

## Zadania projektowe cz. 3:

### Zadania programistyczne z wizualizacją sygnałów automatyki (AiR)

(C++, z wykorzystaniem GUI w Win API, zewnętrzne dane tekstowe z czujników)

Każde zadanie wymaga implementacji zadania nadrzędnego (elementów GUI) oraz właściwego programu w zad 1-17. Finalny program należy wysłać na adres: [Tomasz.Merta@pg.gda.pl](mailto:Tomasz.Merta@pg.gda.pl) przed wysłaniem należy usunąć nadmiarowe pliki (w Visual Studio/Express wybrać BUILD-> CLEAN SOLUTION, usunąć duże pliki \*.sdf, folder ipch itp.). **Wysyłane pliki nie mają przekraczać 1 MB !!!**

#### Wymagania dotyczące implementacji (funkcjonalności GUI) oraz wczytywania danych:

- W każdym zadaniu należy zaimplementować proste GUI, w skład którego wchodzi:
  - przyciski odpowiedzialne za zmianę podziałki czasowej (przy rysowaniu wykresu oraz animacji),
  - przyciski odpowiedzialne za zmianę skali amplitudy (przy rysowaniu wykresu),
  - suwak lub 2 przyciski służące do zmiany okna uśredniania sygnału (jeżeli jest to wymagane),
  - okno służące do wizualizacji sygnałów (wykres czasowy/animacja),
  - element wczytujący odpowiednie kolumny z pliku tekstowego,
  - odrzucenie pierwszych n próbek z danymi.

#### Opis pliku z danymi tekstowymi (pliki z danymi są dostępne w archiwum data.zip):

Liczby zawarte w pliku są zapisem danych wyjściowych z sensorów. Każdy sensor podaje dane z częstotliwością 25 Hz (25 linii w pliku zawiera dane wyjściowe opisujące 1 sekundę ruchu sensora). Dane są grupowane w trójki (obejmują wskazania dotyczące każdej z 3 osi). W kolejnych kolumnach zapisane są od lewej odpowiednio:

- położenie katowe (roll, pitch, yaw) w stopniach,
- przyspieszenie ( $a_x$ ,  $a_y$ ,  $a_z$ ) w G ( $1G = 9,81m/s^2$ ),
- wskazania magnetometru ( $m_x$ ,  $m_y$ ,  $m_z$ ) - natężenie pola,
- wskazania żyroskopu ( $\omega_x$ ,  $\omega_y$ ,  $\omega_z$ ) w stopniach na sekundę.

Opis:

roll - obrót prawoskrętny względem osi X, pitch - względem osi Y, yaw - względem osi Z.

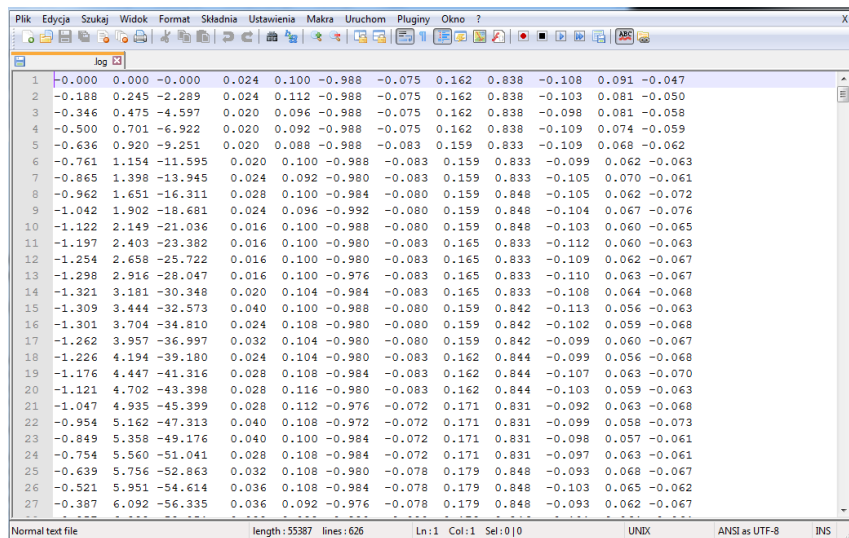
$a_x$  - przyspieszenie sensora wzdłuż osi X sensora,

$m_x$  - natężenie pola magnetycznego na osi X sensora,

$\omega_x$  - prędkość obrotowa względem osi X (obróć prawoskrętny względem osi X)

Początkowe próbki położenia katowego są błędne. Algorytm wyznaczający położenie katowe potrzebuje czasu, aby znaleźć położenie początkowe. Należy umożliwić, aby rysowanie danych na wykresie zaczynało się od próbki wybranej przez użytkownika (odrzucenie pierwszych n próbek).

Przykładowy plik z danymi:



Linia	roll	pitch	yaw	$a_x$	$a_y$	$a_z$	$m_x$	$m_y$	$m_z$	$\omega_x$	$\omega_y$	$\omega_z$
1	-0.000	0.000	-0.000	0.024	0.100	-0.988	-0.075	0.162	0.838	-0.108	0.091	-0.047
2	-0.188	0.245	-2.289	0.024	0.112	-0.988	-0.075	0.162	0.838	-0.103	0.081	-0.050
3	-0.346	0.475	-4.597	0.020	0.096	-0.988	-0.075	0.162	0.838	-0.098	0.081	-0.058
4	-0.500	0.701	-6.922	0.020	0.092	-0.988	-0.075	0.162	0.838	-0.109	0.074	-0.059
5	-0.636	0.920	-9.251	0.020	0.088	-0.988	-0.083	0.159	0.833	-0.109	0.068	-0.062
6	-0.761	1.154	-11.595	0.020	0.100	-0.988	-0.083	0.159	0.833	-0.099	0.062	-0.063
7	-0.865	1.398	-13.945	0.024	0.092	-0.980	-0.083	0.159	0.833	-0.105	0.070	-0.061
8	-0.962	1.651	-16.311	0.028	0.100	-0.984	-0.080	0.159	0.848	-0.105	0.062	-0.072
9	-1.042	1.902	-18.681	0.024	0.096	-0.992	-0.080	0.159	0.848	-0.104	0.067	-0.076
10	-1.122	2.149	-21.036	0.016	0.100	-0.988	-0.080	0.159	0.848	-0.103	0.060	-0.065
11	-1.197	2.403	-23.382	0.016	0.100	-0.980	-0.083	0.165	0.833	-0.112	0.060	-0.063
12	-1.254	2.658	-25.722	0.016	0.100	-0.980	-0.083	0.165	0.833	-0.109	0.062	-0.067
13	-1.298	2.916	-28.047	0.016	0.100	-0.976	-0.083	0.165	0.833	-0.110	0.063	-0.067
14	-1.321	3.181	-30.348	0.020	0.104	-0.984	-0.083	0.165	0.833	-0.108	0.064	-0.068
15	-1.309	3.444	-32.573	0.040	0.100	-0.988	-0.080	0.159	0.842	-0.113	0.056	-0.063
16	-1.301	3.704	-34.810	0.024	0.108	-0.980	-0.080	0.159	0.842	-0.102	0.059	-0.068
17	-1.262	3.957	-36.997	0.032	0.104	-0.980	-0.080	0.159	0.842	-0.099	0.060	-0.067
18	-1.226	4.194	-39.180	0.024	0.104	-0.980	-0.083	0.162	0.844	-0.099	0.056	-0.068
19	-1.176	4.447	-41.316	0.028	0.108	-0.984	-0.083	0.162	0.844	-0.107	0.063	-0.070
20	-1.121	4.702	-43.398	0.028	0.116	-0.980	-0.083	0.162	0.844	-0.103	0.059	-0.063
21	-1.047	4.935	-45.399	0.028	0.112	-0.976	-0.072	0.171	0.831	-0.092	0.063	-0.068
22	-0.954	5.162	-47.313	0.040	0.108	-0.972	-0.072	0.171	0.831	-0.099	0.058	-0.073
23	-0.849	5.358	-49.176	0.040	0.100	-0.984	-0.072	0.171	0.831	-0.098	0.057	-0.061
24	-0.754	5.560	-51.041	0.028	0.108	-0.984	-0.072	0.171	0.831	-0.097	0.063	-0.061
25	-0.639	5.756	-52.863	0.032	0.108	-0.980	-0.078	0.179	0.848	-0.093	0.068	-0.067
26	-0.521	5.951	-54.614	0.036	0.108	-0.984	-0.078	0.179	0.848	-0.103	0.065	-0.062
27	-0.387	6.092	-56.335	0.036	0.092	-0.976	-0.078	0.179	0.848	-0.093	0.062	-0.067

Zad 1.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota (sygnał z osi X). Program ma usuwać składową stałą z sygnału, wyznaczyć i wyświetlać (w GUI) przyspieszenie, prędkość, drogę. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 1a: outputRobotForwardB01.log

Zad 1b: outputRobotForwardB02.log

Zad 1c: outputRobotForwardB03.log

Zad 2.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru umieszczonego na sprężynie robota (ruch w pionie). Program ma wyznaczyć i wyświetlać (w GUI) przyspieszenie, prędkość, położenie sprężyny oraz współczynnik tłumienia drgań sprężyny (można przybliżyć funkcją  $e^{-k}$ ), gdzie  $k$  jest współczynnikiem tłumienia). W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 2a: outputSpring01.log

Zad 2b: outputSpring02.log

Zad 2c: outputSpring03.log

Zad 3.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota (sygnał z osi X,Y,Z - bez ruchu). Program ma wyznaczyć przyspieszenie ziemskie  $g$ , wyznaczyć i wyświetlać bieżącą długość wektora  $g$  oraz składowe X, Y, Z (należy użyć timera do animacji). W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 3a: outputRobotForwardA01.log

Zad 3b: outputRobotForwardB02.log

Zad 3c: outputStatic01.log

Zad 4.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota (sygnał z osi X,Y,Z - w ruchu). Program ma wyznaczyć przyspieszenie ziemskie  $g$ , wyznaczyć i wyświetlać długość wektora  $g$  oraz składowe X,Y,Z zmiany rzeczywistego przyspieszenia bez przyspieszenia ziemskiego (od sygnału należy odjąć wektor  $g$ ). W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wielu sygnałów jednocześnie).

Zad 4a: outputRobotForwardA03.log

Zad 4b: outputRobotForwardB01.log

Zad 4c: outputStatic02.log

Zad 5.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota (sygnał z osi X,Y,Z). Program ma wyznaczyć przyspieszenie ziemskie  $g$  oraz wyznaczyć sygnał, który sprawdza, czy wystąpił spadek swobodny (przyspieszenie ziemskie bliskie 0). Wyznaczyć i wyświetlać (w GUI) wszystkie występujące sygnały. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 5a: outputFreeFall01.log

Zad 5b: outputFreeFall02.log

Zad 6.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota poruszającego się do przodu (sygnał z osi X,Y,Z). Program ma obliczyć przyspieszenie ziemskie  $g$ , wykrywać i zliczać czas trwania postoiu robota i czas trwania ruchu robota. Należy wyznaczyć i wyświetlać (w GUI) wszystkie występujące sygnały oraz liczbę kroków (ruchu oraz postoiu). W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 6a: outputRobotForwardA02.log

Zad 6b: outputRobotForwardB02.log

Zad 6c: outputRobotForwardA01.log

Zad 7.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z żyroskopu umieszczonego na robocie mobilnym. Program ma wyświetlać (w GUI) aktualną prędkość kątową, sygnał uśredniony, położenie katowe. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 7a: outputRotateA01.log

Zad 7b: outputRotateB01.log

Zad 7c: outputRotateB02.log

Zad 8.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z żyroskopu umieszczonego na robocie mobilnym (na środku platformy robota o rozmiarach 10cm x 10 cm). Program ma wyświetlać (w GUI) aktualną prędkość kątową, aktualne położenie wierzchołków platformy. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 8a: outputPendulum01.log

Zad 8b: outputPendulum02.log

Zad 9.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje dane aktualnego położenia katowego (roll, pitch, yaw) robota mobilnego. Program ma wyświetlać (w GUI) aktualną linię horyzontu w rzucie z góry, z przodu i z boku. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie wymaganych elementów.

Zad 9a: outputRotateB01.log

Zad 9b: outputCatapult01.log

Zad 9c: outputRotateB02.log

Zad 10.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje dane aktualnego położenia katowego (roll, pitch, yaw) robota mobilnego. Program (w GUI) ma rysować prosty kompas i wyświetlać aktualny kierunek robota, zakładając, że na początku ruchu robot był skierowany na północ. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie.

Zad 10a: outputCatapult01.log

Zad 10b: outputRotateB01.log

Zad 10c: outputPendulum02.log

Zad 11.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje dane aktualnego położenia katowego (roll, pitch, yaw) robota mobilnego. Program ma wyświetlać (w GUI) aktualną trajektorię (drogę przebytą przez robota) przy założeniu, że robot porusza się ze stałą prędkością (wartość prędkości należy zmieniać za pomocą odpowiednich przycisków GUI).

Zad 11a: outputRobotForwardB01.log

Zad 11b: outputRobotForwardA02.log

Zad 11c: outputPendulumOrt01.log

Zad 12.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z żyroskopu umieszczonego na robocie mobilnym. Założono, że robot porusza się ze stałą prędkością (wartość prędkości można zmieniać za pomocą odpowiednich przycisków GUI). Program ma wyznaczyć i wyświetlać (w GUI) animowaną bieżącą trajektorię (drogę przebytą przez robota) - należy wykorzystać timer.

Zad 12a: outputRobotForwardA01.log

Zad 12b: outputRobotForwardB02.log

Zad 12c: outputPendulumOrt02.log

Zad 13.

Zaimplementować program zawierający GUI w środowisku WinAPI, który wczytuje, przetwarza i wizualizuje sygnał z akcelerometru robota jadącego pod górę (do przodu). Zakładając, że robot porusza się ze stałą prędkością bądź stał w miejscu należy wyznaczyć okres czasu w jakim robot się poruszał. Po wyznaczeniu wektora grawitacji g należy określić, pod jaką wysokość górę podjechał robot. Aktualna wysokość (podczas jazdy pod górę) oraz g mają być wyświetlane na wykresie. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie (należy umożliwić wyświetlanie wszystkich sygnałów jednocześnie).

Zad 13a: outputRobotForwardB01.log

Zad 13b: outputRobotForwardB02.log

Zad 14.

Zaimplementować program, który obsługuje funkcjonalności GUI oraz wczytuje i przetwarza sygnał z akcelerometru umieszczonego na wahadle. Wykorzystując wektor grawitacji (wyznaczony z danych z akcelerometru) oraz timer, należy wykreślić aktualne położenie wahadła (maksymalną wysokość przy każdym wachnięciu). Należy przyjąć długość wahadła 1m. Aktualna wartość g oraz czas trwania poprzedniego wachnięcia mają być wyświetlane na wykresie. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie.

Zad 14a: outputPendulum01.log

Zad 14b: outputPendulum02.log

Zad 14c: outputPendulumOrt01.log

Zad 15.

Zaimplementować program, który obsługuje funkcjonalności GUI oraz wczytuje i przetwarza dane aktualnego położenia katowego (roll, pitch, yaw) wahadła. Wykorzystując te dane oraz timer należy wykreślić aktualne położenie wahadła (maksymalną wysokość przy każdym wachnięciu). Należy przyjąć długość wahadła 0,5m. Aktualna kąt wychylenia wahadła oraz czas trwania poprzedniego wachnięcia mają być wyświetlane na wykresie. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie tych sygnałów na wykresie.

Zad 15a: outputPendulum01.log

Zad 15b: outputPendulum02.log

Zad 15c: outputPendulumOrt02.log

Zad 16.

Zaimplementować program, który obsługuje funkcjonalności GUI oraz wczytuje i przetwarza dane aktualnego położenia katowego (roll, pitch, yaw) obracającego się elementu. Należy wyznaczyć ile obrotów (w którą stronę) wykonał element. Wykorzystując timer należy animować aktualny kąt obrotu (jeśli element wykonał 2,5 obrotu powinna być wyświetlona końcowa wartość 900 stopni). W GUI należy dodać przyciski odpowiedzialne za wyświetlanie wymaganych elementów.

Zad 16a: outputRotateA01.log

Zad 16b: outputRotateA02.log

Zad 16c: outputRotateB01.log

Zad 16d: outputRotateB02.log

Zad 17.

Zaimplementować program, który obsługuje funkcjonalności GUI oraz wczytuje i przetwarza sygnał z akcelerometru oraz dane aktualnego położenia katowego (roll, pitch, yaw) umieszczonego na uderzanym elemencie. Należy wyznaczyć z jakiego kierunku element został uderzony oraz jakiego dostał przyspieszenia. Wykorzystując timer należy wyświetlić w jakiej chwili czas miało miejsce uderzenie. W GUI należy dodać przyciski odpowiedzialne za wyświetlanie wymaganych elementów.

Zad 17a: outputPunch01.log

Zad 17b: outputPunch02.log

#### **Dodatkowe materiały:**

Metody do wczytania danych z pliku tekstowego:

<http://www.cplusplus.com/reference/fstream/ifstream/>

Średnia krocząca (prosta):

[http://pl.wikipedia.org/wiki/Średnia\\_krocząca](http://pl.wikipedia.org/wiki/Średnia_krocząca)

Składowa stała:

[http://pl.wikipedia.org/wiki/Sygnał\\_okresowy](http://pl.wikipedia.org/wiki/Sygnał_okresowy)

Dla potrzeb projektu składowa stała można wyznaczyć poprzez odjęcie wartości średniej sygnału od każdej próbki sygnału. Można przyjąć wartość średnia jako średnią (kroczącą) z ostatnich 25 próbek.

Całkowanie numeryczne (metoda prostokątów)

[http://pl.wikipedia.org/wiki/Całkowanie\\_numeryczne](http://pl.wikipedia.org/wiki/Całkowanie_numeryczne)

Dokumentacja Microsoft Developer Network

[http://msdn.microsoft.com/en-us/library/ff818516\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ff818516(v=vs.85).aspx)

WIN API (po polsku)

<http://cpp0x.pl/kursy/Kurs-WinAPI-C++/167>