

1. Module introduction and port binding usage

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 AI SoC 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

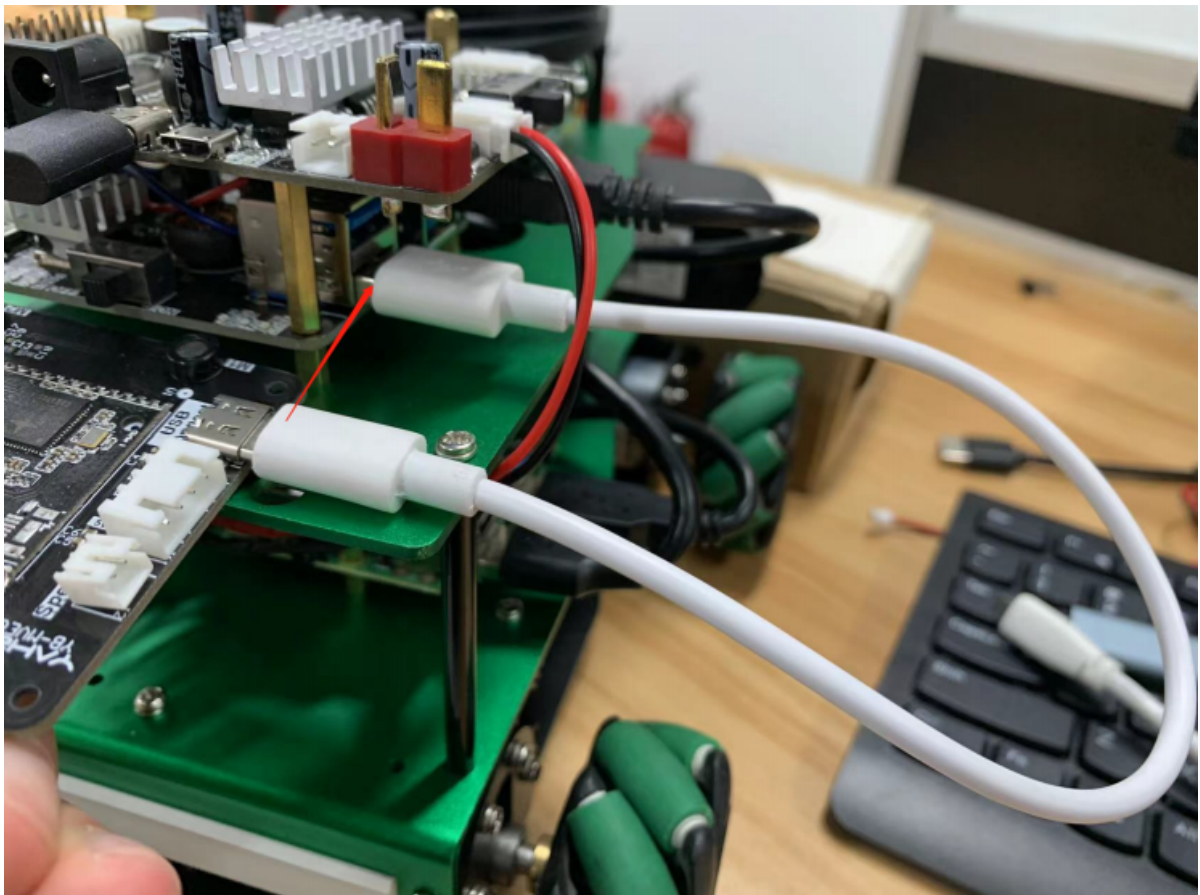
- Far-field sound pickup: The front end adopts iFlytek's dual-microphone array algorithm, which can achieve 360-degree far-field 5m user sound pickup. Equipped with automatic gain for vocals, which can be adaptively adjusted according to the user's volume to ensure that the overall audio experience is 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. Using modules

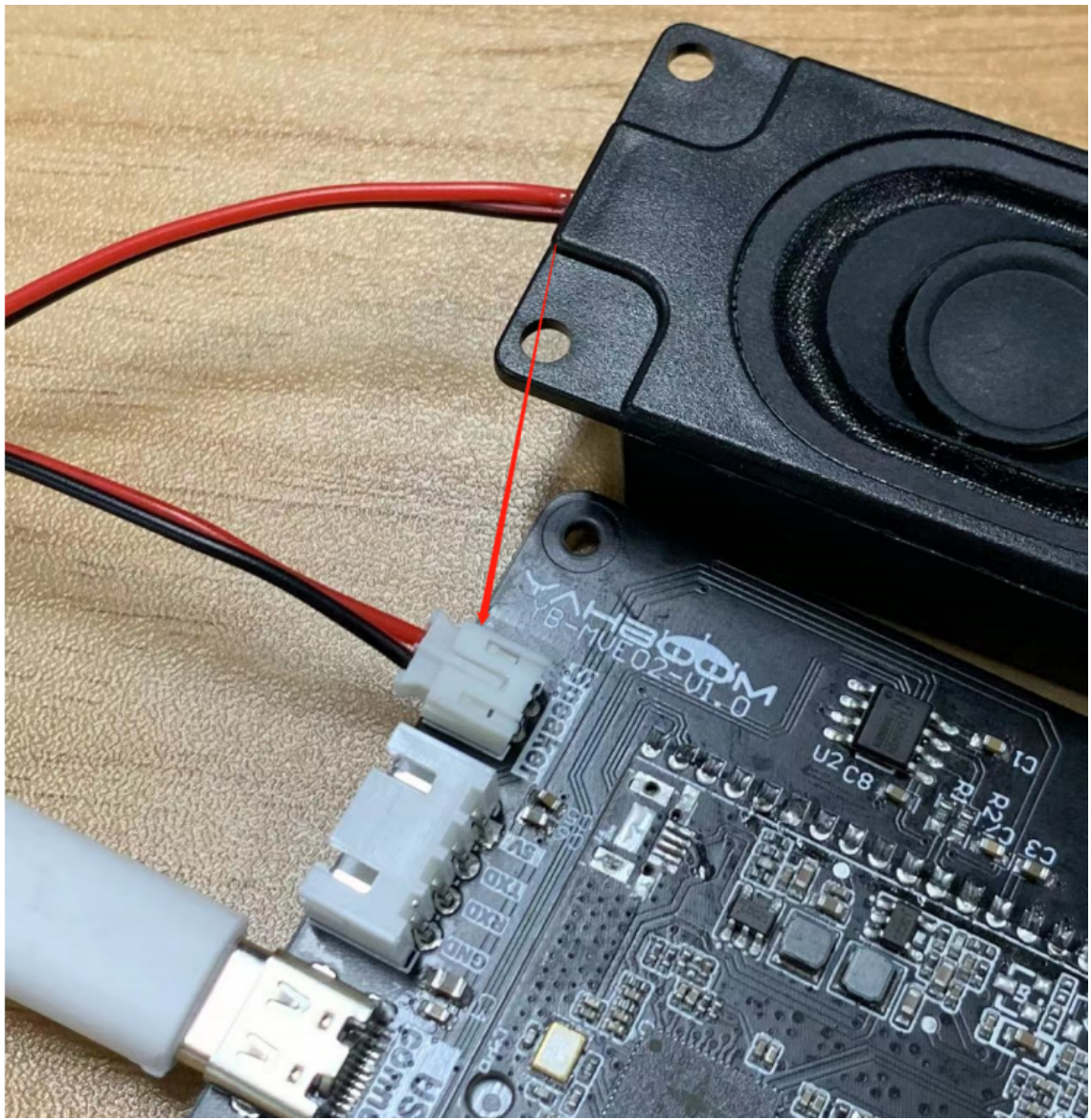
1.2.1. Wiring

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 picture 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 line port is connected to the Speaker port on the module.



1.2.2. Wake-up word

The wake-up word is "Hello, Xiaoya". 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.

1.2.3. Command words

1. Voice control car movement

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
小车停车	\$B002#	\$A002#	好的，已停止
小车前进	\$B004#	\$A004#	好的，正在前进
小车后退	\$B005#	\$A005#	好的，正在后退
小车左转	\$B006#	\$A006#	好的，正在向左转
小车右转	\$B007#	\$A007#	好的，正在向右转
小车左旋	\$B008#	\$A008#	好的，正在向左旋转
小车右旋	\$B009#	\$A009#	好的，正在向右旋转

2. Voice control RGB light strip effect

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
关灯	\$B010#	\$A010#	好的，已关灯
亮红灯	\$B011#	\$A011#	好的，已亮红灯
亮绿灯	\$B012#	\$A012#	好的，已亮绿灯
亮蓝灯	\$B013#	\$A013#	好的，已亮蓝灯
亮黄灯	\$B014#	\$A014#	好的，已亮黄灯
打开流水灯	\$B015#	\$A015#	好的，已打开流水灯
打开渐变灯	\$B016#	\$A016#	好的，已打开渐变灯
打开呼吸灯	\$B017#	\$A017#	好的，已打开呼吸灯
显示电量	\$B018#	\$A018#	好的，已显示电量

3. Voice control color recognition

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
这是什么颜色	\$B060#	\$A061#	这是红色
这是什么颜色	\$B060#	\$A062#	这是蓝色
这是什么颜色	\$B060#	\$A063#	这是绿色
这是什么颜色	\$B060#	\$A064#	这是黄色

4. Voice control color tracking

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
开始追踪黄色	\$B072#	\$A072#	好的，开始追踪黄色
开始追踪红色	\$B073#	\$A073#	好的，开始追踪红色
开始追踪绿色	\$B074#	\$A074#	好的，开始追踪绿色
开始追踪蓝色	\$B075#	\$A075#	好的，开始追踪蓝色
取消追踪	\$B076#	\$A076#	好的，取消追踪

5. Voice-controlled autonomous driving (line patrol)

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
关闭巡线	\$B022#	\$A022#	好的，已关闭巡线功能
巡红线	\$B023#	\$A023#	好的，已开启巡红线功能
巡绿线	\$B024#	\$A024#	好的，已开启巡绿线功能
巡蓝线	\$B025#	\$A025#	好的，已开启巡蓝线功能
巡黄线	\$B026#	\$A026#	好的，已开启巡黄线功能

6), Voice control multi-point navigation

语音识别内容	语音模块发送给主机	主机发送给语音模块	语音播报内容
导航去一号位点	\$B019#	\$A019#	好的，正在去一号位
导航去二号位点	\$B020#	\$A020#	好的，正在去二号位
导航去三号位点	\$B021#	\$A021#	好的，正在去三号位
导航去四号位点	\$B022#	\$A022#	好的，正在去四号位
回到原点	\$B023#	\$A023#	好的，正在回到原点

1.3. Voice control module port binding

Preface: Because the ID device numbers of the ROS expansion board and the voice control module are the same, they cannot be modified according to the method in the previous tutorial.

Row ID device number binding. **Not binding ports may cause port conflicts or device identification errors, and the bound ports cannot be changed at will**

Change position, otherwise the binding will be invalid. This course is completed on the host machine. There is no need to bind it in the dock. This section uses the Raspberry Pi as the host machine to demonstrate. Before binding, you need to annotate the part involving myserial in the original host rule file. off, otherwise the voice board and the ROS expansion board will be recognized as the same device, and the terminal input will

```
sudo gedit /etc/udev/rules.d/serial.rules
```

```
KERNEL=="ttyUSB*", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", MODE=="0777", SYMLINK+="rplidar"
#KERNEL=="ttyUSB*", ATTRS{idVendor}=="1a86", ATTRS{idProduct}=="7523", MODE=="0777", SYMLINK+="myserial"
```

As shown in the figure above, annotate the binding content related to myserial, save and exit, enter the following command in the terminal to reload the rules,

```
sudo udevadm trigger
sudo service udev reload
sudo service udev restart
```

1.3.1. Bind the ROS expansion board device number

```
ll /dev/ttyUSB*
ll /dev/rplidar
```

- First, without connecting to the voice control board, you can get ttyUSB0 and ttyUSB1

```
pi@ubuntu: ~ 80x24
pi@ubuntu:~$ ll /dev/ttyUSB*
crwxrwxrwx 1 root dialout 188, 0 May 16 16:26 /dev/ttyUSB0
crwxrwxrwx 1 root dialout 188, 1 May 16 16:26 /dev/ttyUSB1
pi@ubuntu:~$ ll /dev/rplidar
lrwxrwxrwx 1 root root 7 May 16 16:26 /dev/rplidar -> ttyUSB0
pi@ubuntu:~$
```

Here ttyUSB0 is the device number recognized by the radar, then ttyUSB1 is the device number of the ROS expansion board.

- Then, we first check the port information of the ROS expansion board, mainly to check the device path information and terminal input.

```
udevadm info --attribute-walk --name=/dev/ttyUSB1 |grep devpath
```

The following information is obtained (the actual situation shall prevail here). The red box illustrates the path information of the device.

```
pi@yahboom:~$ udevadm info --attribute-walk --name=/dev/ttyUSB1 |grep devpath
udevadm info starts with the device specified by the devpath and then
ATTRS{devpath}=="1.4.3"
ATTRS{devpath}=="1.4"
ATTRS{devpath}=="1"
ATTRS{devpath}=="0"
```

- Then, we modify the /etc/udev/rules.d/serial.rules file, first bind the port number of the ROS expansion board, and enter it in the terminal.

```
sudo gedit /etc/udev/rules.d/serial.rules
```

As shown below, replace the content

Before replacement

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

After replacement

```
KERNEL=="ttyUSB*",ATTRS{devpath}=="1.4.3",ATTRS{idVendor}=="1a86",ATTRS{idProduct}=="7523",MODE=="0777",SYMLINK+=" myserial"
```

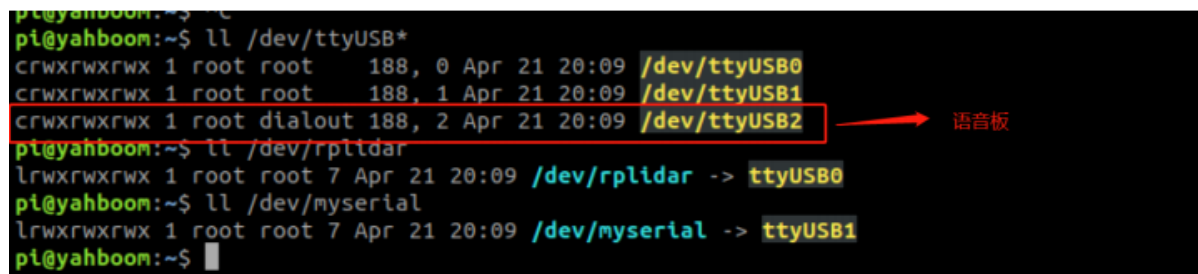
- Exit after saving, enter the following three commands in the terminal to reload the device,

```
sudo udevadm trigger
sudo service udev reload
sudo service udev restart
```

1.3.2. Bind the voice board port number

- Connect to the voice board and enter the following command on the terminal to view the device number.

```
ll /dev/ttyUSB*
```



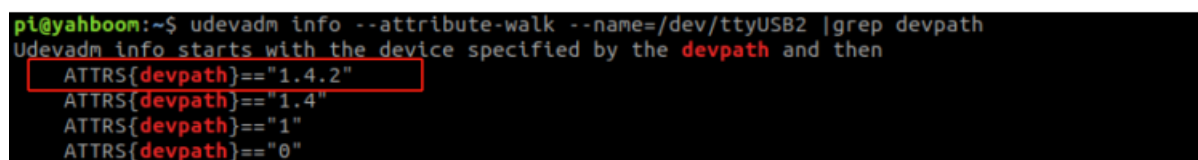
```
pi@yahboom:~$ ll /dev/ttyUSB*
crwxrwxrwx 1 root root 188, 0 Apr 21 20:09 /dev/ttyUSB0
crwxrwxrwx 1 root root 188, 1 Apr 21 20:09 /dev/ttyUSB1
crwxrwxrwx 1 root dialout 188, 2 Apr 21 20:09 /dev/ttyUSB2
pi@yahboom:~$ ll /dev/rplidar
lrwxrwxrwx 1 root root 7 Apr 21 20:09 /dev/rplidar -> ttyUSB0
pi@yahboom:~$ ll /dev/myserial
lrwxrwxrwx 1 root root 7 Apr 21 20:09 /dev/myserial -> ttyUSB1
pi@yahboom:~$
```

Here we find that the system recognizes the voice board as /dev/ttyUSB2, then we enter the following command to view the device path

information,

```
udevadm info --attribute-walk --name=/dev/ttyUSB2 |grep devpath
```

Get the following picture,



```
pi@yahboom:~$ udevadm info --attribute-walk --name=/dev/ttyUSB2 |grep devpath
udevadm info starts with the device specified by the devpath and then
ATTRS{devpath}=="1.4.2"
ATTRS{devpath}=="1.4"
ATTRS{devpath}=="1"
ATTRS{devpath}=="0"
```

- Then, we modify the /etc/udev/rules.d/myspeech.rules file, bind the port number of the voice board, and enter it in the terminal,

```
sudo gedit /etc/udev/rules.d/myspeech.rules
```

Add content as shown in the picture below,

```
KERNEL=="ttyUSB*",ATTRS{devpath}=="1.4.2",ATTRS{idVendor}=="1a86",ATTRS{idProduct}=="7523",MODE:="0777",SYMLINK+=" myspeech"
```

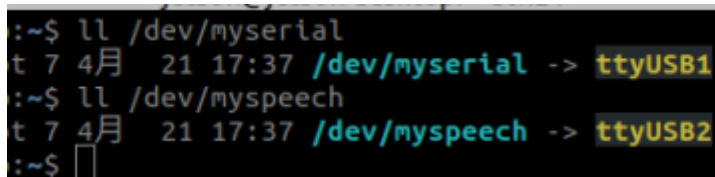
- Exit after saving, enter the following statement in the terminal to reload the system device

```
sudo udevadm trigger
sudo service udev reload
sudo service udev restart
```

1.3.3. Testing

Note where they are wired. This is fixed after binding. Do not change the USB interface position at will later, otherwise the system will not recognize the device, terminal input,

```
ll /dev/myserial
ll /dev/myspeech
```

A terminal window with a black background and green text. It shows the execution of two 'll' (ls -l) commands. The first command is 'll /dev/myserial' and the second is 'll /dev/myspeech'. Both commands show a timestamp 't 7 4月 21 17:37' followed by the device path and an arrow pointing to the device name: '/dev/myserial -> ttyUSB1' and '/dev/myspeech -> ttyUSB2'. The prompt is '~\$' and there is a cursor at the end of the last line.

```
:~$ ll /dev/myserial
t 7 4月 21 17:37 /dev/myserial -> ttyUSB1
:~$ ll /dev/myspeech
t 7 4月 21 17:37 /dev/myspeech -> ttyUSB2
:~$
```

As shown in the figure above, if /dev/myserial and /dev/myspeech can be identified and the port numbers of the following ttyUSB* are inconsistent, it means the binding is successful.

Note: The bound ROS expansion board 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 ports. on the motherboard port.

1.3.4. Precautions

After binding the voice board, when entering the container, you need to mount /dev/myspeech to recognize the voice board in docker. For the mounting method, please refer to [07, Docker-05, Entering the Docker Container of the Robot - Editing Script]. content.