5.6 Use of servo

Note: In the image operating system provided by Yahboom, the APP remote control process is enabled by default, in order to avoid multiple occupations of internal resources, causing some functions to fail to operate normally.

Before you running the code of this course, please follow the following to close the APP remote control process.

If you want to permanently close the function of the APP that starts automatically after booting, execute the following command:

```
sudo systemctl disable jetbotmini_start
```

If you want to permanently open the function of the APP that starts automatically after booting, execute the following command:

```
sudo systemctl enable jetbotmini_start
```

If you do not restart Jetbotmini to restart the APP function, execute the following command:

```
sudo systemctl stop jetbotmini_start
sudo systemctl start jetbotmini_start
```

Use the servo to drive the asynchronous write command to control the servo

Note: The car is not equipped with a steering gear. The steering gear can be purchased separately according to the needs. The steering gear port can choose a PWM steering gear with a duty ratio of 0.5~2.5ms and a current less than 1A, such as a 9g steering gear: SG90.

Because the steering gear is also controlled by a coprocessor, we import the packages we need to use, then create an instance and initialize it.

```
import smbus
import time
bus = smbus.SMBus(1)
Servo_ADD = 0x1B
```

Then we can start to use the example method to control the steering gear rotation.

```
#Swing the steering gear to the center position
bus.write_i2c_block_data(Servo_ADD,3,[1,90])
time.sleep(0.1)
bus.write_i2c_block_data(Servo_ADD,3,[2,90])
#Swing the steering gear to 45 degrees
bus.write_i2c_block_data(Servo_ADD,3,[1,45])
time.sleep(0.1)
bus.write_i2c_block_data(Servo_ADD,3,[2,45])
#Swing the steering gear to 180 degrees
bus.write_i2c_block_data(Servo_ADD,3,[1,180])
time.sleep(0.1)
bus.write_i2c_block_data(Servo_ADD,3,[2,180])
```

Send data to IIC according to the communication protocol. The parameters in the write_i2c_block_data function are (coprocessor address, servo register address, [servo number, servo angle]), where the coprocessor address and servo register address are fixed Servo number 1 corresponds to S1 on the bottom plate, 2 corresponds to S2 on the bottom plate, and the servo angle is set between 0-180.

The corresponding complete source code is located:

/home/jetson/Notebooks/English/7.Use_of_servo/Control_servo.ipynb