The monitoring softwareThe document is outdated

Egor Tsvetkov

August 2022

Contents

Chapter 1

Installation

1.1 Device setup

1.2 Ubuntu linux

```
cat /etc/udev/rules.d/60-yokogawa.rules
```

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0b21", ATTR{idProduct}=="0029", MODE="0666"
```

logout and login

1.3 Windows

This section contains information on how to install the Software on Windows 10, but we hope that these steps work fine on Windows 7, Windows 11 or any future version.

1.3.1 Checking prerequisites

Python interpreter of version 3.10 or higher should be installed and added to the system path. We recommend to install the official Python distribution that can be downloaded from `https://www.python.org/downloads/`. The Python distribution from the Windows store doesn't work[1].

On Windows, you can check whether Python is installed by launching command prompt and typing

```
C:/Users/UserName>py --version
Python 3.10.5
```

For installation from sources, the computer should be connected to Internet to download the required libraries.

1.3.2 Driver installation

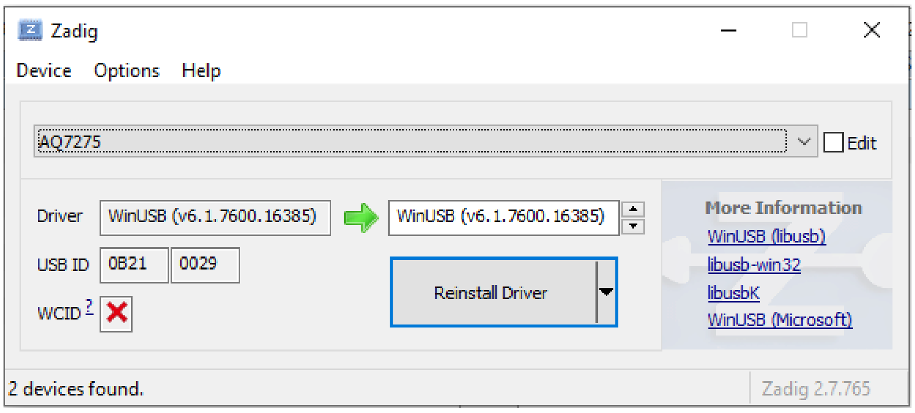
The Software uses the `pyusb` library, which is a Python wrapper to the well known C library `libusb`. This library came from the world of Linux. Since Windows provides a different driver model to communicate with USB devices, the following additional step should be made.

The simplest way to install the correct driver is to use the Zadig software. This software is free and open source. It can be downloaded from `https://zadig.akeo.ie` or using the direct link `https://github.com/pbatard/libwdi/releases/download/v1.4.1/zadig-2.7.exe`.

To install generic winusdb driver, follow the instructions below (see Fig. 1.1).

[1]The Python distribution from Windows store doesn't create `py.exe` file used to find the correct python version.

1. Run `Zadig.exe` as Administrator
2. Connect Yokogawa AQ7275 to a USB port of your computer.
3. Wait until Yokogawa AQ7275 appears in the drop box. Select this device in the drop box. If it didn't appear, try to uncheck and check again 'Options/List all devices' from the main menu.
4. Select the WinUSB driver.
5. Press 'Replace driver' or 'Reinstall driver'.



AQ7275
Driver WinUSB (v6.1.7600.16385) → WinUSB (v6.1.7600.16385)
USB ID 0B21 0029
WCID ✗
2 devices found.
Zadig 2.7.765

Figure 1.1: Zadig window with selected WinUSB driver for Yokogawa AQ7275 device

The winusb driver should be installed only once. If Zadig software shows that winusb driver is already assigned to AQ7275, there is no need to install it again.

1.3.3 Installation

Unpacking

First, we need to extract sources from the archive using any appropriate software. Let PATHTOSOURCES denote the path where the sources are placed, for example, `D:\monitoring`. This path should contain the following files:

```
monitoring
 |__...some subfolders
 |__...some other files
 |__requirements.txt
 |__install.bat
 |__monitoringapp.py
 |__monitoringapp.bat
 |__libusb-1.0.dll (on Windows)
```

Automatic installation on Windows

The simplest way to install the program is to execute `install.bat` by double-clicking or by typing the following commands in the command prompt:

```
cd <PATHTOSOURCES>
install.bat
```

In case of problems, please, follow the commands in the next subsection to trace where the problems arises.

Manual installation on Windows

The `requirements.txt` file is sufficient to download and install all the dependencies that are not included in the sources. To do this, we need to create the virtual environment by typing the following commands: [2]

```
cd <PATHTOSOURCES>
py -m venv monitoring-env
monitoring-env\Scripts\activate.bat
pip3 install -r requirements.txt
```

1.3.4 Running the program

To run the program, just execute `monitoringapp.bat` by double-clicking. The second way to start the program is to type

```
cd <PATH-TO-SOURCES>
set PATH=.;%PATH%
monitoring-env\Scripts\activate.bat
python3 -m monitoringapp
```

[2]These commands create the virtual environment in the separate directory named `monitoring-env`. In general, the path to the virtual environment can be arbitrary, but the good choice is to place the virtual environment near the sources (or inside the sources).

Chapter 2

GUI

The Software consists of three main parts:

1. The GUI part is responsible for graphical representation of the Software and interaction with the user. The GUI part is a glue that connects all other parts together. It is based on PySide6 library which is a Python port of the famous QT library.

2. The driver part is responsible for communications with the reflectometer. It provide the high-level interface functions such as ‘connect’, ‘get parameters’, ‘set parameters’, ’run measurements’ and so on. When one of these functions is called, this part translates it to the appropriate command codes of the Reflectometer (see AQ7270 Series OTDR Communication Interface).

3. Plugins are responsible for data processing and visualisation. When the new reflectogram is obtained, it is sent to all active plugins.

2.1 Control buttons

- Single measurement
- Monitoring
- Autosave

2.2 Plugins

2.2.1 Averager

The averager plugin collects reflectograms, calculates an average and estimates the deviation of the current reflectogram from the average.

We use the following designations.

- Vector y is the recently measured reflectogram (the current reflectogram).
- M is the total number of points (samples) in each reflectogram.
- j is used to enumerate points, $j = 1, \ldots, M$.
- x_j is the distance to j-th point.
- The graphical representation of reflectogram y can be considered as the set of points

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}\,.$$

- Matrix Y contains all previously measured reflectograms concatenated horizontally as columns. In other words, Y_{ji} is the j-th point of i-th reflectogram. The size of Y is $M \times N$, where N is the number of reflectograms stored in Y.

The following steps are made when the new reflectogram is obtained. First, we calculate the **mean reflectogram** as

$$\bar{y}_j = \frac{1}{N} \sum_{i=1}^{N} y_{ji}, \tag{2.1}$$

and the unbiased sample variance as

$$\sigma_j = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (y_{ji} - \bar{y}_j)^2}. \tag{2.2}$$

The **upper and lower bounds** are calculated as

$$y_j^{lower} = \bar{y}_j - n\sigma_j, \tag{2.3}$$
$$y_j^{upper} = \bar{y}_j + n\sigma_j, \tag{2.4}$$

where n is some number, usually n is chosen from interval $[2, 5]$.

The **violation level** is calculated as

$$V = \frac{1}{j_b - j_a + 1} \sum_{j=j_a}^{j_b} \mathbf{I}\left\{y_j < y_j^{lower} \vee y_j > y_j^{upper}\right\}, \tag{2.5}$$

where indexes j_a and j_b define the segment of interest $[j_a, j_b]$. These indexes are calculated using the following formulas:

$$j_a = \max\{j : x_j < a\}, \tag{2.6}$$
$$j_b = \min\{j : x_j > b\}, \tag{2.7}$$

where a and b are the left and right boundaries of the segment, expressed in metres.

Finally, we compare V with threshold Θ. There are two cases.

1. If $V \leq \Theta$, then the current reflectogram is appended to Y. If the number of columns in Y becomes greater than some predefined value, the most oldest columns are removed.

2. If $V > \Theta$, the alarm is turned on. In this case, the current reflectogram y is not appended to the end of Y.

The parameters selection window is shown in Fig. 2.1. The parameters have the following sense.

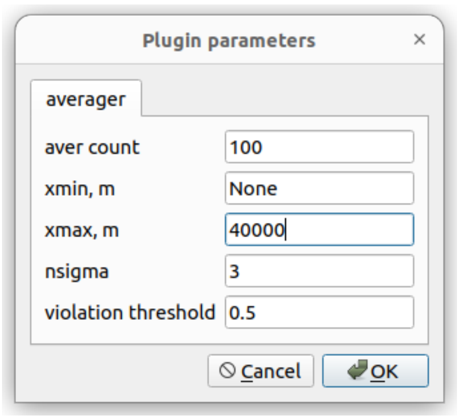
- 'Aver count' is the maximum number of reflectograms stored by the plugin. In other words, it is the maximum number of columns in Y.

- 'xmin, m' is the left boundary of the segment of interest, in meters. If it equals to None, then it means that xmin=0.

- 'xmax, m' is the right boundary of the segment of interest, in meters. If it equals to None, then it means that xmax is selected to be equal to the actual number of samples, i.e. xmax $= M$. (corresponds to the right boundary of the reflectogram).

- 'nsigma' corresponds to n in eq. (2.3).

- 'Violation threshold' corresponds to variable Θ defined above.



Plugin parameters

averager

aver count 100

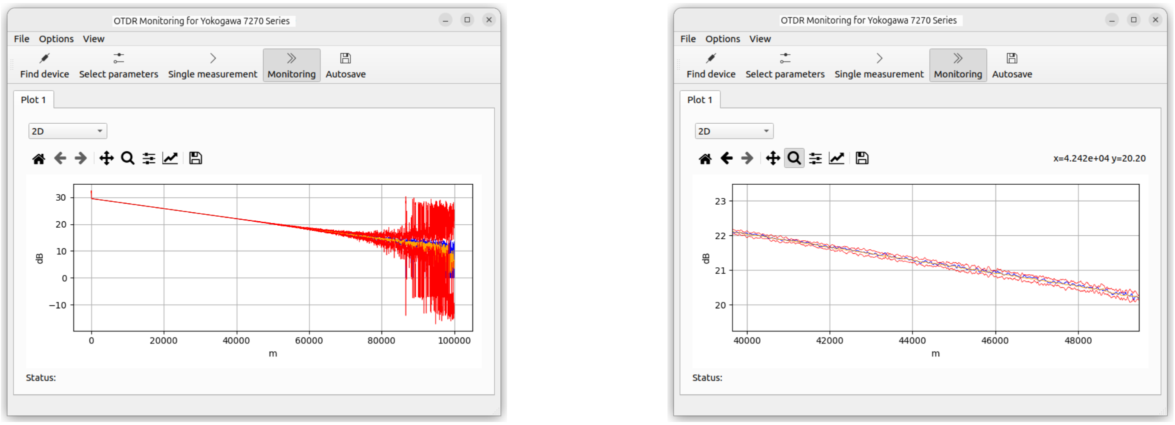
xmin, m None

xmax, m 40000

nsigma 3

violation threshold 0.5

Figure 2.1: The parameters of the averager plugin



(a) The default view of the reflectogram. (b) The magnified view.

Figure 2.2: The reflectogram in conventional axes. Red curves represent lower and upper boundaries. The blue curve is the current reflectogram. The orange curve is the mean reflectogram.

2.2.2 Final remarks

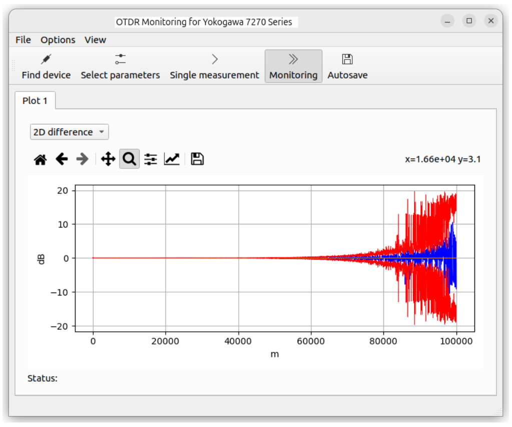
- If N is low, the mean reflectogram is unreliable. Therefore, the averager plugin doesn't calculates the mean value, lower and upper boundaries if N is less than 5. All new reflectograms are stored to Y.

- In practice, reliable statistics can be obtained if N is large (at least, $N > 100$). It should be guaranteed that there is no eyedropper on the initial phase.

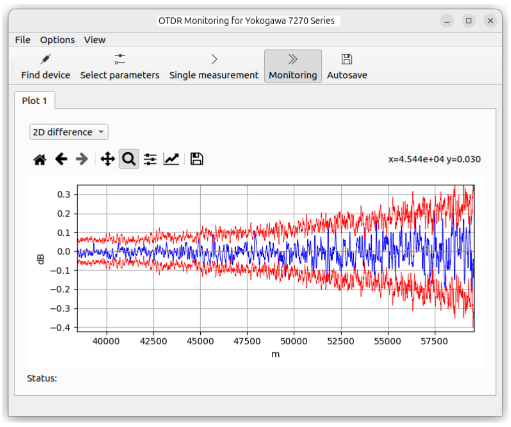
2.2.3 Plotters

The averager plugin has two plotters to represent it state. First plotter draws the current reflectogram, mean reflectogram, and lower and upper boundaries in conventional axes. The vertical axis shows the signal level in dB, that is read directly from the device.

The second plotter shows the deviation of the current reflectogram and lower and upper boundaries from the mean. The vertical axis is graduated in dB too, but shows the difference of the signal levels.



(a) The default view of the reflectogram.



(b) The magnified view.

Figure 2.3: The deviation from the mean. As above, red curves are the lower and upper boundaries, and the blue curve is the current reflectogram. The orange horizontal line is $y = 0$.