L.EEC025 - Fundamentals of Signal Processing (FunSP)

2022/2023 – 1st semester

Week04, 03 Oct 2022

Objectives:

-understanding sampling and reconstruction

- **understanding that filtering effects depend of the frequency response characteristics of the filter (either analog, or discrete-time)**
- **understanding the different components in a complete signal processing chain**
- **understanding basic effects of sampling and reconstruction on sinusoidal, rectangular, triangular, and sawtooth waves**

DSP Education Kit

LAB 3

Understanding sampling and reconstruction

Issue 1.0

Contents

1 Introduction

1.1 Lab overview

The STM32F746G Discovery board is a low-cost development platform featuring a 212 MHz Arm Cortex-M7 floating-point processor. It connects to a host PC via a USB A to mini-b cable and uses the ST-LINK/V2 in-circuit programming and debugging tool. The Keil MDK-Arm development environment, running on the host PC, enables software written in C to be compiled, linked, and downloaded to run on the STM32F746G Discovery board. Real-time audio I/O is provided by a Wolfson WM8994 codec included on the board.

This laboratory exercise introduces the use of the STM32F746G Discovery board and several of the procedures and techniques that will be used in subsequent laboratory exercises.

2 Requirements

To carry out this lab, you will need:

- An STM32F746G Discovery board
- A PC running Keil MDK-Arm
- MATLAB
- An oscilloscope
- Suitable connecting cables
- An audio frequency signal generator
- Optional: External microphone, although you can also use the microphones on the board

2.1.1 STM32F746G Discovery board

An overview of the STM32F746G Discovery board can be found in the Getting Started Guide.

3 Basic Digital Signal Processing System

A basic DSP system that is suitable for processing audio frequency signals comprises a digital signal processor and analogue interfaces as shown in Figure 1. The STM32F746G Discovery board provides such a system, using a Cortex-M7 floating point processor and a WM8994 codec.

The term codec refers to the *coding* of analogue waveforms as digital signals and the *decoding* of digital signals as analogue waveforms. The WM8994 codec performs both the Analogue to Digital Conversion (ADC) and Digital to Analogue Conversion (DAC) functions shown in Figure 1.

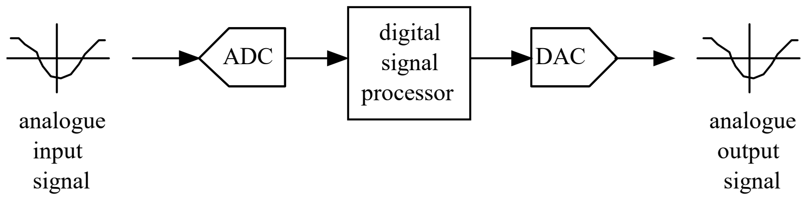

Figure 1: Basic digital signal processing system

Program code may be developed, downloaded, and run on the STM32F746G Discovery board using the *Keil MDK-Arm* integrated development environment. You will not be required to write C programs from scratch, but you will learn how to compile, link, download, and run the example programs provided, and in some cases, make minor modifications to their source files.

You will learn how to use a subset of the features provided by MDK-Arm in order to do this (using the full capabilities of MDK-Arm is beyond the scope of this set of laboratory exercises). The emphasis of this set of laboratory exercises is on the digital signal processing concepts implemented by the programs.

Most of the example programs are quite short, and this is typical of real-time DSP applications. Compared with applications written for general purpose microprocessor systems, DSP applications are more concerned with the efficient implementation of relatively simple algorithms. In this context, efficiency refers to speed of execution and the use of resources such as memory.

4 Filtering a square wave using two filters

4.1 Program operation

This section of the LAB guide is preparatory for the lab experiments suggested in Section 4.2. It is meant to highlight that a discrete-time filter influences the shape of waveform at its output as a consequence of the two parts of its frequency response: the magnitude of the frequency response, and the phase of the frequency response. Although at this point Students are not required to explain exactly how that happens (that will be treated soon in one of the lectures when the concept of ***group delay*** is discussed), Students should be just aware of the fact that two filters having the exact same magnitude frequency response, but having different phase frequency responses, produce at their output different wave shapes when the two filters are excited with the same input signal.

The following Matlab code (`FPS_lab03.c`) is available on Moodle and generates a discrete-time square wave and then filters it using two filters whose impact on the signal you should observe.

```
%
% FPS/PSF lab 03
% Author: AJF, 28 Sept 2022
%

% sampling frequency (FS) and frequency of square wave (Freq)
FS=48000; TS=1/FS; Freq=5E2;

% time vector
t=[0:TS:8E-3];
x = square(2*pi*Freq*t);
subplot(3,2,1)
plot(t,x)
xlabel('Time (s)'); ylabel('Amplitude'); title('Original wave')
axis([1.5E-3 7.5E-3 -1.2 1.2])

subplot(3,2,2)
plot(t,x)
xlabel('Time (s)'); title('Original wave')
axis([1.5E-3 7.5E-3 -1.2 1.2])


% filter order
order = 80;
% filter length
N=order+1;
n=[0:(N-1)];
% group delay, this will be discussed in a forthcoming lecture
atraso=(N-1)/2;


% this is a filter design method that we will discuss in a few weeks
% note the Matlab defnition: sinc(x) = sin(pi*x) / (pi*x)
hfilter = sinc(0.25*(n-atraso));
win=hamming(N).'; hfilter=hfilter.*win;
```

```
[H W]=freqz(hfilter, 1, 'whole');
subplot(3,2,3)
% plot the magnitude of the frequency response of filter 1
plot(W/pi, abs(H));
xlabel('\omega/\pi'); ylabel('Magnitude')
title('Frequency response 1')

% find the output of filter 1
y=conv(x, hfilter);
subplot(3,2,5)
plot(t, y(1:length(t)))
xlabel('Time (s)'); ylabel('Amplitude'); title('Filtered wave 1')
axis([1.5E-3 7.5E-3 -5.2 5.2])

% you don't need to understand this piece of code for now, you
% just need to understand that filter 2 is a transformation
% of filter 1, in what sense ? (hint: check its frequency response)
% all you need to know for now is that we have two filters whose
% impulse responses are known
hroots=roots(hfilter); gain=sum(hfilter);
for k=1:length(hroots)
    mult=abs(hroots(k));
   if( mult > 1 )
        hroots(k) = 1/conj(hroots(k));
   end

end
hfilter2=poly(hroots);
hfilter2 = gain*hfilter2/sum(hfilter2);

% plot the magnitude of the frequency response of filter 2
[H W]=freqz(hfilter2, 1, 'whole');
subplot(3,2,4)
plot(W/pi, abs(H));
xlabel('\omega/\pi');
title('Frequency response 2')

% find the output of filter 2
y2=conv(x, hfilter2);
subplot(3,2,6)
plot(t, y2(1:length(t)))
xlabel('Time (s)'); ylabel('Amplitude'); title('Filtered wave 2')
axis([1.5E-3 7.5E-3 -5.2 5.2])
```

The above code generates Figure 2.

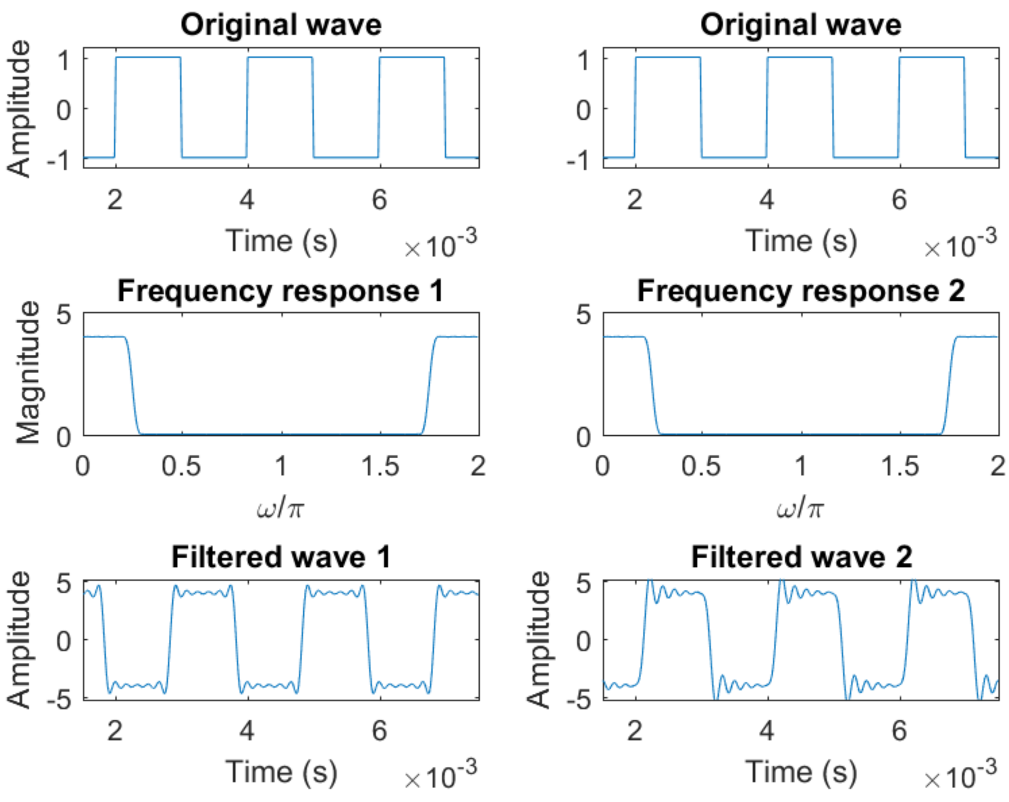

Figure 2: Figure generated by the Matlab code `FPS_wee03.m`

Question 1: what type of filter is this code using (lowpass, highpass, bandpass, bandstop) ? What is the approximate cut-off frequency of both filters ?

Question 2: what is common to and what is different between the two filters ?

Question 3: considering your answer to the previous question, try to explain why the two output signals look different. In particular, what is common to "Filtered Wave 1" and "Filtered Wave 2" ?

4.2 Sampling and reconstruction using stm32f7_loop_intr.c

Here, we repeat the experiment in Lab 1 that uses **`stm32f7_loop_intr.c`** .

```
// stm32f7_loop_intr.c

#include "stm32f7_wm8994_init.h"
#include "stm32f7_display.h"

#define SOURCE_FILE_NAME "stm32f7_loop_intr.c"
```

```
extern int16_t rx_sample_L;
extern int16_t rx_sample_R;
extern int16_t tx_sample_L;
extern int16_t tx_sample_R;

void BSP_AUDIO_SAI_Interrupt_CallBack()
{
// when we arrive at this interrupt service routine
// the most recent input sample values are (already) in global
// variables rx_sample_L and rx_sample_R
// this routine should write new output sample values in
// global variables tx_sample_L and tx_sample_R

  tx_sample_L = rx_sample_L;
  tx_sample_R = rx_sample_R;
  BSP_LED_Toggle(LED1);
  return;
}

int main(void)
{
  stm32f7_wm8994_init(AUDIO_FREQUENCY_48K,
                      IO_METHOD_INTR,
                      INPUT_DEVICE_DIGITAL_MICROPHONE_2,
                      OUTPUT_DEVICE_HEADPHONE,
                      WM8994_HP_OUT_ANALOG_GAIN_6DB,
                      WM8994_LINE_IN_GAIN_0DB,
                      WM8994_DMIC_GAIN_17DB,
                      SOURCE_FILE_NAME,
                      NOGRAPH);
  while(1) {}
}
```

Given that we will be using as analog input the signal generated by the function generator in the lab, we select the analog LINE-IN in the STM32F746G Discovery board instead of the microphones on the board. This is done by passing the parameter INPUT_DEVICE_INPUT_LINE_1 (instead of INPUT_DEVICE_DIGITAL_MICROPHONE_2) to function `stm32f7_wm8994_init()`. You can do this by editing the source file `stm32f7_loop_intr.c`, re-building the project, downloading, and running the program.

Once the program is running, adjust the function generator to output a 1 kHz square wave of about 2 Vpp. Connect the output of the function generator to the LINE IN input of the STM32F746G board using the **BLUE** mini-jack and the attached (little) interface board which includes a resistor divider (**CAUTION**: do NOT use here the gray mini-jack which does not include a protection to the STM32F746G LINE IN input). Use an appropriate cable to connect also the output of the function generator to CHAN 1 of the oscilloscope.

Use the gray mini-jack to connect the analog output of the STM32F746G board to CHAN 2 of the oscilloscope. You should see a figure similar to Figure 3.

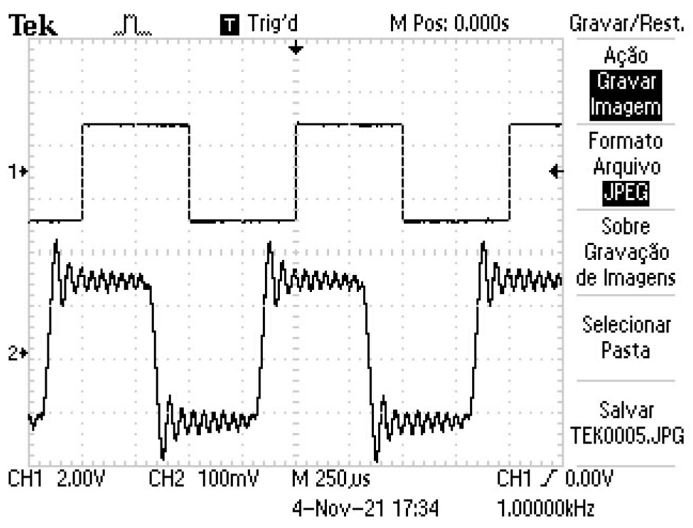

Figure 3: Oscilloscope view of the function generator square wave and analog output of the STM32F746G.

Question 4: the STM32F746G board implements the block diagram illustrated in Figure 4 and that has been discussed in one of the recent FPS lectures. What simplification in this block diagram does the above `stm32f7_loop_intr.c` code enforce ?

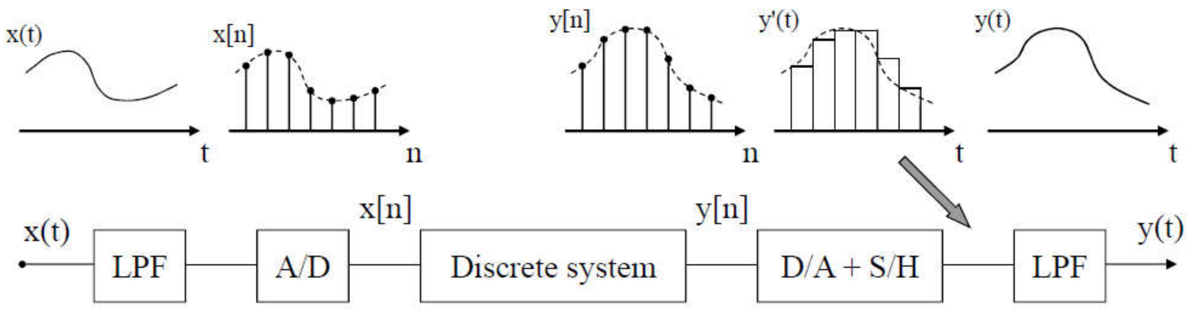

Figure 4: Complete sampling and reconstruction chain implemented on the STM32F746G board.

Question 5: considering the system setting enforced by the `stm32f7_loop_intr.c` code, and recalling the above Matlab experiment, what type of filter does the STM32F746G board operation correspond to ? By looking at the ripples of the STM32F746G output wave as in Figure 3, what can you say about the filter characteristics that generate those non-symmetric ripples ?

Question 6: now, increase the frequency of the square wave, in steps of 1 kHz, until the STM32F746G output wave looks like a perfect sinusoid, as it is illustrated in Figure 5. What is the frequency after which the output wave looks like a sinusoid and how do you explain it (i.e. how do you understand that it is that value and not some other value) ?

Hint: you should recall here the spectral structure of an ideal square wave. In particular: does it contain all the harmonics ?

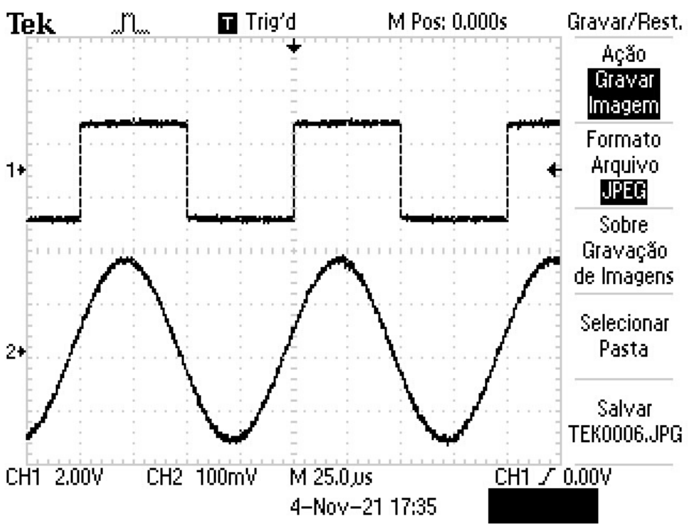

Figure 5: Oscilloscope view of the function generator square wave and analog output of the STM32F746G.

Question 7: repeat the previous experiment (question 6) with a triangular wave (make sure its duty cycle is 50%). You should obtain a figure similar to that represented in Figure 6. Now, increase the frequency of the triangular wave, in steps of 1 kHz, until the STM32F746G output wave looks like a perfect sinusoid. Is the frequency after which the output wave looks like a sinusoid the same, lower, or higher than that that you observed in Question 6 ? How do you explain that ?

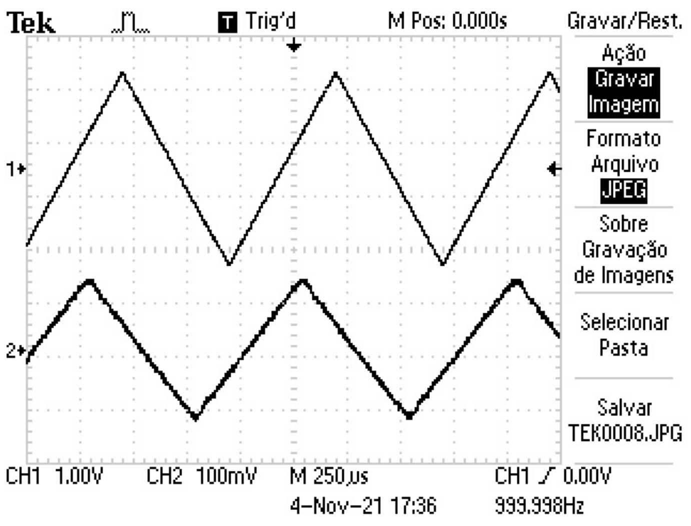

Figure 6: Oscilloscope view of the function generator triangular wave and analog output of the STM32F746G.

Question 8: repeat again the previous experiments (questions 6 or 7) this time with a sawtooth wave that is obtained by setting the duty cycle of a triangular wave close to 0%. You should obtain a figure similar to that represented in Figure 7. Now, increase the frequency of the sawtooth wave, in steps of 1 kHz, until the STM32F746G output wave looks like a perfect sinusoid. Is the frequency after which the output wave looks like a sinusoid the same, lower, or higher than that that you observed in Question 6 or in Question 7 ? How do you explain that ?

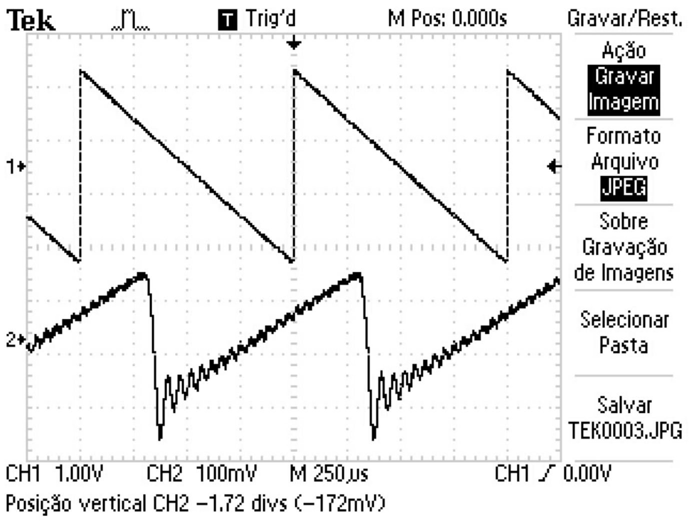

Figure 7: Oscilloscope view of the function generator sawtooth wave and analog output of the STM32F746G.

5 Conclusions

At the end of this lab class, you should have a clear understanding of the impact of a complete sampling (A/D) and (D/A) reconstruction chain as implemented on the DSP Education Kit (STM32F746G board). This understanding is instrumental to maximize the learning opportunities elicited by subsequent lab experiments.

6 Additional References

Link to Board information and resources:

https://www.st.com/en/evaluation-tools/32f746gdiscovery.html#overview