Hardware Control course-- IR control car

Tips:

1)In order to avoid the interference of sunlight on the infrared sensor, we must carry out the experiment indoors.

2) During remote control, the infrared remote control needs to be aimed at the infrared receiver on the expansion board.

1. Learning target

In this course, we mainly learn how to use IR controller control robot car.

2. Principle of experimental

Infrared sending and receiving codes are divided into: **guide code, user code, user reverse code, operation code(data code), operation reverse code(data reverse code).**

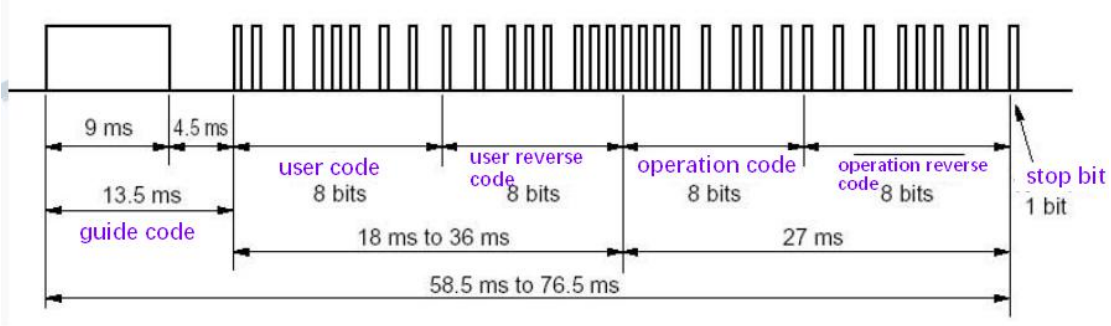
Below we will introduce these kinds of encoded waveform:

For the part of infrared transmission, the infrared remote control we provided has been set up. Therefore, we do not explain the knowledge about infrared transmission here, but mainly explain some points of infrared reception.

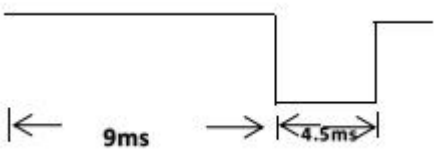
We mainly use the **NCE** protocol:

The protocol stipulates that the lower bits are transmitted first.

A string of information first sends a 9ms high pulse of AGC (Automatic Gain Control) . Then, send a 4.5ms low level. Next, send four-byte address code and a command code. These four bytes are : Address code, address reverse code; command code; command reverse code.



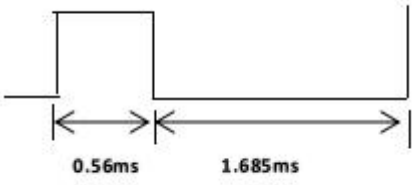
1)Guide code: The guide code defined by the uPD6121G of the NEC protocol is 9ms high level+ 4.5ms low level, waveform as shown in the figure below:



2)User code and operation code: The user code and operation code defined by the uPD6121G of the NEC protocol is:

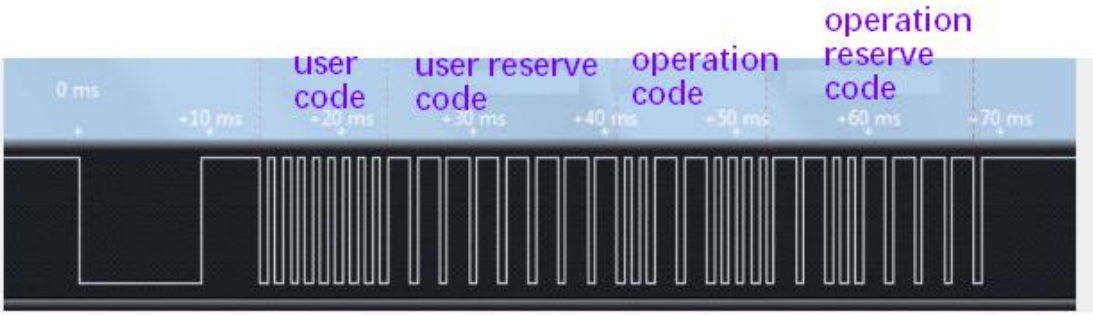
"Logic 0" : 0.56ms high level + 0.565ms low level;

"Logic1" : 0.56ms high level + 1.685ms low level;
Waveform as shown below :

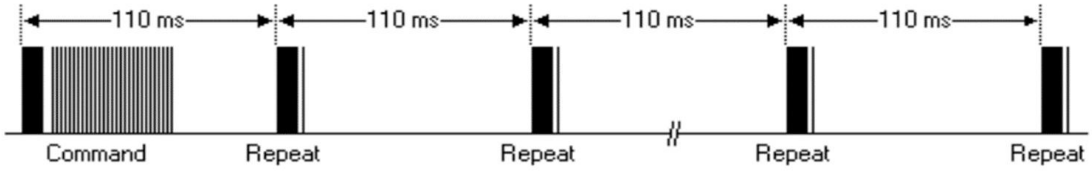


Logic 0 --waveform Logic 1--waveform

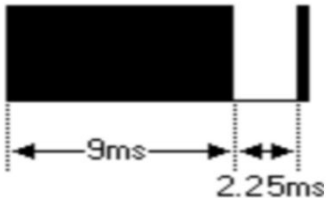
The wave output after passing through the infrared receiver, as shown below:



If you keep pressing that button, a series of messages can only be sent once. If you keep pressing, the repeating code is sent with a period of 110ms. Waveform as shown below.



The repeat code is composed of a high AGC level of 9ms, a low level of 2.25ms, and a high level of 560us, until the key is released. Waveform as shown below:



The process of infrared receiving and controlling the car is as follows:

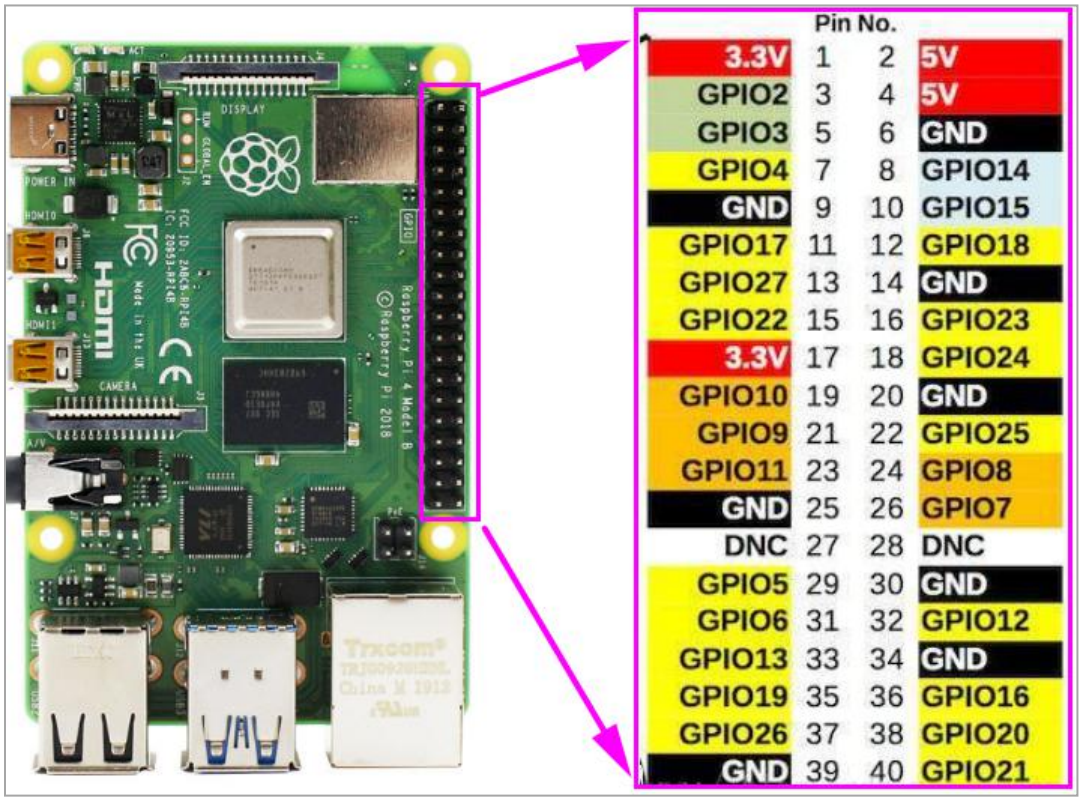
1) Generate a falling edge, enter the interrupt function of external interrupt 0, and check whether the IO port is still low level after delay time. If it is low level, wait for the low level of 9ms to pass. Wait for the 9ms low level to pass, and then wait for the 4.5ms high level to pass.
2) Start receiving the 4 sets of data transmitted, wait for the low level of 560us to

pass, and detect the duration of the high level. If it exceeds 1.12ms, it is high level (the duration of the high level is 1.69ms, the low level Duration is 5.65ms.)
3) Detect whether the received data is the same as the data reverse code (operation reverse code), and wait for the same data. If it is the same, it is proved that the corresponding data sent by the infrared remote control is received. Then, the car moves according to the function set in the program.

For the Raspbot car, we use 4 TT DC gear motors. They are driven by the TB6612 chip. The driver chip is not directly connected to the Raspberry Pi pins.
Raspberry Pi communicates with STM8 MUC through IIC, and then STM8 MCU drives TB6612 chip to drive the motor.

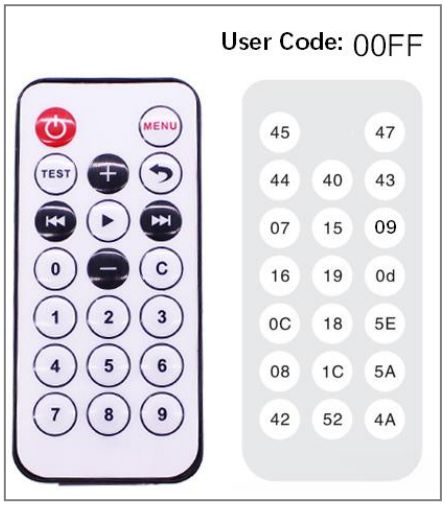
3. Coding method

In this course, we use BOARD coding method.
According to the hardware interface manual, we see that IR recevicer connected to the 36 pin of the Raspberry Pi board.
STM8 is connected to SDA.1, SCL.1 on the Raspberry Pi board.
We have provided a library text dedicated to driving motors and servos --YB_Pcb_Car.py.



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

The code value corresponding to each key of IR controller as shown below.



4. About code

Path: /home/pi/Yahboom_project/Raspbot/2.Hardware Control course/010.IR control car/IR control car.ipynb

1) Import time, GPIO and YB_Pcb_Car library, and define the car class is used to drive motors or servos.

Import library

```
#!/usr/bin/python3
# -*- coding:utf-8 -*-
import RPi.GPIO as GPIO
import time
import YB_Pcb_Car

car = YB_Pcb_Car.YB_Pcb_Car()
```

2) Set the GPIO coding mode, set the GPIO of the infrared receiver to the pull-up input mode. Define whistle function.

```
PIN = 36; #Define IR pin
buzzer = 32; #Define buzzer pin

#Set the GPIO port to BIARD encoding mode
GPIO.setmode(GPIO.BOARD)

#Ignore the warning message
GPIO.setwarnings(False)

ir_repeat_cnt = 0

def init():
    GPIO.setup(PIN,GPIO.IN,GPIO.PUD_UP)  #The pin of the red external device needs to be se
    GPIO.setup(buzzer,GPIO.OUT)          #Buzzer pin be set to output mode

    print("IR control start...")

#whistle
def whistle():
    p = GPIO.PWM(buzzer, 440)
    p.start(50)
    time.sleep(0.5)
    p.stop()
```

3) Judge the code value of the received infrared remote control button and complete the corresponding movement.

```
def exec_cmd(key_val):
    if key_val==0x45:  #Power button
        car.Ctrl_Servo(1, 90)
        car.Ctrl_Servo(2, 90)
        car.Car_Stop()
    elif key_val==0x40:   #+ button
        car.Car_Run(100, 100)   #car advance
    elif key_val==0x15:   #Stop button
        car.Car_Stop()
    elif key_val==0x07:   #Left button
        car.Car_Left(100, 100)
    elif key_val==0x47:   #MENU button
        whistle()         #buzzer whistle
    elif key_val==0x09:   #Right button
        car.Car_Right(100, 100)
    elif key_val==0x16:   #0 button
        car.Car_Spin_Left(100, 100)
    elif key_val==0x19:   #- button
        car.Car_Back(100, 100)
    elif key_val==0x0d:   #C button
        car.Car_Spin_Right(100, 100)
    elif key_val==0x0c:   #1 button
        car.Ctrl_Servo(1, 0)
    elif key_val==0x18:   #2 button
        car.Ctrl_Servo(1, 90)
    elif key_val==0x5e:   #3 button
        car.Ctrl_Servo(1, 180)
    elif key_val==0x08:   #4 button
        car.Ctrl_Servo(2, 0)
    elif key_val==0x1c:   #5 button
        car.Ctrl_Servo(2, 90)
    elif key_val==0x5a:   #6 button
        car.Ctrl_Servo(2, 180)
    else:
        print(key_val)
        print("no cmd")
```

4) According to the NCE protocol, receive the key value sent by the infrared remote control and judge the release.

```
try:
    init()
    while True:
        if GPIO.input(PIN) == 0:
            ir_repeat_cnt = 0;
            count = 0
            while GPIO.input(PIN) == 0 and count < 200:
                count += 1
                time.sleep(0.00006)

            count = 0
            while GPIO.input(PIN) == 1 and count < 80:
                count += 1
                time.sleep(0.00006)

            idx = 0
            cnt = 0
            data = [0,0,0,0]
            for i in range(0,32):
                count = 0
                while GPIO.input(PIN) == 0 and count < 15:
                    count += 1
                    time.sleep(0.00006)

                count = 0
                while GPIO.input(PIN) == 1 and count < 40:
                    count += 1
                    time.sleep(0.00006)

                if count > 9:
                    data[idx] |= 1<<cnt
                if cnt == 7:
                    cnt = 0
                    idx += 1
                else:
                    cnt += 1
            if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF:
                print("Get the key: 0x%02x" %data[2])
                exec_cmd(data[2])
        else:
            if ir_repeat_cnt > 110:
                ir_repeat_cnt = 0
                car.Car_Stop()
            else:
                time.sleep(0.001)
                ir_repeat_cnt += 1
except KeyboardInterrupt:
    pass
print("Ending")
GPIO.cleanup()
```

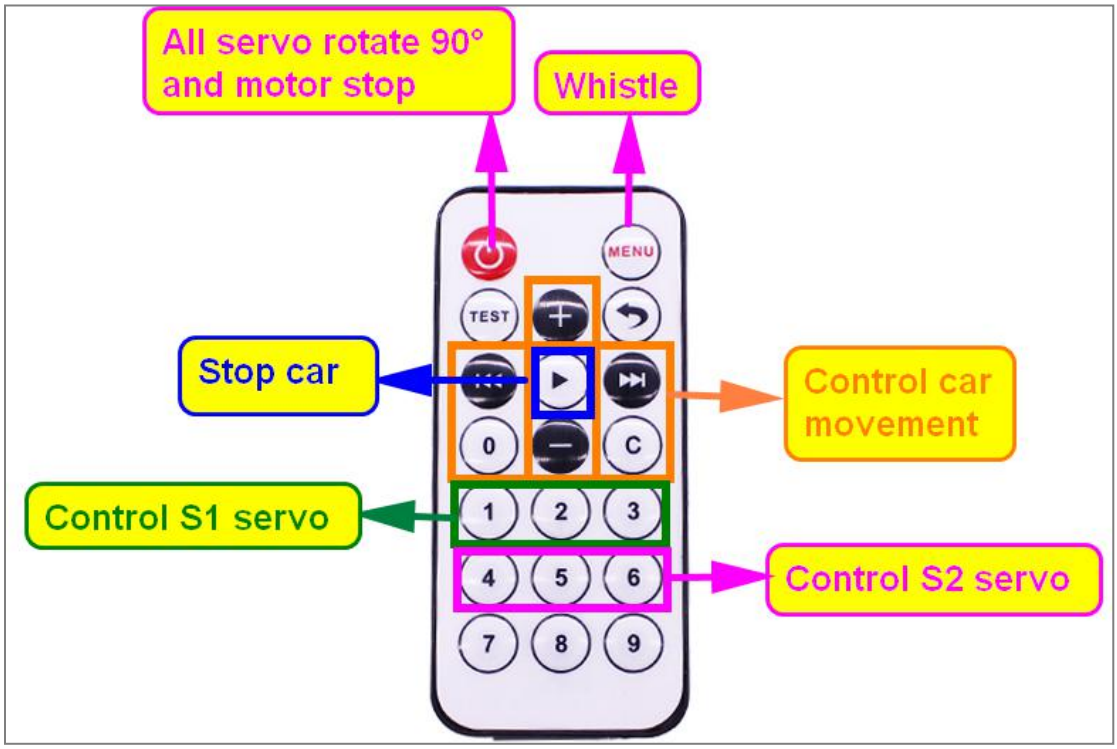
5. Running code

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

6. Experimental phenomena

After the program runs, point the infrared remote control at the infrared receiver, and press different buttons to complete the corresponding actions.

The corresponding functions of the infrared remote control as shown below.


All servo rotate 90° and motor stop
Whistle
MENU
TEST
Stop car
Control car movement
0
C
Control S1 servo
1
2
3
4
5
6
Control S2 servo
7
8
9