First Light SDK
First Light SDK User Manual

Revision history

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Para</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>21/11/2019</td>
<td>JTU</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
Table of contents

1. Introduction ......................................................................................................................... 3
2. Installation .......................................................................................................................... 3
3. How to use the SDK............................................................................................................. 4
   3.1. Setting the environment ................................................................................................. 4
   3.2. Example project ............................................................................................................ 4
   3.3. C++ API ....................................................................................................................... 7
   3.4. C API ............................................................................................................................. 10
   3.5. Python (for Windows and Linux) ................................................................................... 11
   3.6. LabVIEW (only for Windows) ..................................................................................... 11
4. Plugins ................................................................................................................................ 12
   4.1. Micro Manager (for Windows and Linux) ................................................................... 12
1. Introduction

First Light SDK is a Software Development Kit developed by First Light Imaging allowing users to control First Light Imaging Cameras from their own GUI.

The SDK is provided with some development API languages (C/C++/Python/LabVIEW) and some plugins to interface with other software (Micro Manager and more to come).

Currently, the SDK can be used with C-RED 2 and C-RED 3 cameras on Windows 10 and Linux 16.04 LTS.

2. Installation

During the SDK installation, you will be able to choose which component you want to install:

![Installation page](image)

Fig. 1: Installation page

In the section 'Demo software' you can find the software demo executable and its C++ source file.
In the section 'First Light SDK', the C/C++ SDK, the LabVIEW library (with a demo) and the python library (with a demo also).
In the section 'plugins' you can find the MicroManager plug-in.
In the section 'redistributable' you can find the supported grabbers run-time redistributable.
3. How to use the SDK

3.1. Setting the environment

To set the project environment you can use the environment variable FLISDK_DIR set during the installation:

- FLISDK_DIR/lib/release for .dll and .lib
- FLISDK_DIR/includes for the .h

FLISDK_DIR is automatically set during the install process, it points to the SDK installation directory and it must never be changed.

3.2. Example project

A sample project is available in the installation directory, you can start using it installing QtCreator. QtCreator can be retrieved from https://www.qt.io/download.

- Copy the directory Demo_sources in the desktop and open “FliSdkDemo.pro”.

![Configure Project](image)

Fig. 2: Qt configuration page
• Configure your project and you will have the view above:

Fig. 3 : Qt project view

• Compile the project:

Fig. 4 : Qt compilation view
• Connect a camera to the computer and start the software:

Fig. 5: Demo software
3.3. C++ API

Here is an explanation on how to use the SDK in your own code. You can refer to the source file which is available here: FLISDK_DIR/Example/FliSdkExample.cpp.

- First you have to initialize the SDK:

```cpp
#include "FliSdk.h"
#include <vector>
#include <iostream>

using namespace std;

int main()
{
    FliSdk* fli = new FliSdk();

    // detection of all the grabbers connected to the computer
    cout << "Detection of grabbers..." << endl;
    vector<string> listOfGrabbers = fli->detectGrabbers();
    if(listOfGrabbers.size() == 0)
    {
        cout << "No grabber detected, exit." << endl;
        return -1;
    }

    cout << "Done." << endl;
    cout << "List of detected grabber(s):" << endl;
    for(auto g : listOfGrabbers)
    {
        cout << "- " << g << endl;
    }

    // detection of the cameras connected to the grabbers
    cout << "Detection of cameras..." << endl;
    vector<string> listOfCameras = fli->detectCameras();
    if(listOfCameras.size() == 0)
    {
        cout << "No camera detected, exit." << endl;
        return -1;
    }

    cout << "Done." << endl;
    cout << "List of detected camera(s):" << endl;
    for(auto c : listOfCameras)
    {
        cout << "- " << c << endl;
    }

    cout << "Setting camera " << listOfCameras[0] << endl;
```
When the SDK is initialized, you can send commands to the camera. It is possible to send commands to the camera thanks to a pointer provided by the SDK. When a command is not specific to a camera, you need to use the generic camera interface.

```cpp
double fps = 0;
  fli->camera() -> getFps(fps);
  cout << "Fps read: " << fps << endl;

cout << "Fps to set? " << endl;
cin >> fps;
  fli->camera() -> setFps(fps);

cout << "Fps read: " << fps << endl;
```

When a command is specific to a camera, you need to use the specific interface.

```cpp
if(_fli->credTwo())
  _fli->credTwo() -> setFanSpeed(100);
else if(_fli->credThree())
  _fli->credThree() -> enableAgc(true);
```

Then you can get images from the camera:

```cpp
fli->start();
  uint8_t* image8b = fli->getImage();
  fli->display8bImage(image8b);
  uint8_t* image16b = (uint8_t*)fli->getRawImage();
  fli->display16bImage(image16b);
  fli->stop();
```
The function `getImage()` without parameter will return the last image in the ring buffer with format RGB888 directly displayable. Some processing (colormap, rotation sharpening…) can be added to the image as in the following example.

```cpp
int rotation = 0;
int rotation = 0;
cout << "Image rotation for processed image, in degrees between 0 and 360: " << endl;
cin >> rotation;
cin >> rotation;
fl->imageProcessing()->setRotationAngle(rotation);
fl->imageProcessing()->setRotationAngle(rotation);
```

It is possible to get the processed 16 bits pixel images. To do so, use the `getImage16b()`, but this image cannot be directly displayed on screen.

If you want faster image acquisition, please use the `getRawImage()` API. With this API no processing nor memory copy are done.

It is possible to get images by inheriting from the interface `IRawImageReceivedObserver`:

```cpp
#include "FliSdk.h"

class Demo : public IRawImageReceivedObserver
{
    Demo();
    virtual void imageReceived(uint8_t* image) override;
    virtual double fpsTrigger() override;
private:
    FliSdk* _fli;
};

//------------------------------------------------------------
void Demo::Demo()
{
    _fli = new FliSdk();
    _fli = new FliSdk();
    //Do the init process...
    //Do the init process...
    //Add this class in observer
    //Add this class in observer
    _fli->addRawImageReceivedObserver(this);
    _fli->addRawImageReceivedObserver(this);
}

//------------------------------------------------------------
void Demo::imageReceived(uint8_t* image)
{
    //here process the image
    //here process the image
}

//------------------------------------------------------------
double Demo::fpsTrigger()
{
    //triggered at 100 fps
    return 100.0; //triggered at 100 fps
    return 100.0; //triggered at 100 fps
}
For each image received, the SDK parses the list of observers, and for each observer, it reads the fpsTrigger. This value is used to know at which speed “imageReceived” should be called for each observer. The special value ‘0’ means full speed.

To access pixel value, you must cast the "uint8_t*" to "int16_t*" or "uint16_t*" depending on the pixel format indicated on the camera user manual.

You can also include "FliSdk_utils.h", it defines pixel type for the cameras.

- Finally, you can change the camera used or the running mode:

```c
if(listOfCameras > 1)
{
    _fli->setCamera(listOfCameras[1]);
    _fli->update();
}

_fli->setMode(FliSdk::Mode::GrabOnly);
// In GrabOnly mode you have to set a grabber
_fli->setGrabber(listOfGrabbers[0]);
_fli->update();
```

**Note:** You can find the full description of the API as Doxygen format in FLISDK_DIR/Doc folder.

### 3.4. C API

In addition to the C++, a C API is available. The C API is mainly a collection of wrapper functions on the C++ API. More description can be found in the C++ API documentation.

Here is an example of code:

```c
#include "FliSdk_C.h"

/*...*/
FliSdk_init();

uint8_t nbGrabbers = 0;
const char** listOfGrabbers = FliSdk_detectGrabbers(&nbGrabbers);

uint8_t nbCameras = 0;
const char** listOfCameras = FliSdk_detectCameras(&nbCameras);

if(nbCameras > 0)
{
    FliSdk_setCamera(listOfCameras[0]);
    FliSdk_update();
    FliSdk_start();
}

if(FliSdk_isStarted())
```
{  
    CameraModel model = FliSdk_getCameraModel();  

    char sensibility[100];  
    if(model == CameraModel::Cred2)  
        Cred2_getSensibility(sensibility);  
    else if(model == CameraModel::Cred3)  
        Cred3_getSensibility(sensibility);  

    uint8_t* image = FliSdk_getProcessedImage(-1);  // -1 to get the last image
}

3.5. Python (for Windows and Linux)

In the install directory of the SDK, you can find Python/lib/FliSdk.py that you can import in your python project. A code example is available in Python/demo.

To run it, you need to install the following dependencies:

- PyQt5: pip install PyQt5
- Pillow: pip install Pillow

Then, you can start the demo with the command "python FLiSdkDemo.py".

3.6. LabVIEW (only for Windows)

In the install directory of the SDK you can find LabVIEW/lib/FliSdk.lvlib. You can open with LabVIEW and use the provided functions to build a project. An example project is available in Labview/Demo/FliSdkDemo_Labview.vi.

Run it, click on “set camera”, then “start” and the video stream will be displayed.
4. Plugins

4.1. Micro Manager (for Windows and Linux)

A dynamic library is available to load the SDK and use the camera with MicroManager. To do so, copy MicroManager/mmgr_dal_FirstLightImaging.dll in the install directory of MicroManager. Start MicroManager and go in “Devices” -> “Hardware Configuration Wizard”:
Click “Next”, then in the list in the bottom search for “FirstLightImaging” and double-click on “FliSdk: First Light Imaging camera”.

Fig. 8: MicroManager configuration wizard
Click “Ok” in the pop up window and “Next” until the end of the wizard. Click on ”Snap” or ”Live” button to get images from camera.

![MicroManager with image grabbed](image)

**Fig. 9:** MicroManager with image grabbed

You have access to the camera parameters by clicking on “Devices” -> “Device Property Browser”.

![MicroManager camera config](image)

**Fig. 10:** MicroManager camera config

The line “FliSdk-Send command” allows the user to send any command to the camera.