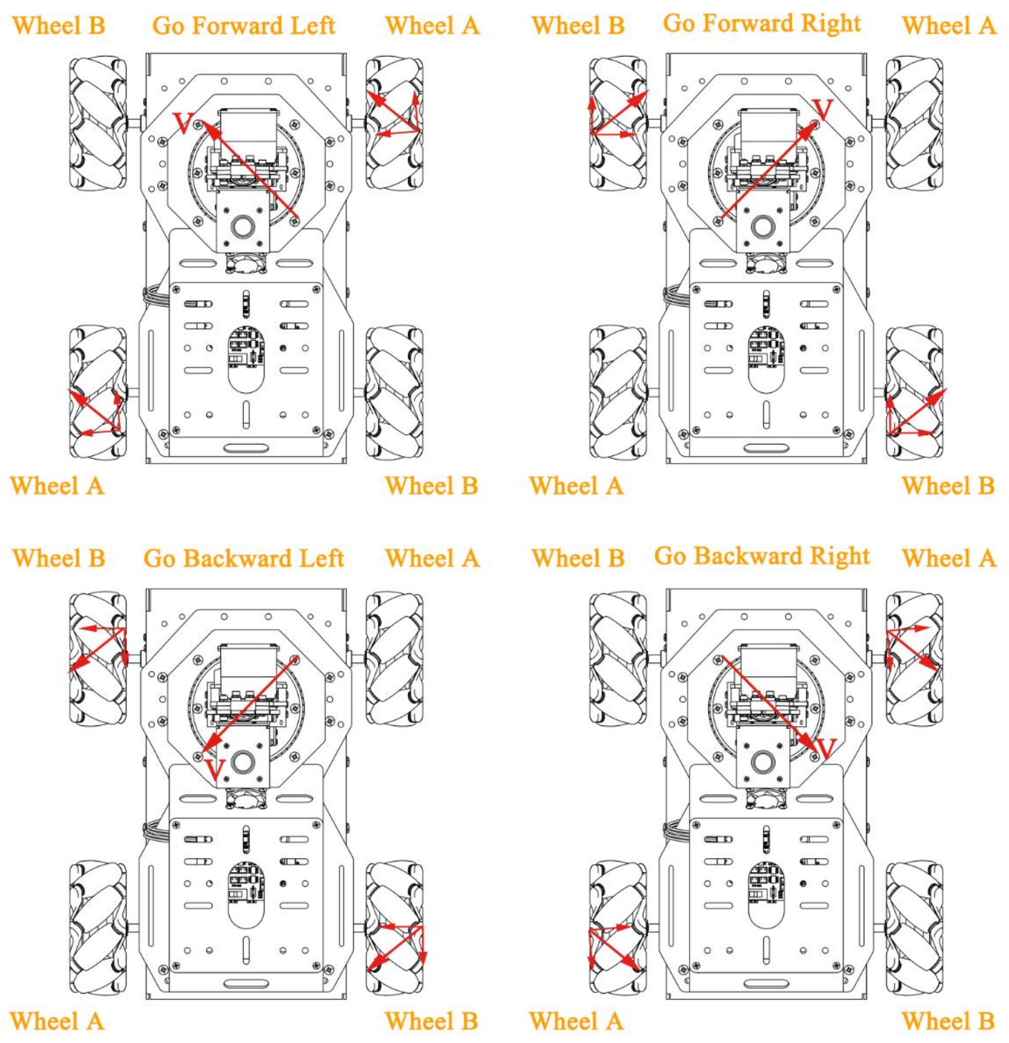
Lesson 6 Oblique Movement

1. Working Principle

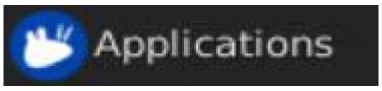
According to the characteristics of mecanum wheel , when wheel A does not move and wheel B rotates clockwise, the car moves to the left front. When wheel B rotates counterclockwise, the car moves to the right rear. When the wheel B does not move and wheel A rotates clockwise, the car will move to the front right. When the wheel A rotates counterclockwise, the car move to the left rear. The force analysis for oblique movement:

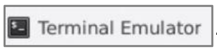


2. Operation Steps

It should be case sensitive when entering command, and the "Tab" key can be used to complete the keywords.

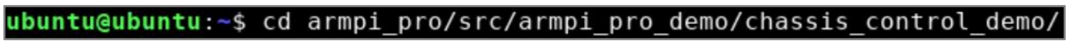
1) Please refer to the content in "7. ArmPi Pro Basic Lesson/1.Mecanum Wheel Chassis Lesson/ Lesson 2 Set Environment Development" to connect system desktop via No Machine.

2) Click 

in the lower left corner and select 

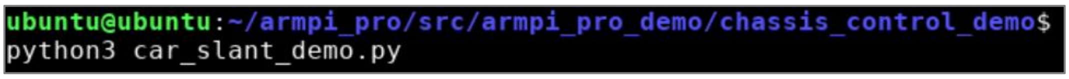
to enter the terminal.

3) Enter command "**cd armpi_pro/src/armpi_pro_demo/chassis_control_demo/**" and press "Enter" to enter the directory of game programmings.



```
ubuntu@ubuntu:~$ cd armpi_pro/src/armpi_pro_demo/chassis_control_demo/
```

4) Enter command "**python3 car_slant_demo.py**" and press "Enter" to start game.



```
ubuntu@ubuntu:~/armpi_pro/src/armpi_pro_demo/chassis_control_demo$
python3 car_slant_demo.py
```

5) If want to exit the game, press "Ctrl+C" in terminal. If fail to exit, please keep trying until the program is closed.

3. Project Outcome

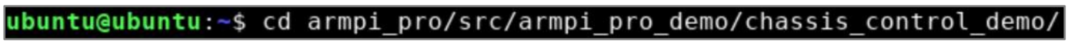
After starting the game, ArmPi Pro will move to the right front, to the right rear, to the left rear, and to the front left in sequence.

4. Function Extension

The default angle of oblique movement is 45 degrees and the value can be modified to adjust the angle. This section will modify the value to 60 and the specific operation steps are as follow:

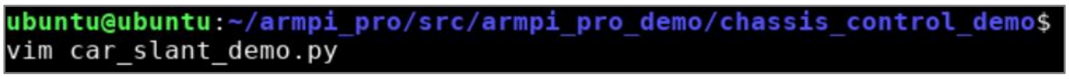
1) Click and select to enter the terminal.

2) Enter command "**cd armpi_pro/src/armpi_pro_demo/chassis_control_demo/**" and press "Enter" to come to the directory of game programmings.



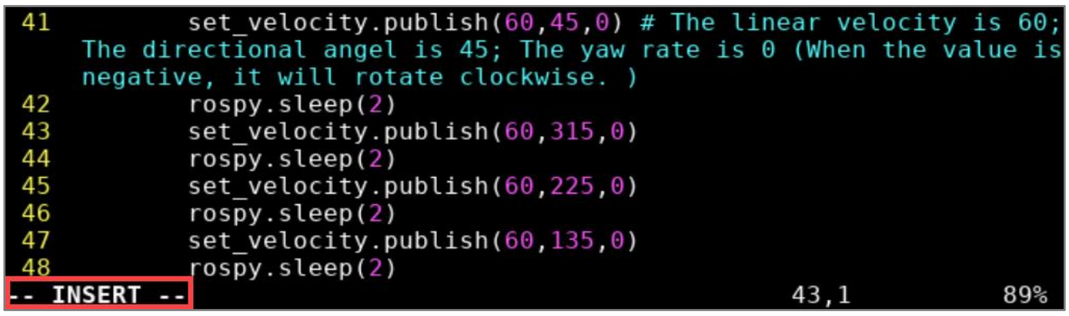
```
ubuntu@ubuntu:~$ cd armpi_pro/src/armpi_pro_demo/chassis_control_demo/
```

3) Enter command "**vim car_slant_demo.py**" and press "Enter" to open the program file.



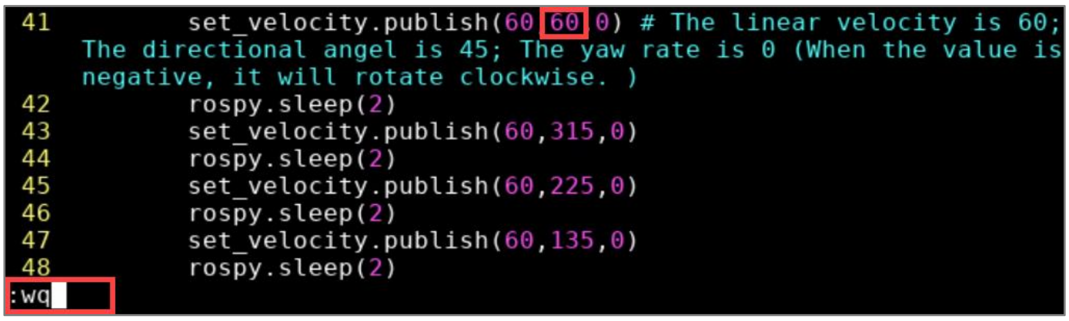
```
ubuntu@ubuntu:~/armpi_pro/src/armpi_pro_demo/chassis_control_demo$
vim car_slant_demo.py
```

4) Find the code to be modified and press "i". When the prompt "INSERT" appears in the lower left corner, it means the terminal has been switched to the editing mode.



```
        set_velocity.publish(60,45,0) # The linear velocity is 60;
 The directional angel is 45; The yaw rate is 0 (When the value is
 negative, it will rotate clockwise. )
        rospy.sleep(2)
        set_velocity.publish(60,315,0)
        rospy.sleep(2)
        set_velocity.publish(60,225,0)
        rospy.sleep(2)
        set_velocity.publish(60,135,0)
        rospy.sleep(2)
-- INSERT --                                     43,1          89%
```

5) In "**set_velocity.publish()**" function, the second parameter represents the directional angle of moving forwards and we change it to 60. After modifying, press "Esc" and enter ":wq", and then press "Enter" to save and exit.

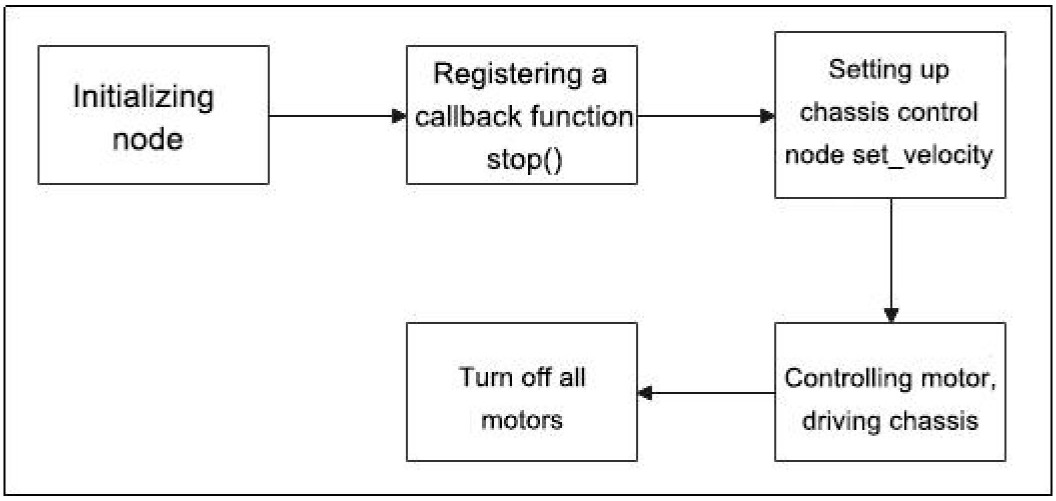


```
        set_velocity.publish(60,60,0) # The linear velocity is 60; The directional angel is 45; The yaw rate is 0 (When the value is negative, it will rotate clockwise. )
        rospy.sleep(2)
        set_velocity.publish(60,315,0)
        rospy.sleep(2)
        set_velocity.publish(60,225,0)
        rospy.sleep(2)
        set_velocity.publish(60,135,0)
        rospy.sleep(2)
:wq
```

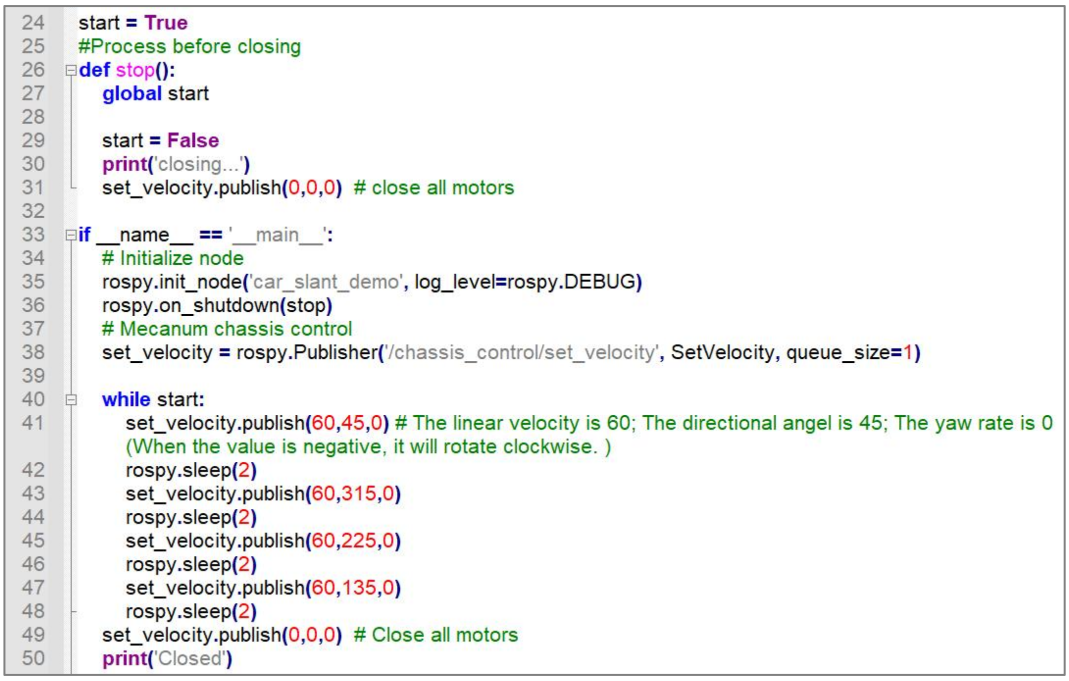
Note: The adjustable range of the directional angle is from 0 to 360. 0 refers to move to the right; 90 refers to move forwards; 180 refers to move to the left; 270 refers to move backwards.

6) After modifying, you can refer to the content of “2.Operation Steps” to check the effect.

5. Program Analysis


Initializing node
Registering a callback function stop()
Setting up chassis control node set_velocity
Controlling motor, driving chassis
Turn off all motors

The source code of program is located in: **home/ubuntu/armpi_pro/src/armpi_pro_demo/chassis_control_demo/car_slant_demo.py**



```
start = True
#Process before closing
def stop():
    global start

    start = False
    print('closing...')
    set_velocity.publish(0,0,0)  # close all motors

if __name__ == '__main__':
    # Initialize node
    rospy.init_node('car_slant_demo', log_level=rospy.DEBUG)
    rospy.on_shutdown(stop)
    # Mecanum chassis control
    set_velocity = rospy.Publisher('/chassis_control/set_velocity', SetVelocity, queue_size=1)

    while start:
        set_velocity.publish(60,45,0) # The linear velocity is 60; The directional angel is 45; The yaw rate is 0 (When the value is negative, it will rotate clockwise. )
        rospy.sleep(2)
        set_velocity.publish(60,315,0)
        rospy.sleep(2)
        set_velocity.publish(60,225,0)
        rospy.sleep(2)
        set_velocity.publish(60,135,0)
        rospy.sleep(2)
    set_velocity.publish(0,0,0)  # Close all motors
    print('Closed')
```

Control motor through **set_velocity.publish()**. There are three parameters in function. Take the code "**chassis.set_velocity(60,45,0)**" as an example:

1) The first parameter "60" represents the motor speed, its unit is mm/s and it ranges from -100 to 100. When the value is negative, the motor rotates counterclockwise.

2) The second parameter "45" represents the movement direction of car, its unit is degree and it ranges from 0 to 360. The value of 90° refer to move forward. 270° refers to move backward. 0° refers to move to the right. 180° refers to move the left. Other movement directions are obtained according to the same reference method.

3) The third parameter "0" represents the rotation speed of the car, its unit is 5°/s and it ranges from -2 to 2. When the parameter value is positive, the car will rotate clockwise. When the parameter value is negative, the car will rotate counterclockwise.