4 Robot keyboard control

4 Robot keyboard control
4.1 button description
4.1.1 Direction control
4.1.2 Speed control
4.2 run the program
4.2.1 Keyboard control code yahboom_keyboard.py path:
4.2.2 Operation
4.2.3 Analysis of yahboom_keyboard.py

According to different models, you only need to set the purchased model in [.bashrc], X1(ordinary four-wheel drive) X3(Mike wheel) X3plus(Mike wheel mechanical arm) R2(Ackerman differential) and so on. Section takes X3 as an example

```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to Jetson Orin-Docker/05, Enter the robot's docker container
~/run_docker.sh
#Multiple ros commands require multiple terminals to enter the same docker container for execution, please refer to Jetson Orin-Docker/05, Section 5.8 tutorial
```

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameter and modify the corresponding model

```
export ROBOT_TYPE=X3   # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

4.1 button description

4.1.1 Direction control

[i] or [I]	[linear,0]	[u] or [U]	[linear,angular]
[,]	[-linear,0]	[o] or [O]	[linear,- angular]
[j]or[J]	[0, angular]	[m]or[M]	[- linear,- angular]
[l]or[L]	[0,- angular]	[.]	[- linear,angular]

4.1.2 Speed control

button	speed change	button	speed change
[q]	Linear and angular velocity are both increased by 10%	[with]	Linear and angular velocities are both reduced by 10%

button	speed change	button	speed change
[in]	10% increase in line speed only	[x]	10% reduction in line speed only
[and]	10% increase in angular velocity only	[c]	Only the angular velocity is reduced by 10%
[t]	Line speed X-axis/Y-axis direction switching	[s]	stop keyboard control

4.2 run the program

4.2.1 Keyboard control code yahboom_keyboard.py path:

```
~/yahboomcar_ws/src/yahboom_ctrl/scripts
```

4.2.2 Operation

```
roslaunch yahboomcar_bringup bringup.launch        # chassis start
roslaunch yahboomcar_ctrl yahboom_keyboard.launch  # keyboard control node
```

4.2.3 Analysis of yahboom_keyboard.py

1. the topic released: cmd_vel

```
pub = rospy.Publisher('cmd_vel', Twist, queue_size=1)
```

So, just package the speed and publish it through pub.publish(twist) the speed subscriber of the chassis can receive the speed data, and then drive the car.

2. the main modules used

- The select module is mainly used for socket communication, seeing changes in file descriptions, and completing non-blocking work
- The termios module provides an interface for IO-controlled POSIX calls to ttys
- The tty module is mainly used to change the mode of the file descriptor fd

3. mobile dictionary and speed dictionary

- The mobile dictionary mainly stores the relevant characters of the direction control

```
moveBindings = {
    'i': (1, 0),
    'o': (1, -1),
    'j': (0, 1),
    'l': (0, -1),
    'u': (1, 1),
    ',': (-1, 0),
    '.': (-1, 1),
    'm': (-1, -1),
    'I': (1, 0),
    'O': (1, -1),
    'J': (0, 1),
```

```
    'L': (0, -1),
    'U': (1, 1),
    'M': (-1, -1),
}
```

- The speed dictionary mainly stores the relevant characters of speed control

```
speedBindings = {
    'Q': (1.1, 1.1),
    'Z': (.9, .9),
    'W': (1.1, 1),
    'X': (.9, 1),
    'E': (1, 1.1),
    'C': (1, .9),
    'q': (1.1, 1.1),
    'z': (.9, .9),
    'w': (1.1, 1),
    'x': (.9, 1),
    'e': (1, 1.1),
    'c': (1, .9),
}
```

4. get the current key information

```
 def getKey():
  # tty.setraw(): change the file descriptor fd mode to raw; fileno(): return
an integer file descriptor(fd)
  tty.setraw(sys.stdin.fileno())
   # select(): Directly call the IO interface of the operating system;
monitor all file handles with the fileno() method
    rlist, _, _  =  select.select([ sys.stdin ], [], [], 0.1)
    # Read a byte from the input stream
    if rlist: key = sys.stdin.read(1)
    else: key = ''
    # tcsetattr sets the tty attribute of the file descriptor fd from the
attribute
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return  key
```

5. determine whether t/T or s/S is pressed

```
 if key=="t" or key == "T": xspeed_switch = not xspeed_switch
            elif key == "s" or key == "S":
                print ("stop keyboard control: {}".format(not stop))
                stop = not stop
```

6. determine whether the string is in the dictionary

```
#Whether the key string is in the mobile dictionary
if key in moveBindings.keys():
                x = moveBindings[key][0]
                th = moveBindings[key][1]
                count = 0
#Whether the key character is in the speed dictionary
 elif key in speedBindings.keys():
                speed = speed * speedBindings[key][0]
                turn = turn * speedBindings[key][1]
                count = 0
```

6. speed limit

Both the angular velocity and the linear velocity have a limit value, which cannot be increased all the time. When starting, the program will first obtain the speed limit value, and when increasing the speed, it will judge and process the increased value.

```
linear_limit  =  rospy.get_param('~linear_speed_limit', 1.0)
angular_limit  =  rospy.get_param('~angular_speed_limit', 5.0)
```

7. program flow chart