

USB Camera, Servo PTZ User Guide (Must Read)

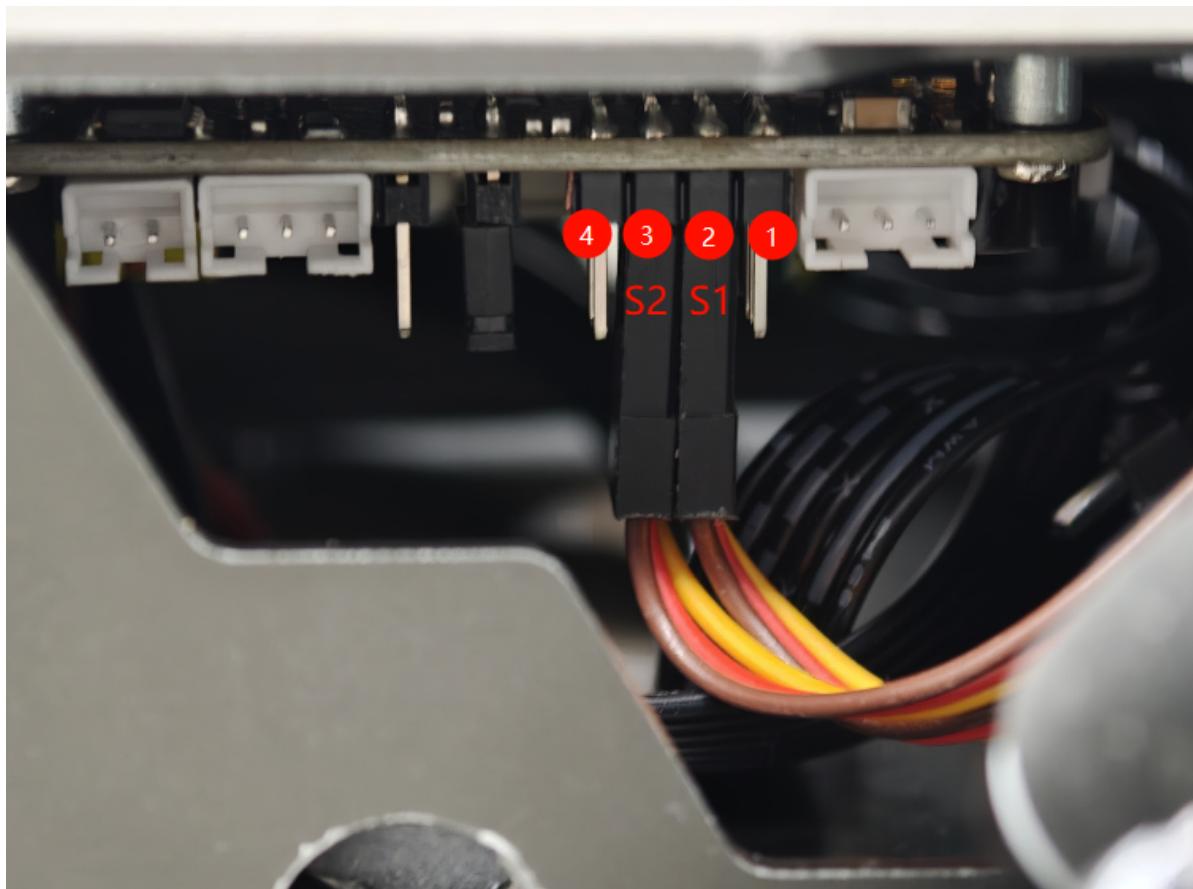
USB Camera, Servo PTZ User Guide (Must Read)

1. Installation
2. Initial Value Settings
 - 2.1 Auto-start Program Servo Initial Values
 - 2.2 Vision Case Servo Initial Values

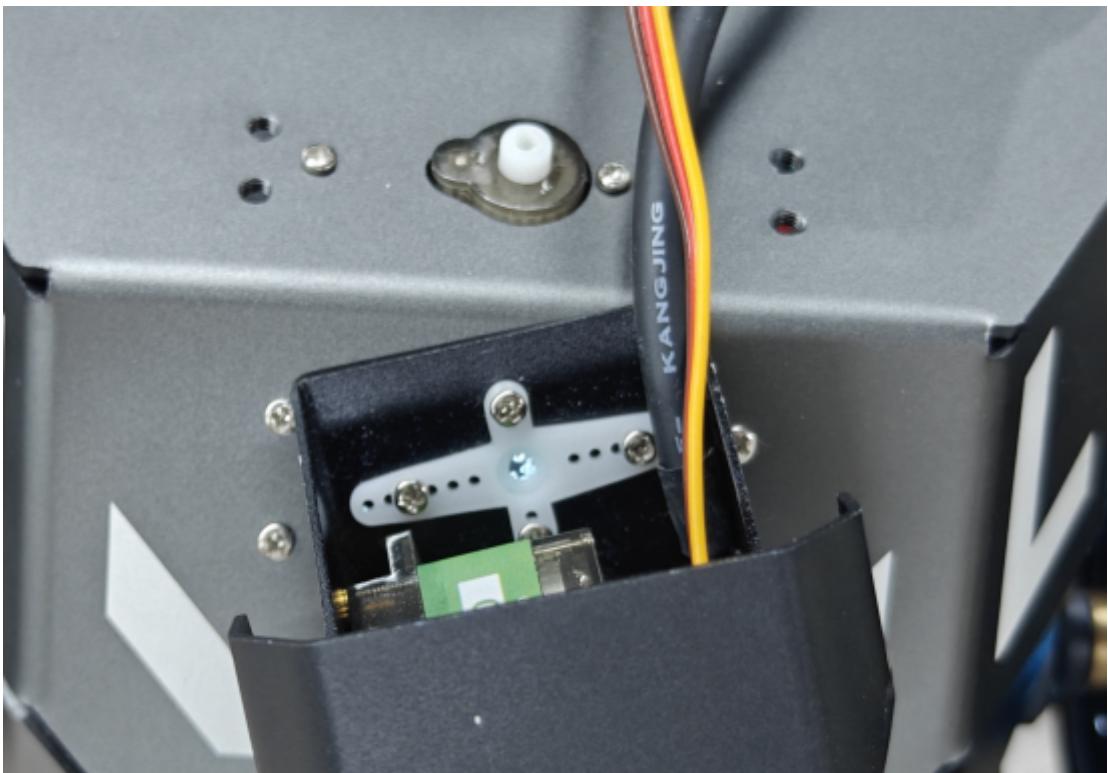
Note: For the servo PTZ of the ROSMASTER-M1 robot, we need to perform the following operations before assembly to initialize the angles.

1. Installation

1. With the robot powered off, connect the servo wiring (S1 is the horizontal servo, S2 is the vertical servo) to the corresponding connections.



2. Keep the PTZ servos disassembled initially. Power on the robot, and the S1 and S2 servos will reset to their initial angles.



3. Then install the PTZ servos. Be careful not to rotate the horizontal servo gear angle at this time. After powering off the robot, tighten the screws.



2. Initial Value Settings

2.1 Auto-start Program Servo Initial Values

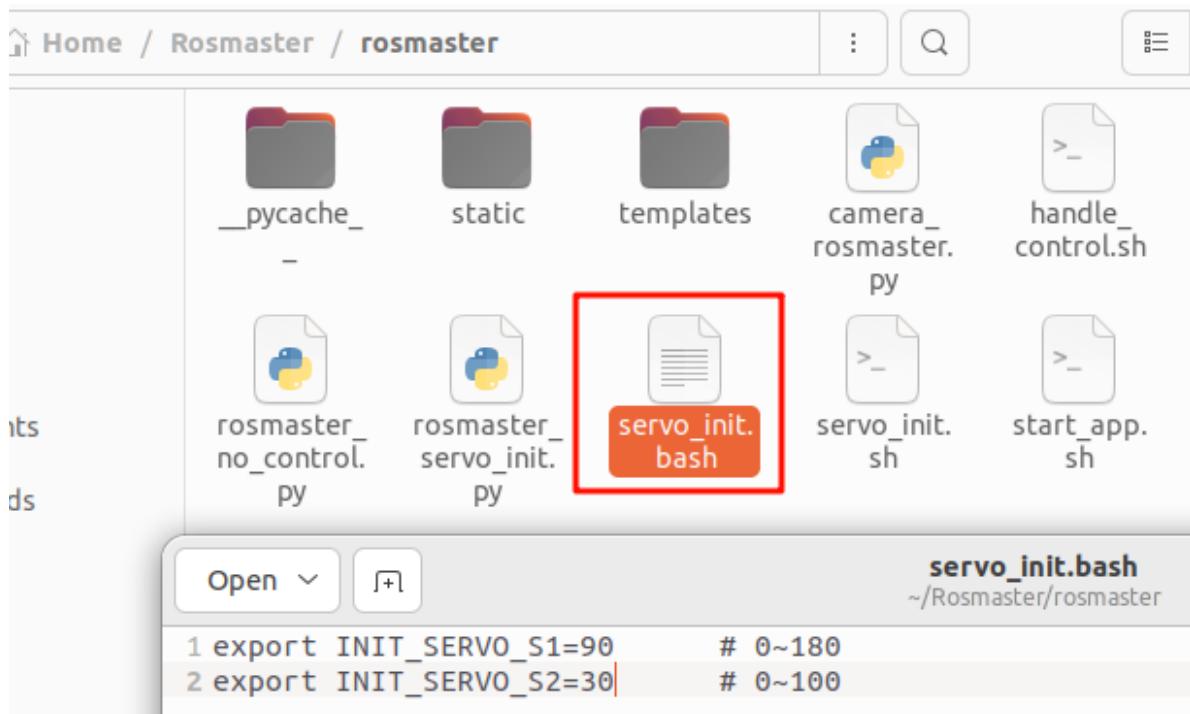
Each time the robot powers on, the ROS robot control board will set the PTZ to the initial angle (90,90). After entering the system, the `Servo Initial Position Program` will auto-start and set it to a preset angle (90,30).

⚠ Due to the servo structure itself, the gears will inevitably have an angle offset, so setting it to 90 degrees is not necessarily the camera's horizontal center angle.

Modification method:

- **ORIN Users:**

You can modify the parameters in `/home/jetson/Rosmater/rosmaster/servo_init.bash`.



☞ This configuration will be applied to the `Servo Initial Position Program` and `Boot Auto-start Handle Control Program`

- **Pi5/Jetson Nano Users:**

You can modify the parameters in `rosmaster_servo_init.py` in `/home/pi/Rosmaster/rosmaster` or `/home/jetson/Rosmater/rosmaster`.

The screenshot shows a terminal window titled "rosmaster" with the path "/home/pi/Rosmaster/rosmaster". The directory contains several files: rosmaster_.py, servo_init.srv, start_handle.desktop, start_ros2_humble.sh, and start_servo.desktop. Below the terminal is a code editor with the file "rosmaster_servo_init.py" open. The code editor has tabs for "yahboom_oled.py", "camera.launch.py", and "rosmaster_servo_init.py". The left sidebar shows project structure with "Variables", "bot [6]", "Imports", "Rosmaster [4]", and "os [5]". The main code area contains the following Python script:

```
#!/usr/bin/env python3
# coding=utf-8

from Rosmaster_Lib import Rosmaster
import os
bot = Rosmaster()
bot.set_pwm_servo(2, 90) # 0~180
bot.set_pwm_servo(3, 35) # 0~100
del bot
```

 This configuration will be applied to the [Servo Initial Position Program](#)

After entering Docker, (**if you don't understand the operation, please refer to the Docker tutorial**)

```
vim run_handle.sh
```

```
ROS_DOMAIN_ID: 61 | ROS: humble  
my_robot_type: A1 | my_lidar: tmini | my_camera: nuwa  
root@raspberrypi:~# vim run_handle.sh
```

```
root@raspberrypi: ~
File Edit Tabs Help
#!/bin/bash

bash -c "source /opt/ros/humble/setup.bash;

source /root/yahboomcar_ros2_ws/yahboomcar_ws/install/setup.bash;

echo "Controller_control";

export INIT_SERVO_S1=90;
export INIT_SERVO_S2=35;

export ROS_DOMAIN_ID=61;

ros2 launch yahboomcar_ctrl yahboomcar_joy_launch.py;"
```

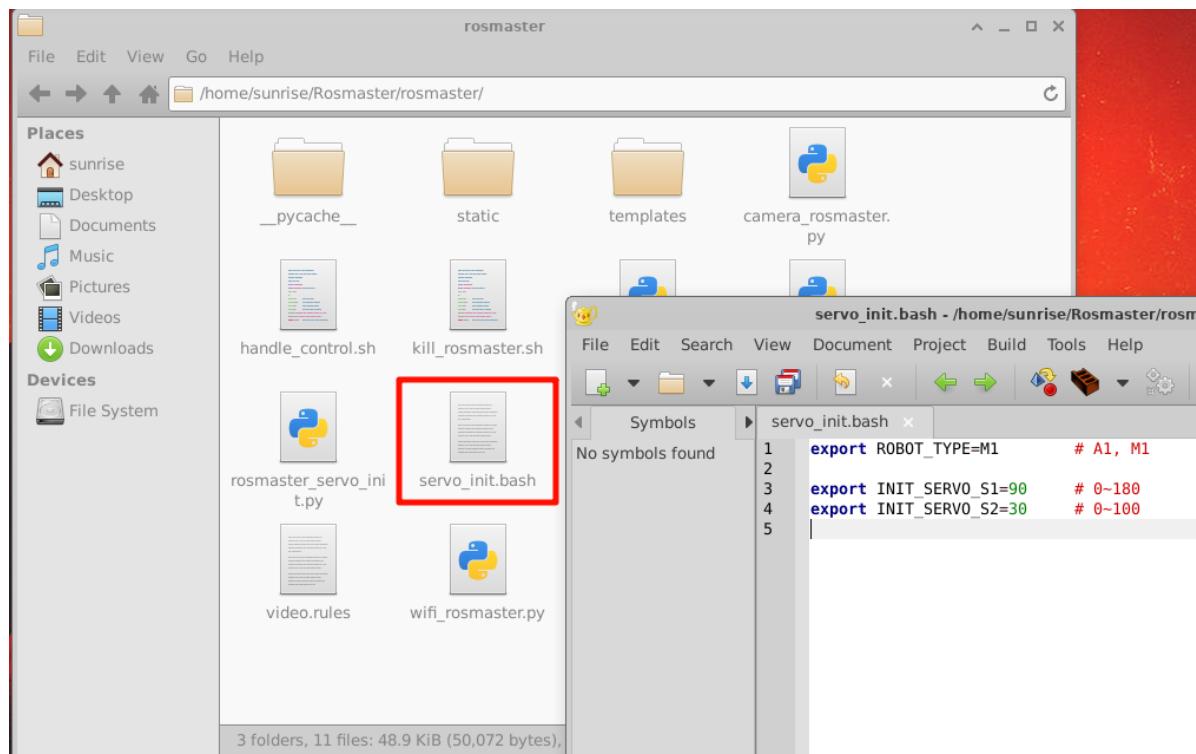
exit 0

Save the file and exit,

 This configuration will be applied to the **Boot Auto-start Handle Control Program**

- **RDK X5 Users:**

You can modify the parameters in `/home/sunrise/Rosmaster/rosmaster/servo_init.bash`.



This configuration will be applied to the `Servo Initial Position Program` and `Boot Auto-start Handle Control Program`

2.2 Vision Case Servo Initial Values

Most visual tracking/following cases will set the servo to an initial value when starting. This is done by retrieving the preset servo angle from environment variables for reset.

Modification method:

- **ORIN Users:**

```
sudo vim ~/.bashrc
```

```

jetson@yahboom: ~
jetson@yahboom: ~ 132x39

fi
fi
export LIBV4L_LOG_LEVEL=0
export PATH=/usr/local/cuda-12.6/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-12.6/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
export PYTHONPATH=/usr/local/lib/python3.10/site-packages/:$PYTHONPATH

export ROS_DOMAIN_ID=62
export ROBOT_TYPE=A1
export RPLIDAR_TYPE=c1      # c1, tmini
export CAMERA_TYPE=usb      # usb, nuwa
export INIT_SERVO_S1=90      # 0~180
export INIT_SERVO_S2=30      # 0~100

```

- Pi5/Jetson Nano Users:

After entering Docker, (if you don't understand the operation steps, please refer to the tutorial [4. Enter Docker (Jetson-Nano and Raspberry Pi 5)] in this chapter)

```
sudo vim ~/.bashrc
```

```
# . /etc/bash_completion
#fi
export ROS_DOMAIN_ID=61
export ROBOT_TYPE=A1
export RPLIDAR_TYPE=tmini      # c1, tmini
export CAMERA_TYPE=nuwa        # usb, nuwa

export INIT_SERVO_S1=90          # 0~180
export INIT_SERVO_S2=35          # 0~100

echo "-----"
echo -e "ROS_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m | \033[34mROS: $(printenv ROS_DISTRO)\033[0m"
echo -e "my_robot_type: \033[32m$ROBOT_TYPE\033[0m | my_lidar: \033[32m$RPLIDAR_TYPE\033[0m | my_camera: \033[32m$CAMERA_TYPE\033[0m"
echo "-----"
```

- RDKX5 Users:

```
sudo vim ~/.bashrc
```

```
sunrise@ubuntu: ~ 80x24
# use ssh
case $(tty 2>/dev/null) in
    /dev/tty[A-z]*) [ -x /usr/bin/resize_tty ] && /usr/bin/resize_tty >/dev/null;;
esac
#
export ROS_DOMAIN_ID=64
export ROBOT_TYPE=M1      # A1, M1
export RPLIDAR_TYPE=c1    # c1, tmini
export CAMERA_TYPE=usb    # usb, nuwa

export INIT_SERVO_S1=90    # 0~180
export INIT_SERVO_S2=35    # 0~100

source /opt/ros/humble/setup.bash
source /home/sunrise/yahboomcar_ros2_ws/software/library_ws/install/setup.bash
source /home/sunrise/yahboomcar_ros2_ws/yahboomcar_ws/install/setup.bash

# Read the number of rows and columns from the terminal size
read -r rows cols < <(stty size)

# Define the system information text
SystemInfo="System Information"
```

Enter :wq to save the file and exit,

☞ This configuration will be applied to visual Tracking/Following Cases