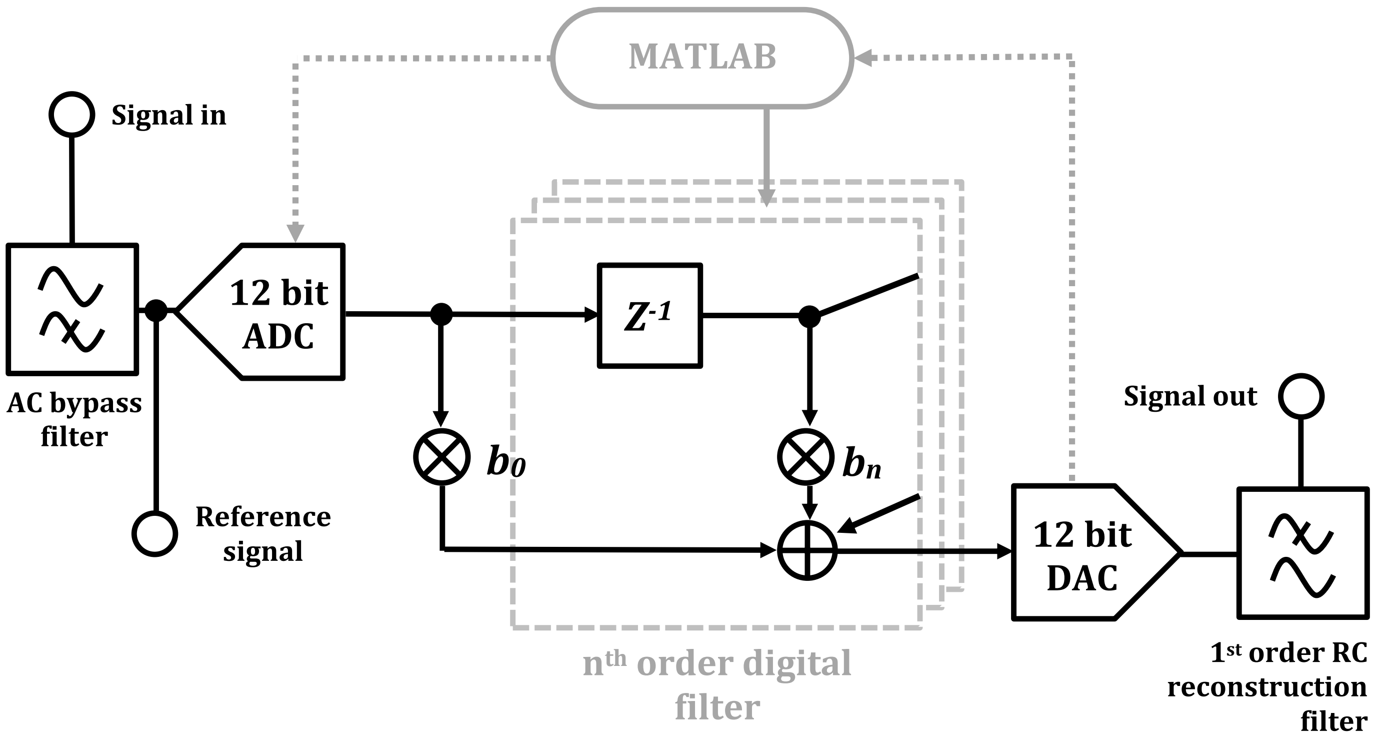
Digital Filter Workbench manual

Contents

0. Introduction

System, which is thoroughly explained in this document, is designed for learning and testing basic DSP filtering algorithms. Data processing functionality is implemented with **ARM Cortex M3** microcontroller. Front-end user interface is a **MATLAB** based application.

Filter data is uploaded to the device and stored in the temporary memory until reset. Then, device can process signals by sampling them in real time. Also it can receive and process digital sequences directly from MATLAB.



Prior to sampling, input signal is preconditioned with zero offset 10 Hz high-pass filter and biased for an ADC input. Sampled signal is processed with algorithm one sample at a time; delay line is updated each step. Reconstruction is done with customizable low-pass filter.

When operating on digital sequences, MATLAB will feed fixed-point data directly in the DSP algorithm by blocks.

0.1 Device overview

Mini-USB socket

Serial communication to the PC workstation is performed through this port.

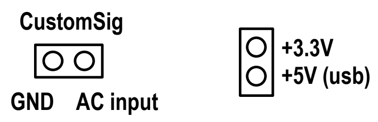
Reset button

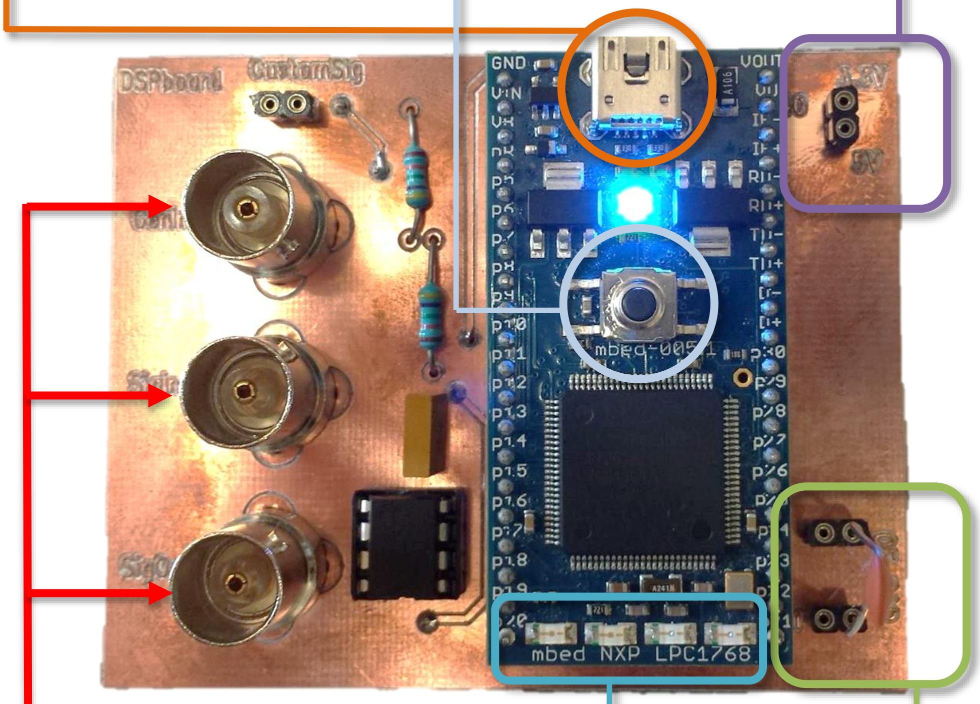
Use this button to reset the device. This button is also used for *programming the microcontroller*.

Blue LED indicates that device is powered OK.

Prototyping sockets

These are used for connecting the device to the prototyping board with ordinary single-core wires.





BNC connectors

- Function generator input
- AC bypass signal (reference)
- Reconstructed output

Filter indicators

LEDs will indicate when device is not in standby and is filtering real-time ADC data:

🟦 ⬜ ⬜ ⬜

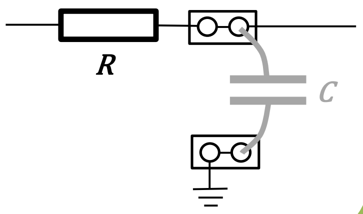
FIR filter is active

⬜ 🟦 ⬜ ⬜

IIR filter is active

Reconstruction filter

Capacitors for the RC reconstruction filter are inserted between these two sockets:

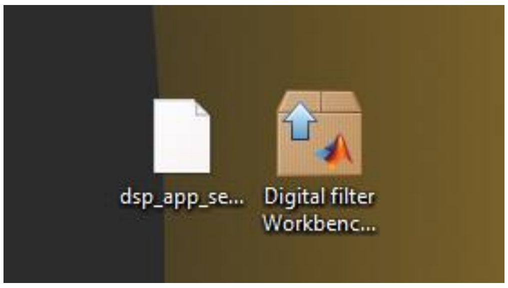


1. Setup and installation

1.1 Download program files

You will need two files by following name (example):

- **Digital filter Workbench v1.1**.mlappinstall
- **dsp_app_server-1.1**.bin



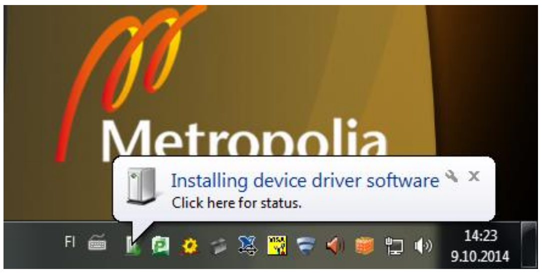
File location in intranet is specified by teacher. Select the latest file version to make sure that you are up-to-date.

1.2 Set up the device

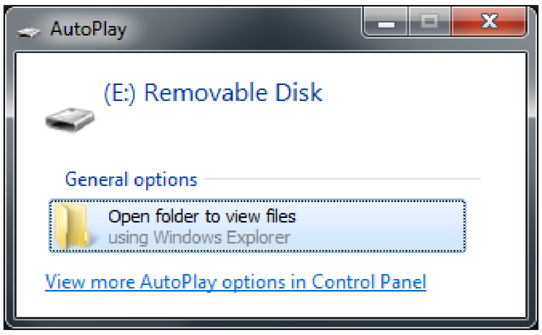
✓ Close MATLAB, if it is running

This is important step, because *MATLAB can detect the device only on startup*. When MATLAB is running on the workstation, it unfortunately cannot react to any hardware changes nor detect the device.

✓ Connect device to the workstation



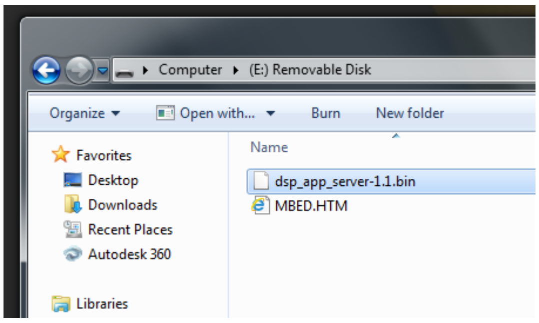
Connect the device to the workstation with Mini-USB cable. If you connect it for the first time, driver update window should appear. Even if **it takes long**, it will find the proper drivers all right.


AutoPlay
(E:) Removable Disk
General options
Open folder to view files
using Windows Explorer
View more AutoPlay options in Control Panel

If the device drivers are installed successfully, AutoPlay window should appear. Now you can put files to the internal flash memory buffer.

☹ If after cable connection nothing happens, no drivers are installing, and there is no removable drive, check everything (cable, device, USB port) and try connecting other device or go to another workstation.

✓ Program the MBED with application .bin file



Now you can copy the *dsp_app_server-1.1.bin* file into the MBED drive. Then push the Reset button on MBED (near the blue LED). After this, it should blink couple of times and light up again.

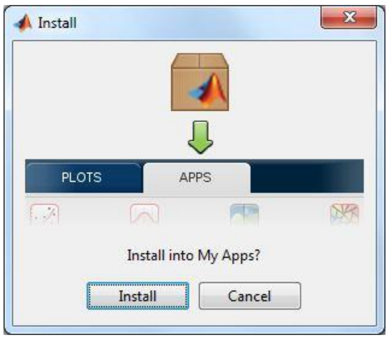
☺ Don't worry if the removable disk has many .bin files, programmer will pick the *latest uploaded file*.

☺ You can skip this step only if you are 100% sure that MBED has our DSP firmware installed on it. If you are not sure, erase and put the right program.

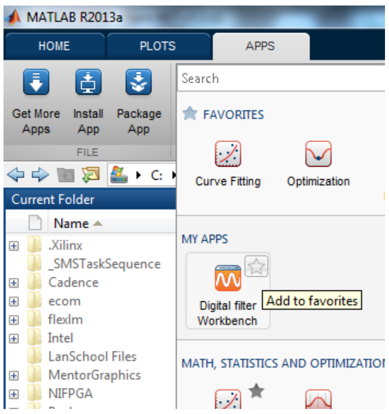
1.3 Set up MATLAB

✓ Install the application from .mlappinstall file

Double click on the *Digital filter Workbench v1.1.mlappinstall* file so that installation window should appear. If MATLAB is not open, then it will start automatically to display this dialog. Click Install and you are done.



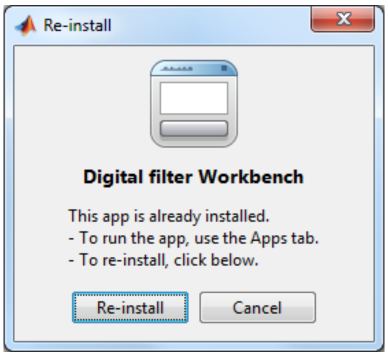
After this step, application can be found under Apps tab. You can also add it to favorites by pressing the star in the corner. Then application will appear at the top bar.



❖ Re-install

If some version of the same application is already installed, MATLAB will ask whether you want to reinstall it. Do it **only if you are updating** the application. If you are installing it first time, but:

- 😐 MATLAB says that App *is already installed*
- 😐 Digital filter Workbench App is *not anywhere under Apps tab*


Re-install

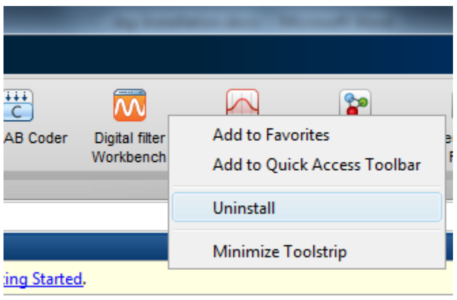
Digital filter Workbench

This app is already installed.
- To run the app, use the Apps tab.
- To re-install, click below.

Then, press Cancel, and go to
My Documents
└ **MATLAB**
└ **Apps** and *delete every folder* with name
└ **"DigitalFilterWorkbench"** or similar

This will clear all garbage which MATLAB could not wipe itself after previous app has been reinstalled. If you did everything properly, next time MATLAB will ask you to *install application anew*.

☹ If after new installation or update application does not work properly and prints red text into MATLAB console when you try to interact with an application (press buttons, input text or click on interface), try to uninstall the application by right-clicking on its icon and clicking on Uninstall.
After application icon is gone, go through above steps and delete the remaining garbage.



2. Application interface

2.1 Signal processor

Toolbar

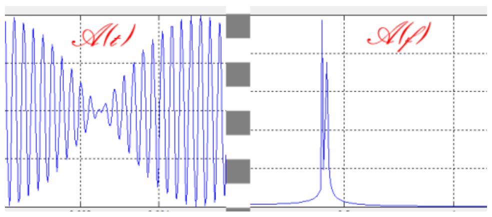
Use these buttons to zoom graph data in and out

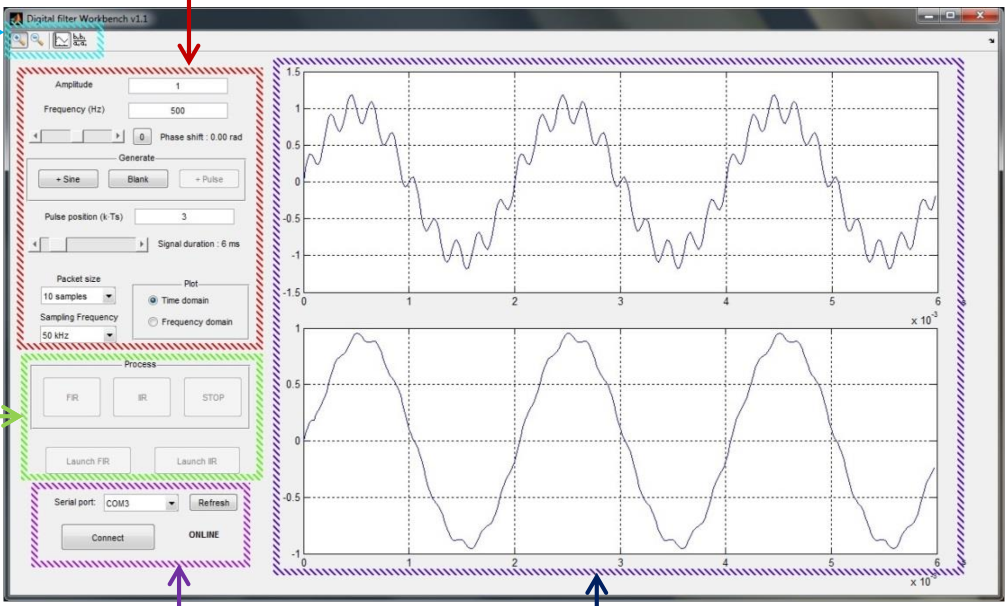
This button shows signal processor (default view), and

This button shows coefficient editor

Signal generator panel

Use this panel to set up and combine sine waves or impulses. Here you can select how signal is sampled, and how long should it be. You can also select how data is presented, i.e. in time or frequency domain:





Digital filter Workbench v1.1

Amplitude: 1

Frequency (Hz): 500

0 Phase shift : 0.00 rad

Generate: + Sine | Blank | + Pulse

Pulse position (k·Ts): 3

Signal duration : 6 ms

Packet size: 10 samples

Sampling Frequency: 50 kHz

Plot: [✓ Time domain] Frequency domain

Process: FIR | IIR | STOP

Launch FIR | Launch IIR

Serial port: COM3 | Refresh

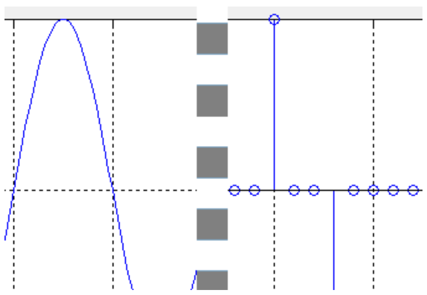
Connect | ONLINE

Connection panel

Use this controls to establish connection with the device via serial COM port.

Data display

Upper graph shows data generated by the user and lower graph shows processed data. Pulse data is displayed as discrete and sine data - as continuous:



Processing panel

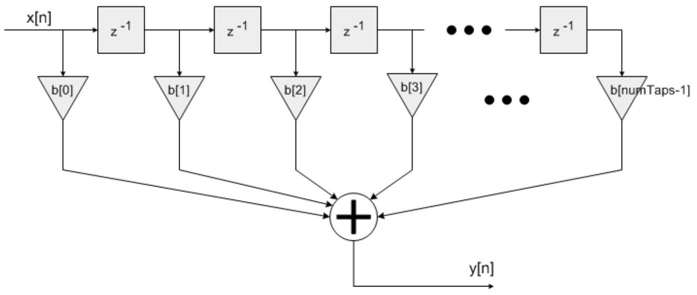
This panel contains commands to interact with the digital filter. Discrete-time data analysis and real time operation are executed from here.

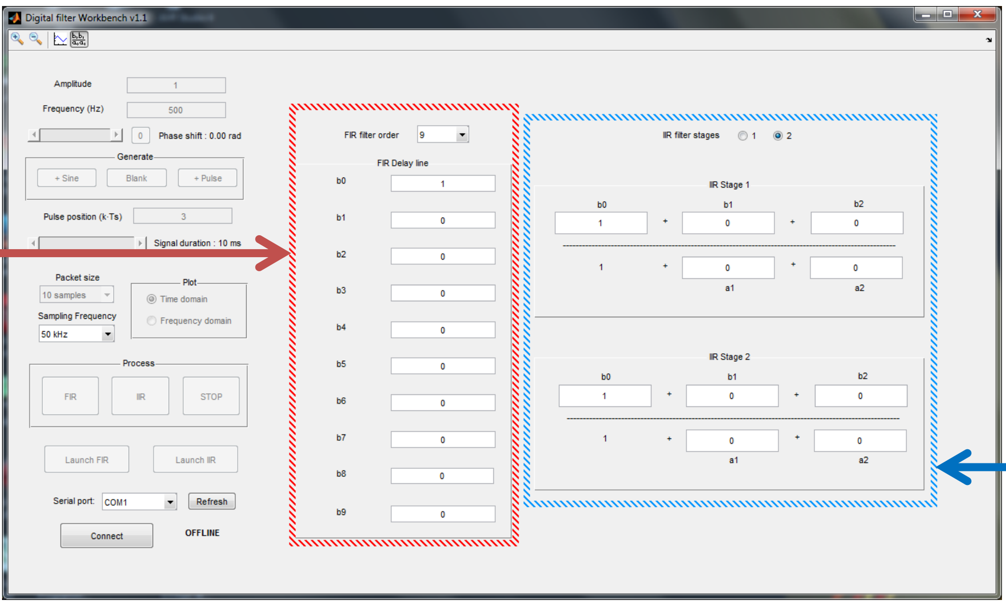
2.2 Coefficient editor

Finite impulse response filter algorithm is based upon a sequence of multiply-accumulate (MAC) operations. Each filter coefficient b[n] is multiplied by a state variable which equals a previous input sample x[n].

Implementation of filter up to 9th order is possible (10 coefficients). The transfer function of the filter is defined as follows:

$$H(z) = b_0 \cdot z^0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + \dots + b_n \cdot z^{-n}$$





Digital filter Workbench v1.1

Amplitude	1
Frequency (Hz)	500
Phase shift : 0.00 rad	0
Pulse position (k·Ts)	3
Signal duration : 10 ms	
Packet size	10 samples
Sampling Frequency	50 kHz
Plot	[✓ Time domain] Frequency domain
Serial port:	COM1

OFFLINE

FIR filter order: 9

FIR Delay line

b0	1
b1	0
b2	0
b3	0
b4	0
b5	0
b6	0
b7	0
b8	0
b9	0

IIR filter stages: 1 [✓ 2]

IIR Stage 1

b0		b1		b2
1	+	0	+	0
1	+	0	+	0
		a1		a2

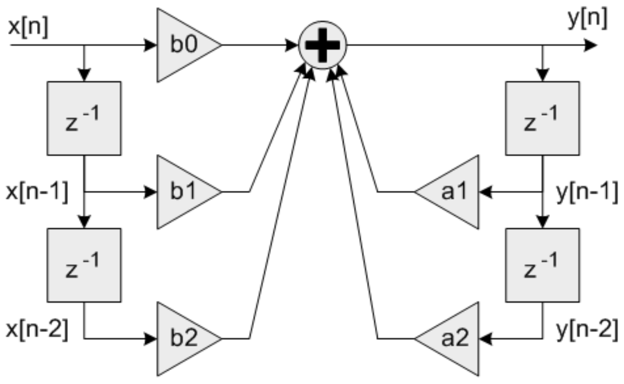
IIR Stage 2

b0		b1		b2
1	+	0	+	0
1	+	0	+	0
		a1		a2

Infinite impulse response filter is implemented in the form of *Biquadratic cascade* filter. It consists of cascaded 2nd order stages with 3 feed-forward and 2 feedback coefficients each.

One or two stages can be defined, so that output of first stage becomes an input of the second stage. The transfer function of each stage is defined as follows:

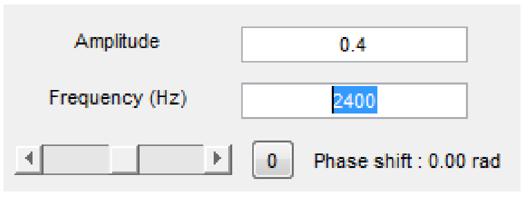
$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$



3. Using the application

3.1 Generate the signal

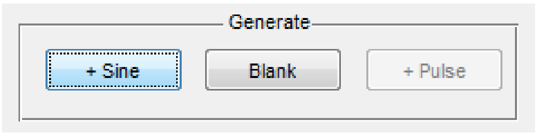
✓ Type in the signal parameters



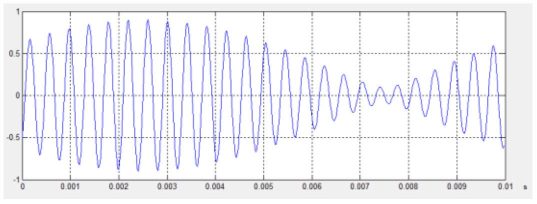
Start by putting some values in **amplitude** and **frequency** input boxes, for example, *0.4* at *2400 Hz*. Use the **phase shift** slider to delay or advance the sine wave. Note the small zero button next to that slider – it resets phase shift back to zero radians.

😐 Consider using a **decimal point** instead of comma. MATLAB will skip interpret comma, and will use value 105 instead of 1,05 (example)

✓ Add the signal to the graph



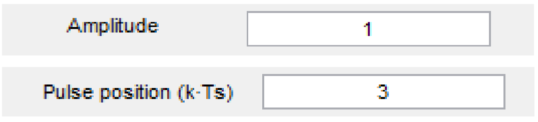
Now press the **+Sine** button, and your signal will appear at the upper graph. This is how you generate the signal. After you put something on the graph, **Blank** button will erase the signal.



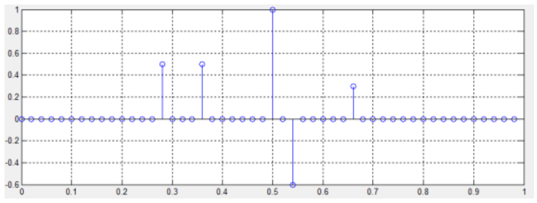
Now, *while the first signal is on the graph*, type new values into boxes. For instance, set amplitude to *0.5* and frequency to *2500 Hz*. Then, change the phase shift to about *-1.5 rad* and press +Sine button again. Now two signals are added together. With example values you should get something like on the picture:

🔍 You can also use **+/- zoom** buttons from the toolbar (left upper corner)

✓ Combine pulses



You can do the same addition between impulses. Use the **amplitude** and **pulse position** fields to specify the desired parameters. Pulse position is given in integer number of sampling periods.



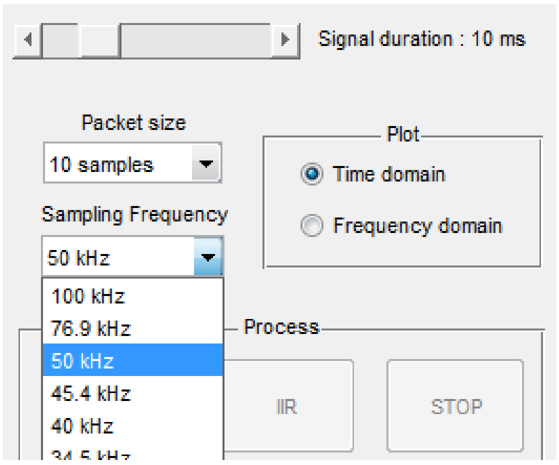
You *cannot combine pulses and sinusoids*, so all wave signals should be erased in order to add pulses and vice versa.

Impulse data will be displayed on the graph as discrete points.

☹ If any of the above steps did not work, and MATLAB prints red text into command prompt, try to *restart MATLAB*. If it doesn't help, consider *reinstalling the app*, as described in chapter 1.3

3.2 Manipulate the signal

✓ Adjust the sampling parameters



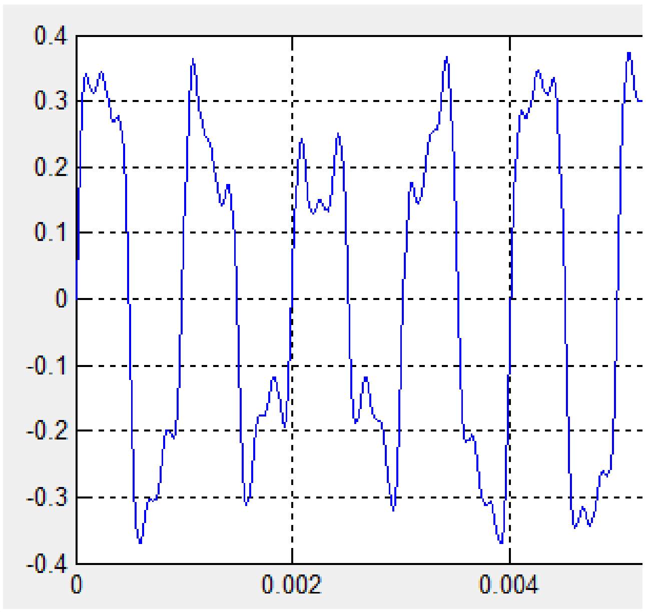
Signal duration slider is controlling the length of displayed signals. When you move the slider, **packet size** value will change automatically. Packet size is explained in chapter 3.5.

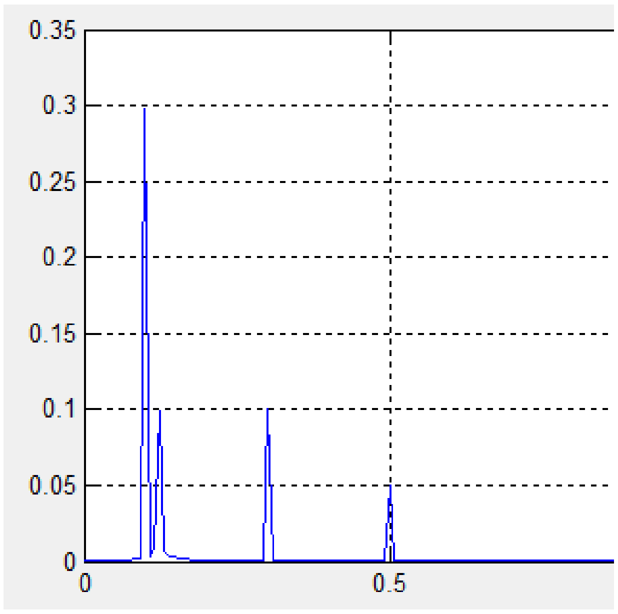
You can choose **sampling frequency** from the list, such that signal will be generated with corresponding amount of points per second:

- 100 000 Hz
- 76 923 Hz
- 50 000 Hz
- 45 454 Hz
- 40 000 Hz
- 34 482 Hz
- 30 303 Hz
- 25 000 Hz
- 20 000 Hz
- 14 925 Hz
- 10 000 Hz

✓ Analyze the spectrum of your signal

By using **plot** panel, you can choose between **time domain** and **frequency domain** of the signal:





By dragging the **signal duration** slider, total amount of points in the signal is changed. This affects the DFT algorithm used to generate the spectrum, and because of that you can see how *spectrum precision is changed with signal duration*.

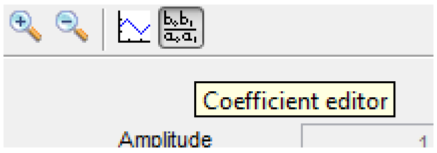
You can also zoom the spectrum with **+/- zoom** buttons from the toolbar (left upper corner)

☹ If any of the above steps did not work, and MATLAB prints red text into command prompt, try to *restart MATLAB*. If it doesn't help, consider *reinstalling the app*, as described in chapter 1.3

3.3 Edit filter coefficients

✓ Go to the Coefficient editor view

Use the rightmost button at the toolbar panel to switch into coefficient editor mode.

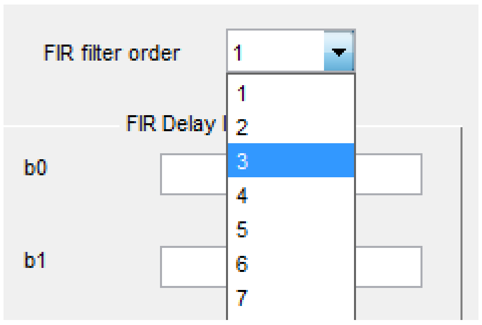


☞ To set up FIR filter

Suppose we want to implement finite response filter of 3th order with such transfer function:

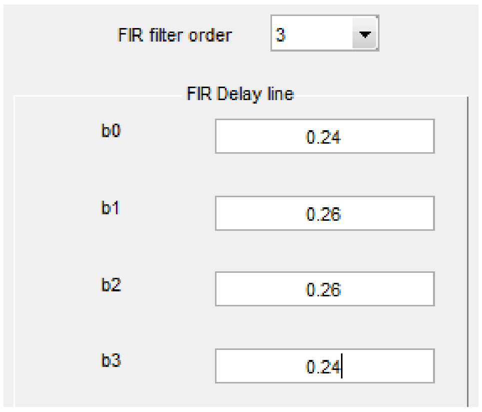
$$H(z) = 0.24 \cdot z^{0} + 0.26 \cdot z^{-1} + 0.26 \cdot z^{-2} + 0.24 \cdot z^{-3}$$

Start with selecting **order of the filter** from the pop-up menu. When you select the desired order, appropriate number of input boxes appears.



By looking on the FIR filter model, described in chapter 2.2, fill in the appropriate coefficient values. That is, $\mathbf{b_0}$ stands for z^0 coefficient, $\mathbf{b_1}$ for z^{-1} and so on.

☺ Consider using a **decimal point** instead of comma. MATLAB will skip the comma, and will use value 105 instead of 1,05 (example)



FIR filter order 3

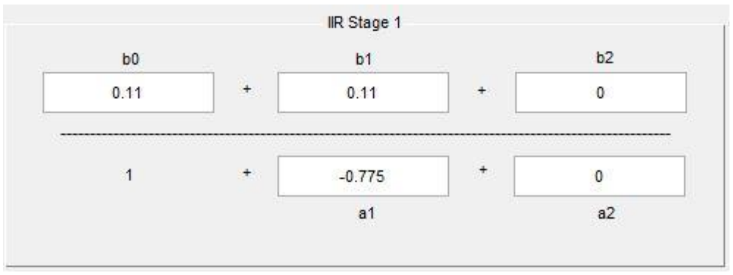
FIR Delay line

b0	0.24
b1	0.26
b2	0.26
b3	0.24

☞ To set up FIR filter

IIR coefficient panel works in way similar to FIR panel. For example, if given following infinite response filter transfer function of first order

$$H(z) = \frac{0.11 + 0.11z^{-1}}{1 - 0.775z^{-1}}$$



IIR Stage 1

b0		b1		b2
0.11	+	0.11	+	0
1	+	-0.775	+	0
		a1		a2

we can implement on the second order biquadratic stage, just by *putting z^{-2} coefficients to zero*. According to the model described in chapter 2.2, the transfer function will look something like this:

$$H(z) = \frac{0.11 + 0.11z^{-1} + 0 \cdot z^{-2}}{1 + (-0.775)z^{-1} + 0 \cdot z^{-2}}$$

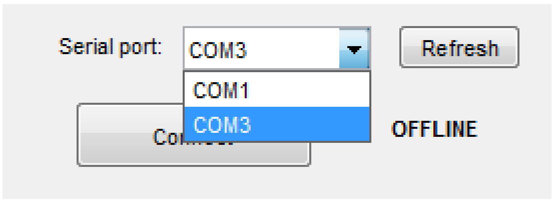
☹ If any of the above steps did not work, and MATLAB prints red text into command prompt, try to *restart MATLAB*. If it doesn't help, consider *reinstalling the app*, as described in chapter 1.3

3.4 Connect to the device

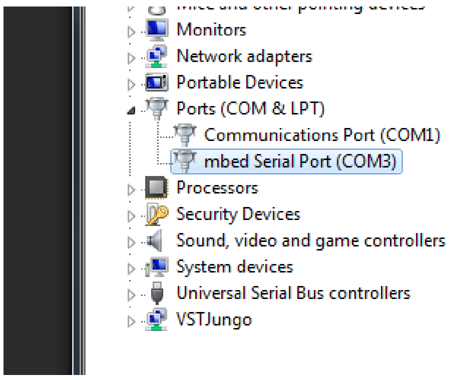
😐 Due to our workstations' specific operating environment, *MATLAB can detect the device only once*. When MATLAB is running on the workstation, it unfortunately cannot react to any hardware changes nor detect/discard the device.
It is strongly recommended that you plug the device **always before** you start working in MATLAB, otherwise you will have to restart the program.

✓ Locate the device from the list of available ports

All the serial COM ports available to MATLAB will be shown in the **Serial Port** drop-down list. Unfortunately, ports are distinguished only by number, so you have to locate the device yourself. There are several options:



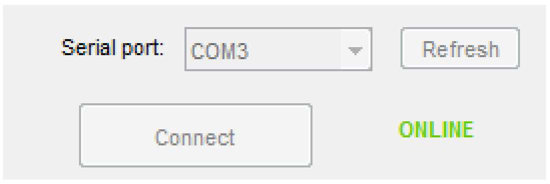
- Trial and error by attempting to connect to each available port (generally faster)
- Going to Start -> Devices and Printers menu and search for the *mbed Serial Port (COM)*
- Looking up the device port from the Windows' Device Manager. Type *device manager* in the Start menu, and click on the icon. You will have to enter your user password to advance. In the device manager window, find *Ports (COM&LPT)*
  └ *mbed Serial Port* and the necessary port number next to it.



☹ If no *mbed Serial Port* can be seen under the Ports section, even that the device is properly connected, please check everything (cable, device, USB port) and refer to chapter 1.2 for details.

✓ Establish the connection

Do it by simply pressing the **Connect** button. If the connection is successful, interface will be grayed out and you will see text **ONLINE** in bold green letters.



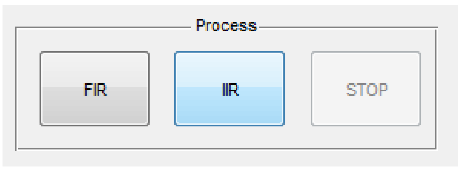
☹ If program does not connect, and you are really trying the correct port, *refresh and try to connect again couple of times*. If this does not fix the link, re-plug the device and try again with the same port.

☹ If any of the above steps did not work, and MATLAB prints red text into command prompt, try to *restart MATLAB*. If it doesn't help, consider *reinstalling the app*, as described in chapter 1.3

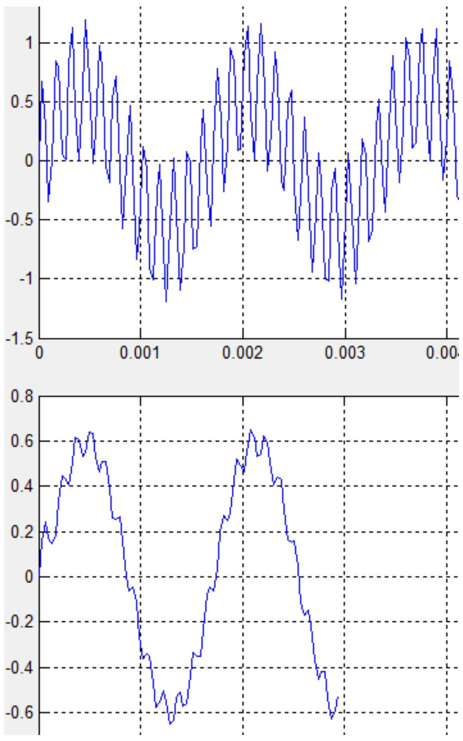
3.5 Process the signal

✓ Use the process panel

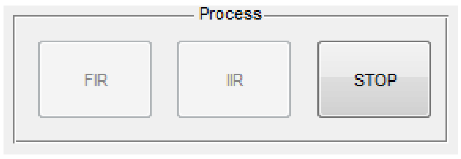
Once the device is connected, **FIR** and **IIR** buttons on the process panel (Signal processor window) will become available. Pressing the button will execute the corresponding filter.



Generated data will be split to *packets of n samples* (specified in **Packet size** menu) and sent to the device one by one. Processed data is gradually displayed on the lower graph. Depending on the amount of points set by **Signal duration** slider, whole process may take a while.



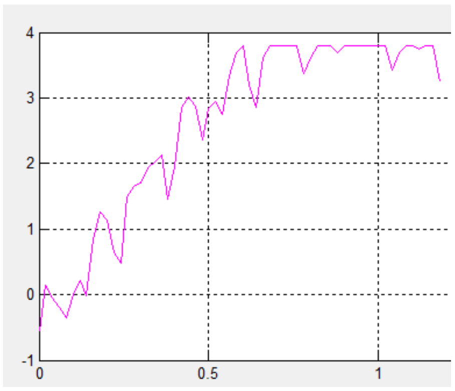
You can always interrupt the process by pressing the **Stop** button.



❖ **Saturation behavior**

☹ Because of fixed-point DSP arithmetic, result of processing can be saturated. Data that has been saturated is displayed in **magenta**. *Saturation distorts the original signal and its spectrum.*

Saturation happens when your filtering algorithm has positive **gain** at certain frequencies, and the input value is big enough to cause output accumulator to overflow. For processed signal shown on the picture, total amplitude of input sequence was *larger than one*. In this case, signal has been scaled down to the range $[+1, -1]$ suitable for fixed-point, processed and rescaled back to the original.



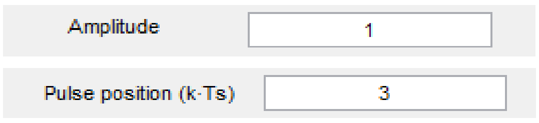
✓ To avoid approaching the bounds in positive gain conditions, try using **amplitudes a decade lower**, for example 0.1 or 0.01.

❖ Filter impulse response analysis

In this example, IIR filter from chapter 3.3 is analyzed. Sampling frequency is set to *50 kHz*.

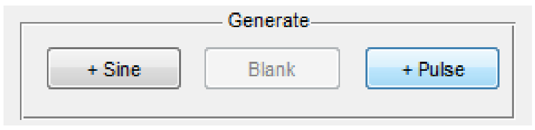
✓ Put one impulse on the blank graph

Go to the Signal processor window (). Start by specifying the **amplitude** and **pulse position** in the corresponding data fields. It's always better to place impulse closer to zero time.



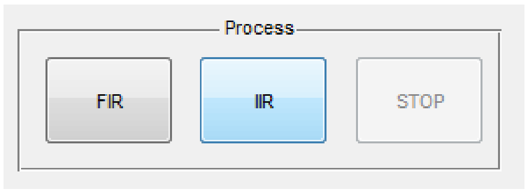
Amplitude	1
Pulse position (k·Ts)	3

Put *one pulse* to the graph.

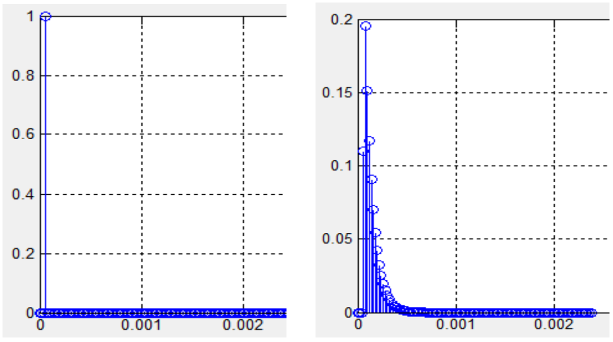


✓ Process the single impulse signal

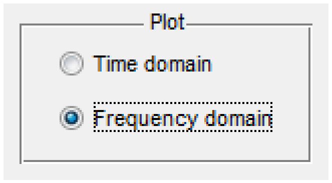
Connect to device, if it is offline, and then press the corresponding filter button on the process panel.



Your input and output should look something like that on the picture:



Now you can select **Frequency domain** presentation from the Plot panel. You should get frequency response of the filter in red, like shown on the figure below.

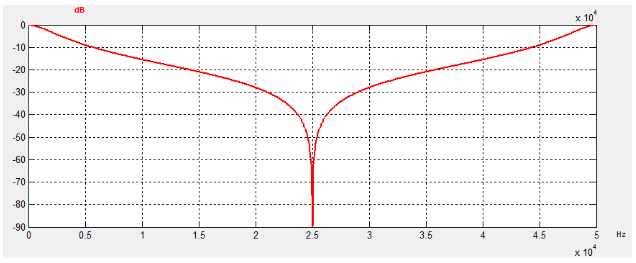


Plot

Time domain [✓ Frequency domain]

Use **+/- zoom** buttons to examine the spectrum closer

☺ Decibels are calculated once and only when input is a **single pulse** (not even two pulses stack on one point). When the original generated pulse is changed, frequency domain will be shown as is, not in decibel presentation.

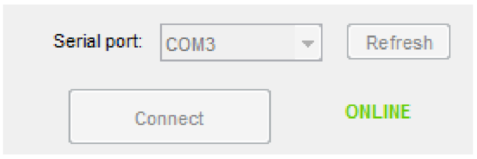


☹ If any of the above steps did not work, and MATLAB prints red text into command prompt, try to *restart MATLAB*. If it doesn't help, consider *reinstalling the app*, as described in chapter 1.3

3.6 Setting up the real-time filter

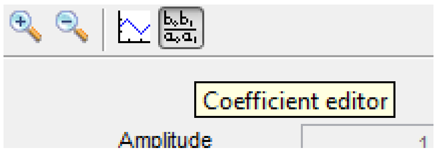
✓ Connect device to the application

First of all, go through standard procedure described in the chapter 3.4.



✓ Go to the coefficient editor view

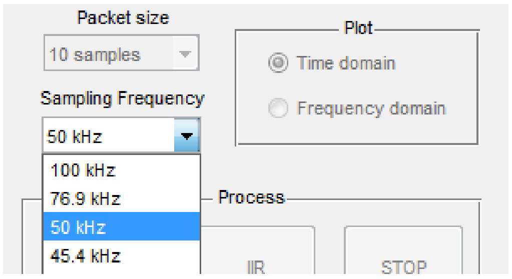
Use the rightmost button at the toolbar panel to switch into coefficient editor mode.



✓ Set the desired filter settings

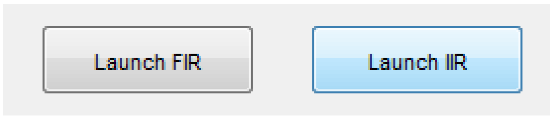
Define the transfer function of the filter by putting its coefficients into appropriate boxes as described in chapter 3.3.

Then, select the **sampling frequency** of the filter from the drop-down list. Algorithm will be synchronized to match this frequency.



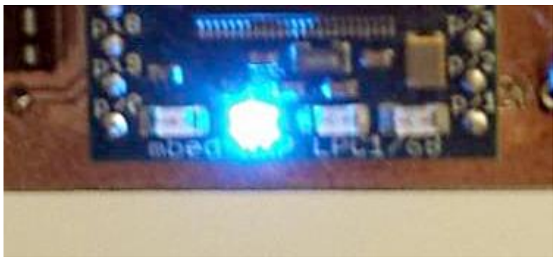
✓ Upload the coefficients to the device

Once the device is connected, **Launch FIR / IIR** buttons will be active in the Coefficient editor mode. Press the button appropriate for your filter.



☺ After this, the device will disconnect and enter *real-time filtering mode*. In order to edit coefficients or process numerical sequences, you will have to **reset** the device and connect again.

Corresponding LED will be lit on the microcontroller board, indicating real-time filtering mode. In this example, IIR type filter is used, as you can see from the second LED.

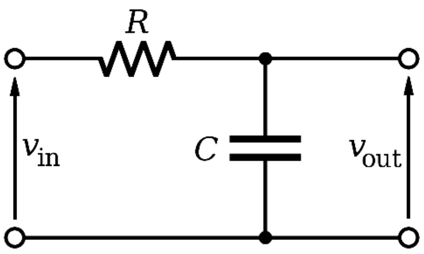


☺ If the device is reset or plugged out of the computer, filter data, which has been stored in temporary memory, is lost (LED won't be lit). In such case, you will have to connect device and launch filter again.

❖ RC reconstruction filter

A customizable first order low-pass RC filter is added at the output of the DAC to limit the frequency content of converted signal. RC filter cut-off frequency is defined as

$$f_c = {}^{1}/_{2\pi RC}$$



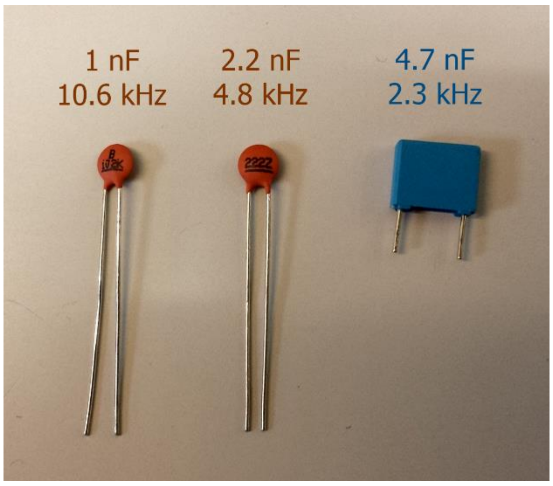
Capacitor C of desired capacitance can be placed into slot, while resistor R has fixed value of **15 kΩ**.

We can rearrange the formula, so that for desired cut-off frequency we must place capacitance of value

$$C = {}^{1}\!/_{15 \cdot 10^3 \cdot \pi \cdot f_c}\ [\mathrm{F}]$$

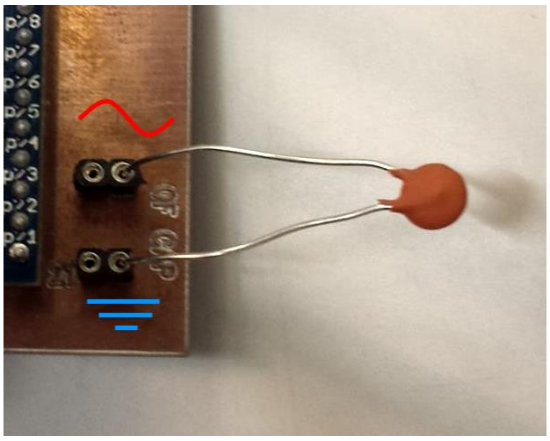
For example, some easily available lab capacitors will give the following cut-off frequencies:

- 1 nF: f_c = 10.6 kHz (-20dB @ 100kHz)
- 2.2 nF: f_c = 4.8 kHz (-20dB @ 50kHz)
- 4.7 nF: f_c = 2.3 kHz (-20dB @ 25kHz)



Use the reconstruction filter sockets to insert the capacitor between ground and signal like shown on the picture. It is also possible to combine two capacitors in parallel in order to increase total capacitance.

☹ If device is left without capacitor in the RC filter slot, signal will be fed right from the DAC into the oscilloscope, resulting mostly in “staircase” readings.



✓ Connect device to the oscilloscope

Use BNC cables to connect the device to oscilloscope:

GenIn – signal generator

SigIn – some reference channel (not necessary)

SigOut – some reading channel