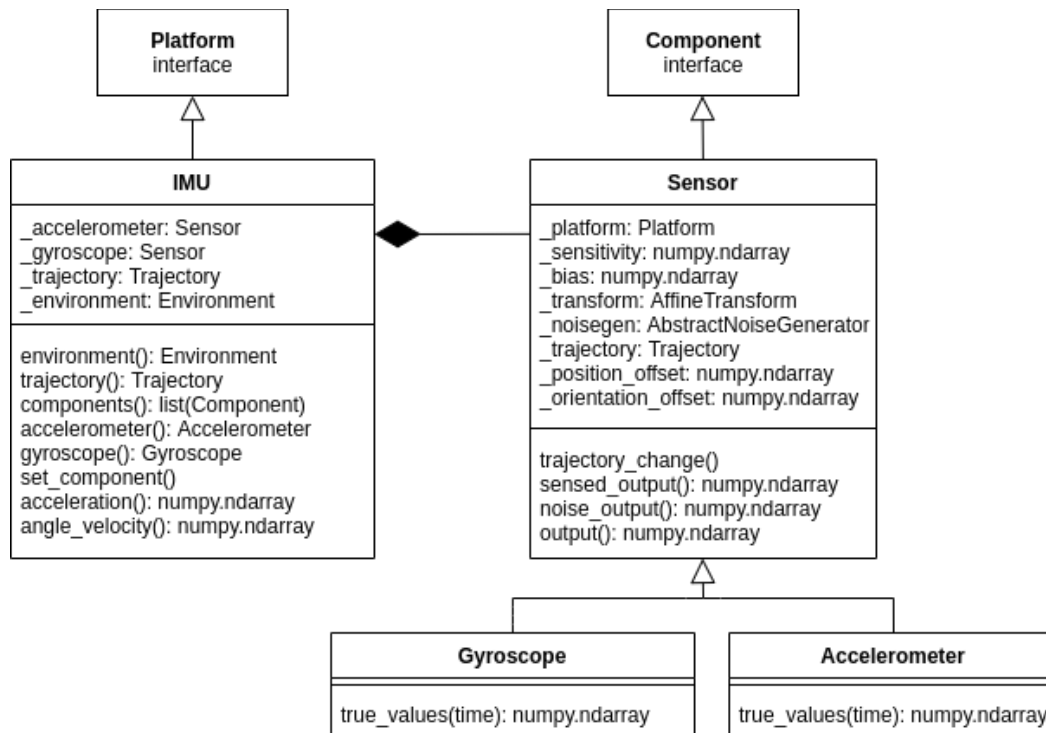
Część implementacyjna platformy symulacyjnej napisana jest w języku Python i zawiera w sobie dwa główne moduły:

- **Imu_model** – moduł, służący do symulacji czujników inercyjnych wchodzących w skład inercyjnej jednostki pomiarowej. W tym module można zrealizować symulację ruchu idealnego lub zakłóconego czujnika zgodnie z wybraną trajektorią. Dodatkowo istnieje możliwość importowania rzeczywistych pomiarów czujników.
- **Data_processing** – moduł z narzędziami do przetwarzania wyników otrzymanych od czujników typu IMU. Jednym z takich narzędzi jest wariancja Allana.

3.2.1 Architektura modułu IMU

Cała struktura danego modułu bazuje na dwóch abstrakcyjnych klasach **Platform** oraz **Component**. Klasa **Platform** określa interfejs do pracy z modelem obiektu fizycznego, który porusza się po zadanej trajektorii (określa się przez obiekt klasy **Trajectory**) w określonym środowisku (zadaje się przez obiekt klasy **Environment**). Klasa **Component** wyznacza interfejs do pracy z modelami fizycznymi komponentów obiektów, które należą do klasy **Platform**.

Zgodnie z zasadą narzuconą przez jednostki strukturalne programu opisanego powyżej, klasa **IMU**, określająca model inercyjnej jednostki pomiarowej, imituje interfejs klasy **Platform**. Klasy, które definiują modele czujników **Sensor**, wzorują się odpowiednio do interfejsu **Component**. Opisane z góry zależności są przedstawione na poniższym diagramie UML:



Rys. 3.1 UML – schemat budowy modułu **imu_model**

Klasa **Trajectory** jest finalna dla obu abstrakcyjnych klas **Rotation_Trajectory** oraz **Position_Trajectory**, które odpowiednio kierują interfejs do opisywania ruchu obrotowego oraz translacyjnego obiektu klasy **Platform**. Plik „*imu_model/trajectories.py*” posiada przykład trajektorii **Static_Trajectory** (klasa symulująca statyczne położenia ciała w przestrzeni).

Klasa **Environment** opisuje fizyczny model zewnętrznego środowiska. Minimalne wymaganie do danej klasy to definicja modelu przyspieszenia grawitacyjnego jako stałe pole wektorowe z modułem $g=9.81 \text{ m/s}^2$.

Plik „*imu_model/noise.py*” implementuje mechanizm generowania składników szumowych sygnału wyjściowego czujników klasy **Sensor** w postaci szumu kolorowego.

Plik „*imu_model/sensor.py*” zawiera implementację podstawowej klasy **Sensor** trójosiowych wektorowych czujników i również klasy finalnych **Accelerometr** oraz **Gyroscope**. Obie klasy służą do opisywania modelu trójosiowego czujnika. Podczas inicjalizacji obiektów klasy **Sensor** nawiązuje do wzoru (2.1). *Bias* podawany w postaci NumPy-macierzy. *Noise* transferowany jako obiekt-generator z interfejsem **Abstract_Noise_Generator** lub *None*, służy do symulacji idealnego ruchu czujnika.

Klasa **Sensor** posiada następujące metody:

- *True_values(self, time)* – zwraca wartość mierzonej wielkości wektorowej w momentach czasu *time*, wyznaczone przez trajektorię ruchu platformy, gdzie dany czujnik został zainstalowany. Dana metoda jest abstrakcyjna, a jej realizacja została dokładnie omówiona w opisie określonej klasy czujników (na przykład, **Accelerometr** lub **Gyroscope**).
- *Sensed_output(self, time)* – zwraca wartość wyjściowego sygnału czujnika w określonych momentach czasu *time*. Faktycznie wartości obliczone (2.1) bez brania pod uwagę *Noise* oraz *Input*, otrzymano je podczas wywoływania metody *self.true_values(time)*.

- *Noise_output(self, time)* – zwraca macierz wartości szumów sygnału wyjściowego czujnika w określonych momentach czasu *time*, które zostały zdegenerowane (w przypadku symulacji „idealnego” czujnika) używając definiowanego podczas inicjalizacji generatora szumów z interfejsem **Abstract_Noise_Generator**.

- *Output(self, time)* – zwraca wartości sygnału wyjściowego czujnika w określonych momentach czasowych *time*. Faktycznie w przypadku z symulacją czujnika wyjście jest sumą wyników *self.true_values(time)* oraz *self.noise_output(time)*.

Oprócz tego finalna klasa posiada metodę *__call__*, która zwraca wartości sygnału wyjściowego czujnika, transformowane do jednostek obliczeniowych posługując się wzorem:

$$Result = Sensitivity^{-1}(Out - Bias) \quad (3.1)$$

Finalna klasa **Accelerometr**, pochodząca od **Sensor**, w metodzie *true_values(self, time)* oblicza różnice między bezwzględnym przyspieszeniem platformy oraz przyspieszeniem grawitacyjnym, które zostało zdefiniowane przez środowisko przedstawione jako obiekt klasy **Environment**.

Plik *imu_model/imu.py* zawiera inicjalizację klasy **IMU** – modeli inercyjnego czujnika. Podczas inicjalizacji obiektu danej klasy niezbędne jest zdefiniowanie następujących obiektów:

- *Accelerometr* – obiekt klasy **Sensor**, symuluje ruch akcelerometru. W przypadku gdy wartość wejściowa ma znaczenie *None*, zostaje stworzony obiekt modeli idealnego akcelerometru (obiekt klasy **Accelerometr** bez generowania szumu).

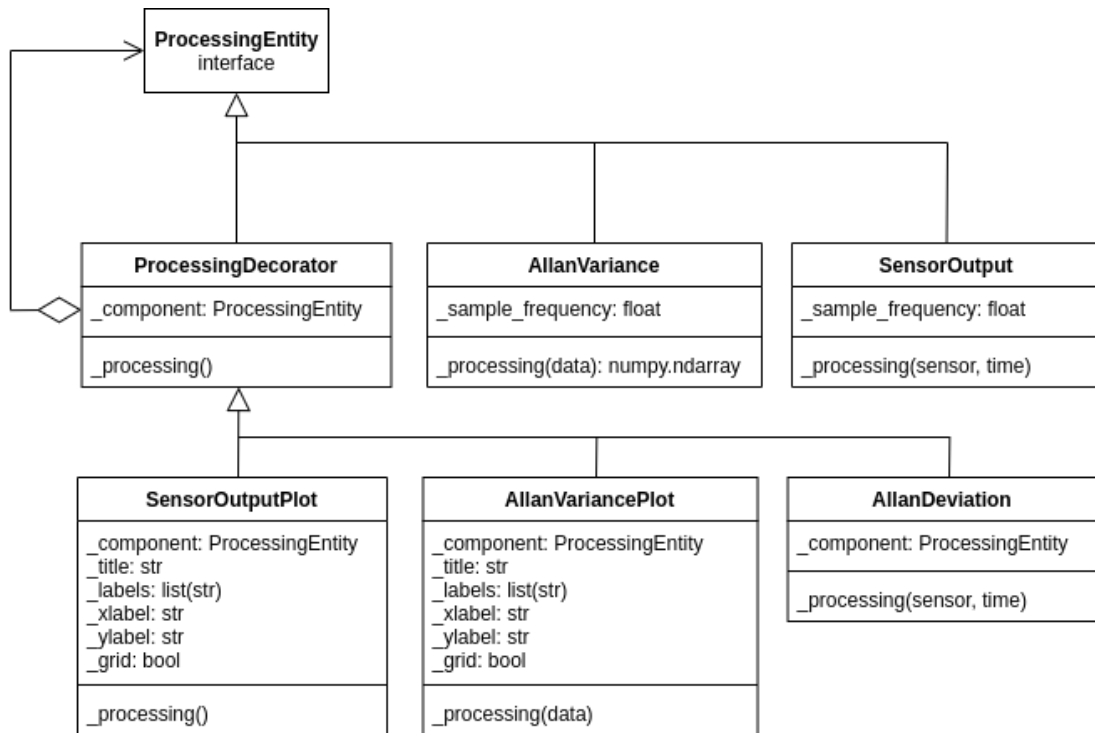
- *Gyroscope* – obiekt klasy **Sensor**, symuluje ruch żyroskopu. W przypadku gdy wartość wejściowa ma znaczenie *None*, zostaje stworzony obiekt idealnego żyroskopu (obiekt klasy **Gyroscope** bez generatora szumu).

- *Trajectory* – obiekt symulujący ruch trajektorii. Domyślnie pozostaje stworzony obiekt klasy **Static_Trajectory**, która odpowiada za stacjonarne położenia ciała.

- *Environment* – obiekt klasy **Environment**. Zaleca się użycie wartości domyślnej, która implementuje stałe przyspieszenie grawitacyjne.

3.2.2 Architektura modułu *data_processing*

Dane modułu zostały zaprojektowane na bazie dwóch abstrakcyjnych klas: **Processing_Entity** oraz **Processing_Decorator**. **Processing_Entity** definiuje interfejs do pracy z narzędziami przetwarzania danych pochodzących z symulowanych obiektów. **Processing_Decorator** jest interfejsem dla dodatkowych narzędzi używanych w połączeniu z podstawowym algorytmem przetwarzania danych (na przykład budowanie wykresów, obliczenia statycznych charakterystyk itp.) Dana klasa współdziała z klasą bazową **Processing_Entity** (rysunek 3.1).



Rys. 3.2 UML – schemat modułu **data_processing**

Plik „*data_processing/allan_variance.py*” zawiera implementację klasy **Allan_Variance** i służy do obliczenia wariancji Allana. Do tworzenia wykresów wariancji Allana w skali logarytmicznej stosowano klasy **Allan_Variance_Plot**.

Podczas inicjalizacji obiektu klasy **Allan_Variance** niezbędnym jest wyznaczenie częstotliwości próbkowania danych wejściowych. Metoda `_processing(self, data)` implementuje algorytm klasyczny obliczania wariancji Allana według wzoru.

Zaleca się wykonanie metody `_processing(self, data)` poprzez metodę `__call__` danej klasy, ponieważ wtedy zrealizowana zostaje możliwość przetwarzania danych wielu urządzeń pomiarowych.

Podczas inicjalizacji obiektów klasy **Allan_Variance_Plot** warto określić następujące parametry:

- *Processing_entity* – obiekt z interfejsem **Processing_Entity**
- *Title* – nazwa wykresu
- *Labels* – lista nazw wszystkich linii, które zostały przedstawione na wykresie
- *Xlabel* – oś odciętej wykresu
- *Ylabel* – oś rzędnych wykresu
- *Grid* – wyświetlanie siatki (True or False).

Otrzymanie macierzy wartości początkowej oraz ich wizualizację opisane są w klasie **Sensor_Output**. Wyświetlanie wykresów umożliwia klasa **Sensor_Output_Plot**. Obie klasy zostały napisane w pliku „*data_processing/data_visualization.py*”.

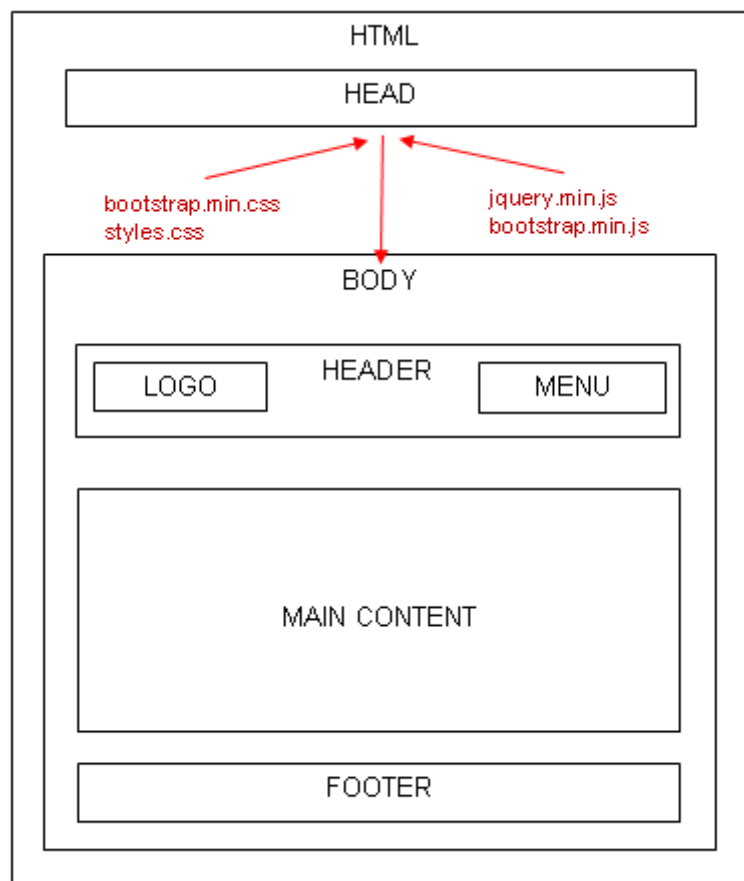
Podczas inicjalizacji obiektu klasy **Sensor_Output_Plot** konieczne jest zdefiniowanie parametrów podobnych do **Allan_Variance_Plot**, a mianowicie:

- *Processing_entity* – obiekt z interfejsem **Processing_Entity**
- *Title* – nazwa wykresu
- *Labels* – lista nazw wszystkich linii, które zostały przedstawione na wykresie
- *Xlabel* – oś odciętej wykresu
- *Ylabel* – oś rzędnych wykresu
- *Grid* – wyświetlanie siatki (True or False).

3.3 Architektura strony webowej

Interfejs platformy symulacyjnej reprezentuje strona webowa, która została napisana w języku programowania PHP, hipertekstowym języku HTML, języku CSS przy użyciu biblioteki Bootstrap wersji 4.0 oraz języka programowania JavaScript wraz z biblioteką jQuery.

Bazowym plikiem strony jest **index.php**. Ten plik znajduje się w katalogu głównym folderu *Dyplom*. Poniżej przedstawiona jest jego architektura, szczegółowo została ona opisana w podrozdziale 4.2:



Rys. 3.3 Architektura strony webowej

Schemat UML strony webowej wygląda w następujący sposób:



Rys. 3.4 Schemat UML strony internetowej

W pozycjach menu „Główna” i „Teoria” otwierają się strony z treścią, na których znajduje się tytuł pracy i krótki opis teoretyczny.

Pozycja menu „Ustawienia” umożliwia na konfigurowanie ścieżek do plików **python.exe** i **script.py**, z których uruchamiane są skrypty pracy. Ścieżki do skryptów Pythona pokazują rzeczywiste wyniki pracy i znajdują się w pliku **config.txt**. Za pomocą przycisku „zapisz” można zmienić ustawienia zgodnie z aktualnymi ścieżkami odpowiednich plików na komputerze. Wyniki pracy w idealnych warunkach (punkt menu „Symulacja”) można uzyskać dla żyroskopu lub akcelerometru, wybierając odpowiednie parametry: time, frequency, color i naciśnięciu przycisku „oblicz”.

Można pobrać własne dane wejściowe i uzyskać wyniki skryptu w punkcie menu „Pomiary rzeczywiste”, a także można zobaczyć opis pomiarów rzeczywistych wraz z plikiem załadowanym domyślnie. Wyczyścić wyniki skryptu można za pomocą przycisku „wyczyść”.

Struktura i nawigacja strony są napisane tak, aby użytkownik mógł w łatwy sposób ją zrozumieć i wykorzystać.

4. Implementacja platformy symulacyjnej

W tym rozdziale zostaną omówione najważniejsze etapy implementacji platformy symulacyjnej: interfejs, implementacja środowiska, ruchu czujników, składników szumowych, wariancja Allana, MPU6050.

4.1 Instrukcja uruchomienia programu

Uruchomienie platformy składa się z dwóch etapów. Pierwszy polega na odpalaniu interfejsu, drugi – skryptów źródłowych.

Zaczynamy od instalacji technologii wykorzystanych bezpośrednio do pisania kodu źródłowego platformy:

1. Python (wersja 3.6). Można ściągnąć bezpośrednio ze strony internetowej. <https://www.python.org/downloads/>
2. Edytor kodu Pycharm firmy JetBrains. <https://www.jetbrains.com/pycharm/download/#section=windows>

Po instalacji środowiska należy go skonfigurować. Do tego celu należy wykonać następujące kroki:

1. Przejdź do katalogu z projektem. **File**→**Open**→„...\\imu_simulation-master\\main.py”.
2. Konfiguracja venv dla projektu. Dodatkowo w IDE Pycharm należy sprawdzić, czy zostało podłączone środowisko wirtualne. **File**→**Settings** →**Project: imu_simulation-master**. W widoku **Project Interpreter** wybieramy właściwą wersję środowiska wirtualnego, w przypadku niniejszej pracy jest to Python 3.6.
3. Instalacja bibliotek zdefiniowanych w pliku **requirements.txt**.

Po odpowiedniej konfiguracji środowiska implementacyjnego należy zainstalować technologie webowe. W ramach tej pracy został użyty lokalny serwer Xampp — narzędzie które służy do odpalenia strony webowej lokalnie.

Instrukcja odpalania strony webowej:

1. Instalujemy Xampp. <https://www.apachefriends.org/pl/download.html>
2. Folder ze stroną internetową umieszczamy: „.....\\xampp\\htdocs”
3. W wyszukiwarce wpisujemy adres „http://localhost/strona/index.php”

4.2 Implementacja interfejsu

W pliku **index.php** wykonywane jest oznaczenie strony: górna część – tag **<header>**, w której znajdują się bloki z logotypem po lewej stronie oraz menu po prawej, blok treści – tag **<body>** oraz dolna część – tag **<footer>**. W tagu **<head>** zostały podłączone komponenty odpowiadające za wygląd strony z perspektywy użytkownika końcowego. Są to pliki o rozszerzeniach ***.css** i ***.js**. Pliki **bootstrap.min.css** z podstawowymi stylami biblioteki Bootstrap oraz plik **styles.css** zawierający style customowe znajdują się w folderze **css**. Pliki skryptowe **jquery.min.js** biblioteki jQuery oraz plik **bootstrap.min.js** zawarte są w folderze **js**. Dodatkowe rozszerzenie ***.min** oznacza tzw. *Minifikację*, czyli napisanie całego kodu w

jedną linię. Pozwala to na oszczędzanie pamięci plików oraz szybszą czytelność kodu przez przeglądarki.

W zależności od tego, do której pozycji menu przechodzimy, ładują się odpowiednie pliki: **page.php**, **page1.php**, **page2.php**, **page3.php** oraz **page4.php**. Dane pliki połączone są między sobą funkcją języka PHP **include** oraz zmienną **p**.

```
<div class="row">
  <div class="content-col col-12 py-2">
    <?php $p=(isset($_GET['p']))?$_GET['p']:''; include dirname(__FILE__)."/page".$p.'.php'; ?>
  </div>
</div>
```

Rys. 4.1 Interakcję między stronami

Pozycja menu „*Główna*” przedstawia użytkownikowi tytuł pracy, a pozycja „*Teoria*” – krótki opis teoretyczny.

Pozycja menu „*Ustawienia*” umożliwia na konfigurowanie ścieżek do plików **python.exe** oraz **script.py**. Oba pliki służą do uruchomienia skryptu pracy. Ścieżki do skryptów Pythona pokazujące praktyczne wyniki pracy, znajdują się w pliku **config.txt**. Można je edytować za pomocą funkcji **getConfig()** oraz **setConfig()**, które odpowiednio odczytują i zapisują dane w pliku **config.txt**.

W naszym przypadku ścieżka do pliku wygląda następująco:

C:\python_project\env\Scripts\python.exe

C:\python_project\imu_simulation-master\main.py

Wyniki symulacji ruchu czujnika w warunkach idealnych zostały przedstawione w pozycji menu „*Symulacja*”. Ścieżka do danych rzeczywistych została podana w punkcie menu „*Ustawienia*”. Wyniki analizy danych rzeczywistych zostały podane w pozycji menu „*Dane rzeczywiste*”.

Wywoływanie programu zewnętrznego (**python.exe**) wykonujemy za pomocą funkcji PHP **shell_exec()**, która pobiera dwa parametry. Pierwszy to ścieżka bezwzględna do pliku **python.exe**, drugi – ścieżka do pliku programu skryptowego napisanym w Pythonie (plik **main.py**).

Używając spacji, można podać dodatkowe parametry do funkcji **shell_exec()**, które służą jako dane wejściowe do pracy skryptu Pythona. Niektóre dane podane z góry przez użytkownika są niezbędne do symulacji. Na przykład wartości „czas” oraz „częstotliwość”, przy użyciu przycisku „*Oblicz*” są przekazywane za pośrednictwem znacznika html **<form>** do funkcji „pythonowskich”. Oczyszczanie danych, czyli wykres Allana oraz dane wyjściowe można wykonać przy użyciu przycisku „*Wyczyść*”. Plik z danymi wejściowymi jest przesyłany za pomocą biblioteki UploadFile znajdującej się w pliku **uploadfile.class.php** w folderze **lib**. Użytkownik ma możliwość wgrać dane w formatach *.csv lub *.txt.

4.3 Interakcja między Pythonem a PHP

Ze strony internetowej wprowadzone pomiary są przekazywane za pomocą funkcji `shell_exec()` parametrem `$output`.

```
require_once(dirname(__FILE__).'/lib/uploadfile.class.php');
$sel_time = isset($_POST['time'])?$_POST['time']:100;
$sel_fs = isset($_POST['fs'])?$_POST['fs']:100;

if (isset($_POST['run']) && $_POST['run'] == 1) {
    $type = $_POST['type'];
    $fs = $_POST['fs'];
    $time = $_POST['time'];
    $file = dirname(__FILE__).'/files/mpu6050_1.csv';
    $command = escapeshellcmd(getConfig()[0].'.getConfig()[1].'. $type.' ' '.$fs.' ' '.$time.' ' '.$file);
    $output = shell_exec($command);
    echo $output;
}
```

Rys. 4.2 Wysyłania danych do Pythona

W języku Python do otrzymania przekazywanych plików należy skorzystać ze standardowej biblioteki `sys`. Przesyłane pomiary są w postaci listy: `[sys.argv[0], sys.argv[1], sys.argv[2], sys.argv[3], sys.argv[4]]`.

- `sys.argv[0]` – domyślna nazwa skryptu, przekazywane dane wejściowe zaczynają się od pierwszego argumentu
- `sys.argv[1]` – rodzaj symulacji. Posiada cztery możliwe wartości:
 - *Ideal* – symulacja ruchu idealnego czujnika bez zakłóceń pomiarowych
 - *Acc* – symulacja ruchu zakłóconego akcelerometru
 - *Gyro* – symulacja ruchu zakłóconego czujnika żyroskopowego
 - *Real* – rzeczywiste pomiary
- `sys.argv[2]` – zmienna reprezentująca częstotliwość symulacji (*frequency*)
- `sys.argv[3]` – zmienna reprezentująca czas trwania symulacji (*time*)
- `sys.argv[4]` – zależy od wartości podanej w `sys.argv[1]`. W przypadku *Ideal* będzie pustą. W przypadku symulacji ruchu czujnika *Acc* lub *Gyro* przekazuje kolor. Ostatnia opcja zawiera plik z pomiarami rzeczywistymi.

W wyniku interakcji Python dostaje następujące zmienne: *time*, *simulator*, *fs* (skrót „frequency”), *color*, *file*. Dwie ostatnie otrzymujemy w zależności od wartości `sys.argv[1]`.

4.4 Implementacja IMU

Interfejs dla modułu IMU został napisany w języku programowania Python przy użyciu bibliotek `numpy`, `scipy`, abstrakcyjnej klasy `abc`, `matplotlib`, `pyquaternion`. Moduł IMU zawiera klasy i metody służące do opisanego środowiska, ruchu czujnika oraz generowania składników szumowych.

Bazowym plikiem do interakcji ze stroną webową jest **main.py**. W nim wykonywane są interakcje między językami `php` → `python`, które zostały opisane w rozdziale 4.3. Symulacja ruchu idealnego – funkcja *ideal_imu*, symulacja ruchu zakłóconego czujnika funkcja – *noise_imu* oraz analiza pomiarów rzeczywistych – funkcja *real_imu*. Komponenty modułu IMU zostały podłączone do funkcji, których zadanie polega na wygenerowaniu syntetycznych pomiarów. Są to pliki **sensor.py**, do którego należą klasy potomne **Accelerometr** i **Gyroscope**

noise.py oraz **imu.py**. Plik **sensor.py** zawiera realizację modelu trzyosiowego czujnika klas **Accelerometr** i **Gyroscope**, które zostały opisane według wzoru podanego w algorytmach (2.1) oraz obiektach klas abstrakcyjnych **Platform** i **Component** opisujących ruch czujnika. Realizacja modelu trzyosiowego czujnika opiera się na wektorach *bias*, *position_offset*, *orientation_offset* oraz macierzach *sensitivity*, *noise*.

W klasie **sensor** są zawarte metody służące do modyfikacji sygnału. Na przykład dopasowanie składników szumowych do sygnału odbywa się w dwa etapy. Pierwszy etap polega na wywołaniu metody **noise_output**, która odnosi się do pliku **noise.py** i zwraca wartości wygenerowanego szumu lub macierz zerową, w przypadku ruchu idealnego. Drugi etap opisany jest w metodzie **output**, polega na sumowaniu wektorów ruchu oraz szumu. Implementacja wzoru (3.1) realizowana w metodzie `__call__` i zwraca wartości sygnału wyjściowego, które są transformowane do jednostek obliczeniowych, używając przekształcenia afinicznego. Finalne klasy **Gyroscope** i **Accelerometr** są pochodnymi klasy **sensor**. Klasa **Gyroscope** stosuje metody **angular_velocity** w celu dopasowania ruchu czujnika żyroskopowego do ruchu platformy, natomiast klasa **Accelerometr** oblicza różnicę przyspieszenia bezwzględnego z przyspieszeniem platformy i wyraża ją jako wektor *gravity*.

W pliku **noise.py** został zaimplementowany mechanizm do generowania syntetycznych składników szumowych. Interfejs interakcji z generatorem jest realizowany za pomocą abstrakcyjnej klasy **Abstract_Noise_Generator** z jedną metodą *generate*, która zwraca dwuwymiarową numpy-macierz o danym widmie. Ten interfejs umożliwia dalsze wdrażanie dodatkowych opcji do modelowania szumu pomiarowego. Obecnie tylko wdrożona klasa **Colored_Noise_Generator** jest w stanie wygenerować biały, różowy, fioletowy, czerwony szum oraz dryft. Przy tym został wykorzystany dość rozpowszechniony sposób przetwarzania białego szumu Gaussa, używając filtrów o określonej charakterystyce amplitudowo-częstotliwościowej.

W pliku **trajektory.py** został zaimplementowany mechanizm do opisu ruchu obrotowego oraz translacyjnego obiektu. Interfejs ruchu czujników jest realizowany za pomocą klas **StaticTrajectory**. Metody zaimplementowane w tej klasie symulują statyczne położenie ciała w przestrzeni, orientację, pozycję itp.

W pliku **imu.py** zostały zaimplementowane modele inercyjnego czujnika, które faktycznie są zestawem współosiowego żyroskopu i akcelerometru.

4.5 Implementacja Data_processing

Folder **Data_processing** składa się z następujących plików: **allan_variance.py** – ten plik zawiera w sobie klasy **AllanVariance**, **AllanDeviation** oraz **AllanVariancePlot** służące odpowiednio do implementacji metody wariancji Allana, odchylenia Allana oraz rysowania wykresu odchylenia Allana; **data_visualization.py** służy do rysowania wykresów reprezentujących zależność sygnału czujników IMU od czasu; **processing_entity.py** składa się z dwóch klas abstrakcyjnych **ProcessingEntity** oraz **ProcessingDecorator**.

Pliki pomocnicze do implementacji modułu **data_processing** to zbiór danych pomiarowych, który jest przechowywany w folderze **datasets** oraz **MUP6050.py** (służy do transformacji pomiarów rzeczywistych).

Abstrakcyjna klasa **ProcessingEntity** zawiera tylko jedną metodę `_processing(self, *args, **kwargs)` będącą wynikiem zastosowania metody `__call__`. Ta metoda rozpoczyna przetwarzanie danych wejściowych zgodnie z algorytmem zdefiniowanym podczas inicjalizacji obiektu klasy. Potomna klasa **ProcessingDecorator** zawiera w sobie dodatkowe narzędzia, które wykorzystywane są podczas budowania wykresów, analizowania statycznych charakterystyk itp.

Klasa **AllanVariance** zawiera w sobie abstrakcyjną metodę *_processing*, która pobiera dane otrzymane od modułu IMU w postaci macierzy *acc_out* lub *gyro_out*. Kolejny typ danych wejściowych do metody *__processing* to dane otrzymane i odpowiednio przekształcone przy użyciu metod klasy **AccelerometrMPU6050** lub jej odpowiednika **GyroscopeMPU6050**. Metoda *_processing* zwraca wartości *tau* oraz *avar* (pobierane metodą **AllanDevitation**).

Podczas inicjalizacji obiektu wartości wyjściowych klasy **Sensor_Output** oraz **AllanVariance** warto określić tylko częstotliwość próbkowania danych wejściowych. Metoda *_processing(self, sensor, time, unit_reduction)* umożliwia pobranie macierzy wartości wyjściowych z obiektu *sensor* w określonym czasie *time*, przy czym w przypadku *unit_reduction=True* macierz wartości wyjściowych będzie zredukowana do jednostek wejściowych, posługując się wzorem 3.1 (sposób odpalania metody *__call__* obiektu *sensor*). Przy *unit_reduction=False* dane zostaną otrzymane w wyniku zastosowania metody *output* obiektu *sensor*.

5. Prezentacja

Ten rozdział został podzielony na dwie części. Pierwsza z nich reprezentuje interfejs oraz omawia najważniejsze aspekty związane z interfejsem. W pierwszej części została zrobiona szczegółowa prezentacja strony internetowej na podstawie potrzeb użytkownika ([HU1-HU10](#)).

Druga część omawia rozwiązania, opisuje wykresy sygnałów i odchylenia Allana. Dodatkowo w tej części została zrobiona weryfikacja rozwiązań zgodnie z założeniami projektowymi ([1.5 Plan pracy](#)).

5.1 Interfejs

Po otwarciu wyszukiwarki i wpisaniu odpowiedniego adresu ([instrukcja uruchomienia strony](#)) domyślnie otwiera się witryna strony. Od razu widzimy tytuł, logo, menu oraz inną informację. Strona webowa może zostać odpalona nawet w przypadku braku zainstalowanych technologii implementacyjnych (Python). Wtedy wygenerowanie wykresów już nie będzie możliwe.



Rys. 5.1 Witryna strony

*Więcej informacji na temat czujników pomiarowych, wariancji Allana oraz opisanie podstawowych pojęć „sygnał”, „szum”, „zakłócenia” można znaleźć na podstronie **Teoria**, klikając w odpowiednią pozycję menu.*



Teoria

Platforma symulacyjna służy do analizowania części szumowej sygnału. Jako element wejściowy mamy sygnał, który może być wygenerowany lub rzeczywisty. Sterownikiem jest algorytm filtracji sygnału. Element wyjściowy – szum.



Koncepcja Pracy

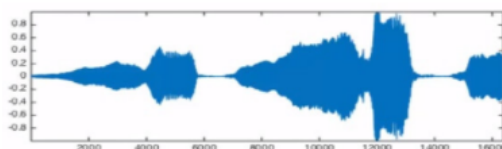
[Sygnał](#)[Szum](#)[Zakłócenia](#)[Akcelerometr](#)[Żyroskop](#)[Odchylenie Allana](#)

Sygnał

Sygnałem nazywamy pewną wielkość, której wartość może zostać zmierzona i która umożliwia przenoszenie informacji. Różnego typu sygnałów jest nieskonczenie wiele i z tego powodu zostały one podzielone na grupy.

Przykłady sygnałów:

Mówienie



Rys. 5.2 Materiał teoretyczny – sygnał



Teoria

Platforma symulacyjna służy do analizowania części szumowej sygnału. Jako element wejściowy mamy sygnał, który może być wygenerowany lub rzeczywisty. Sterownikiem jest algorytm filtracji sygnału. Element wyjściowy – szum.



Koncepcja Pracy

[Sygnał](#)[Szum](#)[Zakłócenia](#)[Akcelerometr](#)[Żyroskop](#)[Odchylenie Allana](#)

Akcelerometr – co to?

Urządzenie, które mierzy własny ruch w przestrzeni. Może być użyte do określenia położenia telefonu mierząc przyspieszenia liniowe. Znajdują się w niemal wszystkich smartfonach produkowanych dzisiaj.

Do czego służy?


Do diagnostyki pracy maszyn, urządzeń czy konstrukcji poddawanych dużym naprężeniom, np. konstrukcji stalowych masztów, mostów czy obiektów budowlanych. Akcelerometry stosowane są również m.in. do ochrony dysków twardych przed uszkodzeniem, w sprzęcie medycznym i sportowym, w aparatach i kamerach, w smartfonach, pilotach, kontrolerach czy w systemach nawigacji.

Z czego się składa?

Do niedawna był on wykorzystywany głównie w grach przy zmianie orientacji ekranu. Ostatnio jednak dzięki popularyzacji technologii VR na żyroskop spada o wiele więcej obowiązków. Teraz telefon musi śledzić dokładne położenie głowy i robić to możliwie jak najszybciej i precyzyjniej, tak by móc jak najlepiej odczytać obraz i dać lepszą orientację użytkownikowi wirtualnego świata.

Rys. 5.3 Materiał teoretyczny – akcelerometr

Wygenerowanie syntetycznych wykresów jest możliwe w punkcie menu **Symulacja**. Widok symulacji zawiera w sobie wyjaśnienia, na czym polega symulacja syntetycznych sygnałów, cel takiej symulacji oraz w jaki sposób można regulować zmienne. Generowanie sygnału jest możliwe dla czujnika żyroskopowego i akcelerometru.

 Politechnika Wrocławska

Główna Teoria **Symulacja** Pomiary rzeczywiste Ustawienia

Symulacja

Pod tytułem „symulacja” zostały określone pojęcia, które charakteryzują idealny ruch czujnika, czyli warunki, dla których na pomiary nie wpływają żadne czynniki zewnętrzne lub wewnętrzne. Wtedy sygnał nie posiada żadnych niepożądanych składników.

Teoretycznie w stanie nieruchomym sygnał podany od czujnika powinien zostać niezmieniony, czyli musimy na wyjściu dostać linię prostą, a wykres Allana wtedy powinien być pusty. W praktyce wciąż na czujnik oddziałują różnego rodzaju zakłócenia, dlatego otrzymanie „czystego” sygnału jest niemożliwe.

W ramach projektu moduł symulacja został stworzony do następujących celów.

- Wygenerować szumy syntetyczne.
- W warunkach idealnych pojedynczo połączyć ich z sygnałem idealnym.
- Porównanie wykresów.

Przy pomiarach syntetycznych istnieje możliwość regulowania następujących zmiennych:

Czas – regulowanie czasu trwania pomiarów.
Częstotliwość – regulowanie częstotliwości pomiarów.
Kolor – wybór odpowiedniego szumu.

Ogólna liczba wygenerowanych syntetycznie pomiarów - $\text{czas} \times \text{częstotliwość}$.

Żyroskop

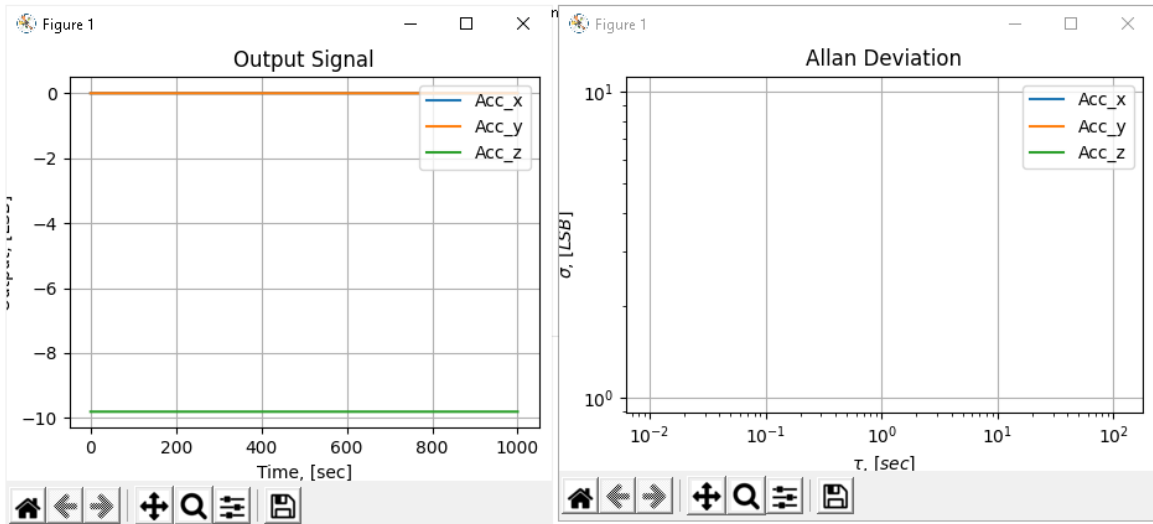
Czas:

Częstotliwość:

Kolor:

Rys. 5.4 Ilustracja symulacji – teoria

W symulacji są stworzone idealne warunki do wygenerowania ruchu czujnika lub wybranego szumu. Do wygenerowania ruchu idealnego należy dokonać odpowiedniego wyboru czujnika, ustalić czas trwania sygnału i jego częstotliwość. Rysunek poniżej przedstawia wykres ruchu idealnego akcelerometru, jego wariancję oraz ich opis:

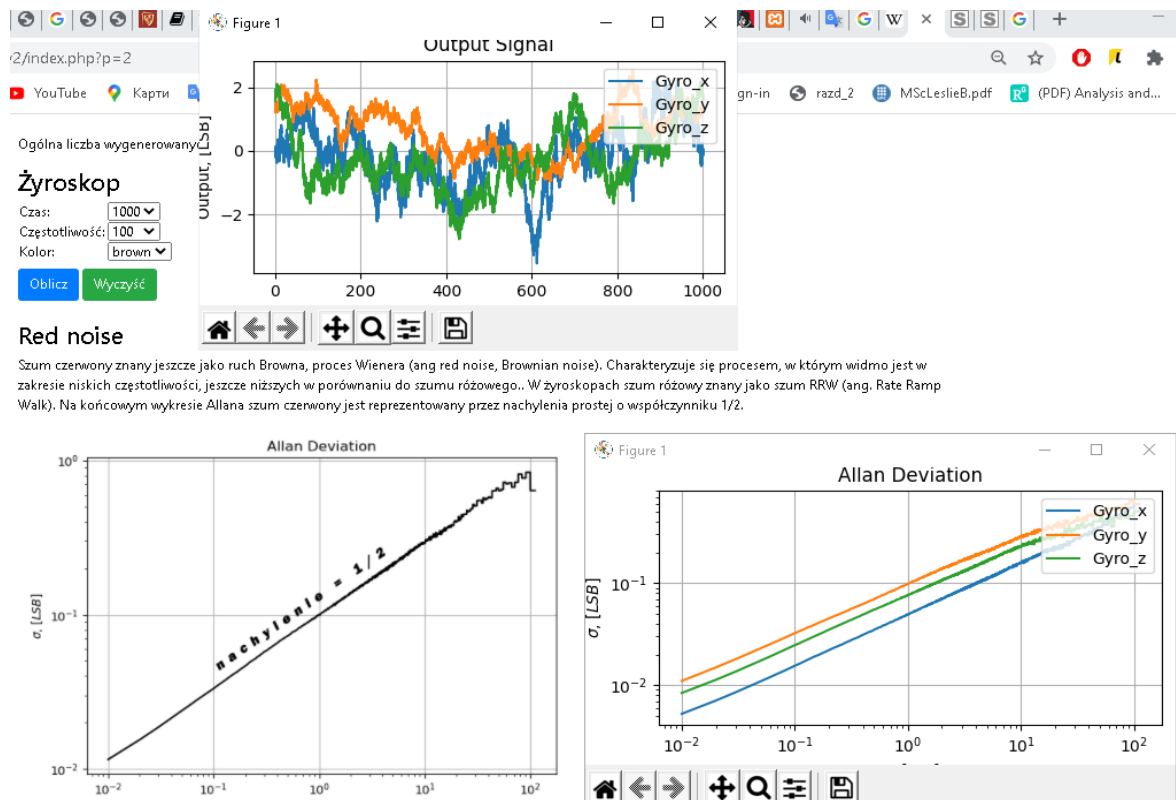


Ideal Akcelerometr

Na wykresie został przedstawiony sygnał trzyosiowego akcelerometru w warunkach idealnych. Czujnik znajduje się w pozycji nieruchomej. Akcelerometr jest bardzo podatny na różne zakłócenia. W pozycji stacjonarnej na oś Z wpływa grawitacja. Z tego powodu ona została zamieszczona. Pozostałe osie są niezmienniczone. Reprezentacja sygnału w idealnych warunkach będzie linią prostą. Przy braku składników szumowych wykres odchylenia Allana będzie pusty.


Rys. 5.5 Ilustracja symulacji – idealny ruch akcelerometru

Procedura generowania kolorowych szumów jest podobna, wystarczy dokonać właściwego wyboru. Zaletą strony internetowej jest jej funkcjonalność, można na raz wygenerować dwa lub więcej wykresów. Ograniczeniem w tym przypadku będą możliwości procesora każdego komputera.



Rys. 5.6 Ilustracja symulacji – żyroskop zakłócony szumem czerwonym

Do wyczyszczenia informacji pojawiającej się dynamicznie (dotyczy opisu składników szumowych), należy użyć przycisku „wyczyść”. W celu dokonania działań nad pomiarami rzeczywistymi należy przejść do punktu menu **Pomiary rzeczywiste**. W tej kategorii widoczny opis teoretyczny, dotyczący niektórych szczegółów oraz dwie różne opcje do analizowania pomiarów. Pierwsza pod tytułem **Pomiary rzeczywiste** zawiera jeden plik z pomiarami, który łąduje się domyślnie wraz z opisem. Druga opcja **Twoje dane** udostępnia użytkownikowi załadowanie własnych pomiarów.

Politechnika Wrocławska

GłównaTeoriaSymulacjaPomiary rzeczywisteUstawienia

Pomiary rzeczywiste

Pomiary rzeczywiste otrzymane używając czujnik MPU-6050, w który został wbudowany trzyosiowy akcelerometr i żyroskop.

Przykładowe pomiary domyślnie otwierają jeden z plików wraz z opisem składników szumowych. W razie potrzeby dodatkowo można wgrać własne pomiary.

Przy pomiarach rzeczywistych istnieje możliwość regulowania następujących zmiennych:

Czas – regulowanie czasu trwania pomiarów
Częstotliwość – regulowanie częstotliwości pomiarów

Uwaga!
Warto pamiętać że **czas** \times **częstotliwość** jest równa ogólnej ilości pomiarów. Dla tego **czas** \times **częstotliwość** nie może być większa za ilość mierzonej próbki. Tylko mniejsza lub maksymalnie równa.

Pomiary rzeczywiste

Czas:

Częstotliwość:

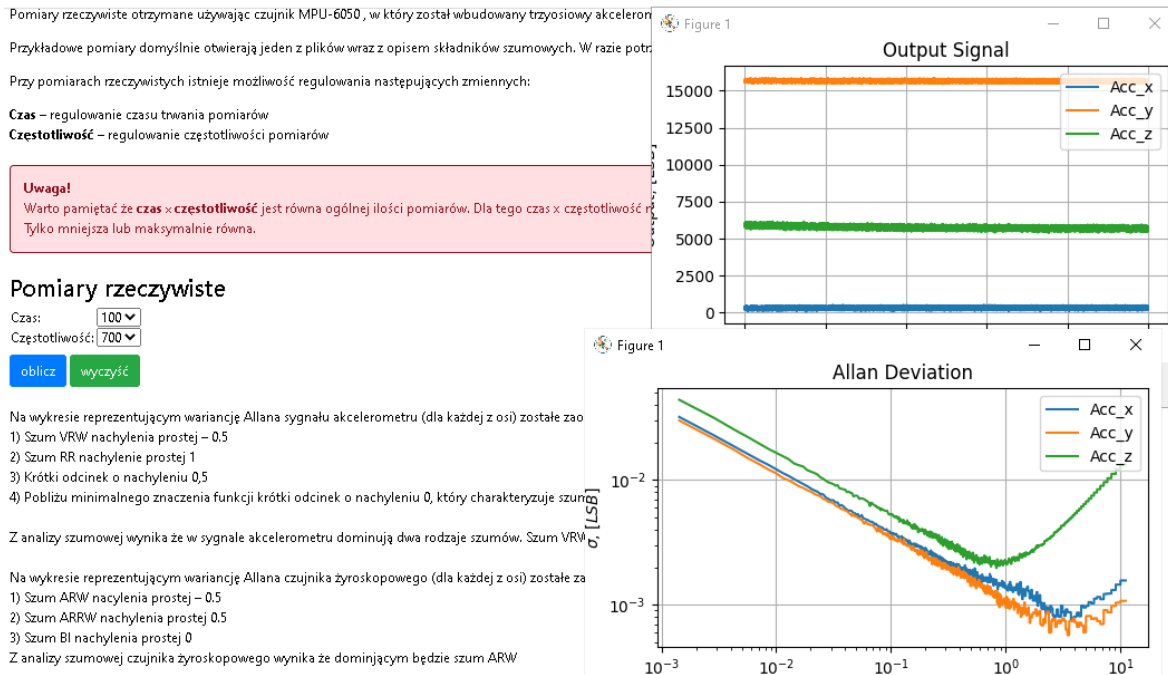
Twoje dane

Czas:

Częstotliwość:

File (*.csv or *.txt): Файл не вибрано

Rys. 5.7 Ilustracja interfejsu dla pomiarów rzeczywistych



Rys. 5.8 Ilustracja pomiarów rzeczywistych – analiza

Podstrona **Ustawienia** powstała jako dodatkowa w celu konfiguracji strony internetowej z Pythonem a projektem. We właściwych polach należy podać odpowiednie ścieżki:

[Główna](#)
[Teoria](#)
[Symulacja](#)
[Pomiary rzeczywiste](#)
[Ustawienia](#)

Ustawienia

ścieżka do pliku python.exe (ex.: C:\program\demo\Scripts\python.exe)

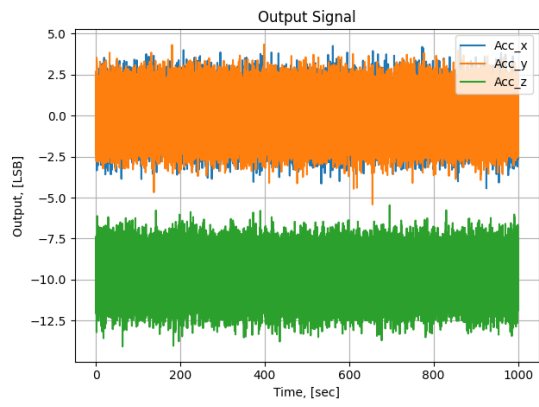
ścieżka do pliku script.py (ex.: C:\program\projects\python\script.py)

Rys. 5.9 Ustawienia strony

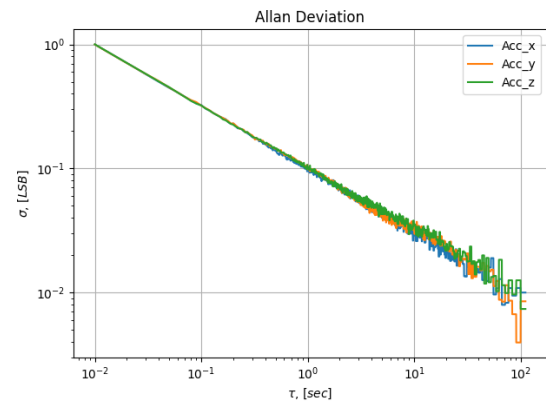
5.2 Rozwiązania

Generowanie szumów oraz ich analiza z wykorzystaniem takich narzędzi, jak wariancja Allana nie jest zadaniem trywialnym. Zgodnie z założeniami projektowymi, pierwsza część polegała na wygenerowaniu syntetycznych sygnałów pomiarowych oraz ich analizowaniu pod kątem składników szumowych. W idealnych warunkach każdy szum został przeanalizowany osobno. Otrzymane wykresy odchylenia Allana idealnie odzwierciedlają odpowiedni szum.

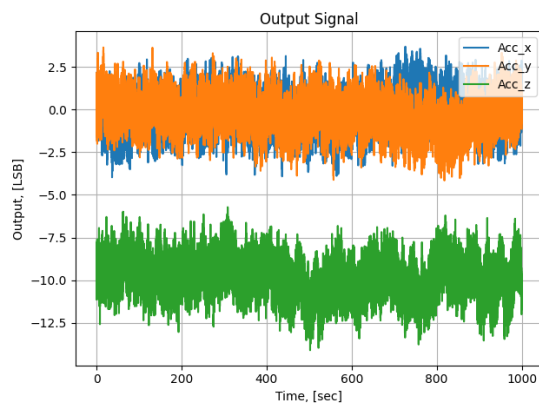
(a)



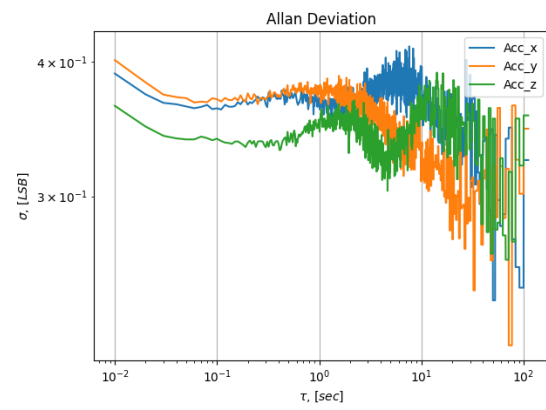
(a)



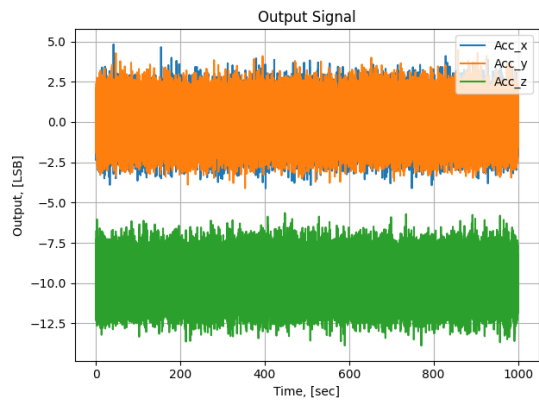
(b)



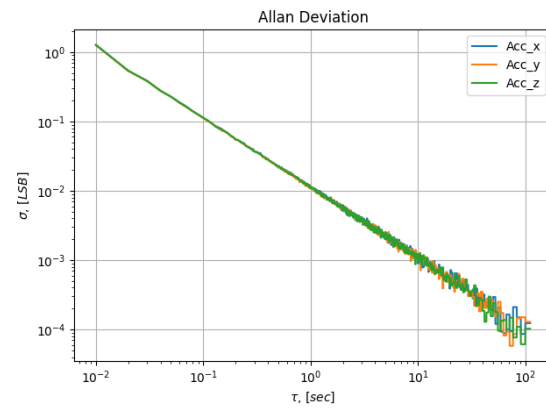
(b)

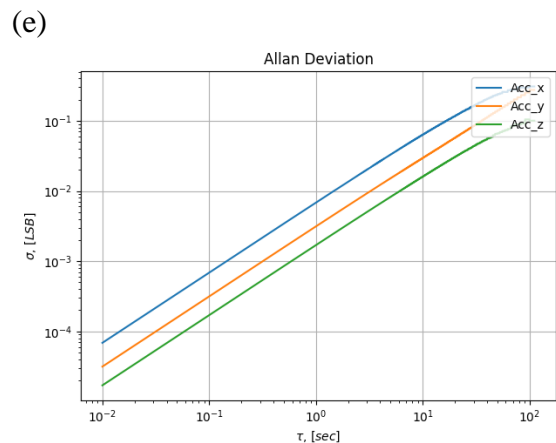
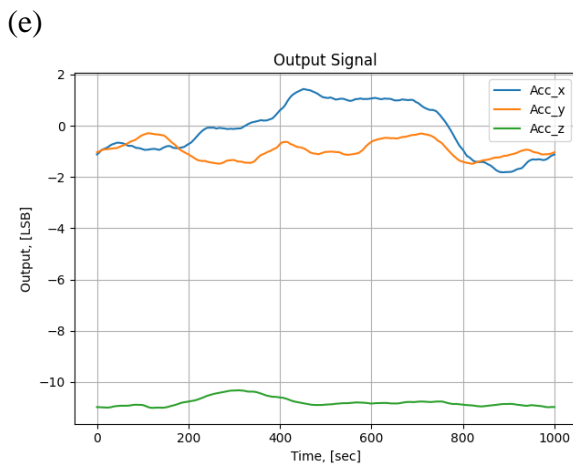
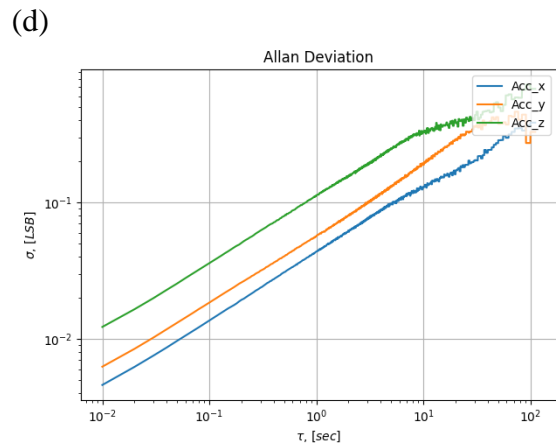
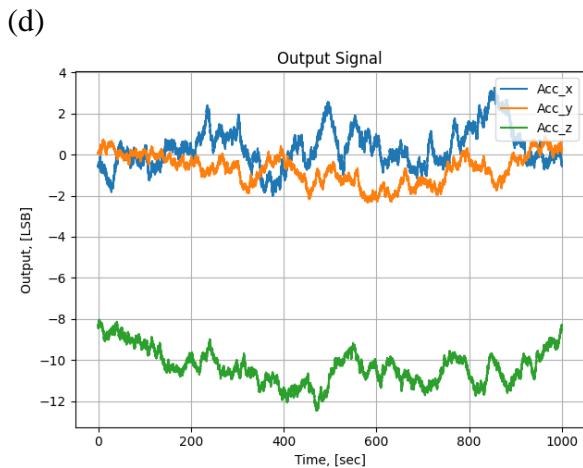


(c)



(c)





Rys. 5.10 Reprezentacja szumowa akcelerometru

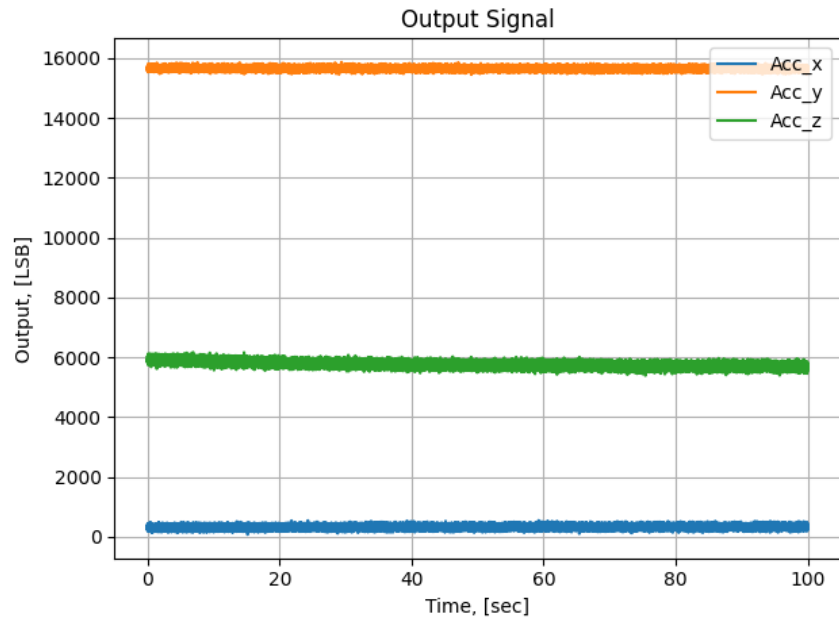
- a) szum biały b) szum różowy
- c) szum czerwony d) szum fioletowy
- e) dryft

Rys. 5.11 Wariancja Allana

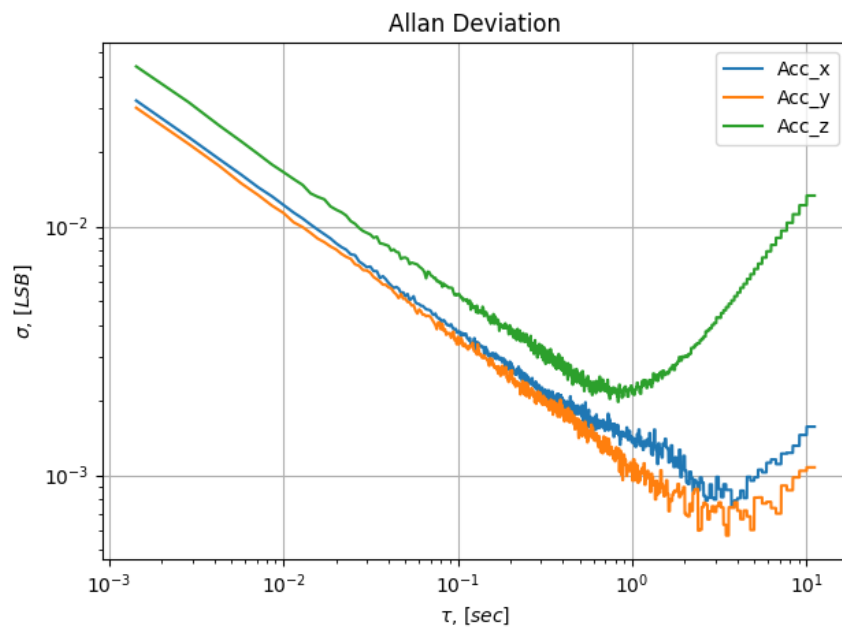
- a) szum VRW b) szum BI
- b) szum kwantyzacji d) szum VRRW
- e) szum RR

Na rysunku 5.11 przedstawione są nachylenia prostej, dla każdego szumu występującego w sygnale akcelerometru w idealnych warunkach. W przypadku czujnika żyroskopowego wykresy Allana będą identyczne. Warto uwagi pozostaje wykres szumu BI. W porównaniu do innych wykresów skala została zwiększona domyślnie w trzy razy. Ten przypadek demonstruje rozrzut pomiarów dla szumu BI. Jeżeli zmniejszyć skalę, do rozmiarów innych wykresów przedstawionych na rysunku 5.11, wtedy dostaniemy prostą o nachyleniu 0. To rozwiązanie zostało zaprezentowane podczas generowania przykładowego wykresu dla szumu BI (rysunek 2.17). Z tego można wywnioskować, że zwiększając skalę dla każdego typu szumu również, otrzymamy podobny rozrzut pomiarów.

Druga część założeń projektowych polegała na możliwości załadowania rzeczywistych pomiarów oraz ich analizie. W ramach niniejszej pracy pomiary rzeczywiste zostały uzyskane na podstawie czujnika MPU6050. Najprostszy sposób analizy szumowej polega na wygenerowaniu prostej z odpowiednim kątem nachylenia i dopasowaniu do wykresu odchylenia Allana. Otrzymane pomiary pierwszego zestawu zostały przedstawione na rysunkach 5.12 – 5.13.



Rys. 5.12 Pomiary rzeczywiste dla akcelometru

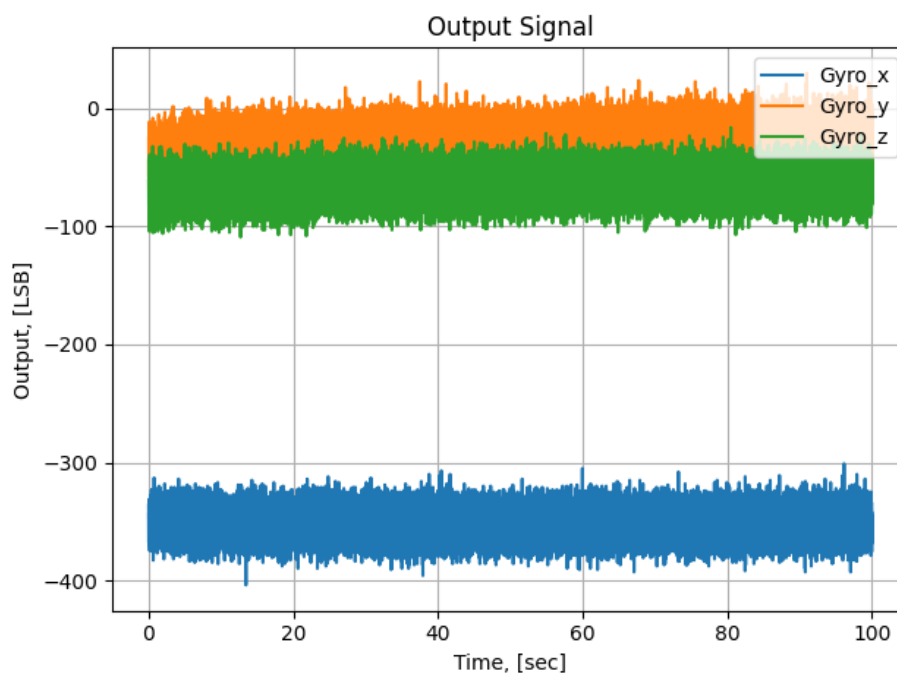


Rys. 5.13 Wariancja Allana dla akcelometru

Analizowanie nachylenia prostych sygnału akcelometru pozwala wyciągnąć następujące wnioski. Na dłuższym odcinku zostało zaobserwowane nachylenie prostej -0.5 , to znaczy, że obserwujemy biały szum. Dodatnie nachylenie $+1$, które jest widoczne dla osi Z, demonstruje szum RR, który jest skutkiem zgromadzenia błędów szumu BI, charakteryzujący się krótkim odcinkiem o nachyleniu bliskim zera. Szum VRRW jest widoczny dla każdej z osi na krótkim odcinku.

Podsumowując, na wykresie zostały zaobserwowane (dla każdej osi) następujące składniki szumowe: szum VRW, szum BI, szum VRRW. Oprócz tego dla osi Z został zaobserwowany szum RR.

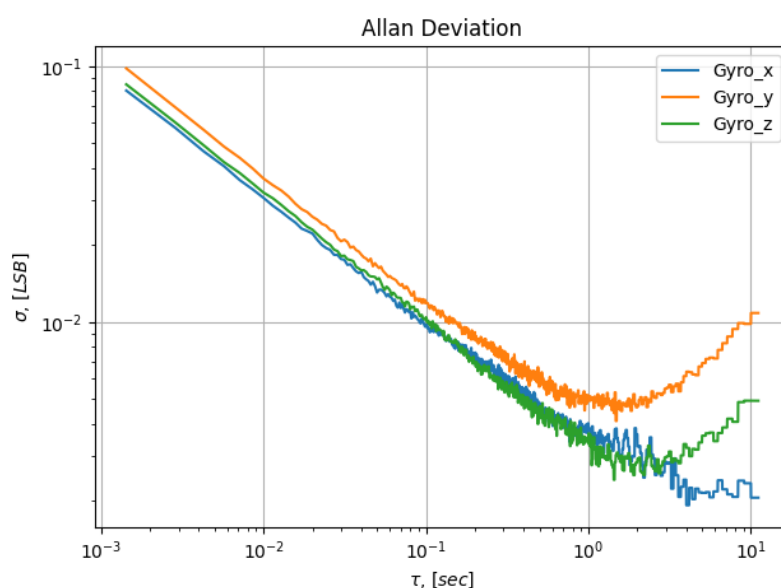
Poniżej przedstawione pomiary (rysunek 5.14) dla czujnika żyroskopowego. Wykres odchylenia Allana został przedstawiony na rysunku 5.15:



Rys. 5.14 Pomiary rzeczywiste dla żyroskopu

Analiza odchyleń prostych dla czujnika żyroskopowego pozwala na wyciągnięcie następujących wniosków. Długi odcinek o nachyleniu -0.5 charakteryzuje szum ARW. Szum BI widoczny na krótkim odcinku prostej i znajduje się w punkcie minimum. Dla osi Y i Z widoczne są odcinki o nachyleniu 0.5, charakteryzujące obecność szumu ARRW.

Podsumowując, na wykresie zostały zaobserwowane (dla każdej osi) następujące składniki szumowe: szum ARW oraz szum BI. Szum VRRW obserwowany jest dla osi Y i Z.



Rys. 5.15 Wariacja Allana dla żyroskopu

6. Podsumowanie

Celem niniejszej pracy była analiza składników szumowych występujących w sygnale. Sygnał był otrzymywany w dwa sposoby. Pierwszy polegał na pobraniu pomiarów w zaprojektowanym urządzeniu pomiarowym, drugi – na otrzymaniu pomiarów w sygnale syntetycznym. Do tego sygnału składniki szumowe zostały wygenerowane i dopasowane.

Platforma symulacyjna umożliwia analizę sygnału metodą wariancji Allana. Największą wadą tej platformy jest to, iż analizowanie pomiarów rzeczywistych odbywa się wyłącznie na podstawie danych pomiarowych czujnika MPU6050. Do analizy metodą Allana pomiary należy doprowadzić do jednostek wyrażonych we właściwych wartościach przyspieszenia [m/s^2] (akcelerometr) i prędkości kątowej [$^\circ/\text{s}$] (żyroskop). Kolejna wada polega na braku analizy wykresu końcowego środowiskiem implementacyjnym oraz braku zastosowania innych metod do analizy szumowej.

Interfejs platformy służy do tego, aby przedstawić rozwiązanie zgodnie z zasadą UX (user experience). Strona webowa działa lokalnie na komputerze.

Początkowe cele pracy zostały spełnione. Platforma umożliwia generowanie sztucznych sygnałów i ich analizę wraz z rzeczywistymi pomiarami. Istnieje możliwość załadowania pomiarów rzeczywistych otrzymywanych na bazie czujnika MPU6050.

6.1 Kierunki dalszego rozwoju

1) Metody usuwania szumów

Wpływ składników szumowych powinien być sprowadzony do minimum. W tym celu należy użyć specjalnych metod służących do usuwania szumów: falkowa redukcja szumu (ang. *Wavelet denoising*) lub filtr medianowy (ang. *Median denoise*). Platforma po analizie powinna udostępniać możliwość odszumiania sygnału.

2) Wielofunkcyjny wybór metody do analizy składników szumowych sygnału

Do celu analizy sygnału na składniki szumowe warto stosować różne metody, na przykład jeszcze widmową gęstość mocy. Opis końcowy powinien zawierać w sobie połączoną informację od różnych metod.

Bibliografia

- [1] „Różnica między żyroskopem i akcelerometrem,” 2021. <https://pl.avktarget.com/articles/tehnologii/raznica-mezhdu-giroskopom-i-akselerometrom.html>
- [2] Xin Zhang, Yong Li, Peter Mumford, Chris Rizos, “Allan Variance Analysis on Error Characters of MEMS Inertial Sensors for an FPGA-based GPS/INS System,” *School of Surveying and Spatial Information Systems University of New South Wales, Australia*, 2008
- [3] “Akcelerometr – Jak to działa,” 2019. <https://botland.com.pl/blog/akcelerometr-jak-to-dziala/>
- [4] „Zasada działania żyroskopu. Jak to działa: żyroskop. Jak sprawdzić czy twój telefon ma żyroskop”, 2021. <https://bar812.ru/pl/princip-deistviya-giroskopa-kak-eto-rabotaet-giroskop-kak-uznat-est-li-na.html>
- [5] Krzysztof Brzostowski, „Czujniki pomiarowe” w *Zastosowanie przetwarzania sygnałów w fuzji danych strumieniowych*, Wrocław, 2021, pp. 101 – 131
- [6] „Arduino Pro Mini for Beginners,” 2019. <https://projectiot123.com/2019/04/09/arduino-pro-mini-for-beginners/>
- [7] „Arduino Pro Mini- charakterystyka techniczna,” 2021. <https://micro-pi.ru/arduino-pro-mini-%D0%BE%D0%B1%D0%B7%D0%BE%D1%80-%D0%BF%D0%BE%D0%B4%D0%BA%D0%BB%D1%8E%D1%87%D0%B5%D0%BD%D0%B8%D0%B5/>
- [8] „Charakterystyka czujnika MPU6050” <https://3d-diy.ru/wiki/arduino-datchiki/giroskop-i-akselerometr-gy521-mpu6050/>
- [9] Leslie Barreda Pupo, “Characterization of errors and noise in MEMS inertial sensors using Allan variance method.”, Barcelona, 2016
- [10] “Allan Variance: Noise analysis for gyroscope”, 2015 <https://telesens.co/wp-content/uploads/2017/05/AllanVariance5087-1.pdf>
- [11] „Pycharm,” 2020 <https://pl.wikipedia.org/wiki/PyCharm>
- [12] Sharova M.A., Diadin S.S. Allan variance in dynamically tuned inertial-unit gyro error analysis. *Journal of «Almaz – Antey» Air and Space Defence Corporation*, Moskov 2019;69-77
- [13] Matveev V.V. “Niedokładności pomiaru czujnika żyroskopowego” w *Systemy nawigacji inercyjnej.*, Tuła, 2012, pp. 149 – 196

- [14] “Accelerometer & Gyro Tutorial”, <https://www.instructables.com/Accelerometer-Gyro-Tutorial/>

Spis rysunków

- Rys. 1.1: Reprezentacja czasowa i częstotliwościowa sygnału
- Rys. 1.2: Ilustracja wykresów końcowych dla obu metod
- Rys. 1.3: Wariancję Allana oparta na kątach wyjściowych
- Rys. 1.4: Ilustracja środowiska prezentującego rozwiązania
- Rys. 1.5: Schemat Arduino Pro Mini
- Rys. 1.6: Czujnik MPU6050
- Rys. 2.1: Wstępna koncepcja pracy
- Rys. 2.2: Sygnał akcelerometru w środowisku idealnym
- Rys. 2.3: Sygnał czujnika żyroskopowego w środowisku idealnym
- Rys. 2.4: Reprezentacja szumu białego
- Rys. 2.5: Filtr szumu różowego
- Rys. 2.6: Reprezentacja szumu różowego
- Rys. 2.7: Filtr szumu czerwonego
- Rys. 2.8: Reprezentacja szumu czerwonego
- Rys. 2.9: Filtr szumu fioletowego
- Rys. 2.10: Reprezentacja szumu fioletowego
- Rys. 2.11: Filtr dryftu czujników pomiarowych
- Rys. 2.12: Reprezentacja dryftu
- Rys. 2.13: Schemat układu pomiarowego
- Rys. 2.14: Ilustracja błędów opisanych wariancją Allana
- Rys. 2.15: Szum ARW/VRW
- Rys. 2.16: Szum kwantyzacji
- Rys. 2.17: Szum BI
- Rys. 2.18: Szum RRW
- Rys. 2.19: Szum RR
- Rys. 3.1: UML - schemat budowy modułu **imu_model**
- Rys. 3.2: UML – schemat modułu **data_processing**
- Rys. 3.3: Architektura strony internetowej
- Rys. 3.4: Schemat UML strony internetowej
- Rys. 4.1: Interakcję między stronami
- Rys. 4.2: Wysyłania danych do Pythona
- Rys. 5.1: Witryna strony
- Rys. 5.2: Materiał teoretyczny - sygnał
- Rys. 5.3: Materiał teoretyczny - akcelerometr

Rys: 5.4: Ilustracja symulacji - teoria
Rys: 5.5: Ilustracja symulacji – idealny ruch akcelerometru
Rys: 5.6: Ilustracja symulacji – żyroskop zakłócony szumem czerwonym
Rys: 5.7: Ilustracja interfejsu dla pomiarów rzeczywistych
Rys: 5.8: Ilustracja pomiarów rzeczywistych - analiza
Rys: 5.9: Ustawienia strony
Rys: 5.10: Reprezentacja szumowa akcelerometru
Rys: 5.11: Wariancja Allana
Rys: 5.12: Pomiary rzeczywiste dla akcelerometru
Rys: 5.13: Wariancja Allana dla akcelerometru
Rys: 5.14: Pomiary rzeczywiste dla żyroskopu
Rys: 5.15: Wariancja Allana żyroskop

Spis tabel

Tabela 1.1: Porównanie metod analizy szumowej

Tabela 1.2: Specyfika techniczna Arduino Pro Mini

Tabela 1.3: Specyfika techniczna czujnika MPU6050

Tabela 2.1: Opis podsystemu symulacji

Tabela 2.2: Opis podsystemu identyfikacji

Tabela 2.3: Opis podsystemu reprezentacji