# Generation of the signal

We produce a noisy signal to feed our FIR filter in order to test its performances. The clean signal we produce is a monochromatic sinusoid so that it is easy to reconstruct how good is the filtering process. For the noise contribute, we decided to add a high frequency monochromatic noise which amplitude is comparable (slightly bigger) than the clean signal wave. The choice for a monochromatic noise has two main reasons:

1. Even if the noise has the same amplitude of the signal, the two contributes are still quite distinguishable. With a white noise of the same amplitude as the signal, the signal would have been totally hidden.
2. Having a specific noise frequency allows to set up a FIR filter which filters out the noise only. In our particular case, the noise has a higher frequency with respect to the signal, so a low-pass filter should make the job. A white noise would be much more difficult to be filtered out, because it would take all the frequencies and we could not remove it with a simple high/low-pass filter.

The final signal is produced by summing the noise and the clean signal. In the plot below, there is the graphical representation of the signal.

IMMAGINE DEL PLOT

??? to add the specifics about the signal we used? Like frequency, sampling rate, amplitude, etc.. ???

# Scipy FIR filter

Our implementation of the FIR filter has 6 taps, and we need the coefficients to set these taps in order to filter out the noise. For this purpose, we use the utility function provided by scipy. We specify as cut-off frequency ???, which lies in between the signal and the noise frequencies.

??? say that the more taps we have the more steep can be the gain function? Every tap is a coefficient in the discrete fourier transform of the signal???

Scipy provides also a function which performs the action of a FIR filter, so we rely on its output in order to check whether our implementation works properly. The processed signal is shown in the plot below.

IMMAGINE DEL SEGNALE PROCESSATO CON SEGNALE ORIGINALE E SEGNALE SPORCO. MARIKA SCEGLI I COLORI

The processed signal resembles the clean original signal, even if they don’t match perfectly. Of course, having a finite number of taps (which is very low in our particular case) implies that the filter does not perform a perfect cut-off with the noise, but the result is still acceptable.

Once we know what to expect from the FIR filter, we can feed our signal into the programmed FPGA. Though, this cannot be done directly, because our signal is encoded in double precision (each value uses 64-bits), but the connection with the FPGA only supports 8-bit variables. Thus we need a further pre-processing of the signal.

# Data conversion

The FPGA accepts and outputs 8-bits data. When working in a Python environment, the default is to use 64-bits variables. We need two routines to convert data from one format to the other.

## Signed variables

First of all, we must consider that our signal assumes both positive and negative values, so we must keep track of the sign. For this reason, the first of the 8 bits has to map the sign, and we are then left with only 7 bits to store the information on the magnitude.

Before considering the magnitude, we must pay attention to the way signed variables are treated in VHDL, because the standard requires that the first bit is 0 for positive numbers and 1 for negatives. This is a tricky behaviour, since in this representation the lowest negative number is greater than the highest positive one. We thus need to shift of all the negative numbers above the positive ones.

The magnitude of the variables must be rescaled keeping into account that there are only 7 bits available to represent the values, which means that the representable values are the integers in the range , so every 64-bits value must be mapped into this range. The minor cardinality of the 8-bits set of values implies that we must perform an approximation on the original 64-bits values, which must be minimized in order to lose as small information as possible.

??? are we sure to perform the computation like that? ???