1.Module introduction and port binding

1.1. Introduction to voice control module

1.1.1.CSK4002 chip

The voice control module on ROSMASTER is developed based on the **CSK4002** chip. **CSK4002** is a high-performance, powerful computing power, low power consumption, and resource-rich AISoC developed and designed for the AIoT field. It can be widely used in smart homes, smart appliances, and emerging consumer electronics industries.

- CSK4002 uses the Andes D1088 core. Its AI/DSP acceleration module MVA supports a variety of Neural Network operators and vector operations. It is deeply adapted to the iFlytek AI algorithm of HKUST and has a computing power of up to 128GOPS.
- Comes with 8M Flash, 8M PSRAM, 1M SRAM.
- Supports 8 channels of PDM audio input and 16 channels of I2S Audio Input data processing.
- Integrate rich mainstream peripheral interfaces: GPIO/UART/I2C/SPI/QSPI/SDIO/USB1.1/SDIO, etc.
- Equipped with low-latency embedded operating system Free RTOS, complete BSP driver, and comprehensive development tool resources.

1.1.2. Module features

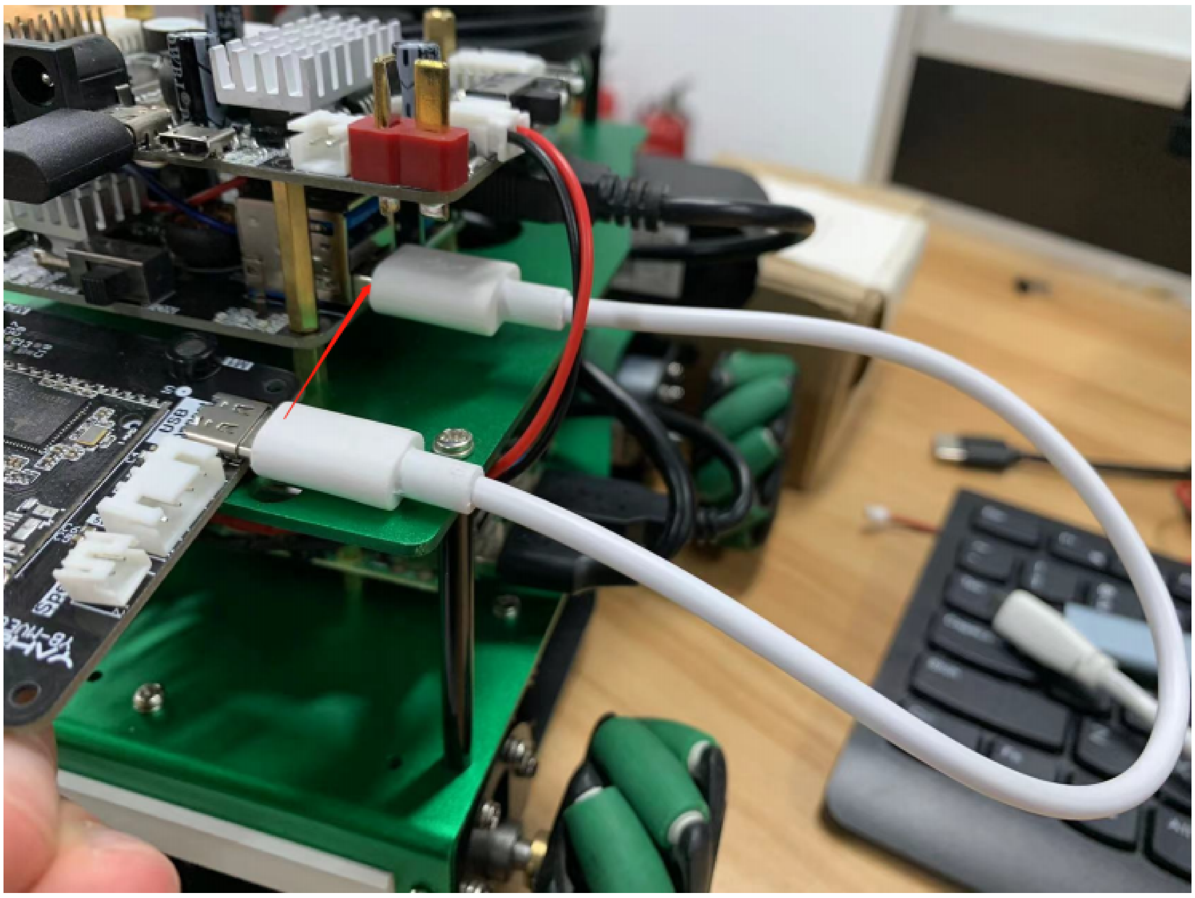
- Far-field sound pickup: The front end adopts iFlytek's dual-microphone array algorithm, which can achieve 360-degree far-field 5m user sound pickup. Equipped with automatic gain for vocals, which can be adaptively adjusted according to the user's volume to ensure that the overall audio experience is consistent after noise reduction.
- Echo cancellation: During user interaction, when the device is broadcasting content or music, the user can wake up and interrupt the broadcast process for the next round of interaction, making the interaction experience more natural.
- Voice broadcast: Voice broadcast means that the user wakes up the device and speaks command words, and the device performs corresponding reply broadcast responses; or active prompts. The purpose of voice broadcast is to give feedback to the user by broadcasting a reply when the user issues a voice command or in an appropriate scenario.
- Offline command: When the device is awake, the user speaks a command word (instruction) within a specified range. After receiving the information, the voice module performs related processing based on the content of the command word, or transmits the content information to the host computer for related processing.
- Peripheral communication: The module receives input from the microphone for processing, and then communicates with other devices through USB, I2S, SDIO and other interfaces. There is also general programmable IO to communicate with other devices.
- Environmental noise reduction: It is widely used for environmental noise reduction in homes, cars, offices and other scenes. It can reduce noise while retaining human voice information to the greatest extent.

1.2. Use module

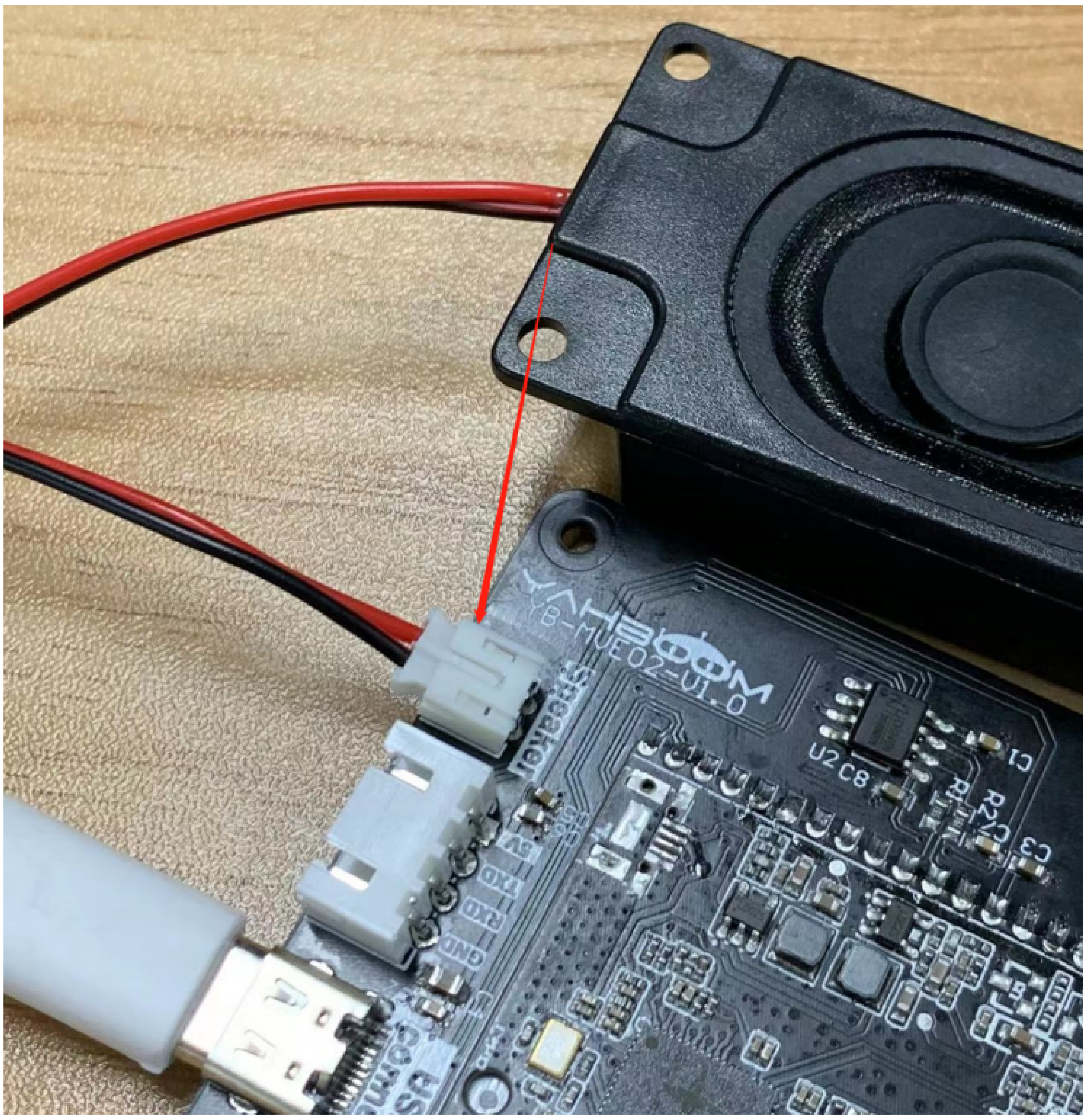
1.2.1. Wiring

The module is connected to the ROSMASTER main control (or HUB board) through a universal Type-c data cable, and the speaker and module are connected through a PH2.0 data cable.

- As shown in the figure below, one end of the Micro-USB data cable is connected to the USB-connet port of the module, and the other end is connected to the port of the ROSMASTER motherboard (*After this is plugged in, it cannot be modified after the later ports are bound* , please refer to the next section for details)



- As shown in the figure below, the PH2.0 data cable port is connected to the Speaker port on the module.



1.2.2. Wake-up word

The wake-up word is "Hello, yahboom". When waking up, the speaking speed needs to be slowed down. If it is too fast, the module will not recognize it. After waking up the module, other command words can be recognized later. Within 20 seconds of waking up, there is no need to wake up again, just say the command word.

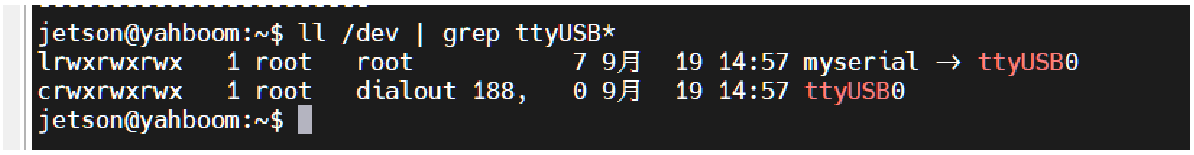
1.3. Voice control module port binding

Foreword: Because the ID device numbers of the ROS expansion board and the voice control module are the same, the ID device numbers cannot be bound to them according to the method in the previous tutorial.At this time, you need to bind the USB port. The previously bound serial port and lidar device also need to be bound in the new way.After binding, the USB port cannot be changed at will, and each device can only be connected to a fixed USB port. The specific operations are as follows:

1.3.1. Rebind serial device

1. First disconnect all devices and the main control, reconnect the serial device, open the host terminal and enter:

```
ll /dev | grep ttyUSB*
```

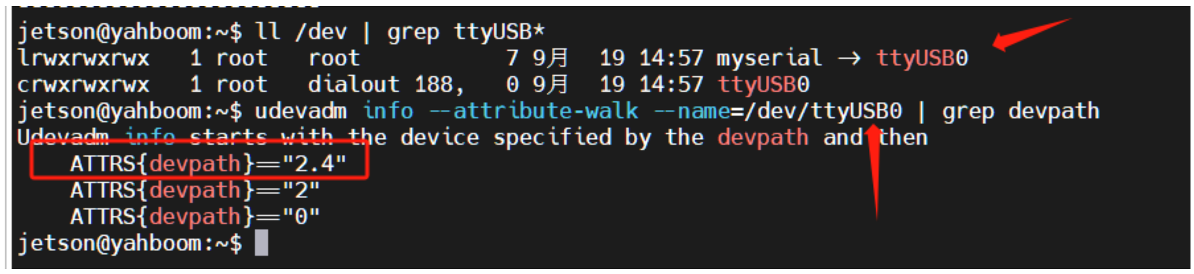


```
jetson@yahboom:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root    root          7 9月   19 14:57 myserial -> ttyUSB0
crwxrwxrwx   1 root    dialout 188,   0 9月   19 14:57 ttyUSB0
jetson@yahboom:~$
```

2. Then, we check the port information of the serial port, mainly to check the device path information. Enter the host terminal:

```
udevadm info --attribute-walk --name=/dev/ttyUSB0 | grep devpath   # Note that
ttyUSB0 here must be consistent with the device number queried above.
```

Get the following information (subject to the actual situation you inquired here), the red box indicates the path information of the device:



```
jetson@yahboom:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root   root         7 9月   19 14:57 myserial -> ttyUSB0
crwxrwxrwx   1 root   dialout 188,   0 9月   19 14:57 ttyUSB0
jetson@yahboom:~$ udevadm info --attribute-walk --name=/dev/ttyUSB0 | grep devpath
Udevadm info starts with the device specified by the devpath and then
    ATTRS{devpath}=="2.4"
    ATTRS{devpath}=="2"
    ATTRS{devpath}=="0"
jetson@yahboom:~$
```

3. Edit the usb.rules file on the host machine as follows:

```
cd /etc/udev/rules.d/
sudo vim usb.rules
```

The original statement to bind the serial port is:

```
KERNEL=="ttyUSB*", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523",
MODE:="0777", SYMLINK+="myserial"
```

Add ATTRS{devpath}=="2.4" and modify it to:

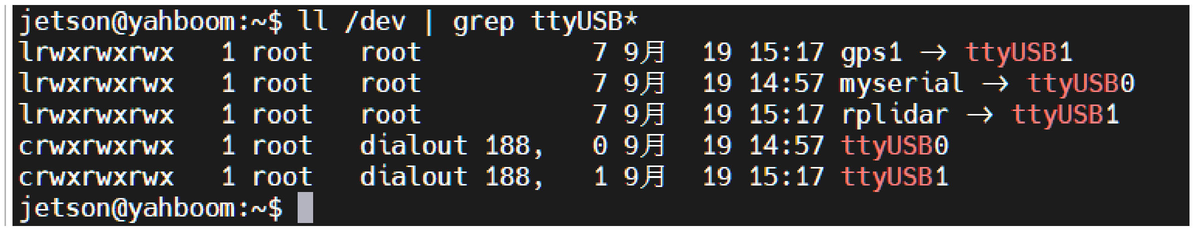
```
KERNEL=="ttyUSB*", ATTRS{devpath}=="2.4", ATTRS{idVendor}=="1a86",
ATTRS{idProduct}=="7523", MODE:="0777", SYMLINK+="myserial"
```

At this time, the serial port has been bound in the new way.

1.3.2. Rebind lidar device

1. Rebind the lidar device in the same way, connect the lidar device, open the host terminal and enter:

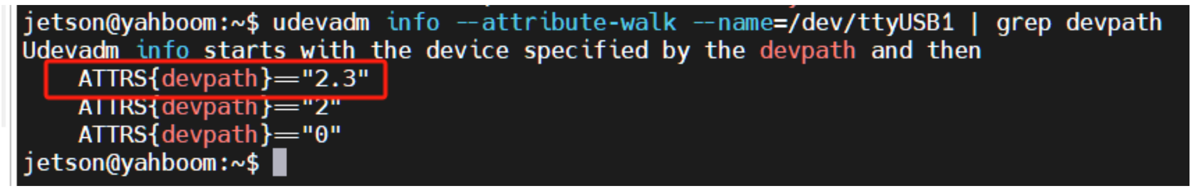
```
ll /dev | grep ttyUSB*
```



```
jetson@yahboom:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root   root          7 9月   19 15:17 gps1 -> ttyUSB1
lrwxrwxrwx   1 root   root          7 9月   19 14:57 myserial -> ttyUSB0
lrwxrwxrwx   1 root   root          7 9月   19 15:17 rplidar -> ttyUSB1
crwxrwxrwx   1 root   dialout 188,  0 9月   19 14:57 ttyUSB0
crwxrwxrwx   1 root   dialout 188,  1 9月   19 15:17 ttyUSB1
jetson@yahboom:~$
```

2. The lidar device number queried here is ttyUSB1. Enter the terminal:

```
udevadm info --attribute-walk --name=/dev/ttyUSB1 | grep devpath  #Note that
this is ttyUSB1, modify it according to your query.
```



```
jetson@yahboom:~$ udevadm info --attribute-walk --name=/dev/ttyUSB1 | grep devpath
Udevadm info starts with the device specified by the devpath and then
    ATTRS{devpath}=="2.3"
    ATTRS{devpath}=="2"
    ATTRS{devpath}=="0"
jetson@yahboom:~$
```

3. Edit the usb.rules file on the host machine as follows:

```
cd /etc/udev/rules.d/
sudo vim usb.rules
```

The original statement to bind the serial port is:

```
KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60",
MODE:="0777", SYMLINK+="rplidar"
```

Add ATTRS{devpath}=="2.3" and modify it to:

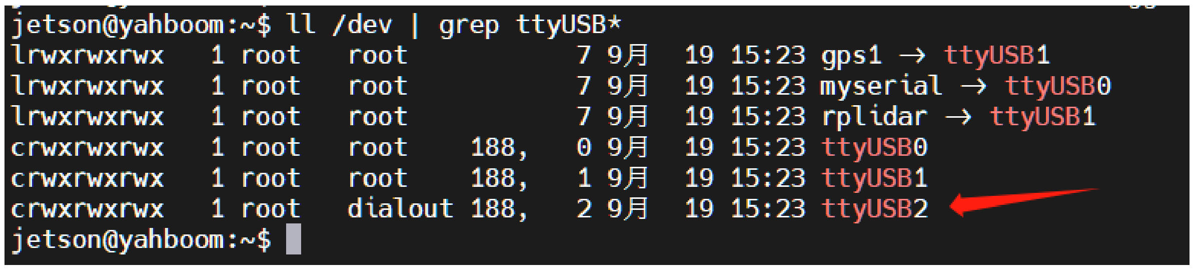
```
KERNEL=="ttyUSB*", ATTRS{devpath}=="2.3", ATTRS{idVendor}=="10c4",
ATTRS{idProduct}=="ea60", MODE:="0777", SYMLINK+="rplidar"
```

At this time, the lidar has been bound in a new way.

1.3.3. Bind voice control module

1. Bind the voice control module in the same way, connect the voice control module, open the host terminal and input:

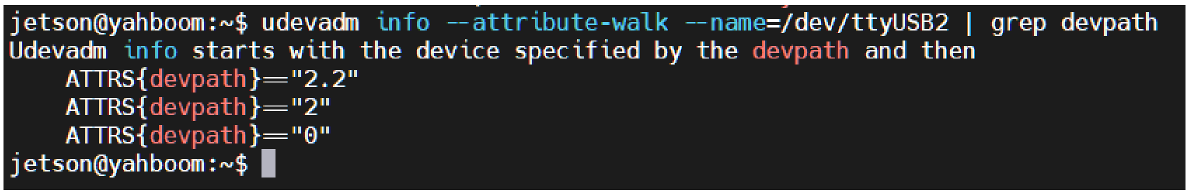
```
ll /dev | grep ttyUSB*
```



```
jetson@yahboom:~$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root   root          7 9月   19 15:23 gps1 -> ttyUSB1
lrwxrwxrwx   1 root   root          7 9月   19 15:23 myserial -> ttyUSB0
lrwxrwxrwx   1 root   root          7 9月   19 15:23 rplidar -> ttyUSB1
crwxrwxrwx   1 root   root    188,  0 9月   19 15:23 ttyUSB0
crwxrwxrwx   1 root   root    188,  1 9月   19 15:23 ttyUSB1
crwxrwxrwx   1 root   dialout 188,  2 9月   19 15:23 ttyUSB2
jetson@yahboom:~$
```

2. The voice control module device number queried here is ttyUSB2, and the terminal input:

```
udevadm info --attribute-walk --name=/dev/ttyUSB2 | grep devpath  #Note that
this is ttyUSB2, modify it according to your query.
```



```
jetson@yahboom:~$ udevadm info --attribute-walk --name=/dev/ttyUSB2 | grep devpath
Udevadm info starts with the device specified by the devpath and then
    ATTRS{devpath}=="2.2"
    ATTRS{devpath}=="2"
    ATTRS{devpath}=="0"
jetson@yahboom:~$
```

3. Edit the usb.rules file on the host machine as follows:

```
cd /etc/udev/rules.d/
sudo vim usb.rules
```

Add the statement binding the voice control module:

```
KERNEL=="ttyUSB*",ATTRS{devpath}=="2.2", ATTRS{idVendor}=="1a86",
ATTRS{idProduct}=="7523", MODE:="0777", SYMLINK+="myspeech"
```

At this point, the serial port, lidar, and voice control devices are all bound. The complete content of the "usb.rules" file is as follows:

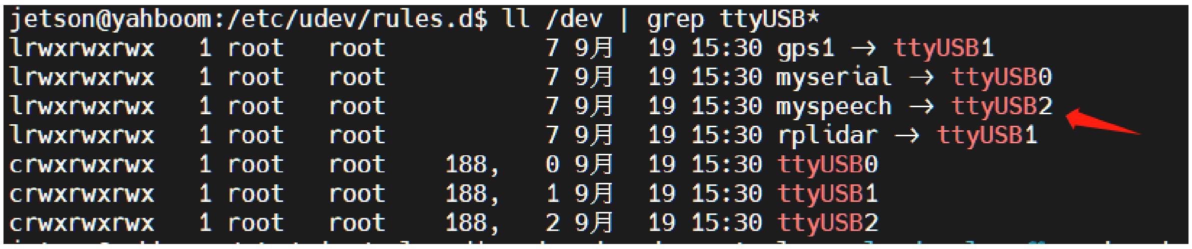
```
KERNEL=="ttyUSB*",ATTRS{devpath}=="2.4", ATTRS{idVendor}=="1a86",
ATTRS{idProduct}=="7523", MODE:="0777", SYMLINK+="myserial"
KERNEL=="ttyUSB*",ATTRS{devpath}=="2.3", ATTRS{idVendor}=="10c4",
ATTRS{idProduct}=="ea60", MODE:="0777", SYMLINK+="rplidar"
KERNEL=="ttyUSB*",ATTRS{devpath}=="2.2", ATTRS{idVendor}=="1a86",
ATTRS{idProduct}=="7523", MODE:="0777", SYMLINK+="myspeech"
```

Execute the following statement to make the binding effective:

```
sudo udevadm control --reload-rules && sudo udevadm trigger
```

Execute again:

```
ll /dev | grep ttyUSB*
```



```
jetson@yahboom:/etc/udev/rules.d$ ll /dev | grep ttyUSB*
lrwxrwxrwx   1 root    root           7 9月   19 15:30 gps1 -> ttyUSB1
lrwxrwxrwx   1 root    root           7 9月   19 15:30 myserial -> ttyUSB0
lrwxrwxrwx   1 root    root           7 9月   19 15:30 myspeech -> ttyUSB2
lrwxrwxrwx   1 root    root           7 9月   19 15:30 rplidar -> ttyUSB1
crwxrwxrwx   1 root    root     188,   0 9月   19 15:30 ttyUSB0
crwxrwxrwx   1 root    root     188,   1 9月   19 15:30 ttyUSB1
crwxrwxrwx   1 root    root     188,   2 9月   19 15:30 ttyUSB2
```

You can see that the serial port, lidar, and voice control devices have all been bound.

Note: The bound serial port and voice board cannot be plugged into other ports, otherwise the device number will not be recognized.If both are connected to the HUB expansion board, the HUB board cannot be plugged into other motherboard ports.

1.3.4. Mount voice control device in docker

After binding the voice board, when entering the container, you need to mount /dev/myspeech to recognize the voice board in docker. The mounting method is as follows:

The original content of the docker startup script [run_docker.sh] is as follows:

```
#!/bin/bash
xhost +

docker run -it \
--net=host \
--env="DISPLAY" \
--env="QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v ~/temp:/root/yahboomcar_ros2_ws/temp \
-v /dev/bus/usb/001/010:/dev/bus/usb/001/010 \
-v /dev/bus/usb/001/011:/dev/bus/usb/001/011 \
--device=/dev/astradepth \
--device=/dev/astrauvc \
--device=/dev/video0 \
--device=/dev/myserial \
--device=/dev/rplidar \
--device=/dev/input \
-p 9090:9090 \
-p 8888:8888 \
yahboomtechnology/ros-foxy-muto:2.5.0 /bin/bash
```

/dev/myspeech needs to be mounted in the startup script:

```
--device=/dev/myspeech      # Add a voice control device to the container. If
Muto is not connected to a voice control device, please remove this line.
```

The newly modified [run_docker.sh] content is as follows:

```
#!/bin/bash
xhost +

docker run -it \
--net=host \
--env="DISPLAY" \
--env="QT_X11_NO_MITSHM=1" \
-v /tmp/.X11-unix:/tmp/.X11-unix \
-v ~/temp:/root/yahboomcar_ros2_ws/temp \
-v /dev/bus/usb/001/010:/dev/bus/usb/001/010 \
-v /dev/bus/usb/001/011:/dev/bus/usb/001/011 \
--device=/dev/astradepth \
--device=/dev/astrauvc \
--device=/dev/video0 \
--device=/dev/myserial \
--device=/dev/rplidar \
--device=/dev/myspeech \                #Add this
--device=/dev/input \
-p 9090:9090 \
-p 8888:8888 \
yahboomtechnology/ros-foxy-muto:2.5.0 /bin/bash
```

Then the container started with this script can recognize the voice control device.