OWLIFT SDK
開発者ガイド

Rev 1.9.3

2023-06-01

本書についての注意事項

- 本書の内容の一部または全部を弊社に無断で転載することをお断り致します。
- 本書の内容は将来予告なしに変更することがあります。
- 本書にご不明な点、誤りがあれば弊社までご連絡ください。

本製品についての注意事項

- 本書の内容に従わない方法で製品を扱わないでください。故障などの原因となることがあります。万一故障などが発生した場合は弊社で責任を負い兼ねることがあります。
- 本製品を安全性や信頼性が求められる用途や、生命に直接影響を及ぼす可能性のある用途ではご使用されないようお願い致します。 また、それらの用途においてご使用された場合に生じたいかなる結果についても、弊社では責任を負い兼ねます。
- 本製品の使用によって生じたデータの消失や破損、その他いかなる結果や異常についても、弊社では責任を負い兼ねます。
- 本製品の分解や改造、ファームウェアの逆コンパイルや逆アセンブルなどのリバースエンジニアリングに相当する行為は固くお断り致します。

目次

1 概要

本書は遠赤外線カメラ OWLIFT（オーリフト）の開発キット OWLIFT SDK について説明した文書です。

1.1 SDK の内容

OWLIFT SDK には以下のものが含まれています。

- Windows 版開発キット
  - Windows 版 C ライブラリ
  - Windows 版.NET ライブラリ
  - Windows 版 Python ライブラリ
  - サンプルコード
  - コマンドラインツール
    - OWLIFTConfig: センサのレジスタにアクセスするツール
    - OWLIFTDump: OWLIFT の画像出力を取得するツール

- Linux 版開発キット
  - Linux 版 C ライブラリ
  - Linux 版 Python ライブラリ
  - サンプルコード
  - コマンドラインツール
    - OWLIFTConfig: センサのレジスタにアクセスするツール
    - OWLIFTDump: OWLIFT の画像出力を取得するツール

- Android 版開発キット
  - Android 版 C ライブラリ
  - Android 版 JAR ライブラリ
  - サンプルコード

1.2 動作環境

1.2.1 ランタイム環境

〔Windows 版 C/.Net ライブラリ〕

- Windows 10 64bit

〔Linux 版 C ライブラリ〕

- Linux Kernel 2.6.32 以上 x86 / x86_64, glibc 2.11 以上, 要 Vidoe4Linux2
- Linux Kernel 3.2.0 以上 armhf, glibc 2.13 以上, 要 Vidoe4Linux2

〔Android 版 Java ライブラリ〕

- Android 7.0 - 11
  ※ 全ての Android 端末で動作することを保証するものではありません。

〔Python ライブラリ〕

Python 3.4 - 3.10

- Windows 10 64bit
- Linux Kernel 4.4.0 以上 x86 / x86_64, glibc 2.19 以上, 要 Vidoe4Linux2
- Linux Kernel 4.9.0 以上 armhf, glibc 2.19 以上, 要 Vidoe4Linux2

1.2.2 ビルド環境

〔Windows 版 C/.Net ライブラリ〕

- コンパイラ: Visual Studio 2017
- 32bit ビルド、64bit ビルドに対応。

〔Linux 版 C ライブラリ〕

- コンパイラ: g++-4.6
- x86 ビルド、x64 ビルド、armhf (Raspbian)ビルドに対応。

〔Android 版 Java ライブラリ〕

- Android Studio

1.3 表記

文章中で以下の表記を用います。

・ %OWLIFT_SDK_DIR%

OWLIFT SDK をインストールしたフォルダの絶対パスを表します。

2 Windows 版 C ライブラリ

2.1 インストール

OWLIFT-SDK-#.#.#.zip を任意のフォルダに展開します。#.#.#の部分は実際はバージョン番号を示します。以下、展開したフォルダのパスを %OWLIFT_SDK_DIR% と表記します。

2.2 アンインストール

展開したフォルダ（%OWLIFT_SDK_DIR% ）を削除します。

2.3 コンパイル

Windows 版 C ライブラリを利用するにはコンパイラで以下の設定を行います。

1. インクルードパスに %OWLIFT_SDK_DIR%¥include を追加します。
2. ライブラリパスに以下のパスを追加します。ライブラリファイル（owlift.lib）は自動的にリンクされます。
   - x86 バイナリ … %OWLIFT_SDK_DIR%¥lib¥x86
   - x64 バイナリ … %OWLIFT_SDK_DIR%¥lib¥x64

2.4 実行時に必要なファイル

Windows 版 C ライブラリを利用したプログラムを実行する際には以下のファイルが必要です。実行時に PATH 環境変数へ親ディレクトリを追加するか、プログラムと同じディレクトリにコピーします。

- x86 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x86¥owlift.dll
- x64 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x64¥owlift.dll

2.5 インクルードファイル

以下のファイルをインクルードします。

・ OWLIFTLib.h

2.6 リファレンスマニュアル

以下の URL の“OWLIFT C Library for Windows”を参照してください。

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

2.7 画像を表示する処理の流れ

以下に画像を表示する処理について説明します。各関数の詳細はリファレンスマニュアルをご参照ください。実際のコードについてはサンプルソースコードを参考にしてください。

1. OwLib_GetDevices() により、デバイスハンドルを取得します。
2. OwLib_CaptureSetup() により、画像取得の設定を行います。
3. OwLib_CaptureStart() を実行して、画像取得を開始します。

4. OwLib_CaptureSetup() で登録したコールバック関数がフレーム毎に呼ばれます。
5. コールバック関数内で、OwLib_Decode() によりデコード処理を行います。
6. 必要に応じて OwLib_GetTempTable() により温度データを取得します。
7. OwLib_FinishDecode() を実行して、デコード処理を完了します。6.よりも後に実行してください。
8. 5.の出力を任意の方法で描画します。また、必要に応じて 6.の出力を任意の方法で描画します。
9. 画像取得を終了するときは、OwLib_CaptureStop() を実行します。デバイスハンドルを解放するには、OwLib_Release() を実行します。

2.8 サンプルコード

2.8.1 OWLIFTView

OWLIFTView は Visual C++のサンプルソースコードです。SDK にビルド済みの実行ファイルが含まれます。

以下に示す基本的な処理を実装しています。

- フレームデータの取得と表示。
- 温度テーブルの取得。
- センサのコマンド実行。
- フレームデータに含まれる Telemetry Data の取得と表示。

Visual Studio 上でコンパイルして実行する際は、実行パス上に owlift.dll が存在している必要があります。パスを通す方法の一つとして、プロジェクトの設定を変更する方法を以下に示します。

1. OWLIFTView プロジェクトのプロパティの[デバッグ]を開きます。
2. [作業ディレクトリ]を “$(ProjectDir)¥..¥..¥bin¥x86” に変更します。

3 Windows 版.NET ライブラリ

3.1 インストール

“2.1 インストール”と同じです。

3.2 アンインストール

“2.12.2 アンインストール”と同じです。

3.3 コンパイル

Windows 版.NET ライブラリを利用するにはコンパイラで以下の設定を行います。

参照設定に以下の DLL を追加します。

- x86 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x86¥owliftnet.dll
- x64 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x64¥owliftnet.dll

3.4 実行時に必要なファイル

Windows 版.NET ライブラリを利用したプログラムを実行する際には以下のファイルが必要です。実行時に適切にパスを通すか、プログラムと同じディレクトリにコピーします。

- x86 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x86¥owlift.dll
  %OWLIFT_SDK_DIR%¥bin¥x86¥owliftnet.dll
- x64 バイナリ … %OWLIFT_SDK_DIR%¥bin¥x64¥owlift.dll
  %OWLIFT_SDK_DIR%¥bin¥x64¥owliftnet.dll

3.5 リファレンスマニュアル

以下の URL の“OWLIFT.NET Library for Windows”を参照してください。

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

3.6 画像を表示する処理の流れ

以下に画像を表示する処理について説明します。各関数の詳細はリファレンスマニュアルをご参照ください。実際のコードについてはサンプルソースコードを参考にしてください。

1. OwDev.GetDevices() により、デバイスハンドルを取得します。
2. OwDev.CaptureSetup() により、画像取得の設定を行います。
3. OwDev.CaptureStart() を実行して、画像取得を開始します。
4. OwDev.CaptureSetup() で登録したコールバック関数がフレーム毎に呼ばれます。
5. コールバック関数内で、OwDev.Decode() によりデコード処理を行います。
6. 必要に応じて OwDev.GetTempTable() により温度データを取得します。
7. OwDev.FinishDecode() を実行して、デコード処理を完了します。6.よりも後に実行してください。
8. 5.の出力を任意の方法で描画します。また、必要に応じて 6.の出力を任意の方法で描画します。

9. 画像取得を終了するときは、OwDev.CaptureStop() を実行します。デバイスハンドルを解放するには、OwDev オブジェクトを格納している変数に null を代入します。

3.7 サンプルコード

Windows、.NET のサンプルソースコードを次に示します。

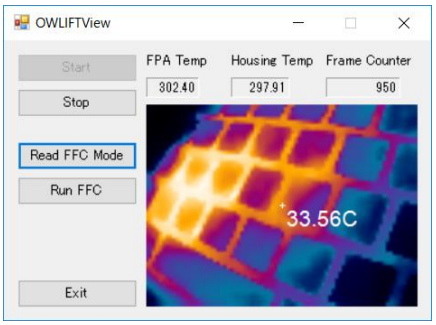
【注意点】

Visual Studio 上でコンパイルして実行する際は、実行パス上に owlift.dll と owliftnet.dll が存在している必要があります。パスを通す方法の一つとして、プロジェクトの設定を変更する方法を以下に示します。

1. プロジェクトのプロパティの[デバッグ]を開きます。
2. [作業ディレクトリ]で %OWLIFT_SDK_DIR%¥bin¥x86 を選択します。
   （%OWLIFT_SDK_DIR%は OWLIFT SDK がインストールされているフォルダ）

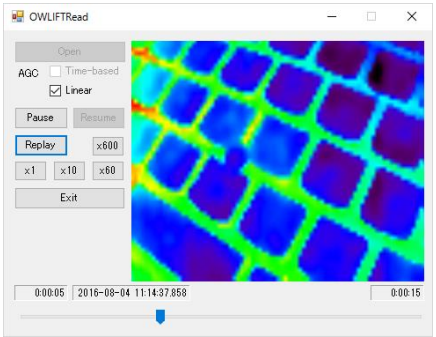
3.7.1 OWLIFTViewCS

OWLIFTViewCS は C#で記述されたサンプルソースコードで、Visual C++で実装された OWLIFTView と同等の機能を持ちます。SDK にビルド済みの実行ファイルが含まれます。



3.7.2 OWLIFTReadCS

OWLIFTReadCS は Raw 録画ファイル再生を行うサンプルソースコードです。SDK にビルド済みの実行ファイルが含まれます。



4 Linux 版 C ライブラリ

4.1 インストール

以下に示すファイルを任意の場所に配置します。#.#.#の部分は実際はバージョン番号を示します。

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

root ユーザに切り替わり、.deb ファイルを配置したディレクトリに移動して次のコマンドを実行します。

```
x86 の場合
# dpkg -i libowlift-dev_#.#.#-1_i386.deb
x64 の場合
# dpkg -i libowlift-dev_#.#.#-1_amd64.deb
armhf(Raspberry Pi)の場合
# dpkg -i libowlift-dev_#.#.#-1_armhf.deb
```

4.2 アンインストール

root ユーザに切り替わり、次のコマンドを実行します。

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

4.3 コンパイル

Linux 版 C ライブラリをインストールすると標準で /usr/include の下にインクルードファイルが配置され、/usr/lib の下に共有ライブラリが配置されます。プログラムをコンパイルするには、以下の設定を行ってください。

・ リンカのオプションに“-lowlift -lpthread”を追加します。

4.4 実行時に必要なファイル

Linxu 版 C ライブラリを利用したプログラムを実行する際には以下のファイルが必要です。

・ /usr/lib/libowlift.so.#

（# は実際は共有ライブラリのバージョンを表します。SDK のバージョンとは異なります。）

本書に示す方法でインストールした場合は、実行時の設定は不要です。共有ライブラリが標準的な場所にないときは、環境変数 LD_LIBRARY_PATH により、適切にパスを通してください。

4.5 インクルードファイル

以下のファイルをインクルードします。

・ OWLIFTLib.h

4.6 リファレンスマニュアル

以下の URL の“OWLIFT C Library for Linux”を参照してください。

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

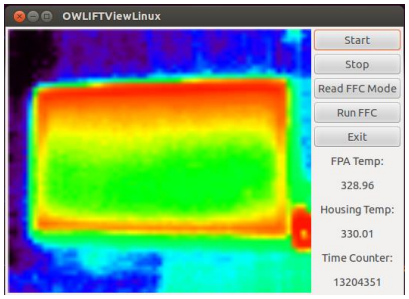
4.7 画像を表示する処理の流れ

Windows 版 C ライブラリと同じです。3.6 を参照してください。

4.8 サンプルコード

4.8.1 OWLIFTViewLinux

OWLIFTViewLinux は GTK+を使用した Linux 向けのサンプルソースコードです。Windows 向けのサンプルソースコードである OWLIFTView とほぼ同じ機能を持ちます。Linux 版 C ライブラリをインストールすることで、OWLIFTViewLinux の実行ファイルもインストールされます。



バイナリのパッケージは SDK 内の以下のファイルです。

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

(#.#.#はバージョン番号)

ソースコードは以下のファイルです。

- owliftviewlinux.tar.gz

コンパイルするにはアーカイブを展開して、以下のコマンドを実行します。GTK+ 2.0 ライブラリ開発用ファイル（libgtk2.0-dev）、および SDL 1.2 ライブラリ開発用ファイル(libsdl1.2-dev)が必要です。

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

5 Android 版 Java ライブラリ

5.1 ライブラリ配置

サンプルソースコードを利用した Android Studio でのライブラリ配置場所について記載します。

- JAR ファイル：プロジェクトフォルダ直下に”libs”フォルダを作成し配置
- SO ファイル：プロジェクトフォルダ/app/src/main/jniLibs を作成し、プラットフォームごとにディレクトリとライブラリを配置

5.2 コンパイル

サンプルソースコードを任意の位置に配置して、Android Studio を起動してください。
“Open an existing Android Studio project”を選択し、配置したサンプルコードを選択してください。
必要に応じて、Android Studio の設定で JDK、Android SDK を設定してください。

5.3 リファレンスマニュアル

以下の URL の“OWLIFT Java Library for Android”を参照してください。

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

5.4 画像を表示する処理の流れ

以下に画像を表示する処理について説明します。各関数の詳細はリファレンスマニュアルをご参照ください。実際のコードについてはサンプルソースコードを参考にしてください。

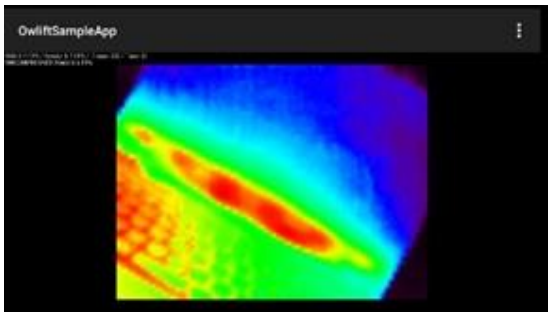
1. UsbOwliftHost を UsbOwliftHost.getUsbHost()で取得します。
2. UsbOwliftHost.openDev()によりデバイスをオープンします。
3. UsbOwliftDev.getFuncs()を利用して UvcOwliftFunc を取得します。
4. UvcOwliftFunc.startStream()を実行することで画像の取得が開始します。
5. UvcOwliftFunc.startStream()に登録したコールバック関数がフレーム毎に呼ばれます。
6. 画像取得の停止時には、UvcOwliftFunc.stopStream()、または UsbOwliftHost.close()を実行します。

5.5 サンプルコード

5.5.1 OwliftSampleApp

OwliftSampleApp は Android 向けのサンプルソースコードです。以下に示す基本的な処理を実装しています。

- フレームデータの取得と表示。
- 録画、静止画キャプチャ。
- 最大、最少の温度取得と表示。



6 Python 版ライブラリ

6.1 インストール

開発環境のプラットフォームに合った whl ファイルを pip コマンドでインストールします。
（#.#.#.# はバージョン番号）

```
Windows 32 ビットの場合
# pip install owlift-#.#.#.#-py3-none-win32.whl
Windows 64 ビットの場合
# pip install owlift-#.#.#.#-py3-none-win_amd64.whl
Linux x86 の場合
# pip install owlift-#.#.#.#- py3-none-linux_i686.whl
Linux x64 の場合
# pip install owlift-#.#.#.#- py3-none-linux_amd64.whl
armv7l(Raspberry Pi)の場合
# pip install owlift-#.#.#.#- py3-none-linux_armv7l.whl
```

6.2 アンインストール

pip コマンドでアンインストールします。

```
# pip uninstall owlift
```

6.3 リファレンスマニュアル

以下の URL の“OWLIFT Python Library”を参照してください。

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

6.4 サンプルコード

Python 版ライブラリのサンプルコードについては、サンプルコードのアーカイブに含まれている README_ja.txt を参照してください。

7 コマンドラインツール

7.1 OWLIFTConfig

OWLIFTConfig はセンサのレジスタにアクセスするコマンドラインインターフェースのプログラムです。以下の機能を持ちます。

- センサに対してコマンドタイプ GET、SET、RUN を実行。
- バッチ処理。
- デバイスの設定を変更。

コマンドオプションの詳細は、以下のコマンドを実行してご確認ください。本書で紹介されていないコマンドオプションも表示されます。

```
> OWLIFTConfig --help
```

7.1.1 GET、SET、RUN

実行例）Module ID=0x0200, Command ID Base=0x0C に対して、2 ワード（32bit）の GET を実行します。

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

実行例）Module ID=0x0200, Command ID Base=0x24 に対して、0x00000005 の SET を実行します。

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

実行例）Module ID=0x0200, Command ID Base=0x40 に対して、RUN を実行します。

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

※ センサのコマンド仕様については、 10 参考文献-2 を参照してください。また、センサのコマンドを実行した結果の挙動について、弊社では責任を負い兼ねます。

7.1.2 バッチ処理

--file オプションは、ファイルに記述した一連のコマンドオプションを連続で実行します。

実行例）config.txt ファイルに記述されたコマンドオプションを実行します。

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

config.txt の内容

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

7.1.3 デバイスの設定

USB を接続している間は、センサには常にパワーオンの状態です。以下のコマンドオプションにより画像取得中のみオンで、取得していないときは OFF となります。これは省電力を目的とした機能です。

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

7.2 OWLIFTDump

OWLIFTDump は温度データや Raw データを取得するコマンドラインインターフェースのプログラムです。以下の機能を持ちます。

- Telemetry Data と指定範囲の温度平均値（またはセンサ出力値）をフレーム毎に出力。
- Telemetry Data とフレームの各ピクセルに対応する温度値（またはセンサ出力値）をフレーム毎に出力。

コマンドオプションの詳細は、以下のコマンドを実行してご確認ください。本書で紹介されていないコマンドオプションも表示されます。

```
> OWLIFTDump --help
```

7.2.1 温度平均値の出力

実行例）矩形範囲(x1,y1)-(x2,y2)=(10,10)-(19,19)の温度平均値を出力します。

```
> 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
```

各項目の意味

- SYSTEMTIME: OS のシステム時間
- TIME: Telemetry Data の“Time Counter”
- TIME_FFC: Telemetry Data の“Time Counter at last FFC”
- FCS: Telemetry Data の“Status Bits”
- FPAT: Telemetry Data の“FPA Temp”
- FPATFC: Telemetry Data の“FPA Temp at last FFC”
- HOUT: Telemetry Data の“Housing Temp”
- HOUTFC: Telemetry Data の“Housing Temp at last FFC”
- TAVE1: 矩形範囲の平均温度値。
- TMIN1: 矩形範囲の最低温度値。
- TMAX1: 矩形範囲の最高温度値。

--region は最大 4 個まで指定可能です。--region を追加すると、出力項目に TAVE2/TMIN2/TMAX2、TAVE3/TMIN3/TMAX3、TAVE4/TMIN4/TMAX4 が順次追加されます。

7.2.2 Raw 録画

--write-owi で Raw 録画を実行します。Raw 録画ファイルは OWLIFTDump、OWLIFTCap で再生可能です。

実行例）thermal.owi に録画します。

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

7.2.3 Raw 録画ファイルの再生

--read-owi で Raw 録画ファイルを再生します。

実行例）thermal.owi を再生します。

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

8 温度出力の動作

温度を出力する機能（OwLib_GetTempTable()、OwDev.GetTempTable()）の仕様と注意事項について説明します。

8.1 温度出力の仕組み

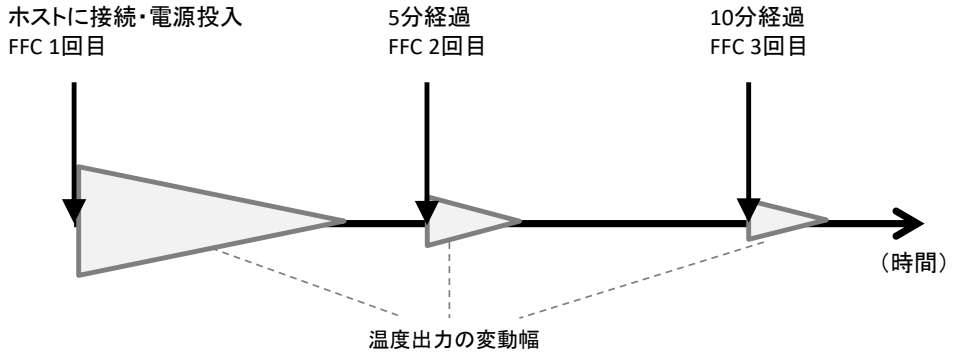
OWLIFT が出力するデータはセンサが出力する赤外線強度を表す 14bit データであり、温度出力の計算はホスト上のライブラリで行われます。OWLIFT に搭載されている赤外線センサ（500-0659-01）は定期的に FFC（Flat Field Correction）と呼ばれる補正処理を実行します。ライブラリは FFC 実行時の情報と、センサが出力するセンサ自体の温度を元に、フレームデータを温度値に変換します。

FFC 実行時の情報とセンサ自体の温度は、フレームデータと共に取得される Telemetry Data に含まれています。

8.2 経過時間と温度出力の関係

温度の出力値は様々な要因から影響を受けます。一般的な要因については「OWLIFT ユーザーズガイド」の「3.5 温度表示」を参照してください。ここでは FFC に関連する要因について説明します。

FFC はセンサの出力値に大きな影響を与え、温度出力にも同様に影響を与えます。OWLIFT において FFC は電源投入直後に 1 回、以降は OWLIFT Type-A/A2/A3/B の場合は 5 分毎に実行されます。OWLIFT Type-F の場合は最初は数秒ごと、時間が経過すると数分の間隔で実行されます。赤外線センサの特性上、電源投入直後と FFC 実行直後はセンサの出力値が大きく変動します。測定対象物の温度に変化がなくても、FFC に伴って温度の出力値が変動します。下図に OWLIFT Type- A/A2/A3/B の場合の経過時間と温度出力の変動幅の関係を示します。



電源投入直後は温度出力の変動幅が最も大きな状態です。徐々に安定して 2 回目の FFC が実行されると再び大きく変動します。このときの変動幅は電源投入直後よりも小さくなります。その後、5 分毎の FFC 実行直後に変動し、センサ自体の温度が平衡状態に入ったとき最も安定します。通常、安定するまでに、1 回目の FFC から 5～10 分程度要します。安定した状態に至っても、FFC が実行された直後には多少変動します。

なお、温度出力が安定するにはセンサ自体の温度が平衡状態に至ることが重要なので、平衡状態に至った後に周辺の温度が急激に変化すると温度出力の変動幅が大きくなります。

8.3 センサのコマンドと温度出力の関係

OWLIFT は電源投入直後にデバイス内部で赤外線センサモジュールのコマンドである SYS FFC Mode Control を実行することで、FFC を実行するタイミングの設定を行っています。

ライブラリを通じて SYS FFC Mode Control を上書きすることは可能です。また、SYS Run FFC Normalization により任意のタイミングで FFC を実行することも可能です。しかし、ライブラリは 5 分毎の FFC を前提として温度を計算するため、FFC の実行タイミングを変更した場合、ライブラリの温度出力の誤差が大きくなる可能性があることに注意してください。

OWLIFT では赤外線センサモジュールの Radiometry Mode を使用していません。Radiometry Mode を有効なときのライブラリの温度出力の動作は不定です。

9 制限事項

■ センサの制御

- センサのコマンドの仕様について弊社ではお答えできません。センサのコマンド仕様については、10 参考文献-2 を参照してください。また、センサのコマンドを実行した結果の挙動について、弊社では責任を負い兼ねます。
- 画像の取得中に、短時間で連続的にセンサのコマンドを実行すると、コマンドによっては画像の出力が停止することがあります。停止する場合は、間隔を空けてコマンドを実行してください。
- 読み書き可能な 500-0659-01 のレジスタは DATA 0～DATA 15 のみです。
- センサに搭載されている AGC Mode には対応していません。AGC は OWLIFT SDK によりホスト上で実現されます。
- SYS Telemetry Location の Header には対応していません。

10 参考文献

1. 赤外線センサのデータシート
- https://infinitegra.co.jp/static/owlift/docs/Sensor.pdf

2. 赤外線センサのインターフェース
- https://infinitegra.co.jp/static/owlift/docs/IDD.pdf

11 リリースノート

1.9.3

新しい機能

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

問題の修正

- [Linux] Raw 録画ファイルを 2GB 以上書き込めない問題を修正。

その他の変更

- [Python] Raw 録画ファイル再生時、ファイル終端に達したとき OwDevice.frame が（None, None, None）を返すようになった。OwDevice.frame を実行する直前は OwDevice.alive は True であることに注意。
- [Linux] Raw 録画ファイル再生時の CPU 負荷を低減。

1.9.2

新しい機能

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

1.9.1

新しい機能

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

問題の修正

- [Windows] OwLib_Reconnect() が Type-F に対して動作していなかった問題を修正。ただしファームウェアバージョン 3.1 以降（2020/6 以降の出荷版）で動作します。
- [Linux/Python] エラー発生時に適切なエラーコードで例外が生成されていないことがあった問題を修正。

1.9.0

新しい機能

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

その他の変更

- OWLIFT Type-A の保護窓補正のアルゴリズムを変更しました。

1.8.0

新しい機能

- OWLIFT Type-F に対応。
- [Win32/C, Linux/C] 追加 API:
  - OwLib_GetUndistortion(), OwLib_SetUndistortion()
  - 定数 OWFRAMERATE_DEFAULT
- [Win32/C#]追加 API:
  - OwDev.Undistortion
  - OwFrameRate.DEFAULT
- [Python]追加 API:
  - OwDevice.undistortion

その他の変更

- OWLIFT Type-B 使用時のノイズフィルタの強度を低い方へ変更。

1.7.0

新しい機能

- Python ライブラリを追加。
- [Win32/C, Linux/C] 追加 API:
  - OwLib_GetDisconnected()
- [Win32/C#]追加 API:
  - OwDev.Disconnected

本書の変更履歴

Rev	日付	内容
1.9.3	2023-06-01	・URL リンクを修正。
1.9.3	2022-04-12	・リリースノートを更新。 ・サポート対象の Python バージョンを変更。 ・サポート対象の gcc バージョンを変更。
1.9.2	2021-06-01	・サポート対象の Android バージョン、Python バージョン、Visual Studio バージョンを変更。
1.9.2	2021-01-14	・リリースノートを更新。 ・古い変更履歴、リリースノートを一部削除。
1.9.1		・リリースノートを更新。
1.9.0		・リリースノートを更新。
1.8.0		・1.2 Windows 7/8.1、Windows 32bit サポートを削除。 ・1.2 Visual Studio 2013 サポートを削除。 ・6.1 Python の.whl ファイル名を変更。 ・8.2 Type-F のシャッター間隔について追記。 ・リリースノートを更新。
1.7.0		・1.2 動作環境を更新。 ・6.Python 版ライブラリを追加。 ・DirectShow フィルタに関する説明を削除。 ・見出しの構成を変更。 ・リリースノートを更新。