7. Install Raspbot V2 driver library

7. Install Raspbot V2 driver library
- 7.1. Statement before installing the driver library
- 7.2. Download Python driver library file
- 7.3. Transfer files to Raspberry Pi 5
- 7.4. Start installation
- 7.5. Import library files
- 7.6、API简介
- 7.7 API Testing

7.1. Statement before installing the driver library

The factory image system of the car has already installed the latest driver library, so there is no need to install it again. If you are not using the factory image, or the driver library has updated content, you need to install the driver library.

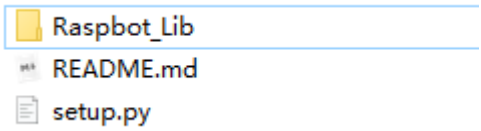
The storage path of the driver library that comes with the factory system: ~/py_install

Please refer to the following steps to install the driver library. Here, the installation of version V0.0.2 is taken as an example.

7.2. Download Python driver library file

The latest version of Raspbot V2 driver library is provided in this course material, named py_install.zip.

The compressed package contains the following files:

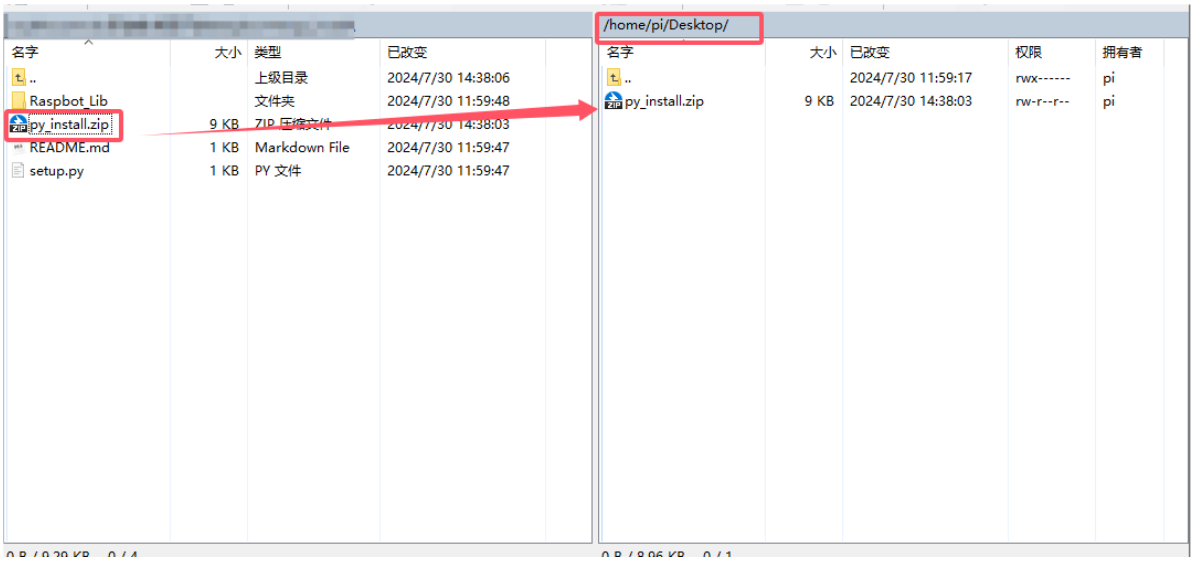


7.3. Transfer files to Raspberry Pi 5

If you use the browser that comes with Raspberry Pi 5 to download files from Yabo Smart Official Website, please download the files to a user-operable path, such as the desktop.

If you use the driver library compressed package file in the data, or download the driver library file with a computer browser, you can use WinSCP software to drag the driver library compressed package file to the Raspberry Pi 5 desktop.

After successful installation, the driver library file can be deleted.



If you don't know how to use WinSCP to transfer files, you can refer to **5. Remote file transfer**

7.4. Start installation

Open the terminal of Raspberry Pi 5 and enter the following command to decompress.

Enter the desktop and check if the file exists. The target file is in the red box.

```
cd ~/Desktop && ls
```

Unzip the file

```
unzip py_install.zip
```

```
pi@yahboom: ~/Desktop $ unzip py_install.zip
Archive:  py_install.zip
   creating: Raspbot_Lib/
  inflating: Raspbot_Lib/__init__.py
   creating: Raspbot_Lib/__pycache__/
  inflating: Raspbot_Lib/__pycache__/__init__.cpython-311.pyc
  inflating: Raspbot_Lib/__pycache__/Raspblock.cpython-311.pyc
  inflating: Raspbot_Lib/__pycache__/Raspblock_Lib.cpython-311.pyc
  inflating: Raspbot_Lib/Raspbot_Lib.py
  inflating: README.md
  inflating: setup.py
```

Note: The entire document example uses the py_install.zip compressed package placed on the Raspberry Pi 5 system desktop as an example. If the compressed package is stored in a different path, please enter the corresponding directory according to the actual path.

Enter the driver library folder

```
cd py_install
```

Run the installation command. If you see the installation version number at the end, it means the installation is successful. This command will overwrite the previously installed Raspbot V2 driver library.

```
sudo python3 setup.py install
```

```
pi@yahboom: ~/Desktop $ sudo python3 setup.py install
running install
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py install is depr
ecated. Use build and pip and other standards-based tools.
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:146: EasyInstallDeprecationWarning: easy_install comm
and is deprecated. Use build and pip and other standards-based tools.
  warnings.warn(
running bdist_egg
running egg_info
creating Raspbot_Lib.egg-info
writing Raspbot_Lib.egg-info/PKG-INFO
writing dependency_links to Raspbot_Lib.egg-info/dependency_links.txt
writing top-level names to Raspbot_Lib.egg-info/top_level.txt
writing manifest file 'Raspbot_Lib.egg-info/SOURCES.txt'
file Raspbot_Lib.py (for module Raspbot_Lib) not found
reading manifest file 'Raspbot_Lib.egg-info/SOURCES.txt'
writing manifest file 'Raspbot_Lib.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-aarch64/egg
running install_lib
running build_py
file Raspbot_Lib.py (for module Raspbot_Lib) not found
creating build
creating build/lib
creating build/lib/Raspbot_Lib
copying Raspbot_Lib/__init__.py -> build/lib/Raspbot_Lib
copying Raspbot_Lib/Raspbot_Lib.py -> build/lib/Raspbot_Lib
file Raspbot_Lib.py (for module Raspbot_Lib) not found
creating build/bdist.linux-aarch64
creating build/bdist.linux-aarch64/egg
creating build/bdist.linux-aarch64/egg/Raspbot_Lib
copying build/lib/Raspbot_Lib/__init__.py -> build/bdist.linux-aarch64/egg/Raspbot_Lib
copying build/lib/Raspbot_Lib/Raspbot_Lib.py -> build/bdist.linux-aarch64/egg/Raspbot_Lib
byte-compiling build/bdist.linux-aarch64/egg/Raspbot_Lib/__init__.py to __init__.cpython-311.pyc
byte-compiling build/bdist.linux-aarch64/egg/Raspbot_Lib/Raspbot_Lib.py to Raspbot_Lib.cpython-311.pyc
creating build/bdist.linux-aarch64/egg/EGG-INFO
copying Raspbot_Lib.egg-info/PKG-INFO -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Raspbot_Lib.egg-info/SOURCES.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Raspbot_Lib.egg-info/dependency_links.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Raspbot_Lib.egg-info/top_level.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
creating dist
creating 'dist/Raspbot_Lib-0.0.2-py3.11.egg' and adding 'build/bdist.linux-aarch64/egg' to it
removing 'build/bdist.linux-aarch64/egg' (and everything under it)
Processing Raspbot_Lib-0.0.2-py3.11.egg
Removing /usr/local/lib/python3.11/dist-packages/Raspbot_Lib-0.0.2-py3.11.egg
Copying Raspbot_Lib-0.0.2-py3.11.egg to /usr/local/lib/python3.11/dist-packages
Raspbot-Lib 0.0.2 is already the active version in easy-install.pth

Installed /usr/local/lib/python3.11/dist-packages/Raspbot_Lib-0.0.2-py3.11.egg
Processing dependencies for Raspbot-Lib==0.0.2
Finished processing dependencies for Raspbot-Lib==0.0.2
```

7.5. Import library files

Refer to the following picture to import the driver library. If no error message appears, the installation is successful.

```
pi@yahboom: ~/Desktop $ 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 Raspbot_Lib import Raspbot
>>>
```

7.6、API简介

```
class Raspbot():
"""
Raspberry Pi car control class, used to communicate with the Raspberry Pi car via
I2C.
"""

def get_i2c_device(self, address, i2c_bus):
"""
Get the I2C device instance.

Args:
address (int): I2C device address.
i2c_bus (int): I2C bus number.

Returns:
SMBus: I2C device instance.
"""
```

```
def __init__(self):
"""
Initialize the Raspbot instance and create an I2C device.
"""

def write_u8(self, reg, data):
"""
Write a byte of data to the specified register.

Args:
reg (int): register address.
data (int): data to be written.
"""

def write_reg(self, reg):
"""
Command to write a byte to the device.

Args:
reg (int): command byte.
"""

def write_array(self, reg, data):
"""
Write a series of bytes of data to the specified register.

Args:
reg (int): register address.
data (list): list of data to be written.
"""

def read_data_byte(self):
"""
Read a byte of data from the device.

Returns:
int: data read.
"""

def read_data_array(self, reg, length):
"""
Read a series of bytes of data from the specified register.

Args:
reg (int): register address.
length (int): length of data to be read.

Returns:
list: list of data read.
"""

def Ctrl_Car(self, motor_id, motor_dir, motor_speed):
"""
Control the direction and speed of the motor.
```

```
Args:
motor_id (int): motor ID.
motor_dir (int): Motor direction (0 is forward, 1 is reverse).
motor_speed (int): Motor speed (0-255).
"""

def Ctrl_Muto(self, motor_id, motor_speed):
"""
Controls the motor forward and reverse, accepts speed values ••from -255 to 255.

Args:
motor_id (int): Motor ID.
motor_speed (int): Motor speed (-255 to 255).
"""

def Ctrl_Servo(self, id, angle):
"""
Controls the angle of the servo.

Args:
id (int): Servo ID.
angle (int): Servo angle (0-180 degrees).
"""
def Ctrl_WQ2812_ALL(self, state, color):
"""
Control the state and color of all LED lights.

Args:
state (int): The state of the light (0 is off, 1 is on).
color (int): Color code.
"""

def Ctrl_WQ2812_Alone(self, number, state, color):
"""
Control the state and color of a single LED light.

Args:
number (int): The number of the light.
state (int): The state of the light (0 is off, 1 is on).
color (int): Color code.
"""

def Ctrl_WQ2812_brightness_ALL(self, R, G, B):
"""
Set the RGB brightness of all LED lights.

Args:
R (int): Red brightness (0-255).
G (int): Green brightness (0-255).
B (int): Blue brightness (0-255).
"""

def Ctrl_WQ2812_brightness_Alone(self, number, R, G, B):
"""
```

```
Set the RGB brightness of a single LED.

Args:
number (int): The number of the LED.
R (int): Red brightness (0-255).
G (int): Green brightness (0-255).
B (int): Blue brightness (0-255).
"""

def Ctrl_IR_Switch(self, state):
"""
Control the on/off state of the infrared remote control.

Args:
state (int): Switch state (0 is off, 1 is on).
"""

def Ctrl_BEEP_Switch(self, state):
"""
Control the on/off state of the buzzer.

Args:
state (int): Switch state (0 is off, 1 is on).
"""

def Ctrl_Ulatist_Switch(self, state):
"""
Controls the switch state of the ultrasonic ranging module.

Args:
state (int): Switch state (0 is off, 1 is on).
"""

class LightShow():
"""
LED light effect control class, used to achieve various lighting effects.
"""

def __init__(self):
"""
Initialize the LightShow instance, set basic parameters and initialize the
Raspberry Pi car control object.
"""

def execute_effect(self, effect_name, effect_duration, speed, current_color):
"""
Execute the specified lighting effect.

Args:
effect_name (str): Effect name.
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
current_color (int): Current color code.
"""
```

```
def turn_off_all_lights(self):
"""
Turn off all LED lights.
"""

def run_river_light(self, effect_duration, speed):
"""
Execute the running light effect.

Args:
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
"""

def breathing_light(self, effect_duration, speed, current_color):
"""
Execute the breathing light effect.

Args:
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
current_color (int): Current color code.
"""

def random_running_light(self, effect_duration, speed):
"""
Execute the random running light effect.

Args:
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
"""

def starlight_shimmer(self, effect_duration, speed):
"""
Executes the starlight shimmer effect.

Args:
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
"""

def gradient_light(self, effect_duration, speed):
"""
Executes the gradient effect.

Args:
effect_duration (float): Effect duration (seconds).
speed (float): Effect execution speed (seconds).
"""

def rgb_remix(self, val):
"""
RGB value mixing algorithm.
```

```
Args:
val (int): Input RGB value.

Returns:
int: RGB value after mixing.
"""

def rgb_remix_u8(self, r, g, b):
"""
RGB value mixing algorithm (for 8-bit values).

Args:
r (int): Red value.
g (int): Green value.
b (int): Blue value.

Returns:
tuple: RGB value after mixing.
"""

def calculate_breath_color(self, color_code, breath_count):
"""
Calculate the color value of the breathing light effect.

Args:
color_code (int): Color code.
breath_count (int): Breath count.

Returns:
tuple: RGB color value.
"""

def stop(self):
"""
Stop the currently executing lighting effect.
"""
```

7.7 API Testing

```
# test
#car = Raspbot()

#Read the line patrol sensor address, this value is only 1 bit
# track =car.read_data_array(0x0a,1)
# track = int(track[0])
# x1 = (track>>3)&0x01
# x2 = (track>>2)&0x01
# x3 = (track>>1)&0x01
# x4 = (track)&0x01
# print(track,x1,x2,x3,x4)

#Read the ultrasonic sensor address, this value is only 2 bits
# car.Ctrl_Ulatist_Switch(1)#open
# time.sleep(1)
# diss_H =car.read_data_array(0x1b,1)[0]
```

```
# diss_L =car.read_data_array(0x1a,1)[0]
# dis = diss_H<< 8 | diss_L
# print(dis+"mm")
# car.Ctrl_Ulatist_Switch(0)#close

#Read the value of infrared remote control
# car.Ctrl_IR_Switch(1)
# time.sleep(3)
# data =car.read_data_array(0x0c,1)
# print(data)
# car.Ctrl_IR_Switch(0)

#Buzzer test
# car.Ctrl_BEEP_Switch(1)
# time.sleep(1)
# car.Ctrl_BEEP_Switch(0)
# time.sleep(1)

#Motor test
# car.Ctrl_Car(0,0,150) #L1 motor forward 150 speed
# car.Ctrl_Car(1,0,150) #L2 motor forward 150 speed
# car.Ctrl_Car(2,0,150) #R1 motor forward 150 speed
# car.Ctrl_Car(3,0,150) #R2 motor forward 150 speed
# time.sleep(1)
# car.Ctrl_Car(0,1,50) #L1 motor backward 50 speed
# time.sleep(1)
# car.Ctrl_Car(0,0,0) #L1 motor stop
# car.Ctrl_Car(1,0,0) #L2 motor stop
# car.Ctrl_Car(2,0,0) #R1 motor stop
# car.Ctrl_Car(3,0,0) #R2 motor stop
#Servo test
# car.Ctrl_Servo(1,0) #s1 0 degrees
# car.Ctrl_Servo(2,180) #s2 180 degrees
# time.sleep(1)
# car.Ctrl_Servo(1,180) #s1 180 degrees
# car.Ctrl_Servo(2,0) #s2 0 degrees
# time.sleep(1)

# light bar test
# car.Ctrl_WQ2812_ALL(1,0)# red
# time.sleep(1)
# car.Ctrl_WQ2812_ALL(1,1)# green
# time.sleep(1)
# car.Ctrl_WQ2812_ALL(1,2)# blue
# time.sleep(1)
# car.Ctrl_WQ2812_ALL(1,3)# yellow
# time.sleep(1)
# car.Ctrl_WQ2812_ALL(0,0) # off

# single light test
# car.Ctrl_WQ2812_Alone(1,1,0)# red No. 1
# time.sleep(1)
# car.Ctrl_WQ2812_Alone(2,1,3)# yellow No. 1
# time.sleep(1)
# car.Ctrl_WQ2812_Alone(1,0,3)#1 off
```

```
# time.sleep(1)
# car.Ctrl_WQ2812_Alone(10,1,2)#10 green
# time.sleep(1)
# car.Ctrl_WQ2812_ALL(0,0) #off

#control brightness test all
# for i in range(255):
# car.Ctrl_WQ2812_brightness_ALL(i,0,0)
# time.sleep(0.01)
```