

Install Transbot library

Install Transbot library

- 1、Statement before installing the driver library
- 2、Download the Python driver library file
- 3、Transfer file to Jetson Nano system
- 4、Install steps
- 5、Check version
- 6、Basic usage of the driver library
- 7、API

1、Statement before installing the driver library

The Yahboom Transbot image system has been installed with the latest driver library. If you are using the image file provided by us, you didn't need to install the driver library file repeatedly.

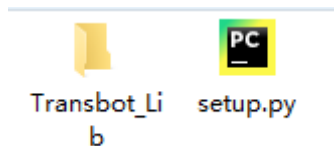
Driver library file path in Yahboom Transbot: ~/Transbot/py_install

For the method of installing the driver library, please refer to the following steps.

The installation of version V3.1.0 is taken as an example.

2、Download the Python driver library file

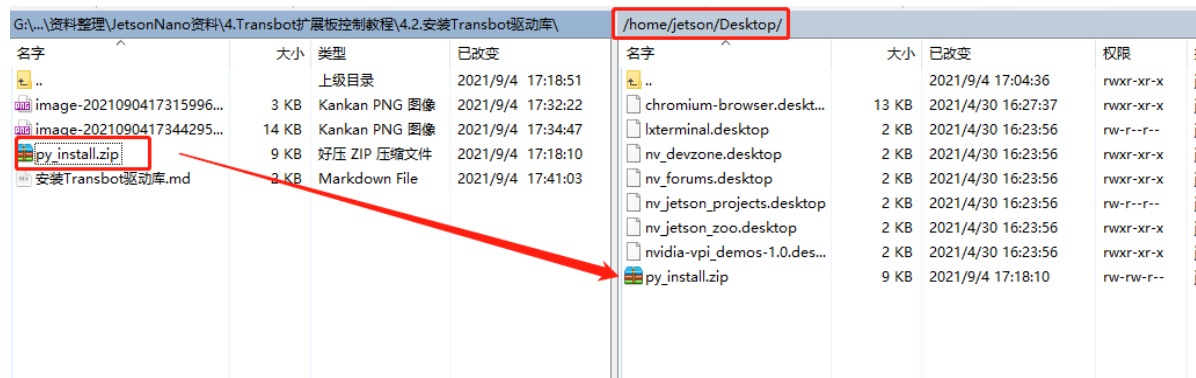
We have provided Transbot Python library -- py_install.zip.



Or you can download this file on our website, [Download]--[Transbot Library].

3、Transfer file to Jetson Nano system

You can transfer file into Raspberry Pi system by WinSCP software. This software download link: <https://winscp.en.softonic.com/>



4、 Install steps

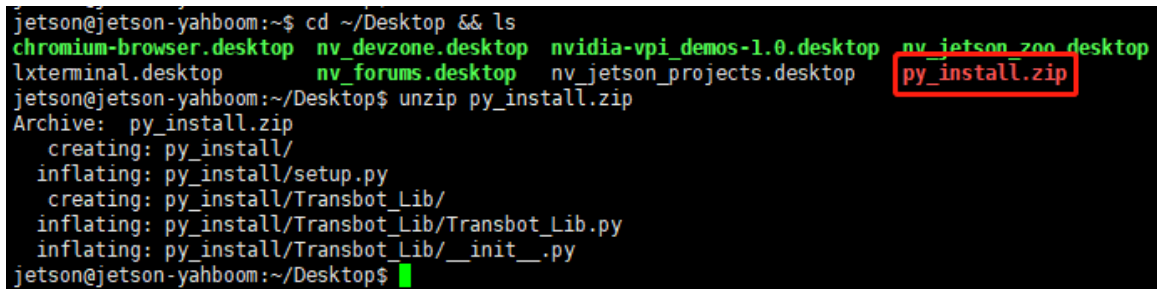
Open Jetson Nano's terminal and enter the following command to extract this zip file.

Go to the desktop and check whether the file exists, as shown below.

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

Input following command to extract file.

```
unzip py_install.zip
```

A terminal window screenshot showing the command 'cd ~/Desktop && ls' and its output, which lists various .desktop files and 'py_install.zip' (highlighted with a red box). Below this, the command 'unzip py_install.zip' is executed, showing the process of creating and inflating the 'py_install' directory and its subdirectories, including 'Transbot_Lib' and its files.

```
jetson@jetson-yahboom:~$ cd ~/Desktop && ls
chromium-browser.desktop  nv_devzone.desktop  nvidia-vpi_demos-1.0.desktop  nv_jetson_zoo.desktop
lxterminal.desktop        nv_forums.desktop   nv_jetson_projects.desktop    py_install.zip
jetson@jetson-yahboom:~/Desktop$ unzip py_install.zip
Archive:  py_install.zip
  creating: py_install/
  inflating: py_install/setup.py
  creating: py_install/Transbot_Lib/
  inflating: py_install/Transbot_Lib/Transbot_Lib.py
  inflating: py_install/Transbot_Lib/__init__.py
jetson@jetson-yahboom:~/Desktop$
```

Note: This tutorial takes the py_install.zip compressed package on the desktop of the Jetson Nano system as an example.

If you store the zip package in a different path, please enter the corresponding directory according to the actual path to operate.

Input following command to enter library folder.

```
cd py_install
```

Input following command, when you see the version number, which means it will be installed successfully. This command will overwrite the previously installed Transbot_Lib driver library.

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

```

jetson@jetson-yahboom:~/Desktop$ cd py_install
jetson@jetson-yahboom:~/Desktop/py_install$ sudo python3 setup.py install
running install
running bdist_egg
running egg_info
creating Transbot_Lib.egg-info
writing Transbot_Lib.egg-info/PKG-INFO
writing dependency links to Transbot_Lib.egg-info/dependency_links.txt
writing top-level names to Transbot_Lib.egg-info/top_level.txt
writing manifest file 'Transbot_Lib.egg-info/SOURCES.txt'
reading manifest file 'Transbot_Lib.egg-info/SOURCES.txt'
writing manifest file 'Transbot_Lib.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/Transbot_Lib
copying Transbot_Lib/__init__.py -> build/lib/Transbot_Lib
copying Transbot_Lib/Transbot_Lib.py -> build/lib/Transbot_Lib
creating build/bdist.linux-aarch64
creating build/bdist.linux-aarch64/egg
creating build/bdist.linux-aarch64/egg/Transbot_Lib
copying build/lib/Transbot_Lib/__init__.py -> build/bdist.linux-aarch64/egg/Transbot_Lib
copying build/lib/Transbot_Lib/Transbot_Lib.py -> build/bdist.linux-aarch64/egg/Transbot_Lib
byte-compiling build/bdist.linux-aarch64/egg/Transbot_Lib/__init__.py to __init__.cpython-36.pyc
byte-compiling build/bdist.linux-aarch64/egg/Transbot_Lib/Transbot_Lib.py to Transbot_Lib.cpython-36.pyc
creating build/bdist.linux-aarch64/egg/EGG-INFO
copying Transbot_Lib.egg-info/PKG-INFO -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Transbot_Lib.egg-info/SOURCES.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Transbot_Lib.egg-info/dependency_links.txt -> build/bdist.linux-aarch64/egg/EGG-INFO
copying Transbot_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/Transbot_Lib-3.1.0-py3.6.egg' and adding 'build/bdist.linux-aarch64/egg' to it
removing 'build/bdist.linux-aarch64/egg' (and everything under it)
Processing Transbot_Lib-3.1.0-py3.6.egg
Copying Transbot_Lib-3.1.0-py3.6.egg to /usr/local/lib/python3.6/dist-packages
Removing Transbot-Lib 2.5.3 from easy-install.pth file
Adding Transbot-Lib 3.1.0 to easy-install.pth file

Installed /usr/local/lib/python3.6/dist-packages/Transbot_Lib-3.1.0-py3.6.egg
Processing dependencies for Transbot-Lib==3.1.0
Finished processing dependencies for Transbot-Lib==3.1.0
jetson@jetson-yahboom:~/Desktop/py_install$

```

5、 Check version

Input following command in the terminal to view the version number of Transbot-Lib,

```
pip3 list | grep Transbot
```

```

jetson@jetson-yahboom:~/Desktop/py_install$ pip3 list | grep Transbot
Transbot-Lib          3.1.0
WARNING: You are using pip version 21.2.3; however, version 21.2.4 is available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade pip' command.
jetson@jetson-yahboom:~/Desktop/py_install$

```

You can import this library by following code.

```
from Transbot_Lib import Transbot
```

6、 Basic usage of the driver library

Check video tutorial.

Code path: /home/pi/py_install_V3.2.5/py_install/Samples/2.test_transbot.ipynb

7、API

Transbot drive library API

```
| clear_auto_report_data(self)
|     # Clear the buffered data automatically sent by the MCU
|
| create_receive_threading(self)
|     # Start the thread for receiving and processing data
|
| get_accelerometer_data(self)
|     # Get three-axis data of accelerometer, return a_x, a_y, a_z
|
| get_battery_voltage(self)
|     # Get battery voltage value
|
| get_gyroscope_data(self)
|     # Get the three-axis data of the gyroscope, and return g_x, g_y, g_z
|
| get_imu_state(self)
|     # Get the state of the gyroscope-assisted trolley movement, return True
to indicate that it is on, return False to indicate that it is off, and return
-1 if reading fails.
|
| get_motion_data(self)
|     # Get the speed of the robot, return the linear velocity v, the angular
velocity a
|
| get_motion_pid(self)
|     # Get the motion PID parameters of the car, return kp, ki, kd
|
| get_uart_servo_angle(self, s_id)
|     # Read the angle of the bus servo, s_id represents the ID number of the
servo to be read, s_id=[7-9]
|
| get_uart_servo_angle_array(self)
|     # Read the angles of three servos at once [xx, xx, xx], if one of the
servos is wrong, the one bit is -1
|
| get_uart_servo_value(self, servo_id)
|     # Read bus servo position parameter, servo_id=[1-250], return: ID read,
current position parameter
|
| get_version(self)
|     #Get the version number of the MCU, such as 1.1
|
| reset_flash_value(self)
|     # Reset the data saved in the flash of the car and restore the factory
default values.
|
| set_auto_report_state(self, enable, forever=False)
|     # The MCU automatically returns the data status bit, the default is on,
if the setting is off, it will affect the function of reading data.
|     # enable=True, the bottom expansion board will send data every xx
milliseconds. enable=False, do not send.
|     # forever=True for permanent preservation,=False for temporary use.
|
```

```

|   set_beep(self, on_time)
|       # Buzzer switch, on_time=0: off, on_time=1: always beeping,
|       # on_time>=10: automatically turn off after xx milliseconds (on_time is a
multiple of 10).
|
|   set_car_motion(self, velocity, angular)
|       # Car motion control, velocity=[-0.45, 0.45], angular=[-2.00, 2.00]
|
|   set_car_run(self, speed)
|       # Set the car to move forward or backward, and automatically use the
gyroscope to assist the direction of movement.
|       # speed=[-0.45,0.45], =0 stop, =positive number forward, =negative number go
back.
|
|   set_colorful_effect(self, effect, speed=255, parm=255)
|       # RGB programmable lights with special effects display.
|       # effect=[0, 6], 0: stop light effect, 1: running water light, 2: marquee
light, 3: breathing light, 4: gradient light, 5: starlight, 6: battery display
|       # speed=[1, 10], the smaller the value, the faster the speed changes.
|       # parm, can be left blank, as an additional parameter. Usage 1: Breathing
light effect is passed in [0, 6] to modify the color of breathing light.
|
|   set_colorful_lamps(self, led_id, red, green, blue)
|       # RGB programmable light strip control, which can be controlled
individually or collectively. The special effects of RGB light need to be stopped
before control.
|       # led_id=[0, 16], control the corresponding number of RGB lights;
led_id=0xFF, control all lights.
|       # red,green,blue=[0, 255], which means the color RGB value.
|
|   set_floodlight(self, light)
|       # Control the searchlight, light controls the brightness, light = [0,
100], 0 is off, 100 is the brightest.
|
|   set_imu_adjust(self, enable, forever=False)
|       # Set the gyroscope to assist in adjusting the direction of the car's
movement. After the setting is turned on, the set_car_motion function will work
when the angular velocity is set to 0.
|       # enable=True is turned on, =False is turned off. forever=True for
permanent preservation, =False for temporary use.
|
|   set_motor(self, index, speed)
|       # Control the motor's PWM pulse to control the speed (the encoder is not
used for speed measurement). index:[1,2], speed:±100
|
|   set_pid_param(self, kp, ki, kd, forever=False)
|       # PID parameter control will affect the change of the speed of the car
controlled by the set_car_motion function. No adjustment is required by default.
|       # kp ki kd = [0, 10.00], decimals can be entered.
|       # forever=True for permanent preservation, =False for temporary use.
|       # Since permanent storage needs to be written to the chip flash, the
operation time is longer, so delay delay time is added to avoid the problem of
packet loss caused by the single-chip microcomputer.
|       # Temporary action is quick to respond, single-time effective, and data
will not be retained after restarting the single chip.
|
|   set_pwm_servo(self, servo_id, angle)

```

```

|      # Servo control, servo_id: Corresponding ID number: X = 1, Y = 2, angle:
Corresponding servo angle value
|      # servo_id=[1, 2], angle=[0, 180]
|
|      set_speed_limit(self, line_limit, angular_limit, forever=False)
|      # Set the minimum speed limit value (minimum value) of the trolley
movement. It can not be set by default.
|      # line_limit=[0, 0.20], angular_limit=[0, 1.00]
|      # forever=True for permanent,=False for temporary.
|
|      set_uart_servo(self, servo_id, pulse_value, run_time=500)
|      # Control the bus servo. servo_id:[1-255], represents the ID number of
the servo to be controlled, when id=254, it controls all connected servos.
|      # pulse_value=[96,4000] indicates the position to which the steering gear
will run.
|      # run_time represents the running time (ms), the shorter the time, the
faster the steering gear will rotate. The minimum is 0, the maximum is 2000
|
|      set_uart_servo_angle(self, s_id, s_angle, run_time=500)
|      # Set the bus servo angle interface: id:7-9, angle: 7:[0, 225], 8:[30,
270], 9:[30, 180], set the angle to which the servo will move.
|      # Set the upright clamping state, the three servos are all 180 degrees,
7/8 clockwise (down) to decrease, counterclockwise (up) to increase, loosen the
clamp to decrease, and to clamp increase.
|      # run_time represents the running time (ms), the shorter the time, the
faster the steering gear will rotate. The minimum is 0, the maximum is 2000
|
|      set_uart_servo_angle_array(self, angle_7, angle_8, angle_9, run_time=500)
|      # Control the angle of three servos at the same time.
|      # Set the upright clamping state, the three servos are all 180 degrees,
7/8 clockwise (down) to decrease, counterclockwise (up) to increase, loosen the
clamp to decrease, and to clamp increase.
|      # angle_7=[0, 225], angle_8=[30, 270], angle_9=[30, 180]
|      # run_time represents the running time (ms), the shorter the time, the
faster the steering gear will rotate. The minimum value is 0, the maximum value
is 2000
|
|      set_uart_servo_id(self, servo_id)
|      # Set the ID number of the bus servo, servo_id=[1-250].
|      # Before running this function, please make sure that only one bus servo
is connected, otherwise all connected bus servos will be set to the same ID,
causing control confusion.
|
|      set_uart_servo_torque(self, enable)
|      # Turn off/on the torque of the bus servo, enable=[0, 1].
|      # enable=0: Turn off the servo torque, you can turn the servo by hand,
but the command cannot control the rotation;
|      # enable=1: Turn on the torque force, the command can control the
rotation, and the steering gear cannot be turned by hand.

```

