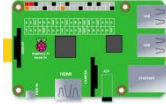
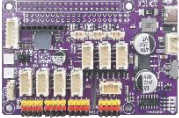
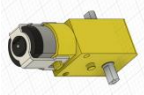


# Lesson 10 Control the DC Motor to Work for Mecanum Wheels

## 10.1 Overview

In this lesson, we'll learn to control a DC motor with a Raspberry Pi and Adept Robot HAT V3.2. We'll cover components, principles, wiring, run a Python program for forward - backward rotation, and analyze the code. By the end, you can control the motor's direction and speed.

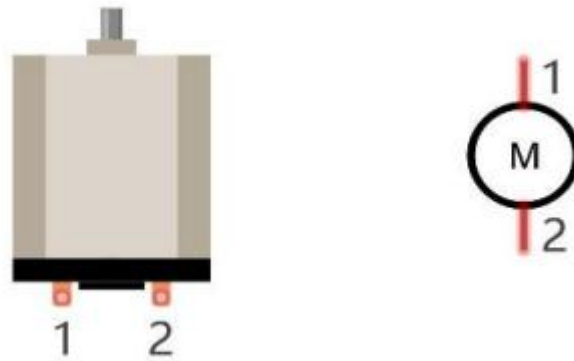
## 10.2 Required Components

Components	Quantity	Picture
Raspberry Pi	1	
Adept Robot HAT V3.2	1	
DC Motor	4	

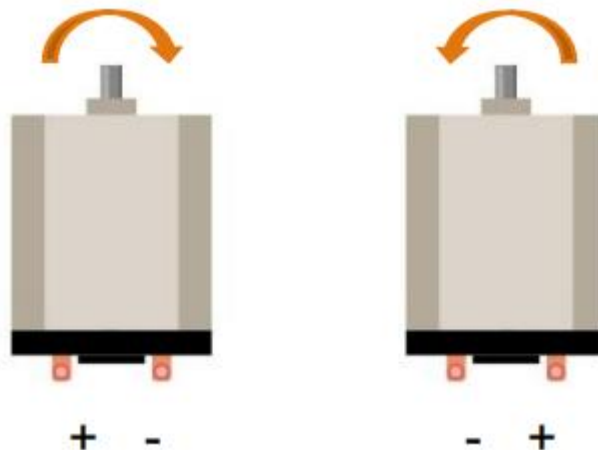
## 10.3 Principle Introduction

Our products use DC motor as a power device. A motor is a device that converts electrical energy into mechanical energy. Motor consists of two parts: stator and rotor. When motor works, the stationary part is stator, and the rotating part is rotor. Stator is usually the outer case of motor,

and it has terminals to connect to the power. Rotor is usually the shaft of motor, and can drive other mechanical devices to run. The schematic below is a small DC motor with two pins.



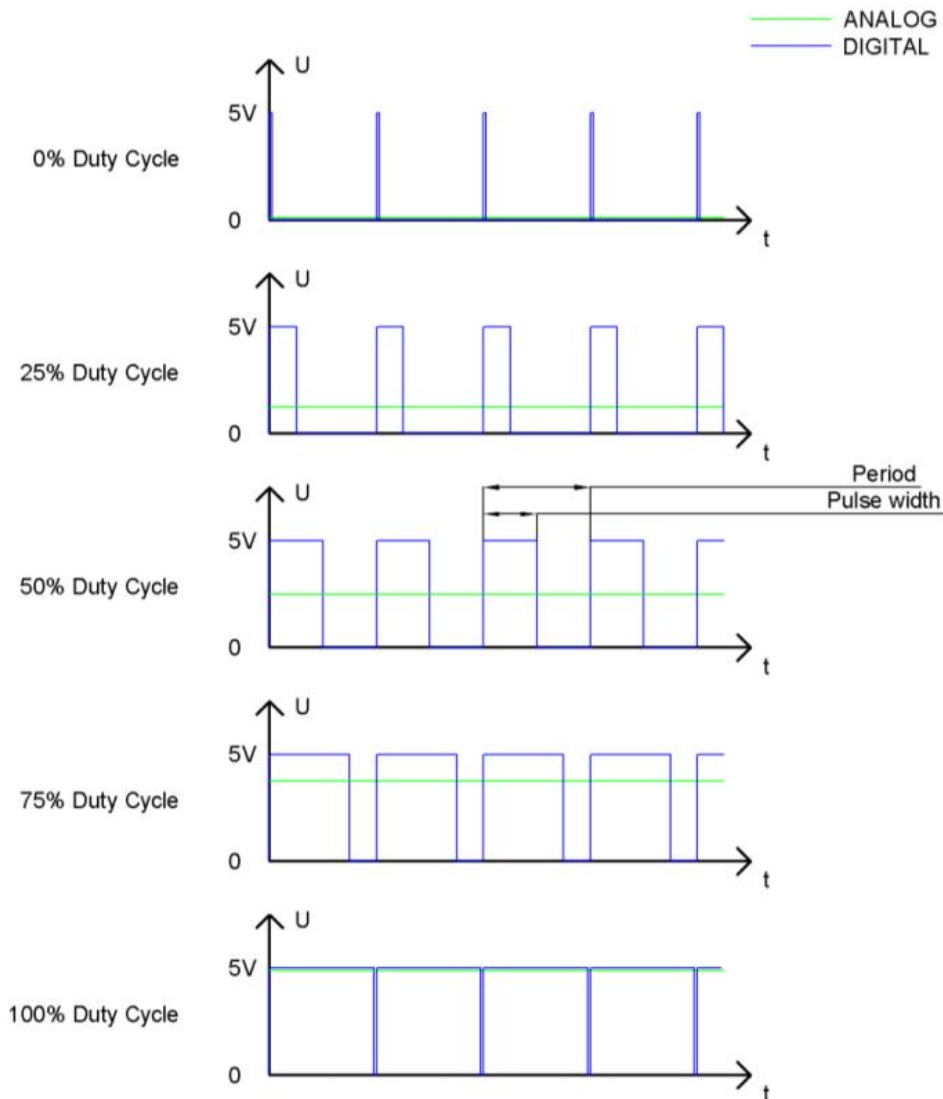
When a motor gets connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, then the motor rotates in opposite direction.



## PWM

PWM, Pulse Width Modulation, uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

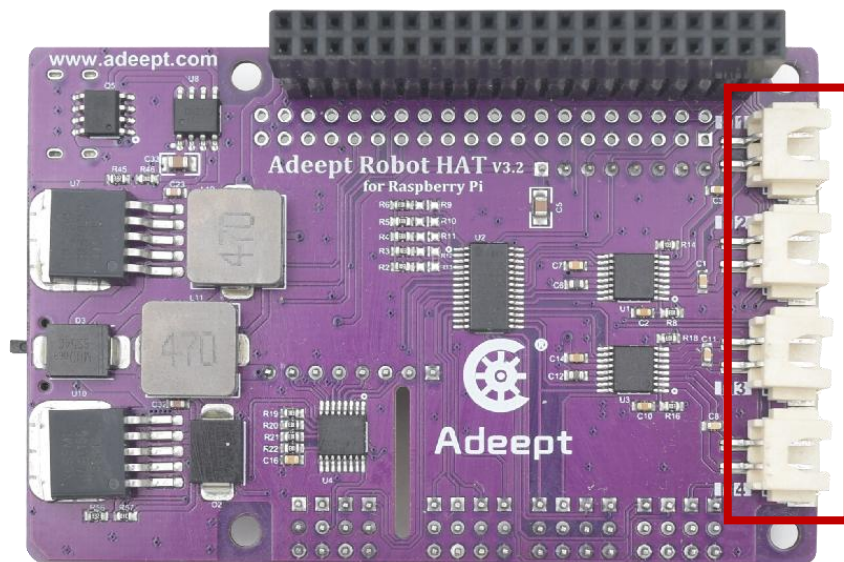
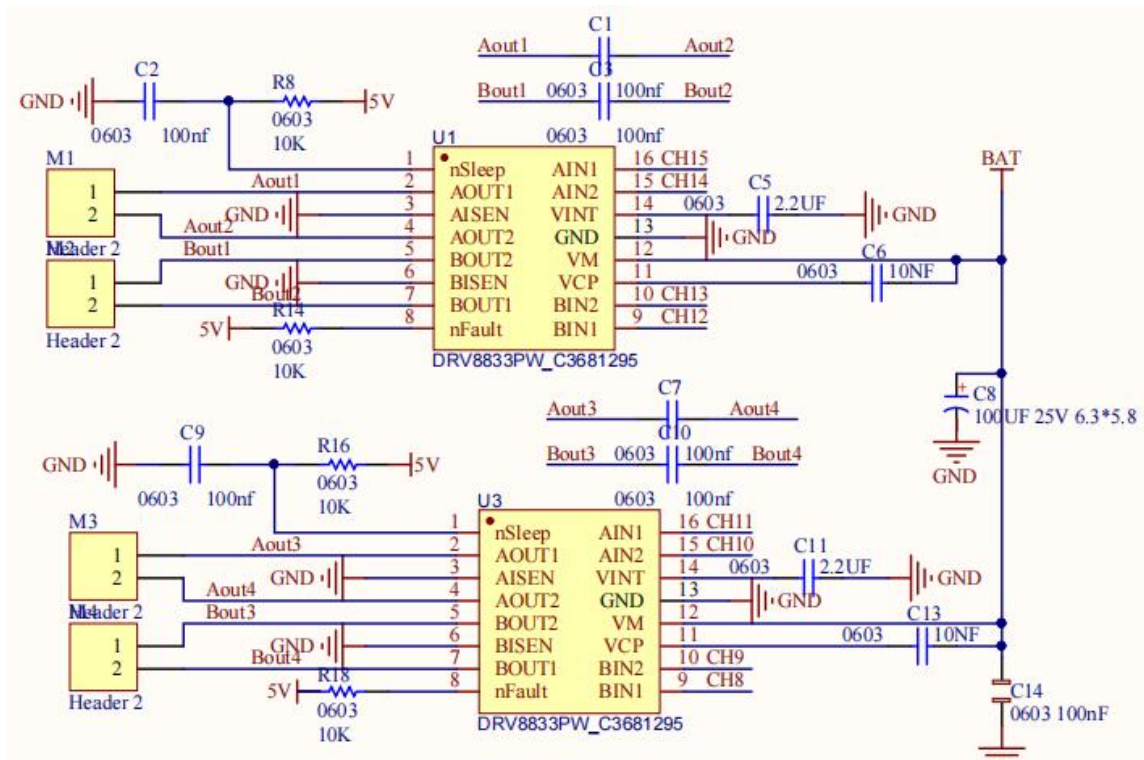
The longer the output of high levels last, the larger the duty cycle and the higher the corresponding voltage in analog signal will be. The following figures show how the analog signal voltage vary between 0V-5V(high level is 5V) corresponding to the pulse width 0%-100%:



The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on.

## 10.4 Wiring Diagram

When the DC Motor module is in use, it needs to be connected to the M1 interface on the Adept Robot HAT V3.2 expansion board.



## 10.5 About Mecanum Wheels

Mecanum Wheel is a patent of the Swedish Mecanum Company, which is often used in the field of robotics to achieve omnidirectional movement. The Mecanum wheel consists of two parts: the hub and the roller. The hub is the main support for the entire wheel, and the rollers are the drums mounted on the hub. The hub axle is at a  $45^\circ$  angle to the roller axis. (The angle between the hub axle of the omnidirectional wheel and the roller is 90 degrees)

Mecanum wheels, like conventional wheels, can be mounted on axes parallel to each other.

Wheat wheels are generally used in groups of four, two left-handed wheels and two right-handed wheels. The difference between left-handed and right-handed wheels is shown in the figure below



The Mecanum wheel car uses Mecanum wheels on the basis of ordinary cars, and each wheel can be controlled independently.

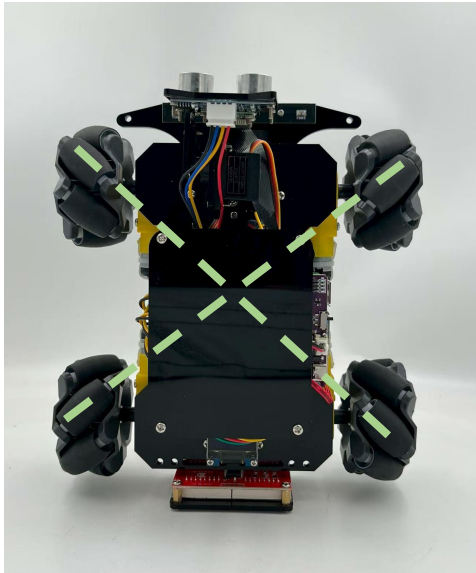
There are also many ways to install Mecanum wheels on cars. Mainly divided into:

X-square (X-square), X-rectangle (X-rectangle), O-square (O-square), O-rectangle (O-rectangle).

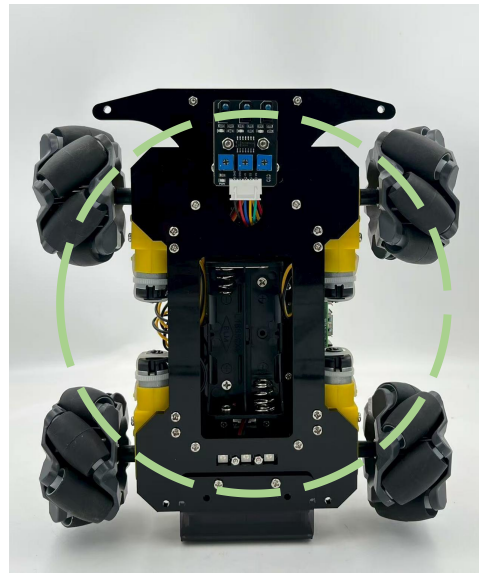
Where X and O represent the figure formed by the rollers in contact with the ground of the four wheels; square and rectangle refer to the shape enclosed by the contact points of the four wheels with the ground.

The installation method of our Mecanum wheel car products is the common O-rectangular

installation method.



X



O

**Note:** The Mecanum wheel car adopts the O-rectangle assembly method. The actual wheel is "X" when viewed from above, and is "O" when it is actually in contact with the ground. (looking up from below the Mecanum wheel car)

## 10.6 Demonstration

1. **Remotely log:** Remotely log in to the Raspberry Pi terminal.
2. **Navigate to the Program Folder:** Enter the following command in the terminal and press Enter to access the folder where the program is located:

```
cd Adept_4WD_Smart_Car_for_RPi/Examples/04_Motor/
```

```
pi@raspberrypi:~ $ cd Adept_4WD_Smart_Car_for_RPi/Examples/04_Motor/  
pi@raspberrypi:~/Adept_4WD_Smart_Car_for_RPi/Examples/04_Motor $
```

3. **View Directory Contents:** Type "ls" in the terminal and press Enter. This will display all the files in the current directory, ensuring that the "**MotorCtrl.py**" file is present:



```
ls
```

```
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/04_Motor $ ls
Motor_Mecanum.py  Motor_Ordinary.py
```

4. **Run the Program:** Enter the command below and press Enter to start the Motor control program:

```
sudo python3 Motor_Mecanum.py
```

```
pi@raspberrypi:~/Adeept_4WD_Smart_Car_for_RPi/Examples/04_Motor $ sudo python3 Motor_Mecanum.py
Forward
Backward
Turn left
Turn right
Drift left
Drift right
```

5. **Observation and Termination:** Once the program runs successfully, you'll observe the DC motor rotating forward and backward. The program first sets the motor to rotate forward at a speed of 50 (scaled to the appropriate duty - cycle value) for 2 seconds, then rotates it backward at the same speed for 2 seconds. This process repeats 10 times. To stop the running program, simply press the "**Ctrl + C**" shortcut on the keyboard. This action will stop the motor and release the PCA9685 resources.

## 10.7 Code

Complete code refer to [Motor\\_Mecanum.py](#)

```
001  #!/usr/bin/env/python3
002  # File name   : Motor_Mecanum.py
003  # Website    : www.Adeept.com
004  # Author     : Adeept
005  # Date      : 2025/03/6
006  import time
007  from board import SCL, SDA
008  import busio
009  from adafruit_pca9685 import PCA9685
010  from adafruit_motor import motor
011
012  MOTOR_M1_IN1 = 15      #Define the positive pole of M1
013  MOTOR_M1_IN2 = 14      #Define the negative pole of M1
014  MOTOR_M2_IN1 = 12      #Define the positive pole of M2
015  MOTOR_M2_IN2 = 13      #Define the negative pole of M2
016  MOTOR_M3_IN1 = 11      #Define the positive pole of M3
017  MOTOR_M3_IN2 = 10      #Define the negative pole of M3
018  MOTOR_M4_IN1 = 8        #Define the positive pole of M4
019  MOTOR_M4_IN2 = 9        #Define the negative pole of M4
020
```

```
021 def map(x,in_min,in_max,out_min,out_max):
022     return (x - in_min)/(in_max - in_min) *(out_max - out_min) +out_min
023
024 i2c = busio.I2C(SCL, SDA)
025 pwm_motor = PCA9685(i2c, address=0x5f)
026 pwm_motor.frequency = 50
027
028 motor1 = motor.DCMotor(pwm_motor.channels[MOTOR_M1_IN1],pwm_motor.channels[MOTOR_M1_IN2] )
029 motor1.decay_mode = (motor.SLOW_DECAY)
030 motor2 = motor.DCMotor(pwm_motor.channels[MOTOR_M2_IN1],pwm_motor.channels[MOTOR_M2_IN2] )
031 motor2.decay_mode = (motor.SLOW_DECAY)
032 motor3 = motor.DCMotor(pwm_motor.channels[MOTOR_M3_IN1],pwm_motor.channels[MOTOR_M3_IN2] )
033 motor3.decay_mode = (motor.SLOW_DECAY)
034 motor4 = motor.DCMotor(pwm_motor.channels[MOTOR_M4_IN1],pwm_motor.channels[MOTOR_M4_IN2] )
035 motor4.decay_mode = (motor.SLOW_DECAY)
036
037 def Motor(channel,direction,motor_speed):
038     if motor_speed > 100:
039         motor_speed = 100
040     elif motor_speed < 0:
041         motor_speed = 0
042     speed = map(motor_speed, 0, 100, 0, 1.0)
043     if direction == 1:
044         speed = -speed
045
046     if channel == 1:
047         motor1.throttle = speed
048     elif channel == 2:
049         motor2.throttle = speed
050     elif channel == 3:
051         motor3.throttle = speed
052     elif channel == 4:
053         motor4.throttle = speed
054
055 def motorStop():#Motor stops
056     motor1.throttle = 0
057     motor2.throttle = 0
058     motor3.throttle = 0
059     motor4.throttle = 0
060
061 def destroy():
062     motorStop()
063     pwm_motor.deinit()
064
065
066 if __name__ == '__main__':
067     try:
068         for i in range(10):
069             speed = 80
070
071             Motor(1, 1, speed) #forward
072             Motor(2, 1, speed)
073             Motor(3, 1, speed)
074             Motor(4, 1, speed)
075             print("Forward")
076             time.sleep(2)
```



```
077
078     Motor(1,-1, speed) #backward
079     Motor(2,-1, speed)
080     Motor(3,-1, speed)
081     Motor(4,-1, speed)
082     print("Backward")
083     time.sleep(2)
084
085     Motor(1,-1, speed) #turn left
086     Motor(2,-1, speed)
087     Motor(3,1, speed)
088     Motor(4,1, speed)
089     print("Turn left")
090     time.sleep(2)
091
092     Motor(1,1, speed) #turn right
093     Motor(2,1, speed)
094     Motor(3,-1, speed)
095     Motor(4,-1, speed)
096     print("Turn right")
097     time.sleep(2)
098
099     Motor(1, -1, speed) #drift left
100     Motor(2, 1, speed)
101     Motor(3, -1, speed)
102     Motor(4, 1, speed)
103     print("Drift left")
104     time.sleep(2)
105
106     Motor(1, 1, speed) #drift right
107     Motor(2, -1, speed)
108     Motor(3, 1, speed)
109     Motor(4, -1, speed)
110     print("Drift right")
111     time.sleep(2)
112
113     Motor(1,1, 0) #drift front-left
114     Motor(2,1, speed)
115     Motor(3,1, 0)
116     Motor(4,1, speed)
117     print("Drift front-left")
118     time.sleep(2)
119
120     Motor(1,-1, 0) #drift rear-right
121     Motor(2,-1, speed)
122     Motor(3,-1, 0)
123     Motor(4,-1, speed)
124     print("Drift rear-right")
125     time.sleep(2)
126
127     Motor(1,1, speed) #drift front-right
128     Motor(2,1, 0)
129     Motor(3,1, speed)
130     Motor(4,1, 0)
131     print("Drift front-right")
132     time.sleep(2)
```

```
133
134     Motor(1,-1, speed) #drift rear-left
135     Motor(2,-1, 0)
136     Motor(3,-1, speed)
137     Motor(4,-1, 0)
138     print("Drift rear-left")
139     time.sleep(2)
140     destroy()
141 except KeyboardInterrupt:
142     destroy()
143
144
145
146
147
148
149
150
```

## Code explanation

### Initialization Stage:

Connect the PCA9685 module (address 0x5F) via the I2C protocol. Set the PWM frequency to 50Hz and initialize 4 DC motors: M1 - M4 (bind them to GPIO pins respectively and configure them in slow decay mode).

### Loop Control Process:

The main program loops 10 times, sequentially executes 10 movement modes, each lasting 2 seconds. The process is as follows:

- Forward movement: All motors rotate forward
- Backward movement: All motors rotate backward
- Left turn: Left motors (M1, M2) rotate backward, right motors (M3, M4) rotate forward
- Right turn: Left motors rotate forward, right motors rotate backward
- Left lateral movement: M1/M3 rotate backward, M2/M4 rotate forward
- Right lateral movement: M1/M3 rotate forward, M2/M4 rotate backward
- Front-left drift: M2/M4 rotate forward, M1/M3 stop
- Rear-right drift: M2/M4 rotate backward, M1/M3 stop
- Front-right drift: M1/M3 rotate forward, M2/M4 stop

- Rear-left drift: M1/M3 rotate backward, M2/M4 stop

Safety Mechanism:

motorStop(): Immediately stop all motors.

destroy(): Release the PCA9685 resources.

Press **Ctrl+C** to trigger a safe exit.