4.1 Using of GPIO

1. Introduction to GPIO port resources

As shown in the figure below, it is the GPIO port table on the Jetson Nano of the main control board used by Jetbot. Its GPIO port is the same as the Raspberry Pi interface. Its driver file and its usage method are all related to the Raspberry Pi GPIO port. similar.

Jetson Nano J41 Header

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
	3.3 VDC *Power*	1	2	**5.0 VDC** *Power*	
	I2C_2_SDA *I2C Bus 1*	3	4	**5.0 VDC** *Power*	
	I2C_2_SCL *I2C Bus 1*	5	6	**GND**	
gpio216	**AUDIO_MCLK**	7	8	**UART_2_TX** */dev/ttyTHS1*	
	GND	9	10	**UART_2_RX** */dev/ttyTHS1*	
gpio50	**UART_2_RTS**	11	12	**I2S_4_SCLK**	gpio79
gpio14	**SPI_2_SCK**	13	14	**GND**	
gpio194	**LCD_TE**	15	16	**SPI_2_CS1**	gpio232
	3.3 VDC *Power*	17	18	**SPI_2_CS0**	gpio15
gpio16	**SPI_1_MOSI**	19	20	**GND**	
gpio17	**SPI_1_MISO**	21	22	**SPI_2_MISO**	gpio13
gpio18	**SPI_1_SCK**	23	24	**SPI_1_CS0**	gpio19
	GND	25	26	**SPI_1_CS1**	gpio20
	I2C_1_SDA *I2C Bus 0*	27	28	**I2C_1_SCL** *I2C Bus 0*	
gpio149	**CAM_AF_EN**	29	30	**GND**	
gpio200	**GPIO_PZ0**	31	32	**LCD_BL_PWM**	gpio168

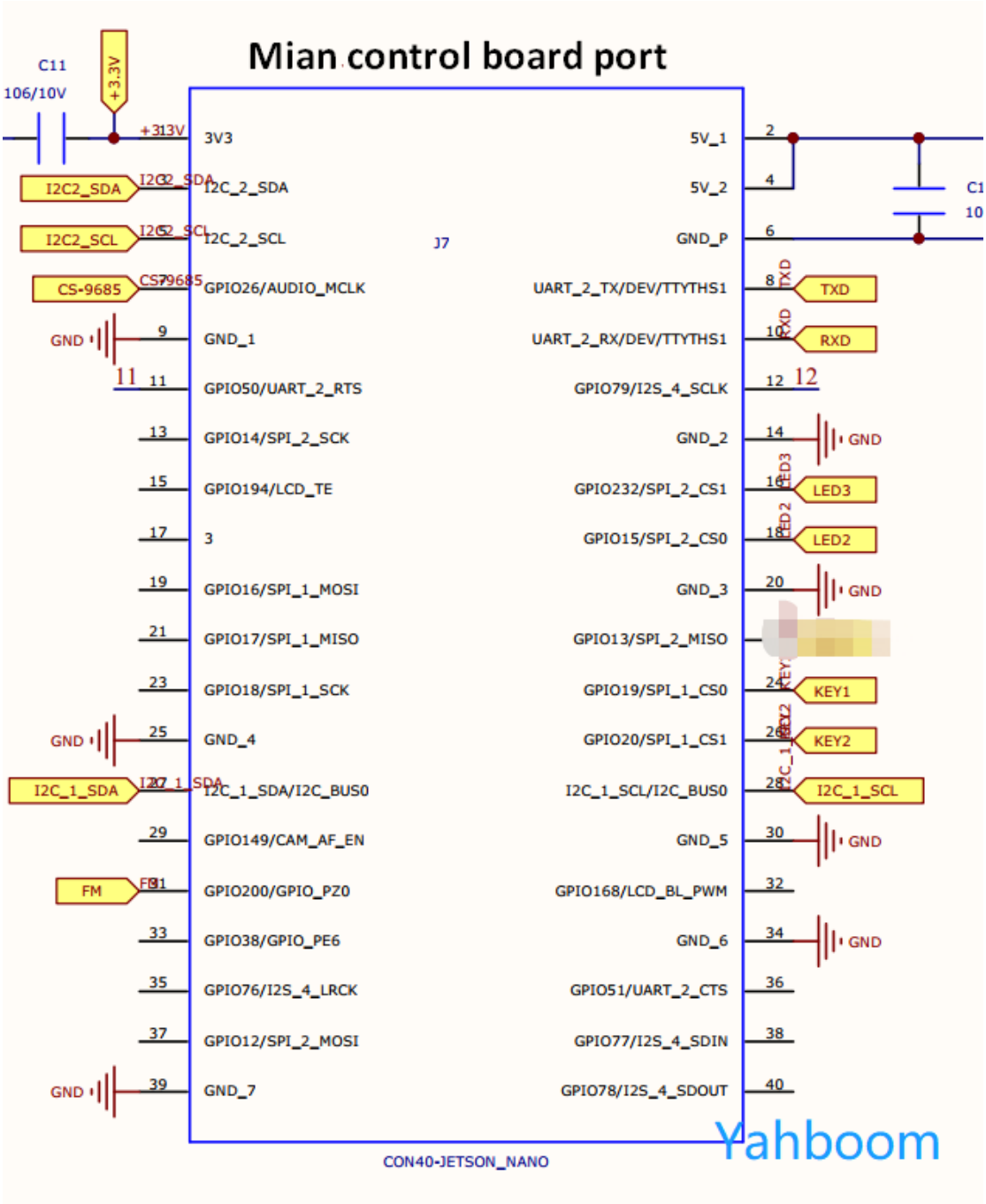
gpio38	**GPIO_PE6**	33	34	**GND**	
gpio76	**I2S_4_LRCK**	35	36	**UART_2_CTS**	gpio51
gpio12	**SPI_2_MOSI**	37	38	**I2S_4_SDIN**	gpio77
	GND	39	40	**I2S_4_SDOUT**	gpio78

But when we program, the BCM code number is generally used, as shown in the following figure:

BCM	Function	Physical pin		Function	BCM
	3V3	1	2	5V	
2	SDA	3	4	5V	
3	SCL	5	6	GND	
4	D4	7	8	D14(TXD)	14
	GND	9	10	D15(RXD)	15
17	D17	11	12	D18	18
27	D27	13	14	GND	
22	D22	15	16	D23	23
	3V3	17	18	D24	24
10	D10	19	20	GND	
9	D9	21	22	D25	25
11	D11	23	24	D8	8
	GND	25	26	D7	7
0	DO(ID_SD)	27	28	D1(ID_SC)	1
5	D5	29	30	GND	
6	D6	31	32	D12	12
13	D13	33	34	GND	
19	D19	35	36	D16	16
26	D26	37	38	D20	20
	GND	39	40	D21	21

The interface pin diagram used by the Jetbot robot car expansion board is shown below.

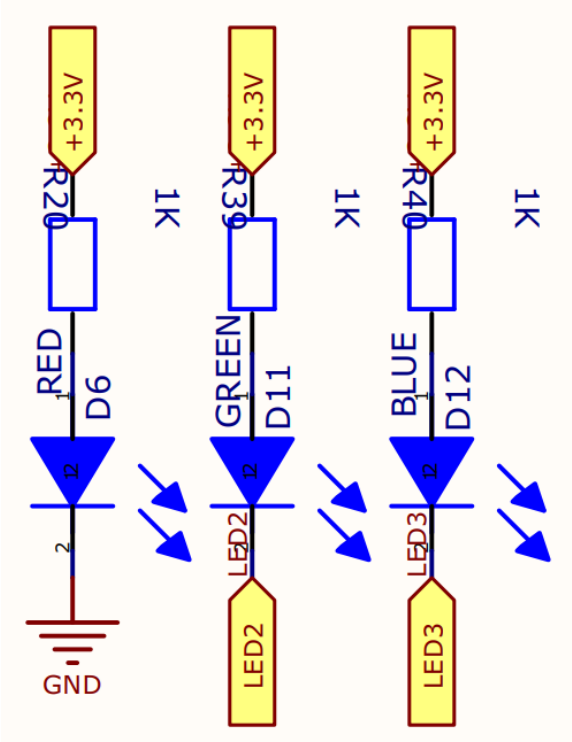
Number of BCM	Function	Corresponding peripheral
1、0	I2C1: SCL、SDA	OLED
3、2	I2C2: SCL、SDA	PCA9685+RGB strip
14	UART2: 2 TX	Serial port Servo
24、23	LED2、LED3	On board Light
6	FM	Buzzer
8、7	KEY1、KEY2	Button



!!!Note:

If you are not using the official Jetbot factory image provided by Yahboom, please confirm whether you can use the peripheral permissions of 【2.4 Install Jetbot】---【1. Enable peripheral permissions we will use】 before using GPIO. The operations in this section open the relevant usage rights.

2.Light up On board LED



Jetson nano's GPIO port structure and Raspberry Pi GPIO port driver can be shared. The name of the called GPIO driver in Jetson nano is called RPi.GPIO or Jetson.GPIO. When we call Jetsbot's GPIO port with RPi. Both GPIO and Jetson.GPIO can successfully call the GPIO driver.

Cycle flashing onboard green LED

Import GPIO objects from RPi.GPIO, initialize pin objects Set the GPIO port mode to BCM mode, set to output mode, and the initial level is high. Then every 0.25 enters the function to flip the GPIO port level to make the LED flash

```
import RPi.GPIO as GPIO
import time

# Define pins
LED2GREEN_pin = 24  # BOARD pin 12, BCM pin 18

def main():
    # Pin Setup:
    # Board pin-numbering scheme
    GPIO.setmode(GPIO.BCM)
    # Set pin to output mode, and set level is high
    GPIO.setup(LED2GREEN_pin, GPIO.OUT, initial=GPIO.HIGH)

    print("Starting demo now!")
    curr_value = GPIO.HIGH
    try:
        while True:
            # Flip the output every 0.25 seconds
            time.sleep(0.25)
            print("Outputting {} to pin {}".format(curr_value, LED2GREEN_pin))
            GPIO.output(LED2GREEN_pin, curr_value)
            curr_value ^= GPIO.HIGH
    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

We can see that the general steps for lighting the LEDs using the GPIO port are

①Set GPIO port mode

②Set GPIO output mode / input mode

③Set the initial level of the GPIO port
It can be seen from the schematic diagram that the GPIO port is turned on and the LED light is turned on. After executing the above code, we can see that the green onboard LED indicator will be displayed at a speed of 4 times per second and 2 times.
The corresponding complete source code is located:
/home/jetbot/Notebook/3.Using of GPIO /1.Lighting onboard LED
The code of the point 【Lighting onboard blue LED】 is the same except that the GPIO port of the driver is different.

3. Use of buttons

The use of the buttons on the Jetbot robot is different from the LED pins in that the GPIO is set to the input mode.

Button event detection

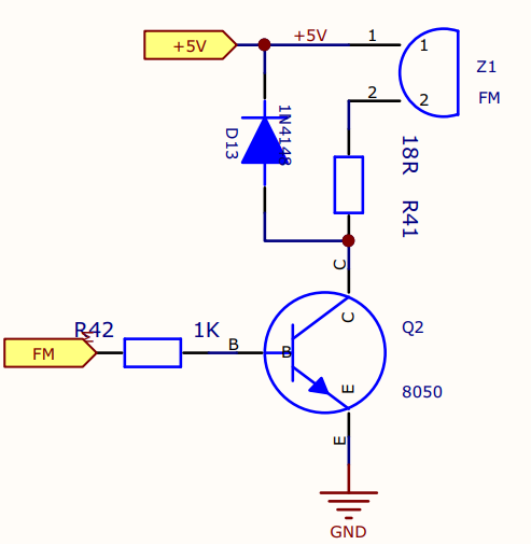
Import GPIO objects from RPi.GPIO, initialize pin objects Set the button GPIO port mode to BCM mode and set to input mode. When the state of button 1 changes, the green LED will follow the state change and print the current button GPIO port status.

```
import RPi.GPIO as GPIO
import time

# Pin Definitons:
led_pin = 24  # BOARD pin 24 GREEN
but_pin = 8  # BOARD pin 8   key1

def main():
    prev_value = None

    # Pin Setup:
    GPIO.setmode(GPIO.BCM)  # BCM
    GPIO.setup(led_pin, GPIO.OUT)  # LED pin set as output
    GPIO.setup(but_pin, GPIO.IN)  # Button pin set as input

    # Initial state for LEDs:
    GPIO.output(led_pin, GPIO.LOW)
    print("Starting demo now!")
    try:
        while True:
            curr_value = GPIO.input(but_pin)
            if curr_value != prev_value:
                GPIO.output(led_pin, not curr_value)
                prev_value = curr_value
                print("Outputting {} to Pin {}".format(curr_value, led_pin))
#             time.sleep(0.5)
    finally:
        GPIO.cleanup()  # cleanup all GPIO

if __name__ == '__main__':
    main()
```

We use the status of the onboard green LED to indicate the state of button 1. When button 1 is pressed, the green LED is illuminated. When button 1 is released, the green indicator is off. When a button event occurs. At the moment the button is pressed and the button is released, the current changed state is printed and displayed below the cell.
The corresponding complete source code is located：
/home/jetbot/Notebook/3.Using of GPIO/2.Using of buttons

The code of【Using of Button 2】is to indicate the state of the button 2 with the status of the onboard blue LED. When the button 2 is pressed, the blue LED indicator is illuminated, and when the pressed button 2 is released, The green indicator light is turned off. When the button event occurs, the current changed status will also be displayed below the cell when the button is pressed and the button is released.

4. Starting Buzzer



The edge buzzer circuit driven by the above-mentioned triode can know that the on-board indicator light is low level, and the level of the buzzer is turned on to turn on the triode, so that the buzzer sounds.

Code 1：

Turn on the buzzer

Turn the buzzer on or off every 1 second

```
import RPi.GPIO as GPIO
import time

# Pin Definitions
BEEP_pin = 6

def main():
    # Pin Setup:
    # Board pin-numbering scheme
    GPIO.setmode(GPIO.BCM)
    # set pin as an output pin with optional initial state of HIGH
    GPIO.setup(BEEP_pin, GPIO.OUT, initial=GPIO.HIGH)

    print("Starting demo now!")
    curr_value = GPIO.HIGH
    try:
        while True:
            time.sleep(1)
            # Toggle the output every second
            print("Outputting {} to pin {}".format(curr_value, BEEP_pin))
            GPIO.output(BEEP_pin, curr_value)
            curr_value ^= GPIO.HIGH
    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

The phenomenon after the above code is executed is to turn the buzzer on or off every second.

Code2:

Buzzer whistle three times

The buzzer GPIO port is flipped 6 times (Separate from the previous cell code, this is a separate code)

```
import RPi.GPIO as GPIO
import time

# Pin Definitions
BEEP_pin = 6

# Pin Setup:
# Board pin-numbering scheme
GPIO.setmode(GPIO.BCM)
# set pin as an output pin with optional initial state of HIGH
GPIO.setup(BEEP_pin, GPIO.OUT, initial=GPIO.LOW)

GPIO.output(BEEP_pin, GPIO.HIGH)
time.sleep(0.1)
GPIO.output(BEEP_pin, GPIO.LOW)
time.sleep(0.2)
GPIO.output(BEEP_pin, GPIO.HIGH)
time.sleep(0.1)
GPIO.output(BEEP_pin, GPIO.LOW)
time.sleep(0.2)
GPIO.output(BEEP_pin, GPIO.HIGH)
time.sleep(0.1)
GPIO.output(BEEP_pin, GPIO.LOW)
time.sleep(0.2)
```

The corresponding complete source code is located：
/home/jetbot/Notebook/3.Using of GPIO/3.Starting buzzer

After the above code is executed, the phenomenon is a fast whistle three times, that is, the prompt tone of the user after all the functions of the APP big program corresponding to our Jetbot are initialized.