



ED-AIC3000

User Manual

by EDA Technology Co., Ltd

built: 2026-05-21

1 Hardware Manual

This chapter introduces the product overview, packing list, appearance, indicator and interface.

1.1 Overview

The ED-AIC3000 is an industrial smart camera based on the Raspberry Pi CM5 core module, featuring 12-megapixel imaging capability and a sampling rate of up to 60 FPS. To accommodate different application scenarios and requirements, the device supports flexible storage configurations:

- RAM options include 2GB, 4GB, and 8GB
- eMMC flash storage options include 16GB, 32GB, and 64GB

The ED-AIC3000 integrates a modular lighting system and is equipped with an M12 fixed-focus lens (with liquid lens zoom capability). It supports both bright-field and dark-field illumination modes, enabling optimal lighting effects on normal, etched, high-gloss, and textured surfaces to enhance imaging contrast and defect detection.

The ED-AIC3000 provides power interfaces, I/O interfaces, an RS232 serial port, and a Gigabit Ethernet port. All interfaces utilize rugged M12 aviation connectors, supporting Ethernet connectivity and real-time data transmission. It is primarily suited for demanding applications such as machine vision inspection, AI visual analysis, industrial automation, and quality control.

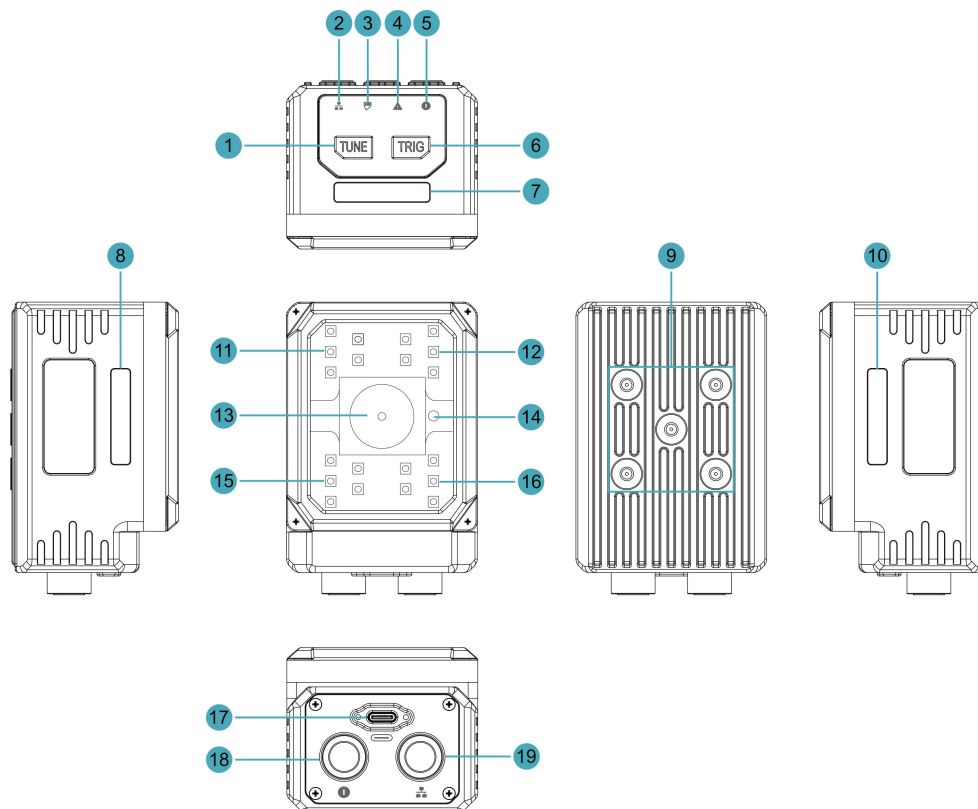


1.2 Packing List

1 x ED-AIC3000 Unit

1.3 Appearance

Introduce the functions and definitions of product interfaces, buttons and indicators.



NO.	Function Definition
1	1 x TUNE button, one-touch autofocus button or user-defined button.
2	1 x Network connection indicator, which is used to check the status of the network connection.
3	1 x working status indicator, which is used to check the working status of the device.
4	1 x System fault indicator, which is used to check if a system fault occurs.
5	1 x Power indicator, which is used to check the power-on status of the device.
6	1 x TRIG button, using to camera triggering or user-defined one-touch button.
7, 8, 10	3 x RGB lights (1 set), lights can be set to red, green, blue, yellow and white, user customizable.
9	5 x M4 screw holes for bracket mounting.
11, 12, 15, 16	4 x light sources, using for supplementary lighting when the device is working.
13	1 x M12 Lens, using to take photos.
14	1 x laser light, red circular spot, for image positioning.
17	1 x USB type-c port, using to flash to eMMC.
18	1 x power interface, including power input port, I/O ports and RS232 serial port, using M12 12-Pin aviation connector.
19	1 x communication interface, Gigabit Ethernet interface, using M12 8-Pin A-code aviation connector for access to Ethernet.

1.4 Button

The ED-AIC3000 device include 2 buttons, an TUNE button and a TRIG button.

- The TUNE button has "TUNE" printed on the case. Pressing the button can realize one-touch automatic focus, and it supports that users to customize its functions.
- The TRIG button has "TRIG" printed on the case. Pressing the button can trigger the camera, and it support that users to customize its functions.

Pin Definition

The button pins are defined as follows:

Button	CM5 Pin
TUNE button	GPIO20
TRIG button	GPIO12

1.5 Indicator

This section describes the various states and meanings of the indicators on the ED-AIC3000 device.

Indicator	Status	Description
Network connection indicator	Yellow solid	The Ethernet has been connected normally
	Off	No Ethernet connection
Working status indicator	Green solid (controlled and set to ON via SDK)	The system is working normally
	Red solid (controlled and set to ON via SDK)	System working status is abnormal
System fault indicator	On	System failure
	Off	The system has not failed
Power indicator	Red solid	The device is powered on
	Off	The device is powered off

Pin Definition

Indicator	CM5 Pin
Power indicator	N/A
System fault indicator	GPIO21

Indicator	CM5 Pin
Working status indicator	GPIO7 (abnormal) GPIO16 (normal)
Network connection indicator	N/A

1.6 Interface

This section describes the definition and function of each interface in the product.

1.6.1 Power & I/O Interface

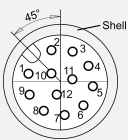
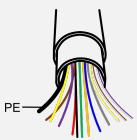
The ED-AIC3000 device feature a Power & I/O Interface with a 12-pin M12 aviation connector. This interface integrates:

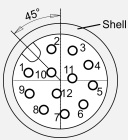
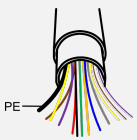
- 1 power input
- 1 serial port (RS232)
- 1 digital input (DI)
- 2 digital outputs (DOs)

The Power & I/O cable connects to this interface:

- One end is terminated with an M12 connector for camera integration.
- The other end consists of bare wires for connecting to power, DI, DO, and RS232 devices.

The pinout definitions for the M12 connector and the corresponding bare wire assignments are detailed in the following table:

M12 Pin	Bare Wire Color	Definition
		
1	Yellow 	DC-
2	White/Yellow 	DC+
3	Brown 	COMMON_IN
4	White/Brown 	DI_1
5	Purple 	Trigger
6	White/Purple 	COMMON_OUT
7	Red 	External Strobe
8	Black 	DO_1
9	Green 	DO_2

M12 Pin	Bare Wire Color	Definition
		
10	Orange 	RS232_GND
11	Blue 	RS232_TX
12	Gray 	RS232_RX
Shell	Black (Thick) 	PE

One DI (Digital Input) and 2 DO (Digital Output) channels correspond to the CM5's GPIO pins, as detailed in the following table:

Signal	CM5 Pin
DI1	GPIO17
DO1	GPIO22
DO2	GPIO27

TIP

- The thickest black bare wire is PE.
- For Raspberry Pi HQ Camera, Pins 5 and 7 of the M12 connector are undefined.
- During installation, ensure that wire colors align with pin definitions and properly connect the PE to the ground terminal to guarantee safety and signal integrity.
- If there are unused bare wires during wiring, cut off the exposed metal ends or insulate them with electrical tape.

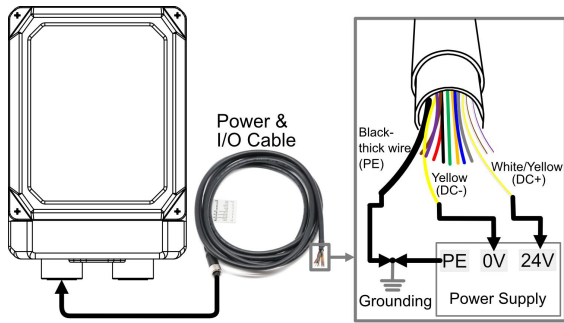
WARNING

Strictly follow the wiring instructions below to connect the power & I/O cable; incorrect wiring may result in device damage.

1.6.1.1 Power Connection

For the power & I/O cable:

- White/Yellow and Yellow wires connect to the positive (+) and negative (-) terminals of the external power supply, respectively.
- The thick black wire (PE) connects to the ground terminal.



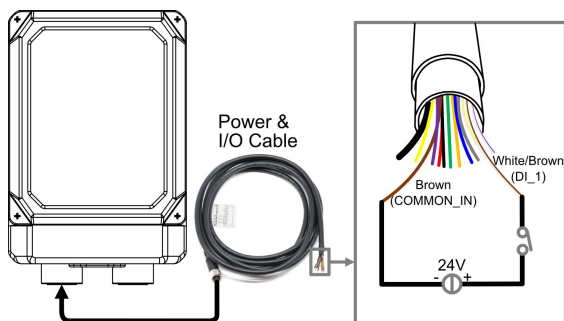
WARNING

- Device Power Supply: DC24V ($\pm 10\%$). It is recommended to use a DC 24V 2A power adapter.
- When connecting the power cable, connect the PE in the bare wires and the external power supply's PE to ground terminal to ensure proper grounding.

1.6.1.2 DI Connection

For the power & I/O cable:

- White/Brown is DI Channel 1.
- Brown is the DI common terminal (COMMON_IN).
- Supports NPN-type or PNP-type sensor.



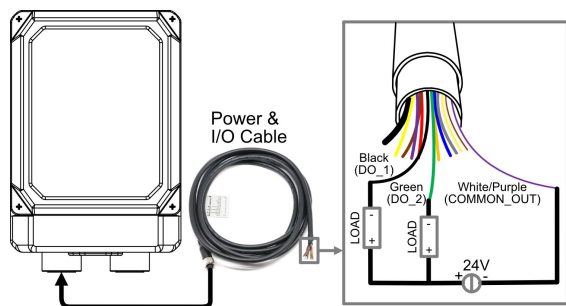
1.6.1.3 DO Connection

For the power & I/O cable:

- Black and green wires correspond to the 2 DO channels.
- White/purple wire serves as the common terminal for the DO channels.

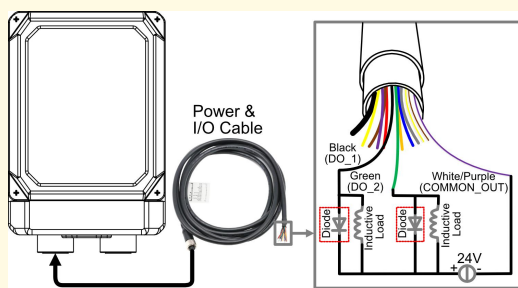
DO Specifications:

- Single-channel load: 0.25A (max).
- Total current for 2 consecutive channels: 0.5A (max).
- Supported load types: Resistive loads, lamp loads, and inductive loads.



WARNING

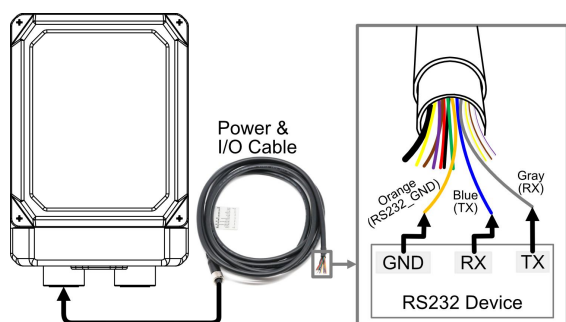
- Do not connect the white/purple wire (COMMON_OUT) to the 24V positive terminal.
- If an inductive load is connected to the DO channel, it is recommended to add a Diode in the circuit (as shown in the figure below) for protection. Select an appropriate Diode based on the specifications of the inductive load.



1.6.1.4 RS232 Connection

For the power & I/O cable:

- Orange: RS232_GND
- Blue: RS232_TX
- Gray: RS232_RX
- Compatible with RS232 devices.

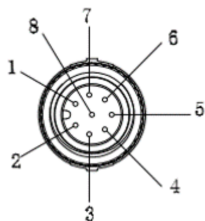


1.6.2 Communication interface

ED-AIC3000 device include 1 communication interface with M12 8-Pin aviation connector.

Pin Definition

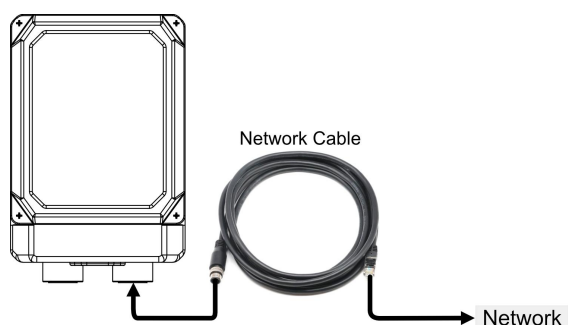
The pin definitions are as follows:



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

Cable Connection

The communication interface is used to connect an Ethernet cable, enabling the device to access the network. One end of the cable is terminated with an M12 connector for connecting to the Camera, while the other end features an RJ45 connector for network integration.

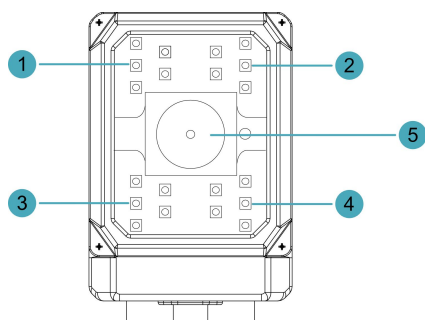


1.6.3 USB-C interface

ED-AIC3000 include a USB Type-C interface, which supports flashing to eMMC by connecting to a PC.

1.7 Light source and lens

The ED-AIC3000 series equipment consists of four light sources and a lens.



NO.	Description
1	Light source part 1 (TopLeft), supports individual enabling and disabling.
2	Light source part 2 (TopRight), supports individual enabling and disabling.
3	Light source part 3 (BottomLeft), supports individual enabling and disabling.
4	Light source part 4 (BottomRight), supports individual enabling and disabling.
5	Lens, M12 fixed focal length lens with liquid module zoom.

1.8 RGB Lights

ED-AIC3000 device contain 3 RGB lights, 3 RGB lights as a group. It supports to set the lights as red, green, blue, yellow or white through software, the default is off, users can customize according to the actual needs.

TIP

3 RGB lights as a group, only support simultaneous setting.

2 Installing Device

This chapter introduces how to install the device.

Preparation :

- One M4 and one M6 allen head screwdrivers have been prepared.
- The mounting bracket and mounting screws (3xM4*8 with washers, 1xM6*10 with washers) have been prepared as shown below.



TIP

Mounting bracket and mounting screws are optional and need to be purchased separately.

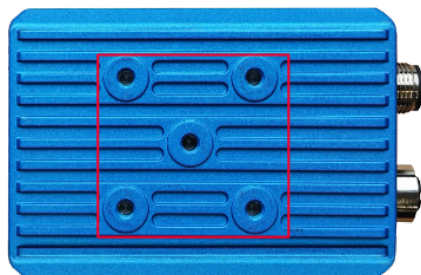
Steps :

TIP

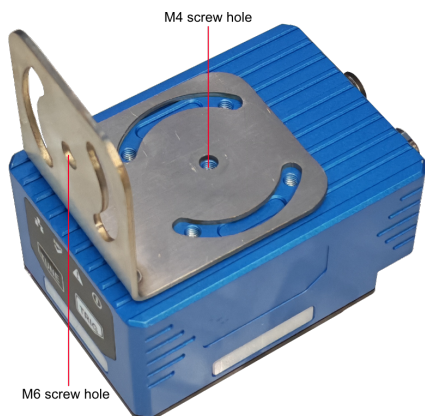
When using M4 and M6 screws, refer to the following diagram to place spring washer and flat washer.



1. Determine the location of the installation hole on the device, as shown in the red frame in the figure below.



2. Place the Mounting bracket above the camera installation hole, so that the bracket (the side with M4 screw hole) is aligned with the central screw hole of the camera.



3. Insert 1 M4 screw with washer at the location of the center screw hole, then use an M4 hexagonal screwdriver to tighten the M4 screw clockwise to secure the bracket to the device.



4. (Optional) Rotate the Mounting bracket to adjust the mounting direction as required.
5. Insert 1 M4 screw with washer at the location of the peripheral screw hole, then tighten the M4 screw clockwise using an M4 hexagonal screwdriver.



TIP

It is recommended to use a center screw and one peripheral screw for fixing.

4. Use M6 screws to fix the Mounting bracket and other devices.

TIP

Adjust the appropriate installation position and angle according to different project requirements on site.

It is recommended to use a center screw and one peripheral screw for fixing.

3 Booting the Device

This chapter introduces how to connect cables and boot the device.

3.1 Connecting Cables

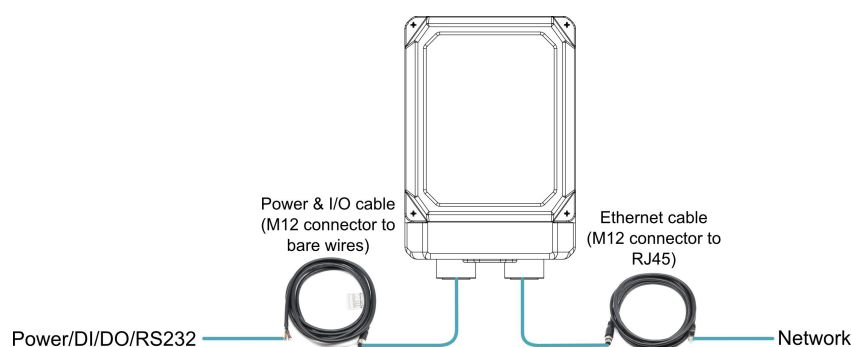
This section describes how to connect cables.

Preparation:

- The Network Cable (M12 connector to RJ45) and Power & I/O Cable (M12 connector to bare wires) to be connected have been obtained.
- A DC 24V 2A power adapter and auxiliary wiring connector have been prepared.
- Prepare the connectors and wiring tools required for the Power & I/O Cable as needed.

Schematic diagram of connecting cables:

For the pin definitions and wiring methods of each interface, see [1.6 Interface](#).



TIP

The wire colors may vary across different batches of network cables.

3.2 Booting The System For The First Time

The ED-AIC3000 device has no power switch. After the power is connected, the system will start.

- The power indicator is always red solid, indicating that the device is powered normally.
- The working status indicator light is green solid (controlled and set to ON via SDK), indicating that the system starts normally.

After the system starts, the default username and password are used for login. Because the camera cannot be connected to the monitor, it is necessary to log in to the system remotely through the PC.

The factory image of ED-AIC3000 device enables VNC by default, and sets the device IP to a static IP: 192.168.1.10. Users can remotely connect to the device through VNC and enter the device's desktop system. For specific operations, refer to [Connect to the device desktop via vnc](#).

TIP

Default username: pi; default password: raspberry.

4 Installing OS (optional)

The device is shipped with an operating system by default. If the OS is corrupted during use or the user needs to replace the OS, it is necessary to re-download the appropriate system image and install it. Our company supports to install the OS by installing the standard Raspberry Pi OS first, and then install the Firmware package.

The following section describes the specific operations of image download, eMMC flashing and installation of Firmware packages.

4.1 Downloading OS File

It is recommended to download the official system image of Raspberry Pi OS (Desktop) 64-bit. The download path is provided in the table below:

OS	Download Path
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)

TIP

Our engineers are currently adapting and developing firmware packages for Raspberry Pi OS-trixie (Debian 13), so it is temporarily not supported. We recommend using the Raspberry Pi OS 64-bit-bookworm (Debian 12) version of the operating system.

4.2 Flashing to eMMC

It is recommended to use the Raspberry Pi official tools. The download paths are as follows:

- 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/>)
- Rpiboot: https://github.com/raspberrypi/usbboot/raw/master/win32/rpiboot_setup.exe (https://github.com/raspberrypi/usbboot/raw/master/win32/rpiboot_setup.exe)

Preparation:

- A Windows PC has been acquired and the downloading and installation of the official tools to the PC have been completed.
- A USB-C to USB-A flashing cable has been prepared.
- An 8-Pin M12 male to RJ45 network cable has been prepared.
- A 12-pin M12 female to bare wire power I/O cable has been prepared.

- The OS file has been obtained.

Steps:

The steps are described using Windows system as an example.

1. Connect the power cord and USB flashing cable (USB-C to USB-A).

- Connect the flashing cable: one end is connected to the USB type-C port on the device side (the interface in the red box in the figure below), and the other end is connected to the USB port on the Linux PC.



- Connect the power cable: one end is connected to the M12 12-Pin power interface on the device side (the interface in the red box in the figure below), and the other end is connected to the external power supply.



2. Disconnect the device power supply, then press and hold the TRIG button, and power on the device again. The device will automatically enter the flashing mode.
3. Open the installed `rpi-mass-storage-gadget64.bat` tool (as shown in the red box in the picture below) to automate the disk tokenization.

TIP

The `rpi-mass-storage-gadget64.bat` tool is located in the installation directory of Rpiboot.

 cygwin1.dll

2024/9/26 23:06

 rpiboot.exe

2024/11/13 17:49

 rpi-mass-storage-gadget64.bat

2024/11/23 2:05

 Uninstall.exe

2025/2/10 19:01

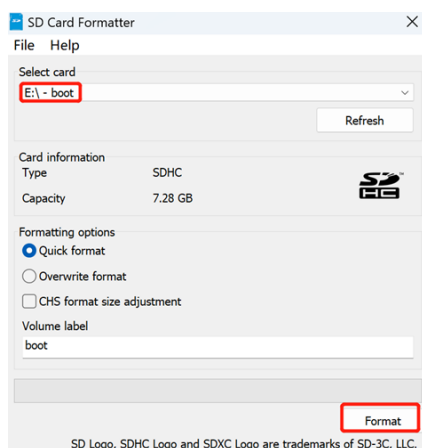
```

C:\WINDOWS\system32\cmd
USB mass storage gadget for Raspberry Pi 5
RPiBOOT: build-date Nov 13 2024 version 20240422-085300 e3e0fa29
Loading: mass-storage-gadget64/bootfiles.bin
Using mass-storage-gadget64/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: mass-storage-gadget64/config.txt
File read: config.txt
Loading: mass-storage-gadget64/boot.img
File read: boot.img
Second stage boot server done

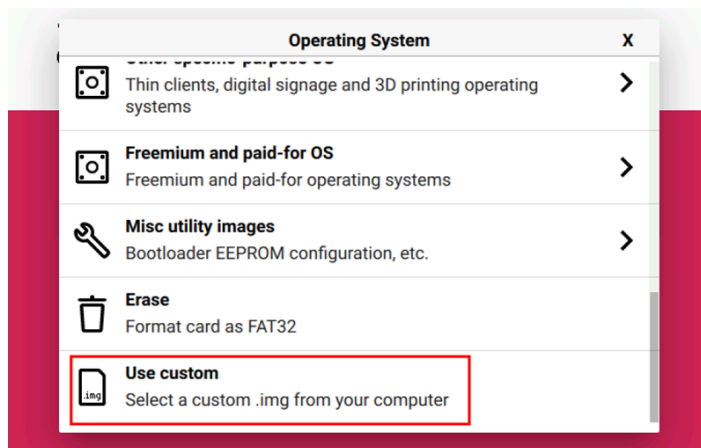
Raspberry Pi Mass Storage Gadget started
EMMC/MMIO devices should be visible in the Raspberry Pi Imager in a few seconds.
For debug, you can login to the device using the USB serial gadget - see COM ports in Device Manager.
Press a key to close this window.

```

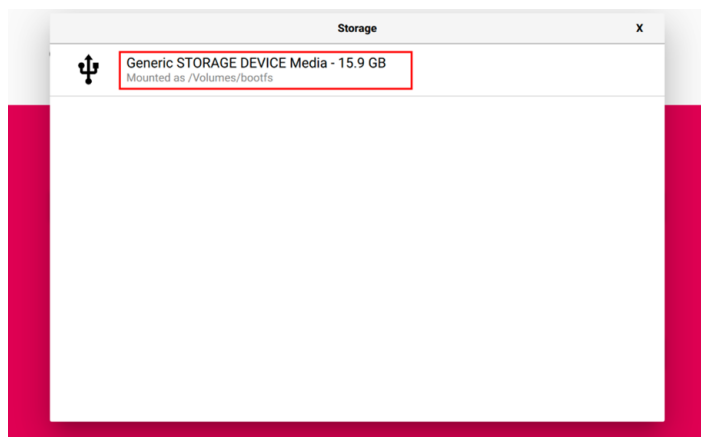
- After the disk symbolization is completed, close the `rpi-mass-storage-gadget64.bat` tool, the disk symbol will pop up in the lower right corner of the computer.
- Open `SD Card Formatter`, select the formatted drive letter, and click "Format" at the lower right to format.



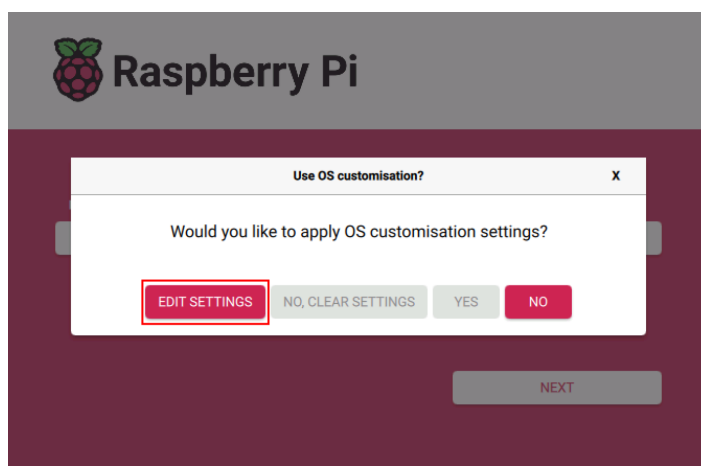
- In the pop-up prompt box, select "Yes".
- When the formatting is complete, click "OK" in the prompt box.
- Close `SD Card Formatter`.
- Open `Raspberry Pi Imager`, select "CHOOSE OS" and select "Use Custom" in the pop-up pane.



10. According to the prompt, select the OS file under the user-defined path and return to the main page.
11. Click "CHOOSE STORAGE", select the default device in the "Storage" interface, and return to the main page.



12. Click "NEXT", and in the "Use OS customisation?" prompt, select "EDIT SETTINGS".



13. On the "OS customisation" interface, under the "General" tab, customize the Hostname and Username & Password.

OS Customisation

General Services Options

☒ Set hostname: .local ①

☒ Set username and password

Username: ②

Password:

☐ Configure wireless LAN

SSID:

Password:

☐ Show password ☐ Hidden SSID

- Hostname: Customize a name for easy identification when locating the device's IP address in the router later.
- Username & Password: Customize credentials to enable automatic login after the device boots.

14. Under the "Services" tab in the "OS customisation" interface, enable SSH, then click "SAVE".

OS Customisation

General Services Options

☒ Enable SSH ①

☒ Use password authentication

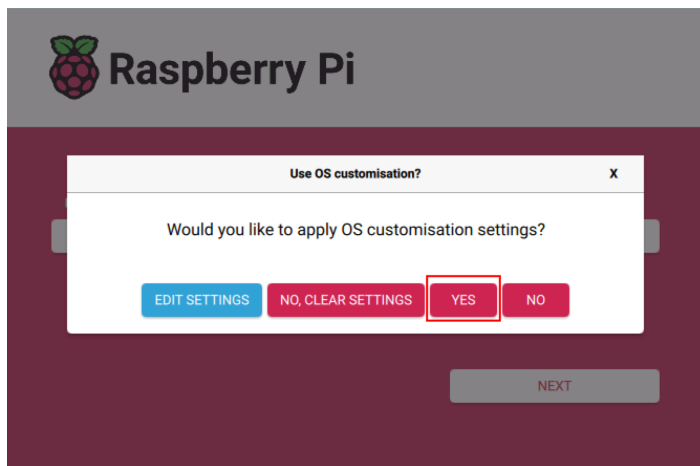
☐ Allow public-key authentication only

Set authorized_keys for 'pi':

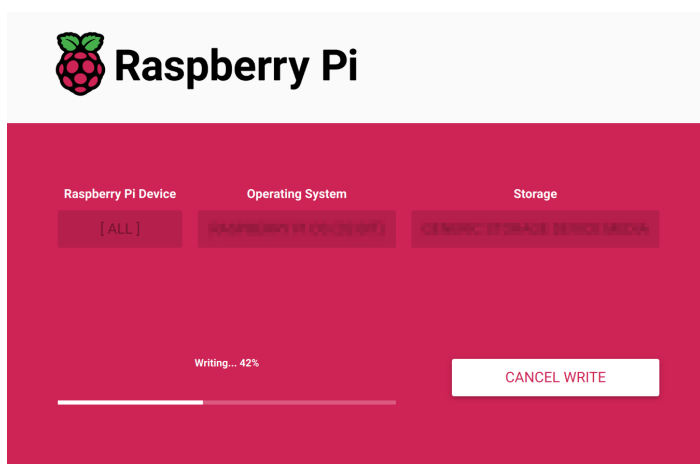
②

- Enable SSH allows remote login to the device.

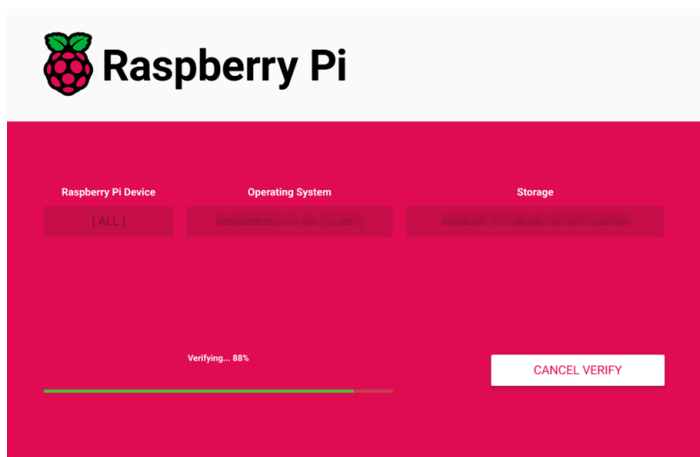
15. In the "OS customisation?" prompt, select "YES".



16. Select “YES” in the pop-up “Warning” pane to start writing the image.



17. After the OS writing is completed, the file will be verified.



18. After the verification is completed, click “CONTINUE” in the pop-up “Write Successful” box.

19. Close **Raspberry Pi Imager** , remove USB cable and power on the device again.

4.3 Connecting to Device and Installing Firmware Package

Since the ED-AIC3000 device does not have an HDMI interface, you need to obtain the device's IP address before installing the Firmware package, and then connect to the device via SSH.

4.3.1 Finding Device IP

After the device has booted up successfully, it is recommended to check the device IP address by logging into your router.

Preparation:

- The device is connected to a network via a router.
- You have the IP address (e.g., 192.168.X.X) and the administrator password for the router in your network.

Steps:

1. Open a browser, type the IP address of your router (e.g., 192.168.X.X) into the address bar, and press **Enter** to access the router login interface.
2. Enter the router's administrator password as prompted to log into the router's management interface.
3. Locate the device's IP address in the list of client devices within the management interface, typically identifiable by its hostname.

4.3.2 Connecting To The Device Via SSH

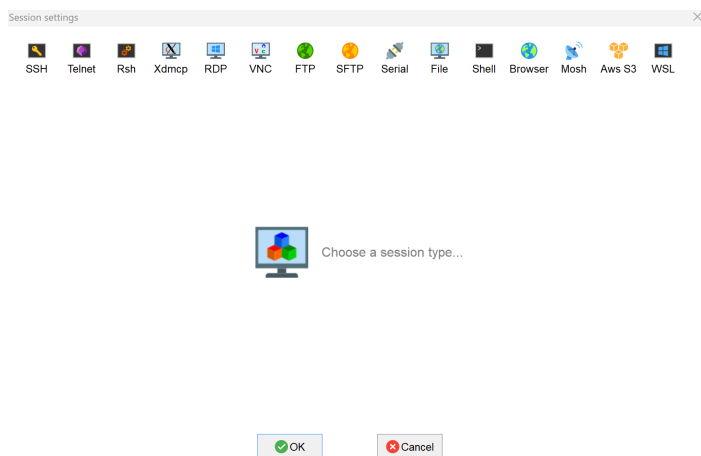
After the device boots up successfully, you can choose to remotely connect to it via SSH for configuration or debugging. You may select your preferred remote login tool. The following instructions use **MobaXterm** as an example.

Preparation:

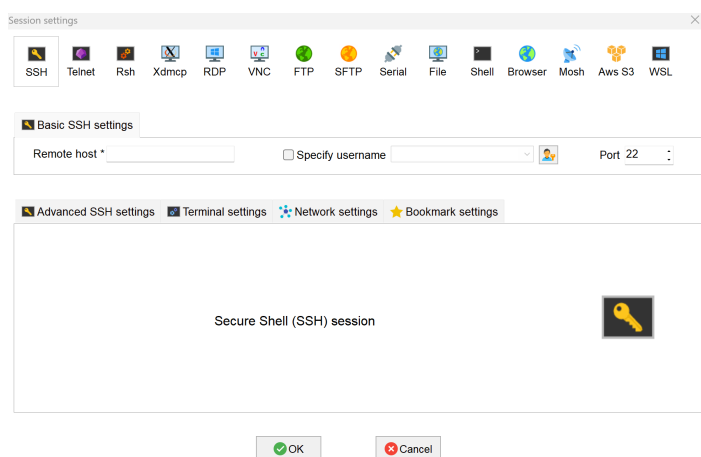
- **MobaXterm** is installed on your PC.
- The device is connected to the network via a router.
- The device's IP address has been obtained.

Steps:

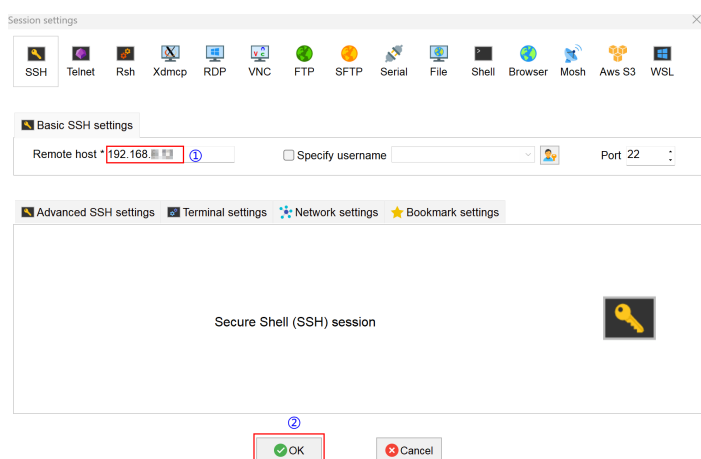
1.  Open **MobaXterm**, click the **Session** icon to open the session creation window, as shown below.



2.  Click the **SSH** icon in the upper left corner to open the SSH session interface.



3. Enter the obtained IP address of the device, then click "OK".



4. Click "Accept" in the pop-up prompt box to enter the system login interface.
5. Enter the username and password as prompted to complete the login and access the system.

TIP

The username and password are those set in the [Raspberry Pi Imager](#) before flashing to eMMC.

```
login as: 
• MobaXterm Personal Edition v23.0 •
(SSH client, X server and network tools)

> SSH session to pi@192.168.1.100
• Direct SSH      : ✓
• SSH compression : ✓
• SSH-browser     : ✓
• X11-forwarding  : ✓ (remote display is forwarded through SSH)
> For more info, ctrl+click on help or visit our website.

Linux raspberrypi 6.12.47+rpt-rpi-v7 #1 SMP Raspbian 1:6.12.47-1+rpt1 (2025-09-16) armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 26 17:13:53 2025 from 192.168.1.100

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~$
```

4.3.3 Installing Firmware Package

After installing the standard Raspberry Pi OS on the ED-AIC3000 device. You need to configure the system by adding edatec apt source and installing firmware packages to make the system work properly, the following is an example of Debian 12 (bookworm) desktop version.

Preparation:

- A Raspberry Pi standard bookworm image has been flashed.
- The device has been booted normally and the relevant boot configuration has been completed.

Steps:

1. After the device has booted normally, execute the following commands sequentially in the command pane to add the [edatec apt](#) source.

```
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/edatec
sudo apt update
```

```

pi@raspberrypi:~$ curl -s https://apt.edatec.cn/pubkey.gpg | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
pi@raspberrypi:~$ echo "deb https://apt.edatec.cn/raspbian stable main" | sudo tee /etc/apt/sources.list.d/edatec.list
deb https://apt.edatec.cn/raspbian stable main
pi@raspberrypi:~$ sudo apt update
Hit:1 http://deb.debian.org/debian bookworm InRelease
Hit:2 http://deb.debian.org/debian-security bookworm-security InRelease
Hit:3 http://archive.raspberrypi.com/debian bookworm InRelease
Hit:4 http://deb.debian.org/debian bookworm-updates InRelease
Get:5 https://apt.edatec.cn/raspbian stable InRelease [3,243 B]
Get:6 https://apt.edatec.cn/raspbian stable/main armhf Packages [7,699 B]
Get:7 https://apt.edatec.cn/raspbian stable/main arm64 Packages [15.6 kB]
Fetched 26.5 kB in 2s (14.7 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
50 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: https://apt.edatec.cn/raspbian/dists/stable/InRelease: Key is stored in legacy trusted.gpg keyring (/etc/apt/trusted.gpg), see the
in apt-key(8) for details.

```

2. Execute the following command to installing the corresponding firmware package.

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

sh

```

pi@raspberrypi:~$ sudo apt install ed-aic3x00-lib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ed-aic3x00-lib
0 upgraded, 1 newly installed, 0 to remove and 286 not upgraded.
Need to get 285 kB of archives.
After this operation, 1,278 kB of additional disk space will be used.
Get:1 https://apt.edatec.cn/raspbian stable/main arm64 ed-aic3x00-lib arm64 1.20250218.1 [285 kB]
Fetched 285 kB in 0s (794 kB/s)
Selecting previously unselected package ed-aic3x00-lib.
(Reading database ... 130917 files and directories currently installed.)
Preparing to unpack .../ed-aic3x00-lib_1.20250218.1_arm64.deb ...
Unpacking ed-aic3x00-lib (1.20250218.1) ...
Setting up ed-aic3x00-lib (1.20250218.1) ...
Processing triggers for libc-bin (2.36-9+rpt2+deb12u7) ...

```

3. After the installation is complete, execute the following command to reboot the device.

```
sudo reboot
```

sh

4. Execute the following command to check whether the firmware package is installed successfully.

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

sh

The result in the following figure indicates that the firmware package has been installed successfully.

```

pi@raspberrypi:~$ dpkg -l | grep ed-
ii  ed-aic3x00-lib          1.20241217.2
ii  ed-libgpt-to-v2        1.20241217
ii  libparted-fs-resize0:arm64 3.5-3
ii  libshim3:arm64         3.1.1-2
ii  shared-mime-info        2.2-1
ii  usr-is-merged           37~deb12u1
pi@raspberrypi:~$

```

TIP

If the wrong firmware package is installed, you can remove it by running `sudo apt-get --purge remove package`, where package is the name of the package.

5 SDK Development Guide

This chapter introduces the SDK overview, functional description, and development examples.

5.1 SDK Overview

Introducing the definition and composition of SDK to help users understand the SDK better.

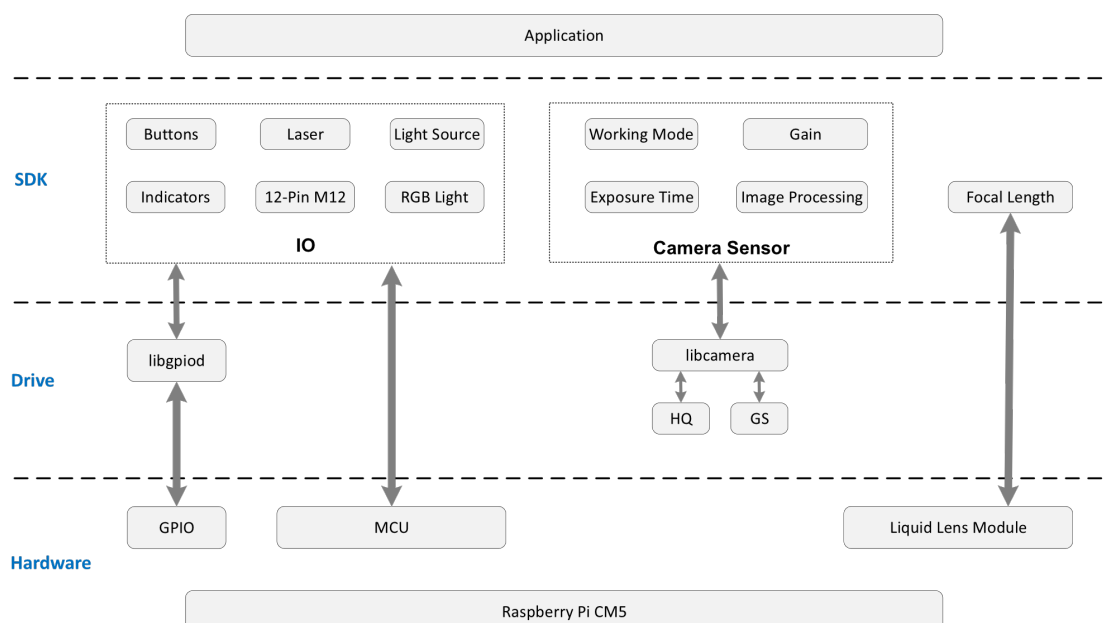
5.1.1 SDK Introduction

The SDK of ED-AIC3000 Camera is a set of Software Development Kit (SDK), which provides users with the interfaces required by the upper layer applications, and facilitates the secondary development of the Camera.

The SDK functions of the ED-AIC3000 Camera include:

- Definition of the Trigger/Tune buttons;
- Definition of DI (Digital Input) in the 12-Pin M12 connector;
- Control of the laser switch;
- Control of the status indicator;
- Control of the alarm indicator;
- 2-channel DO (Digital Output) control;
- RGB lighting;
- Light source control;
- Configuration of operating modes;
- Adjustment of gain and exposure time;
- Image processing;
- Automatic zoom control.

The location of SDK in the camera system is shown in the figure below.



5.1.2 SDK Composition

The SDK of camera is composed of multiple header files and library files. The details file names and installation paths are as follows.

Function	Definition	Filename	Installation Path
I/O control	header file	eda-io.h	/usr/include/eda/
	library file	libeda_io2.so	/usr/lib/
	Dynamic library files	libedaio2.so	/usr/lib/python3/dist-packages/
Camera Sensor Control	Open source library	picamera2	picamera2 User Manual (https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf)

During the development process, users can complete the development of upper-layer applications based on actual needs and refer to the corresponding function description below.

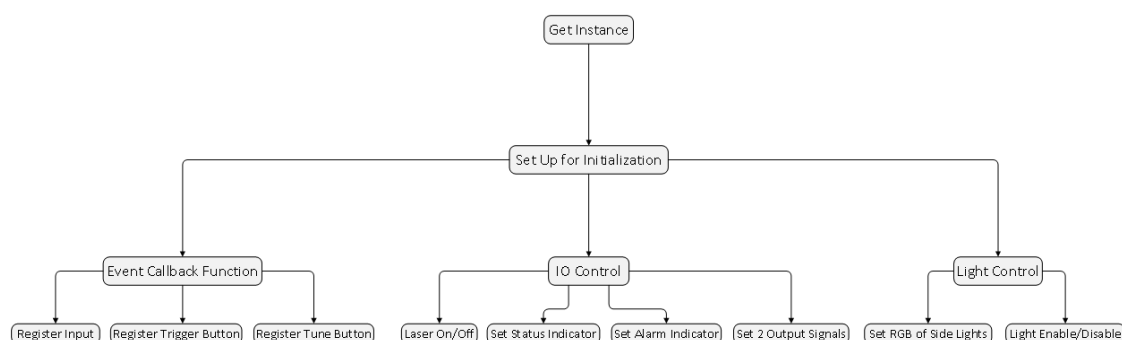
5.2 Function Description

This chapter introduces how to write the code corresponding to each function to help users write the code required for upper-layer applications.

5.2.1 I/O Control(C++)

This section describes the specific operations of indicator control, laser control, event listening, and output control.

5.2.1.1 Flow Diagram



5.2.1.2 Getting Instance and Initializing

Before operating I/O, you need to obtain an I/O instance and initialize the instance. The steps are as follows.

1. Getting an I/O instance.

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

2. Initializing the instance.

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

5.2.1.3 Event Callback Function

I/O control supports registering callback functions for events, including registering Input, registering Trigger button, and registering Tune button.

- DI 1 trigger event

```
em->registerInput(trigger_input);
```

The COMMON_IN pin in the 12-Pin M 12 interface is connected to ground signal, and the DI1 pin is connected to 5V signal for triggering.

- Registering Trigger button

```
em->registerTrigger(trigger_trigger);
```

- Registering Tune button

```
em->registerTune(trigger_tune);
```

Sample:

```
#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();
    ....
}
```

C++

5.2.1.4 Controlling I/O

The I/O is used to control the laser on/off, status indicator on/off, alarm indicator on/off and 2 output signals enable/disable.

Preparation

Initialization of the instance has been completed.

Operating Instructions

- Laser On/Off


```
em->openLaser();
```

```
em->closeLaser();
```

- Status indicator On/Off

```
em->setScanStat(true);
```

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

- Alarm indicator On/Off

```
em->openAlarm();
```

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

- 2 outputs enable/disable

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

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

5.2.1.5 Controlling light

Both the camera side lights and area lights can be controled independently.

Preparation

Initialization of the instance has been completed.

Operating Instructions

- Side light color

```
em->setRgbLight(1);
```

- 0: Closing side light
- 1: Setting the color to Red
- 2: Setting the color to Green
- 3: Setting the color to Blue
- 4: Setting the color to Yellow
- 5: Setting the color to White

- Area lights

- Enable (The default state)

```
em->enableLightSection(1);
```

The value range is 1~4, corresponding to different partitions.

- Disable

```
em->disableLightSection(1);
```

The value range is 1~4, corresponding to different partitions.

Enabling/disabling the area light does not turn on/off the light. The area light and the camera are linked. The area light will only turn on when the area light is enabled and the camera is turned on.

- Control of the light source PWM output
 - Adjust brightness

```
em->setBrightnessValue(50);
```

Range: 0~100, where 100 indicates maximum brightness and 0 indicates minimum brightness.

- Turn on/off the light source

```
em->open_light();
```

```
em->close_light();
```

TIP

In an operating temperature environment of 25°C, it is recommended that the single activation duration of the light source should be $\leq 25\text{ms}$, and the single deactivation duration should be $\geq 330\text{ms}$, ensuring a duty cycle $\leq 7\%$. Here, the duty cycle = light-on time / (light-on time + light-off time).

5.2.1.6 Code Examples

I/O Control Class (C++)

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

```
class EdaIo{
public:
    static EdaIo* getInstance();
    static void close_io();
    ~EdaIo();
    /**
     * @brief Laser On
     *
     */
    void openLaser();
    /**
     * @brief Laser Off
     *
     */
    void closeLaser();
    /**
     * @brief set status indicator
     *
     * @param good
     */
    void setScanStat(bool good);
    /**
     * @brief alarm indicator On
     *
     */
    void openAlarm();
    /**
     * @brief alarm indicator Off
     *
     */
```

C++

```

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 set output1 to [high/low]
 *
 * @param high
 */
void setDo1High(bool high);
/**
 * @brief set output2 to [high/low]
 *
 * @param high
 */
void setDo2High(bool high);
// void setAimerColor(RGBColor color);
/**
 * @brief register input trigger callback function
 *
 * @param callback
 */
void registerInput(IoTrigger callback);
/**
 * @brief register button callback function
 *
 * @param callback
 */
void registerTrigger(IoTrigger callback);
/**
 * @brief register Tune button callback function
 *
 * @param callback
 */
void registerTune(IoTrigger callback);
/**
 * @brief set RGB light
 *
 * @param light 0: Close; 1: Red; 2: Green; 3: Blue,
 * @return int

```

```

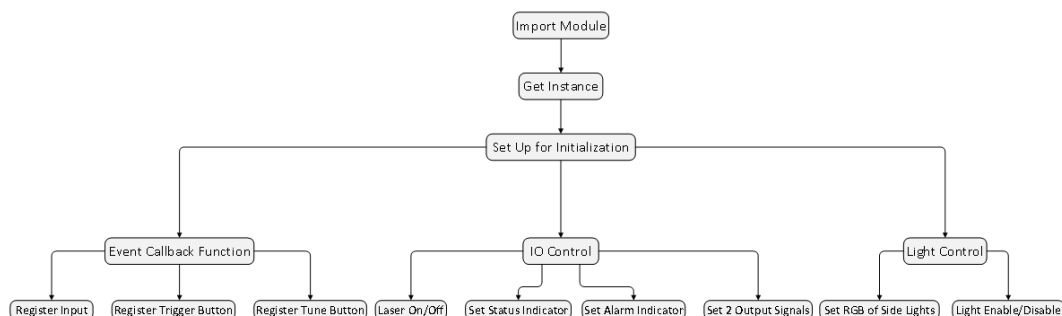
    */
void setRgbLight(uint8_t light);
/**
 * @brief Set the RGB Light
 *
 * @param r red
 * @param g green
 * @param b blue
 */
/**
 * @brief Enable manual brightness adjustment.
 * @param Parameter values:
 *     - 0 Disabled
 *     - 1 Enabled
 */
int setStrobeManual(int value);
/**
 * @brief Set PWM brightness parameter
 * @param PWM PWM brightness values:
 *     - 100 Maximum brightness
 *     - 50 Default
 *     - 0 Minimum brightness
 */
int setBrightnessValue(int value);
/**
 * @brief Custom light source mode, manually turn on the light source.
 *     This will change the operating mode and automatically set it to 5.
 * @param timeout Maximum light-on duration: 3000 milliseconds (3 seconds). Exceeding this
 * @return int
 */
int open_light();
/**
 * @brief Custom light source mode, manually turn off the light source.
 *
 * @return int
 */
int close_light();
void setup();
};

```

5.2.2 I/O Control (Python)

This section introduces the operations of indicator control, laser control, event callback, and output control.

5.2.2.1 Flow Diagram



5.2.2.2 Import Module

Before operating I/O, you need to import modules.

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

5.2.2.3 Getting Instance and Initializing

After importing the library environment, you need to obtain an IO instance and initialize the instance. The steps are as follows.

1. Getting an I/O instance.

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

2. Initializing the instance.

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

5.2.2.4 Event Callback Function

I/O control supports registering callback functions for events, including registering Input, registering Trigger button, and registering Tune button.

- DI 1 trigger event

```
registerInput(func_input);
```

The COMMON_IN pin in the 12-Pin M 12 interface is connected to ground signal, and the DI1 pin is connected to 5V signal for triggering.

- Registering Trigger button

```
registerTrigger(func_trigger);
```

- Registering Tune button

```
registerTune(func_tune);
```

Sample

```
#!/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()
```

C++

5.2.2.5 Controlling I/O

Using I/O to control the on/off of the laser, the on/off of the status indicator, the on/off of the alarm indicator and the enable/disable of the 2 outputs.

Preparation

Initialization of the instance has been completed.

Operating Instructions

- Laser On/Off

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

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

- Status indicator On/Off

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

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

- Alarm indicator On/Off

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

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

- 2 outputs enable/disable

```
eda.setDo1High(False);
```

```
eda.setDo2High(TRUE);
```

5.2.2.6 Controlling light

Both the camera side lights and area lights can be controlled independently.

Preparation

Initialization of the instance has been completed.

Operating Instructions

- Side light color

```
eda.setRgbLight(1);
```

- 0: Closing side light
- 1: Setting the color to Red
- 2: Setting the color to Green
- 3: Setting the color to Blue
- 4: Setting the color to Yellow
- 5: Setting the color to White

- Area lights

- Enable (The default state)

```
eda.enableLightSection(1);
```

The value range is 1~4, corresponding to different partitions.

- Disable

```
eda.disableLightSection(1);
```

The value range is 1~4, corresponding to different partitions.

Enabling/disabling the area light does not turn on/off the light. The area light and the camera are linked. The area light will only turn on when the area light is enabled and the camera is turned on.

- Control of the light source PWM output

- Adjust brightness

```
eda.setBrightnessValue(50);
```

Range: 0~100, where 100 indicates maximum brightness and 0 indicates minimum brightness.

- Turn on/off the light source

```
eda.open_light();
```

```
eda.close_light();
```

TIP

In an operating temperature environment of 25°C, it is recommended that the single activation duration of the light source should be $\leq 25\text{ms}$, and the single deactivation duration should be $\geq 330\text{ms}$, ensuring a duty cycle $\leq 7\%$. Here, the duty cycle = light-on time / (light-on time + light-off time). :::

5.2.2.7 Code Examples

I/O Control (Python 3)

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

def func_trigger(v):
    print("[Debug] Trigger: trigger button!", v)
```

py

```

...
eda = EdaIo.singleton(); # Get IO control instance
registerTrigger(func_trigger); # Register Trigger button callback
# registerInput(func_trigger); # Register Input callback
# registerTune(func_trigger); # Register Tune button callback
eda.setup(); # Initialization
...
eda.openLaser(); # Laser On
# eda.closeLaser(); # Laser Off
eda.setScanStat(True); # Set status indicator
eda.openAlarm(); # Alarm indicator on
# eda.closeAlarm(); # Alarm indicator off
eda.setDo1High(True); # Set output1
eda.setDo2High(False); # Set output2
eda.setRgbLight(1); # Set side light ,0: Off; 1: Red; 2: Green; 3: Blue 4: Yellow 5: White
eda.setStrobeManual(1); # Enable light source PWM output
eda.open_light(); # Turn on the light source
eda.setBrightnessValue(50); # Set brightness Range: 0~100, 0: minimum brightness, 100: maximum
# eda.close_light(); # Turn off the light source

```

5.2.3 Sensor Control Examples

Camera Sensor control software based on open source libraries `picamera2` , official information [Picamera2 Manual \(https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf\)](https://datasheets.raspberrypi.com/camera/picamera2-manual.pdf) , here are some simple instructions and examples.

5.2.3.1 Operating Steps

Before operating Camera, you need to import the IO module and then get the IO instance and initialise it (for details, see [2.2.2 Importing Module](#) and [2.2.3 Getting Instance and Initialising](#)), and then do the following.

1. Importing modules

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

2. Get the Camera instance

```
picam2 = Picamera2()
```

3. Create the preview configuration

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

4. Apply the preview configuration

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

5. Start the camera preview function, NULL means do not show the picture on the screen


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

6. Turn on the camera

```
picam2.start()
```

7. (Optional) Adjust the focal length of the liquid lens module

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

The value range is 0–46000, and users can adjust it according to their needs.

8. Capture the picture

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

9. Close the camera

```
picam2.close()
```

 8. close the camera

5.3 Examples

This chapter introduces detailed code examples, including writing code, compiling code, and running code.

5.3.1 Code Example

The following is an example of how to implement the function 'Turn on the camera and wait for 2s to capture a picture', using Python lines to write the code.

The detailed code is as follows

```
from picamera2 import Picamera2, Preview # Import Picamera2 library and preview function.
import time # Import the time module for time-delay operations.

picam2 = Picamera2() # Create a Picamera2 object instance.
camera_config = picam2.create_preview_configuration() # Create the camera's preview configuration
picam2.configure(camera_config) # configure the camera
picam2.start_preview(Preview.NULL) # start the camera's preview function (NULL preview is selected)
picam2.start() # start the camera
eda.setLiquid(22000) # adjust the focal length of the liquid lens module
time.sleep(2) # wait for 2 seconds to make sure the camera is stable
picam2.capture_file('test.jpg') # capture the photo and save it to the current directory, the file name is test.jpg
picam2.close() # shut down the camera, freeing up resources
```

After writing is completed, save it as `test123.py` file.

TIP

The file name can be customized.

5.3.2 Compiling and Running Code

After the Python code is written, you need to log in to the camera device, compiling and running it on the Raspberry Pi OS.

Preparation:

- The connection of camera cables has been completed. For detailed operations, please refer to the [Boot device](#)
- The camera has been powered and connected to the network through the router.
- Obtained the camera IP address and successfully logged in to the camera system.

Steps:

Create a folder on the camera OS and upload the code files written in Chapter [3.1 Code Example](#) to the folder.

Execute the following command to view the files in the folder and ensure that the code file has been uploaded successfully.

```
ls
```

1. Execute the following command to compile the written code.

```
sudo python test123.py
```

sh

- `test123.py` : means the code file written in Chapter [3.1 Code Example](#).

TIP

After successful operation, you can see that the laser lights up and goes out after waiting for 2 seconds.