

Install driver library

Install driver library

1. Introduction
2. Install dependent libraries
 1. smbus library
 2. Adafruit_SSD1306 library
3. Python driver library file
4. Transfer files to the motherboard system
5. Driver library installation
6. Import library file
- VII. Check the firmware version number
 1. Check the version number
 2. bot = CubeRaspberry(i2c_bus=1)
8. API Introduction

1. Introduction

After installing the CubeRaspberry driver library, you can call the corresponding API function to control the RGB lights and fan peripherals.

You can also follow the steps below to update the driver library!

Please refer to the following steps to install the driver library. Here, we take the installation of version V1.0.2 as an example.

Note: The CubeRaspberry chassis will automatically turn on the RGB lighting effect and fan when it is powered on.

2. Install dependent libraries

1. smbus library

The installation of the CubeRaspberry driver library requires the import of the time library and the smbus library. The time library comes with Python and does not need to be installed by yourself. You can install the smbus library by entering the following command in the terminal.

```
pip3 install smbus
```

Python smbus library functions are used to access serial I2C devices. Smbus library functions can be used to read and write I2C device information.

2. Adafruit_SSD1306 library

Since OLED display also requires related dependent libraries, we can install the required dependent libraries at the same time. Enter the following command in the terminal to install the Adafruit_SSD1306 library.

```
pip3 install Adafruit_SSD1306 --break-system-packages
```

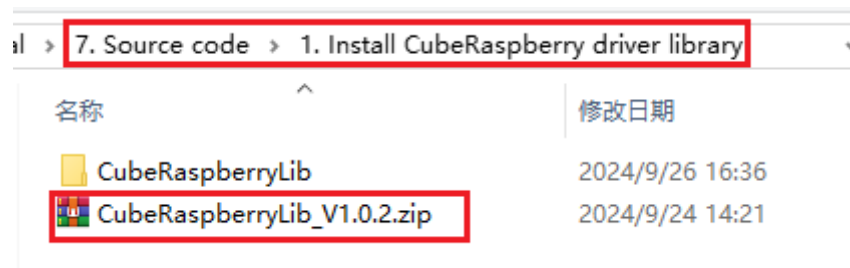
The python Adafruit_SSD1306 library is used to control the data display of our OLED display.

Note: If the OLED display tutorial prompts that the PIL library is missing, you can take the following solution.

```
sudo apt-get install libjpeg-dev zlib1g-dev
pip3 install setuptools --break-system-packages
pip3 install Pillow --break-system-packages
```

3. Python driver library file

This course material provides the latest version of the CubeRaspberry Python driver library, named CubeRaspberryLib_V1.0.2



7. Source code > 1. Install CubeRaspberry driver library	
名称	修改日期
CubeRaspberryLib	2024/9/26 16:36
CubeRaspberryLib_V1.0.2.zip	2024/9/24 14:21

4. Transfer files to the motherboard system

The following is a brief introduction to several methods of transferring files. You can choose the method you are familiar with for specific operations.

- VNC remote login to transfer files
- WinSCP to transfer files
- Online file transfer: such as WeChat file transfer assistant web version, etc.

Choose one of the methods to transfer the CubeRaspberryLib_V1.0.2.zip file to the motherboard system.

5. Driver library installation

Open the terminal in the folder directory where the compressed package is located and enter the following command:

The purpose of this command is to decompress and enter the setup.py directory to run the installation command. You don't have to worry about whether the version number is consistent.

```
unzip CubeRaspberryLib_V1.0.2.zip
cd CubeRaspberryLib
sudo python3 setup.py install
```

The image shows a Raspberry Pi terminal and a file manager window. The file manager displays the contents of the directory `/home/pi/cube_pi`, which includes `CubeRaspberryLib`, `CubeRaspberry`, and `CubeRaspberryLib_V1.0.2.zip`. The terminal shows the command `unzip CubeRaspberryLib_V1.0.2.zip` being executed, followed by the installation of the library using `sudo python3 setup.py install`. The output of the installation command is shown, indicating that the library is installed successfully.

```
pi@raspberrypi:~/cube_pi $ unzip CubeRaspberryLib_V1.0.2.zip
Archive: CubeRaspberryLib_V1.0.2.zip
  creating: CubeRaspberryLib/
  inflating: CubeRaspberryLib/.gitignore
  creating: CubeRaspberryLib/CubeRaspberryLib/
  inflating: CubeRaspberryLib/CubeRaspberryLib/CubeRaspberryLib.py
  inflating: CubeRaspberryLib/CubeRaspberryLib/__init__.py
  inflating: CubeRaspberryLib/README.md
  inflating: CubeRaspberryLib/setup.py
pi@raspberrypi:~/cube_pi $
```

```
pi@raspberrypi:~/cube_pi $ cd CubeRaspberryLib/
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $ sudo python3 setup.py install
running install
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py
install is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:146: EasyInstallDeprecationWarning: e
asy_install command is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
running bdist_egg
running egg_info
creating CubeRaspberryLib.egg-info
writing CubeRaspberryLib.egg-info/PKG-INFO
writing dependency_links to CubeRaspberryLib.egg-info/dependency_links.txt
writing top-level names to CubeRaspberryLib.egg-info/top_level.txt
writing manifest file 'CubeRaspberryLib.egg-info/SOURCES.txt'
reading manifest file 'CubeRaspberryLib.egg-info/SOURCES.txt'
writing manifest file 'CubeRaspberryLib.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-aarch64/egg
running install_lib
running build_py
creating build
creating build/lib
creating build/lib/CubeRaspberryLib
copying CubeRaspberryLib/CubeRaspberryLib.py -> build/lib/CubeRaspberryLib
copying CubeRaspberryLib/__init__.py -> build/lib/CubeRaspberryLib
creating build/bdist.linux-aarch64
creating build/bdist.linux-aarch64/egg
creating build/bdist.linux-aarch64/egg/CubeRaspberryLib
copying build/lib/CubeRaspberryLib/CubeRaspberryLib.py -> build/bdist.linux-aarch64/egg/CubeRaspberryLi
b
copying build/lib/CubeRaspberryLib/__init__.py -> build/bdist.linux-aarch64/egg/CubeRaspberryLib
byte-compiling build/bdist.linux-aarch64/egg/CubeRaspberryLib/CubeRaspberryLib.py to CubeRaspberryLib.c
python-311.pyc
byte-compiling build/bdist.linux-aarch64/egg/CubeRaspberryLib/__init__.py to __init__.cpython-311.pyc
creating build/bdist.linux-aarch64/egg/EGG-INFO
copying CubeRaspberryLib.egg-info/PKG-INFO -> build/bdist.linux-aarch64/egg/EGG-INFO
copying CubeRaspberryLib.egg-info/SOURCES.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying CubeRaspberryLib.egg-info/dependency_links.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying CubeRaspberryLib.egg-info/top_level.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating dist
creating 'dist/CubeRaspberryLib-1.0.2-py3.11.egg' and adding 'build/bdist.linux-aarch64/egg' to it
removing 'build/bdist.linux-aarch64/egg' (and everything under it)
Processing CubeRaspberryLib-1.0.2-py3.11.egg
Copying CubeRaspberryLib-1.0.2-py3.11.egg to /usr/local/lib/python3.11/dist-packages
Adding CubeRaspberryLib 1.0.2 to easy-install.pth file

Installed /usr/local/lib/python3.11/dist-packages/CubeRaspberryLib-1.0.2-py3.11.egg
Processing dependencies for CubeRaspberryLib==1.0.2
Finished processing dependencies for CubeRaspberryLib==1.0.2
```

If you see the installation version number at the end, it means the installation is successful. This command will overwrite the previously installed CubeRaspberryLib driver library.

You can also use the following command to confirm whether the library is installed successfully

```
pip3 list | grep "Cu"
```

The installation is successful, as shown in the figure

```
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $ pip3 list | grep "Cu*"
CubeRaspberryLib 1.0.2
CubeRaspberryLib 1.0.2
CubeRaspberryLib 1.0.2
types-Flask-Cors 3.0
types-JACK-Client 0.5
types-pyRFC3339 1.1
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $
```

6. Import library file

The name of the CubeRaspberry driver library is CubeRaspberryLib. Use CubeRaspberryLib to import the library in the program.

```
from CubeRaspberryLib import CubeRaspberry # Import CubeRaspberry driver library
```

VII. Check the firmware version number

1. Check the version number

(Python interactive interface: each statement needs to be run separately)

Enter python in the terminal to enter the interactive interface

```
from CubeRaspberryLib import CubeRaspberry # Import CubeRaspberry driver
library
bot = CubeRaspberry(i2c_bus=1) # Create an object, where 1 represents /dev/i2c-
1
version = bot.get_Version() # Get the version number
print("version:",version) # Print the version number
del bot # Delete the object
```

```
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $ python
Python 3.11.2 (main, May 2 2024, 11:59:08) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> from CubeRaspberryLib import CubeRaspberry
>>> bot = CubeRaspberry(i2c_bus=1)
>>> version = bot.get_Version()
>>> print("version:",version)
version: 5
>>> del bot
CubeRaspberry End!
>>>
```

2. bot = CubeRaspberry(i2c_bus=1)

Using the smbus library function, we need to create an object, where the 1 in i2c_bus=1 represents i2c - 1. The I2C device we are currently using is mounted on this device bus.

We can query the bus and address of the specific device through the following steps.

Query the I2C bus

```
i2cdetect -l
```

All the lists displayed by the system need to be queried. We need to find the bus where the two devices are mounted.

Query I2C devices

```
i2cdetect -y -r 1
```

Find the device bus with I2C address 0e and 3c. This bus is the device bus object we need to create.

```
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $ i2cdetect -l
i2c-1    i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-4    i2c          Synopsys DesignWare I2C adapter      I2C adapter
i2c-11   i2c          107d508200.i2c      I2C adapter
i2c-12   i2c          107d508280.i2c      I2C adapter
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $ i2cdetect -y -r 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  3c  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~/cube_pi/CubeRaspberryLib $
```

8. API Introduction

The following is an introduction to the API of the CubeRaspberry driver library. The function usage and parameter content will be introduced in the control course later.

Help on CubeRaspberry in module CubeRaspberryLib.CubeRaspberryLib object:

```
class CubeRaspberry(builtins.object)
|   # v1.0.1
|
|   Methods defined here:
|
|   __del__(self)
|
|   __init__(self, i2c_bus=7, delay=0.002, debug=False)
|       Initialize self.  See help(type(self)) for accurate signature.
|
|   get_Version(self)
|       # 获取固件版本号
|       # Obtain the firmware version number
|
|   set_Fan(self, state)
|       # 控制风扇 start=0 关闭风扇 start=1 打开风扇
|       # control Fan start=0 close start=1 open
|
|   set_RGB_Color(self, color)
|       # 设置RGB灯特效颜色 0-6:
|       # 0红色,1绿色,2蓝色,3黄色,4紫色,5青色,6白色
|       # Set RGB light effect color 0-6:
|       # 0 red, 1 green, 2 blue, 3 yellow, 4 purple, 5 cyan, 6 white
|
|   set_RGB_Effect(self, effect)
|       # 控制RGB灯特效: 0关闭特效,1呼吸灯,2跑马灯,3彩虹灯,4炫彩灯,5流水灯,6循环呼吸灯
|       # Control RGB light effect:
```

```

|         # 0 off effect, 1 breathing light, 2 marquee light, 3 rainbow light
|         # 4 dazzling lights, 5 running water lights, 6 Circulation breathing
lights
|
|     set_RGB_Speed(self, speed)
|         # 设置RGB灯特效速度 1-3: 1低速,2中速,3高速
|         # Set RGB light effect speed 1-3:1 low speed, 2 medium speed, 3 high
speed
|
|     set_Single_Color(self, index, r, g, b)
|         # 设置单个RGB灯颜色
|         # Set the individual RGB light color
|         # index表示灯珠序号0-13, index=255表示控制所有灯;
|         # r代表红色, g代表绿色, b代表蓝色
|         # index indicates the serial number of the lamp bead 0-13, index=255
means to control all lamps;
|         # r stands for red, g stands for green, and b stands for blue
|
|     -----
|     Data descriptors defined here:
|
|     __dict__
|         dictionary for instance variables (if defined)
|
|     __weakref__
|         list of weak references to the object (if defined)
(END)

```