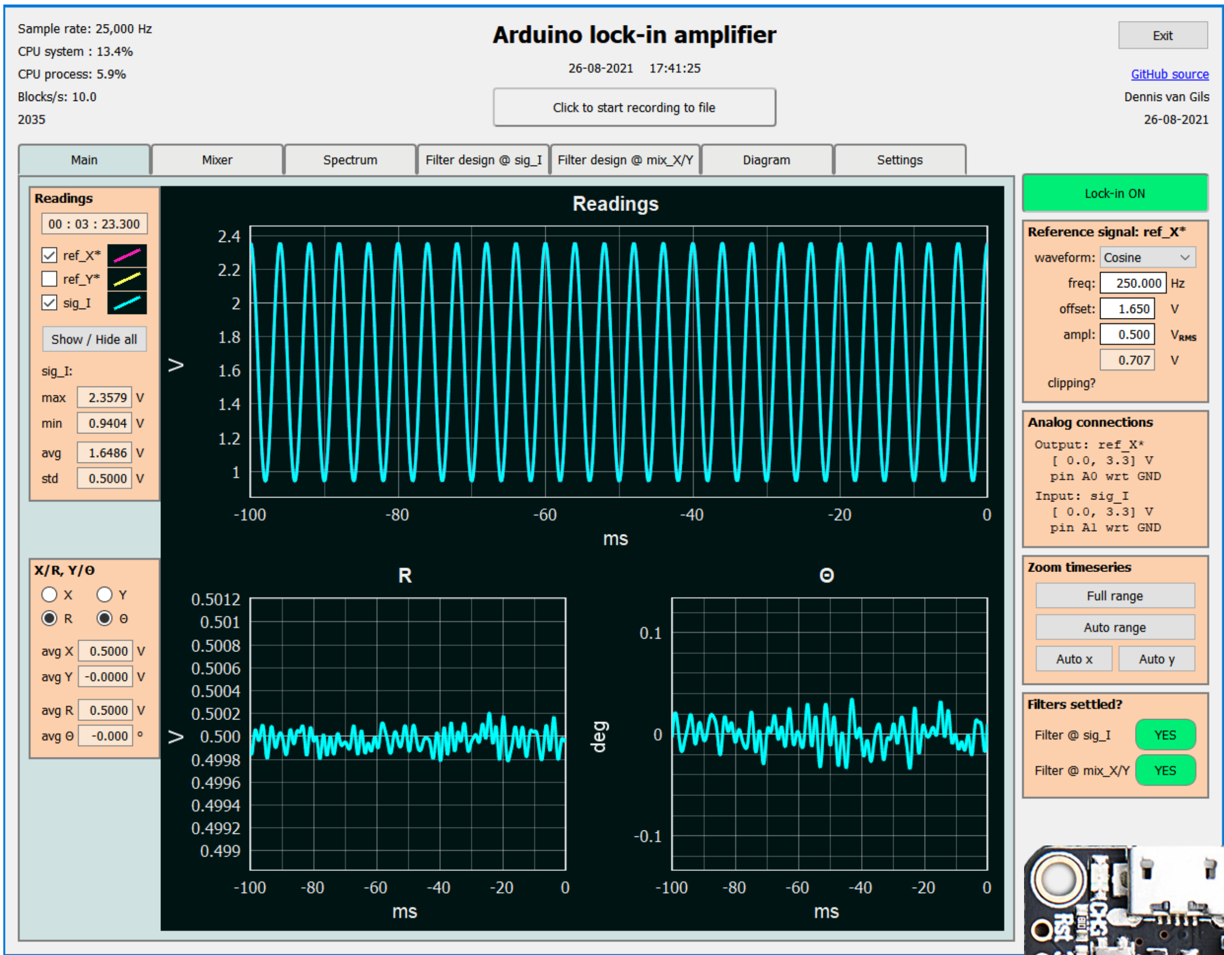
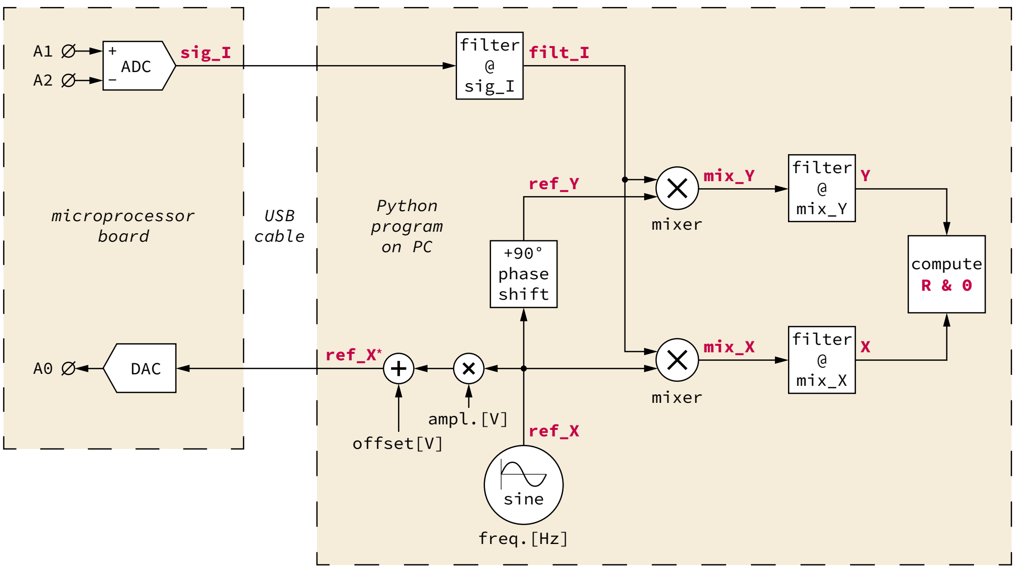
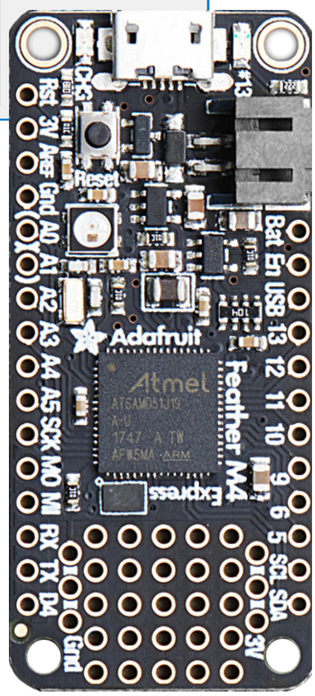
ARDUINO LOCK-IN AMPLIFIER
student user manual

Dr. ir. Dennis P.M. van Gils

August 31, 2021







This document describes a lock-in amplifier running on an Adafruit Feather M4 Express microcontroller board in combination with a laptop running Python. It is part of the lab assignments of the course *'Small Signals & Detection'* of the University of Twente, Enschede, The Netherlands.

Contents

1 Introduction

A lock-in amplifier is an electronic measurement device that is able to acquire small signals that otherwise would be washed out by noise. The underlying principle relies on offering a reference carrier wave at a fixed frequency to the device under test (DUT), i.e. your sensor circuit, and retrieving the response signal of the DUT. By locking in to the reference frequency embedded inside of the response signal, one can filter out noise sources and drastically improve the signal-to-noise ratio.

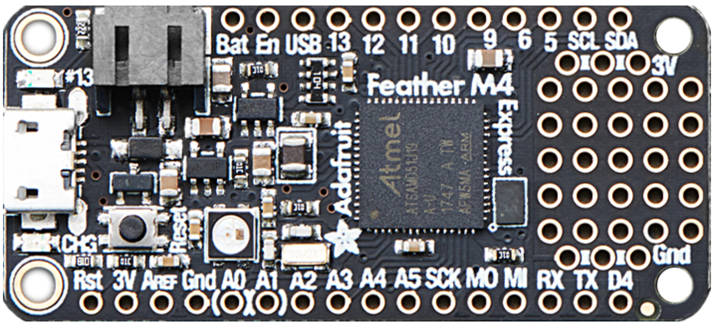


Figure 1: The Adafruit Feather M4 Express microprocessor board.

We use a microcontroller board (Adafruit Feather M4 Express, see fig. 1) to generate the output reference signal **ref_X***. Subsequently, it will also acquire the input response signal **sig_I**. This data is sent over USB to your laptop running the main graphical user interface in Python, see Fig. 2. The main Python program shows the waveform graphs of the signals in real-time, performs the heterodyne mixing and filtering of the signals similar to a lock-in amplifier, and provides logging to disk.

Although we use a microprocessor board from Adafruit, this project started out using an Arduino. Hence the name 'Arduino lock-in amplifier'. The name got stuck because either boards can be used in this project.

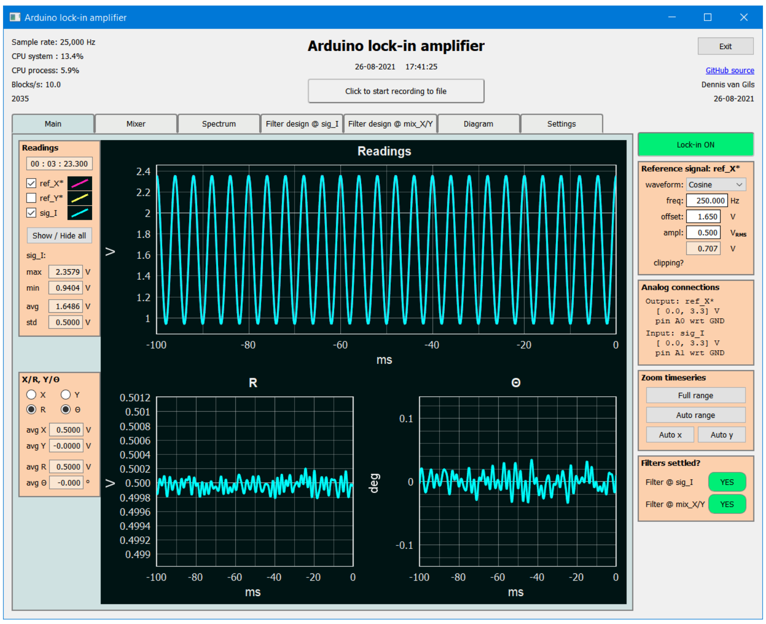


Figure 2: The main Python program showing the graphical user interface to control the 'Arduino lock-in amplifier'.

2 Software installation

Follow these steps in order to get the Arduino lock-in amplifier to work on your laptop.

2.1 GitHub source

We are going to download the lock-in amplifier program from GitHub. Browse to https://github.com/Dennis-van-Gils/DvG_Arduino_lock-in_amp/. Click on the green button labeled *'Code'* and chose *'Download ZIP'*. Extract the zip-file in the current folder. This will create the folder:

```
<Download folder>\DvG_Arduino_lock-in_amp-master
```

2.2 Python distribution

The preferred Python distribution is Anaconda Python. It comes in two sizes.

If you are concerned about disk-space or won't be needing Anaconda again after this practicum, you can use the minimal version called Miniconda. Download the Python 3.9 64-bit version suited for your operating system from:
https://docs.conda.io/en/latest/miniconda.html.

If, on the other hand, you want to later explore the full scientific power that Anaconda has to offer, you can download the latest full version from:
https://www.anaconda.com/distribution/.

Next, install the downloaded Anaconda Python on your laptop with the default install options as presented to you by the installer.

Now that the installation is finished, we are going to create an isolated Python environment called *'lia'* that will be used specifically for our lock-in amplifier. Because it is an isolated environment it will and can not interfere with other software on your operating system that also requires Python.

Start Anaconda Prompt and navigate to the folder:

```
<Download folder>\DvG_Arduino_lock-in_amp-master
```

> Hint: To list the current folder contents in Linux and Mac, you can use the command `ls`. For Windows you use the command `dir`. To navigate to a specific folder you use the command `cd` followed by a space and the folder you want to navigate to. Pressing `<TAB>` after the `cd` command will auto-complete any possible matches.

Enter these commands, line for line, in Anaconda Prompt and answer Yes when prompted:

```
> conda update -n base -c defaults conda

> conda create -n lia -c conda-forge  --force -y python=3.8.10

> conda activate lia

> pip install -r requirements.txt
```

> Remember: Every time you start a fresh Anaconda Prompt for this practicum, change to the *'lia'* environment. You do this by entering:
>
> ```
> > conda activate lia
> ```

2.3 Adafruit drivers

If you are running Linux, Mac or Windows 10 you can skip this section. If you are running an older Windows version you will have to install the Adafruit drivers to successfully connect to the Adafruit Feather M4 Express microcontroller board. Download the latest drivers at https://learn.adafruit.com/adafruit-arduino-ide-setup/windows-driver-installation and install with the default options.

2.4 Serial-port access for Linux and Mac

If you are running Linux or Mac you probably need to add rights to your user account granting you access to the serial-port interface with the microprocessor board. For Ubuntu Linux, and perhaps other distributions as well, you should run the command:

```
> sudo gpasswd --add ${USER} dialout
```

or,

```
> sudo usermod -a -G dialout $USER
```

and log out and in again.

Another solution might be to run the main Python program described in section §2.6 as a super-user by calling:

```
> sudo ipython DvG_Arduino_lockin_amp.py
```

For further help, please search online the keywords: `permission serial port {your OS}`.

2.5 Flashing firmware

Before the Adafruit Feather M4 Express can be used as a lock-in amplifier it needs to be flashed with the correct firmware. **This is already done for you.** If the firmware got corrupted somehow, you can perform the following procedure to restore it:

Connect the M4 board via USB to your laptop. Quickly double-click on the tiny reset button of the M4 board. The LED on the board should turn green as indication that we have entered Bootloader Mode. Once the bootloader is running, check your computer. You should see a USB Disk drive called `'FEATHERBOOT'`. This folder contains the file `CURRENT.UF2` which is the current contents of the microcontroller flash. You can upload new firmware to the M4 board by copying over this file. The lock-in amplifier firmware can be found in the downloaded GitHub source files from §2.1 in folder:

```
\mcu_firmware\v1.0.0_VSCODE\adafruit_feather_m4__25kHz
```

The M4 board should automatically reboot when the file is copied over and will now be running the lock-in amplifier firmware.

2.6 Running the main Python program

Connect the M4 board via USB to your laptop. Start Anaconda Prompt and navigate to the folder `<Download folder>\DvG_Arduino_lock-in_amp-master`.

> Hint: To list the current folder contents in Linux and Mac, you can use the command `ls`. For Windows you use the command `dir`. To navigate to a specific folder you use the command `cd` followed by a space and the folder you want to navigate to. Pressing `<TAB>` after the `cd` command will auto-complete any possible matches.

The Anaconda Prompt should now look something like:

```
(base) C:\Downloads\DvG_Arduino_lock-in_amp-master>
```

Next, we have to change to the *'lia'* environment. You do this by entering:

```
> conda activate lia
```

The Anaconda Prompt should now look something like:

```
(lia) C:\Downloads\DvG_Arduino_lock-in_amp-master>
```

Now we are ready to run the main program by entering:

```
> ipython DvG_Arduino_lockin_amp.py
```

This will start a graphical user interface to control the Arduino lock-in amplifier.

3 Hardware information

3.1 Analog out

The digital-to-analog converter (DAC) of the microprocessor is programmed to output a sinusoidal waveform between 0 to 3.3 Volts at 12 bit resolution and at a 25 kHz sample rate. The frequency, amplitude and offset of the waveform can be set by the user in the main Python program. This output signal is called `ref_X`* and is available on pin `A0` with respect to the `GND` pin. The maximum frequency of `ref_X`* is limited to 1250 Hz.

3.2 Analog in

The analog-to-digital converter (ADC) of the microprocessor is programmed to acquire a signal between -3.3 to 3.3 Volts at 12 bit resolution and at a 25 kHz sample rate. This input signal is called `sig_I` and can be acquired by connecting up pins `A1` and `A2`. Pin `A1` should be connected to the positive side of the input signal, and pin `A2` to the negative side of the input signal which could be the electrical ground of the device under test, i.e. your sensor circuit. The input signal `sig_I` is plotted in the main Python program and its power spectrum can be investigated in real-time.

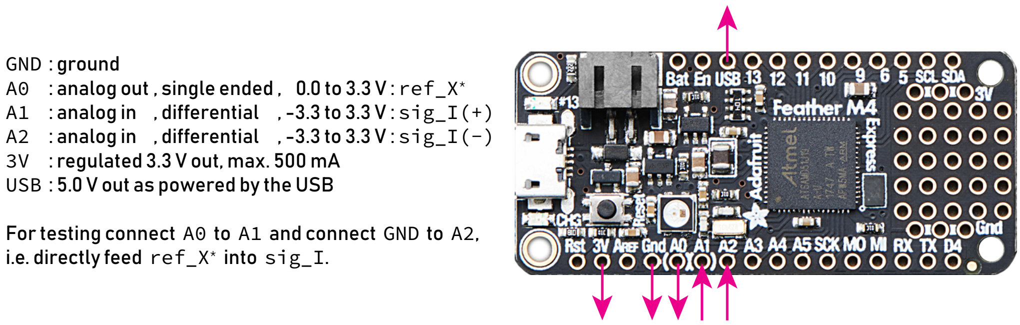


Figure 3: Pin-out of the Adafruit Feather M4 Express microprocessor board.

3.3 Grounding

IMPORTANT:

Always connect the `GND` pin of the microprocessor board to the ground of the device under test. Failing to do so can result in unstable signals and might even damage the electronics.

4 Lock-in amplifier signal processing

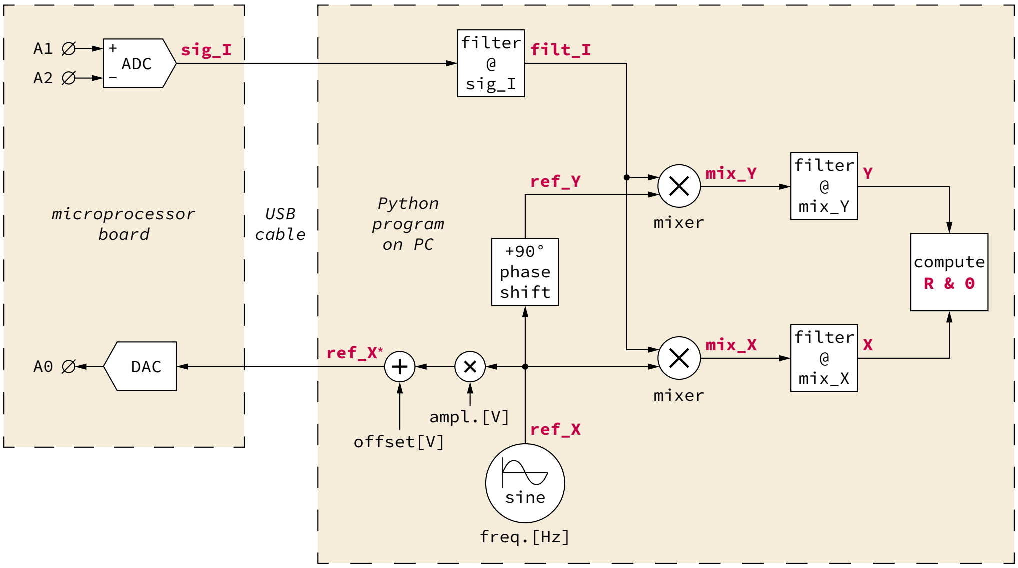


Figure 4: Diagram of the variable names and signal processing steps as defined and used by the Arduino lock-in amplifier. The starting points to follow in the diagram are the ADC-converter signal labeled `sig_I` and the cosine reference signal labeled `ref_X`. Follow the direction of the arrows to see the subsequent signal paths and operations.

5 Graphical user interface

This section shows the graphical user interface (GUI) of the lock-in amplifier running in Python.

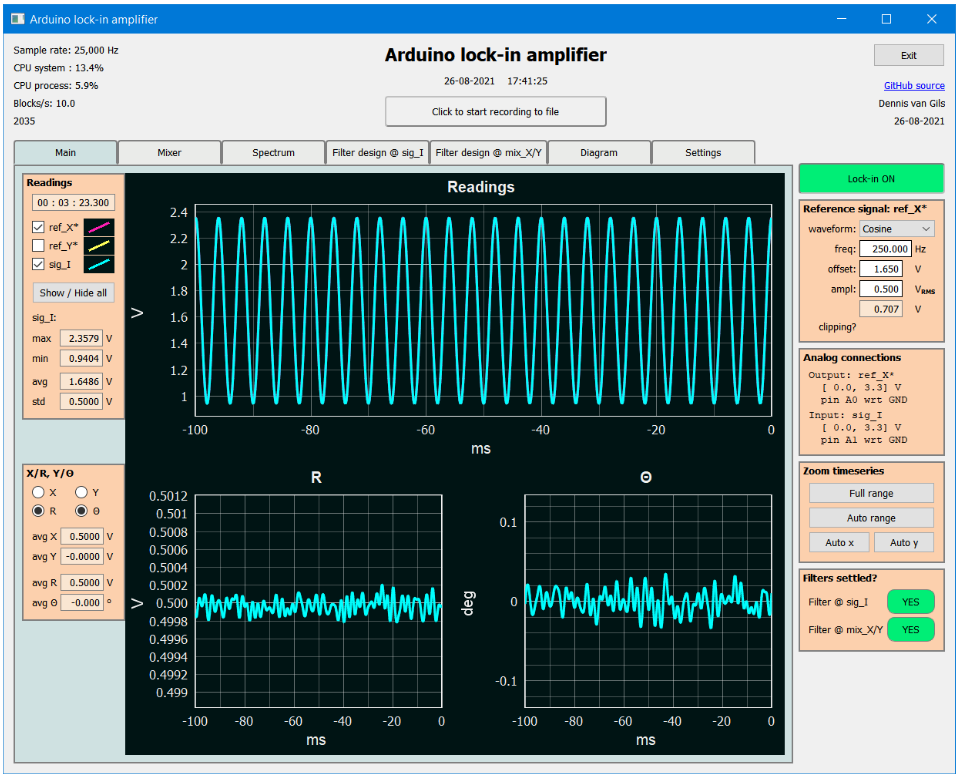


Figure 5: The *Main* tab page of the lock-in amplifier GUI.

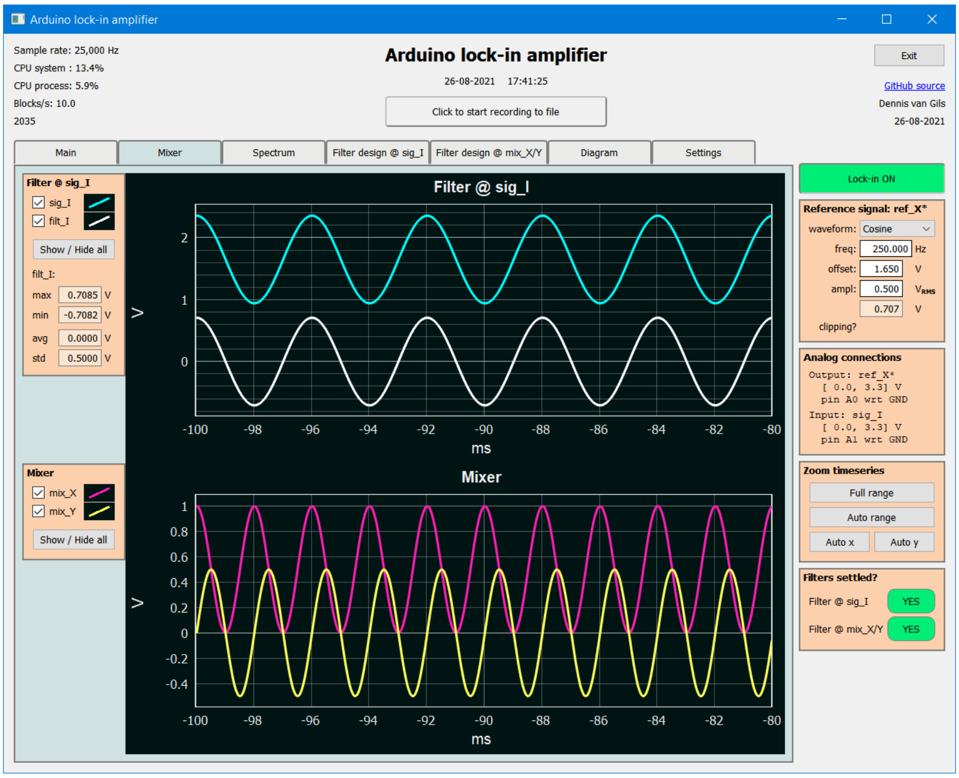


Figure 6: The *Mixer* tab page of the lock-in amplifier GUI.

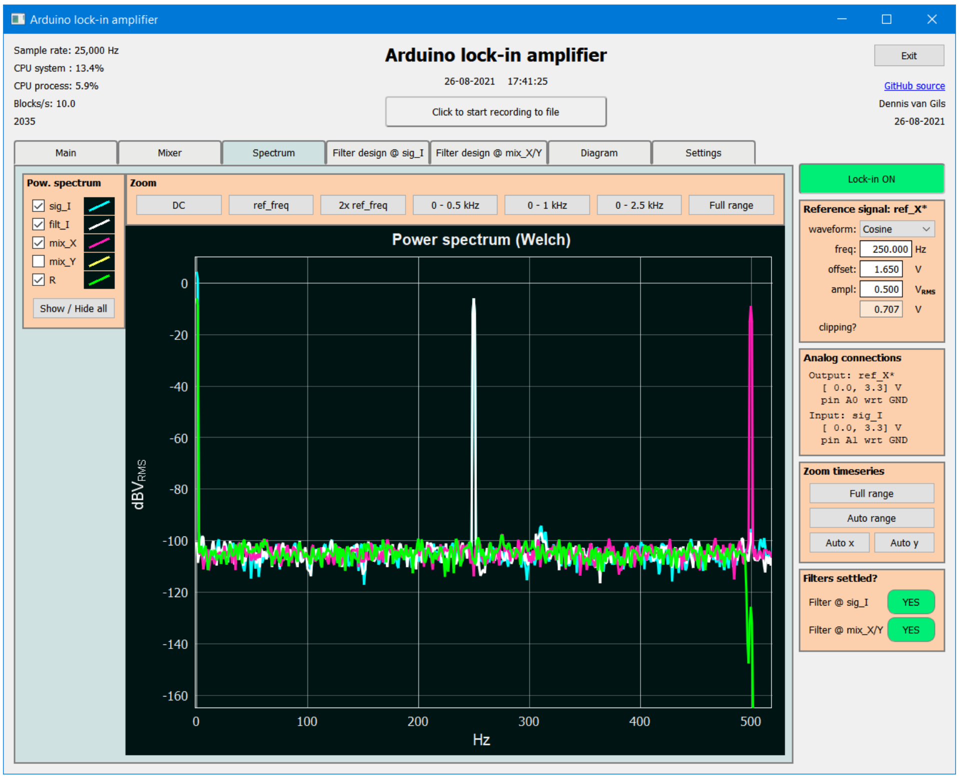


Figure 7: The *Spectrum* tab page of the lock-in amplifier GUI.

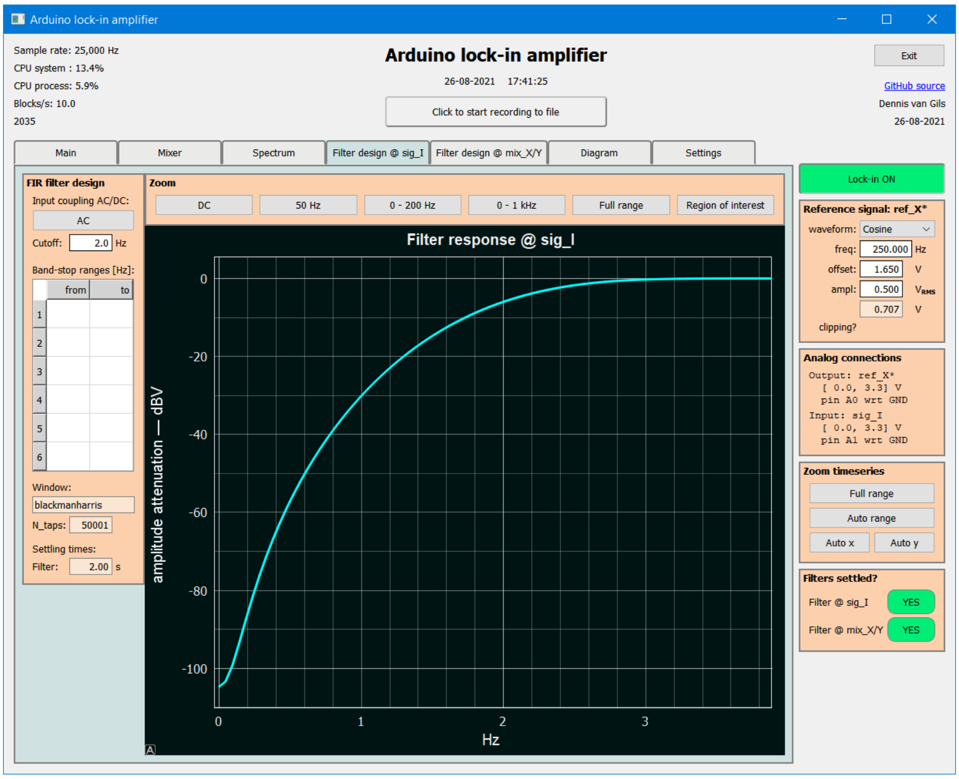


Figure 8: The *Filter design @ sig_I* tab page of the lock-in amplifier GUI.

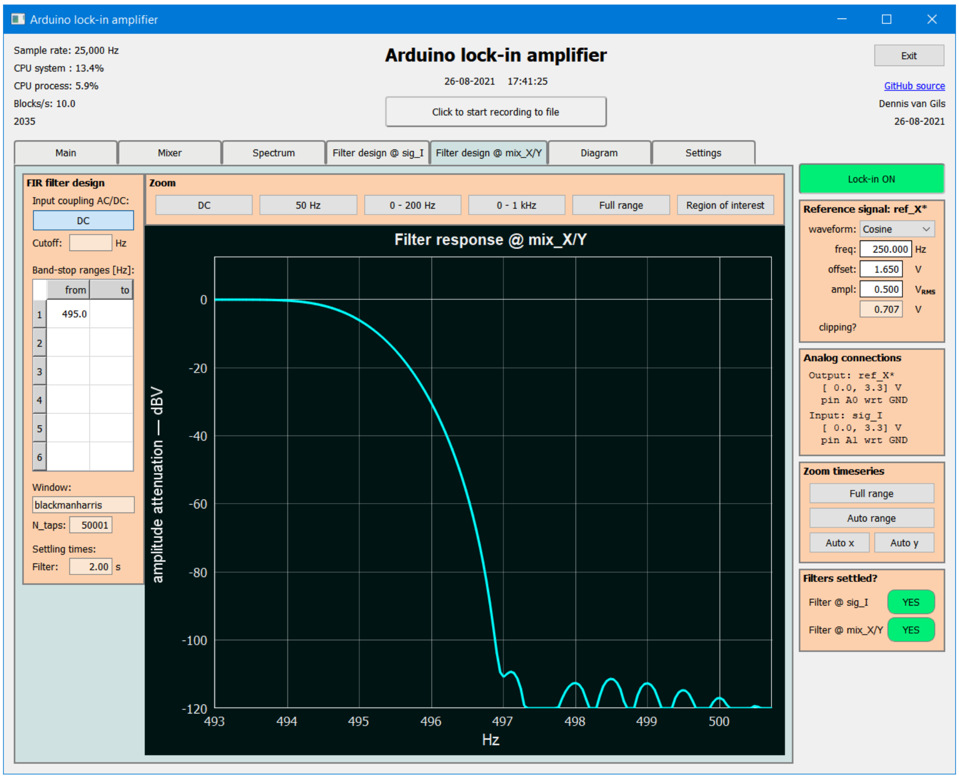


Figure 9: The *Filter design @ mix_X/Y* tab page of the lock-in amplifier GUI.

6 Troubleshooting

1) I have spikes in my voltage signal that should not be there.
Try running your laptop on battery power only and unplug the adapter from the power socket. Your laptop might have a low-quality or failing power adapter or battery.

2) My voltage signal seems to be unstable and/or drift.
Did you forget to connect the grounds together of the microprocessor board and the electronic circuit that is under test? And did you connect both input pins `A1` and `A2`?

3) I get error messages:
 * `Permission denied /dev/ttyS0` or `/dev/ttyACM0` or similar.
 * `Device not found`

You might lack the rights to access the serial port. Please see section §2.4.

4) The python program does not show the graphical user interface in Linux.
You might need to install the following:

```
> sudo apt-get install libxcb-xinerama0
```