OWLIFT SDK
Developer’s Guide

Rev 1.9.3

2023-06-01

Notice for this document

- Infinitegra, Inc. (hereinafter referred to as "Infinitegra") has a right of all information in this document. Infinitegra allows nobody to reproduce or redistribute any information in this document without gaining Infinitegra's approval.
- Infinitegra may change the information in this document without further notice.
- Contact Infinitegra if you find wrong information or any unclear points in this document.
- This document is an English translated version from a Japanese (original) version for a reference purpose. If there are any discrepancy between the English version and the Japanese version, the Japanese version should prevail.

Notice for the product / Disclaimer

- Do not operate the product against the contents of this document. It may be cause of troubles. Infinitegra cannot accept any responsibility for troubles that are caused by an illegal operation.
- Do not use for the usage that requires safety and reliability and has a possibility to affect one's life.
- Infinitegra doesn't take responsibility for any loss or damage incurred by this application.
- Infinitegra allows nobody to modify, dis-assemble, change, alter, reverse-assemble, reverse-compile, reverse-engineer or undertake any similar action to software, firmware or hardware of the product.

Contents

1 Outline

This document describes the softweare development kit of the compact thermal camera OWLIFT.

1.1 Contents of SDK

The OWLIFT SDK includes the following contents.

- Development kit for Windows
  - C library for Windows
  - .NET library for Windows
  - Python library for Windows
  - Sample source codes
  - Command line tools
    - OWLIFTConfig: The tool that accesses to the registers of the OWLIFT.
    - OWLIFTDump: The tool that gets thermal data from the OWLIFT.

- Development kit for Linux
  - C library for Linux
  - Python library for Linux
  - Sample source codes
  - Command line tools
    - OWLIFTConfig: The tool that accesses to the registers of the OWLIFT.
    - OWLIFTDump: The tool that gets thermal data from the OWLIFT.

- Development kit for Android
  - C library for Android
  - Java library for Android
  - Sample source codes

1.2 System Requirements

1.2.1 Runtime environments

〔C/.Net library for Windows〕

- Windows 10 64bit

〔C Library for Linux〕

- Linux Kernel 2.6.32 (or higher) x86 / x86_64, glibc 2.11 (or higher), Vidoe4Linux2 is required
- Linux Kernel 3.2.0 (or higher) 32bit / 64bit, glibc 2.13 (or higher), Vidoe4Linux2 is required

〔Java library for Android〕

- Android 7.0 - 11

〔Python library〕

Python 3.4 - 3.10

- Windows 10 64bit
- Linux Kernel 4.4.0 (or higher) x86 / x86_64, glibc 2.19 (or higher), Vidoe4Linux2 is required
- Linux Kernel 4.9.0 (or higher) 32bit / 64bit, glibc 2.19 (or higher), Vidoe4Linux2 is required

1.2.2 Build environments

〔C library for Windows〕

- Compiler:　Visual Studio 2017
- Supports 32/64bit builds

〔C library for Linux〕

- Compiler:　g++-4.6
- Supports x86 build, x64 build, armhf (Raspbian) build

〔Java library for Android〕

- Android Studio

1.3 Terminologies

These are terminologies used in this document.

- %OWLIFT_SDK_DIR%

  It represents the absolute path that OWLIFT SDK has been installed.

2 C library for Windows

2.1 Install

Unzip the OWLIFT-SDK-#.#.#.zip into any folder. “#.#.#” shows the version number of the SDK.

2.2 Uninstall

Remove the unzipped folder (%OWLIFT_SDK_DIR%).

2.3 Compile

Set the following configuration to the compiler in order to use the C library for Windows.

1. Add %OWLIFT_SDK_DIR%\include to the include path.
2. Add the following path to the library path. The library file (owlift.lib) will be linked automatically.
   - ➢ The directory for x86 … %OWLIFT_SDK_DIR%\lib\x86
   - ➢ The directory for x64 … %OWLIFT_SDK_DIR%\lib\x64

2.4 Mandatory file to run

You need the following file in order to run programs that use the C library for Windows. So you have to add the parent directory of the file or copy it to the working directory.

- ➢ The binary file for x86 … %OWLIFT_SDK_DIR%\bin\x86\owlift.dll
- ➢ The binary file for x64 … %OWLIFT_SDK_DIR%\bin\x64\owlift.dll

2.5 Include file

Include the following file.

- ・ OWLIFTLib.h

2.6 Reference manual

Refer to “OWLIFT C Library for Windows” in the following URL.
https://infinitegra.co.jp/static/owlift/sdk

2.7 Flow to display

These steps show a flow to display the thermal image. Refer to the reference manual for each function’s detail. Refer also to the sample codes for programing your system.

1. Get a device handle by OwLib_GetDevices().
2. Set a method to get images by OwLib_CaptureSetup().
3. Start getting images by OwLib_CaptureStart().
4. Every time frame data is input, the callback function that has been registered by OwLib_CaptureSetup() will be called.
5. In the callback function, decode the frame data by OwLib_Decode().

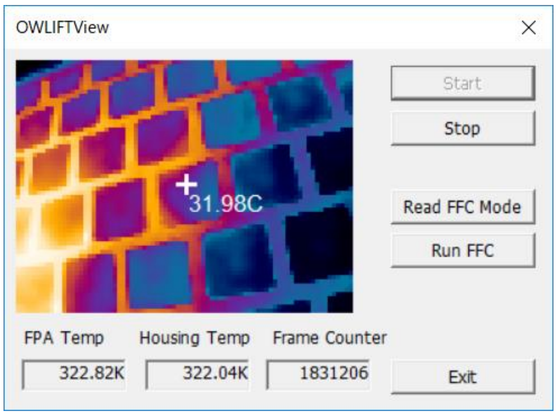
6. Get thermal data by OwLib_GetTempTable(), if you need.
7. Finish decoding the frame data by OwLib_FinishDecode(). You have to do it after the step 6.
8. Display the output in the step 5. Also display the output in the step 5, if you need.
9. OwLib_CaptureStop() terminates getting images. OwLib_Release() releases the device handle.

2.8 Sample souce codes

2.8.1 OWLIFTView

The OWLIFTView is a sample source code for Visual C++. The SDK includes the executable the OWLIFTView that has been built. The OWLIFTView equips the following features.

- Getting frame data and displaying it
- Getting temperature table
- Controlling the Sensor
- Getting and displaying telemetry data included in frame data



FPA Temp	Housing Temp	Frame Counter
322.82K	322.04K	1831206

owlift.dll has to exist on the execution path when you run the sample program with Visual Studio. The following steps show an example that changes a project setting in order to make a path to it.

1. Open [Debug] in a property of the OWLIFTView project
2. Change [Working Directory] to “$(ProjectDir)\..\..\bin\x86”

3 .NET library for Windows

3.1 Install

Same as “2.1 Install”.

3.2 Uninstal

Same as “2.2 Uninstall”.

3.3 Compile

Set the following configuration to the compiler in order to use the .NET library for Windows.

Add the following DLLs to the reference setting.

- The binary file for x86 … %OWLIFT_SDK_DIR%\bin\x86\owliftnet.dll
- The binary file for x64 … %OWLIFT_SDK_DIR%\bin\x64\owliftnet.dll

3.4 Mandatory file to run

You need the following files in order to run programs that use the .NET library for Windows. So you have to add the parent directory of the files or copy them to the working directory.

- The binary files for x86 … %OWLIFT_SDK_DIR%\bin\x86\owlift.dll
  %OWLIFT_SDK_DIR%\bin\x86\owliftnet.dll
- The binary files for x64 … %OWLIFT_SDK_DIR%\bin\x64\owlift.dll
  %OWLIFT_SDK_DIR%\bin\x64\owliftnet.dll

3.5 Reference manual

Refer to “OWLIFT .NET Library for Windows” in the following URL.
https://infinitegra.co.jp/static/owlift/sdk

3.6 Flow to display

These steps show a procedure to display the thermal image. Refer to the reference manual for each function’s detail. Refer also to the sample codes for programing your system.

1. Get a device handle by OwDev.GetDevices().
2. Set a method to get images by OwDev.CaptureSetup().
3. Start getting images by OwDev.CaptureStart().
4. Every time frame data is input, the callback function that has been registered by OwDev.CaptureSetup() will be called.
5. In the callback function, decode the frame data by OwDev.Decode().
6. Get thermal data by OwDev.GetTempTable(), if you need.
7. Finish decoding the frame data by OwDev.FinishDecode(). You have to do it after the step 6.

8. Display the output in the step 5. Also display the output in the step 5, if you need.
9. OwDev.CaptureStop() terminates getting images. Set a NULL to a variable that stores the OwDev pbject in order to release the device handle.

3.7 Sample source codes

The followings are sample source codes written in C#.

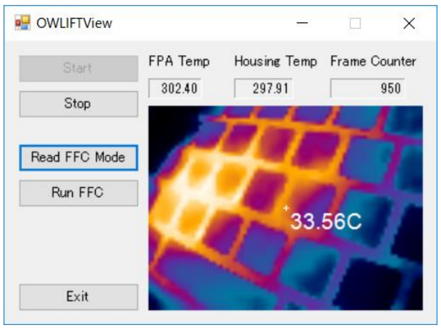
[Notice]

owlift.dll and owliftnet.dll have to exist on the execution path when you run the sample program with Visual Studio. The following steps show an example that changes a project setting in order to make a path to it.

1. Open [Debug] in a property of the OWLIFTView project
2. Select %OWLIFT_SDK_DIR%\bin\x86 in [Working Directory]
   (%OWLIFT_SDK_DIR% means a folder that the OWLIFT SDK is installed in)

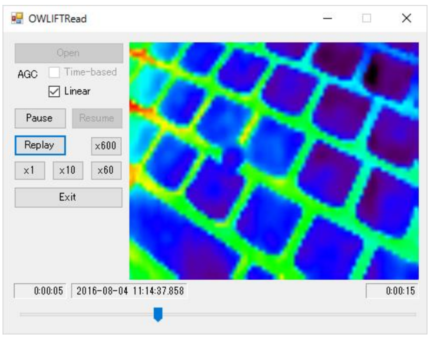
3.7.1 OWLIFTViewCS

OWLIFTViewCS is written in C#. Its feature is equivalent to the OWLIFTView written in Visual C++.



3.7.2 OWLIFTReadCS

OWLIFTReadCS is written in C# and has the function to play a raw-recording file.



4 C library for Linux

4.1 Install

Copy the following files into any folder. “#.#.#” shows the version number of the SDK.

- For x86: libowlift-dev_#.#.#-1_i386.deb
- For x64: libowlift-dev_#.#.#-1_amd64.deb
- For armhf (Raspbian): libowlift-dev_#.#.#-1_armhf.deb

Switch to the root user. Change the current directory to the directory that has .deb file and run the following command.

```
For x86
# dpkg -i libowlift-dev_#.#.#-1_i386.deb
For x64
# dpkg -i libowlift-dev_#.#.#-1_amd64.deb
For armhf
# dpkg -i libowlift-dev_#.#.#-1_armhf.deb
```

4.2 Uninstall

Switch to the root user and run the following command.

```
# dpkg --purge libowlift-dev
```

4.3 Compile

After the C library for Linux is installed, the include files will be put in /usr/include and the common library files will be put in /usr/lib. Add the following setting in order to compile your program.

- Add “-lowlift -lpthread” to your linker option

4.4 Mandatory file to run

You need the following file in order to run programs that use the C library for Linux.

- /usr/lib/libowlift.so.#
  (“#” means a version number of the common library. The version number is different from a version number of the SDK.)

You don’t need settings for running your programs if you installed the C library by the method mentioned by this document. Set an appropriate path by using environment variable LD_LIBRARY_PATH if you can’t find the shared library in a typical directory.

4.5 Include file

Include the following file.

- OWLIFTLib.h

4.6 Reference manual

Refer to “OWLIFT C Library for Linux” in the following URL.

https://infinitegra.co.jp/static/owlift/sdk

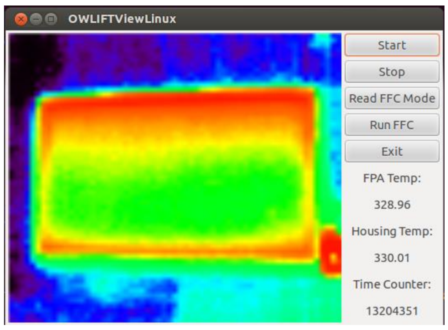
4.7 Flow to display

This is the same as the procedure for Windows mentioned at the section 2.7. Refer to it.

4.8 Sample souce codes

4.8.1 OWLIFTViewLinux

The OWLIFTViewLinux is a sample code created by GTK+ for Linux. It has similar features to the Windows version "OWLIFTView”. The executable file of the OWLIFTViewLinux is also installed when the C library for Linux is installed.



The binary packages are following.

• x86	: owliftviewlinux_#.#.#-1_i386.deb
• x64	: owliftviewlinux_#.#.#-1_amd64.deb
• Raspbian	: owliftviewlinux_#.#.#-1_armhf.deb

(#.#.# represents a version number)

The source code is included in owliftviewlinux.tar.gz of the SDK. You can compile it by unzipping the archive and running the following command. The GTK+ 2.0 library development file (libgtk2.0-dev) and the SDL 1.2 library development file (libsdl1.2-dev) are needed.

```
$ ./configure
$ make
```

5 Java library for Android

5.1 Install

Install the C library and the JAR library into an appropriate directory suitable for to your development environment.

5.2 Library deployment

Put the libraries to the following folders in order to use sample source codes with Android Studio.

- The JAR files: Make a folder “libs” under your project folder and put them in “libs”.
- The SO files: Make a project folder “/app/src/main/jniLibs” and put directories and libraries for each platform.

5.3 Compile

Put the sample source code in some folder and run Android Studio. Choose “Open an existing Android Studio project” and select the sample source codes that you have put. Configure JDK and Android SDK in the Android Studio setting if you need.

5.4 Reference manual

Refer to “OWLIFT Java Library for Android” in the following URL.
https://infinitegra.co.jp/static/owlift/sdk

5.5 Flow to display

These steps show a procedure to display the thermal image. Refer to the reference manual for each function’s detail. Refer also to the sample codes for programing your system.

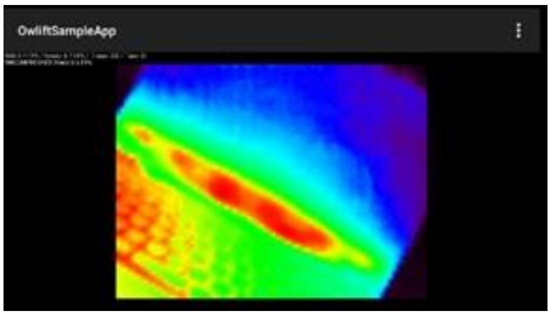
1. Get UsbOwliftHost by UsbOwliftHost.getUsbHost().
2. Open a device by UsbOwliftHost.openDev().
3. Get UvcOwliftFunc by UsbOwliftDev.getFuncs().
4. Start getting images by UvcOwliftFunc.startStream().
5. Every time frame data is inputted, the callback function that has been registered by UvcOwliftFunc.startStream() will be called.
6. Terminate getting images by UvcOwliftFunc.stopStream() or UsbOwliftHost.close().

5.6 Sample souce codes

5.6.1 OwliftSampleApp

The OwliftSampleApp is a sample source code for Android. It equips the following basic features.

- Getting and displaying frame data
- Recording and still image capture
- Getting and displaying a maximum and minimum temperature



6 Python library

6.1 Install

Execute pip command with the whl file that matches the platform of your environment.

(#.#.#.# represents a version number)

```
Windows 32bit:
# pip install owlift-#.#.#.#-py3-none-win32.whl
Windows 64bit:
# pip install owlift-#.#.#.#-py3-none-win_amd64.whl
Linux x86:
# pip install owlift-#.#.#.#-py3-none-linux_i686.whl
Linux x64:
# pip install owlift-#.#.#.#-py3-none-linux_amd64.whl
armv7l(Raspberry Pi):
# pip install owlift-#.#.#.#-py3-none-linux_armv7l.whl
```

6.2 Uninstall

Execute pip command to uninstall.

```
# pip uninstall owlift
```

6.3 Reference manual

Refer to "OWLIFT Python Library" in the following URL.

https://infinitegra.co.jp/static/owlift/sdk

6.4 Sample source codes

See README_ en.txt in the archive file of Python library's sample source codes.

7 Command line tools

7.1 OWLIFTConfig

The OWLIFTConfig is a command line interface program that accesses registers of the sensor. It has the following features.

- Execute command types GET, SET, RUN to the sensor
- Batch processing
- Change configurations of the device

Run the following command in order to refer to command option details. You will find command options that are not mentioned here.

```
> OWLIFTConfig --help
```

7.1.1 GET、SET、RUN

e.g.) Executing the 2 words (32bits) GET command to Module ID=0x0200 and Command ID Base=0x0C.

```
> OWLIFTConfig --get=0200,0C,2
```

e.g.) Executing the SET command with Module ID=0x0200 and Command ID Base=0x24.

```
> OWLIFTConfig --set=0200,24,5,0
```

e.g.) Executing the RUN command with Module ID=0x0200 and Command ID Base=0x40.

```
> OWLIFTConfig --run=0200,40
```

Note: Refer to the reference documents - 2 about specification of the sensor. We can’t guarantee system behavior brought from executing the command of the sensor. Execute it by your own responsibility.

7.1.2 Batch processing

The option “--file” executes sequencially a series of command options described in a file.

e.g.) Executing command options that is described in the file “config.txt”.

```
> OWLIFTConfig -file config.txt
```

The contents of config.txt

```
--get=0200,0C,2
--wait=100
--get=0200,0C,2
```

7.1.3 Device configuration

The sensor is being supplied a power while it connects to a USB. So the following command turns off a power supply while it doesn’t get image. This feature reduces power consumption.

```
> OWLIFTConfig --disable-always-on
```

7.2 OWLIFTDump

The OWLIFTDump is a command line interface program that gets temperature data, raw data and so on. Its features are as follows.

- It outputs telemetry data and average temperature values (or the sensor output values) for each frame. The average temperature value is an average in the specified area.
- It outputs telemetry data and temperature values of all pixels (or the sensor output values) for each frame.

Run the following command in order to refer to command option details. You will find command options that are not mentioned here.

```
> OWLIFTDump —help
```

7.2.1 Output average temperature values

e.g.) Outputting an average temperature of a rectangular field (x1,y1)-(x2,y2)=(10,10)-(19,19).

```
> OWLIFTDump --region=10,10,19,19
       SYSTIME        TIME   TIME_FFC FCS   FPAT FPATFC   HOUT HOUTFC  TAVE1  TMIN1  TMAX1
27/11:18:22.566     937404     912370   0 327.40 327.12 328.08 327.64 295.15 294.58 295.73
27/11:18:22.681     937520     912370   0 327.40 327.12 328.08 327.64 295.13 294.58 295.73
27/11:18:22.797     937636     912370   0 327.40 327.12 328.08 327.64 295.11 294.58 295.64
27/11:18:22.913     937752     912370   0 327.40 327.12 328.08 327.64 295.13 294.68 295.64
```

The each above item means;

- SYSTEMTIME: The system time of OS
- TIME: “Time Counter” of the telemetry data
- TIME_FFC: “Time Counter at last FFC” of the telemetry data
- FCS: “Status Bits” of the telemetry data
- FPAT: “FPA Temp” of the telemetry data
- FPATFC: “FPA Temp at last FFC” of the telemetry data
- HOUT: “Housing Temp” of the telemetry data
- HOUTFC: “Housing Temp at last FFC” of the telemetry data
- TAVE1: The average temperature of the rectangular field
- TMIN1: The minimum temperature of the rectangular area
- TMAX1: The maximum temperature of the rectangular area

You can set up to a maximum of 4 “--region” options. When you add “--region” options, TAVE2/TMIN2/TMAX2, TAVE3/TMIN3/TMAX3 and TAVE4/TMIN4/TMAX4 are added in output columns sequentially.

7.2.2 Raw-recording

--write-owi executes raw-recording. Raw-recording files can be played by OWLIFTDump and OWLIFTCap.

e.g.) Records thermal.owi.

```
> OWLIFTDump --write-owi=thermal.owi
```

7.2.3 Play raw-recording file

--read-owi plays a raw-recording file.

e.g.) Plays thermal.owi.

```
> OWLIFTDump --read-owi=thermal.owi
```

8 Temperature output

This section shows specification and notice of OwLib_GetTempTable() and OwDev.GetTempTable() that are functions of outputting temperature values.

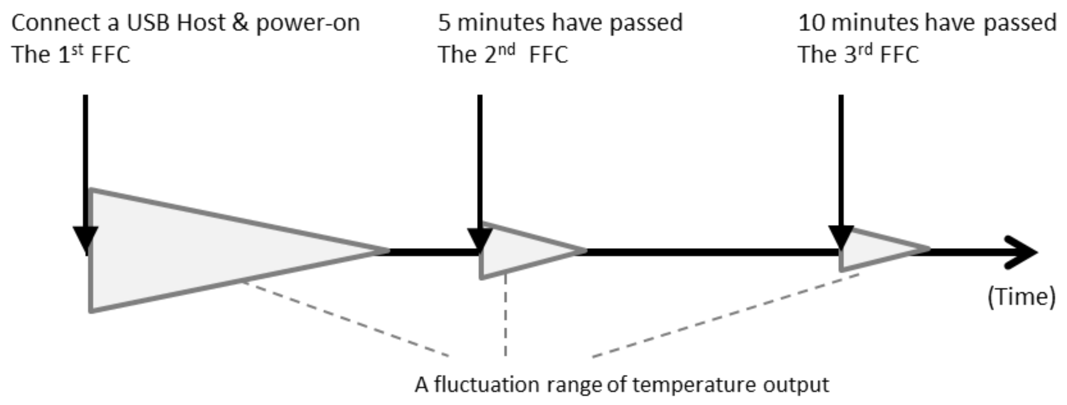
8.1 Mechanism

The OWLIFT outputs 14 bits data that shows IR intensity and is outputted by the sensor. The Library running on a USB host calculates it and outputs temperature values. The IR sensor that is equipped in the OWLIFT performs correction routinely, it is called FFC (Flat Field Correction). The library converts frame data into temperature values by using the information at the time which FFC is executed and a temperature of the sensor itself that is output by the sensor. The information at the time which FFC is executed and the temperature of the sensor itself are included in the telemetry data that is gotten along with frame data.

8.2 Relation between elapsed time and temperature output

The temperature output is affected by various factors. Refer to the section 3.5 "Temperature Output" in "OWLIFT User's Guide" regarding general factors. This section shows a factor only regarding FFC.

FFC affects output of the sensor much and also affects temperature output. The OWLIFT Type-A/A2/A3/B performs FFC immediately after power-on and also does every 5 minutes. The OWLIFT Type-F performs FFC immediately after power-on, does every few seconds at first and every few minutes as time goes by. An output value of the sensor fluctuates after power-on and FFC because of characteristics of the sensor. By a FFC action, the output value will fluctuate in spite of the fact that an observation object keeps same temperature. The following figure shows the relation of the range of fluctuation between the elapsed time and the temperature output about OWLIFT Type-A/A2/A3/B.



The temperature output after power-on is the largest fluctuation. It is making stable little by little, but it again fluctuates much after the second FFC action. The fluctuation band is smaller than a point of power-on. After that, it fluctuates at the FFC action that is performed every five minutes, and it will make most stable when the sensor's temperature becomes stable. It will generally become stable in 5 - 10 minutes after the first FFC action. But it might fluctuate slightly after a FFC action in spite of being stable.

Furthermore, it is important in order to make the temperature output stable that the sensor's temperature make stable. Therefore the temperature output will fluctuate much when the ambient temperature changes much after it makes stable.

8.3 Relation between sensor command and temperature output

The OWLIFT sets an interval of performing the FFC action immediately after power-on by running the SYS FFC Mode Control command of the IR sensor module in the device.

You can overwrite the SYS FFC Mode Control by using the library. Also you can perform the FFC action in any time by using the SYS Run FFC Normalization. But you have to know that a measurement error of temperature output values may increase if you change the interval of the FFC action because the library is supposing that the interval of the FFC action is 5 minutes.

The OWLIFT doesn't support the Radiometry Mode of the IR sensor module. Therefore the temperature output values will be indeterminate when the Radiometry Mode is turned on.

9 Restrictions

■ About controlling the sensor

- We can't answer a question about specification of commands of the sensor. Refer to the reference documents - 2 for specification of commands of the sensor. But we can't guarantee system behavior brought from executing the command of the sensor. Execute it by your own responsibility.
- Image output might be stopped when a series of certain commands of the sensor is performed continuously in a short time during getting images. Perform commands at enough intervals if it stops.
- The readable and writable registers are only DATA0 - DATA15.
- The OWLIFT doesn't support the AGC mode equipped by the sensors. The OWLIFT SDK on a USB host processes the AGC function for the OWLIFT.
- The OWLIFT doesn't support the "Header" for the SYS Telemetry Location.

10 Reference documents

1. Datasheets of IR sensors
   - https://infinitegra.co.jp/static/owlift/docs/Sensor.pdf
2. Interface of IR sensors
   - https://infinitegra.co.jp/static/owlift/docs/IDD.pdf

11 Release Notes

1.9.3

New features

- [Python] New API:
  - OwDevice.upside_down_enabled
  - OwDevice.magnification_enabled
  - OwDevice.rgb_order
- [C] New API:
  - OwLib_GetUpsideDown(), OwLib_SetUpsideDown()
  - OwLib_Magnify3()
  - OwLib_Gray8ToRGB()
  - OwLib_GetRGBOrder(), OwLib_SetRGBOrder()
- [C#] New API:
  - OwDev.UpsideDown
  - OwDev.RGBOrder

Fixed problems

- [Linux] Fixed the problem that raw-recorded files could not be written over 2GB.

Misc. changes

- [Python] When playing a raw-recorded file, OwDevice.frame now returns (None, None, None) when the end of the file is reached. Note that OwDevice.alive is True just before OwDevice.frame returns (None, None, None).
- [Linux] Reduced CPU load when playing raw-recorded files.

1.9.2

New features

- [Python] New API:
  - OwDevice.command_get()
  - OwDevice.command_set()
  - OwDevice.command_run()

1.9.1

New features

- [Win32/C] New API:
  - OwLib_EnableHighGainMode()
- [Linux/C] New API:
  - OwLib_EnableHighGainMode()
  - OwLib_Reconnect()
- [Win32/C#] New API:
  - OwDev.EnableHighGainMode()
- [Python] New API:
  - OwDevice.enable_high_gain_mode()
  - OwDevice.reconnect ()

Fixed problems

- [Windows] Fixed the problem that OwLib_Reconnect() does not work against Type-F. Note that this problem was fixed after the firmware version 3.1 (shipped from June, 2020).
- [Linux/Python] Fixed the problem that sometimes the exception was generated with the incorrect error code when an error occurred.

1.9.0

New features

- [Win32/C, Linux/C] New API:
  - OwLib_SetReflectCorrFile()
  - OwLib_SetAGCRange(),OwLib_GetAGCRange()
  - OwLib_SetAGCROIMask()
- [Win32/C#] New API:
  - OwDev.SetWindowCorrection()
  - OwDev.AGCRange
  - OwDev.SetAGCROIMask()
- [Python] New API:
  - OwDevice.set_reflection_correction_file()
  - OwDevice.agc_range
  - OwDevice.agc_roi_mask

Misc. changes

- Changed the algorithm of the Protection Window Correction for OWLIFT Type-A.

1.8.0

New features

- Added support for OWLIFT Type-F.
- [Win32/C, Linux/C] New API:
  - OwLib_GetUndistortion(), OwLib_SetUndistortion()
  - 定数 OWFRAMERATE_DEFAULT
- [Win32/C#] New API:
  - OwDev.Undistortion
  - OwFrameRate.DEFAULT
- [Python] New API:
  - OwDevice.undistortion

Misc. changes

- Lowered the strength of Noise Filter for OWLIFT Type-B.

1.7.0

New features

- Added Python Library.
- [Win32/C, Linux/C] New API:
  - OwLib_GetDisconnected()
- [Win32/C#] New API:
  - OwDev.Disconnected

Change history

Rev	Date	Content
1.9.3	2023-06-01	・ Fixed URL links.
1.9.3	2022-04-12	・ Updated release notes. ・ Changed supported versions of Python. ・ Changed supported version of gcc.
1.9.2	2021-06-01	・ Changed supported versions of Android, Python and Visual Studio.
1.9.2	2021-01-14	・ Updated release notes. ・ Removed old history and release notes.
1.9.1		・ Updated release notes.
1.9.0		・ Updated release notes.
1.8.0		・ 1.2 Stopped supporting Windows SP1/8.1 and Windows 32bit. ・ 1.2 Stopped supporting Visual Studio 2013. ・ 6.1 Renamed Python .whl files. ・ 8.2 Added description about the shutter interval of Type-F. ・ Updated release notes.
1.7.0		・ 1.2 Updated system requirements. ・ 6. Added python library. ・ Deleted descriptions about DirectShow filters. ・ Reconstruct indicies. ・ Updated release notes.