Working with Analog signals

Fabián Torres Álvarez, Karthik Ganesan, Juan Camilo Vega, Prof. Bruno Korst

Introduction

In this lab you will use the analog signal capabilities of the STM32F446ZE board which you have learned about in class. The first part of this lab uses an analog to digital converter (ADC) to sample an arbitrary input signal and transmit the results via serial port (UART). For the second part, you will use a digital to analog converter (DAC) to reproduce a sampled signal on an analog output. Finally, you will use the DAC to synthesize arbitrary signals from their FFT coefficients.

1. Oscilloscope

This lab involves working with analog signals in and out of a Nucleo board. The tool you will use to generate and visualize said signals is the Agilent MSO-X 3024A mixed signal oscilloscope available at the labs.
A hardware overview and instruction manual is available at this link from the Digital Embedded Systems Labs website (DESL). If you haven't used this oscilloscope before, please review pages 9-12 of the manual (Tutorial 1) to learn about signal generation and visualization. The rest of this handout will assume that you are familiar with basic operation of the oscilloscope.

2. Sampling with ADC

Turn on the oscilloscope and use the *Wave Gen* utility to generate a sine wave with the following specifications:

- Frequency: 120 Hz
- Amplitude: 3.30 Vpp
- Offset: 1.65 V

Warning: before connecting any signal to the board, double check that the voltage does not exceed your board's specifications. Connecting out of range values may damage your board permanently.

Connect a BNC-to-alligator adapter (shown in Figure 1) to the `Gen Out` port of the oscilloscope. Use M-M jumper cables to connect the black and red alligator clips to pins 1 and 2 from Figure 2 respectively.
Write code to sample this analog input in the main loop and print the conversion value via serial port. You will need the following functions to interact with the ADC:

1. **HAL_ADC_Start()**: triggers a conversion in the ADC hardware. This function is non-blocking, so the processor may perform any other tasks while waiting for the result.
2. **HAL_ADC_PollForConversion()**: blocks execution to wait until an ADC conversion has completed.
3. **HAL_ADC_GetValue()**: returns the ADC conversion result.

Observe the conversion results on the serial port and answer the following:

1. What are the maximum and minimum values?
2. Can you recognize any pattern in the conversion results?
3. What happens when you lower the frequency in the oscilloscope to 100 mHz? What happens if you increase it to 2 MHz?

Figure 1. BNC to alligator adapter.

3. Analog ADC/DAC echo

Connect a probe to the oscilloscope with the accessory shown on Figure 3. Use a M-M jumper cable to connect the probe tip to pin 3 from Figure 2, and the black alligator clip to the wave generator ground.
Adapt your code to produce an analog voltage in the DAC with the magnitude read from the ADC. You will need the following functions to interact with the DAC.

1. **HAL_DAC_Start(&hdac, DAC_CHANNEL_1)**: turn on the DAC hardware.
2. **HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, val)**: convert the value of `val` into an analog voltage on the board.

Observe a periodic wave form on the oscilloscope and answer the following.

1. What shape do you see form on the oscilloscope? Why?
2. What is the voltage range of the output? How does it compare to the input?
3. What happens when the input frequency is increased or decreased?
4. For what frequency range is the board capable of echoing the input with little or no distortion?

Sampling resolution The ADC and DAC of the STM32F446ZE board are configured for 12 bit resolution. To see the effect of sampling at lower resolutions, you can mask the lower bits output by the ADC. Mask the lower 2, 4, 6 and 8 bits of the ADC output and see how that affects the DAC output. At what resolution do you start to see quantization noise?

Try it yourself: Use your knowledge from Digital Signal Processing to implement a filter on the input signal. Compare the performance between FIR and IIR filters for a variety of inputs.

4. (Optional) Synthesize arbitrary waveforms

For this final optional part, you will use the DAC to generate arbitrary waveforms from their FFT series. This will be the first part of Lab 3, so if you get this working, that will give you a head start for the next lab.

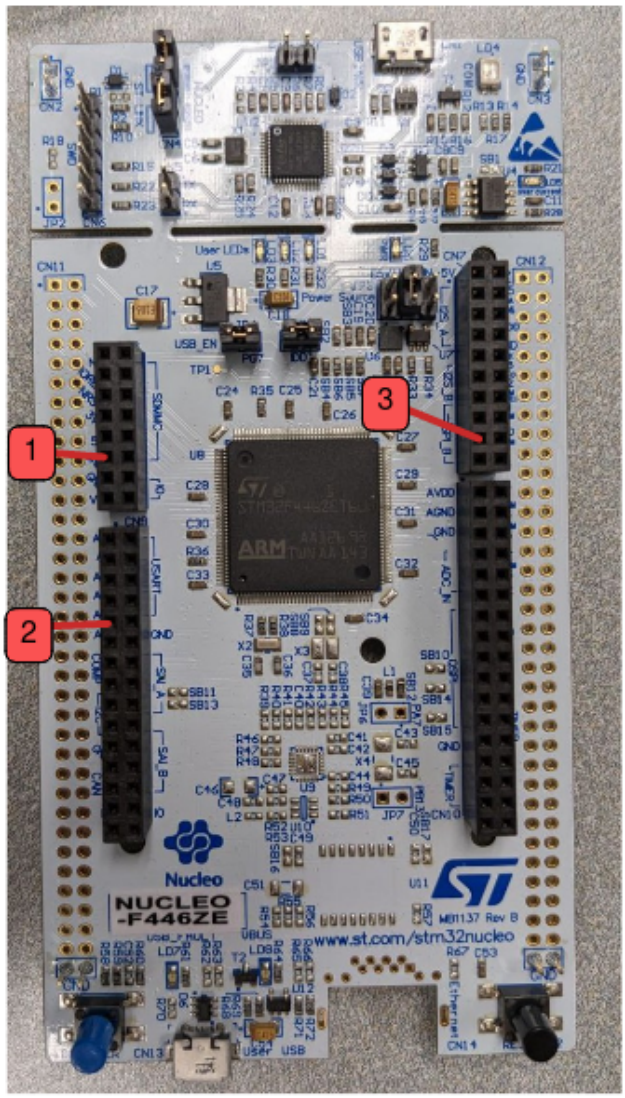

Figure 2. Board connections.

Figure 3. Probe with tip with accessory.

Use the `sin()` function from `<math.h>` to generate a sine wave of arbitrary frequency with the DAC. What is the fastest frequency you can produce that looks smooth on the oscilloscope?

Find the Fourier series coefficients for a square wave signal. Write the code to implement the Fourier series with `sin()` and

produce analog voltages with the DAC. Observe the produced signal on the oscilloscope and answer the following.

1. How many terms do you need to add for the signal to “look square”?
2. What is the trade-off between maximum frequency and number of coefficients?
3. How many Fourier terms can you compute at 1 kHz?

Try it yourself: Use the push button on the board, and the keypad from Lab 1 to control the amplitude, frequency, or shape of the generated signal. Congratulations! You’ve built your first synthesizer.