Sprawozdanie

Spis treści:

1. Treść zadania

```
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
```

2. Struktury danych

Program wykorzystuje tablice typu *double* zadeklarowane globalnie o rozmiarze 2500. Liczba ta jest dopasowana do wielkości układu współrzędnych.

`double tab[2500]`- przechowuje informacje wczytane z pliku
`double przysp[2500]`- przechowuje wartości przyspieszenia
`double predkosc[2500]`- przechowuje wartości prędkości
`double droga[2500]`- przechowuje wartości drogi
`double srednia[2500]`- przechowuje średnie wartości przyspieszenia z ostatnich 25 pomiarów

3. Wykorzystane wzory i metody

a) Wzory

$$1\ G = 9{,}81\ m/s^2$$

Wzór na prędkość:

$$v = v_0 + \int_{t_0}^{t_1} a(t)dt$$

Wzór na drogę:

$$s = s_0 + \int_{t_0}^{t_1} |v(t)dt|$$

Metoda lewych prostokątów do przybliżenia wartości całek:

$$\int_{x_0}^{x_n} f(x)dx = h\sum_{i=0}^{n-1} f(x_i)$$

$$h = \frac{x_n - x_0}{n} = 0{,}04s$$

n- liczba prostokątów

Prosta średnia ruchoma z ostatnich 25 prób:

$$SMA = \frac{a_0 + a_1 + \cdots + a_{24}}{25}$$

b) Rysowanie

- Najpierw decydujemy o liczbie rysowanych punktów, dzieląc liczbę danych przez *t*. Wartość podziałki na osi X można obliczyć ze wzoru $x_0 = 0{,}04 * t * 10$ (10 to liczba pikseli w podziałce; 0,04 to okres pomiędzy pomiarami).
- Następnie, rysujemy co *t* punkt pomnożony przez *y* (współczynnik amplitudy) i odjęty od 385 (to wysokość osi X). Wartość podziałki na osi Y można obliczyć ze wzoru $y_0 = \frac{10}{y}$ (10 to liczba pikseli w podziałce).

4. Obsługa programu

Wygląd po uruchomieniu:

Rysunek 1

Rysunek 2

Program posiada 4 kategorie przycisków:

a) Wybór amplitudy
   - y=0,1 -> oznacza, że jedna podziałka na pionowej osi wynosi 0,1 danej jednostki (m/s dla prędkości, m dla drogi, m/s^2 dla przyspieszenia)
   - y=0,2 -> oznacza, że jedna podziałka na pionowej osi wynosi 0,2 danej jednostki (m/s dla prędkości, m dla drogi, m/s^2 dla przyspieszenia)

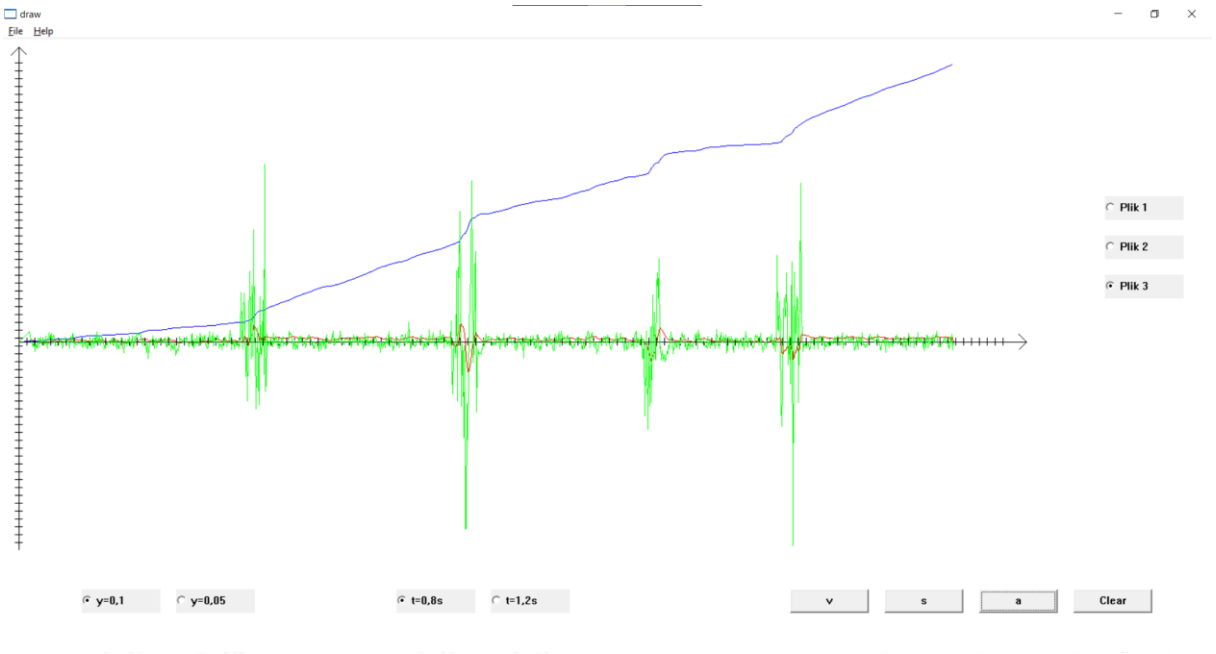
b) Wybór przedziału czasowego
   - t=0,8s -> oznacza, że jedna podziałka na poziomej osi wynosi 0,8 sekundy
   - t=1,2s -> oznacza, że jedna podziałka na poziomej osi wynosi 1,2 sekundy

c) Wybór pliku
   - Plik 1 -> otwiera plik outputRobotForwardB01.log
   - Plik 2 -> otwiera plik outputRobotForwardB02.log
   - Plik 3 -> otwiera plik outputRobotForwardB03.log

d) Wybór rysowania funkcji
   - v -> rysuje wykres prędkości kolorem czerwonym
   - s -> rysuje wykres drogi kolorem niebieskim
   - a -> rysuje wykres przyspieszenia kolorem zielonym
   - Clear -> czyści ekran, pozostawia sam układ współrzędnych

Na początku należy wybrać amplitudę, czas i plik, by móc narysować wykres. Każda zmiana amplitudy, czasu lub pliku (a także użycie przycisku Clear) zmienia wygląd okna na ten z rysunku 2. Zaleca się używanie programu w trybie pełnoekranowym.

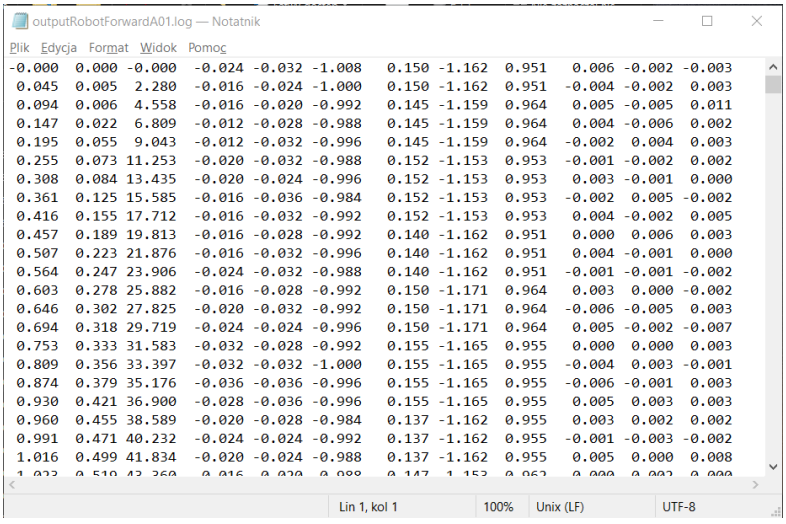
Wykresy prędkości, drogi i czasu mogą się na siebie nakładać.



Rysunek 3

5. Format danych wejściowych

Program wczytuje dane z pliku tekstowego, który zawiera 12 kolumn liczb typu *double*. Program ten używa tylko kolumny 4, która musi zawierać przyspieszenie sensora wzdłuż osi X w G odczytywane co 0,04 sekundy.



Rysunek 4

Program jest w stanie zwizualizować plik zawierający do 2500 wierszy. W przypadku większej ilości danych wczytane zostanie jedynie 2500 początkowych wierszy.

Pliki z danymi znajdują się w folderze *draw*.

6. Ważniejsze zmienne

   a) Zmienne globalne

      `fstream plik`- służy do otwarcia i odczytania danych z wybranego pliku

      `int ile`- przechowuje liczbę danych

   b) Zmienne lokalne

      `int odrz`- wykorzystywana w funkcji `wpis2()`, jest licznikiem w pętli wypełniającą tablicę `srednia[]`

      `double pred, drog`- suma poprzednich policzonych wartości- odpowiednio prędkości i drogi, wykorzystywana w funkcji `wpis2()`

      `double suma`- suma poprzednich 25 wartości przyspieszenia, wykorzystywana w funkcji `wpis2()`

      `int ilep`- określa ile punktów zostanie narysowanych, wykorzystywana w funkcjach `RysPrzysp()`, `RysDroga()`, `RysPredkosc()`

      `int pocz`- określa, od którego punktu ma zacząć się rysowanie, wykorzystywana w funkcjach `RysPrzysp()`, `RysDroga()`, `RysPredkosc()`

      `UINT a, b, c, d`- służą do sprawdzania, który przycisk radiowy jest wciśnięty, wykorzystywane w funkcji `WndProc()`

7. Opisy funkcji

`void rysuj(HDC hdc)`
Funkcja, która rysuje układ współrzędnych. Parametry: `hdc`- uchwyt do kontekstu urządzenia

`void wpis(int ktory)`
Odczytuje dane z pliku i zapisuje je do tablicy `tab` i liczy ilość wierszy w pliku, jeżeli liczba wierszy przekracza 2500 to pomija resztę danych. Parametry: `ktory`- określa, który plik należy wczytać

`void wpis2()`
Liczy średnią z ostatnich 25 danych przyspieszeń, zamienia jednostkę z G na m/s^2, a następnie zapisuje wyniki w tablicy `srednia`. Następnie liczy wartości przyspieszenia, prędkości i drogi bez wartości średniej i zapisuje je kolejno do tablic `przysp`, `predkosc`, `droga`. Parametry: brak

`void RysPredkosc(HDC hdc, int y, int t)`
Rysuje wykres prędkości od czasu. Parametry : `hdc`- uchwyt do kontekstu urządzenia `y`- odpowiada za skalę amplitudy wykresu `t`- odpowiada za zmianę wartości podziałki czasowej

`void RysDroga(HDC hdc, int y, int t)`
Rysuje wykres drogi od czasu. Parametry: `hdc`- uchwyt do kontekstu urządzenia `y`- odpowiada za skalę amplitudy wykresu `t`- odpowiada za zmianę wartości podziałki czasowej

`void RysPrzysp(HDC hdc, int y, int t)`
Rysuje wykres przyspieszenia od czasu. Parametry: `hdc`- uchwyt do kontekstu urządzenia `y`- odpowiada za skalę amplitudy wykresu `t`- odpowiada za zmianę wartości podziałki czasowej

`void czysc(HWND hWnd, HDC& hdc, PAINTSTRUCT& ps, RECT* drawArea)`
Czyści ekran, a następnie rysuje układ współrzędnych. Parametry: `hWnd`- uchwyt okna `hdc`- uchwyt do kontekstu urządzenia `ps`- struktura używana przez aplikację do rysowania w programie `drawArea`- wskaźnik na współrzędne lewego górnego i prawego dolnego narożnika prostokąta

`void repaintWindow(HWND hWnd, HDC &hdc, PAINTSTRUCT &ps, RECT *drawArea , int y, int t, int ktorywykres)`
`hWnd`- uchwyt okna `hdc`- uchwyt do kontekstu urządzenia `ps`- struktura używana przez aplikację do rysowania w programie `drawArea`- wskaźnik na współrzędne lewego górnego i prawego dolnego narożnika prostokąta `y`- odpowiada za skalę amplitudy wykresu `t`- odpowiada za zmianę wartości podziałki czasowej `ktorywykres`- decyduje, który wykres ma być narysowany

`int APIENTRY _tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)`
Funkcja główna. Parametry: `hInstance`- uchwyt aplikacji

hPrevInstance- uchwyt do poprzedniego wystąpienia aplikacji lpCmdLine- zawiera linię poleceń, z jakiej został uruchomiony program nCmdShow- określa stan okna programu

ATOM MyRegisterClass(HINSTANCE hInstance)
Zawiera informacje o klasie okna. Parametry: hInstance- uchwyt aplikacji

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
Tworzy przyciski. Parametry: hInstance- uchwyt aplikacji nCmdShow- określa stan okna programu

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
Definiuje, co program ma wykonać w przypadku naciśnięcia przycisków. Parametry: hWnd- uchwyt okna message- kod komunikatu