Awaiba API

Revision History:

Version	*Date*	*Modifications*	*Author*
1.1.0	14-03-13	Document creation	João Santos
1.2.0	03-09-13	Update Nan EyeGS and Frame Processing	João Santos
1.3.0	28-10-13	Updated NanEyeProviderDemo Project	João Santos
1.4.0	30-04-14	Overall Update	João Santos
1.5.0	13-05-14	Updated UDK3.0	João Santos
1.6.0	26-08-14	Added Idule Provider	João Santos

Table of Contents

Index of Tables

Index of Figures

1 Introduction

Awaiba API allows to communicate with Nan Eye and Nan Eye GS, allowing the user to receive images from those sensors.
Regarding the **NanEye** camera, it allows to use either the **EFM01** board or the **NanoUSB2**. It allows to run more than one Nan Eye simultaneously.

As for **Nan Eye GS**, it uses the **Trenz Module** and receive data from one sensor. Also allows to use the **Idule Module** to get data from the NanEyeGS.

The project **NanEyeProvider Demo** is now included in the API.
That project has 4 different forms:

- NanEye using EFM01 Board (the red Box);
- Nan Eye using NanoUSB2 Board (the blue or green Box);
- Nan Eye Stereo with EFM01 Board;
- Nan Eye Stereo with NanoUSB2 Board.

There is two different ways to use the **NanEye API:**

- You can use either the Error: Reference source not foundError: Reference source not found or the 4 NanEyeProviderDemo.

In the chapterError: Reference source not found you need to writer more functions and also do the image creation (from the raw data (or processed data) array to a bitmap, for example).

In the chapter 4 all of those functions are made, and you just need to use the functions as it is shown in the **NanEyeProviderDemo** project.

2 Awaiba API

In Figure 1, we have the file structure:

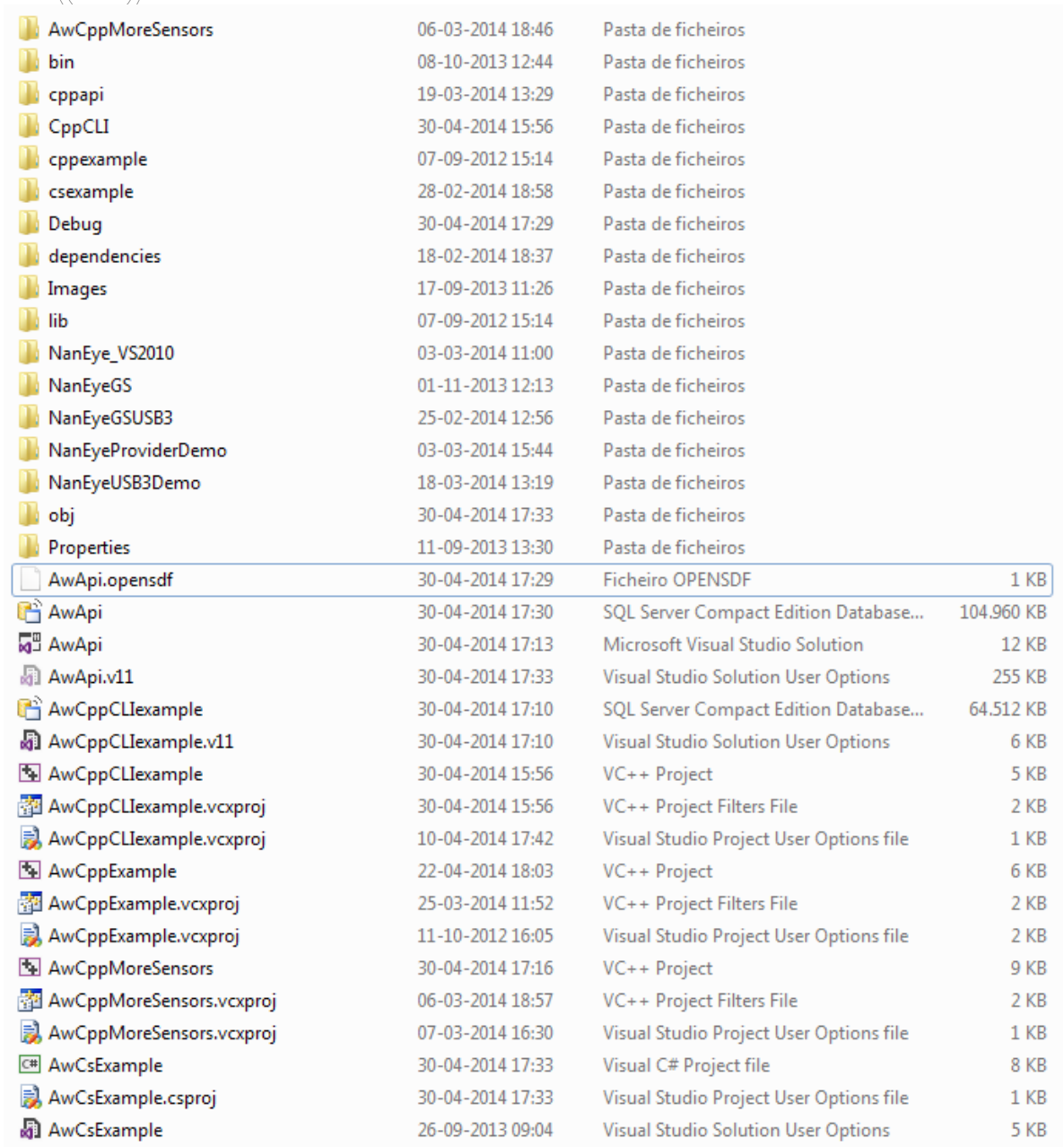

Figure 1: API File Structure

In there:

- **AwCppCLIexample** project has the files in **CppCLI** folder and in **cppapi** folder;
- **AwCppExample** project has the files in the **cppexample** folder and in **cppapi** folder;
- **AwCppMoreSensors** project has the files in **AwCppMoreSensors** folder and in **cppapi** folder;
- **NanEyeGS_CppCLI** project has the files in the **NanEyeGS_CppCLI** folder;
- **AwCsExample** project has the files in the **csexample** folder;
- **NanEyeGS** project, and its files in the **NanEyeGS** folder;
- **IduleProvider** project and its files in the **IduleProvider** folder;
- **NanEyeGSUSB3** project has its files in the **NanEyeGSUSB3** folder;
- **NanEyeProviderDemo** project has its files in the **NanEyeProvider** folder;
- **NanEyeUSB3** project has its files in the **NanEyeUSB3** folder;
- **NanEyeVS2010** is a project in Visual Studio 2010 that isn't part of the **AwAPI** solution.

All of this projects (except the NanEyeVS2010) are part of the **AwAPI** solution that was created with **Visual Studio 2012**.

Although, it can be used with older Visual Studio versions. For that the user needs to create the solution, and then import the project (from the ones described above).

The C++ AwCppCLIExample, AwCppExample and AwCppMoreSensors projects use the files from the **cppapi** folder and the **awcore.lib** file (**api/lib/x86 or x64**);

3 Cpp Projects

3.1 AwCppCLIExample

This project shows how to use the **AwFrameProcessing** inside a **C++/CLI project**. The main.cpp file is written in c++ language, and then it uses the .NET to do the connection with the Awaiba frame processing library.

3.1.1 Output folder and Files required

The output folder is **api/CppCLI/bin/x86/Release:**

awcore.dll → Does the connection between the main.cpp and the UDK3 files;
udk3-1.0-x86.dll → UDK3 file. Responsible for getting data from the board;
udk3mod-1.0-winusb-x86.dll → UDK3 file. Responsible for getting data from the board;
AwFrameProcessing.dll → Where all the awaiba frame processing functions are stored;
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.

3.2 AwCppExample

This project shows how to connect with the NanEye camera using the C++ language.

3.2.1 Output folder and Files required

The output folder is **api/cppexample/bin/x86/Release:**

awcore.dll → Does the connection between the main.cpp and the UDK3 files;
udk3-1.0-x86.dll → UDK3 file. Responsible for getting data from the board;
udk3mod-1.0-winusb-x86.dll → UDK3 file. Responsible for getting data from the board;

3.3 AwCppMoreSensors

This project allows to receive data from 2 or more sensors simultaneously. The sensors instantiation is done in the main method, one by one. Then, there are created as many threads as sensors, to receive the data simultaneously from every sensor.

3.3.1 Output folder and Files required

The output folder is **api/AwCppMoreSensors/bin/x86/Release:**

awcore.dll → Does the connection between the main.cpp and the UDK3 files;
udk3-1.0-x86.dll → UDK3 file. Responsible for getting data from the board;
udk3mod-1.0-winusb-x86.dll → UDK3 file. Responsible for getting data from the board;

3.4 NanEyeGS Cpp/CLI

In this project there are used several references that are shown in Figure 2.
This references load the .NET libraries in the c++ project.

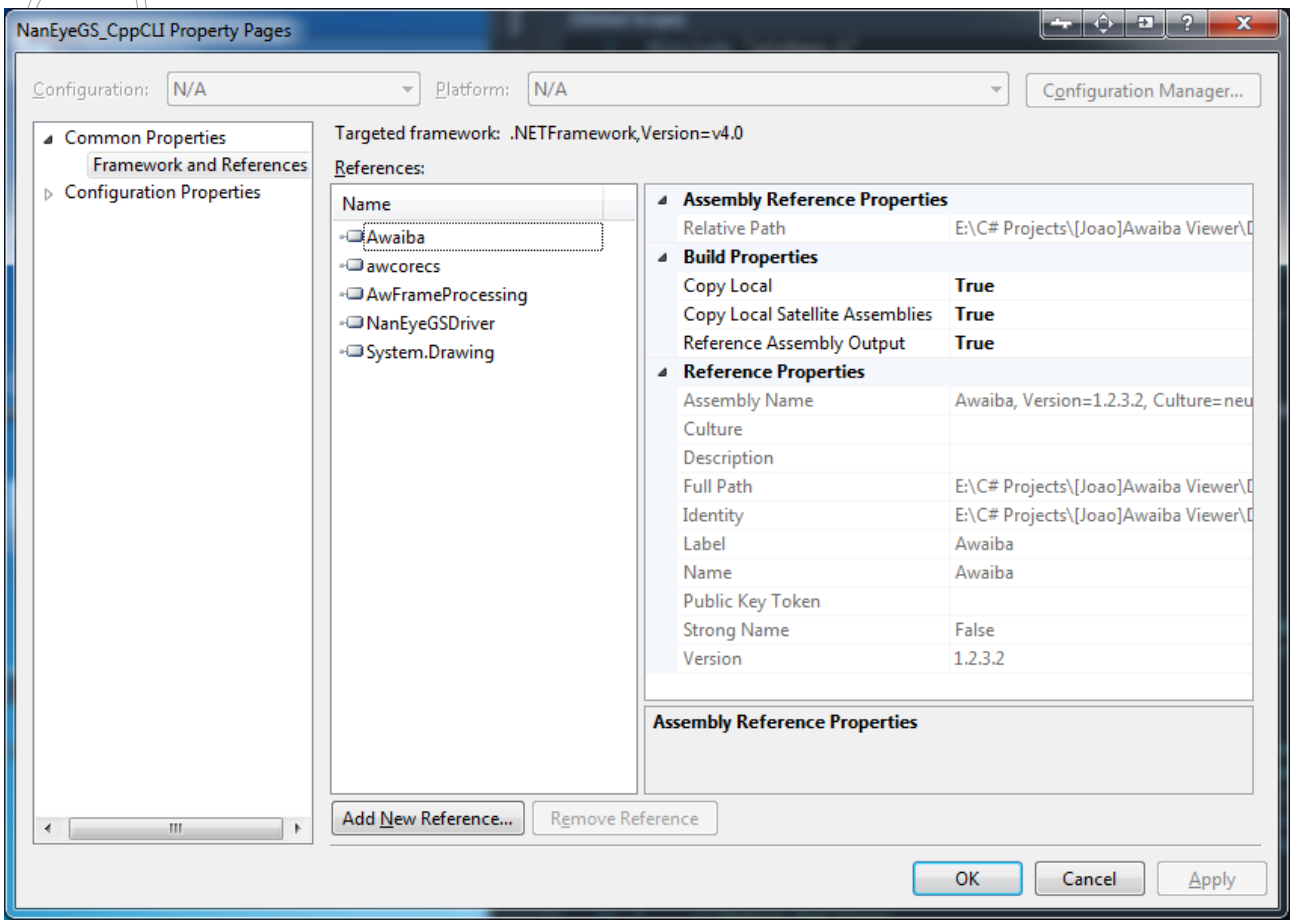

Figure 2: NanEyeGS_CppCLI references

Then there are used the following namepaces:

```
using namespace Awaiba;
using namespace Awaiba::Drivers::Grabbers;
using namespace Awaiba::FrameProcessing;
```

This allows to create the NanEyeGS instance, that allows to get data from the sensor. That data is obtained using an event. There is also an event to catch the eventual exceptions that are raised by the inner dll's:

```
NanEyeGsUsb2Extender^ gs = gcnew NanEyeGsUsb2Extender();
gs->ImageProcessed += gcnew EventHandler<OnImageReceivedBitmapEventArgs^>(receivedImage);
gs->Exception += gcnew EventHandler<OnExceptionEventArgs^>(exception);
```

When the **ImageProcessed** event is raised, all the processing has already been done. To change the image processing data, the user needs to use the **ProcessingWrapper::Pr** instance, as it is explained in **main.cpp** of the **NanEyeGS_CppCLI** project.

3.4.1 Output folder and Files required

The output folder is **api/NanEyeGS_CppCLI/bin/x86/Release:**

awaiba.dll → Awaiba general library;
awcorecs.dll → File that does the .NET connection with the awcore;
awFrameProcessing.dll → Where all the awaiba frame processing functions are stored.
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.
NanEyeGSDriver.dll → File that has the NanEyeGS interface;
CyUSB.dll → Used by the NanEyeGSDriver to communicate with the cypress devices;

4 NanEyeProviderDemo

To create an instance of any of the cameras, you use the following line:

Sensor	Board	Instance
NanEye 2C	EFM01 - Red Board	`provider = new NanEye2CEFM01Provider();`
NanEye 2D	EFM01 - Red Board	`provider = new NanEye2DEFM01Provider();`
NanEye Stereo	EFM01 - Red Board	`provider = new NanEyeEFM01StereoProvider();`
Nan Eye 2C	NanoUSB2 – Blue Board	`provider = new NanEye2CNanoUSB2Provider();`
Nan Eye 2D	NanoUSB2 – Blue Board	`provider = new NanEye2DNanoUSB2Provider();`
NanEye Stereo	NanoUSB2 – Blue Board	`provider = new NanEyeNanoUSB2StereoProvider();`

Table 1: Create instance

As shown above, you can create different instances according to the board and sensor you have.

The following pages will shown how to get images, how to send registers, etc.

To receive the image, we can use the **OnImageReceivedBitmapEventArgs**, and to handle the errors the **OnExceptionEventArgs**.

4.1 Output folder and Files required

The output folder is **api/NanEyeProvider/bin/x86/Release:**

awcore.dll → Does the connection between the main.cpp and the UDK3 files;
udk3-1.0-x86.dll → UDK3 file. Responsible for getting data from the board;
udk3mod-1.0-winusb-x86.dll → UDK3 file. Responsible for getting data from the board;
awaiba.dll → Awaiba general library;
awcorecs.dll → File that does the .NET connection with the awcore;
awFrameProcessing.dll → Where all the awaiba frame processing functions are stored.
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.
NanEyeGSDriver.dll → File that has the NanEyeGS interface;
CyUSB.dll → Used by the NanEyeGSDriver to communicate with the cypress devices;
CesysProvider.dll → File that has all the functions that called in the **AwCsExample** project (Manager, Source, etc).

The **NanEyeGSDriver** and the **CyUSB** files are only used in this project to do the register creation

5 NanEyeUSB3Demo

In figure 3 is shown the interface of the NanEyeUSB3 Demo.

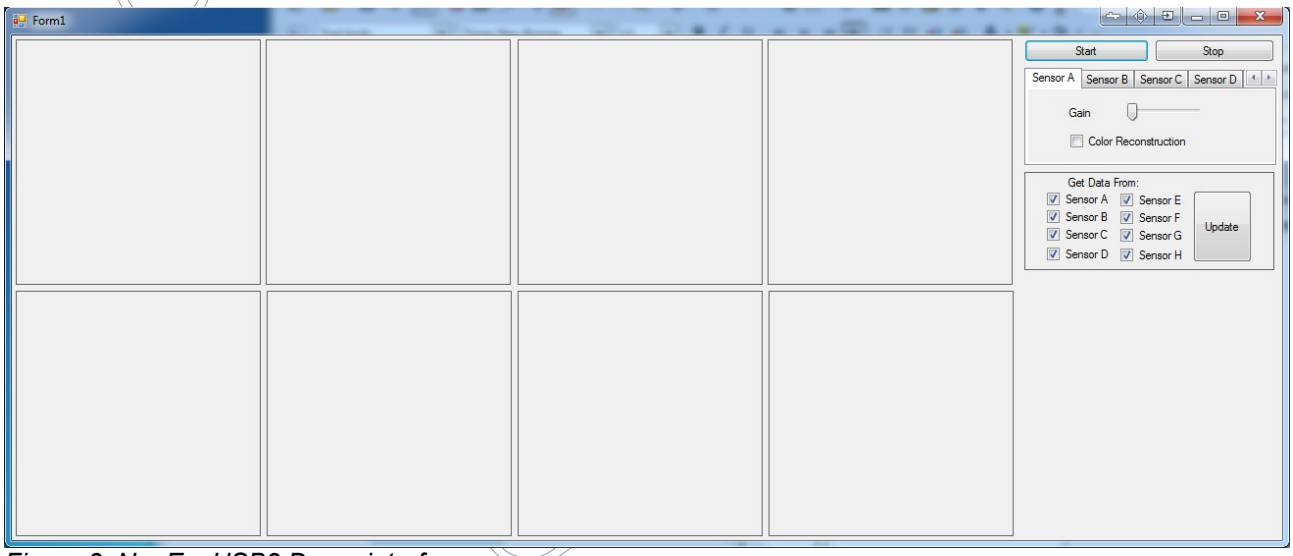

Figure 3: NanEyeUSB3 Demo interface

With v03 fpga code, the user is able to select from which sensor to receive data from, up to 8 sensors.

To do that, the variable **sensors** needs to be updated with a **List<bool>** of 8 positions. In each position, the true and false will mean that sensor will output data or not, as is explained in the **Form1.cs** of the **NanEyeUSB3Demo** project.

5.1 Constructor

```
NanEyeUSB3Provider provider = new NanEyeUSB3Provider();
```

5.2 Events

Receive the images → `provider.ImageProcessed += provider_ImageProcessed;`

Handle the errors → `provider.Exception += provider_Exception;`

5.3 Output folder and Files required

The output folder is **api/NanEyeUSB3/bin/x86/Release:**

awaiba.dll → Awaiba general library;
awcorecs.dll → File that does the .NET connection with the awcore;
awFrameProcessing.dll → Where all the awaiba frame processing functions are stored;
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.
NanEyeGSDriver.dll → File that has the NanEyeGS interface;
CyUSB.dll → Used by the NanEyeGSDriver to communicate with the cypress devices;
NanEyeUSB3Provider.dll → File that has the constructor for the NanEyeUSB3 instance.

6 Nan Eye GS and NanEye GS Stereo

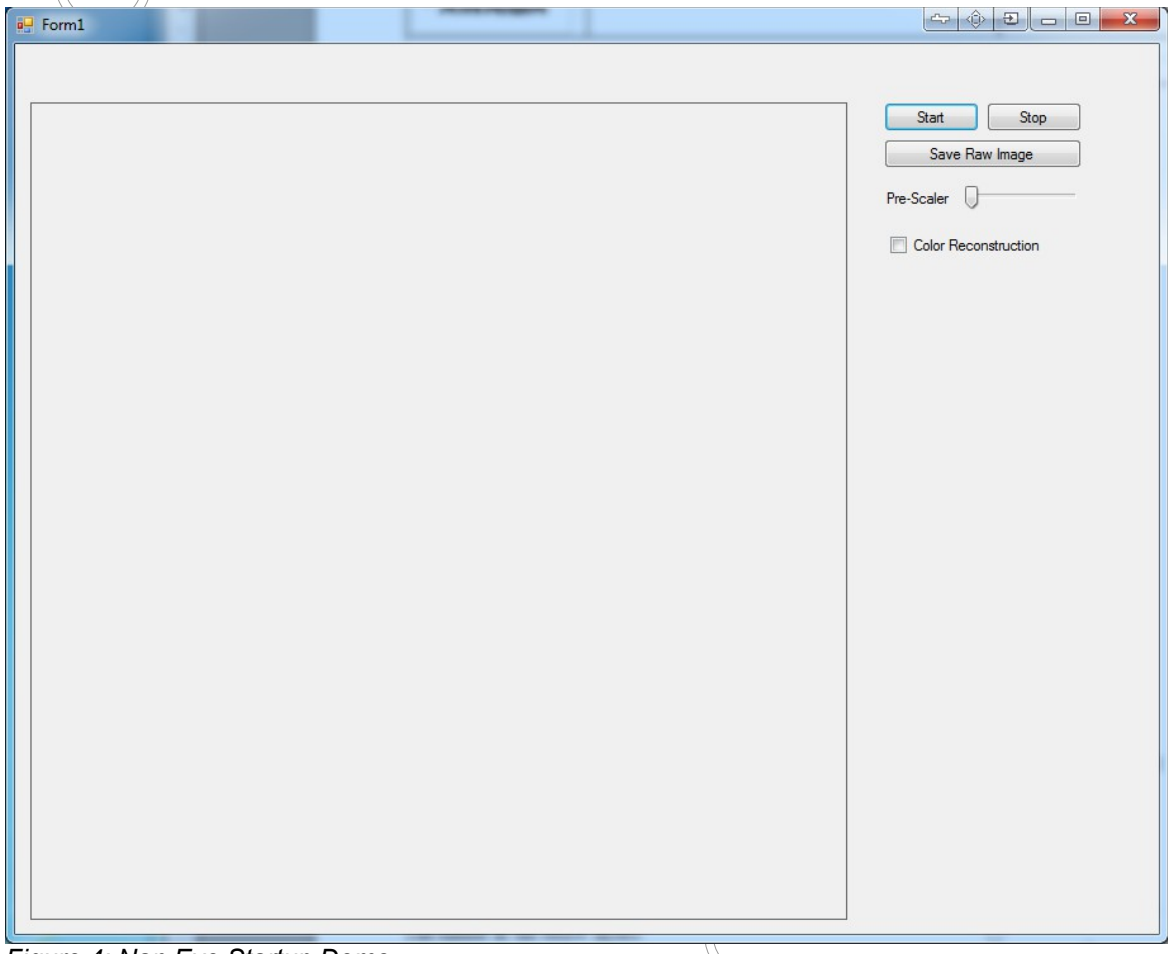

Figure 4: Nan Eye Startup Demo

This API allows to receive images from Nan Eye GS using the Trenz Module and also from NanEyeGS Stereo using the USB3.

NanEyeGS and NanEyeGSUSB3 projects have two examples of how to use the NanEyeGS with the trenz module and with the USB3 board.

In the **Form1.cs** file, there is an example of how to use all the API functions.

In its constructor, we have, for the **NanEyeGS** the creation of the instance:

```
NanEyeGsUsb2Extender camera = new NanEyeGsUsb2Extender();
```

As for **NanEyeGS stereo (USB3),** the instance is created like this:

```
NanEyeGsUsb3Extender camera = new NanEyeGsUsb3Extender();
```

Creation of two event handlers, to take care of the received frames and the exceptions that can handle in the below layers:

```
camera.ImageProcessed += camera_ImageProcessed;
camera.Exception += LiveImage_OnException;
```

6.1 Nan Eye GS Registers

To change the registers you need to use a `NanEyeRegisterPayload` instance. This allows you to choose directly the address and the value to change.

The register addresses and values are explained in the "**NanEyeGS Specification**" document.

6.2 API Functions

All the other API functions (StartCapture, StopCapture, for example) are explained in section 4 .

6.3 Output folder and Files required

The output of this project is located in **api/NanEyeGS/bin/x86/Release** folder;

awaiba.dll → Awaiba general library;
awcorecs.dll → File that does the .NET connection with the awcore;
awFrameProcessing.dll → Where all the awaiba frame processing functions are stored.
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.
NanEyeGSDriver.dll → File that has the NanEyeGS interface;
CyUSB.dll → Used by the NanEyeGSDriver to communicate with the cypress devices;

7 Idule Provider

This project allows to receive images from the Idule Module.
In terms of user interface, it is the same as in chapter 6, Nan Eye GS and NanEye GS Stereo.

The instance is created like this:

```
IduleProvidercs camera = new IduleProvidercs();
```

Creation of two event handlers, to take care of the received frames and the exceptions that can handle in the below layers:

```
camera.ImageProcessed += camera_ImageProcessed;
camera.Exception += LiveImage_OnException;
```

7.1 Nan Eye GS Registers

To change the registers you need to use a `NanEyeRegisterPayload` instance. This allows you to choose directly the address and the value to change.

The values that can be changed are:

- Pre-Scaler
- Exposure

The register addresses and values are explained in the "**NanEyeGS Specification**" document.

7.2 API Functions

All the other API functions (StartCapture, StopCapture, for example) are explained in section 4 .

7.3 Output folder and Files required

The output of this project is located in **api/NanEyeGS/bin/x86/Release** folder:

awaiba.dll → Awaiba general library;
awcorecs.dll → File that does the .NET connection with the awcore;
awFrameProcessing.dll → Where all the awaiba frame processing functions are stored.
opencv_core249.dll → Used in AwFrameProcessing.dll;
opencv_imgproc249.dll → Used in AwFrameProcessing.dll.

NanEyeGSDriver.dll → File that has the NanEyeGS interface;
CyUSB.dll → Used by the NanEyeGSDriver to communicate with the cypress devices;
IduleProvider.dll → Does the connection between the board and the PC;
IduleProvidercs.dll → .NET wrapper of IduleProvider.dll.

8 Registers

The registers are sent using a `NanEyeRegisterPayload` instance. It contains:

- `AllDevices (true if the register is to be sent to all devices simultaneously)`
- `Register address;`
- `IsSensorRegister;`
- `segment;`
- `SensorId;`
- `Value;`

8.1 Nan Eye Registers

In Nan Eye, this allow to change the Gain, Offset, Rows In Reset, Vrst_Pixel, Vref_Cds, Digipot:

Setting	Address	Value Range	Default
Gain	0x01	[0-3]	2
Offset	0x02	[0-3]	3
Exposure	0x03	[250-0]	249
Vrst_Pixel	0x07	[0-3]	1
Vref_Cds	0x08	[0-3]	2
Digipot	0x04	[1700-2400]	1800
Led_State	0x05	True-False	False
Led_Value	0x06	[0-4096]	0
DAC_DREG0	0x09	[1700-2400]	1750
DAC_DREG1	0x0A	[1700-2400]	1710
DAC_DREG2	0x0B	[1700-2400]	1700
DAC_DSTEP1	0x0C	[-100 - 100]	40
DAC_DSTEP2	0x0D	[-100 - 100]	40
DAC_DREGEN_REG	0x0E	[0x000 - 0x111]	0x111

Table 2: Nan Eye Registers

For example, to change the sensor's **offset** to **2**:

```
provider.WriteRegister(new NanEyeRegisterPayload(false, 0x02, true, 0, 0, 0x02));
```

8.2 Nan Eye GS Registers

The registers to be sent to Nan Eye GS are described in the document **NanEye_GS ASIC Specification Preliminary.**

As an example, below I'll change the **offset** to **2** and the **gain** to **5**:

```
provider.WriteRegister(new NanEyeRegisterPayload(false, 0x0B, true, 0, 0, (2 << 4) | (5)));
```

According to the specification, the 4 MSB of register **0x0B** are to defined the O**ffset** value and the 4 LSB of **0x0B** are to define the G**ain**.

Important: In case of the **NanEyeGS Stereo**, when a new value is sent either to the **pre-scaler** (address 0x05) or to the **integration** (0x06), the option **AllDevices** needs to be set to true:

```
camera.WriteRegister(new NanEyeRegisterPayload(true, 0x05, true, 0, trackBar1.Value));
```

9 Processing

The frame processing is handled by the **ProcessingWrapper** class. It can be accessed by its static class. In this class you have all the processing algorithms.
At the moment it can be used with Nan Eye and Nan Eye GS sensors, with any of the boards.

More information regarding this, check on Awaiba's website for the Awaiba Image Processing document.

10 Events and API Functions

10.1 OnImageReceivedBitmapEventArgs

NEED TO BE MOVED TO THE AWFRAMEPROCESSING DOCUMENT! (10.1)

To subscribe this Event Handler you need to create a function that has the following attributes:

```
myProvider_OnImageReceived(object sender, OnImageReceivedBitmapEventArgs e)
```

When subscribing the **ImageProcessed** event handler the **myProvider_OnImageReceived** will be triggered every time a new image is received.

The `OnImageReceivedBitmapEventArgs` has:

- **FrameCount**: The frame id since the sensor started sending data;
- **Width**: The width of the frame;
- **Height**: The hight of the frame;
- **BitsPerPixel**: The number of bits per pixel;
- **TimeStamp**: The timestamp in **miliseconds**, since the sensor started sending data;
- **Bitmap:** The processed frame (with all the image processing including correction masks);
- **SenId:** The identification of which sensor sent the frame.
- **ProcessingTime:** The time (in miliseconds) frame took to be processed (by the frame processing algorithms);
- **ImageData**: Where the raw and processed data from the image is located. More information on this can be checked on Awaiba Image Processing document.

10.2 OnExceptionEventArgs

```
LiveImage_OnException(object sender, OnExceptionEventArgs e)
```

When subscribing this event handler, the user is able to treat the exceptions that occur in the lower level.

The `OnExceptionEventArgs` has:

- **Exception**: The exception that caused the event to be triggered.

10.3 IsCapturing

Allows to check if the camera is capturing:

```
bool capturing = provider.IsCapturing;
```

10.4 IsConnected

Allows to check if the board is connected:

```
bool connected = provider.IsConnected;
```

Note: This feature isn't available for the EFM01 and NanoUSB board. This will always **return true** for this boards.

10.5 StartCapture

When this method is called, the board will start sending images (i.e. the **ImageProcessed** event will be triggered):

```
provider.StartCapture();
```

10.6 StopCapture

When this method is called, the board will stop sending images:

```
provider.StopCapture();
```

10.7 ReadRegister

This method isn't implement.

10.8 WriteRegister

When this method is called it will send to the register to the sensor:
In this case, we are changing the Gain value to 2:

```
provider.WriteRegister(new NanEyeRegisterPayload(false, 0x01, true, 0, 2));
```

In the chapter 8 (Registers) is explained how to use the above method.

11 CesysProvider

11.1 Location and Paths - BinFile

If there is a need to change the BinFile, it should be done using this variables, and before the camera initialization.

The file is chosen by a combination of two variables **FpgaFilesDirectory** (*Awaiba.Drivers.Grabbers.Location.Paths.FpgaFilesDirectory*) and **BinFile** (*Awaiba.Drivers.Grabbers.Location.Paths.BinFile*).

It is used as:

```
Location.Paths.FpgaFilesDirectory + Path.DirectorySeparatorChar + BinFile;
```

Location.Paths.FpgaFilesDirectory → %AppData%/Awaiba/Awaiba Viewer/Fpga Files
Location.Paths.BinFile → Values in table 3:

NanEye2CEFM01Provider	efm01_awaiba_viewer_top_2C_2D.bin
NanEye2CEFM01Provider	efm01_awaiba_viewer_top_2C_2D.bin
NanEye2CNanoUSB2Provider	nanousb2_fpga_v06.bin
NanEye2DNanoUSB2Provider	nanousb2_fpga_v06.bin
NanEyeEFM01StereoProvider	efm01_awaiba_viewer_top_2C_2D.bin
NanEyeNanoUSB2StereoProvider	nanousb2_fpga_v06.bin

Table 3: Bin files