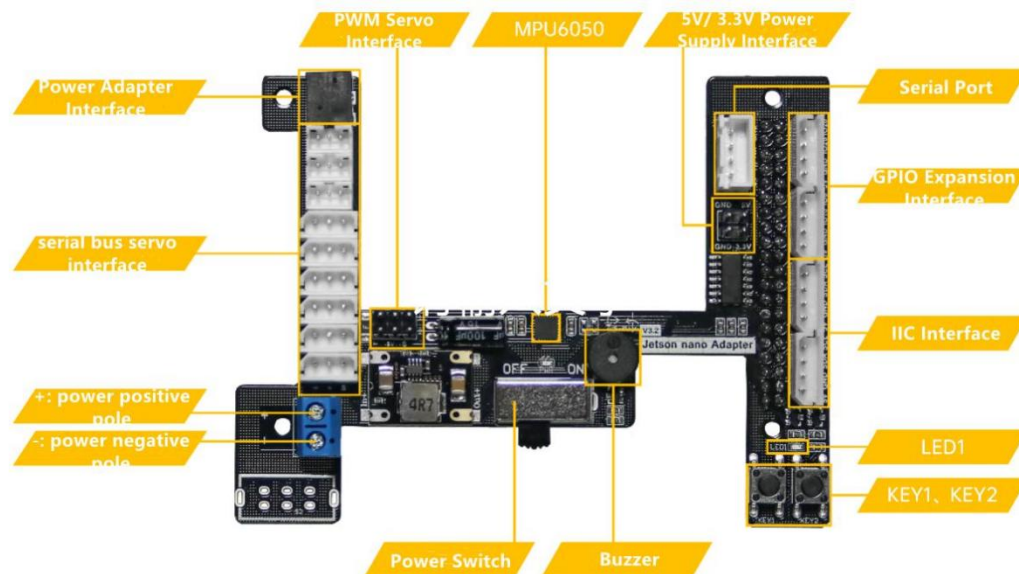


Part 1 Introduction to Jetson Nano Expansion Board

1. Expansion Board Ports



2. Ports for Electronic Module

Port	Function
Power adapter port	It can connect to 12.6V 2A power adapter to supply power
Serial bus servo port	It can connect to and drive serial bus servo as well as read its status
Power port	“+” connects to positive electrode, and “-” connects to negative electrode
PWM servo port	It can connect to and drive the PWM servo
Power switch	Turn on/ off the device
Buzzer	It can make sound through programming
KEY1、KEY2	Customize function When using JetAuto mirroring provided by Hiwonder: KEY1: Long press to switch the connection from LAN mode to direct connection mode. KEY2: Long press to restart JetAuto.
Serial port	For module expansion
5V/3.3V power supply port	
GPIO port	
IIC port	

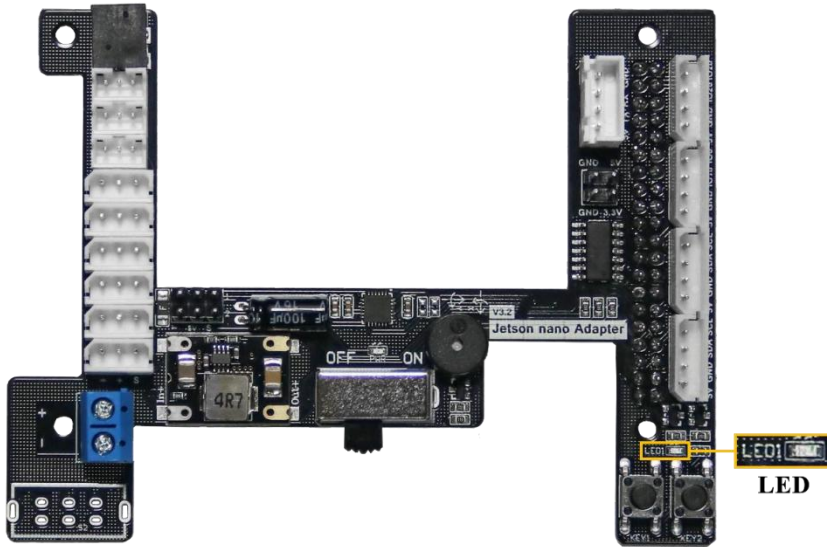
Part 2 Expansion Board Installation✓

Part 3 Application Notice✓

Part 4 Control LED

1. LED Location

There is a LED on Jetson Nano expansion board as pictured



2. Program Logic

The level status of I/O pin determines LED to light up and go out. When the pin is at low level, LED will go out.

The source code of the program is stored in
`/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/led_demo.py`

```
#!/usr/bin/python3
# coding=utf8
import time
from sdk import led

print('led blink at 5hz')

try:
    while True:
        led.on() # led on
        time.sleep(0.1) # led on-delayed time
        led.off() # led off
        time.sleep(0.1) # led off-delayed time
except KeyboardInterrupt:
    led.off() # Turns the led off when shutting down the program
```

3. Operation Steps

1) Start Jetson Nano robotic kit, then connect it to NoMachine.

2) Double click  to open the command line terminal

3) Input command “`cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/`” and press “Enter” to enter the directory where the game files are stored.

4) Input command “`python3 led_demo.py`” and press Enter to run the game program.

5) If you want to exit the program, you can press “**Ctrl+C**”

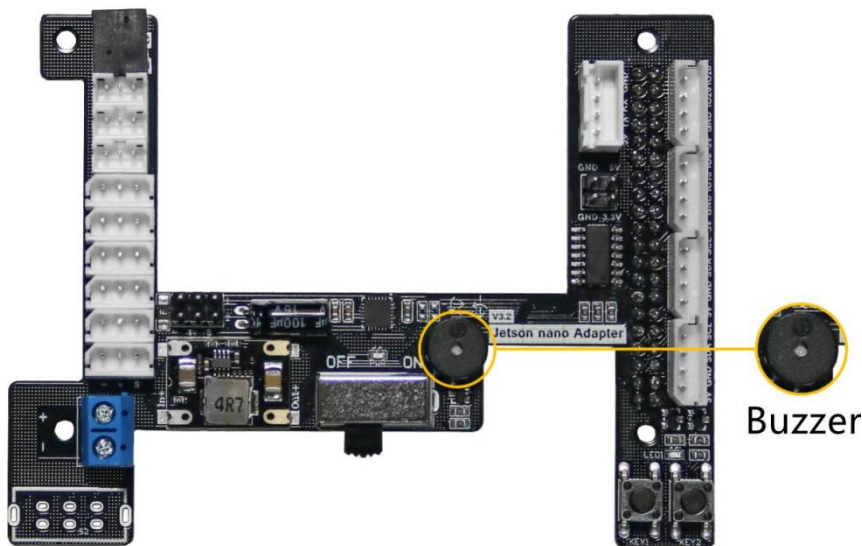
4. Program Outcome

After the game starts, LED will flash twice quickly, then light up once, and it will cycle in this mode.

Part 5 Control Buzzer

1. Buzzer Location

There is a buzzer on Jetson Nano expansion board.



2. Program Logic

The level status of the corresponding pin determines the buzzer to make sound or not. When the pin is at high level, the buzzer will make sound. When it is at low level, it will not make sound.

```
#!/usr/bin/python3
# coding=utf8
import time
from sdk import buzzer

print('buzzer di at 1hz')


try:
    while True:
        buzzer.on() # buzzer on
        time.sleep(0.1)
        buzzer.off() # buzzer off
```

```

time.sleep(0.9)
except KeyboardInterrupt:
    buzzer.off() # buzzer off

```

3. Operation Steps

- 1) Start Jetson Nano robotic kit, then connect it to NoMachine.
- 2) Double click  to open the command line terminal
- 3) Input command “`cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/`” and press Enter to enter the directory where the program file is stored.
- 4) Input command “`python3 buzzer_demo.py`” and press Enter to run the game program.
- 5) If you want to exit this program, you can press “**Ctrl+C**”.

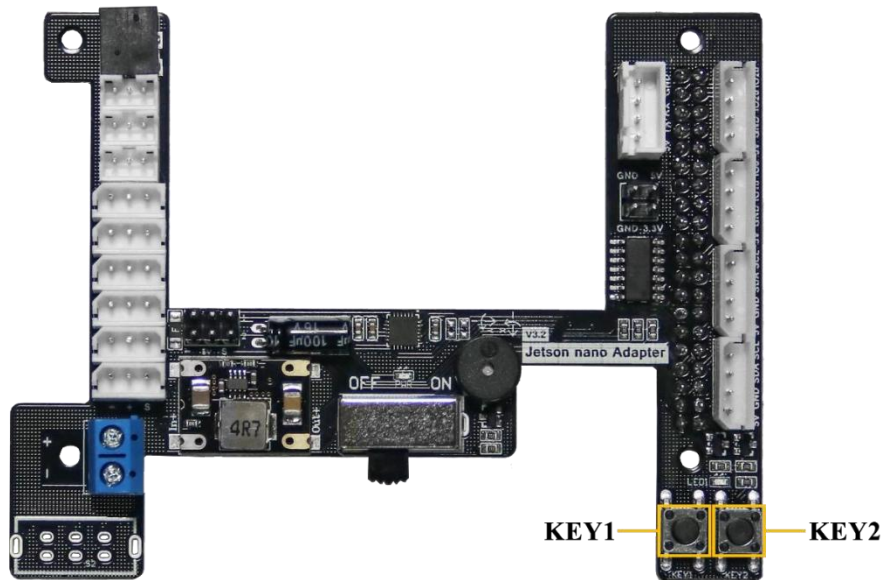
4. Program Outcome

After the game starts, the buzzer will keep making sound.

Part 6 Control Key

1. Key Location

There are two keys on Jetson Nano expansion board as pictured.



2. Program Logic


The status of the key that is whether it is pressed or not can be acquired through reading the level status of I/O key. If the pin is at high level, the key is released. If the pin is at low level, the button is pressed.

The source code of this program is located in
`/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/button_demo.py`

```
#!/usr/bin/python3
# coding=utf8
from sdk import button

while True:
    try:
        print('\rkey1: {}    key2: {}'.format(button.get_button_status('key1'),
button.get_button_status('key2')), end='', flush=True)
        #The flush=True parameter ensures that the output is immediately
        #flushed to the console.
        # 打印 key 状态
    except KeyboardInterrupt:
        break
```

3. Operation Steps

- 1) Start Jetson Nano robot kit, and connect it to NoMachine
- 2) Double click  to open the command line terminal
- 3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/**” and press Enter to enter the directory where the program file is stored.
- 4) Input command “**python3 button_demo.py**” and press Enter to run the game program
- 5) If you want to exit this program, you can press “**Ctrl+C**”.

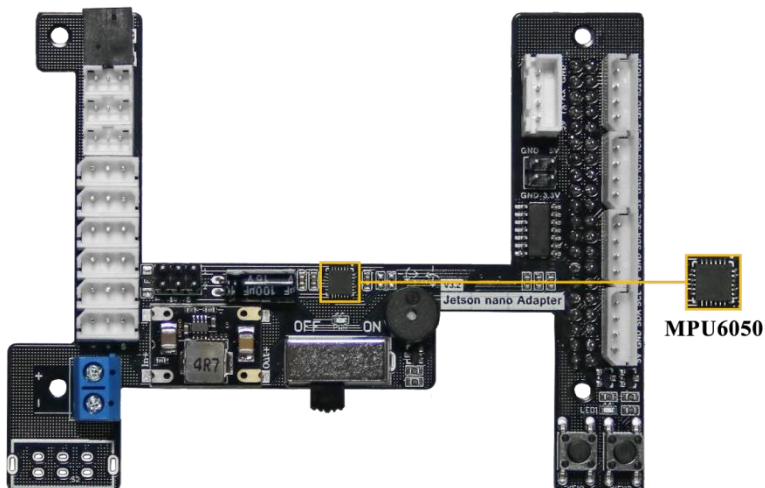
4. Program Outcome

After the game starts, the status of “key1” and “key2” will be printed on the command line terminal. “0” means that the key is pressed, and “1” means that the key is released. Take “key1” for example. When you long press the key1, the number behind key1 will change from “1” to “0”. When you release the key2, the number will change from “0” to “1”.

Part 7 Read Acceleration Sensor

1. MPU6050 Location

Jetson Nano expansion board comes with a MPU6050 as the picture shown.



2. Program Logic

Inertial Measurement Unit is a device used to measure three-axis attitude angle and acceleration. In general, IMU is installed on the center of gravity of the object to be measured.

IMU is mainly composed of acceleration sensor and gyroscope. Acceleration sensor is used to detect the acceleration signal of the object on the three axes of carrier coordinate system. And gyroscope is used to detect angular signal of the carrier relative to the navigation coordinate system. After the signal is processed, the posture of the object can be obtained.

MPU6050, six-axis posture sensor, is a kind of the IMU sensors, which integrates three-axis MEMS acceleration sensor, three-axis MEMS gyroscope and an expandable DMP (Digital Motion Processor).

The source code of this program is stored in

/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/imu_demo.py


```
#!/usr/bin/python3
# coding=utf8
import time
from sdk import imu

my_imu = imu.IMU()
print('read imu data at 10hz')

while True:
    try:
        ax, ay, az, gx, gy, gz = my_imu.get_data()
        # Get the raw data of the IMU device by calling the get_data() method
        # of my_imu object.
        print('ax:{:<8} ay:{:<8} az:{:<8} gx:{:<8} gy:{:<8}
gz:{:<8}'.format(round(ax, 5), round(ay, 5), round(az, 5), round(gx, 5),
round(gy, 5), round(gz, 5)))
        # {:<8} represents a left-aligned placeholder with width 8.
        # and use the round() function to retain five decimal places.
        time.sleep(0.1)
    except Exception as e:
        print(str(e)) #Prints a string representation of the caught exception,
                      # which can help with debugging and error finding.
```

3. Operation Steps

1) Start Jetson Nano robot kit, and connect it to NoMachine

2) Double click  to open the command line terminal

3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/**” and press Enter to enter the directory where the program file is kept.

4) Input command “**python3 imu_demo.py**” and press Enter to run the game program.

5) If you want to exit this program, please press “**Ctrl+C**”.

4. Program Outcome

After the game starts, the status of MPU6050 will be printed on the command line terminal.

```
ax:0.50005 ay:-0.09092 az:8.35728 gx:0.06229 gy:0.19313 gz:0.10506
ax:0.22969 ay:-0.33496 az:8.03667 gx:0.06322 gy:0.64606 gz:-0.06455
ax:-1.15562 ay:-0.57661 az:6.8667 gx:-0.01488 gy:0.65921 gz:0.00093
ax:-2.27056 ay:-0.1938 az:7.83809 gx:0.02218 gy:0.36633 gz:-0.01514
ax:-1.84229 ay:0.05264 az:7.58926 gx:0.0263 gy:0.3055 gz:-0.04636
ax:-2.42368 ay:-0.01675 az:7.929 gx:0.02829 gy:0.40113 gz:-0.01302
ax:-2.67251 ay:-0.08613 az:7.50552 gx:0.03852 gy:0.36699 gz:-0.06787
```

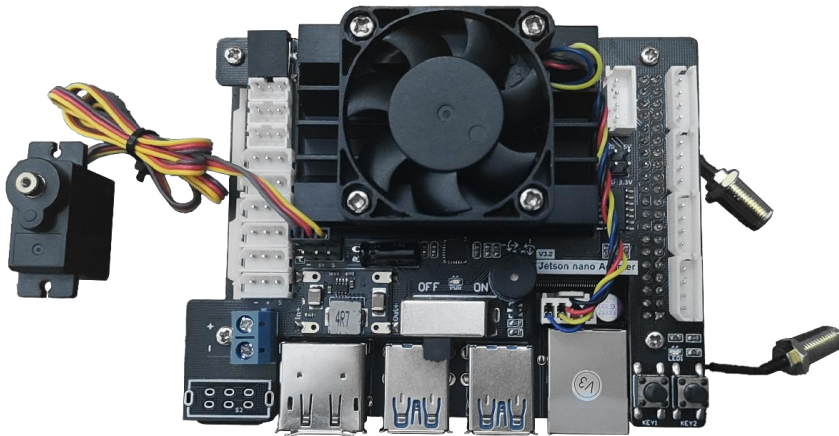
The meaning of the information printed is as follow.

- 1) ax, ay, az: they respectively refer to X-axis component, Y-axis component and Z-axis component of acceleration
- 2) gx, gy, gz: they respectively are X-axis, Y-axis and Z-axis components of the angular speed.

Part 8 Control Single PWM Servo

1. Preparation

Connect the PWM servo to any PWM servo port on the Jetson Nano expansion board. Take LFD-01M servo for example.



Note: please strictly distinguish the positive and negative electrode of the PWM servo. Red wire is the positive electrode, and brown wire is the negative electrode.

2. Program Logic

The program will control PWM servo to rotate by sending the pulse signal. Through sending the command containing position and time for rotation, PWM servo can be controlled to spend the set time rotating to the specific position.

The source code of this program is stored in `/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/pwm_servo/pwm_servo_demo.py`

```

#!/usr/bin/python3
# coding=utf8
import sys
import time
sys.path.append("..")
#Add the higher-level directory to the Python interpreter's search path
#so that you can import the modules in the higher-level directory.
from sdk import pwm_servo

my_servo = pwm_servo.PWMServo(1)
#Creates an instance of the pwm_servo.PWMServo class that represent a connected
servo.
#Parameter 1 represents the number or identifier of the servo.
my_servo.start() #Start the servo to make it ready.

print('servo1 move between 1300-1700')

while True:
    try:
        position = 1300 # Set the servo position to 1300 milliseconds.
        duration = 1000 # Set the move duration to 1000 milliseconds.
        my_servo.set_position(position, duration) # Call the set_position()
method to move the servo to the specified position and set the move duration.
        time.sleep(duration/1000.0) # The program pauses execution for a period
of time, waiting for the servo movement to complete. Here the duration is
converted to seconds

        position = 1700 # Reset the servo position to 1700.
        duration = 1000 # Set the duration of the servo motion to 1000
milliseconds.
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.
    except KeyboardInterrupt:
        position = 1500 # Reset servo position to 1500.
        duration = 1000 # Set the duration of servo motion to 1000 milliseconds
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.
        break

```


Take “**my_servo.set_position(position, duration)**” for example. The meaning of the parameters in bracket is explained below.

The first parameter “**position**” refers to the position to which the servo rotates i.e. pulse width.

Formula: pulse width = $11.1 \times \text{angle} + 500$ (only for reference)

The second parameter “**duration**” is the time for the servo rotation in millisecond.

3. Operation Steps

- 1) Start Jetson Nano robot kit, and connect it to NoMachine
- 2) Double click  to open the command line terminal
- 3) Input command `"cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/pwm_servo/"` and press Enter to enter the directory where the program file is kept.
- 4) Input command `"python3 pwm_servo_demo.py"` and press Enter to run the game program.
- 5) If you want to exit this program, please press **"Ctrl+C"**.

4. Program Outcome


After the game starts, PWM servo will rotate in this rule.

- 1) spend 1s rotating to 72° (1300 pulse width)
- 2) spend 1s rotating to 108° (1700 pulse width)

Then, the PWM servo will cycle in this way. If you press **"Ctrl+C"**, the program will be exited and the PWM servo will restore to 90° (1500 pulse width), which takes 1s.

5. Function Extension

The default servo port in the program is NO.1 port. If you want to connect the PWM servo to NO.2 port, you need to modify the program.

- 1) Double click  to open the command line terminal
- 2) Input command `"rosed jetauto_example pwm_servo_demo.py"` and press Enter to open the program file.
- 3) Locate to the following codes.
`my_servo = pwm_servo.PWMServo(1)`

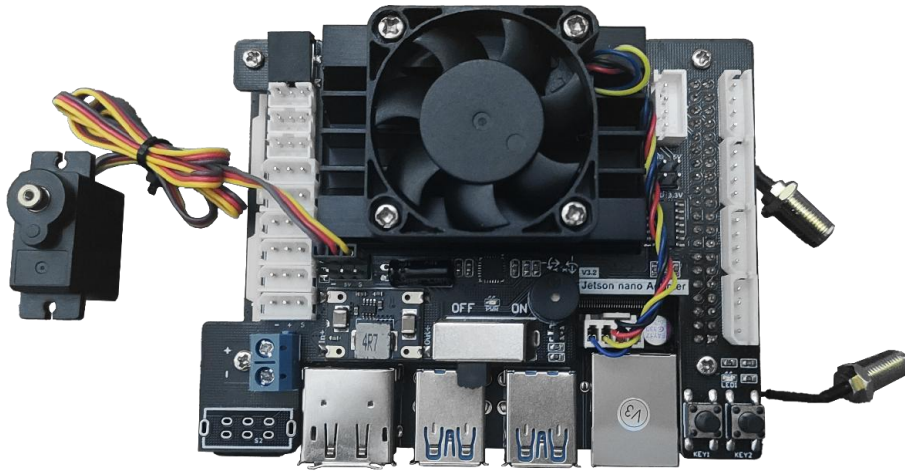
Note: you can input the corresponding line number, then press "Shift+G" key to move to the target line.

- 4) Press **"i"** key to enter the editing mode, then change **"1"** as **"2"**.
`my_servo = pwm_servo.PWMServo(2)`
- 5) After modification, press **"Esc"** key, input **":wq"** and press Enter to save and exit.
- 6) Lastly, restart the game according to the steps in **"3. Operation Steps"**, then you can check servo performance.

Part 9 Control PWM Servo Speed

1. Preparation

Connect the PWM servo to any PWM servo port on the Jetson Nano expansion board. Take LFD-01M servo for example.



Note: please strictly distinguish the positive and negative electrode of the PWM servo. Red wire is the positive electrode, and brown wire is the negative electrode.

2. Program Logic

The program will control PWM servo to rotate by sending the pulse signal. Through sending the command containing position and time for rotation, PWM servo can be controlled to spend the set time rotating to the specific position. And it can change the rotation time but remain the rotation position, the rotation speed of the servo can be adjusted.

The source code of the program is stored in
`/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/pwm_servo/pwm_servo_speed_demo.py`

```
#!/usr/bin/python3
# coding=utf8
import sys
import time
sys.path.append("..")
#Add the higher-level directory to the Python interpreter's search path
#so that you can import the modules in the higher-level directory.

from sdk import pwm_servo

my_servo = pwm_servo.PWMServo(1)
#Creates an instance of the pwm_servo.PWMServo class that represent a connected
servo.
#Parameter 1 represents the number or identifier of the servo.

my_servo.start() #Start the servo to make it ready.
print('servo1 move at different speed')
```

```

while True:
    try:
        position = 1300 # Set the servo position to 1300 milliseconds.
        duration = 500 # Set the move duration to 500 milliseconds.
        my_servo.set_position(position, duration) # Call the set_position()
method to move the servo to the specified position and set the move duration.
        time.sleep(duration/1000.0) # The program pauses execution for a period
of time, waiting for the servo movement to complete. Here the duration is
converted to seconds

        position = 1700 # Reset the servo position to 1700.
        duration = 500 # Set the duration of the servo motion to 500
milliseconds.
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 500 milliseconds) of the set motion duration.

        position = 1300 # Set the servo position to 1300 milliseconds.
        duration = 1000 # Set the duration of servo motion to 1000 milliseconds.
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.

        position = 1700 # Reset servo position to 1700.
        duration = 1000 # Set the duration of servo motion to 1000 milliseconds
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.

    except KeyboardInterrupt:
        position = 1500 # Reset servo position to 1500.
        duration = 1000 # Set the duration of servo motion to 1000 milliseconds.
        my_servo.set_position(position, duration) # Move the servo to the
specified position and set the motion duration.
        time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.
        break

```


Take “**my_servo.set_position(position, duration)**” for example. The meaning of the parameters in bracket is as follows.

The first parameter “**position**” refers to the position to which the servo rotates i.e. pulse width. Formula: pulse width = 11.1 x angle + 500 (only for reference) .

The second parameter “**duration**” is the time for the servo rotation in millisecond.

3. Operation Steps

The input command should be case sensitive, and the keywords can be complemented by “**Tab**” key.

- 1) Start Jetson Nano robot kit, and connect it to NoMachine
- 2) Double click  to open the command line terminal
- 3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/pwm_servo/**” and press Enter to enter the directory where the program file is kept.
- 4) Input command “**python3 pwm_servo_speed_demo.py**” and press Enter to run the game program.
- 5) If you want to exit this program, please press “**Ctrl+C**”.

4. Program Outcome

After the game starts, PWM servo will rotate in this rule.

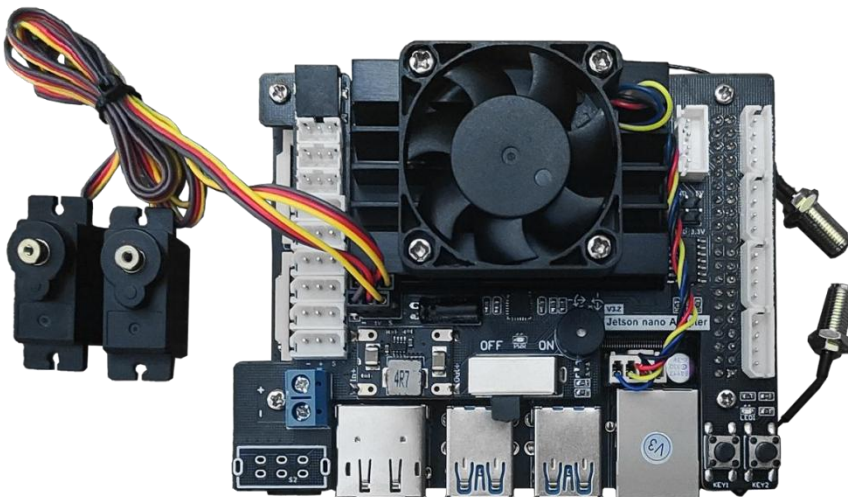
- 1) spend 0.5s rotating to 72° (1300 pulse width), and the angular velocity is 144 degree/ second.
- 2) spend 0.5s rotating to 108° (1700 pulse width), and the angular velocity is 216 degree/ second.
- 3) spend 1s rotating to 72° (1300 pulse width), and the angular velocity is 72 degree/ second.
- 4) spend 1s rotating to 108° (1700 pulse width), and the angular velocity is 108 degree/ second.

Then, the PWM servo will cycle in this way. If you press “**Ctrl+C**”, the program will be exited and the PWM servo will restore to 90° (1500 pulse width), which takes 1s.

Part 10 Control 2 PWM Servos

1. Preparation

Connect 2 PWM servos to any PWM servo port on the Jetson Nano expansion board. Take LFD-01M servo for example.



Note: please strictly distinguish the positive and negative electrode of the PWM servo. Red wire is the positive electrode, and gray wire is the negative electrode.

2. Program Logic

The program will send the pulse signal to control PWM servo to rotate. And by setting the parameters of different servo ports, several servos can be controlled to rotate at the same time.

The source code of this program is stored in

**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
pwm_servo/two_pwm_servo_demo.py**

```
#!/usr/bin/python3
# coding=utf8
import sys
import time
sys.path.append("../")
#Add the higher-level directory to the Python interpreter's search path
#so that you can import the modules in the higher-level directory.

from sdk import pwm_servo

servo1 = pwm_servo.PWMServo(1)
#Creates an instance of the pwm_servo.PWMServo class that represent a connected
servo.
#Parameter 1 represents the number or identifier of the servo.

servo2 = pwm_servo.PWMServo(2)
#Creates an instance of the pwm_servo.PWMServo class that represent a connected
servo.
#Parameter 2 represents the number or identifier of the servo.

servo1.start() #Start the servo to make it ready.
servo2.start() #Start the servo to make it ready.

print('two servo move between 1300-1700')

while True:
    try:
        position = 1300 # Set the servo position to 1300 milliseconds.
        duration = 1000 # Set the move duration to 1000 milliseconds.
        servo1.set_position(position, duration) # Call the set_position()
method to move the servo1 to the specified position and set the move duration.
        servo2.set_position(position, duration) # Call the set_position()
method to move the servo2 to the specified position and set the move duration.
        time.sleep(duration/1000.0) # The program pauses execution for a period
of time, waiting for the servo movement to complete. Here the duration is
converted to seconds
```

```

    position = 1700 # Reset the servo position to 1700.
    duration = 1000 # Set the duration of the servo motion to 1000
milliseconds.
    servo1.set_position(position, duration) # Move the servo1 to the
specified position and set the motion duration.
    servo2.set_position(position, duration) # Move the servo2 to the
specified position and set the motion duration.
    time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 500 milliseconds) of the set motion duration.

except KeyboardInterrupt:
    position = 1500 # Reset servo position to 1500.
    duration = 1000 # Set the duration of servo motion to 1000 milliseconds.
    servo1.set_position(position, duration) # Move the servo1 to the
specified position and set the motion duration.
    servo2.set_position(position, duration) # Move the servo2 to the
specified position and set the motion duration.
    time.sleep(duration/1000.0) # The program pauses execution for the
number of seconds (i.e., 1000 milliseconds) of the set motion duration.
    break

```

Take “**my_servo.set_position(position, duration)**” for example. The meaning of the parameters in bracket is explained below.


The first parameter “**position**” refers to the position to which the servo rotates i.e. pulse width. Formula: pulse width = 11.1 x angle + 500 (only for reference)

The second parameter “**duration**” is the time for the servo rotation in millisecond.

3.Operation Steps

The input command should be case sensitive, and the keywords can be complemented by “**Tab**” key.

1) Start Jetson Nano robot kit, and connect it to NoMachine

2) Double click  to open the command line terminal

3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/pwm_servo/**” and press Enter to enter the directory where the program file is kept.

4) Input command “**python3 two_pwm_servo_demo.py**” and press Enter to run the game program.

5) If you want to exit this program, please press “**Ctrl+C**”.

4. Program Outcome

After the game starts, these two PWM servos will rotate in this rule.

- 1) spend 1s rotating to 72° (1300 pulse width)
- 2) spend 1s rotating to 108° (1700 pulse width)

Then, these two PWM servos will cycle in this way. If you press “**Ctrl+C**”, the program will be exited and the PWM servos will restore to 90° (1500 pulse width), which takes 1s.

Part 11 Control Serial Interface

1. Serial Interface Communication Definition

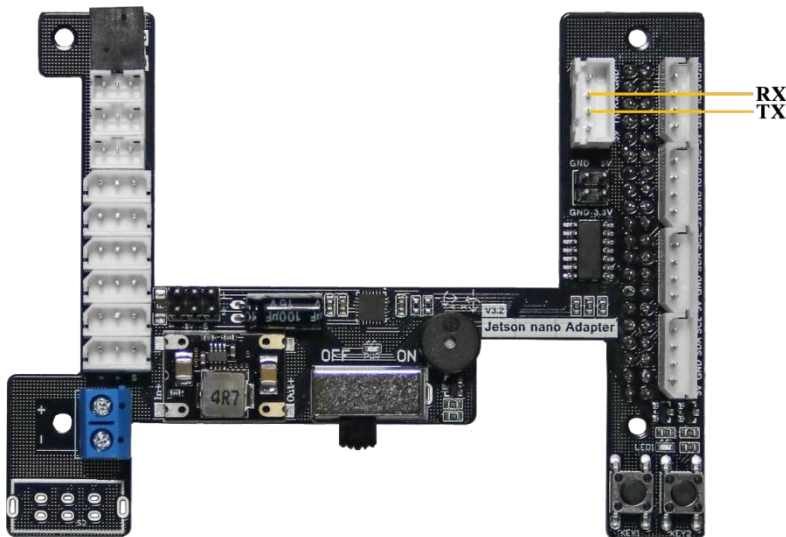
It is easy to control one LED to light up and go out by setting its voltage. But if **you want to control hundreds of LEDs, it is annoying to set the LED voltage one by one in the program.** To deal with this problem, the **serial communication is launched.**

Serial communication is a way to transfer data by bit between external devices and computers through data signal line, control line and ground line. As it can send large amount of data at one time, serial communication can handle various complicated cases of data transmission, which is similar to IIC and SPI communication.

Serial interface is also called serial communication interface. It is an expansion interface that adopts serial communication to send and receive data by bit.

2. Preparation

Connect the TX and RX interfaces on Jetson Nano expansion board together, as the below picture shown.



4. Program Logic

TX and RX communication interfaces will be employed in serial interface communication. And **TX interface will send the data**, and **RX interface will receive the data.** After connecting TX and RX interfaces together, the serial interface communication can be simulated, and the “**serial interface**” can send and receive data.

The source code of this program is located in
**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
serial_send_demo.py**

```
#!/usr/bin/env python3
# encoding: utf-8
import time
import json
import serial
import Jetson.GPIO as GPIO # Import the Jetson.GPIO module, which is used to
control the GPIO pins of the NVIDIA Jetson Nano.

# Defines the RX (receive) and TX (transmit) pin numbers.
rx_pin = 17
tx_pin = 27

def port_as_write(): # Setting GPIO pins to write mode
    GPIO.output(tx_pin, 1) # Pull up TX_CON i.e. GPIO27.
    GPIO.output(rx_pin, 0) # Pull down RX_CON i.e. GPIO17.

def port_as_read(): # Setting GPIO pins to read mode
    GPIO.output(rx_pin, 1) # Pull up RX_CON i.e. GPIO17.
    GPIO.output(tx_pin, 0) # Pull down TX_CON i.e. GPIO27.

def port_init(): # Defines a method to initialize the GPIO pins.
    GPIO.setwarnings(False) # When set to False, warning message output is
                             disabled.
    mode = GPIO.getmode() # The mode used to get the current GPIO pin number.
                           Here it returns the current GPIO pin numbering
                           mode (GPIO.BOARD or GPIO.BCM)
    if mode == 1 or mode is None: # If the current mode is GPIO.BOARD or not
                                   set (None), execute the following code block.
        GPIO.setmode(GPIO.BCM) # Set the GPIO pin numbering mode to GPIO.BCM.
                                In GPIO.BCM mode, the numbering of the
                                pins is defined based on the number of the
                                Broadcom SOC channel, not the number of
                                the physical pin.
    GPIO.setup(rx_pin, GPIO.OUT) # Configure RX_CON i.e. GPIO17 as the output.
    GPIO.output(rx_pin, 0) # Set the output level of the rx_pin pin to low (0)
                           for initializing the rx_pin GPIO pin
    GPIO.setup(tx_pin, GPIO.OUT) # Configure TX_CON i.e. GPIO27 as the output.
    GPIO.output(tx_pin, 1) # Set the output level of the tx_pin pin to high (1)
                           for initializing the tx_pin GPIO pin

port_init() # Call the port_init() method to initialize the GPIO pins.

my_serial = serial.Serial('/dev/ttyTHS1', 115200, timeout=0.01)
```

```

# Initializes a serial port object to connect to the serial device
# /dev/ttyTHS1 with a BAUD rate of 115200 and a timeout of 0.01 seconds.
""" Parameters:
    /dev/ttyTHS1 :Represents the path to the serial device
    115200 :Represents a BAUD rate of 115200 bps (bits per second), which
            is one of the common serial communication rates.
    timeout=0.01 :This means that if no data is read within 0.01 seconds
                  during a serial port read operation, a timeout will be
                  triggered. """

print('serial send hello at 10hz')

while True:
    data = b"hello\r\n" # Defines the data to be sent, using the form of a byte
                        # string.
                        # "b" denotes "Bytes" rather than "Unicode".
                        # Each element of the Byte Sequence is a byte (8
                        # bits) and can represent an integer from 0 to 255.
                        # Characters in Unicode encoding can be Multi-Byte
                        # and can represent all the characters in the
                        # world, not just the ASCII character set.
    my_serial.flushInput() # This is a method of the my_serial object that
                        # empties the serial input Buffer. Serial devices
                        # usually have an input Buffer that is used to store
                        # received data, and in some cases it may be
                        # necessary to empty this Buffer.

    port_as_write() # Call the port_as_write() method to set the GPIO pin to
                    # write mode.

    my_serial.write(data) # Methods to send data to the serial device via the
                        # serial object my_serial
                        # write(data): This is a method of the my_serial
                        # object that writes data to the serial device.

    time.sleep(0.1) # Sleep 0.1 sec, control transmit frequency to 10Hz

# Byte Sequence (Bytes)

""" # Creating a Byte Sequence String
byte_seq = b'hello'

# Printing the type and content of the Byte Sequence
print(type(byte_seq)) # <class 'bytes'>
print(byte_seq)       # b'hello'

```

```

# Get the length of the Byte Sequence
print(len(byte_seq))    # 5

# Access to a individual byte in the Byte Sequence
print(byte_seq[0])      # 104 ""

# Unicode Encoding (Unicode)

""" # Creating a Unicode Encoded String
unicode_str = '你好'

# Printing the type and content of the Unicode Encoding
print(type(unicode_str)) # <class 'str'>
print(unicode_str)       # 你好

# Get the length of the Unicode Encoding
print(len(unicode_str))  # 2

# Access to a individual byte in the Unicode Encoding
print(unicode_str[0])    # 你 ""

```

The source code of this program is located in
**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
serial_read_demo.py**

```

#!/usr/bin/env python3
# encoding: utf-8
import time
import serial
import Jetson.GPIO as GPIO

rx_pin = 17
tx_pin = 27

def port_as_write():
    GPIO.output(tx_pin, 1) # Pull up TX_CON i.e. GPIO27.
    GPIO.output(rx_pin, 0) # Pull down RX_CON i.e. GPIO17.

def port_as_read():
    GPIO.output(rx_pin, 1) # Pull up RX_CON i.e. GPIO17.
    GPIO.output(tx_pin, 0) # Pull down TX_CON i.e. GPIO27.

def port_init():
    GPIO.setwarnings(False)
    mode = GPIO.getmode()
    if mode == 1 or mode is None:
        GPIO.setmode(GPIO.BCM)
    GPIO.setup(rx_pin, GPIO.OUT) # Configure RX_CON i.e. GPIO17 as the output.

```

```

GPIO.output(rx_pin, 0)
GPIO.setup(tx_pin, GPIO.OUT) # Configure TX_CON i.e. GPIO27 as the output.
GPIO.output(tx_pin, 1)

port_init()
port_as_read() # Call the port_as_write() method to set the GPIO pin to read
                mode.

data = []
my_serial = serial.Serial('/dev/ttyTHS1', 115200, timeout=0.01)
# Initializes a serial port object to connect to the serial device /dev/ttyTHS1
with a BAUD rate of 115200 and a timeout of 0.01 seconds.


print('serial read data at 100hz')

while True:
    print(my_serial.read()) # Methods to read data from the serial device via
                            the serial object my_serial
    time.sleep(0.01) # Sleep 0.1 sec, control transmit frequency to 10Hz

```

4. Operation Steps

The input command should be case sensitive, and the keywords can be complemented by “**Tab**” key.

- 1) Start Jetson Nano robot kit, and connect it to NoMachine
- 2) Double click  to open the command line terminal
- 3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/**” and press Enter to enter the directory where the program file is kept.
- 4) Input command “**python3 serial_send_demo.py**” and press Enter to run the program to send the data through serial interface.
- 5) Then open a new terminal, and input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/**” and press Enter.
- 6) Input command “**python3 serial_read_demo.py**” and press Enter to run the program to receive the data through serial interface.
- 7) If you want to exit this program, you can press “**Ctrl+C**” on the corresponding terminal.

5. Program Outcome

After the game starts, the terminal where the program “**serial_read_demo.py**” runs will print the characters received by the serial interface

```

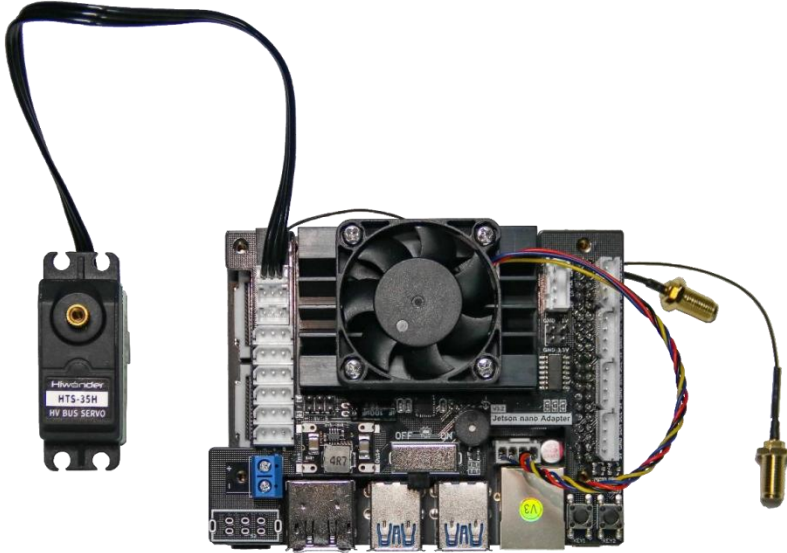
b' '
b'h'
b'e'
b'l'
b'l'
b'o'
b'\r'
b'\n'
b' '

```

Part 12 Control Serial Bus Servo

1. Preparation

Connect the serial bus servo to any serial bus servo port on the Jetson Nano expansion board. Take HTS-35H servo for example.



Note: the servo wire adopts reverse plugging protection. Please don't plug it violently.

2. Program Logic

The program controls the serial bus servo to rotate to the specific position in the designated time by sending command containing servo ID, rotation position (pulse width) and rotation time.

The source code of this program is stored in
**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
serial_servo/serial_servo_move_demo.py**

```
#!/usr/bin/env python3
# encoding: utf-8
import sys
import time
sys.path.append("..")
from sdk import hiwonder_servo_controller

servo_control =
hiwonder_servo_controller.HiwonderServoController('/dev/ttyTHS1', 115200)
# Specify the path and baud rate of the serial device. This instance will be
used to control the servo

print('serial servo move between 400 - 600')

while True:
    try:
        servo_id = 5 # Servo id(0-253)
        position = 400 # Position (0-1000)
```



```

duration = 500 # Time (20-30,000)
servo_control.set_servo_position(servo_id, position, duration)
    #Use the set_servo_position method of the servo controller instance to
    move the servo to the specified position.
time.sleep(duration/1000.0 + 0.1)
    # It takes 100ms longer to fully turn into position, because of the
    time required for data transfer.

servo_id = 5 # Servo id(0-253)
position = 600 # Position (0-1000)
duration = 500 # Time (20-30,000)
servo_control.set_servo_position(servo_id, position, duration)
    #Use the set_servo_position method of the servo controller instance to
    move the servo to the specified position.
time.sleep(duration/1000.0 + 0.1)
    # It takes 100ms longer to fully turn into position, because of the
    time required for data transfer.
except KeyboardInterrupt:
    servo_id = 5 # Servo id(0-253)
    position = 500 # Position (0-1000)
    duration = 500 # Time (20-30,000)
    servo_control.set_servo_position(servo_id, position, duration)
        #Use the set_servo_position method of the servo controller instance to
        move the servo to the specified position.
    break

```

Take “**servo_control.set_servo_position(servo_id, position, duration)**” for example. The meaning of the parameters in the bracket is as follows.

The first parameter “**servo_id**” refers to the servo ID, and the servo ID in this program is “5”. And you can change the number based on the actual case.


The second parameter “**position**” stands for the position to which the servo rotates i.e. pulse width. Formula: pulse = 4.17 x angle (only for reference)

The third parameter “**duration**” is the time taken for rotation in millisecond.

3. Set Servo ID

If you need to change the ID of the serial bus servo, you can follow the steps below.

1) Start Jetson Nano robot kit, and then connect it to NoMachine.

2) Double click  to open the command line terminal.

3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/**” and press Enter.

3) Input command “**cd**”

jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/" and press Enter.

4) Input command "**python3 set_serial_servo_status.py**" and press Enter to run the game program.

```
set serial servo status
-----
1 id
2 dev
3 save_dev
4 angle_range
5 voltage_range
6 temperature_warn
7 lock
8 help
9 exit
-----
*****current status*****
id:1
dev:0
angle_range:(0, 1000)
voltage_range:(4500, 14000)
temperature_warn:85
lock:0
*****
input mode:
```

5) Input "1", and press Enter to modify the servo ID.

6) Input the current servo ID, and press Enter.

7) Input the new servo ID, and press Enter.


After modification, the servo status will be printed on the terminal. At this time, the servo ID is changed successfully.

8) Input "9" and press Enter to exit the program

4.Operation Steps

The input command should be case sensitive, and the keywords can be complemented by "Tab" key.

1) Start Jetson Nano robot kit, and connect it to NoMachine

2) Double click  to open the command line terminal.

3) Input command "**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/**" and press Enter to enter the directory where the program file is stored

4) Input command "**python3 serial_servo_move_demo.py**" and press Enter to run the game program.

5) If you want to exit this program, you can press "**Ctrl+C**".

5.Program Outcome

After the game starts, the serial bus servo will move in this way.

1) Spend 0.5s rotating to 96° (400 pulse width)

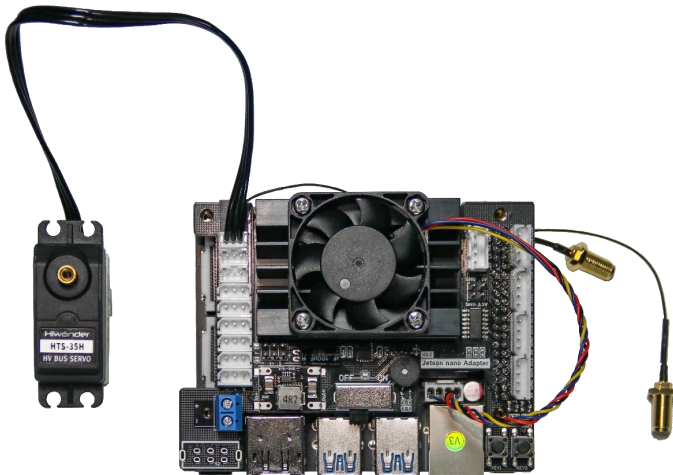
2) Spend 0.5s rotating to 144° (600 pulse width)

Then, the servo will cycle this motion. When you press “**Ctrl+C**”, this program will exit and serial bus servo will restore to 120° (500 pulse width), which takes 0.5s.

Part 13 Adjust Serial Bus Servo Speed

1. Preparation

Connect the serial bus servo to any serial bus servo port on the Jetson Nano expansion board. Take HTS-35H servo for example.



Note: the servo wire adopts reverse plugging protection. Please don't plug it violently.

2.Program Logic

The program controls the serial bus servo to rotate to the specific position in the designated time by sending command containing servo ID, rotation position (pulse width) and rotation time. And the rotation speed of the servo can be adjusted by remaining the rotation position the same and changing the rotation time.

The source code of this program is stored in
`/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/serial_servo_speed_demo.py`

```
#!/usr/bin/env python3
# encoding: utf-8
import sys
import time
sys.path.append("..")
from sdk import hiwonder_servo_controller

servo_control =
hiwonder_servo_controller.HiwonderServoController('/dev/ttyTHS1', 115200)

print('serial servo move at different speed')
```

```

while True:
    try:
        servo_id = 5 # Servo id(0-253)
        position = 400 # Position (0-1000)
        duration = 500 # Time (20-30,000)
        servo_control.set_servo_position(servo_id, position, duration)
        time.sleep(duration/1000.0 + 0.1)

        servo_id = 5
        position = 600
        duration = 500
        servo_control.set_servo_position(servo_id, position, duration)
        time.sleep(duration/1000.0 + 0.1)

        servo_id = 5 # Servo id(0-253)
        position = 400 # Position (0-1000)
        duration = 1000 # Time (20-30,000)
        servo_control.set_servo_position(servo_id, position, duration)
        time.sleep(duration/1000.0 + 0.1)

        servo_id = 5
        position = 600
        duration = 1000
        servo_control.set_servo_position(servo_id, position, duration)
        time.sleep(duration/1000.0 + 0.1)
    except KeyboardInterrupt:
        servo_id = 5
        position = 500
        duration = 500
        servo_control.set_servo_position(servo_id, position, duration)
        break

```

Take “**servo_control.set_servo_position(servo_id, position, duration)**” for example. The meaning of the parameters in the bracket is as follows.

The first parameter “**servo_id**” refers to the servo ID, and the servo ID in this program is “**5**”. And you can change the number based on the actual case.


The second parameter “**position**” stands for the position to which the servo rotates i.e. pulse width. Formula: pulse = 4.17 x angle (only for reference)

The third parameter “**duration**” is the time taken for rotation in millisecond.

3. Set Servo ID

If you need to change the ID of the serial bus servo, you can follow the steps below.

1) Start Jetson Nano robot kit, and then connect it to NoMachine.

2) Double click  to open the command line terminal.

3) Input command "**cd**

jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/" and press Enter to enter the directory where the program files are stored.

4) Input command "**python3 set_serial_servo_status.py**" and press Enter to run the game program.

5) Input "**1**", and press Enter to modify the servo ID.

6) Input the current servo ID, and press Enter.

7) Input the new servo ID, and press Enter.


After modification, the servo status will be printed on the terminal. At this time, the servo ID is changed successfully.

8) Input "**9**" and press Enter to exit the program

4. Operation Steps

The input command should be case sensitive, and the keywords can be complemented by "**Tab**" key.

1) Start Jetson Nano robot kit, and connect it to NoMachine

2) Double click  to open the command line terminal.

3) Input command **cd**

jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/" and press Enter to enter the directory where the program file is stored.

4) Input command "**python3 serial_servo_speed_demo.py**" and press Enter to run the game program.

5) If you want to exit this program, you can press "**Ctrl+C**".

5. Program Outcome

After the game starts, the serial bus servo will move in this way.

1) Spend 0.5s rotating to 96° (400 pulse width), and the angular velocity is 192 degree/ second.

2) Spend 0.5s rotating to 144° (600 pulse width), and the angular velocity is 288 degree/ second.

3) Spend 1s rotating to 96° (400 pulse width), and the angular velocity is 96 degree/ second.

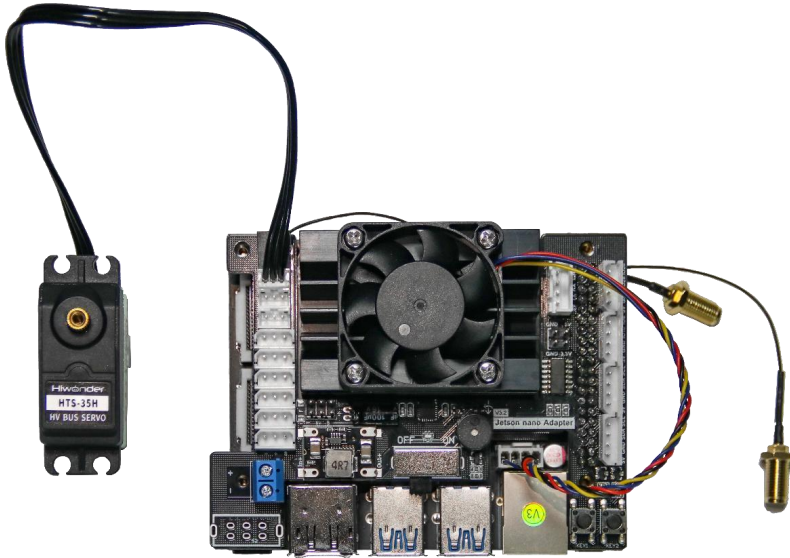
4) Spend 1s rotating to 144° (600 pulse width), and the angular velocity is 144 degree/ second.

Then, the servo will cycle this motion. You can press "**Ctrl+C**" to exit this program, and PWM servo will restore to 120°.

Part 14 Read Serial Bus Servo Status

1. Preparation

Connect the serial bus servo to any serial bus servo port on the Jetson Nano expansion board. Take HTS-35H servo for example.



Note: the servo wire adopts reverse plugging protection. Please don't plug it violently.

2. Program Logic

The program will send command to read the servo status in real time, including servo ID, rotation position and supply voltage.

The source code of this program is stored in
**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
serial_servo/serial_servo_status.py**

```
#!/usr/bin/env python3
# encoding: utf-8
import sys
import time
sys.path.append("..")
from sdk import hiwonder_servo_controller

servo_control =
hiwonder_servo_controller.HiwonderServoController('/dev/ttyTHS1', 115200)

print('get serial servo status')

while True:
    try:
        servo_id = servo_control.get_servo_id() # Call the get_servo_id()
method of the servo controller instance to get the servo ID.
```



```

    pos = servo_control.get_servo_position(servo_id) # Call the
get_servo_position() method of the servo controller instance to get the servo
position.
    dev = servo_control.get_servo_deviation(servo_id) # Call the
get_servo_deviation() method of the servo controller instance to get the servo
deviation.
    if dev > 125: # If the servo deviation is greater than 125, execute the
following.
        dev = -(0xff - (dev - 1)) # Converts the servo deviation value to a
negative number.
            # 0xff: This is a hexadecimal number representing 255 in
binary.
        angle_range = servo_control.get_servo_range(servo_id) # Call the
get_servo_range() method of the servo controller instance to get the servo's
angular range.
        vin_range = servo_control.get_servo_vin_range(servo_id) # Call the
get_servo_vin_range() method of the servo controller instance to get the
voltage range of the servo.
        temperature_warn = servo_control.get_servo_temp_range(servo_id) # Call
the get_servo_temp_range() method of the servo controller instance to get the
servo's temperature warning threshold.
        temperature = servo_control.get_servo_temp(servo_id) # Call the
get_servo_temp() method of the servo controller instance to get the current
temperature of the servo.
        vin = servo_control.get_servo_vin(servo_id) # Use the get_servo_vin()
method of the servo controller instance to get the servo voltage.
        load_state = servo_control.get_servo_load_state(servo_id) # Call the
get_servo_load_state() method of the servo controller instance to get the load
state of the servo.


    print('id:%s'%(str(servo_id).ljust(3)))
        # str(servo_id): converts the servo ID to a string type
        # .ljust(3): This is a string method that left-aligns the string
and fills the right side of the string with spaces until the length of the
string reaches the specified width.
        # %s: This is a way of formatting a string to indicate that the
following argument is inserted as a string at the %s position in the string.
Here, %s is replaced by the result returned by str(servo_id).ljust(3).
    print('pos:%s'%(str(pos).ljust(4)))
    print('dev:%s'%(str(dev).ljust(4)))
    print('angle_range:%s'%(str(angle_range).ljust(4)))
    print('voltage_range:%s'%(str(vin_range).ljust(5)))
    print('temperature_warn:%s'%(str(temperature_warn).ljust(4)))
    print('temperature:%s'%(str(temperature).ljust(4)))
    print('vin:%s'%(str(vin).ljust(4)))
    print('load:%s'%(str(load_state).ljust(4)))

```

```
print('')
except KeyboardInterrupt:
    break
```

3. Set Servo ID

If you need to change the ID of the serial bus servo, you can follow the steps below.


- 1) Start Jetson Nano robot kit, and then connect it to NoMachine.
- 2) Double click  to open the command line terminal.
- 3) Input command "**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/**" and press Enter to enter the directory where the program files are stored.
- 4) Input command "**python3 set_serial_servo_status.py**" and press Enter to run the game program.
- 5) Input "**1**", and press Enter to modify the servo ID.
- 6) Input the current servo ID, and press Enter.
- 7) Input the new servo ID, and press Enter.

After modification, the servo status will be printed on the terminal. At this time, the servo ID is changed successfully.

- 8) Input "**9**" and press Enter to exit the program

4.Operation Steps

The input command should be case sensitive, and the keywords can be complemented by "**Tab**" key

- 1) Start Jetson Nano robot kit, and then connect it to NoMachine.
- 2) Double click  to open the command line terminal.
- 3) Input command "**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/**" and press Enter to enter the directory where the program file is stored.
- 4) Input command "**python3 serial_servo_status.py**" and press Enter to run the game program.
- 5) If you want to exit this program, you can press "**Ctrl+C**".

5. Program Outcome

After the game starts, the serial bus servo will move in this way.

```
id:5
pos:498
dev:0
angle_range:(0, 1000)
voltage_range:(4500, 14000)
temperature_warn:85
temperature:34
vin:10938
lock:1
```

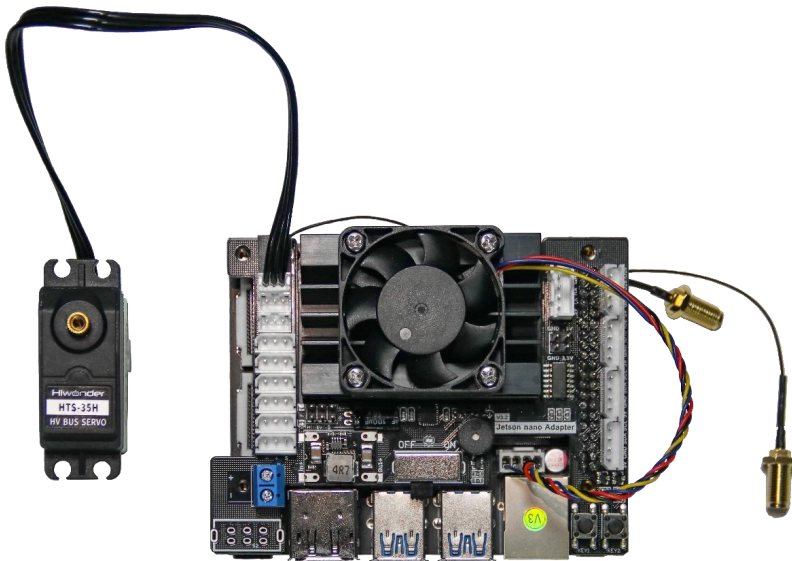
The following information will be printed.

- 1) id: servo ID
- 2) pos: the current position of the servo
- 3) dev: deviation
- 4) angle_range: rotation range
- 5) voltage_range: voltage range
- 6) temperature_warn: threshold of temperature alarm
- 7) temperature: servo temperature
- 8) vin: voltage
- 9) Lock: load or unload reading

Part 15 Battery Voltage Detection

1. Preparation

Connect the serial bus servo to any serial bus servo port on the Jetson Nano expansion board. Take HTS-35H servo for example.



Note: the servo wire adopts reverse plugging protection. Please don't plug it violently.

2. Program Logic

The program can read the servo voltage to get the battery voltage by sending the corresponding command.

The source code of this program is stored in
**/home/jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/
serial_servo/serial_servo_voltage.py**

```
#!/usr/bin/env python3
# encoding: utf-8
import sys
```

```

import time
sys.path.append("..")
from sdk import hiwonder_servo_controller

servo_control =
hiwonder_servo_controller.HiwonderServoController('/dev/ttyTHS1', 115200)


print('get serial servo voltage at 10hz')

while True:
    try:
        vol = servo_control.get_servo_vin(5)
        if vol is not None:
            print('\rvoltage: %sV'%(str(vol/1000.0).ljust(6)), end='',
flush=True)
            time.sleep(0.1)
    except KeyboardInterrupt:
        break

```

3. Set Servo ID

If you need to change the ID of the serial bus servo, you can follow the steps below.


- 1) Start Jetson Nano robot kit, and then connect it to NoMachine.
- 2) Double click  to open the command line terminal.
- 3) Input command “**cd jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/**” and press Enter to enter the directory where the program files are stored.
- 4) Input command “**python3 set_serial_servo_status.py**” and press Enter to run the game program.
- 5) Input “**1**”, and press Enter to modify the servo ID.
- 6) Input the current servo ID, and press Enter.
- 7) Input the new servo ID, and press Enter.

After modification, the servo status will be printed on the terminal. At this time, the servo ID is changed successfully.

- 8) Input “**9**” and press Enter to exit the program

4. Operation Steps

The input command should be case sensitive, and the keywords can be complemented by “**Tab**” key.

- 1) Start Jetson Nano robot kit, and then connect it to NoMachine.
- 2) Double click  to open the command line terminal.

3) Input command "**cd**

jetauto_ws/src/jetauto_example/scripts/jetauto_adapter_example/serial_servo/" and press Enter to enter the directory where the program file is stored.

4) Input command "**python3 serial_servo_voltage.py**" and press Enter to run the game program.

5) If you want to exit this program, you can press "**Ctrl+C**".

5. Program Outcome

After the game starts, the battery voltage will be printed on the terminal.