Realizacja ćwiczenia 3

1. Wstęp

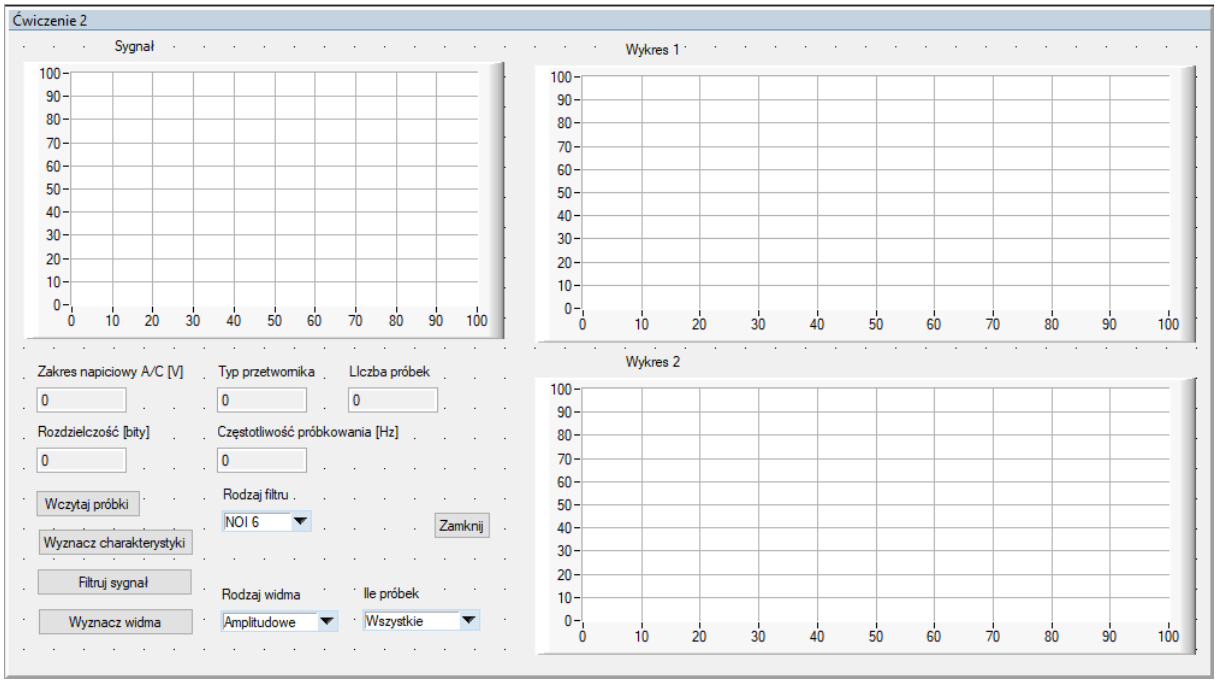
Celem ćwiczenia jest praktyczne poznanie zagadnień związanych z filtracją cyfrową oraz ich **samodzielna** implementacją w języku ANSI C przy wykorzystaniu środowiska programowania LabWindows/CVI firmy National Instruments.

W tym dokumencie omówię szerzej realizację kolejnych zadań zdefiniowanych w instrukcji do ćwiczenia 3. Zakładam, że numery podgrup zastają takie same, jak w ćwiczeniu 2. Tym razem każdej z podgrup przydzielone są po jednym filtrze NOI i SOI zgodnie z tabelą 1 znajdująca się z instrukcją do ćwiczenia 3. Dla przykładu, podgrupa 1 ma do zaimplementowania dolnoprzepustowy filtr NOI Butterwortha rzędu 8 oraz dolnoprzepustowy filtr SOI z oknem Hanninga rzędu 35. Każda z podgrup ma również przypisany inny sygnał do filtracji, którego próbki dostępne są na stronie http://staff.uz.zgora.pl/mkoziol/dydaktyka/2020_lato_cps_21ib-sp/cps.html.

Zanim przejdą państwo do realizacji poszczególnych zadań, dobrze jest dostosować interfejs użytkownika programu wykonanego w ramach ćwiczenia 0 do potrzeb programu tworzonego w tym ćwiczeniu.

W tym celu w środowisku LabWindows należy przejść do edycji pliku UIR i umieścić na panelu tworzonego programu dodatkowo następujące elementy:

- 2 elementy *Graph*,
- 3 elementy *Square Command Button*,
- 3 element *Ring*.

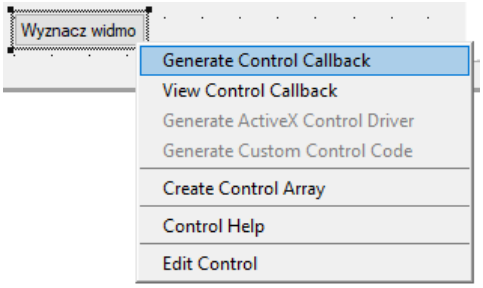


Rys. 1. Przykładowy wygląd okna programu w ćwiczeniu 3

Dwa elementy *Graph* należy umieścić jeden pod drugim, jak na rysunku 1. Będą one służyły do prezentacji różnych wykresów w zależności od wykonanego przez program działania. Z tego też powodu nadano im neutralne nazwy *Wykres 1* i *Wykres 2*. Żadnemu z elementów *Graph*, przynajmniej na razie, nie przypisujemy funkcji callback. Zrobią to państwo dopiero podczas realizacji zadania 2.

Aby podczas implementacji w programie kodu związanego z poszczególnymi zadaniami nie modyfikować funkcji `WczytajProbki`, na panelu użytkownika programu stworzonego w ramach ćwiczenia 0 przewidziano trzy kalwisze *Square Command Button*, którym należy przypisać osobne funkcje callback. Na rysunku 1 są to klawisze z napisem *Wyznacz charakterystyki*, *Filtruj sygnał* i *Wyznacz widma*. Klawisze kładziemy dokładnie tak samo, jak na zajęciach wprowadzających (patrz plik używany przy realizacji pierwszych zajęć laboratoryjnych). W tym samym pliku znajduje się również instrukcja uzupełniania niezbędnych ustawień dla takiego klawisza. Ponieważ klawisze te będą uruchamiać odpowiednie procesy, proszę również wymyśleć odpowiednie nazwy dla stałych je reprezentujących i przypisanych im funkcji callback, aby kojarzyły się z uruchamianym procesem.

Aby do kodu programu dodać szkielet kodu funkcji callback związanej z położonym klawiszem, należy przy kursorze umieszczonym nad tym klawiszem kliknąć prawy klawisz myszy i w otwartym menu podręcznym wybrać pozycję *Generate Control Callback*. Pokazano to na rysunku 2.



Rys. 2. Widok menu podręcznego klawisza

W efekcie tego działania w kodzie programu pojawi się szkielet funkcji callback, który przedstawiono na rysunku 3.

```
123 int CVICALLBACK NazwaFunkcji (int panel, int control, int event,
124                               void *callbackData, int eventData1, int eventData2)
125 {
126   switch (event)
127   {
128     case EVENT_COMMIT:
129
130       break;
131   }
132   return 0;
133 }
134
```

Rys. 3. Wygenerowany w kodzie programu szkielet funkcji callback klawisza z interfejsu użytkownika

Oczywiście `NazwaFunkcji` będzie takim ciągiem znakowym, jaki został wpisany w polu *Callback funkction* okna parametrów kawisza *Command Button*.

Jak zalecałem wcześniej, szkielet tej funkcji warto zmienić do postaci pokazanej na rysunku 4, gdzie zamiast instrukcji wyboru `switch-case` zastosowano instrukcję `if`.

```
123 int CVICALLBACK NazwaFunkcji (int panel, int control, int event,
124                               void *callbackData, int eventData1, int eventData2)
125 {
126   if (event == EVENT_COMMIT)
127   {
128       // Kod związany z wyznaczaniem widma, nakładaniem okien
129       // i uzupełnianiem zerami.
130
131   }
132   return 0;
133 }
134
```

Rys. 4. Zmodyfikowany szkielet funkcji callback nowego klawisza z interfejsu użytkownika

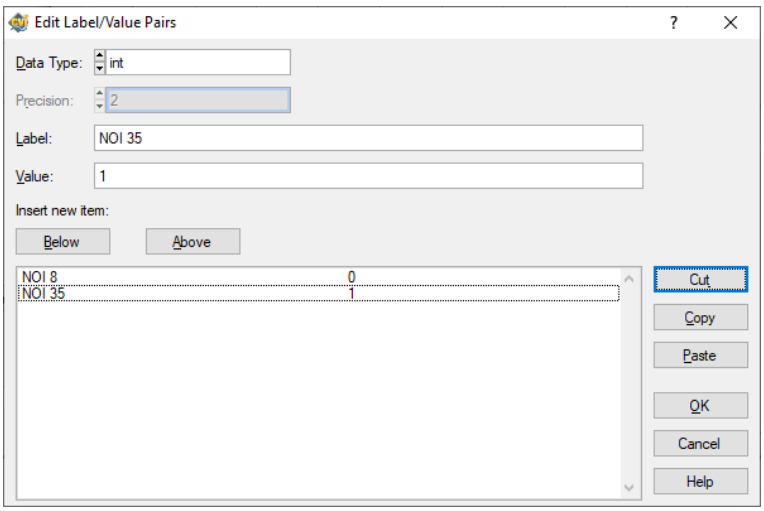
Zakłada się, że użytkownik programu do przetworzenia próbek wczytanego sygnału może wybrać na interfejsie użytkownika jeden z dwóch filtrów. Do wyboru takiego filtru należy wykorzystać jeden z dwóch elementów *Ring*. Na rysunku 1 jest on opisany nazwą *Rodzaj filtru*. Jego rozwijaną listę należy uzupełnić tak, aby w programie była możliwość wyboru jednego z zadanych do implementacji filtrów.

2. Zadanie 1

Informacje pomocne w realizacji zadania 1 zostały przedstawione na slajdach z wykładu „Charakterystyki widmowe układów dyskretnych”. Implementacja kodu związanego z tym zadaniem powinna być zrealizowana w funkcji callback związanej z klawiszem *Wyznacz charakterystyki*.

Aby wyznaczyć charakterystyki konieczna jest znajomość współczynników filtra. Dlatego w początkowej części kodu funkcji callback związanej z klawiszem *Wyznacz charakterystyki* należy odczytać z elementu *Ring* prezentującego dostępne filtry, filtr aktualnie wybrany przez użytkownika i na podstawie tej informacji wywołać odpowiednią funkcję, która wyznaczy jego współczynniki. Funkcje te są opisane w punkcie 4.3 instrukcji do ćwiczenia 3. Wyznaczone współczynniki mogą być zapisywane do tablic o rozmiarze deklarowanym statycznie w kodzie programu.

Dla przykładu, jeśli najwyższy rząd filtra to 35, a rozwijana lista filtrów została zdefiniowana jak na rysunku 5 i nadano temu elementowi *Ring* nazwę *Constant name* postaci FILTRY, to fragment kodu realizujący opisane powyżej działania może mieć postać, jak na listingu 1.



Rys. 5. Okno uzupełniania listy w elemencie *Ring* dla przedstawianego przykładu

Listing 1. Przykładowy kod funkcji callback klawisza odpowiedzialnego za wyznaczanie charakterystyk amplitudowych filtrów

```
int CVICALLBACK CharakterystykiFiltrow (int panel, int control, int event,
                                        void *callbackData, int eventData1,
                                        int eventData2)
{
  int filtr;
  double wspA[36], wspB[36];
  double chakaAmp[L_PUNKTOW+1], chakaFaz[L_PUNKTOW+1], f[L_PUNKTOW+1];

  if (event == EVENT_COMMIT)
  {
    // Odczytanie z listy, dla którego filtru będą wyznaczane charakterystyki
    GetCtrlVal (panelHandle, PANEL_FILTRY, &filtr);
    switch (filtr)
    {
      case 0:
        // Wyznaczenie współczynników filtru o numerze 0 na liście
        // Zapianie do jakiejś zmiennej rzędu filtru
        break;
      case 1:
        // Wyznaczenie współczynników filtru o numerze 1 na liście
        // Zapianie do jakiejś zmiennej rzędu filtru
        break;
    }
    if (filtr == 0)
    {
      // Wyznaczenie punktów charakterystyk filtrów NOI biorąc pod uwagę wartość
      // zmiennej z rzędem filtru
    }
    if (filtr == 1)
    {
      // Wyznaczenie punktów charakterystyk filtrów SOI biorąc pod uwagę wartość
      // zmiennej z rzędem filtru
    }

    // Prezentacja charakterystyk na elementach Graph

  }
  return 0;
}
```

Następnie w miejscach komentarzy należy zaimplementować kod, który w zależności od rodzaju filtra (NOI lub SOI) wyznaczy punkty charakterystyki amplitudowej i fazowej. Wykres charakterystyki amplitudowej należy przedstawić na elemencie *Graph* o nazwie *Wykres 1*, a wykres charakterystyki fazowej na elemencie Graph o nazwie *Wykres 2*.

Po wykonaniu tego zadania proszę przesłać na mój adres e-mail m.koziol@g.wiea.uz.zgora.pl zrzut ekranu pokazujący wykresy charakterystyk zadanych filtrów. W odpowiedzi prześlę informację, czy wynik, który państwo uzyskali jest poprawny, czy nie.

3. Zadanie 2

Jak zrealizować zadanie 2 jest w całości wyjaśnione w instrukcji do ćwiczenia 3.

4. Zadanie 3

Informacje pomocne w realizacji zadania 3 zostały przedstawione na slajdach z wykładu „Filtry cyfrowe”. Implementacja kodu związanego z tym zadaniem powinna być zrealizowana w funkcji callback związanej z klawiszem *Filtruj sygnał*.

Podobnie jak w zadaniu 1, do przeprowadzenia filtracji konieczna jest znajomość współczynników filtra. Dlatego na początku funkcji callback należy odczytać z elementu *Ring* prezentującego dostępne filtry aktualnie wybrany przez użytkownika i na podstawie tej informacji wywołać odpowiednią funkcje, która wyznaczy jego współczynniki. Funkcje te są opisane w punkcie 4.3 instrukcji do ćwiczenia 3. Wyznaczone współczynniki mogą być zapisywane do tablic o rozmiarze deklarowanym statycznie w kodzie programu.

Po wyznaczeniu próbek sygnału wyjściowego z filtra należy na elemencie *Graph* o nazwie *Wykres 1* przedstawić próbki sygnału przed filtracją, a na elemencie *Graph* o nazwie *Wykres 2* próbki sygnału po filtracji.

Po wykonaniu tego zadania proszę przesłać na mój adres e-mail m.koziol@g.wiea.uz.zgora.pl zrzut ekranu pokazujący wykresy sygnałów po filtracji przez każdy z zadanych filtrów. W odpowiedzi prześlę informację, czy wynik, który państwo uzyskali jest poprawny, czy nie.

5. Zadanie 4

W przypadku tego zadania program powinien wyznaczać widma amplitudowe i fazowe zarówno sygnału wejściowego, jak i wyjściowego. W sumie są to 4 wykresy. W proponowanym przeze mnie podejściu z punktu widzenia interfejsu użytkownika mamy dostępne tylko dwa elementy *Graph*. Dlatego zaproponowałem również wprowadzenie drugiego elementu Ring, za pomocą którego użytkownik programu może wybrać, czy chce uwidocznić na tych dwóch elementach *Graph* tylko widma amplitudowe sygnału wejściowego i wyjściowego, czy też tylko widma fazowe tych dwóch sygnałów.

Po wyznaczeniu widm, widmo amplitudowe lub fazowe (w zależności od wyboru na elemencie *Rodzaj widma*) sygnału przed filtracją należy prezentować na elemencie *Graph* o nazwie *Wykres 1*, natomiast na elemencie *Graph* o nazwie *Wykres 2* widmo amplitudowe lub fazowe (w zależności od wyboru na elemencie *Rodzaj widma*) sygnału po filtracji.

Proszę pamiętać, że sama implementacja wyznaczania widm amplitudowych i fazowych została uproszczona poprzez użycie gotowej funkcji bibliotecznej. Jej użycie w programie zostało opisane w punkcie 4.4 instrukcji do ćwiczenia 3. Proszę również nie implementować uzupełniania zerami czy też nakładania okien. W tym programie nie będą one potrzebne.

Wspomniana funkcja biblioteczna umożliwia określenie, dla jakiej części próbek sygnału wyznaczane jest widmo. Dlatego na interfejsie użytkownika pojawił się trzeci element *Ring*, o nazwie *Ile próbek*. W zależności od wyboru elementu z dwupozycyjnej listy (*Wszystkie* lub *Druga połowa*) widmo w programie powinno być wyznaczane odpowiednio na bazie wszystkich próbek sygnałów lub tylko próbek z drugiej połowy. Wyznaczanie widma na podstawie drugiej połowy widma pozwoli na poprawną realizację ostatniego punktu wymaganego w sprawozdaniu z ćwiczenia, tj. analizy poprawności przeprowadzonej filtracji w oparciu o dziedzinę częstotliwości.