A-ULIS

Advanced Universal Lidar Interface Suite

Contents

Change log:

Date	Author	Description
23.07.2014	Szymon Szyszko	First version of the document

Szymon Szyszko

Institute of Geophysics, Polish Academy of Sciences, 2014

1. Abstract

Lidar is a system that measures distance to a target by detecting light scattered by that target. Atmospheric lidar in the Institute of Geophysics, Polish Academy of Sciences is used to investigate water vapour content in the atmosphere and detect aerosol presence. ULIS – Universal Lidar Interface Suite is the original software for data collection. This document describes a new software for lidar measurements called A-ULIS. New software development was possible thanks to reverse engineering techniques. Using the USBlyzer – USB Protocol Analyzer and USB Traffic Sniffer (see www.usblyzer.com) communication process through USB interface between the ULIS application and detectors has been discovered and sufficiently well understood. The A-ULIS was written in C# 5.0 using Visual Studio Express 2013 programming environment.

Chapters 3, 4, 5 are particularly useful to a lidar operator, while chapters 6, 7, 8, 9 provide more information for one who would like to improve or add new features to the A-ULIS application.

2. Motivation

Instability and limitations of the ULIS software delivered by the manufacturer were the main motivation to develop a new tool for lidar measurements. The manufacturer does not provide the source code of ULIS, thus no improvements were possible. Having own code opens doors for developing software better adjusted to a lidar operator’s needs. Moreover, no way to run the ULIS software on Windows 7 or above is a known issue. A-ULIS successfully runs on Windows 7 however due to no version of .NET 4.5 released for Windows XP, it is not possible to run the application on Windows XP.

3. System requirements

Following conditions has to be met to start and use the application successfully:

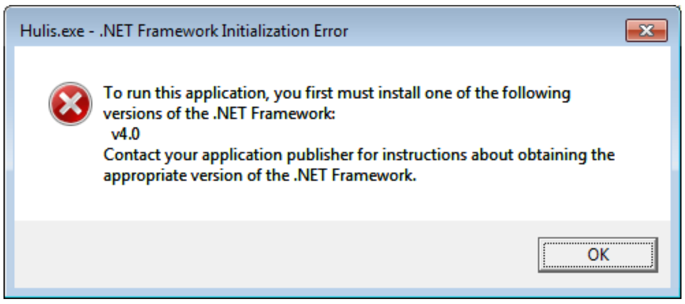
- installed .NET framework at least version 4.5
- operating system Windows 7 (tested), Windows 8 (not tested), Windows 8.1 (not tested)
- cyusb.sys driver installed (install it with CySuiteUSB downloaded from www.cypress.com)
- screen resolution at least 1280 x 768

4. Installation

There is no installer for the application. However, few steps should be taken to run the Hulis.exe file successfully. Follow these steps:

1) Check whether the cyusb.sys driver is installed. Look for the cyusb.sys file in C:\Windows\System32\Drivers directory. In case there is no cyusb.sys file install CySuiteUSB. The application is developed using 3.4.7 version of CySuiteUSB. By default, CySuiteUSB is installed in C:\Cypress. Follow instructions in C:\Cypress\Cypress Suite USB 3.4.7\Driver\CyUSB.pdf to assign the driver to a device. You will need to edit cyusb.inf file. Already edited cyusb.inf file is attached. CySuiteUSB installation file - CySuiteUSB_3_4_7_B204.exe is also attached.

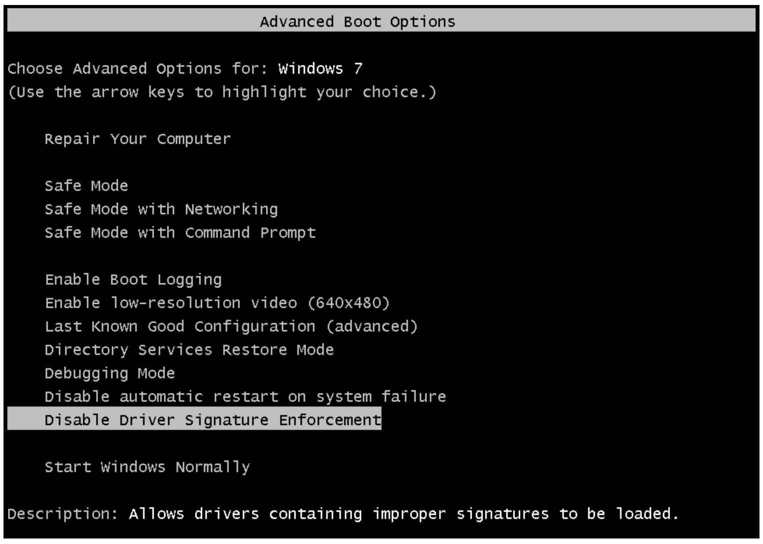
2) Make sure if proper version of .NET framework is installed. Try to run Hulis.exe. When .NET Framework initialization error message is displayed (see figure 1), you need download and install .NET Framework 4.5 though the message mentions version 4.0. Download .NET 4.5 installer from Microsoft pages.


Hulis.exe - .NET Framework Initialization Error

To run this application, you first must install one of the following versions of the .NET Framework:
 v4.0
Contact your application publisher for instructions about obtaining the appropriate version of the .NET Framework.

Figure 1: .NET framework initialization error message.

3) Finally after the cyusb driver and .NET 4.5 are installed it is possible to run the application. The cyusb driver is not digitally signed so you should disable driver signature enforcement. To do so while Windows 7 boots press F8 for advanced boot options. You will see a screen similar to one presented in figure 2. Choose “Disable Driver Signature Enforcement” option.


```
                          Advanced Boot Options

Choose Advanced Options for: Windows 7
(Use the arrow keys to highlight your choice.)

    Repair Your Computer

    Safe Mode
    Safe Mode with Networking
    Safe Mode with Command Prompt

    Enable Boot Logging
    Enable low-resolution video (640x480)
    Last Known Good Configuration (advanced)
    Directory Services Restore Mode
    Debugging Mode
    Disable automatic restart on system failure
    Disable Driver Signature Enforcement

    Start Windows Normally

Description: Allows drivers containing improper signatures to be loaded.
```

Figure 2: Advanced boot options menu displayed after F8 key is pressed.

5. Application window overview

Figure 3 presents the main window of the application.

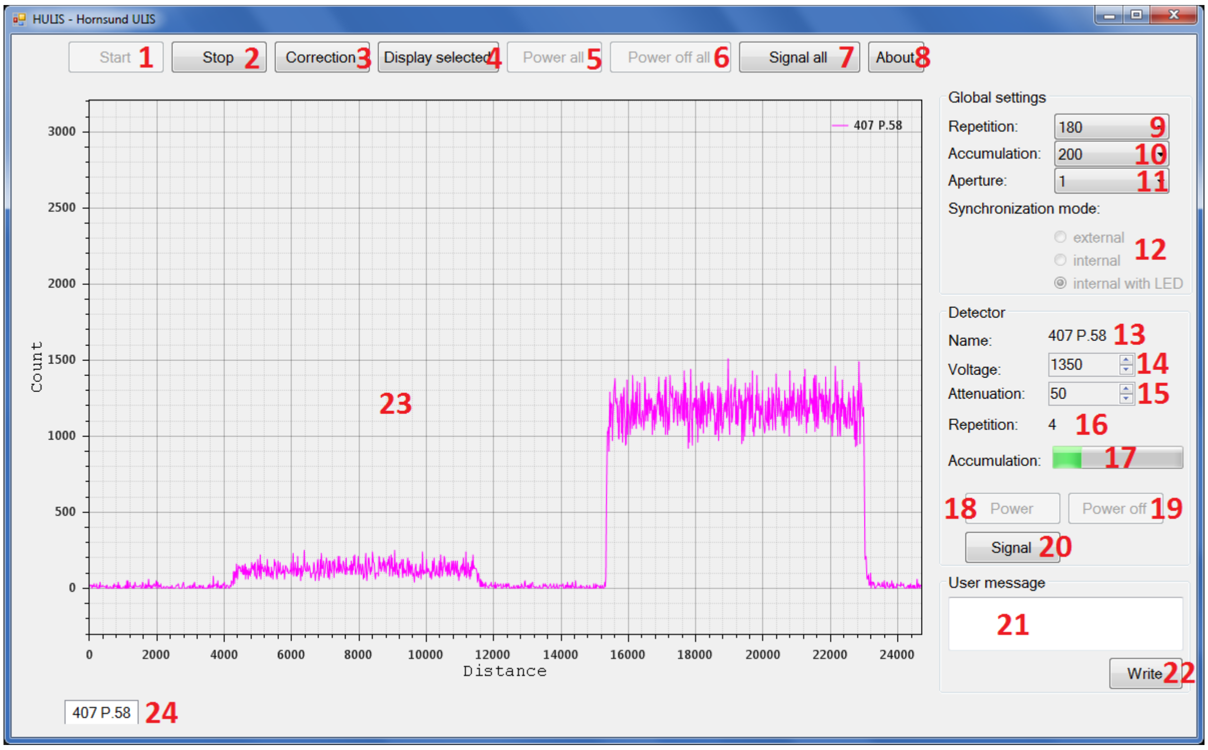


Figure 3: Main window of the application during work in the service mode.

1) [Start] Starts measurement. The button is active when there is at least one device connected and powered and when no measurement is in progress.
2) [Stop] Stops current measurement. The button is active when a measurement is in progress.
3) [Correction] The button is two-state. When the button is pressed the range corrected view is active.
4) [Display selected] The button is two-state. When the button is pressed only signal associated with a selected device is displayed in the plot area.
5) [Power all] Powers all connected and not yet powered devices. The button is active when there is at least one device connected and not powered.
6) [Power off all] Powers off all connected and powered devices. The button is active when there is at least one device connected and powered and when no measurement is in progress for that device.
7) [Signal all] The button is two-state. Measurement results can be presented in the plot area as averaged or so-called signal view. When the state of the button is changed to pressed, then view of every detector is set to the signal mode otherwise, is set to the averaged mode.
8) [About] When clicked shows info window.
9) [Repetition] Sets how many times single readings are averaged and saved to an output file.
10) [Accumulation] Sets how many single readings are averaged under one repetition loop.
11) [Aperture] The aperture of the lidar optical system cannot be changed from the application. The aperture is changed manually, however when you change the aperture in the application, then a message is written to a log file.

12) [Synchronisation mode] Detectors can work in two modes: the measurement mode and the service mode. In the measurement mode, a detector is triggered externally by laser shots. The service mode allows the user to check if a detector is working properly without starting a laser. In the service mode, a detector is triggered internally. Figure 3 presents signal generated by a detector in the service mode with a LED flashing inside a detector box.
13) [Name] The field displays name of a selected detector. The name can be specified in an ini file (see chapter 7 for more details).
14) [Voltage] Allows the user to change voltage applied to a photomultiplier. The greater voltage, the better sensitivity. Voltage change range can be specified in an ini file (see chapter 7 for more details).
15) [Attenuation] Some detectors have attenuation property. The greater attenuation value, the better sensitivity. The attenuation value can be changed from 0 to 100 in steps of 10.
16) [Repetition] The field displays current repetition number. Value 4 on the picture means that 3 loops already finished and forth is in progress.
17) [Accumulation] The progress bar shows how many readings were performed under one repetition loop.
18) [Power] Powers a selected detector. The button is active when a selected detector is not powered.
19) [Power off] Powers off a selected detector. The button is active when a selected detector is powered and not measuring.
20) [Signal] The button is two-state. Switches between the signal view and the averaged view for a selected detector.
21) [User message] In the area user can write a message regarding measurement process. A message will be written to a log file.
22) [Write] Writes a message to a log file and clears the user message area.
23) [Plot area] The place where plots from detectors are shown.
24) [Detectors bar] Every device connected through USB port detector is displayed in the bar.

6. Output measurement file description

Output file is created for each detector. Table 1 lists fields with description. Each repetition is saved using 8024 bytes.

Field name	Bytes number	Number format	Remarks
start year	2	unsigned integer	start vector with time when repetition started
start month	1	byte	
start day	1	byte	
start hour	1	byte	
start minute	1	byte	
start second	1	byte	
start millisecond	2	unsigned integer	
stop year	2	unsigned integer	stop vector with time when repetition finished
stop month	1	byte	
stop day	1	byte	
stop hour	1	byte	
stop minute	1	byte	
stop second	1	byte	
stop millisecond	2	unsigned integer	
signal	8000	single precision float point number	contains 2000 readings
voltage	2	signed integer	value -1 indicates that voltage was changed
attenuation	2	signed integer	value -1 indicates that attenuation was changed
aperture	2	signed integer	aperture multiplied by 100, value -1 indicates that aperture was changed
Total:	8024		

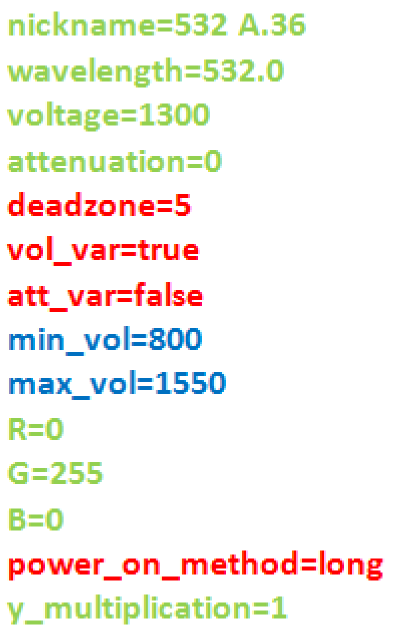
Table 1: Set of fields saved to an output file after one repetition loop is finished.

Use HulisRead Matlab function to translate a binary output file to Matlab variables.

7. Device information file description

Every detector has to have associated ini file to be recognised by the A-ULIS. Ini filename has to be a device serial number with ini extension, for instance, A.4.10.14.4.5.36.ini. Serial number should be found on a detector info label. Ini files should be located in the same directory where Hulis.exe is placed.

The device information file structure has been inherited from the ULIS software. Figure 3 shows sample ini file. Firsts 5 fields have been inherited from ULIS, next 9 were added to fulfil the A-ULIS needs. Fields specific for ULIS and not used by the A-ULIS were not listed. Green fields are optional, red are mandatory, blue are conditionally mandatory – see table 2 for more details.



```
nickname=532 A.36
wavelength=532.0
voltage=1300
attenuation=0
deadzone=5
vol_var=true
att_var=false
min_vol=800
max_vol=1550
R=0
G=255
B=0
power_on_method=long
y_multiplication=1
```

Figure 3: Sample device information ini file.

Field name	Mandatory	Format	Description
nickname	no	string	user-defined name of a detector; nickname is displayed in the detector panel and the plot legend; when not specified a serial number is used as a nickname
wavelength	no	float value	wavelength of light to which a detector is sensitive
voltage	no	integer number divisible by 50 and greater than 0	after a detector is powered, voltage will be set to that value; while applications closes that field is updated; if this field is not specified, value from the min_vol field is used
attenuation	no	integer number from 0 to 100 divisible by 10	after detector is powered on attenuation will be set to that value, while applications closes that field is updated, when not

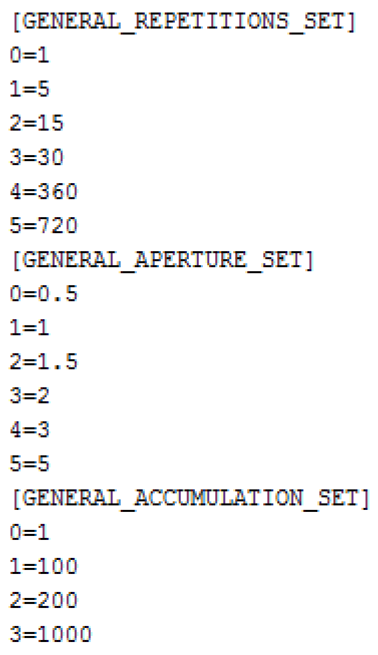
			specified - value 10 is used
deadzone	yes	integer number greater or equal to 0	as a response to an application request for data a device sends 2048 float values, deadzone specifies how many float numbers are skipped from the beginning of data sequence; deadzone value should be specified by the manufacturer of a detector
vol_var	yes	true or false	if true voltage can be changed for a detector, otherwise voltage is fixed
att_var	yes	true or false	if true attenuation can be changed for a detector, otherwise attenuation is fixed
min_vol	yes if vol_var is set to true, otherwise no	integer number divisible by 50 and greater than 0	minimum value of voltage that can be set through the detector panel
max_vol	yes if vol_var is set to true, otherwise no	integer number divisible by 50 and greater or equal to min_vol	maximum value of voltage that can be set through the detector panel
R	no	number from 0 to 255	describes red component of a line representing signal on the plot panel
G	no	number from 0 to 255	describes green component of a line representing signal on the plot panel
B	no	number from 0 to 255	describes blue component of a line representing signal on the plot panel
power_on_method	yes	long or short	can be set to long or short
y_multiplication	no	integer number greater than 0	output signal can be multiplied by this value to adjust signal to the plot scale better

Table 2: Possible fields in a device information file.

Attention:

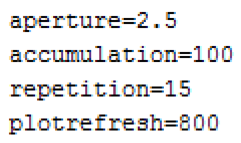
Please use min_vol, max_vol properties with special care. For most detectors used in the lidar laboratory voltage can be changed from 800 V to 1550 V. Let's assume that in ULIS voltage can be increased up to 1550 V. When you set max_vol in the application to 2000 then after level 1550 V is reached it will be still possible to send voltage up commands by the application. It is not known how detector is secured against such operations.

There are two optional ini files: gpset.ini and ulis.ini. Both were also inherited from the ULIS but significantly reduced. The gpset.ini file contains possible values of repetition, accumulation and aperture displayed in the general settings panel. The ulis.ini file contains recently used values of repetition, accumulation and aperture and those fields are updated while application closes. Using the plotrefresh field in the ulis.ini file it is also possible to specify in milliseconds how fast is the plot area refreshed. Both files can be edited manually. Figure 4 and 6 show sample gpset.ini and ulis.ini file.



```
[GENERAL_REPETITIONS_SET]
0=1
1=5
2=15
3=30
4=360
5=720
[GENERAL_APERTURE_SET]
0=0.5
1=1
2=1.5
3=2
4=3
5=5
[GENERAL_ACCUMULATION_SET]
0=1
1=100
2=200
3=1000
```

Figure 4: Sample gpset.ini file.



```
aperture=2.5
accumulation=100
repetition=15
plotrefresh=800
```

Figure 5: Sample ulis.ini file.

8. Development tools used

Tools:

- Microsoft Visual Studio Express 2013
- USBlyzer

External libraries:

- OxyPlot.dll, OxyPlot.WindowsForms.dll – plotting libraries
- CyUSB.dll – .NET interface to cyusb.sys driver

Other:

- Code templates delivered with CySuiteUSB

9. Ideas for future improvements

Idea	Description	Level of difficulty
WPF	Moving from Windows Forms to WPF should be easy because MVP pattern is used.	Easy
Linux version	.NET framework counterpart for Linux is called Mono. Current stable version 3.2.3 is compatible with .NET 4.5. Version 3.2.3 of Mono was released for OpenSuse 11.4 distribution. One way is to port .NET version of the application to Mono version. There is no cyusb.dll counterpart for Mono, for Linux between the application and the driver there are C++ libraries. Another way is to write a new application for Linux in C++.	Medium
3D plot	The 3D plot would allow the user to view how the signal was changing over time quickly. Plot with the third coordinate represented as colour should be easy to implement.	Easy
Laser control from application level	Through RS232 interface it should be possible to control laser from the application.	Medium
Remote viewer	Lidar measurements are usually long lasting. From time to time a lidar operator should check if everything is working correctly. A remote viewer which shows current plot would make measurement easier to a lidar operator. A viewer would receive data through LAN.	Medium
Control of everything	There are few tasks performed manually in the lidar laboratory: aperture change, laser beam adjustment, roof window opening. To make those tasks controlled by the application new devices must be installed.	Hard
Virtual detectors	Virtual detectors would make development and testing possible without connecting real devices through USB ports.	Easy

Table 3: Ideas for future improvements.