ED-AIC3000

用户手册

by EDA Technology Co., Ltd

built: 2025-08-01

1 硬件手册

本章介绍产品概述、包装清单、外观、按键、指示灯和接口等。

1.1 产品概述

ED-AIC3000是一款基于Raspberry Pi CM5的1200万像素的工业智能相机，采样率高达70 FPS，根据不同的应用场景和用户需求，可选择不同规格的RAM和eMMC系统。

- RAM可选规格包含2GB、4GB和8GB
- eMMC闪存可选规格包含16GB、32GB和64GB

ED-AIC3000系列设备集成模块光源设计，采用带液态模组变焦的M12固定焦距镜头，具备亮场和暗场模式，可在正常、蚀刻、高光或纹理化表面实现最佳打光效果。

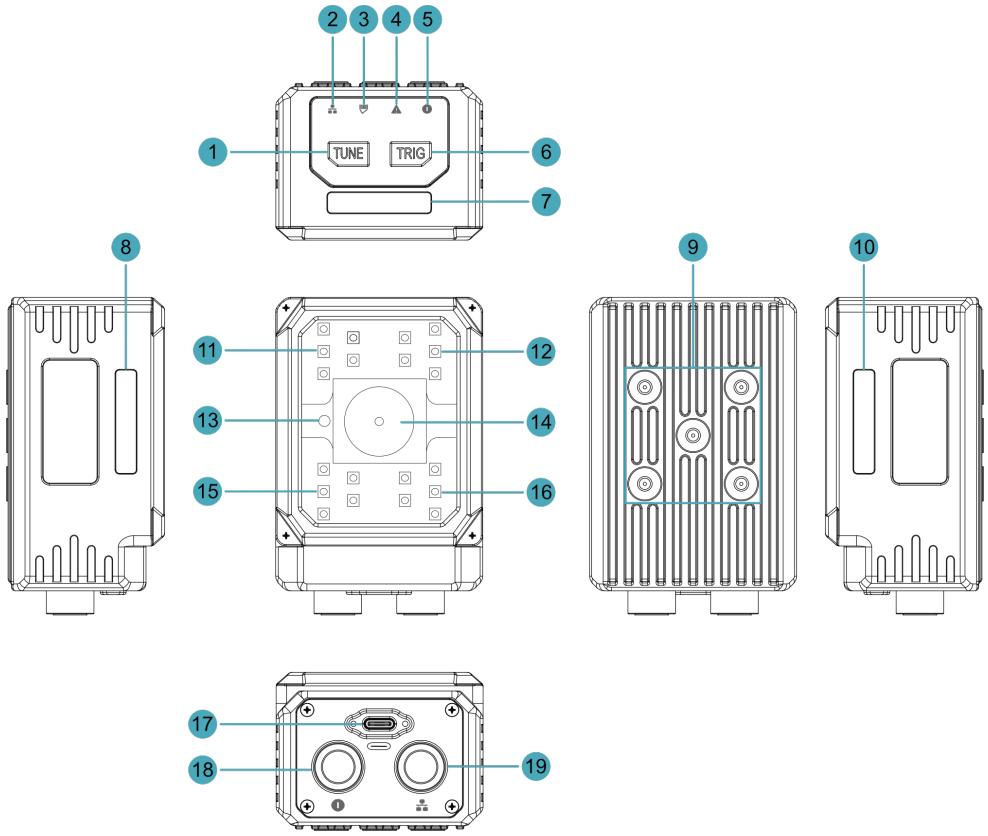
ED-AIC3000系列设备提供电源接口、I/O 接口、 RS232 串口和千兆以太网等接口，采用M12航空连接器，支持 IP65 防水等级，支持通过以太网接入网络，主要应用于机器视觉和人工智能领域。

1.2 包装清单

1 x ED-AIC3000主机

1.3 产品外观

介绍产品接口、按键和指示灯的功能和定义。



编号	功能定义
1	1 x 调节按键（TUNE），一键式自动对焦按钮或用户自定义按钮
2	1 x 网络连接指示灯，用于查看网络连接的状态。
3	1 x 工作状态指示灯，用于查看设备工作的状态。
4	1 x 系统故障指示灯，用于查看系统是否发生故障。
5	1 x 电源指示灯，用于查看设备上电状态。
6	1 x 触发按键（TRIG），用于相机触发或用户自定义的一键式按钮
7、8、10	3 x RGB灯（1组），灯光可设置为红色、绿色、蓝色、黄色和白色，用户可根据实际需要进行自定义
9	5 x M4螺丝孔，用于进行支架安装。
11、12、15、16	4 x 光源，用于设备工作时补光。
13	1 x 激光灯，红色十字激光，用于拍照定位
14	1 x 镜头，用于拍照
17	1 x type-c USB接口，用于eMMC烧录
18	1 x 电源接口，包含电源接口、I/O接口和RS232串口，采用12-Pin M12航空连接器
19	1 x 通信接口，千兆以太网接口，采用8-Pin M12 A-code航空连接器，用于接入以太网。

1.4 按键

ED-AIC3000系列设备包含2个按键，调节按键和触发按键。

- 调节按键（TUEN），在外壳上的丝印为“TUNE”，按下按键可以一键式自动对焦，支持用户自定义功能。
- 触发按键（TRIG），在外壳上的丝印为“TRIG”，按下按键可以触发相机，支持用户自定义功能。

引脚定义

按键引脚定义如下：

按键	CM5 引脚
调节按键（TUEN）	GPIO20
触发按键（TRIG）	GPIO12

1.5 指示灯

介绍ED-AIC3000系列设备包含的指示灯的各种状态及含义。

指示灯	状态	描述
网络连接指示灯	常亮	已正常接入以太网
	熄灭	未接入以太网
工作状态指示灯	闪烁	系统工作状态正常
	熄灭	系统工作状态异常
系统故障指示灯	闪烁	系统出现故障
	熄灭	系统未出现故障
电源指示灯	常亮	设备已上电
	熄灭	设备未上电

引脚定义

指示灯	CM5 引脚
电源指示灯	N/A
故障指示灯	GPIO21
工作状态指示灯	GPIO7(异常) GPIO16(正常)

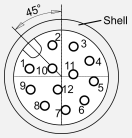
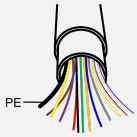
指示灯	CM5 引脚
网络指示灯	N/A

1.6 接口

介绍产品中各接口的定义和功能。

1.6.1 电源&I/O接口

ED-AIC2000系列设备包含1个电源&I/O接口，采用12-Pin M12航空连接器。电源&I/O接口包含1路电源输入、1路串口、1路DI和2路DO。电源&I/O接口用来连接电源&I/O线，线缆的一端为M12端子，用来连接Camera；另一端为裸线，用来连接电源、DI、DO和RS232。其中M12接口的引脚和裸线的定义如下表：

M12 端子引脚 	裸线颜色 	定义 
1	黄	DC-
2	白/黄	DC+
3	棕	COMMON_IN
4	白/棕	DI_1
5	紫	Trigger
6	白/紫	COMMON_OUT
7	红	External Strobe
8	黑	DO_1
9	绿	DO_2
10	橙	RS232_GND
11	蓝	RS232_TX
12	灰	RS232_RX
外壳	黑 (粗)	PE

其中1路DI和2路DO对应的CM5的GPIO引脚，如下表。

信号	CM5 引脚
DI1	GPIO17

信号	CM5 引脚
DO1	GPIO22
DO2	GPIO27

提示

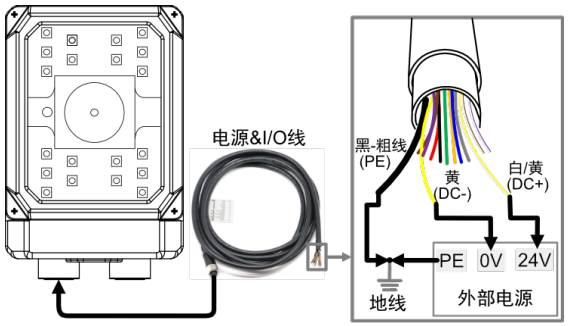
- 裸线中最粗的一条黑色线为PE。
- 针对Raspberry Pi HQ Camera M12，端子中引脚5和引脚7无功能定义。
- 安装时请确保线缆颜色与引脚定义保持一致，并正确连接PE至地线，以确保安全性和信号完整性。
- 接线过程中如果有未使用的裸线，请剪掉裸线顶端的金属部分或者使用绝缘胶带包住。

警告

请严格按照下文来连接电源、DI、DO和RS232，错误的接线可能会导致设备损坏。

1.6.1.1 连接电源

电源&I/O线的裸线中，白/黄和黄色分别用来接入外部电源的正极和负极，黑色粗线为PE用来接入地线。

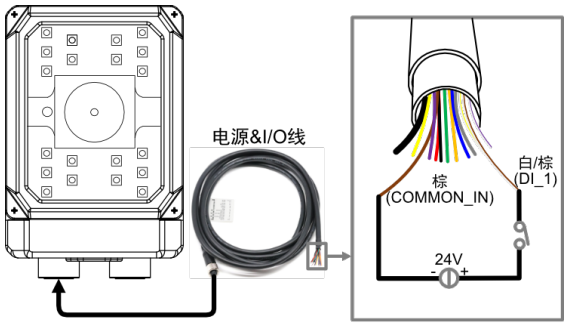


警告

- 设备供电为DC24V (±10%)，建议使用DC 24V 2A的电源适配器。
- 在连接电源线时，需要将裸线中PE和外部电源的PE与地线相连，确保PE接入地线。

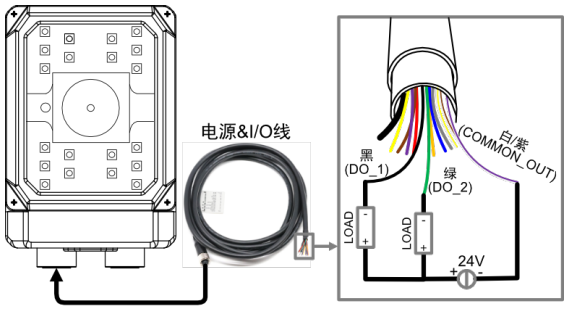
1.6.1.2 连接DI

电源&I/O线的裸线中，白/棕为1路DI，棕色为DI的公共端，支持接入NPN或PNP型传感器。



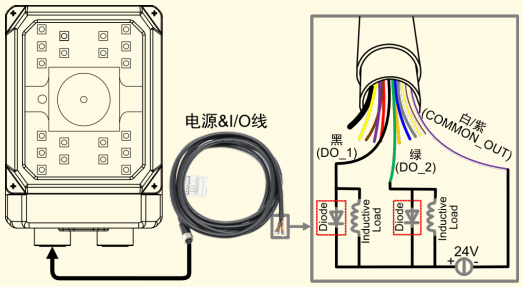
1.6.1.3 连接DO

电源&I/O线的裸线中，黑色和绿色为2路DO，白/紫为DO的公共端。DO单通道负载0.25A，连续的2个通道最大总电流0.5A，支持的负载类型包含阻性负载、灯负载和感性负载。



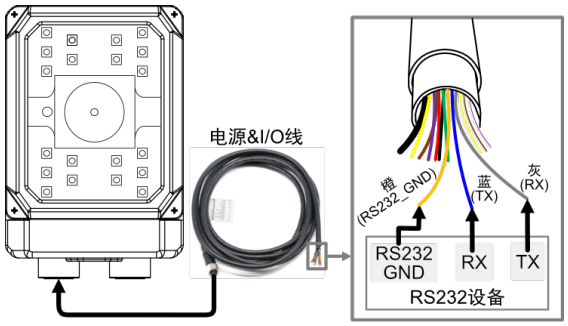
警告

- 请勿将白/紫线 (COMMON_OUT)接入24V的正极。
- 如果DO通道外接感性负载，建议在电路中增加一个续流二极管（如下图所示）作为保护。请根据感性负载的规格来选择合适的续流二极管。



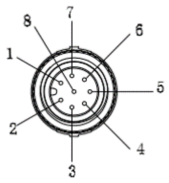
1.6.1.4 连接RS232

电源&I/O线的裸线中，橙色为RS232_GND，蓝色为RS232_TX，灰色为RS232_RX，支持连接到RS232设备。



1.6.2 通信接口

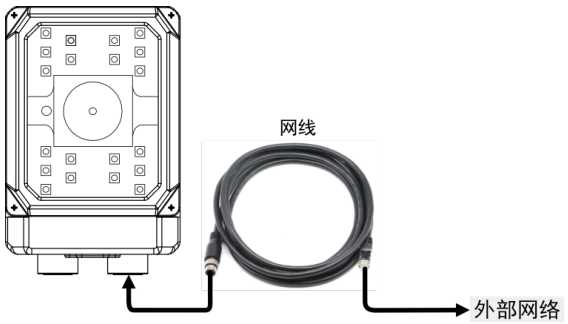
ED-AIC3000系列设备包含1路通信接口，采用8-Pin M12航空连接器，引脚定义如下：



Pin ID	Pin Name
1	TRD0+
2	TRD0-
3	TRD1+
4	TRD2+
5	TRD2-
6	TRD1-
7	TRD3+
8	TRD3-

线缆连接

通信接口用来连接网线使设备接入网络，网线的一端为M12端子，用来连接Camera；另一端为RJ45端子，用来接入网络。

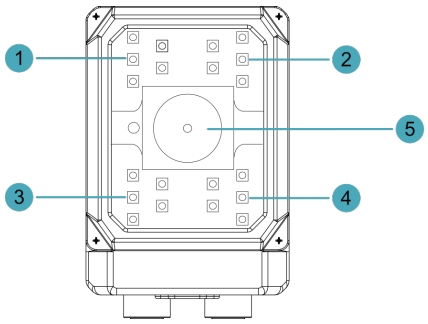


1.6.3 USB-C接口

ED-AIC3000系列设备包含1路USB Type-C接口，支持通过连接PC对设备的eMMC进行烧录。

1.7 光源和镜头

ED-AIC3000系列设备包含四部分光源和一个镜头。



编号	描述
1	光源部分 1，支持单独启用和禁用。
2	光源部分 2，支持单独启用和禁用。
3	光源部分 3，支持单独启用和禁用。
4	光源部分 4，支持单独启用和禁用。
5	镜头，带液态模组变焦的M12固定焦距镜头。

1.8 RGB灯

ED-AIC3000系列设备包含3个RGB侧灯，3个RGB灯为一组。支持通过软件设置灯光为红色、绿色、蓝色、黄色或白色，默认为关闭灯光状态，用户可根据实际需要进行自定义。

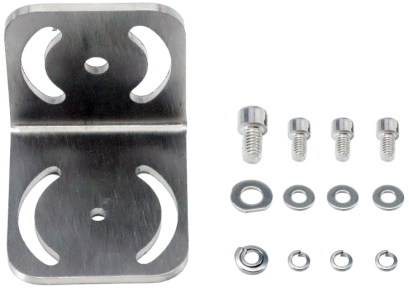
> 提示
>
> 3个RGB灯为一组，仅支持同时设置。

2 安装设备

本章介绍安装设备的具体操作。

前提条件：

- 已准备1把M4和1把M6的内六角螺丝刀。
- 已准备安装支架及安装螺钉(3xM4*8带垫圈、1xM6*10带垫圈)，如下图所示。

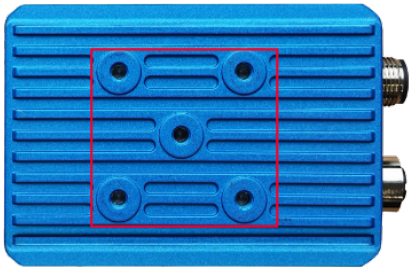


> 提示
>
> 安装支架和安装螺钉是选配件，需要单独购买。

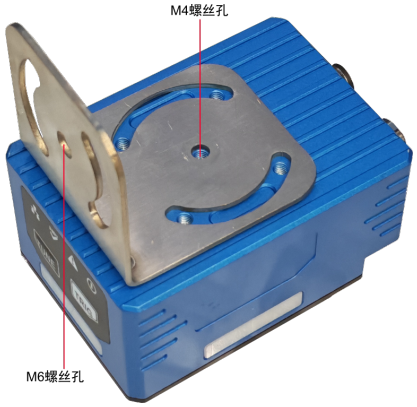
操作步骤：

> 提示
>
> 在使用M4和M6的螺钉时，参考下图放置弹簧垫圈和平垫圈。

1. 确定设备上支架安装孔的位置，如下图红框位置。



2. 将安装支架放置在设备安装孔的上方，使支架(带M4螺丝孔的一侧)和设备的中央螺丝孔位对齐。



3. 在中央螺丝孔的位置插入1颗带垫圈的M4螺钉，再使用M4的内六角螺丝刀将M4螺钉顺时针拧紧将支架固定在设备上。

4. （可选）根据需求旋转支架，调整安装方向。
5. 在外围螺丝孔的位置插入1颗带垫圈的M4螺钉，再使用M4的内六角螺丝刀将M4螺钉顺时针拧紧。

> 提示
>
> 建议使用1颗中央螺钉和1颗外围螺钉固定。

5. 使用M6螺钉固定支架和其他设备。

> 提示
>
> 根据现场不同的工程要求，调整合适的安装位置。
> 建议使用1颗中央螺钉和1颗外围螺钉固定。

3 启动设备

本章介绍连接线缆和启动设备的具体操作。

3.1 连接线缆

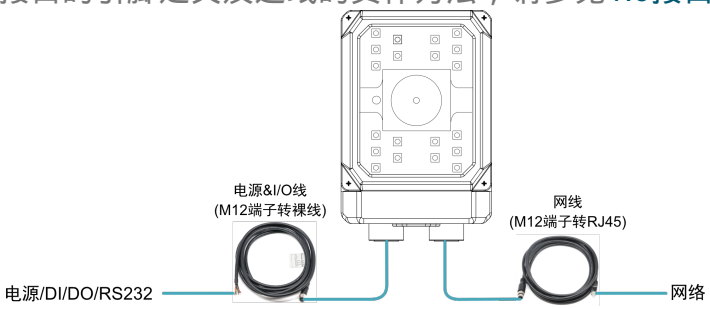
介绍线缆的连接方法。

准备工作：

- 已获取待连接的网线（M12端子转RJ45）和电源&I/O线（M12端子转裸线）。
- 已准备DC 24V 2A的电源适配器和辅助接线的连接器。
- 按需准备电源&I/O线所需的连接器和接线工具。

连接线缆示意图：

各接口的引脚定义及连线的具体方法，请参见1.6接口。



> 提示
>
> 不同批次的网线，颜色可能不同。

3.2 首次启动系统

ED-AIC3000系列设备无电源开关，接入电源后，系统将会开始启动。

- 电源指示灯常亮，表示设备已正常供电。
- 工作状态指示灯常亮，表示系统正常启动。

系统启动后，默认使用用户名和密码进行登录，因为相机无法连接显示器，故需要通过PC远程登录系统。
ED-AIC3000系列设备出厂镜像默认使能VNC，并将设备IP设置为静态IP：192.168.1.10，用户可以通过VNC来远程连接设备，并进入设备的桌面系统。具体的操作参考通过VNC远程连接到桌面。

> 提示
>
> 默认用户名：`pi`；默认密码：`raspberry`。

4 系统配置

本章介绍系统配置的具体操作。

4.1 编译Camera演示

如何编译并启动Camera示例。

1. 进入到Camera示例文件所在的目录。

```sh
cd /usr/share/eda-aic-lib/examples/
```

2. 运行测试代码

- Python

```sh
sudo python test_camera_hq.py
```

- c++

```sh
mkdir test
cd test
sudo cmake .. -DENABLE_HQ=ON
sudo make
sudo ./test_io
```

4.2 Camera I/O 控制

- `eda-io.h` 文件提供操作AI Camera I/O的控制接口(C++)的程序，该文件的存放路径为 `/usr/include/eda/eda-io.h` 。
- `libedaio2.so` 文件提供操作AI Camera I/O的控制接口(Python)的程序，该文件的存放路径为 `/usr/lib/python3/dist-packages/libedaio2.so` 。

4.3 Camera Sensor 控制

ED-AIC3000使用开源的 `picamera2` 库来控制摄像头， `picamera2` 提供一系列api。

4.4 Camera API 示例

AI Camera Sensor的控制函数由`picamera2`库提供，支持打开和关闭传感器、捕获图片等，下表为函数的功能说明。

函数	功能
picam2 = Picamera2()	获取Camera控制实例
preview_config = picam2.create_preview_configuration()	创建一个预览配置
picam2.configure(preview_config)	将预览配置应用到摄像头
picam2.start_preview(Preview.NULL)	启动摄像头预览功能，NULL表示不在屏幕上显示画面
picam2.start()	启动摄像头硬件和软件流管道
picam2.capture_file("test.jpg")	捕获一张图片并将其保存为test.jpg
picam2.close()	关闭摄像头硬件和软件流管道

更多详细内容请查询`picamera2`官方资料Picamera2 Manual (https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf)

4.5 I/O API

I/O API提供了AI Camera I/O的控制函数，支持控制指示灯、控制激光、控制侧灯和控制输出等。

4.5.1 C++环境

AI Camera I/O在C++环境下的控制函数如下表所示：

函数	功能
eda::EdaIo *em = eda::EdaIo::getInstance()	获取IO控制实例
void setup()	初始化IO设置
void openLaser()	打开激光
void closeLaser()	关闭激光
void setScanStat(bool good)	设置状态指示灯
void openAlarm()	打开警告指示灯
void closeAlarm()	关闭警告指示灯
void setDo1High(bool high)	设置output1输出
void setDo2High(bool high)	设置output2输出
void registerInput(IoTrigger callback)	注册input触发回调函数

函数	功能
void registerTrigger(IoTrigger callback)	注册trigger按键回调函数
void registerTune(IoTrigger callback)	注册Tune按键回调函数
void setRgbLight(uint8_t light)	设置RGB灯光

4.5.2 Python3语言环境

AI Camera I/O在Python3语言环境下的控制函数如下表所示：

函数	功能
eda = EdaIo.singleton()	获取IO控制实例
eda.setup()	初始化
eda.openLaser()	打开激光
eda.closeLaser()	关闭激光
eda.open_light()	打开光源
eda.close_light()	关闭光源
eda.eda.enableLightSection()	使能光源模块（可取值：1，2，3，4）
eda.eda.disableLightSection()	关闭光源模块（可取值：1，2，3，4）
eda.setScanStat(True)	设置状态指示灯
eda.openAlarm()	打开警告指示灯
eda.closeAlarm()	关闭警告指示灯
eda.setDo1High(True)	设置output1输出
eda.setDo2High(False)	设置output2输出
registerInput(func_trigger)	注册input触发回调函数
registerTrigger(func_trigger)	注册trigger按键回调函数
registerTune(func_trigger)	注册Tune按键回调函数
eda.setRgbLight(1)	设置RGB灯光

4.5.3 操作说明

4.5.3.1 C++

1. 初始化

- 操作I/O前需要获取I/O实例 `eda::EdaIo *em = eda::EdaIo::getInstance();`

- 对实例初始化 `em->setup();`

2. I/O控制支持对事件注册回调函数

- input 输入事件 `em->registerInput(trigger_input);`
- Trigger 按键 `em->registerTrigger(trigger_trigger);`
- Tune 按键 `em->registerTune(trigger_tune);`

3. 控制I/O（必须先完成初始化）

- 控制激光 `em->openLaser()` 和 `em->closeLaser();`
- 控制状态指示灯 `em->setScanStat(true)` 和 `em->setScanStat(false);`
- 控制警报指示灯 `em->openAlarm()` 和 `em->openAlarm();`
- 控制两路输出 `em->setDo1High(false)` 和 `em->setDo2High(false);`

4. 控制灯光（必须先完成初始化）

- 控制侧灯颜色 `em->setRgbLight(1);`
  - 0: 关闭
  - 1: 红色
  - 2: 绿色
  - 3: 蓝色
  - 4: 黄色
  - 5: 白光
- 控制灯源
  - 使能灯源（默认已使能） `em->enableLightSection(1)` 取值范围1~4，对应不同分区
  - 禁用灯源 `em->disableLightSection(1)` 取值范围1~4，对应不同分区

4.5.3.2 Python

1. 初始化
   - 操作I/O前需要获取I/O实例 `eda = EdaIo.singleton()`
     - 对实例初始化 `eda.setup()`
2. I/O控制支持对事件注册回调函数

- input 输入事件 `registerInput(func_input)`
- Trigger 按键 `registerTrigger(func_trigger)`
- Tune 按键 `registerTune(func_tune)`

3. 控制I/O（必须先完成初始化）

- 控制激光 `eda.openLaser()` 和 `eda.closeLaser()`
- 控制状态指示灯 `eda.setScanStat(true)` 和 `eda.setScanStat(false)`
- 控制警报指示灯 `eda.openAlarm()` 和 `eda.openAlarm();`
- 控制两路输出 `eda.setDo1High(false)` 和 `eda.etDo2High(false);`

4. 控制灯光（必须先完成初始化）

- 控制侧灯颜色 `eda.setRgbLight(1);`
  - 0: 关闭
  - 1: 红色

  - 2: 绿色
  - 3: 蓝色
- 控制灯源
  - 使能灯源（默认已使能） `eda.enableLightSection(1)` 取值范围1~4，对应不同分区
  - 禁用灯源 `eda.disableLightSection(1)` 取值范围1~4，对应不同分区

4.6 gpiochip配置

进入系统后可以通过在/etc/aic/config.ini设置使用的gpiochip。

步骤：

1. 使用以下命令进入config,ini,并修改想要配置的gpiochip。

```sh
sudo nano /etc/aic/config.ini
```

```text
[gpio]
gpiochip=0
```

2. 设置完成后，按下'Ctrl+o', 'Enter'键保存，保存后按下'Ctrl+x'退出。
3. 执行以下命令重启设备。

```sh
sudo reboot
```

5 安装操作系统 (可选)

设备出厂时，默认带有操作系统。如果在使用过程中操作系统被损坏或者用户需要更换操作系统，则需要重新下载合适的系统镜像并进行烧录。我司支持通过先安装标准Raspberry Pi OS，再安装Firmware包，来实现操作系统的安装。

下文介绍镜像下载、eMMC烧录和安装Firmware包的具体操作。

5.1 镜像下载

下载的Raspberry Pi官方系统镜像，下载路径如下表：

> 提示
>
> 目前仅支持Raspberry Pi OS (Desktop) 64-bit系统。

OS	下载路径
Raspberry Pi OS (Desktop) 64-bit-bookworm (Debian 12)	https://downloads.raspberrypi.com/raspios_arm64/images/raspios_arm64-2024-11-19/2024-11-19-raspios-bookworm-arm64.img.xz (https://downloads.raspberrypi.com/raspios_arm64/images/raspios_arm64-2024-11-19/2024-11-19-raspios-bookworm-arm64.img.xz)

5.2 eMMC烧录

建议使用Raspberry Pi官方烧录工具，下载路径如下：

- Raspberry Pi Imager : https://downloads.raspberrypi.org/imager/imager_latest.exe (https://downloads.raspberrypi.org/imager/imager_latest.exe)
- SD Card Formatter : https://www.sdcardformatter.com/download/ (https://www.sdcardformatter.com/download/)

前提条件：

- 已获取1台Windows PC，并完成烧录工具的下载和安装。
- 已准备一根USB-C转USB-A的烧录线。
- 已准备一根8-Pin M12公头转RJ45的网线。
- 已准备一根12-Pin M12母头转裸线的电源IO线。
- 已获取待烧录的镜像文件。
- 已获取1台Linux PC（用于对Raspberry Pi CM5进行盘符化），并接入网络。

> 提示
>
> 由于Raspberry Pi的Rpiboot工具暂时不支持Raspberry Pi CM5在Windows操作系统上进行盘符化，故需要在linux操作系统的设备上进行盘符化。

操作步骤：

操作步骤以Windows系统为例进行说明。

1. 连接好电源线和USB烧录线（USB-C转USB-A）。

- 连接烧录线：一端连接设备侧的USB type-C接口，另一端连接Linux PC上的USB接口。
- 连接电源线：一端连接设备侧的12-Pin M12电源接口 (下图红框中的接口)，另一端连接外部电源。

2. 断开设备电源，保持长按TRIG按键，对设备重新上电，设备会自动进入烧录模式。
3. 通过Linux PC进行盘符化，具体如下。

a. 给Linux PC上电，开机启动系统，通过ssh连接或者通过连接显示器在终端依次执行以下命令从github克隆usbboot。

```sh
sudo apt update
git clone --recurse-submodules --shallow-submodules --depth=1 https://github.com/raspberrypi/u
```

b. 执行以下命令安装编译工具build和依赖。

```sh
sudo apt install git libusb-1.0-0-dev pkg-config build-essential -y
```

c. 依次执行以下命令在usbboot下进行编译。

```sh
cd usbboot/
make
```

```
pi@raspberrypi:~ $ cd usbboot/
pi@raspberrypi:~/usbboot $ make
cc -Wall -Wextra -g -o bin2c bin2c.c
./bin2c msd/bootcode.bin msd/bootcode.h
./bin2c msd/start.elf msd/start.h
./bin2c msd/bootcode4.bin msd/bootcode4.h
./bin2c msd/start4.elf msd/start4.h
cc -Wall -Wextra -g  -o rpiboot main.c bootfiles.c decode_duid.c `pkg-config --cflags --libs libus
085300\"" -DINSTALL_PREFIX=\"/usr\"
```

d. 执行以下命令到mass-storage-gadget64目录下。

```sh
cd mass-storage-gadget64
```

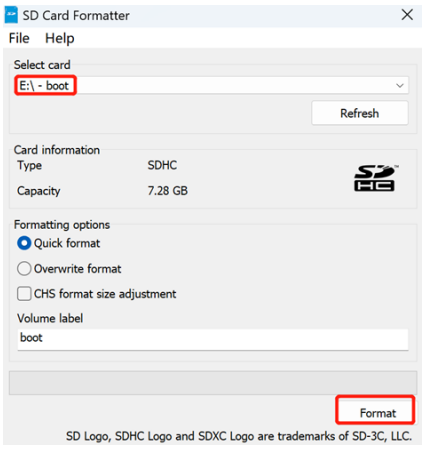
e. 断开设备电源，再按住TRIG按键，对设备重新上电。

f. 在mass-storage-gadget64目录下执行以下命令可开始进行盘符化。

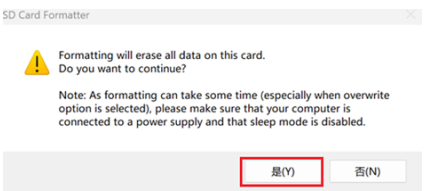
```sh
sudo ../rpiboot -d .
```

```
pi@raspberrypi:~/usbboot $ cd mass-storage-gadget64
pi@raspberrypi:~/usbboot/mass-storage-gadget64 $ sudo ../rpiboot -d .
RPIBOOT: build-date Dec  6 2024 version 20240422~085300 294e74f0
Loading: ./bootfiles.bin
Using ./bootfiles.bin
Waiting for BCM2835/6/7/2711/2712...
Sending bootcode.bin
Successful read 4 bytes
Waiting for BCM2835/6/7/2711/2712...
Second stage boot server
File read: mcb.bin
File read: memsys00.bin
File read: memsys01.bin
File read: memsys02.bin
File read: memsys03.bin
File read: bootmain
Loading: ./config.txt
File read: config.txt
Loading: ./boot.img
File read: boot.img
Second stage boot server done
```

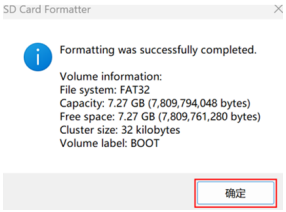
4. Linux PC盘符化成功后设备无需断电，拔出连接到Linux PC的USB烧录线的一端，再将其插入windows PC的USB接口，Windows PC右下角会弹出盘符。
5. 打开SD Card Formatter，选择被格式化的盘符，单击右下方“Format”进行格式化。



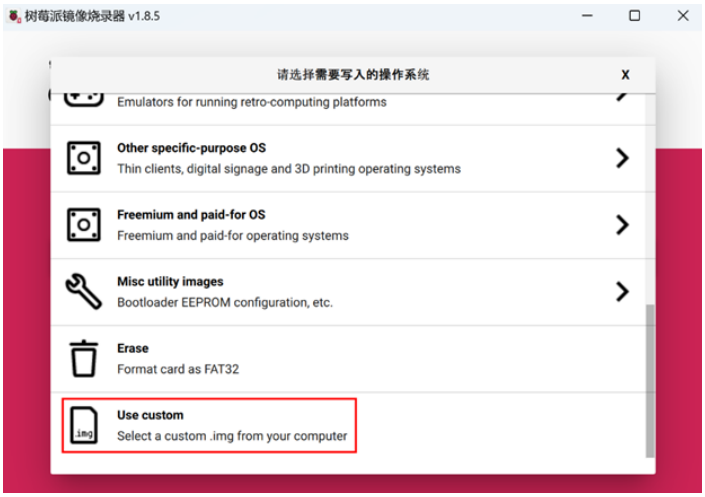
6. 在弹出的提示框中，单击“是”。



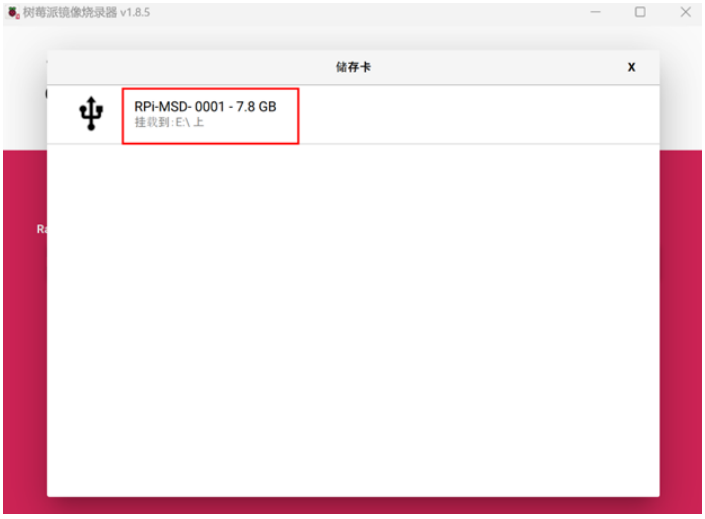
7. 格式化完成后，在提示框中单击 “确定”。



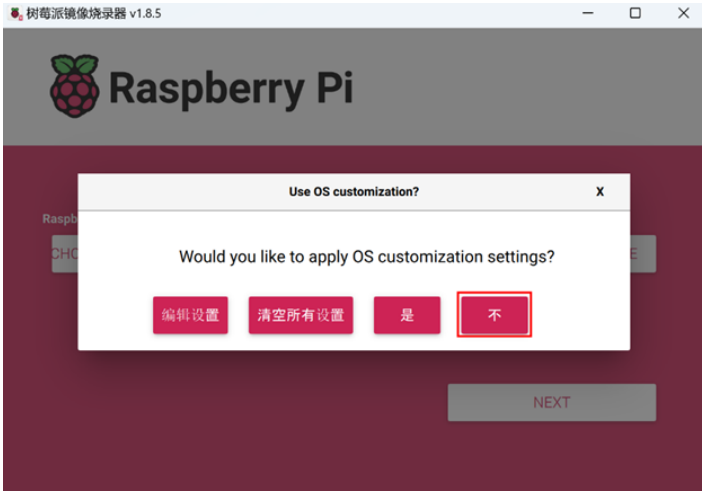
8. 关闭SD Card Formatter。
9. 打开Raspberry Pi Imager，单击“选择操作系统”，在弹出的窗格中选择“Use custom”。



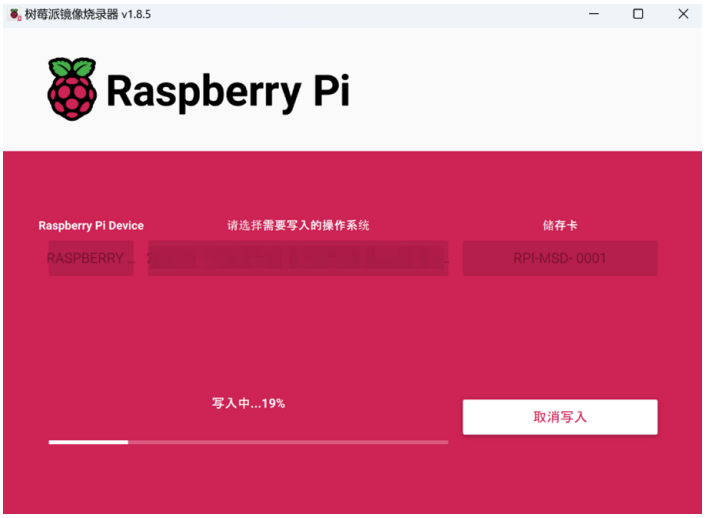
10. 根据提示，在自定义路径下选择已获取的镜像文件，并返回至烧录主界面。
11. 单击“选择SD卡”，在“存储卡”界面选择默认的SD卡，并返回至烧录主界面。



12. 单击“NEXT”，在弹出的“Use OS customization？”提示框中选择“不”，开始写入镜像。



13. 在弹出的“警告”提示框中选择‘”是“，开始写入镜像。



14. 待镜像写入完成后，会进行文件的验证。

15. 验证完成后，弹出“烧录成功”提示框，单击“继续”完成烧录。
16. 关闭Raspberry Pi Imager，取下USB烧录线，重新给设备上电。

5.3 安装firmware包

在ED-AIC3000 设备上烧录标准的Raspberry Pi OS后。需要通过添加edatec apt源和安装firmware包来配置系统，使系统能够正常使用, 下文以Debian 12 (bookworm) 桌面版为例进行说明。

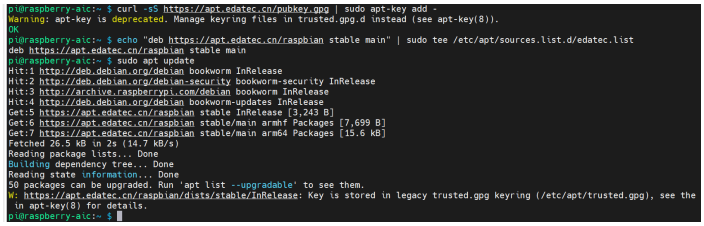
前提条件：

- 已完成Raspberry Pi标准的bookworm镜像的烧录。
- 设备已正常启动，且已完成相关的启动配置。

操作步骤：：

1. 设备正常启动后，在命令窗格依次执行如下命令，添加edatec apt源。

```sh
curl -sS https://apt.edatec.cn/pubkey.gpg | sudo apt-key add -
echo "deb https://apt.edatec.cn/raspbian stable main" | sudo tee /etc/apt/sources.list.d/edate
sudo apt update
```



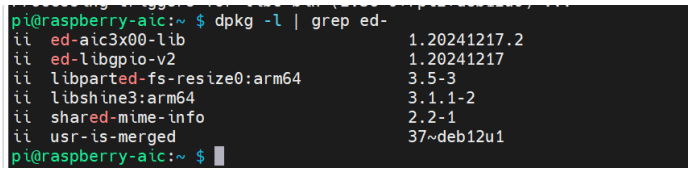
2. 执行如下命令，下载对应的firmware包。

```sh
sudo apt install ed-aic3x00-lib
```

3. 执行如下命令，检查firmware包是否安装成功。

```sh
dpkg -l | grep ed-
```

下图中的结果表示firmware包已安装成功。



2. 安装完成后，执行如下命令重启设备。

```sh
sudo reboot
```

> 提示
>
> 如果安装了错误的firmware包，可以执行 `sudo apt-get --purge remove package` 进行删除，其中package为包的名字。

6 SDK开发指南

本章介绍SDK概述、功能说明和开发示例。

6.1 SDK概述

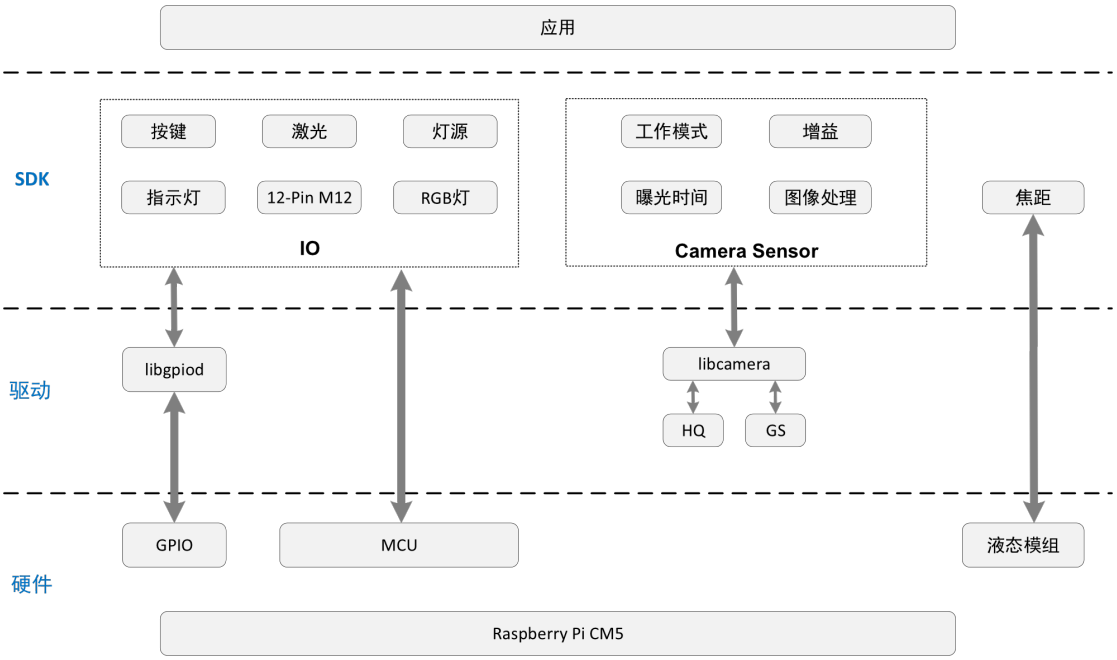
介绍SDK的定义和组成，帮助用户更好的了解SDK。

6.1.1 SDK 简介

ED-AIC3000系列Camera的SDK是一组软件工具开发包（Software Development Kit），给用户提供上层应用所需的接口，便于对Camera进行二次开发。

ED-AIC3000系列Camera的SDK功能包含Trigger/Tune按键的定义、12-Pin M12接口中的DI的定义、激光开关的控制、状态指示灯的控制、报警指示灯的控制、2路DO的控制、RGB灯光、灯源、工作模式、增益、曝光时间、图像处理和自动变焦的控制。

SDK在整个Camera系统中的位置如下图所示。



6.1.2 SDK组成

Camera的SDK是由多个头文件和库文件组成的，具体文件名和安装路径如下表。

功能类型	文件类型	文件名	安装路径
IO控制	头文件	eda-io.h	/usr/include/eda/
	库文件	libeda_io2.so	/usr/lib/
		libedaio2.so	/usr/lib/python3/dist-packages/

功能类型	文件类型	文件名	安装路径
	动态库文件		
Camera Sensor 控制	开源库	picamear2	picamera2用户手册 (https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf)

在开发过程中用户可以根据实际需要实现的功能，参考下文对应的功能代码来完成上层应用的开发。

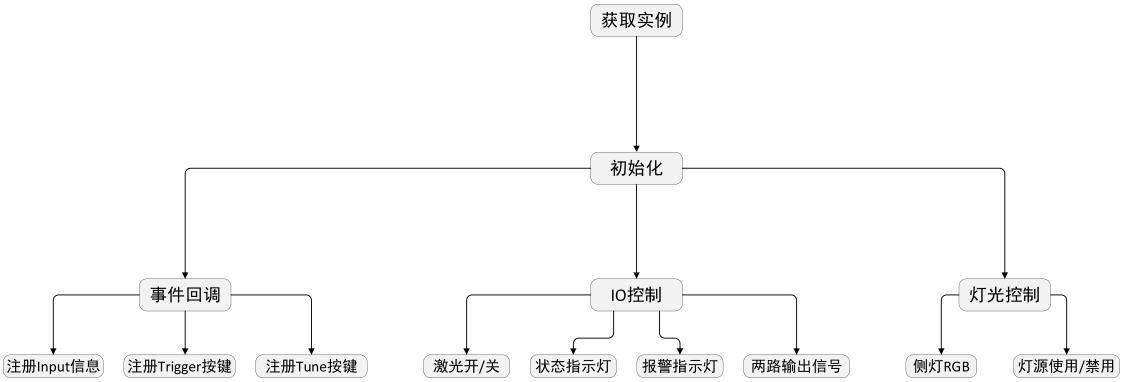
6.2 功能说明

本章介绍各项功能对应代码的编写方法，帮助用户编写上层应用所需要的代码。

6.2.1 I/O控制(C++)

本节介绍指示灯控制、激光控制、事件监听和输出控制等的具体操作。

6.2.1.1 流程图



6.2.1.2 获取实例并初始化

在操作I/O前需要先获取IO实例并对实例进行初始化，操作步骤如下。

1. 获取I/O实例。

```
eda::EdaIo *em = eda::EdaIo::getInstance();
```

2. 对实例进行初始化。

```
em->setup();
```

6.2.1.3 事件回调

I/O控制支持对事件注册回调函数，包含注册Input信息、注册Trigger按键和注册Tune按键。

- DI1触发事件

`em->registerInput(trigger_input);`

12-Pin M12接口中的COMMON_IN引脚接地，DI1引脚接5V触发

- 注册Trigger按键

`em->registerTrigger(trigger_trigger);`

- 注册Tune按键

`em->registerTune(trigger_tune);`

举例：

```c++
#include "eda/eda-io.h"
void trigger_input(int b){
    printf("[Test] Tirgger input: %d\n", b);
}
int main(int argc, char *argv[]){
    eda::EdaIo *em = eda::EdaIo::getInstance();
    em->registerInput(trigger_input);
    em->setup();
    ....
}
```

6.2.1.4 控制I/O状态

通过IO来控制激光的开/关、状态指示灯的点亮/熄灭、报警指示灯的点亮/熄灭和2路输出信号的使能/禁用。

前提条件：

已完成实例的初始化。

操作说明：

- 控制激光

`em->openLaser();`

`em->closeLaser();`

- 控制状态指示灯

`em->setScanStat(true);`

`em->setScanStat(false);`

- 控制警报指示灯

`em->openAlarm();`

```
em->closeAlarm();
```

- 控制2路输出信号

```
em->setDo1High(false);
```

```
em->setDo2High(false);
```

6.2.1.5 控制灯光

Camera侧面灯和区域灯均可独立控制。

前提条件：

已完成实例的初始化。

操作说明：

- 控制侧灯颜色
  `em->setRgbLight(1);`
  - 0: 关闭
  - 1: 红色
  - 2: 绿色
  - 3: 蓝色
  - 4: 黄色
  - 5: 白光
- 控制灯源
  - 使能（默认状态为使能）
  `em->enableLightSection(1);`
  取值范围为1~4，分别对应不同的分区
  - 禁用
  `em->disableLightSection(1);`
  取值范围为1~4，分别对应不同的分区

灯源的使能/禁用不是打开/关闭灯源，灯源与摄像头是联动的，只有当灯源已使能且摄像头打开的条件下灯源才会亮。

- 控制灯源PWM输出
  - 使能灯源PWM控制
  `em->setStrobeManual(1);`
  - 调节亮度
  `em->setBrightnessValue(50);`
  取值范围0~100，100表示最大亮度，0表示最小亮度
  - 打开/关闭灯源
  `em->open_light();`
  `em->close_light();`

6.2.1.6 代码示例

IO控制Class（C++语言）

```c++
typedef void (*IoTrigger)(int level);

class EdaIo{
public:
    static EdaIo* getInstance();
    static void close_io();
    ~EdaIo();
    /**
     * @brief 打开激光
     *
     */
    void openLaser();
    /**
     * @brief 关闭激光
     *
     */
    void closeLaser();
    /**
     * @brief 设置状态指示灯
     *
     * @param good
     */
    void setScanStat(bool good);
    /**
     * @brief 打开alarm 指示灯
     *
     */
    void openAlarm();
    /**
     * @brief 关闭alarm 指示灯
     *
     */
    void closeAlarm();
    /**
     * @brief
     *
     * @param section 1~4
     * @return int
     */
    int enableLightSection(int section);
    /**
     * @brief
     *
     * @param section 1~4
     * @return int
     */
    int disableLightSection(int section);
    /**
     * @brief 设置output1 输出 [高/低]
     *
```

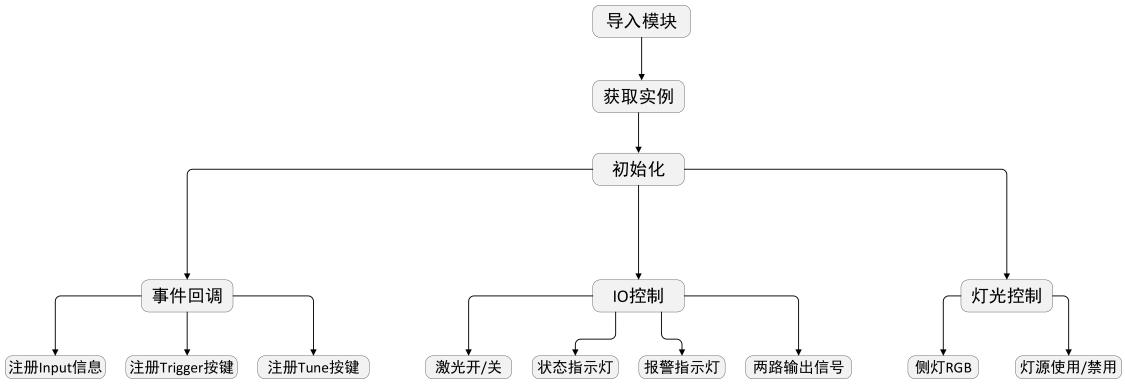
```
     * @param high
     */
    void setDo1High(bool high);
    /**
     * @brief 设置output2 输出 [高/低]
     *
     * @param high
     */
    void setDo2High(bool high);
    // void setAimerColor(RGBColor color);
    /**
     * @brief 注册input触发回调函数
     *
     * @param callback
     */
    void registerInput(IoTrigger callback);
    /**
     * @brief 注册register按键 回调函数
     *
     * @param callback
     */
    void registerTrigger(IoTrigger callback);
    /**
     * @brief 注册Tune按键 回调函数
     *
     * @param callback
     */
    void registerTune(IoTrigger callback);
    /**
     * @brief set RGB light
     *
     * @param light 0: Close; 1: Red; 2: Green; 3: Blue,
     * @return int
     */
    void setRgbLight(uint8_t light);
    /**
     * @brief Set the RGB Light
     *
     * @param r red
     * @param g green
     * @param b blue
     */
    /**
     * @brief 启用亮度功能的手动调节。
     * @param 参数值说明:
     *      - 0 禁用
     *      - 1 启用
     */
    int setStrobeManual(int value);
    /**
     * @brief 设置PWM亮度对象
```

```
     * @param PWM 亮度值说明：
     *           - 100 最大亮度
     *           - 50 默认值
     *           - 0 最小亮度
     */
    int setBrightnessValue(int value);
    /**
     * @brief 自定义光源模式，手动打开光源。
     *          会改变工作模式，自动设置为：5
     * @param timeout 最长打开光源时间毫秒，最大3秒。超过自动关闭。
     * @return int
     */
    int open_light();
    /**
     * @brief 自定义光源模式，手动关闭光源。
     *
     * @return int
     */
    int close_light();
    void setup();
};
```

6.2.2 I/O控制(Python)

本节介绍指示灯控制、激光控制、事件监听和输出控制等的具体操作。

6.2.2.1 流程图



6.2.2.2 导入模块

在操作I/O前需要先导入模块。

```
from libedaio2 import EdaIo,registerInput,registerTrigger,registerTune
```

6.2.2.3 获取实例并初始化

在操作IO前需要先获取IO实例并对实例进行初始化，操作步骤如下。

1. 获取IO实例。

```
eda = EdaIo.singleton();
```

2. 对实例进行初始化。

```
eda.setup();
```

6.2.2.4 事件回调

IO控制支持对事件注册回调函数，包含注册Input信息、注册Trigger按键和注册Tune按键。

- DI1触发事件

```
registerInput(func_input);
```

12-Pin M12接口中的COMMON_IN引脚接地，DI1引脚接5V触发

- 注册Trigger按键

```
registerTrigger(func_trigger);
```

- 注册Tune按键

```
registerTune(func_tune);
```

举例：

```py
#!/usr/bin/python3
from libedaio2 import EdaIo,registerInput
def func_input(v):
    print("[Debug] Trigger: input!", v)
def main() -> int:
    eda = EdaIo.singleton()
    registerInput(func_input)
    eda.setup()

if __name__ == "__main__":
    main()
```

6.2.2.5 控制I/O

通过I/O来控制激光的开/关、状态指示灯的点亮/熄灭、报警指示灯的点亮/熄灭和2路输出信号的使能/禁用。

前提条件：

已完成实例的初始化。

操作说明：

- 控制激光

```
eda.openLaser();
```

```
eda.closeLaser();
```

- 控制状态指示灯

```
eda.setScanStat(True);
```

```
eda.setScanStat(False);
```

- 控制警报指示灯

```
eda.openAlarm();
```

```
eda.closeAlarm();
```

- 控制2路输出信号

```
eda.setDo1High(True);
```

```
eda.setDo2High(False);
```

6.2.2.6 控制灯光

Camera侧面灯和区域灯均可独立控制。

前提条件：

已完成实例的初始化。

操作说明：

- 控制侧灯颜色
  `eda.setRgbLight(1);`
  - 0: 关闭
  - 1: 红色
  - 2: 绿色
  - 3: 蓝色
  - 4: 黄色
  - 5: 白光
- 控制灯源
  - 使能（默认状态为使能）
  `eda.enableLightSection(1);`
  取值范围为1~4，分别对应不同的分区
  - 禁用
  `eda.disableLightSection(1);`

取值范围为1~4，分别对应不同的分区

灯源的使能/禁用不是打开/关闭灯源，灯源与摄像头是联动的，只有当灯源已使能且摄像头打开的条件下灯源才会亮。

- 控制灯源PWM输出
  - 使能灯源PWM控制

  `eda.setStrobeManual(1);`
  - 调节亮度

  `eda.setBrightnessValue(50);`

  取值范围0~100，100表示最大亮度，0表示最小亮度
  - 打开/关闭灯源

  `eda.open_light();`

  `eda.close_light();`

6.2.2.7 代码示例

IO控制（Python3）

```py
from libedaio2 import EdaIo,registerInput,registerTrigger,registerTune

def func_trigger(v):
    print("[Debug] Trigger: trigger button!", v)
...
eda = EdaIo.singleton(); # 获取IO控制实例
registerTrigger(func_trigger); # 注册Trigger 按键回调
# registerInput(func_trigger); # 注册Input输入回调
# registerTune(func_trigger); # 注册Tune 按键回调
eda.setup(); # 初始化
...
eda.openLaser(); # 打开激光
# eda.closeLaser(); # 关闭激光
eda.setScanStat(True); #  设置状态指示灯
eda.openAlarm(); # 打开警告指示灯
# eda.closeAlarm(); # 关闭警告指示灯
eda.setDo1High(True); # 设置第一路输出
eda.setDo2High(False); # 设置第二路输出
eda.setRgbLight(1); # 设置侧灯，0: 关闭; 1: 红色; 2: 绿色; 3: 蓝色 4: 黄色 5: 白色
eda.setStrobeManual(1); # 使能灯源PWM输出
eda.open_light(); # 打开灯源
eda.setBrightnessValue(50); # 设置亮度 取值范围0~100，0:最小亮度，100:最大亮度
# eda.close_light(); # 关闭灯源
```

6.2.3 Camera Sensor控制示例

Camera Sensor控制软件基于开源库 `picamera2` ，官方资料Picamera2 Manual (https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf) ，以下是一些简单的说明和示例。

6.2.3.1 操作步骤

在操作Camera之前，需要先导入IO模块再获取IO实例并初始化（具体操作参见2.2.2 导入模块和2.2.3 获取实例并初始化），再进行如下操作。

1. 导入模块

```
from picamera2 import Picamera2, Preview
```

2. 获取Camera实例

```
picam2 = Picamera2()
```

3. 创建预览配置

```
preview_config = picam2.create_preview_configuration()
```

4. 应用预览配置

```
picam2.configure(preview_config)
```

5. 启动摄像头预览功能，NULL表示不在屏幕上显示画面

```
picam2.start_preview(Preview.NULL)
```

6. 打开摄像头

```
picam2.start()
```

7. (可选) 调节液态模组焦距

```
eda.setLiquid(22000)
```

取值范围0~46000，用户可根据自己的需求来调整。

8. 捕获图片

```
picam2.capture_file("test.jpg")
```

9. 关闭摄像头

```
picam2.close()
```

6.3 示例

本章介绍具体的操作示例，包含编写代码、编译代码和运行代码。

6.3.1 编写代码

下文以实现“打开摄像头等待2s后捕获一张图片”的功能为例，使用Python行编写代码。

编写的内容如下：

```py
from picamera2 import Picamera2, Preview  # 导入 Picamera2 库及预览功能
from libedaio2 import EdaIo,registerInput,registerTrigger,registerTune
import time  # 导入时间模块，用于延时操作

eda = EdaIo.singleton(); # 获取IO控制实例
eda.setup(); # 初始化
picam2 = Picamera2()  # 创建 Picamera2 对象实例
camera_config = picam2.create_preview_configuration()  # 创建相机的预览配置
picam2.configure(camera_config)  # 配置相机
picam2.start_preview(Preview.NULL)  # 启动相机的预览功能（此处选择 NULL 预览，表示没有窗口显示）
picam2.start()  # 开启相机
eda.setLiquid(22000)  # 调节焦距
time.sleep(2)  # 等待 2 秒，确保相机稳定
picam2.capture_file("test.jpg")  # 捕获照片并保存到当前目录，文件名为 "test.jpg"
picam2.close()  # 关闭相机，释放资源
```

编写完成后，保存为`test123.py`文件。

> 提示
>
> 文件名自定义即可。

6.3.2 运行代码

Python代码编写完成后需要登录Camera设备，在Raspberry Pi系统上运行。

前提条件：

- 已完成Camera的硬件部分的连线，具体的操作请参见启动设备。
- 已将Camera上电并正常接入网络。
- 已获取Camera IP地址，并成功登录Camera系统。

操作步骤：

在Camera系统上创建一个文件夹，将章节3.1 编写代码中编写的代码文件上传至文件夹中。

执行`ls`命令，查看文件夹中的文件，确保代码文件已上传成功。

1. 执行如下命令，运行代码。

```sh
sudo python test123.py
```

- `test123.py`:表示章节3.1 编写代码中编写的代码文件。

> 提示

> 运行成功后，可在代码文件所在的目录下查看到 `test.jpg` 文件。