Hardware Control course--Drive buzzer

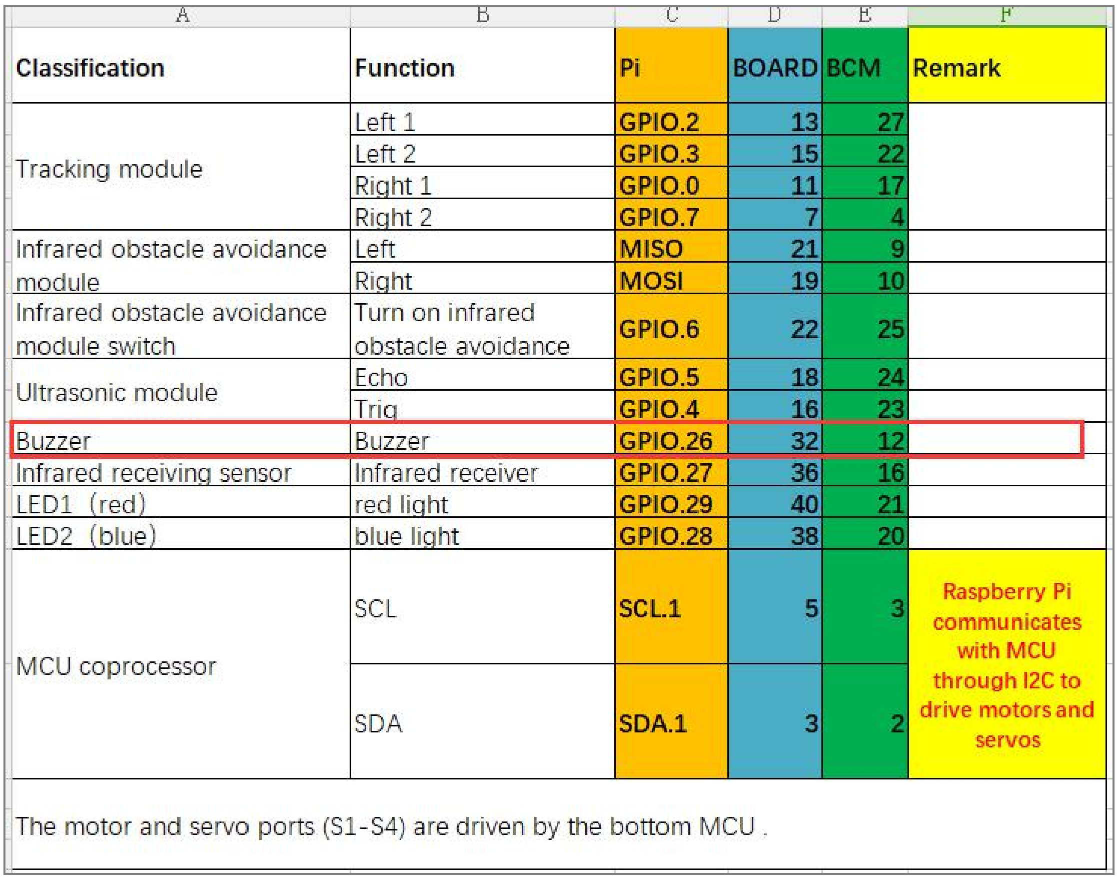
1. Learning target

In this course, we will learn how to drive buzzer on car.

2. Principle

Buzzers are divided into two types: "active buzzers" and "passive buzzers".
Active means that they possess a multi-vibrator inside. It only needs to provide the working voltage externally, it can emit a fixed frequency sound.
Passive means that there is no internal oscillation source, and an external drive circuit is required to provide a certain frequency of the drive signal.
In this experiment, we will use passive buzzer.

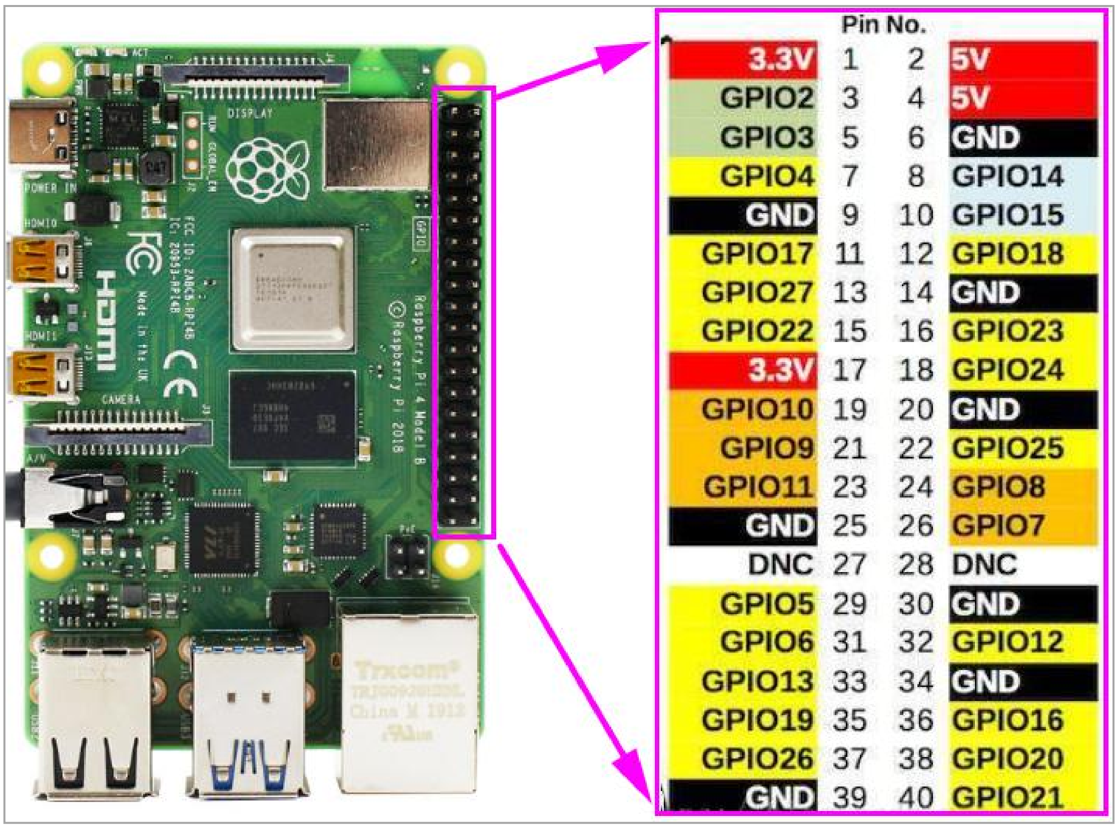
From the hardware interface manual, we can see that buzzer are directly driven by Pin 32 of Raspberry Pi.

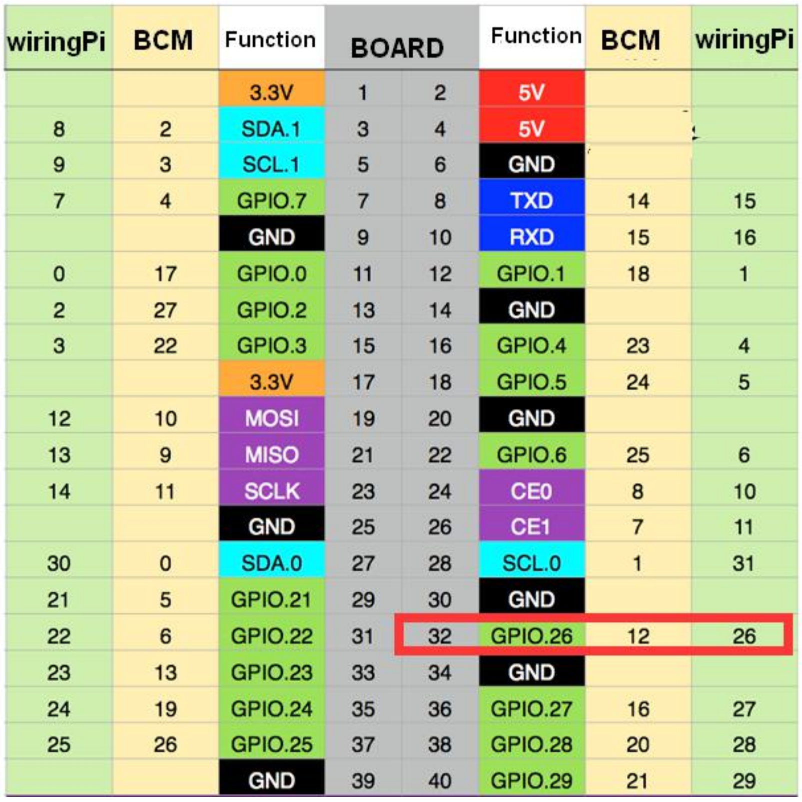


A	B	C	D	E	F
Classification	**Function**	**Pi**	**BOARD**	**BCM**	**Remark**
Tracking module	Left 1	GPIO.2	13	27	
	Left 2	GPIO.3	15	22	
	Right 1	GPIO.0	11	17	
	Right 2	GPIO.7	7	4	
Infrared obstacle avoidance module	Left	MISO	21	9	
	Right	MOSI	19	10	
Infrared obstacle avoidance module switch	Turn on infrared obstacle avoidance	GPIO.6	22	25	
Ultrasonic module	Echo	GPIO.5	18	24	
	Trig	GPIO.4	16	23	
Buzzer	Buzzer	GPIO.26	32	12	
Infrared receiving sensor	Infrared receiver	GPIO.27	36	16	
LED1（red)	red light	GPIO.29	40	21	
LED2（blue)	blue light	GPIO.28	38	20	
MCU coprocessor	SCL	SCL.1	5	3	Raspberry Pi communicates with MCU through I2C to drive motors and servos
	SDA	SDA.1	3	2	
The motor and servo ports (S1-S4) are driven by the bottom MCU .					

3. Coding method

The pin comparison table of Raspberry Pi is shown in the figure below.





wiringPi	BCM	Function	BOARD		Function	BCM	wiringPi
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

In this course, we use BOARD coding method.
we need to control the 32 pins of the Raspberry Pi board in the code.

4. About code

Path: /home/pi/Yahboom_project/Raspbot/2.Hardware Control course/1.Drive buzzer/buzzer test.ipynb

1) Import time and RPI.GPIO library

```
import time
import RPi.GPIO as GPIO
```

2) Set GPIO pin number mode to BOARD and buzzer pin to output mode. Create the PWM output, the duty cycle is 50, and the range is 0-100.

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(32, GPIO.OUT)

p = GPIO.PWM(32, 440)
p.start(50)
```

3) Switch the duty of PWM between 0-100 and keep the cycle in this state

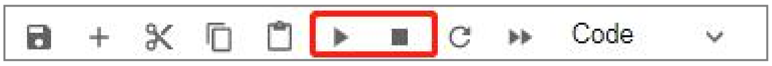
```
try:
    while 1:
        for dc in range(0, 101, 5):
            print ('start_1')
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            print ('start_2')
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
```

4) When we click the stop button in the JupyterLab menu bar, after exiting the loop, stop the PWM and clear the GPIO.

```
p.stop()
print("Ending")
GPIO.cleanup()
```

5. Running code

Click the button shown in the figure below to run the program on the Jupyter Lab interface



6. Experimental phenomena

We will hear the buzzer whistle.

Path: /home/pi/Yahboom_project/Raspbot/2.Hardware Control course/1.Drive buzzer/buzzer sing.ipynb

1) Import time and RPI.GPIO library

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
import time

Buzzer = 32 [illegible]
```

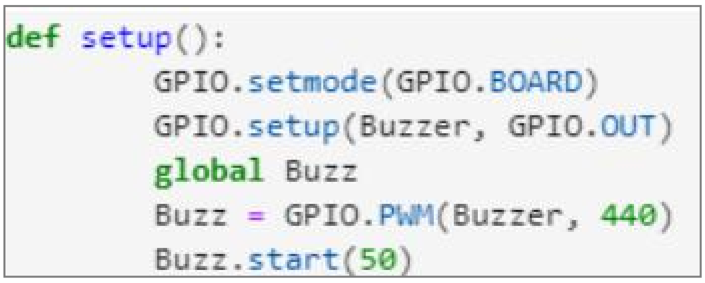
2) According to the frequency list, define the note and beat list of the song by comparing the numbered musical notation of the song.

```
BL1 = 248
BL2 = 278
BL3 = 294
BL4 = 330
BL5 = 371
BL6 = 416
BL7 = 467

B1 = 495
B2 = 556
B3 = 624
B4 = 661
B5 = 742
B6 = 833
B7 = 935

BH1 = 990
BH2 = 1112
BH3 = 1178
BH4 = 1322
BH5 = 1484
BH6 = 1665
BH7 = 1869
```

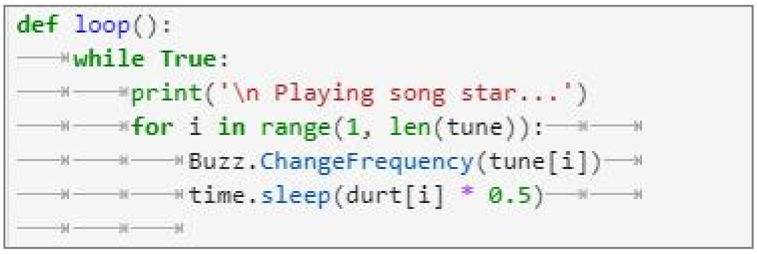
```
tune =  [
NTC3, NTC5, NTC5, NTC3, NTC6, NTC6, NTC7, NTC6, NTC6, NTC6, NTC5, NTCH1, NTCH1, NTCH1, NTCH1, NTC6,
 NTCH1, NTC6, NTC5,

NTC3, NTC5, NTC5, NTC5, NTC3, NTC6, NTC6, NTC7, NTC6, NTC6, NTC6, NTC5, NTCH1, NTCH1, NTCH1,
 NTCH1, NTC6, NTC6, NTCH1,NTCH2,

NTCH5, NTCH5, NTCH5, NTCH5,NTCH5, NTCH3, NTCH2, NTCH1, NTCH1, NTC6, NTCH1, NTC6, NTCH1, NTCH2,
  NTCH2, NTCH2, NTCH2,NTCH2, NTCH1, NTCH3, NTCH2, NTCH2,

NTCH3, NTCH3, NTCH3, NTCH3, NTCH2, NTCH2, NTCH1, NTCH1, NTCH1, NTCH2, NTCH1, NTC6, NTC5, NTC5,
 NTC5, NTC5, NTC6, NTC5,NTCH2, NTCH3,NTCH1,
]
```

```
durt = [
0.5, 0.5, 1.5, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 2,
0.5, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
2,0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 1, 0.5, 0.5, 0.5, 1, 0.25,0.5, 0.5, 0.5, 0.5,
1, 0.25, 2, 0.5, 1, 0.5, 0.5, 0.5, 1, 0.5, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 1, 0.5, 0.5,
0.5, 0.5, 0.5, 2,
]
```

3) Define the initialization function--setup to set GPIO to BOARD coding mode, and set buzzer pin to PWM output.
Running function loop to play music according to the frequency list.

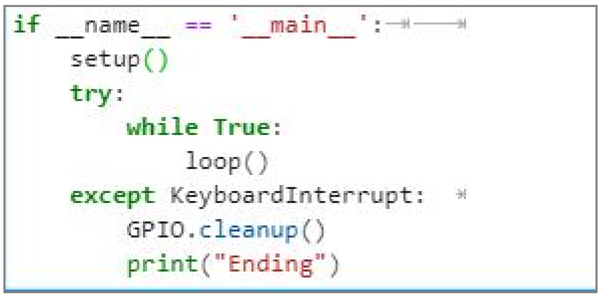


```
def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(Buzzer, GPIO.OUT)
        global Buzz
        Buzz = GPIO.PWM(Buzzer, 440)
        Buzz.start(50)
```



```
def loop():
	while True:
		print('\n Playing song star...')
		for i in range(1, len(tune)):
			Buzz.ChangeFrequency(tune[i])
			time.sleep(durt[i] * 0.5)
```

4) Call the initialization loop function to play music, and exit the loop when the square stop button is clicked, and clear the GPIO.



```
if __name__ == '__main__':
    setup()
    try:
        while True:
            loop()
    except KeyboardInterrupt:
        GPIO.cleanup()
        print("Ending")
```

After running the program, you can hear the buzzer play music