Zi² User manual

Here I'll tell you how to use the application as well as some other tips and tricks ! I'll also include an explanation of the structure of my code, so that if you want to customize your experience with the software you'll be able to without a hassle.

Table of content :

Getting started :

Prerequisites

There are two ways of running this application, either using an simple ***EXE*** for **Windows** that has to be launched within its folder (because they contain all the python modules, appliccation data etc...) and that requieres no installer.

Or by **running the source code** of the application with Python and all the modules. In wich case you'll need :

- **Python version 3.7.7** at https://www.python.org/downloads/
- **Numpy** module version ***1.19.0***, to deal with matrixes and data

  ```
  pip install numpy
  ```

- **Zhinst** module version ***20.1.1335*** *(from Zurich Instruments)*, to communicate with the HF2LI

  ```
  pip install zhinst
  ```

  or get it at https://pypi.org/project/zhinst/

- **Matplotlib** module version ***3.2.2*** *(upper versions don't work if you want the EXE)*, to do annimated 3D plotting

  ```
  pip install matplotlib==3.2.2
  ```

- **Datetime** module (usualy included in Python), to get the current time

  ```
  pip install DateTime
  ```

- **Pillow** module version ***7.2.0***, to load pngs to the GUI

  ```
  pip install pillow
  ```

- **Gwyfile** module version ***0.2.0***, to save microscope's images

  ```
  pip install gwyfile
  ```

The versions of the modules shouldn't cause any problems except for matplolib wich for some reason won't run in an EXE in versions upper than 3.2.2, I just included my versions in case you have problems and want to replicate my setup to avoid any conflict.

Installation

Like i said there no installer, just a folder wich contains all the necessary files. If you want the Windows EXE version than :

- **download** the Github repository as a ZIP
- **UnZIP** it and open it

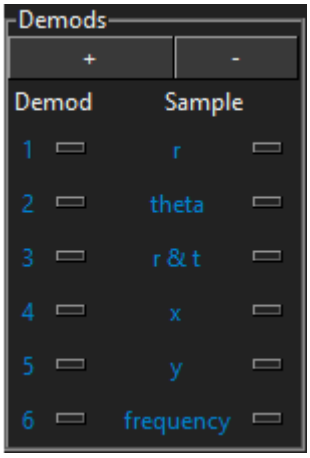
- **UnZIP** the *"Zi²"* part 1 *(had to be on two parts because >25MB)*
- **Open** the folder *"Zi²"* and find the executable named *"Zi²"*

If you want to run the source code than :

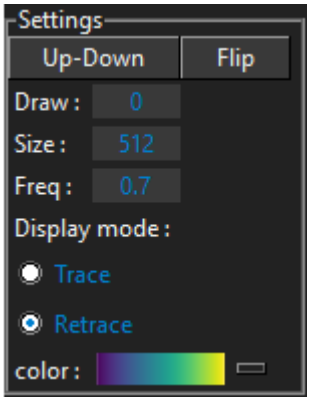
- **download** the Github repository as a ZIP
- **UnZIP** it and open it
- **Go to** the folder named *"Source code"*
- **find** the file *"Zi².py"*
- **Run** it with your editor (Python's IDLE, Vscode, etc...)

How to use :

Here you can see the "Command" part of the GUI which is divided in Tabs.

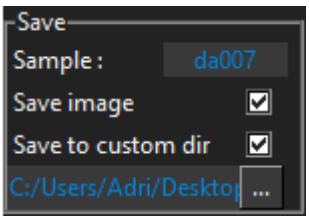


- **Demods** In which you can select the demod you want and the data you want to plot from it

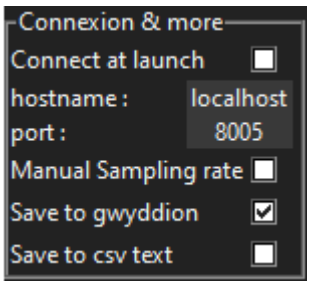


- **Settings** *(line by line)*
    - Tell the software in which direction the microscope is scanning by clicking on **"Up-Down"** which changes to **"Down-up"** and vice versa so that the application starts plotting in the right direction. If you make a mistake you can just click on **"Flip"** to Flip the image.

- **Draw** represents the number of Images you want to make, if you have the Nanoscope V and HF2LI then put 0, it will run continuously like the Nanoscope V does and synchronize with it. If you put another number than 0 the process won't be synchronized but it will make the number of images you requested. **TLDR : Let it at 0 unless you know what you're doing**
- **Size** is simply the size in pixels of the square images you want to make *(limits are 32 and 4096)*
- **Freq** is the frequency of line scanning (number of lines per second) **put the same as in the Nanoscope V software** (capped at 4hz)
- **Display mode** let's you select wether you want to see the trace or retrace of the AFM image, however both are saved so it's just for visualization
- **Color** is just the color grading for the scale in the 3D graphs

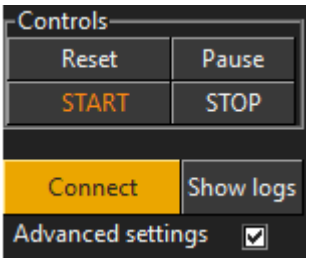


- **Save**
  - **Sample** is the sample name, it will change the name of the files that will be saved, the files will be saved in a different folder that will be named ***"Year-Month-Day"*** and the files in it will be ***"Hour-Minute-Second ; SampleName ; HF2LI id ; demods X sample,Y ; frame Z trace.format"*** and another named the same with **retrace** at the end. **X** is the demod number **Y** the demod data **Z** the number of the frame being saved and **format** the format which can be **csv** or **gwy TLDR name of sample**
  - **Save image** check if you want to save
  - **Save to custom dir** check if you want to save in the directory you've chosen instead of the default address
  - **Custom Directory Path** the path to your custom address, you can click in the **"..."** button to choose one instead of typing



- **Connexion & more** (appears if you check advanced settings)
  - **Connect at launch**, simply does the same thing as if you where to click the connect button but it does it automatically when you launch the application

- **Hostame and port**, let those at default value if you connect to the HF2LI via USB
- **Manual sampling rate**, allows you to overwrite the sampling rates set by the application *(**230khz** for the first demod, **1,8khz** for the rest)*in order to acquire the trigger signals correctly, read the for more information on that.
- **Save to gwy and save to csv** allows you to chose in which format you want to save your data, you can select both.

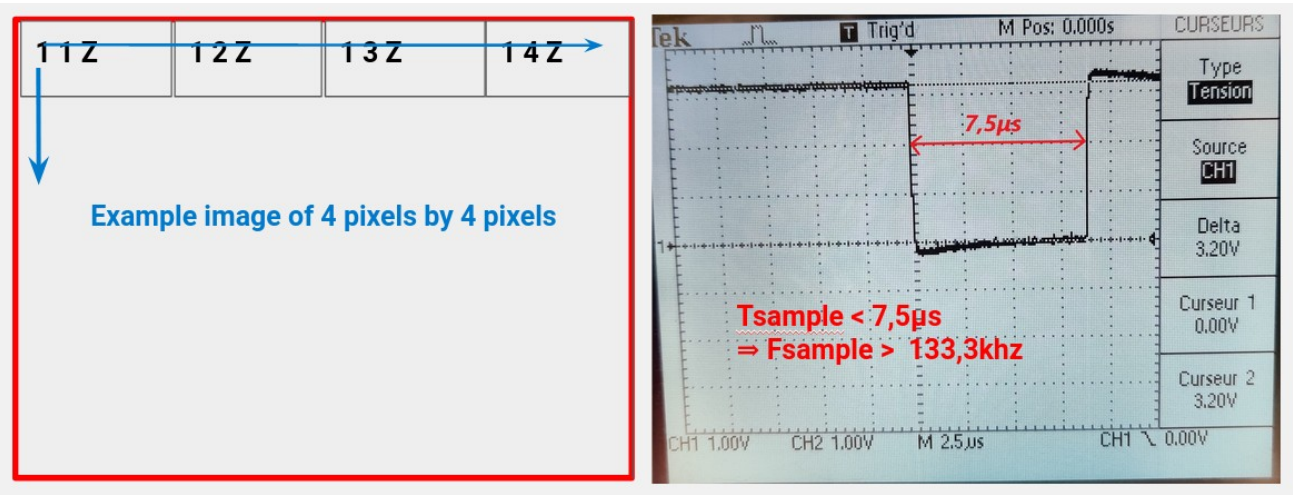


- **Controls**
  - **Reset** , resets all the images and waits for a Vertical Trigger from the Nanoscope V to plot
  - **Pause** , pauses the graphs but does not pause the sampling process, so it's just a graphical pause but not a real pause
  - **Start**, verifies that all the settings are correct, initializes the graphs and then waits for a Vertical Trigger from Nanoscope V to begin the sampling process and plotting
  - **Stop**, stops all the processes and clears the graphs, confirmation box before stopping

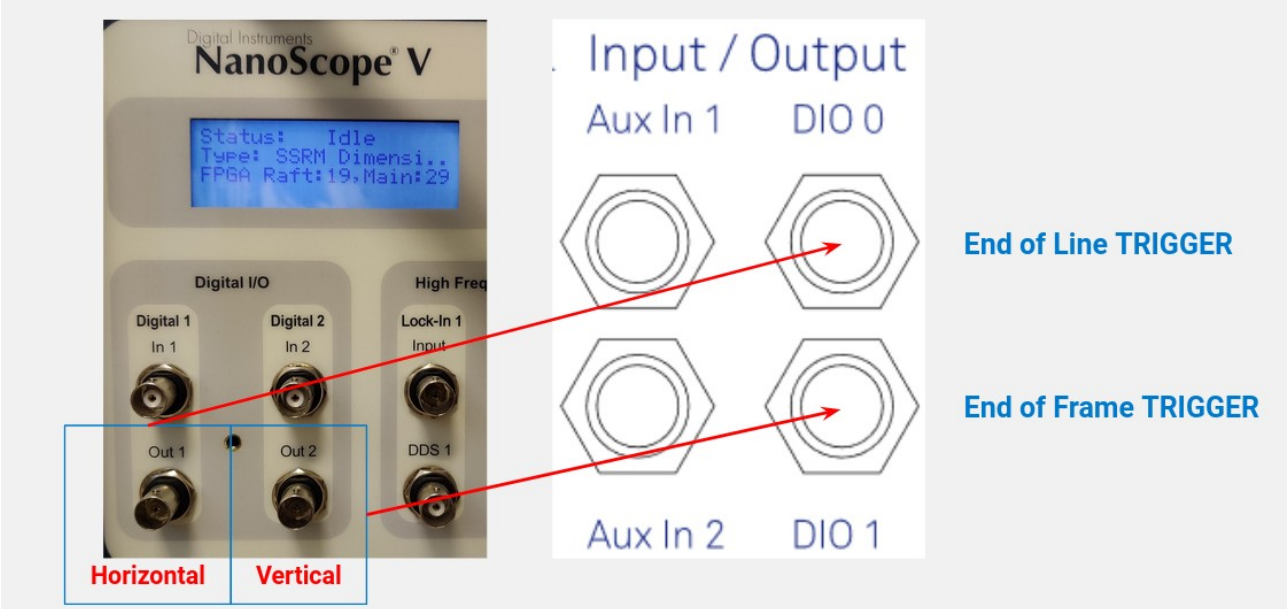
- **Connect** Connects to your HF2LI, make sure Labone is running
- **Show logs** Opens a window with logs and tells you what the program does, also tells you if there are problems so feel free to use it if something goes wrong.
- **Advanced settings**, make the "Connection & more" tab disappear if unchecked and vice versa.

Synchronization (only for Nanoscope V):

In order to make a good image you will need to be synchronized with the microscope. First let's see how the synchronization works on the Nanoscope V :



Let's imagine that the red lines are the « border » of our image, when the microscope tip touches the border it will generate a trigger signal on the corresponding output (output 1 for Horizontal triggering and output 2 for vertical). As we can see on the oscilloscope the sampling frequency must be higher than 133,33khz if we don't want to miss the trigger signals, that's why we put 230khz (next step on HF2LI).



We will use the End of Frame trigger as a start signal of the image, and the End of Line signal as a start signal for the current line being sampled and drawn

Example of how to use :

- First **connect to the HF2LI** using the connect button
    - the connect button will turn blue once it's done
- Then **select all the demods data** you want to see, you can add or remove data with the « + » and « - » buttons
- Verifies that all the settings are right (size, frequency, etc …)
- Check the « Save image » checkbox if you want to save
- **Press START**
    - check that the direction of the image plotting is right « Up down » if you want the image to be drawn from Up to down and vice versa
    - Wait for the microscope to reach the top or bottom of the image to then acquire the trigger signal and begin the plotting of data in the application

Common errors :

The main pitfall you might fall into is the the synchronization, if it does not work then you'll have empty white graphs. Please read what the Logs say to see if you have any errors than can be easily fixed.

For the rest, you'll unfortunately be on your own because i only had one day of testing my application with a real sample… I think i got rid of all the major bugs and for the most part the application works fine in it's intended use (between 256 and 512 pixels, 0.5 and 1.2hz, and 1 to 6 graphs)

good luck ;)

How to modify the code ?

The code is (at least in Visual Studio Code) arranged in «regions», those regions will contain different parts of the code to keep it cleaner and easier to go through. For example if you need to add an element to the GUI, let's say a button, you can create a region for this button like so :

```
#region Button I created

Your code

#endregion
```

You can also take other buttons as an example and you can see the region « Frames » to place your button in the right Tab of the GUI

```
  1   # This code has been made by Adrianos SIDIRAS GALANTE for the INL
  2 > # region LICENSE (EN) ⋯
 43 > # region Imports ⋯
 68 > # region dictionaries and variables definition ⋯
 96 > # region GUI colors theme ⋯
106 > # region GUI Root declaration and config ⋯
158 > # region GUI FRAMES ⋯
207 > # region GUI logs ⋯
270 > # region GUI logs scrollbar ⋯
280 > # region check all folders are present ⋯
284 > # region GUI graphs initialisation ⋯
300 > # region GUI Demodulators list to put in graphs ⋯
388 > # region GUI frames to draw ⋯
399 > # region GUI Size Text And Entry ⋯
406 > # region GUI Frequency Text And Entry ⋯
413 > # region GUI display modes ⋯
422 > # region GUI save image  ⋯
437 > # region GUI sample name + fieldnames ⋯
454 > # region GUI save to custom directory ⋯
569 > # region GUI button to Ask directory ⋯
601 > # region GUI choose the graphs color (CMAP) ⋯
633 > # region GUI display the frequency delta ⋯
638 > # region Animation part 1 ⋯
745 > # region Save image  ⋯
822 > # region Animation part 2 ⋯
841 > # region GUI start button + Checking variables befor launching ⋯
1058 > # region GUI pause ⋯
1074 > # region GUI stop ⋯
1176 > # region GUI reset button ⋯
1200 > # region GUI Up down dirrection ⋯
1236 > # region read the values stored between each launch ⋯
1249 > # region HF2LI detection and Connecxion ⋯
1301 > # region GUI Connection + Advanced settings ⋯
1387  root.mainloop()
```

You can also change the limits of frequency and size if you wish to do so. But keep in mind that the application might not run at it's optimal state (for example at 4hz the « show logs » does not work while the graphs are running)

```
# region dictionaries and variables definition
current_directory = (os.path.dirname(os.path.realpath(__file__)))
current_directory = current_directory.replace("\\", '/')

signal_paths = []  # setting up the array in which the signal adresses will be stored
frequency_index = []
data = {}  # setting up data dictionary
im = {}  # setting up ploted images dictionary
"""
here are the default limits of both size of the image in pixels and the nanoscope head quick axis frequency
which corresponds to the "x" axis scanning frequency of a line = trace + retrace
"""
size_min_limit = 31
size_max_limit = 4096
frequency_min_limit = 0.1
frequency_max_limit = 4.1

launch = 1
ani = None  # defining ani for animation as a global variable, otherwise it won't work with Tkinter and the GUI
"""
Here are defined the two triggers we'll be using wich are DIO 0 and DIO 1
those can be found at the back of the HF2LI and are to be plugged in with a coaxial cable
"""
end_of_line = '.TrigDio0'
end_of_frame = '.TrigDio1'
```

You can also change the color theme :

```
# region GUI colors theme
# default
light_gray_color = "#333333"
dark_gray_color = "#1e1e1e"
white_color = "#f1f1f1"
accent_color = "#007acc"
connect_color = "#eaa100"
start_color = "#ff830f"
selected_color = "#252526"
#endregion
```

However, the regions where you have all the « logic » of the program are Animation part 1, Save, Animation part 2 and Start. But this code will not be easy to change and i recommend to let it at it's default state unless you know what you are doing.

If you want to continue the project :

I'm at the end of my internship, the next thing i want to implement is multi threading for the buttons, so that they feel responsive even though you are plotting a lot of things …

Thank you for reading the manual