Sniffy

Manual

Note: This manual describes only features and use cases linked with Sniffy. It is not a basic manual on how to use laboratory instrumentation. This manual relies on experienced users. For example oscilloscope trigger modes or MCU peripherals are not described. Sniffy runs on a variety of single chip MCUs therefore some functionalities are compromised with limited performance.

Content

1. GUI and control

The application starts to scan for available devices and automatically connects if there is only one available. Manual scan can be performed by scan button.

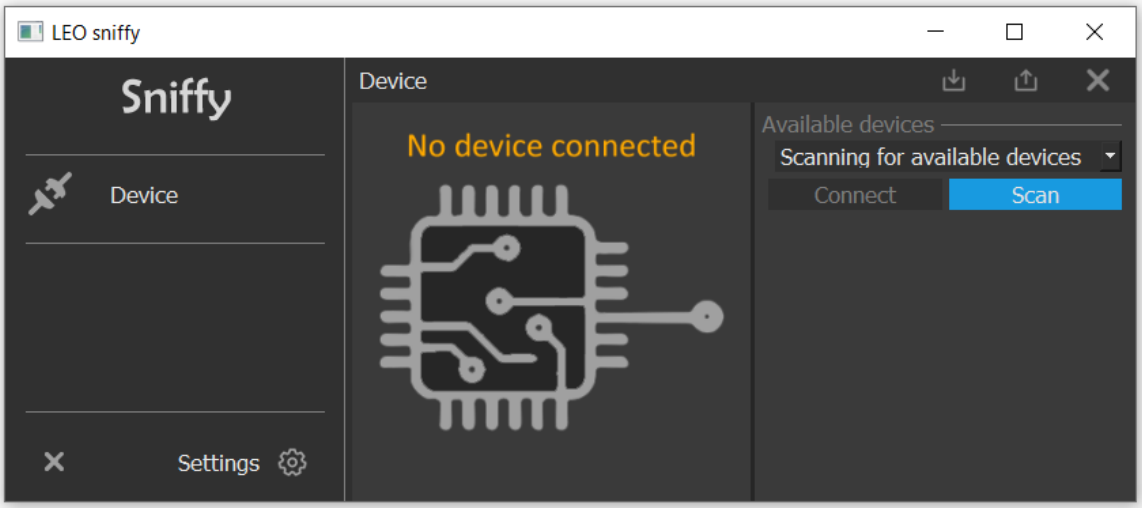
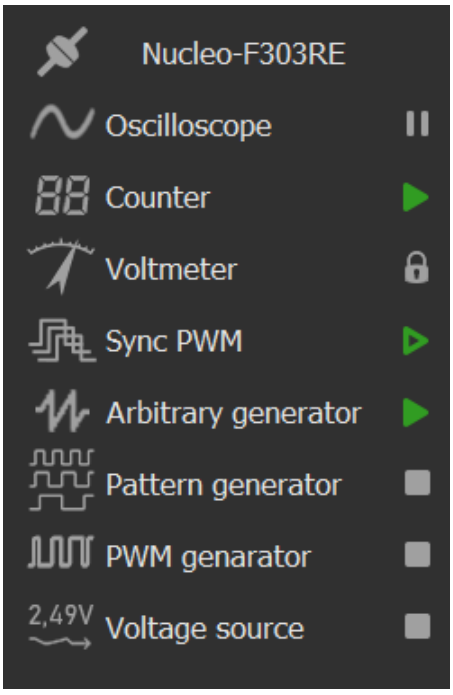


Figure 1. Sniffy GUI after startup

1.1. Modules list

Each MCU Sniffy firmware offers several functionalities similar to standard laboratory instruments which we call modules. The list of available modules is shown after successful connection. The list may vary for different boards and MCUs.



Module can be opened by clicking on it in a list. The modules list also shows the state of the module.

STOP - Module is stopped and not active. Can be clicked to open and run.

PLAY - Module is active and running. By clicking again on a module the status will go to empty play.

EMPTY PLAY - Module is active and running but the window is hidden.

PAUSE - Module is active but doing nothing. e.g. Generator is opened but not generating or Oscilloscope is stopped

LOCK - Module cannot be opened because resources needed are used by another module. Mainly because of timers, ADCs or shared memory.

1.2. Module window

Every module is opened in the new dockable window. So the GUI can be adjusted within the home window or it can be undocked as a standalone window.

The docking can be also controlled by the top-right section in each window known from MS Windows.

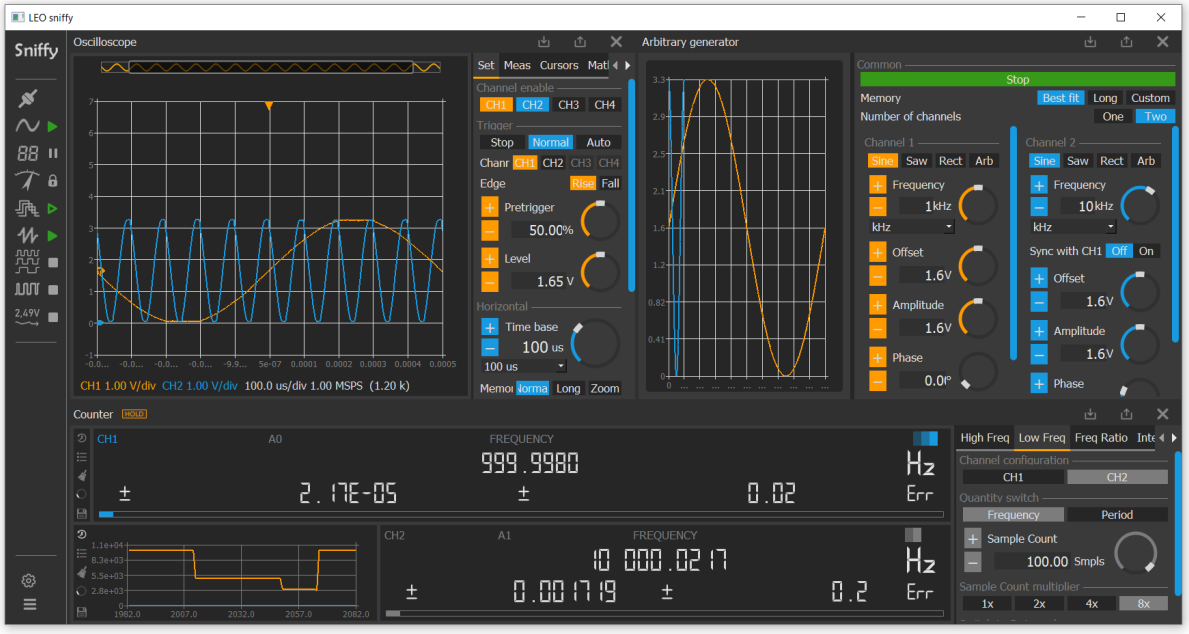


Figure 2. Example of windows docked inside home window

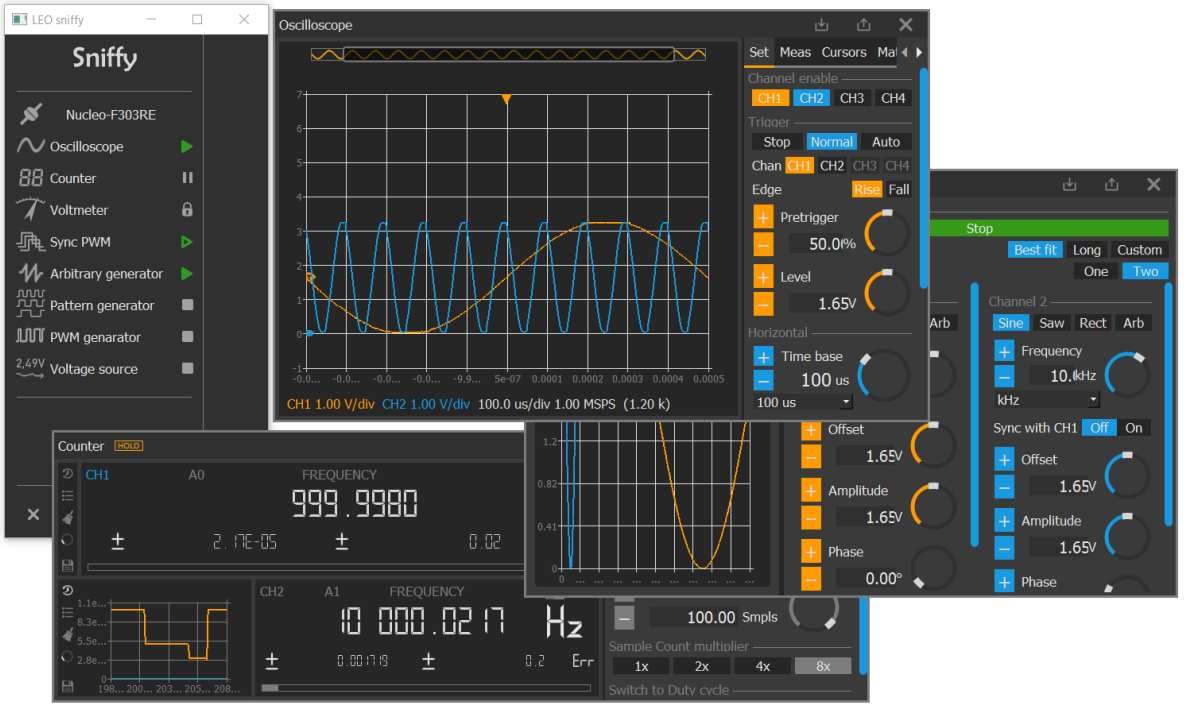


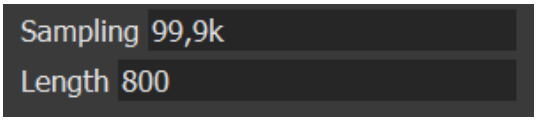
Figure 3. Same setup with windows undocked

1.3. GUI Control and input

The GUI consists of several different controls. They are unified and used across all modules. Buttons, tabs and switches are pretty straightforward to use. Therefore only specific controls are described.

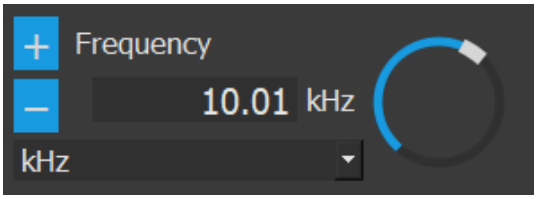
1.3.1. Text or Value input

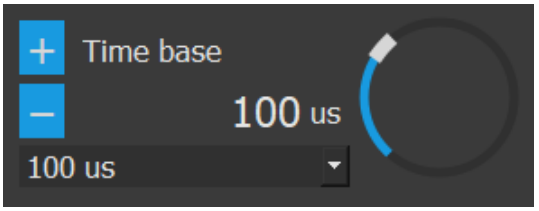
In some cases users can insert text or number values. The decimal numbers accept both separators either local (comma “,” in czech) or decimal point “.”. Numbers can be also inserted with a multiplication unit. For example 9.9k, 9k9 and 9900 are equivalent and valid inputs.



1.3.2. Dial input

Dial input is the most used control in Sniffy. Dials are usually very uncomfortable so behavior of our dials was optimized for maximum effectiveness. There are two types of dials. First type has specified options and the second allows you to set any number. Their usage is similar.





BUTTONS - Increases or decreases value

SELECTION - Selects the base unit for shown value or value from predefined options. This control might be hidden if not needed.

VALUE - In case the dial can set any number the value can be edited. Only 2 decimal digits are shown but all written digits are used. See the *Text or Value input* section for information about valid input.

DIAL - The dial can be controlled by two ways: mouse wheel or mouse drag. Mouse wheel is a way to quickly set morless precise values. Another option is to use click and drag which is a fast way for big value changes. In this case the control value increases or decreases based on distance from the diagonal axis. See the picture below.

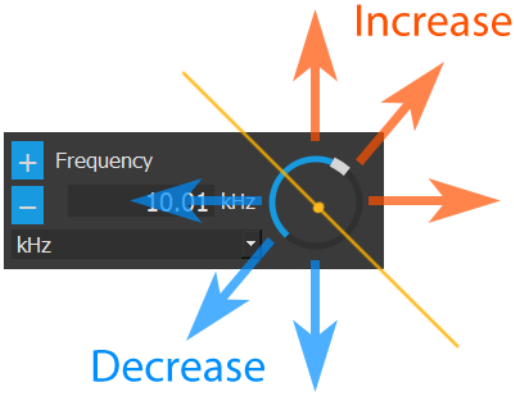


Figure 4. Mouse drag dial control

2. Modules

2.1. Oscilloscope

TODO

2.2. Counter

TODO

2.3. Voltmeter

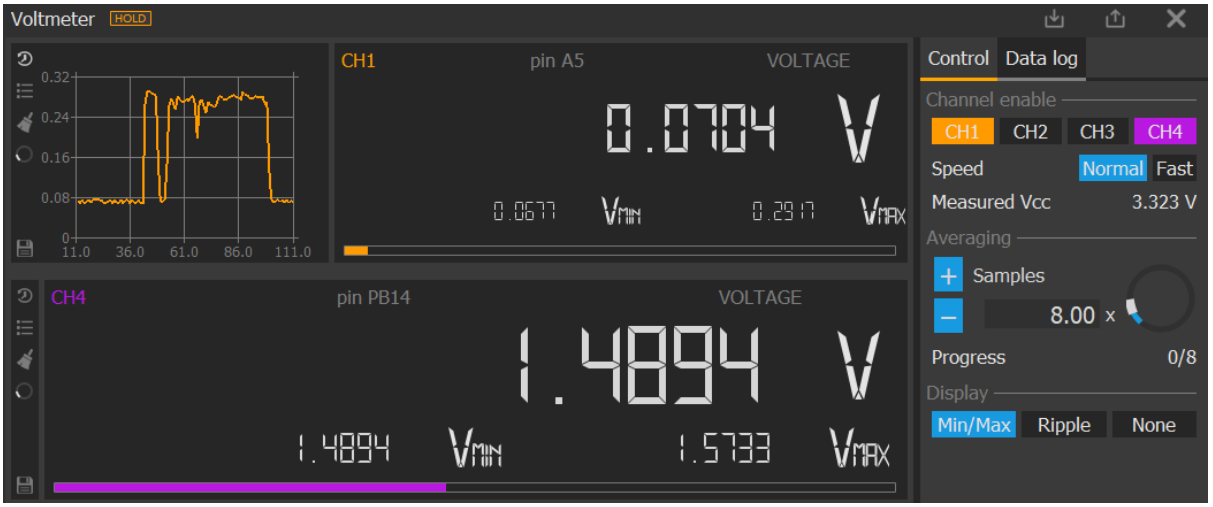


Figure 5. Voltmeter window

Voltmeter is based on the oscilloscope module and therefore they share the resources. The ADC sampling rate is 5ksps and data length 200 samples. Therefore the channel sampling time is fixed to 40ms. Shown value corresponds to the mean of the measured signal. There is a possibility to show the min max of the measured value or signal voltage ripple and its frequency.

The Vdd is sampled once every N samples to calibrate the measured values. Sampling time of Vdd is variable. With a higher number of samples for averaging the Vdd is sampled for a longer time to have better accuracy. The Vdd sampling can be skipped by setting the fast mode.

2.3.1. Data logger

The values from voltmeter can be logged into a file with standard .csv format. Datalog contains date and time when the sample was taken and corresponding Vdd.

	A	B	C	D	E	F	G	H
1	sample ID	date	time	Vdd	Voltage CH1	Voltage CH2	Voltage CH3	Voltage CH4
2	0	01.04.2021	11:39:59.611	3.323	0.0682914	0.0686606	0.209109	1.4729
3	1	01.04.2021	11:40:00.736	3.323	0.0718227	0.0728447	0.222619	1.57114
4	2	01.04.2021	11:40:01.541	3.323	0.0695935	0.06956	0.221218	1.57103
5	3	01.04.2021	11:40:01.874	3.324	0.0687527	0.0712899	0.223016	1.57208

Figure 6. Example of voltmeter datalog output file

2.4. Arbitrary generator

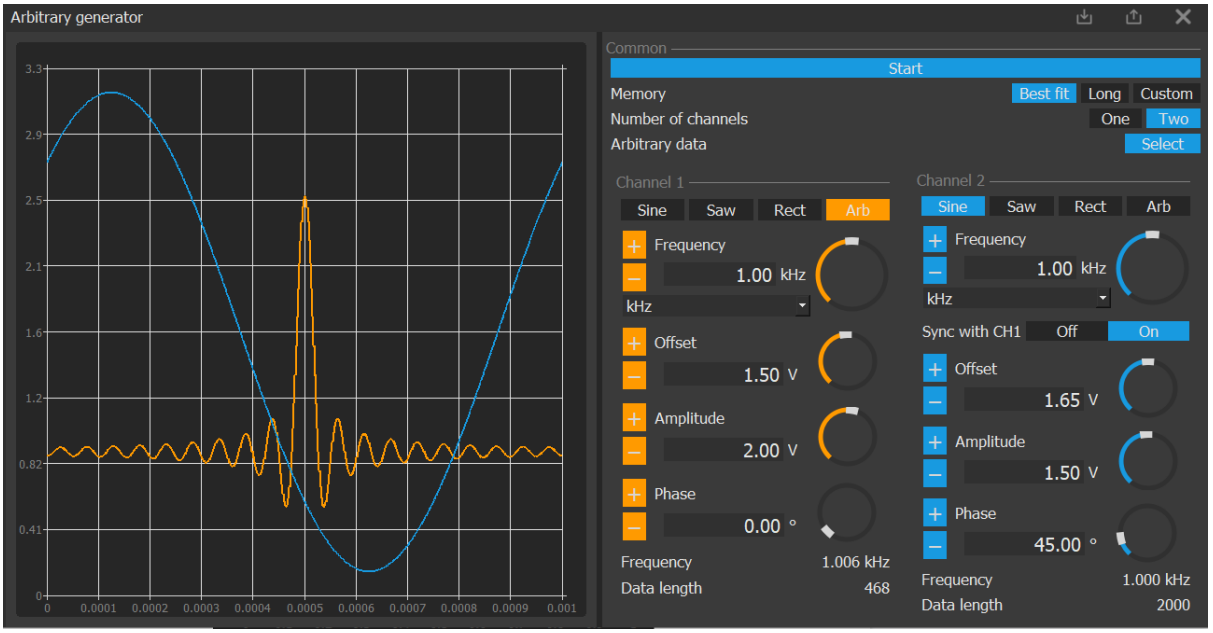


Figure 7. Arbitrary generator window

Arbitrary generator use timers, DMA and DACs to generate analog signals. The signal is represented in a memory as an array of samples and each time the counter overflows the next sample is pushed to DAC. This approach doesn't require any MCU processing time and is done fully in hardware. DDS is not used as it requires processing time and Sniffy would not be that universal.

DACs used in MCU have quite limited abilities in terms of maximum generating speed and output current. There is also some headroom from supply rails as the output buffer cannot go fully to the rails and gets saturated. This causes distortion when the signal is close to supply voltage or GND.

2.4.1. Signal Chart

Signal chart shows what the signals will look like when generated by DAC. Only one period of signal is shown.

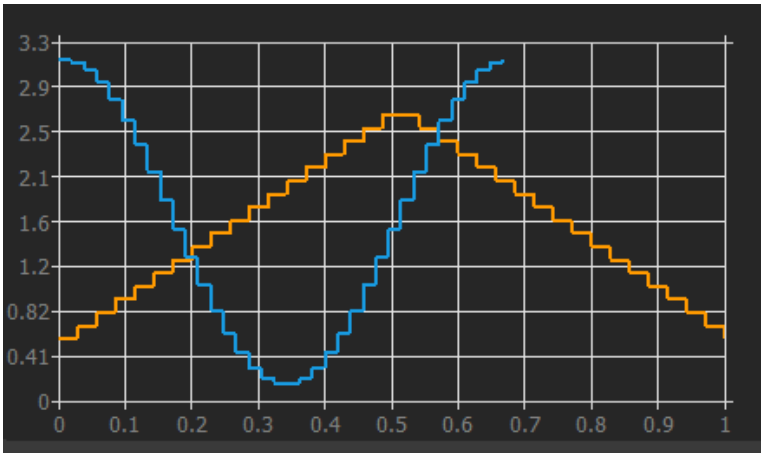


Figure 8. Chart with DAC real output

2.4.2. Memory length

Due to the generating principle used in sniffy the frequencies cannot be set accurately when full lenght of memory is used. The MCU clocks can be divided only by integer values therefore precise frequency cannot be achieved. There is a way to get precise frequency by adjusting the memory length. The actual memory length used, depends on user settings. Following modes are available: Best frequency fit, Longest possible and custom length.

The generating frequency and number of samples are shown in a module. When the generator is started the generating frequency is updated based on real values from MCU.

Best Fit - The length of the buffer is adjusted to achieve the lowest possible error. The minimal buffer length is limited to 1/4 of maximum length.
Long - The longest possible memory length for a given generating frequency is used.
Custom - User defined number of samples to be generated. Maximum frequency is limited accordingly to the maximum sampling rate.

2.4.3. On-the-go Frequency change

Frequency of the output signal can be changed even if the generator is running. The memory length is then locked and cannot be changed. The only parameter which can be changed is sample rate and therefore frequency setting is not accurate.
The maximum sample rate is limited by DAC abilities thus long signals have limited maximum frequency.

$$f_{max} = fs_{DAC}/n$$

For example if DAC max sample rate $fs_{DAC}$is 2MSPS, memory length $n = 1000$then the maximum output signal frequency f_{max} is *2MSPS / 1000 = 2kHz.*

2.4.4. Frequency and phase synchronization

When two channels are used the arbitrary generator offers synchronization frequency with channel 1. The phase is then synchronized too and will be as defined in GUI.

Changing of the frequency and sweep is enabled however phase synchronization might be lost especially for higher frequencies (usually above 10kHz).

Note: In case the phase has drifted due to frequency changes, try switching the channel frequency synchronization off and on to get the phase back.
Note: If the signal lengths don't match the frequencies will probably mismatch too.

2.4.5. Frequency sweep

Output frequency can be sweeped over set frequency range and time. The sweep is implemented in the Sniffy software and generating frequency is updated every 50ms with precision limited by Qt timers. The number of steps and list of frequencies is set by software.

The sweep affects only channel 1 and other channels can be joined by *Frequency sync with CH1* switch in their settings.

Note: The sweep can be enabled and parameters changed on-the-go while generating. All above limitations regarding maximum frequency, memory length and phase synchronization are valid.

2.4.6. Arbitrary file input

When Arbitrary signal type is selected the default sinc signal is loaded and GUI shows an option to load the file. File selector accepts .csv and .txt files either in standard format or any reasonable and parsable format. File parser is able to automatically detect the input format data, value separator, decimal separator, length of the signal, signal range, timescale and so on.

First row - It can be either data or labels. If it is labels (not parsable as a number) then the first row is skipped.

Value separator - Can be comma “,”, semicolon “;” or comma + space “, ”

Decimal separator - Comma or point are accepted.

Signal length - More channels can be inserted with different lengths.

Signal range - Data can be expressed either in voltage or in DAC raw values. (When maximum value in file is > 127 then all data are considered as DAC raw values)

Time scale - First column is identified as a time if the values are equidistantly split. Sampling frequency in gui is then updated accordingly

Parsed signals are shown in the chart and amplitudes with offset are estimated and updated in GUI. All signal controls are enabled and the user can change all parameters except length.

Note: In case of parsing errors the GUI shows the number of errors.

Examples of valid inputs:

```
time, data
0.01, 0.2
0.02, 0.5
0.03, 1
0.04, 1.5
```

```
time, data, data2
0,01, 0,2,  2
0,02, 0,5,  3
0.03, 1  ,  1
0.04, 1.5
0.05, 1.9
```

```
512
1024
2048
1536
1024
```

```
0,1
0.8
1.03
0.04
```

Figure 12. Valid arbitrary inputs