

## Hardware Control course-- IR control test

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 make the infrared receiver on the car read the value of the infrared remote control.

### 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)**.

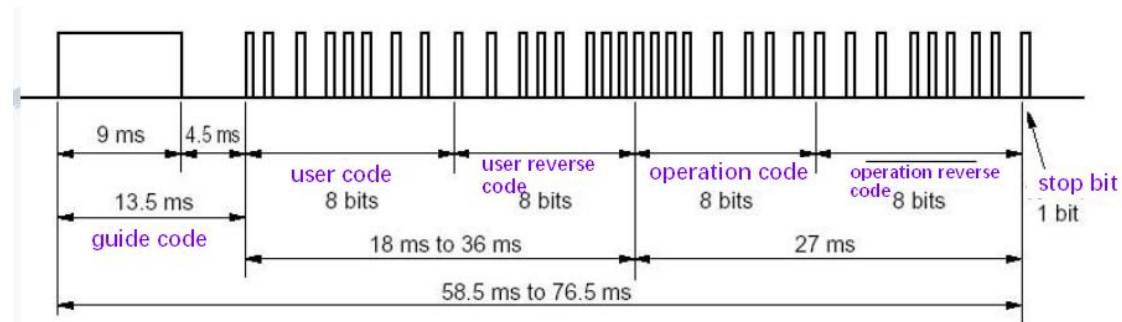
Below we will introduce these kinds of encoded waveform:

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:

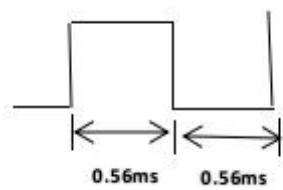


**2)User code and operation code:** The user code and operation code defined by the uPD6121G of the NEC protocol is:

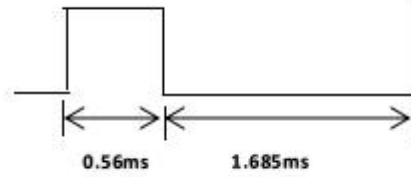
"Logic 0" : 0.56ms high level + 0.565ms low level;

"Logic1" : 0.56ms high level + 1.685ms low level;

Waveform as shown below :

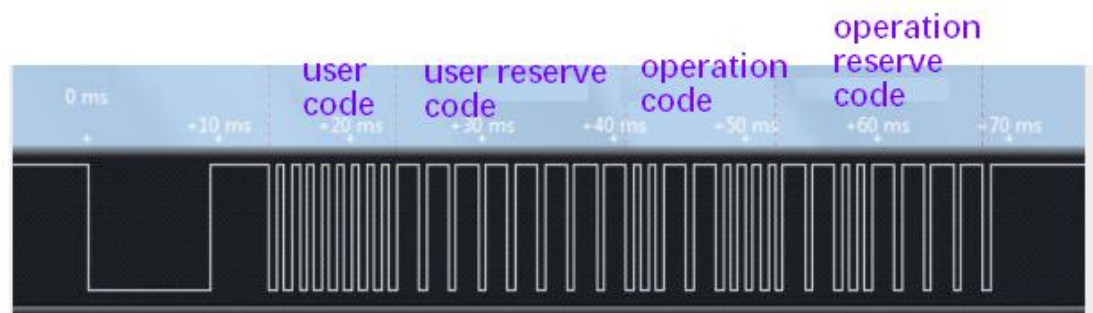


Logic 0 --waveform

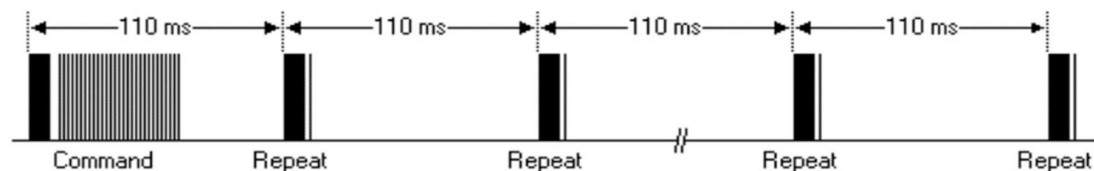


Logic 1--waveform

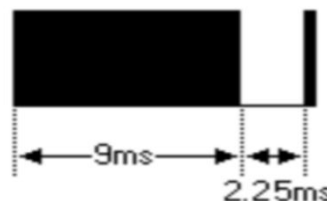
The wave output after passing through the infrared receiver, as shown below:



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.)

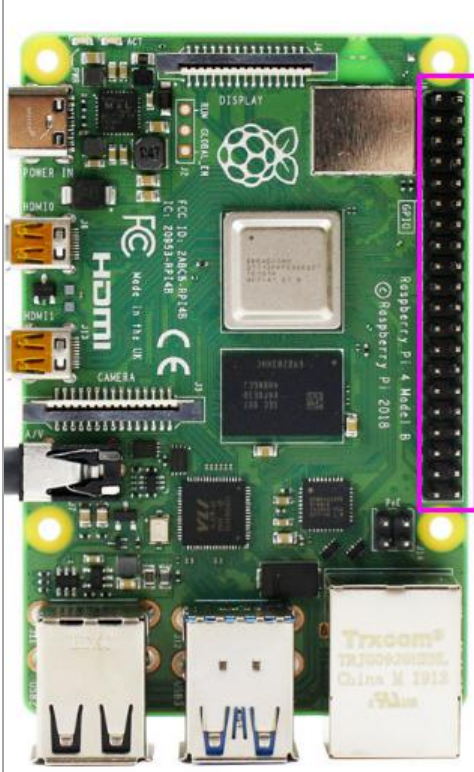
3) Detect whether the received data is the same as the data reverse code (operation reverse code), and wait for the same data. If it is the same, it is proved that the corresponding data sent by the infrared remote control is received. Then, the car moves according to the function set in the program.

### 3. Coding method

In this course, we use BOARD coding method.

According to the hardware interface manual, we know that the Raspberry Pi pin that the infrared receiving sensor is connected to is 36.

The pin comparison table of Raspberry Pi is shown below.



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

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

#### 4. About code

Path: [/home/pi/Yahboom\\_project/Raspbot/2.Hardware Control course/09.IR control test/IR control test.ipynb](#)

##### 1) Import time and GPIO library

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

##### 2) Set the GPIO coding mode, set the GPIO of the infrared receiver to the pull-up input mode

Define the pins.

Set the coding method of GPIO and initial settings.

```
PIN = 36; #Define IR pin

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

#Ignore the warning message
GPIO.setwarnings(False)
ir_repeat_cnt = 0

#The pin of the red external device needs to be set to input pull-up
GPIO.setup(PIN,GPIO.IN,GPIO.PUD_UP)
print("IR test start...") #Initially print "IR test start".
```

3) According to the NCE protocol, it receives the pilot code, address code, address inversion, signal code, and signal inversion of the infrared signal.

```
try:
    print("start")
    while True:
        if GPIO.input(PIN) == 0: #The signal emitted by the infrared remote control is detected
            ir_repeat_cnt = 0;
            count = 0
            while GPIO.input(PIN) == 0 and count < 200: #Judge the boot code of 9ms high Level pulse
                count += 1
                time.sleep(0.00006)

            count = 0
            while GPIO.input(PIN) == 1 and count < 80: #Judge the boot code of 4.5ms low-Level pulse
                count += 1
                time.sleep(0.00006)

            idx = 0
            cnt = 0
            data = [0,0,0,0] #Define data used to store the address code, address inversion, signal code, and :
            for i in range(0,32): #data[0],data[1],data[2],data[3] In total, 8bit*4=32
                count = 0
                while GPIO.input(PIN) == 0 and count < 10: #Start decoding, used to filter the first 560us pul:
                    count += 1
                    time.sleep(0.00006)

                count = 0
                while GPIO.input(PIN) == 1 and count < 40: #After the 560us high-Level pulse, check the remain:
                    ...

                Description:
                According to the infrared NCE agreement:
                The period of logic 1 is 2.25ms, and the pulse time is 0.56ms.Total period-pulse time = the v

                The logic 0 period is 1.12 and the time is 0.56ms. Total period-pulse time = the value we set

                ...
                count += 1
                time.sleep(0.00006)
```



```

if count > 9:
    #This code is used to determine whether the currently received signal is
    #If count>9, it proves that the duration of the current low-level signal
    #For example: when count=10, the low-level signal is 10*60=600us (greater

    data[idx] |= 1<<cnt
if cnt == 7: #When cnt=7, one byte is full, and the next byte is ready to b
    cnt = 0
    idx += 1
else:
    cnt += 1
if data[0]+data[1] == 0xFF and data[2]+data[3] == 0xFF: #It is judged that the c
    print("Get the key: 0x%02x" %data[2]) #Print the command code obtained

```

#### 4) Determine whether the button is released according to the repeated signal

```

else:
    if ir_repeat_cnt > 110: #Judge whether the infrared remote control button is released, because the repetition cyc
        ir_repeat_cnt = 0
    else:
        time.sleep(0.001)
        ir_repeat_cnt += 1

```

### 5. Running code

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



### 6. Experimental phenomena

When we press the infrared remote control, the code value of the button will be printed out.