5.3 Car singing and dancing

Note: Motor speed is affected by battery power.
For this course, when the battery power is high (the power value is above 26000), if the battery power is not enough, we need to charge battery in time or modify the motor speed in the code.

1. Learning Objectives

In this course, we will learn how to drive motors and buzzer play music on Pico robot.

2. About hardware

We need use motors and buzzer on Pico robot.

Note: The effect of the following code is that the car advances 1 second. Before running this code, please make sure the wheels are off the ground or table to avoid impact damage to the car.

3. About code

Code path: Code -> 3.Robotics course -> 3.Singing dancing.py

```
from pico_car import pico_car, ws2812b
import time
from machine import Pin, PWM

Motor = pico_car()
# set buzzer pin
BZ = PWM(Pin(22))
BZ.freq(1000)
num_leds = 8  # Number of NeoPixels
# Pin where NeoPixels are connected
pixels = ws2812b(num_leds, 0)
# Set all led off
pixels.fill(0,0,0)
pixels.show()
# Initialize music
CM = [0, 330, 350, 393, 441, 495, 556, 624]
song = [CM[1],CM[1],CM[5],CM[5],CM[6],CM[6],CM[5],
        CM[4],CM[4],CM[3],CM[3],CM[2],CM[2],CM[1],]
beat = [ 0.5,0.5,0.5,0.5,0.5,0.5,1,0.5,0.5,0.5,0.5,0.5,0.5,1,]
# music
def music_Run():
    for i in range(0,2):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
        time.sleep(0.01)
def music_Back():
    for i in range(2,4):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
```

```
        time.sleep(0.01)
def music_Left():
    for i in range(4,7):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
        time.sleep(0.01)
def music_Right():
    for i in range(7,9):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
        time.sleep(0.01)
def music_TLeft():
    for i in range(9,11):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
        time.sleep(0.01)
def music_TRight():
    for i in range(11,14):
        BZ.duty_u16(500)
        BZ.freq(song[i])
        time.sleep(beat[i])
        BZ.duty_u16(0)
        time.sleep(0.01)
#Car forward
Motor.Car_Run(255,255)
for i in range(num_leds):
    pixels.set_pixel(i,255,255,0)
pixels.show()
music_Run()
#Car back
Motor.Car_Back(255,255)
for i in range(num_leds):
    pixels.set_pixel(i,0,255,255)
pixels.show()
music_Back()
#left
Motor.Car_Run(0,255)
for i in range(num_leds):
    pixels.set_pixel(i,255,0,255)
pixels.show()
music_Left()
#right
Motor.Car_Run(255,0)
for i in range(num_leds):
    pixels.set_pixel(i,255,0,0)
pixels.show()
music_Right()
#Turn left
Motor.Car_Left(255,255)
for i in range(num_leds):
    pixels.set_pixel(i,0,255,0)
pixels.show()
```

```
music_TLeft()
#Turn right
Motor.Car_Right(255,255)
for i in range(num_leds):
    pixels.set_pixel(i,0,0,255)
pixels.show()
music_TRight()
#Car stop
Motor.Car_Stop()
for i in range(num_leds):
    pixels.set_pixel(i,0,0,0)
pixels.show()
```

from pico_car import pico_car, ws2812b

Use pico_car and ws2812b of pico_car to encapsulate the motor driver and RGB light library.

import time

The "time" library. This library handles everything time related, from measuring it to inserting delays into programs. The unit is seconds.

from machine import Pin, PWM

The machine library contains all the instructions that MicroPython needs to communicate with Pico and other MicroPython-compatible devices, extending the language of physical computing, using the Pin and PWM libraries here.

Motor = pico_car()

Initialize the motor drive.

Motor.Car_Run(255,255)

To control the car to move forward, the speed is set to 255, the parameters are (left motor speed, right motor speed), and the speed range is 0-255.

Motor.Car_Stop()

Control the car to stop.

Motor.Car_Back(255,255)

Control the car to back up.

Motor.Car_Run(0,255)

Control the car to turn left.

Motor.Car_Run(255,0)

Control the car to turn right.

Motor.Car_Left(255,255)

Control the car to turn left.

Motor.Car_Right(255,255)

Control the car to rotate right.

pixels = ws2812b(num_leds, 0)

Initialize RGB lights, we have 8 RGB lights, here num_leds is set to 8.

pixels.fill(0,0,0)

Set all lights to 0,0,0, that is, turn off all lights, the parameters are (red, green, blue), and the color brightness is 0-255.

pixels.show()

Display the set lights.

pixels.set_pixel(i,255,255,0)

Use a for loop to set all lights to yellow.

music_Run()

In this function, we cut the previously played music of the buzzer into 6 parts, and each part runs different actions of the car.

4. Experimental Phenomenon

After the code is downloaded, we can see that robot forward and RGB light becomes yellow ---> robot back and RGB light becomes cyan---> robot turn left and RGB light becomes purple---> robot turn right and RGB light becomes red--->robot spin left and RGB light become green---> robot spin right and RGB light become blue--->robot stop and turn off the RGB lights.