ROS2 serial port example

Source code (downloaded and unzipped in browser): anrot_demo_ros2.zip

This document describes how to read ANROTIMU data in ROS2, and provides the c + language code to execute the ROS2 commands, execute the corresponding nodes, and print the data to the terminal. • Testing environment: Ubuntu 20.04.

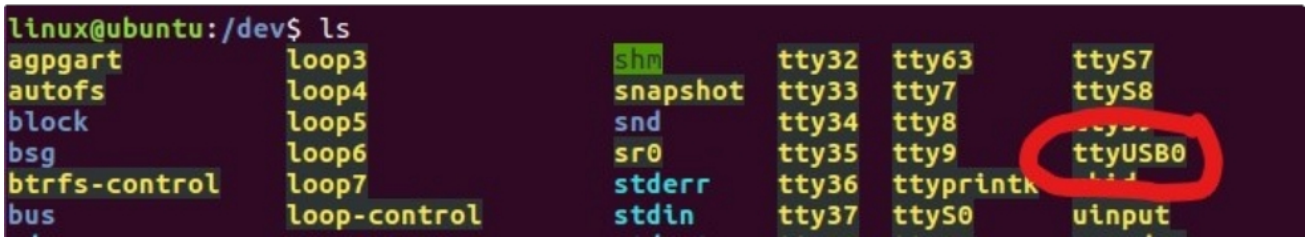
- ROS version: ROS2 Foxy
- Test device: Hi221 Hi226/229 CH100 CH110 CH104 CH108

1. Installation of USB-UART driver

The Ubuntu system comes with the CP210x driver and does not require the serial port driver to be installed by default. When you connect the debugger to your computer, the device will be recognized automatically. After successful identification, a corresponding device will appear under the dev directory: ttyUSBx.

Check if the USB-UART device is recognized by Ubantu:

1. Open the terminal and type ls /dev to view the existing serial port devices.
2. Check if the device file ttyUSBx already exists to confirm the corresponding port number. ttyUSBx indicates the USB device number, which needs to be confirmed because Ubuntu USB device numbers start from zero and accumulate, so the device number is not fixed for multiple devices each time they are booted.
3. Plug in the USB cable, connect the debugging board, and then run ls /dev again. A device has been added to the /dev directory, as shown in the figure:



ttyUSB0 USB Port A device created in ubuntu (the number at the end is variable and may be ttyUSB1 or ttyUSB2).

5. Unlock the executable license of the USB device:

```
$ sudo chmod 777 /dev/ttyUSB0
```

2. Compiling the serial_imu_ws Workspace

1. Open the terminal into the /examples/ROS2/serial_imu_ws directory.
2. Execute the colcon build command, and the following message appears after successful compilation.

```
linux@ubuntu20:~/serial_imu_ws$ colcon build
Starting   € serial_imu
Finished   € serial_imu
           [0.24s]
Summary: 1 package finished
[0.35s]
linux@ubuntu20:~/serial_imu_ws$
```

3. Modify the serial port baud rate and device number.

1. In the Ubuntu environment, the supported baud rates are 115200, 460800, and 921600. The default baud rate used here is 115200, and the default open serial port name is /dev/ttyUSB0.
2. If a higher output frequency is required, modify the #define field in the serial_port.cpp file to another baud rate.

```
#define IMU_SERIAL ("/dev/ttyUSB0")
#define BAUD        (B115200)
```

Note that you need to go back to the `serial_imu_ws` directory and execute the `colcon build` command again.

4. Display data

This example provides a way to view the data:

```
Output the ROS definition of sensor_msgs oImu.
```

4.1: Output ROS Standard Imu.msg

1. Configure the module under Windows to enable quad output.
2. Configure the module using the Window ANROTIMU-UI host computer: Connect the module to the PC host, open the ANROTIMU-UI, connect the corresponding com port, and click **Tools.**
   -> **Configure the module**, in the new window that pops up, click **ATCMD**, then enter the AT command in the input box: AT+SETPTL=0x91, click Send, and **ok will** be displayed at the end of the receive area, indicating that the configuration has been successful, and power off and restart the module. Execute the `ros2 launch serial_imu imu_spec_msg.launch.py` command. After successfully executing the command, you can see the ROS-defined IMU topic message:

```
[listener-2]   ---
[listener-2] header.
[listener-2]    stamp.
[listener-2]      secs:1639099575
[listener-2]      nanosecs:538349240
[listener-2]    frame_id:base_link
[listener-2] orientation.
[listener-2]    x: -0.095125280320644379
[listener-2]    y: -0.483648955821990967
[listener-2]    z: 0.053129896521568298
[listener-2]    w: 0.868453860282897949
[listener-2] orientation_covariance: [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[listener-2] angular_velocity.
[listener-2]    x: -0.000815955184543841
[listener-2]    y: -0.001057390143056437
[listener-2]    z: 0.001062464062371403
[listener-2] angular_velocity_covariance: [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[listener-2] linear_acceleration.
[listener-2]    x: 8.110355603694916482
[listener-2]    y: -2.125157430768013000
[listener-2]    z: 5.013053989410400924
[listener-2] linear_acceleration_covariance: [ 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[listener-2]   ---
```

```
3. Open a separate terminal window and execute `ros2 topic hz /Imu_data` to view the frequency of topic posting.
```

```
linux@ubuntu20:~$ ros2 topic hz /Imu_data
average rate: 100.032
      min: 0.008s max: 0.012s std dev: 0.00058s window: 102
average rate: 100.014
      min: 0.008s max: 0.012s std dev: 0.00054s window: 202
average rate: 100.019
      min: 0.007s max: 0.013s std dev: 0.00064s window: 303
^C
linux@ubuntu20:~$
```

5. FAQ

1. Sometimes the motherboard needs to plug a lot of `usb` devices, in order to facilitate the development, usually a `usb` port constraints file will be written. If the `usb` devices are of different models, they can be distinguished by their `id` numbers. If the devices are of the same model, they all have the same `id` number, so more information is needed to distinguish the different `usb` devices. Here's how to do it

How to distinguish between `usb` devices of the same model.

```
linux@ubuntu:~$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 012: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 002 Device 011: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 002 Device 010: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x UART Bridge / myAVR mySmartUSB light
Bus 002 Device 008: ID 0e0f:0008 VMware, Inc.
Bus 002 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 002 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
linux@ubuntu:~$
```

```
Observe the above screen, and realize that three usb devices have identical id numbers, so you can't differentiate them using simple id numbers, and you need more information about the devices.
```

```
linux@ubuntu:~$ ls /dev
agpgart          loop3           shm       tty32  tty63      ttyS7
autofs           loop4           snapshot  tty33  tty7       ttyS8
block            loop5           snd       tty34  tty8       ttyS9
bsg              loop6           sr0       tty35  tty9       ttyUSB0
btrfs-control    loop7           stderr    tty36  ttyprintk  ttyUSB1
bus              loop-control    stdin     tty37  ttyS0      ttyUSB2
. ..... (not all released)
```

At this point, three `usb` device files have been created in the `dev` file: `ttyUSB0`, `ttyUSB1`, and `ttyUSB2`.Let's look at the details of `ttyUSB0 first`:

```
linux@ubuntu:~$ udevadm info =attribute-walk =name=/dev/ttyUSB0
#This command allows you to view the details of the specified port.
......
    ATTRS{devpath} = "2.2"
    ATTRS{idProduct} = "ea60"
    ATTRS{idVendor} = "10c4"
    ATTRS{ltm_capable} = "no"
    ATTRS{manufacturer} = "Silicon Labs"
    ATTRS{maxchild} = "0"
    ATTRS{product} = "CP2104 USB to UART Bridge Controller"
    ATTRS{quirks} = "0x0"
    ATTRS{removable} = "unknown"
    ATTRS{serial} = "01E34546"
........(There is so much information that I will not put it all out, so you can look at the details for yourself, but only the information that you need to be careful about this time.)
```

Then there is detailed information about `ttyUSB1`:

```
linux@ubuntu:~$ udevadm info =attribute-walk =name=/dev/ttyUSB1
#This command allows you to view the details of the specified port.
......
```

```
    ATTRS{devpath} = "2.3"
    ATTRS{idProduct} = "ea60"
    ATTRS{idVendor} = "10c4"
    ATTRS{ltm_capable} = "no"
    ATTRS{manufacturer} = "Silicon Labs"
    ATTRS{maxchild} = "0"
    ATTRS{product} = "CP2102N USB to UART Bridge Controller"
    ATTRS{quirks} = "0x0"
    ATTRS{removable} = "unknown"
    ATTRS{serial} = "9c1d818b48aeeb119d082897637728c5"
.......(There is so much information that I will not put it all out, so you can look at the details for yourself, but only the information that you need to be careful about this time.)
```

Finally, here are the details of ttyUSB2:

```
linux@ubuntu:~$ udevadm info =attribute-walk =name=/dev/ttyUSB2
#This command allows you to view the details of the specified port.
......
    ATTRS{devnum} = "27"
    ATTRS{devpath} = "2.4"
    ATTRS{idProduct} = "ea60"
    ATTRS{idVendor} = "10c4"
    ATTRS{ltm_capable} = "no"
    ATTRS{manufacturer} = "Silicon Labs"
    ATTRS{maxchild} = "0"
    ATTRS{product} = "CP2104 USB to UART Bridge Controller"
    ATTRS{quirks} = "0x0"
    ATTRS{removable} = "unknown"
    ATTRS{serial} = "02228956"
.......(There is so much information that I will not put it all out, so you can look at the details for yourself, but only the information that you need to be careful about this time.)
```

Looking at the three serial port devices above, I found that ATTRS{serial} = "xxxx" is a particularly arbitrary number. In fact, this is the hardware id number, the only id number for the hardware, and it can be used to give it an alias, so that as long as this hardware id number is recognized, a custom port name device file will appear under the dev, thus permanently linking the port number.

```
linux@ubuntu:~$ cd /etc/udev/rule.d/
linux@ubuntu:/etc/udev/rules.d$ ls
70-snap.core.rules 70-ttyusb.rules 99-vmware-scsi-udev.rules
#This step is to see what constraints are in place to avoid duplicate file names.
linux@ubuntu:~$ sudo vi defined_serial.rules
#This step customizes the name of a port constraint file with the suffix '.rules'.
```

Then type the following in this file:

```
KERNEL=="ttyUSB*",ATTRS{serial}=="9c1d818b48aeeb119d082897637728c5",ATTRS{idVendor}=="10c4",ATTRS{idProduct}=="ea60",MODE:="0777",SYMLINK+="HI226"
KERNEL=="ttyUSB*",ATTRS{serial}=="01E34546",ATTRS{idVendor}=="10c4",ATTRS{idProduct}=="ea60",MODE:="0777",SYMLINK+="BLUETOOCH"
KERNEL=="ttyUSB*",ATTRS{serial}=="02228956",ATTRS{idVendor}=="10c4",ATTRS{idProduct}=="ea60",MODE:="0777",SYMLINK+="CH110"
~
```

The format is as follows:

```
KERNEL = "ttyUSB*", ATTRS{serial} = "xxx", ATTRS{idVendor} = "xxx", ATTRS{idProduct} = "xxx",
MODE:="0777 (port's license)",SYMLINK+="(custom name)"
```

Fill in the corresponding information, save and exit the file, and execute:

```
linux@ubuntu:~$ service udev reload
root privileges required
linux@ubuntu:~$ service udev
restart
linux@ubuntu:~$ ls /dev
agpgart          loop1              sg1       tty32  tty7       ttyS9
autofs           loop2              shm       tty33  tty8       ttyUSB0
block            loop3              snapshot  tty34  tty9       ttyUSB1
BLUETOOCH        loop4              snd       tty35  ttyprintk  ttyUSB2
....
CH110            mcelog             tty0      tty40  ttyS13     vcs1
....
Hi221            rfkill             tty22     tty54  ttyS27     vfio
....
```

As you can see, the customized usb port name is now available, so when you operate it, just operate the corresponding device file, and don't worry about the port number.