POLITECHNIKA POZNAŃSKA

WYDZIAŁ AUTOMATYKI, ROBOTYKI I ELEKTROTECHNIKI

INSTYTUT ROBOTYKI I INTELIGENCJI MASZYNOWEJ

ZAKŁAD STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

SPRAWOZDANIE

SYSTEMY MIKROPROCESOROWE (LABORATORIUM)
[WARIE_2021-22_AIR_DZ_1_5_D_LUCZAK_21/22]

BIBLIOTEKA CMSIS
- OPERACJE MACIERZOWE, FILTRY CYFROWE FIR/IIR

(TEMAT ZAJĘĆ)

KAROL DĘBSKI

(AUTOR I: KAROL.DEBSKI@STUDENT.PUT.POZNAN.PL)

FORMA ZAJĘĆ: LABORATORIUM

PROWADZĄCY:

DR INŻ. DOMINIK ŁUCZAK

DOMINIK.LUCZAK@PUT.POZNAN.PL

POZNAŃ 13-*12-2021* 9-45

(*DATA I GODZINA ZAJĘĆ*)

Spis treści

Zadanie #1

1.1 Specyfikacja

Program implementuje dyskretną postać przestrzeni stanów dla obiektu oscylacyjnego. Do weryfikacji poprawnego działania programu zostanie użyty SWV.

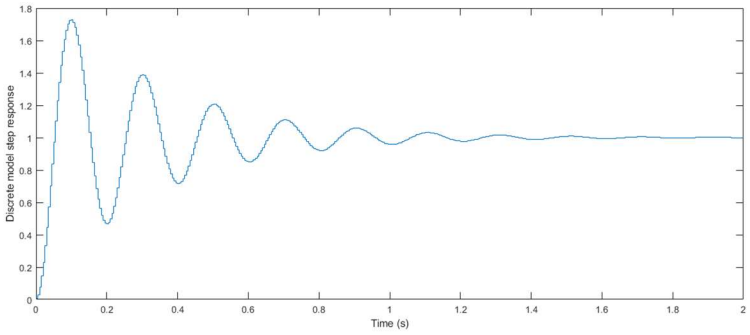
1.2 Implementacja

Listing 1 Callback dla timera

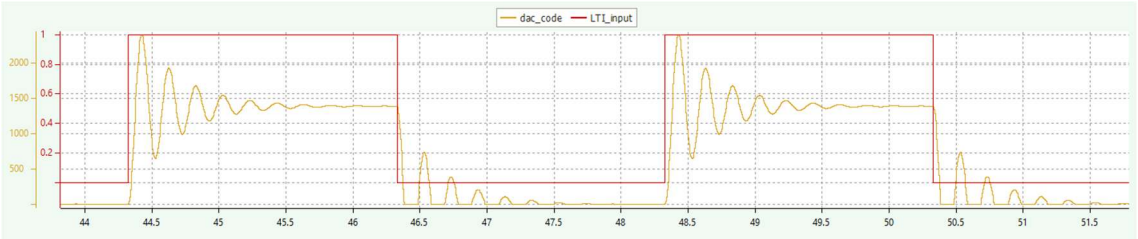
```
void HAL TIM PeriodElapsedCallback(TIM HandleTypeDef *htim)
{
  if(htim == &htim10 ){
            arm_status status;
            temp1.numRows=2; temp1.numCols=1; temp2.numRows=2;temp2.numCols=1;
            status=arm_mat_mult_f32(&Ad,&Xk,&temp1);
            status=arm mat scale f32(&Bd,LTI input,&temp2);
            status=arm mat add f32(&temp1,&temp2,&Xknext);
            temp1.numRows=1; temp1.numCols=1; temp2.numRows=1;temp2.numCols=1;
            status=arm_mat_mult_f32(&Cd,&Xk,&temp1);
            status=arm_mat_scale_f32(&Dd,LTI_input,&temp2);
            status=arm_mat_add_f32(&temp1,&temp2,&Y);

            memcpy(Xk.pData,Xknext.pData,sizeof(float)*Xk.numCols*Xk.numRows);
            float signal=Y.pData[0];
            LTI_input=(t<2.0)? 1.0 : 0.0;
            t=(t<4.0)?t+dt:0.0;
            dac code=(uint32 t)(signal*4095.0/2.95);
            HAL DAC SetValue(&hdac, DAC CHANNEL 2, DAC ALIGN 12B R, dac code);
  }
}
```

W callbacku dla timera który wywoływany jest co 5 ms następuje wyliczenie wyjścia dyskretnego modelu a następnie generowane jest ono na pinie DAC.

1.3 Wynik testów



Rys. 1



Rys. 2

1.4 Wnioski

Odpowiedzi z MATLABa i SWV pokrywają się – program działa poprawnie.

Zadanie #2

1.1 Specyfikacja

Program implementuje filtr dolnoprzepustowy IIR. Do weryfikacji poprawnego działania programu zostanie użyty SWV.

1.2 Implementacja

Listing 2 Callback dla przerwania na ADC

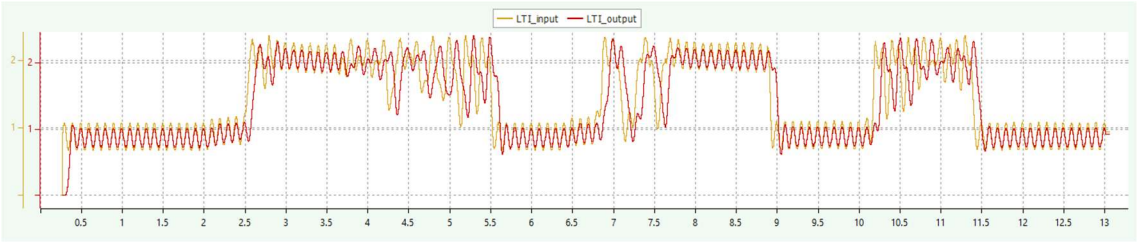
```
void HAL ADC ConvCpltCallback(ADC HandleTypeDef * hadc){
          if(hadc == &hadc1){
                    voltage = 2.950 * (ADC raw)/4096.0;
          }
}
```

Listing 3 Callback dla przerwania na timerach

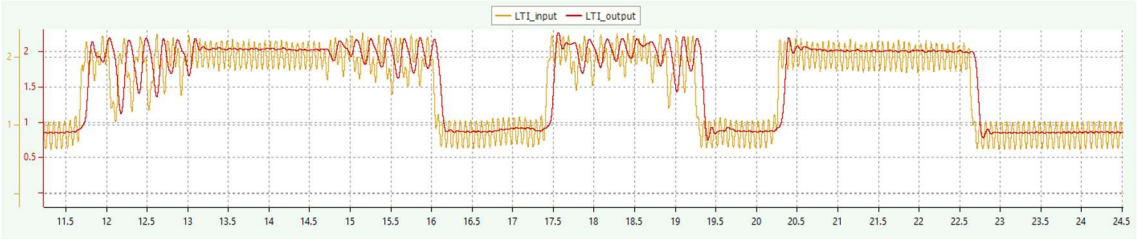
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

  if(htim == &htim10 ){
            faza=fmod((n*2*3.14*f/fs),(2*3.14));
            n++;
            LTI_input=voltage+0.2*sin(faza);
            arm_biquad_cascade_df1_f32(&iir_filter,&LTI_input,&LTI_output,1);
            LTI_output=LTI_output*iir_gain;
            t=(t<10.0)? t+dt : 0.0;
            HAL ADC Start DMA(&hadc1, &ADC raw, 1);
  }
}
```

W callbacku dla timera najpierw dodawany jest szum w postaci sinusa do wejścia filtru. Następnie obliczane jest wyjście filtru a dalej wywoływane jest żądanie konwersji z ADC.

1.3 Wynik testów



Rys. 2 Wejście z dodanym szumem o f=10 Hz i wyjście filtru



Rys. 3 Wejście z dodanym szumem o f=15 Hz i wyjście filtru

1.4 Wnioski

Zgodnie z oczekiwaniami filtr osłabia składowe o wyższych częstotliwościach niż 10 Hz – program działa poprawnie.

Zadanie #3

1.1 Specyfikacja

Program implementuje filtr dolnoprzepustowy FIR. Do weryfikacji poprawnego działania programu zostanie użyty SWV.

1.2 Implementacja

Listing 4 Callback dla przerwania na ADC

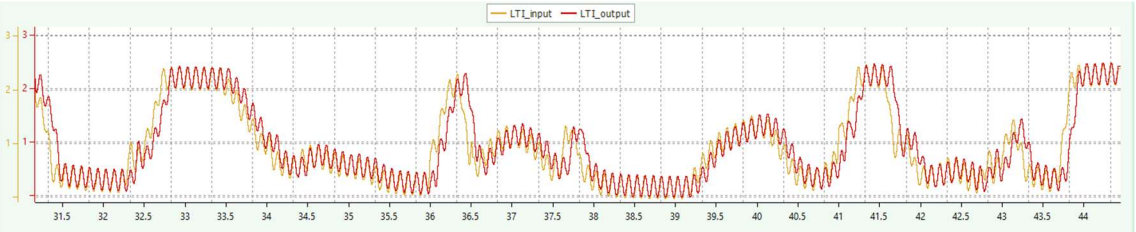
```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef * hadc){
          if(hadc == &hadc1){
                    voltage = 2.950 * (ADC raw)/4096.0;
          }
}
```

Listing 5 Callback dla przerwania na timerach

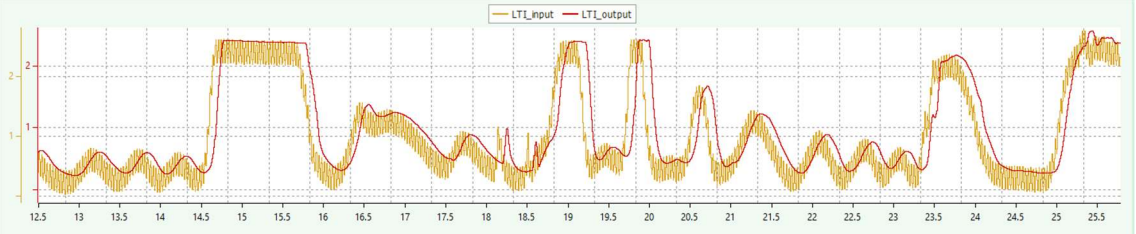
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if(htim == &htim10 ){
            LTI input=voltage+0.2*arm sin f32(2.0*PI*30.0*t);
            arm fir f32(&fir filter,&LTI input,&LTI output,1);
            t=(t<10.0)? t+dt :0.0;
            HAL_ADC_Start_DMA(&hadc1, &ADC_raw, 1);

  }
}
```

Cyklicznie co 0.005 sekundy wywoływany jest callback dla timera który oblicza wyjście filtru na podstawie odczytu z ADC a także dodanego szumu.

1.3 Wynik testów



Rys. 4 Wejście z dodanym szumem o f=10 Hz i wyjście filtru



Rys. 5 Wejście z dodanym szumem o f=30 Hz i wyjście filtru

1.4 Wnioski

Filtr zgodnie z założeniami osłabia częstotliwości powyżej 10 Hz – program działa poprawnie

Podsumowanie

Filtry dobrze filtrują.