

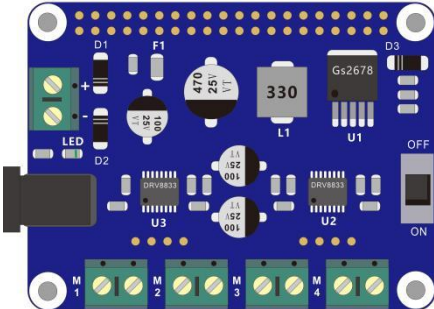
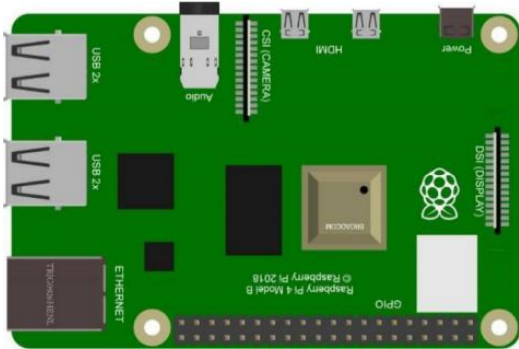
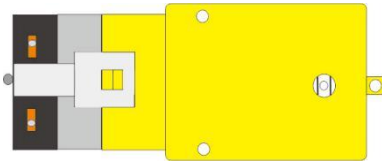

Pi Power & 4WD Hat drive the DC motors to run

Table

1. Component List.....	2
2. Component related knowledge.....	2
2.1 Knowledge of the TT Motor.....	2
2.2 Knowledge of the Raspberry Pi.....	4
3. Circuit Connection.....	7
3.1 Circuit connection diagram:.....	8
4. Download Code and Run.....	8
5. Trouble shooting.....	15
6. Any questions and suggestions are welcome.....	16

1. Component List

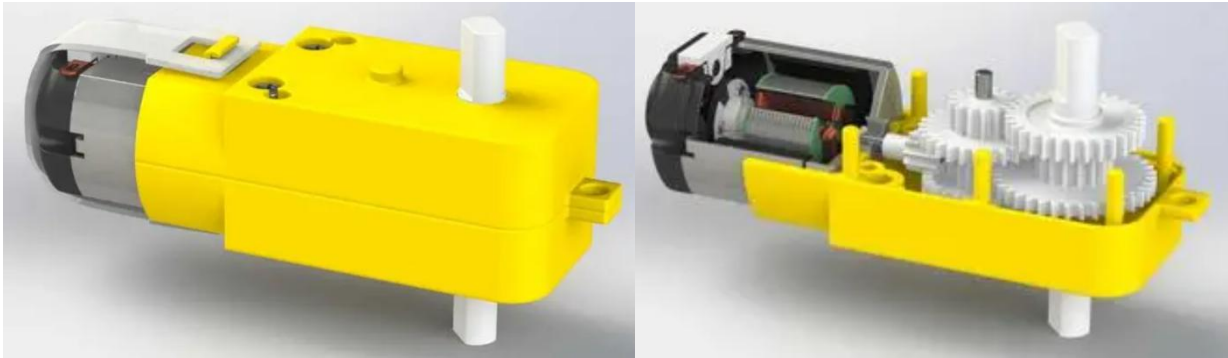
For this test experiment, what do you need to prepare like below list

Component/Module	QTY	Picture	Remark
Cokoino Pi Power & 4WD HAT	1		
Raspberry Pi 4B	1		You need to prepare these by yourself.
TT motor	4		
Battery box with 2pcs 18650 batteries	1		

2. Component related knowledge

2.1 Knowledge of the TT Motor

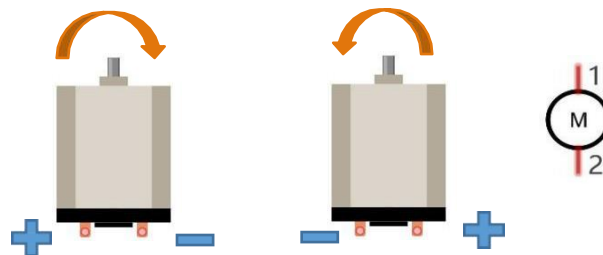
As shown in the figure below, the TT motor consists of a DC motor and related gears, with a yellow outer shell fastened.



DC Motor

When motor is connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, the motor will rotate in the opposite direction.

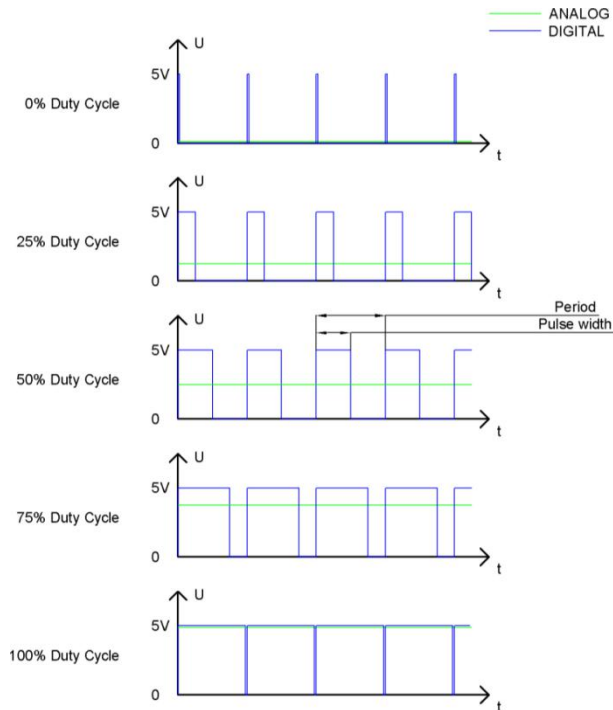
And the speed of motor depends on the voltage between two ends. The larger the voltage, the larger the speed.



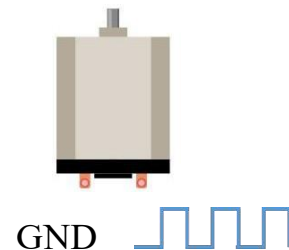
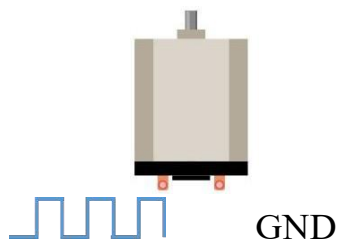
PWM

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 varies 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.



2.2 Knowledge of the Raspberry Pi

GPIO

GPIO: General Purpose Input/Output. Here we will introduce the specific function of the pins on the Raspberry Pi and how you can utilize them in all sorts of ways in your projects. Most RPi Module pins can be used as either an input or output, depending on your program and its functions.

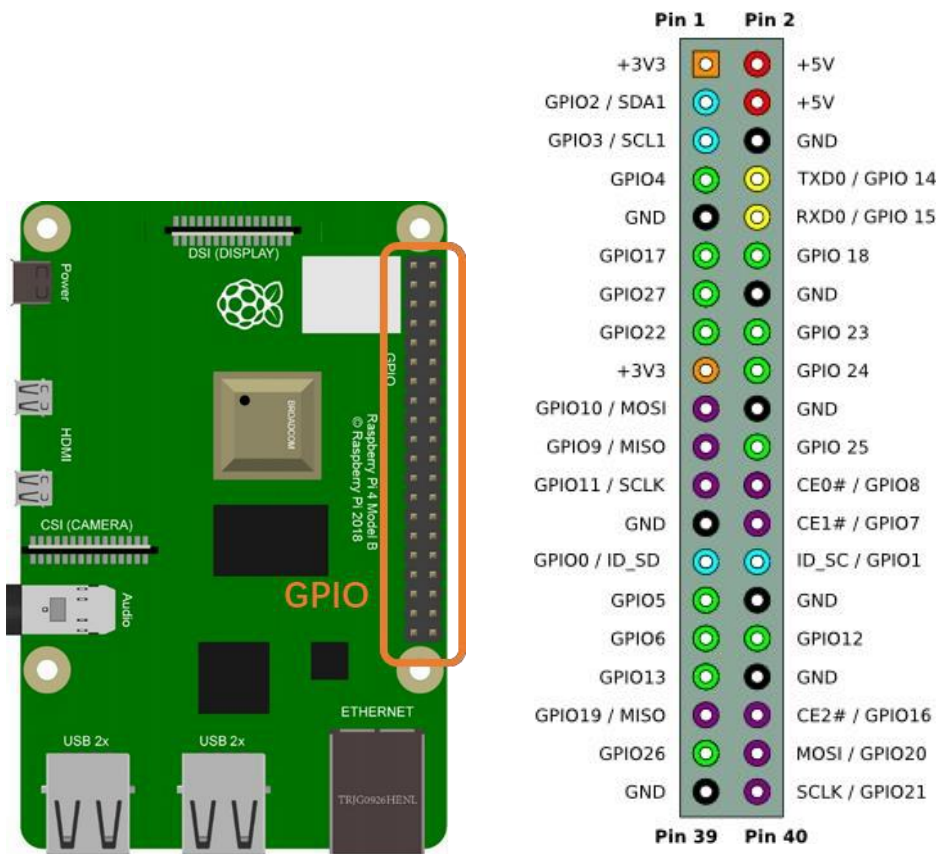
When programming GPIO pins there are 3 different ways to reference them: GPIO Numbering, Physical Numbering and WiringPi GPIO Numbering.

BCM GPIO Numbering

The Raspberry Pi CPU uses Broadcom (BCM) processing chips BCM2835, BCM2836 or BCM2837. GPIO pin numbers are assigned by the processing chip manufacturer and are how the computer recognizes each pin. The pin numbers themselves do not make

sense or have meaning as they are only a form of identification. Since their numeric values and physical locations have no specific order, there is no way to remember them so you will need to have a printed reference or a reference board that fits over the pins.

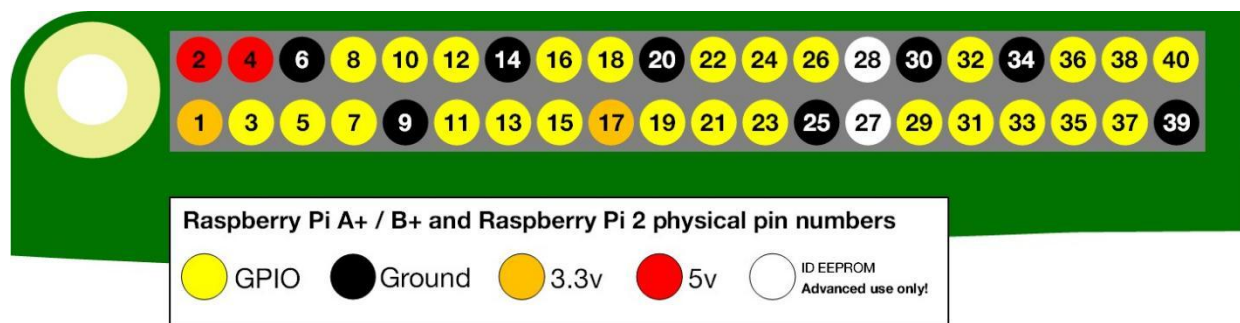
Each pin's functional assignment is defined in the image below:



For more details about pin definition of GPIO, please refer to <http://pinout.xyz/>

PHYSICAL Numbering

Another way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'Physical Numbering', as shown below:



WiringPi GPIO Numbering

Different from the previous two types of GPIO serial numbers, RPi GPIO serial number of the WiringPi are numbered according to the BCM chip use in RPi.

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	
—	—	3.3v	1 2	5v	—	—	For Pi B
8	R1:0/R2:2	SDA	3 4	5v	—	—	
9	R1:1/R2:3	SCL	5 6	0v	—	—	
7	4	GPIO7	7 8	TxD	14	15	
—	—	0v	9 10	RxD	15	16	
0	17	GPIO0	11 12	GPIO1	18	1	
2	R1:21/R2:27	GPIO2	13 14	0v	—	—	
3	22	GPIO3	15 16	GPIO4	23	4	
—	—	3.3v	17 18	GPIO5	24	5	
12	10	MOSI	19 20	0v	—	—	
13	9	MISO	21 22	GPIO6	25	6	
14	11	SCLK	23 24	CE0	8	10	
—	—	0v	25 26	CE1	7	11	
30	0	SDA.0	27 28	SCL.0	1	31	
21	5	GPIO.21	29 30	0V	—	—	
22	6	GPIO.22	31 32	GPIO.26	12	26	
23	13	GPIO.23	33 34	0V	—	—	
24	19	GPIO.24	35 36	GPIO.27	16	27	
25	26	GPIO.25	37 38	GPIO.28	20	28	
—	—	0V	39 40	GPIO.29	21	29	
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	

(For more details, please refer to <https://projects.drogon.net/raspberry-pi/wiringpi/pins/>.)

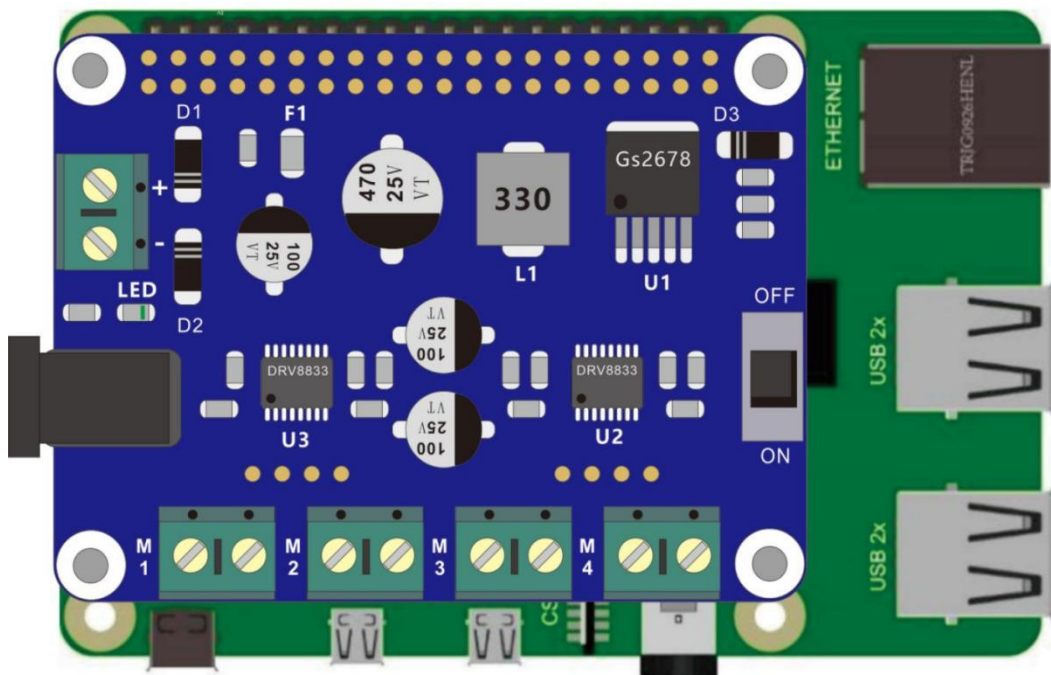
You can also use the following command to view their correlation.

```
gpio readall
```

+-----Pi 4B-----+											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALT0	1	3	4		5v			
3	9	SCL.1	ALT0	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21
+-----Pi 4B-----+											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	

3. Circuit Connection

First, turn off power of the Pi Power & 4WD HAT,Then insert it onto the Raspberry Pi, paying attention to the assembly direction.Taking Raspberry Pi 4B as an example, assemble as shown in the following figure.

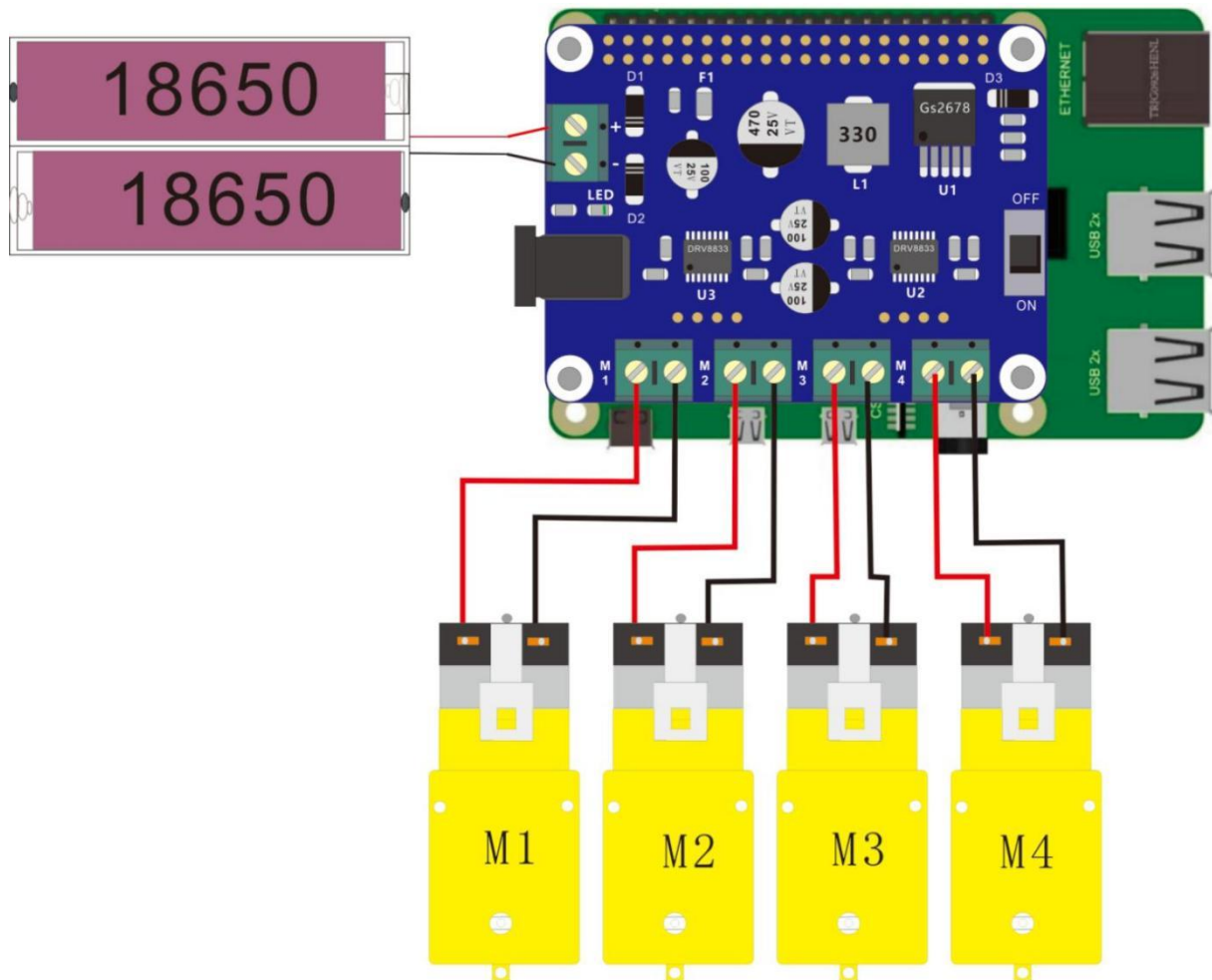


Then build the circuit according to the circuit connection diagrams. After the circuit is built and verified correct, turn on the power of Pi Power & 4WD HAT.

CAUTION: Avoid any possible short circuits (especially connecting 5V or GND, 3.3V and GND)!

WARNING: A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your RPi!

3.1 Circuit connection diagram:



4. Download Code and Run

We recommend use Python code to achieve the function.

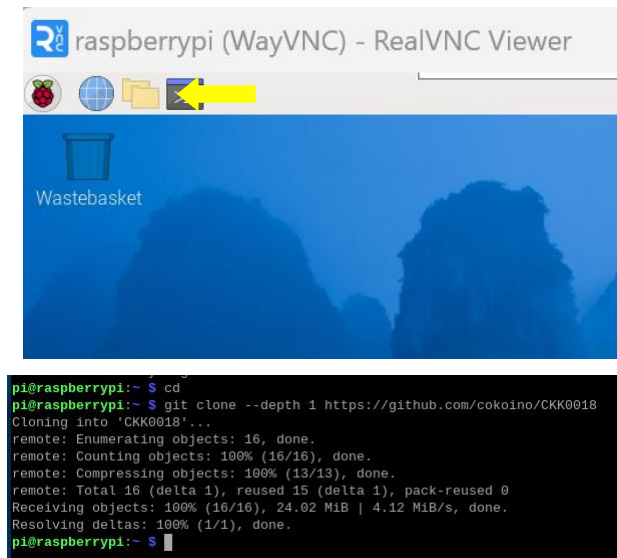
Please visit our GitHub resources at (<https://github.com/cokoino>) to download the latest available project code. We provide **Python** language code for this project .

This is the method for obtaining the code:


```
cd
git clone --depth 1 https://github.com/cokoino/CKK0018
```

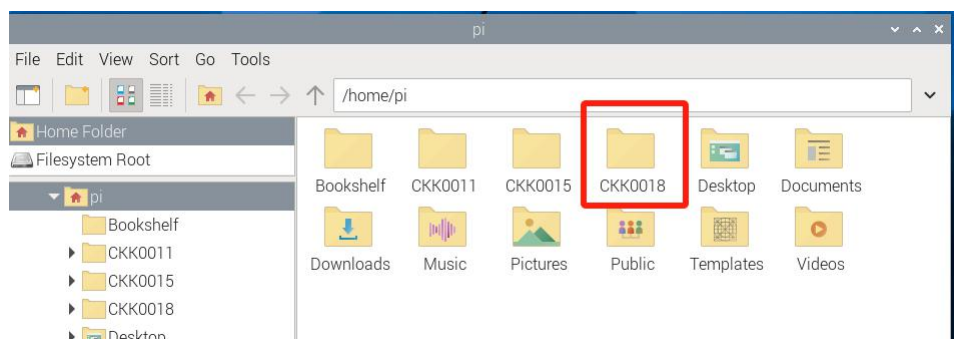
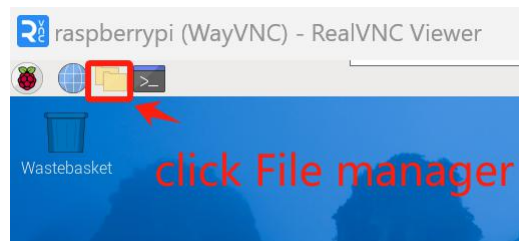
In the pi directory of the RPi terminal, enter the following command.

(There is no need for a password. If you get some errors, please check your commands.)



After the download is completed, a new folder "CKK0018" is generated, which contains all of the tutorials and required code.

Click File manager, you will find the folder "CKK0018"

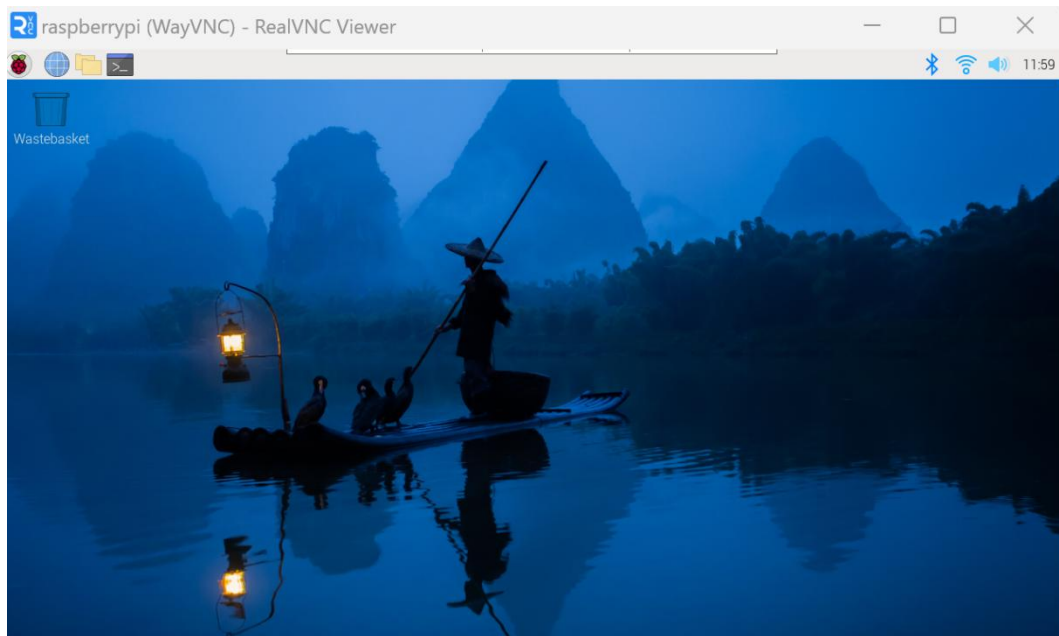


If you have no experience with Python, we suggest that you refer to this website for basic information and knowledge.

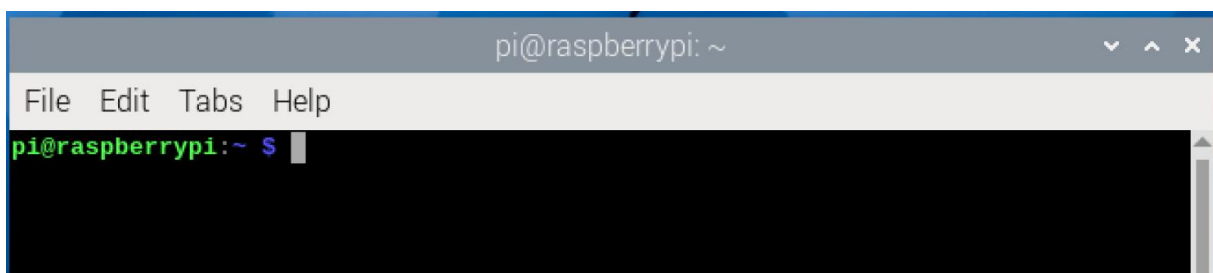
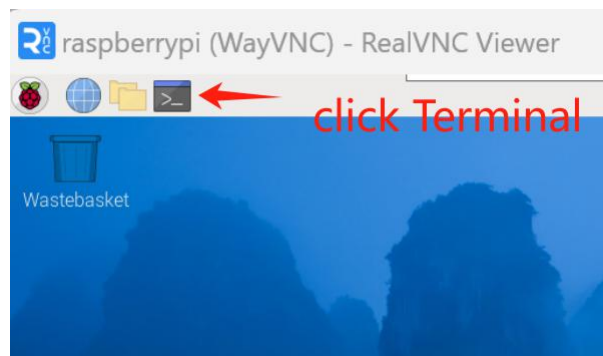
<https://python.swaroopch.com/basics.html>

Then, log in to Raspberry Pi by using VNC Viewer and operated proper setting. (For those who have

no impression or are unclear about the operation, please refer to the PDF document "2 Installing and Configuring Raspberry Pi System").



Click “Terminal”, following interface appears.



Use cd command to enter Demo1 directory of Python code.

```
cd ~/CKK0018/Tutorial/Code
```

Use python command to execute python code Drive_4_motors_run.py

```
python Drive_4_motors_run.py
```

Then the interface displays “The default speed & direction of motor is LOW & Forward rotation ... r-run s-stop f-forward b-reversal l-low m-medium h-high e-exit”

```
pi@raspberrypi: ~/CKK0018/Tutorial/Code
File Edit Tabs Help
pi@raspberrypi:~ $ cd ~/CKK0018/Tutorial/Code
pi@raspberrypi:~/CKK0018/Tutorial/Code $ python Drive_4_motors_run.py

The default speed & direction of motor is LOW & Forward rotation
r-run s-stop f-forward b-reversal l-low m-medium h-high e-exit
```

Finally, Enter the command according to the prompt information and press enter to observe whether the motor will perform the corresponding action. For example, after entering "f" and press enter, the motors start to forward rotation with the low speed; After entering "m"and press enter, the motors rotate speed change to medium;After entering "b"and press enter,the motors start to reversal rotation;After entering "s"and press enter,the motors stop rotating.

At the same time, the remote desktop interface has corresponding information display.

```
pi@raspberrypi: ~/CKK0018/Tutorial/Code
File Edit Tabs Help
pi@raspberrypi:~ $ cd ~/CKK0018/Tutorial/Code
pi@raspberrypi:~/CKK0018/Tutorial/Code $ python Drive_4_motors_run.py

The default speed & direction of motor is LOW & Forward rotation
r-run s-stop f-forward b-reversal l-low m-medium h-high e-exit

f
forward
m
medium
s
stop
b
reversal
s
stop
```

You can enter the“e”command or press “Ctrl+C” to end the program.

The following is the program code:

1. # Python Script
2. # <https://www.electronicshub.org/raspberry-pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/>
- 3.
4. **import** RPi.GPIO as GPIO
5. **from** time **import** sleep
- 6.

```

7.  #define the pin for drv8833#1
8.  NSLEEP1 = 12 #Enabling signal pin for drv8833
9.  AN11 = 17
10. AN12 = 27
11. BN11 = 22
12. BN12 = 23
13. #define the pin for drv8833#2
14. NSLEEP2 = 13
15. AN21 = 24
16. AN22 = 25
17. BN21 = 26
18. BN22 = 16
19. temp1=1
20.
21. GPIO.setmode(GPIO.BCM)
22. #Define pin as output signal
23. GPIO.setup(NSLEEP1,GPIO.OUT)
24. GPIO.setup(NSLEEP2,GPIO.OUT)
25. GPIO.setup(AN11,GPIO.OUT)
26. GPIO.setup(AN12,GPIO.OUT)
27. GPIO.setup(BN11,GPIO.OUT)
28. GPIO.setup(BN12,GPIO.OUT)
29. GPIO.setup(AN21,GPIO.OUT)
30. GPIO.setup(AN22,GPIO.OUT)
31. GPIO.setup(BN21,GPIO.OUT)
32. GPIO.setup(BN22,GPIO.OUT)
33. #Initialize the motor drive signal so that the motor is in a stopped state
34. GPIO.output(AN11,GPIO.LOW)
35. GPIO.output(AN12,GPIO.LOW)
36. GPIO.output(BN11,GPIO.LOW)
37. GPIO.output(BN12,GPIO.LOW)
38. GPIO.output(AN21,GPIO.LOW)
39. GPIO.output(AN22,GPIO.LOW)
40. GPIO.output(BN21,GPIO.LOW)
41. GPIO.output(BN22,GPIO.LOW)
42. p1=GPIO.PWM(NSLEEP1,1000)#Define p1 as a pulse signal of 1000 Hz
43. p2=GPIO.PWM(NSLEEP2,1000)#Define p2 as a pulse signal of 1000 Hz
44. p1.start(30)#P1 defaults to a duty cycle of 30%
45. p2.start(30)#P2 defaults to a duty cycle of 30%
46.
47. print("\n")
48. print("The default speed & direction of motor is LOW & Forward rotation")
49. print("r-run s-stop f-forward b-reversal l-low m-medium h-high e-exit")
50. print("\n")
51.
52. while(1):
53.
54.     x=input()
55.
56.     if x=='r':
57.         print("run")
58.         if(temp1==1):

```

```

59.     GPIO.output(AN11,GPIO.HIGH)
60.     GPIO.output(AN12,GPIO.LOW)
61.     GPIO.output(BN11,GPIO.HIGH)
62.     GPIO.output(BN12,GPIO.LOW)
63.     GPIO.output(AN21,GPIO.HIGH)
64.     GPIO.output(AN22,GPIO.LOW)
65.     GPIO.output(BN21,GPIO.HIGH)
66.     GPIO.output(BN22,GPIO.LOW)
67.     print("reversal")
68.     x='z'
69.     else:
70.         GPIO.output(AN11,GPIO.LOW)
71.         GPIO.output(AN12,GPIO.HIGH)
72.         GPIO.output(BN11,GPIO.LOW)
73.         GPIO.output(BN12,GPIO.HIGH)
74.         GPIO.output(AN21,GPIO.LOW)
75.         GPIO.output(AN22,GPIO.HIGH)
76.         GPIO.output(BN21,GPIO.LOW)
77.         GPIO.output(BN22,GPIO.HIGH)
78.     print("forward")
79.     x='z'
80.
81.
82.     elif x=='s':
83.         print("stop")
84.         GPIO.output(AN11,GPIO.LOW)
85.         GPIO.output(AN12,GPIO.LOW)
86.         GPIO.output(BN11,GPIO.LOW)
87.         GPIO.output(BN12,GPIO.LOW)
88.         GPIO.output(AN21,GPIO.LOW)
89.         GPIO.output(AN22,GPIO.LOW)
90.         GPIO.output(BN21,GPIO.LOW)
91.         GPIO.output(BN22,GPIO.LOW)
92.         x='z'
93.
94.     elif x=='f':
95.         print("forward")
96.         GPIO.output(AN11,GPIO.LOW)
97.         GPIO.output(AN12,GPIO.HIGH)
98.         GPIO.output(BN11,GPIO.LOW)
99.         GPIO.output(BN12,GPIO.HIGH)
100.        GPIO.output(AN21,GPIO.LOW)
101.        GPIO.output(AN22,GPIO.HIGH)
102.        GPIO.output(BN21,GPIO.LOW)
103.        GPIO.output(BN22,GPIO.HIGH)
104.        temp1=0
105.        x='z'
106.
107.     elif x=='b':
108.         print("reversal")
109.         GPIO.output(AN11,GPIO.HIGH)
110.         GPIO.output(AN12,GPIO.LOW)

```



```

111. GPIO.output(BN11,GPIO.HIGH)
112. GPIO.output(BN12,GPIO.LOW)
113. GPIO.output(AN21,GPIO.HIGH)
114. GPIO.output(AN22,GPIO.LOW)
115. GPIO.output(BN21,GPIO.HIGH)
116. GPIO.output(BN22,GPIO.LOW)
117. temp1=1
118. x='z'
119.
120. elif x=='l':
121.     print("low")
122.     p1.ChangeDutyCycle(30)#Set the P1 pulse signal duty cycle to 30%
123.     p2.ChangeDutyCycle(30)#Set the P2 pulse signal duty cycle to 30%
124.     x='z'
125. elif x=='m':
126.     print("medium")
127.     p1.ChangeDutyCycle(60)#Set the P1 pulse signal duty cycle to 60%
128.     p2.ChangeDutyCycle(60)#Set the P2 pulse signal duty cycle to 60%
129.     x='z'
130. elif x=='h':
131.     print("high")
132.     p1.ChangeDutyCycle(90)#Set the P1 pulse signal duty cycle to 90%
133.     p2.ChangeDutyCycle(90)#Set the P2 pulse signal duty cycle to 90%
134.     x='z'
135. elif x=='e':
136.     GPIO.cleanup()
137.     print("GPIO Clean up")
138.     break
139.
140. else:
141.     print("<<< wrong data >>>")
142.     print("please enter the defined data to continue.....")

```

About RPi.GPIO:

RPi.GPIO

This is a Python module to control the GPIO on a Raspberry Pi. It includes basic output function and input function of GPIO, and functions used to generate PWM.

GPIO.setmode(mode)

Sets the mode for pin serial number of GPIO.

mode=GPIO.BOARD, which represents the GPIO pin serial number based on physical location of RPi. mode=GPIO.BCM, which represents the pin serial number based on CPU of BCM chip.

GPIO.setup(pin,mode)

Sets pin to input mode or output mode, “pin” for the GPIO pin, “mode” for INPUT or OUTPUT.

GPIO.output(pin,mode)

Sets pin to output mode, “pin” for the GPIO pin, “mode” for HIGH (high level) or LOW (low level).

For more functions related to RPi.GPIO, please refer to:

<https://sourceforge.net/p/raspberry-gpio-python/wiki/Examples/>

“import time” time is a module of python.

<https://docs.python.org/2/library/time.html?highlight=time%20time#module-time>

In subfunction setup(), GPIO.setmode (GPIO.BOARD) is used to set the serial number for GPIO based on physical location of the pin.

GPIO Numbering Relationship

WingPi	BCM(Extension)	Physical		BCM(Extension)	WingPi
3.3V	3.3V	1	2	5V	5V
8	GPIO2/SDA1	3	4	5V	5V
9	GPIO3/SCL1	5	6	GND	GND
7	GPIO4	7	8	GPIO14/TXD0	15
GND	GND	9	10	GPIO15/RXD0	16
0	GPIO17	11	12	GPIO18	1
2	GPIO27	13	14	GND	GND
3	GPIO22	15	16	GPIO23	4
3.3V	3.3V	17	18	GPIO24	5
12	GPIO10/MOSI)	19	20	GND	GND
13	GPIO9/MOIS	21	22	GPIO25	6
14	GPIO11/SCLK	23	24	GPIO8/CE0	10
GND	GND	25	26	GPIO7/CE1	11
30	GPIO0/SDA0	27	28	GPIO1/SCL0	31
21	GPIO5	29	30	GND	GND
22	GPIO6	31	32	GPIO12	26
23	GPIO13	33	34	GND	GND
24	GPIO19	35	36	GPIO16	27
25	GPIO26	37	38	GPIO20	28
GND	GND	39	40	GPIO21	29

5. Trouble shooting

Using RealVNC Viewer as a remote desktop connection does not connect to Raspberry Pi, and Raspberry Pi does not display when connected to the monitor.

Please check the battery level that supplies power to the cokoino Pi Power&4WD HAT. When the battery voltage is below 6.8V, the Raspberry Pi cannot be powered on. Please fully charge the battery before use.

6. Any questions and suggestions are welcome

THANK YOU for participating in this testing experience!

If you find any errors, omissions or you have suggestions and/or questions about this lesson, please feel free to contact us:

cokoino@outlook.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our Amazon Store by search for "**LK COKOINO**" on Amazon. We will continue to launch fun, cost-effective, innovative and exciting products.

Thank you again for choosing Cokoino products.

LK COKOINO