

# **EE351**

---

12212635

---

1. EE351	3
1.1	3
2.	4
2.1	4
2.2	8
2.3	10
2.4 PCF8691	13
2.5	17
2.6 Lab6	20
2.7	23
2.8 PS2	26
2.9	28
2.10	30

# 1. EE351

---

EE351 "24Fall

4B RaspberryPi OS-64-bit-desktop

- 4B - PCF8591 - - LED - PS2 -

- RPi.GPIO Python - wiringPi C/C++ - python-smbus I2C

## 1.1

---

- 
- 
- 
- PCF8691
- 
- 
- 
- 
- PS2
- 
- 
- :

## 2.

---

### 2.1

---

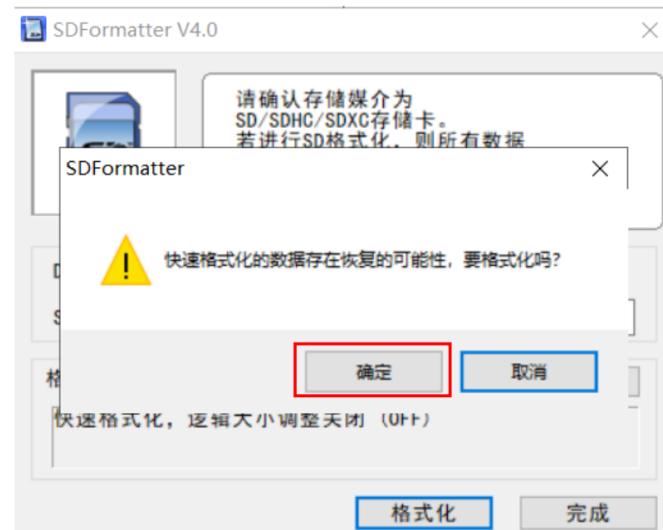
#### Lab1 Raspberry Pi

1. Raspberry Pi
2. Raspberry Pi OS
3. Wi-Fi

1

- 1.
2. Raspberry Pi 4 Model B
3. microSD
4. USB-C ,
5. HDMI HDMI
- 6.
- 7.
8. microSD Raspberry Pi

- SDFormatter



9. GUI
10. Raspberry Pi

2

1. **Raspberry Pi Imager**

2. Raspberry Pi Imager

## Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Ubuntu for x86](#)

[Download for Windows](#)

[Download for macOS](#)

To install on **Raspberry Pi OS**, type

```
sudo apt install rpi-imager
```

in a Terminal window.



3. OS

4. Raspberry Pi Imager "CHOOSE OS" Raspberry Pi OS (32-bit)
5. "CHOOSE STORAGE" microSD
6. "WRITE"
- 7.
8. microSD config.txt

```
1 max_usb_current=1
2 hdmi_force_hotplug=1
3 config_hdmi_boost=7
4 hdmi_group=2
5 hdmi_mode=1
6 hdmi_mode=87
7 hdmi_drive=1
8 display_rotate=0
9 hdmi_cvt 1024 600 60 0 0 0 0
```

```
> hdmi_cvt 1024 600 60 0 0 0 0
```

9. microSD Raspberry Pi
- 10.
11. OS microSD Raspberry Pi
- 12.

3

1. **Wi-Fi**
2. sudo raspi-config
3. "Network Options" Wi-Fi SSID
4. /etc/wpa\_supplicant/wpa\_supplicant.conf Wi-Fi
- 5.

6. ping www.bing.com

4

1.

2. sudo apt-get update

3.

4. sudo apt-get upgrade

5.

6. Python

```
1 sudo apt-get install python3-pip
2 pip3 install --upgrade pip
```

7. Git

```
1 sudo apt-get install git
```

8. Vim bash

```
sudo apt-get install vim
```

9.

10. SSH

```
1 sudo systemctl start ssh
```

11. SSH

```
1 sudo systemctl status ssh
```

12. SSH bash

```
sudo systemctl enable ssh
```

13. **vscode Raspberry Pi**

14. [Remote - SSH](#)

15. ifconfig IP

16. Ctrl+Shift+P Remote-SSH: Connect to Host Raspberry Pi IP

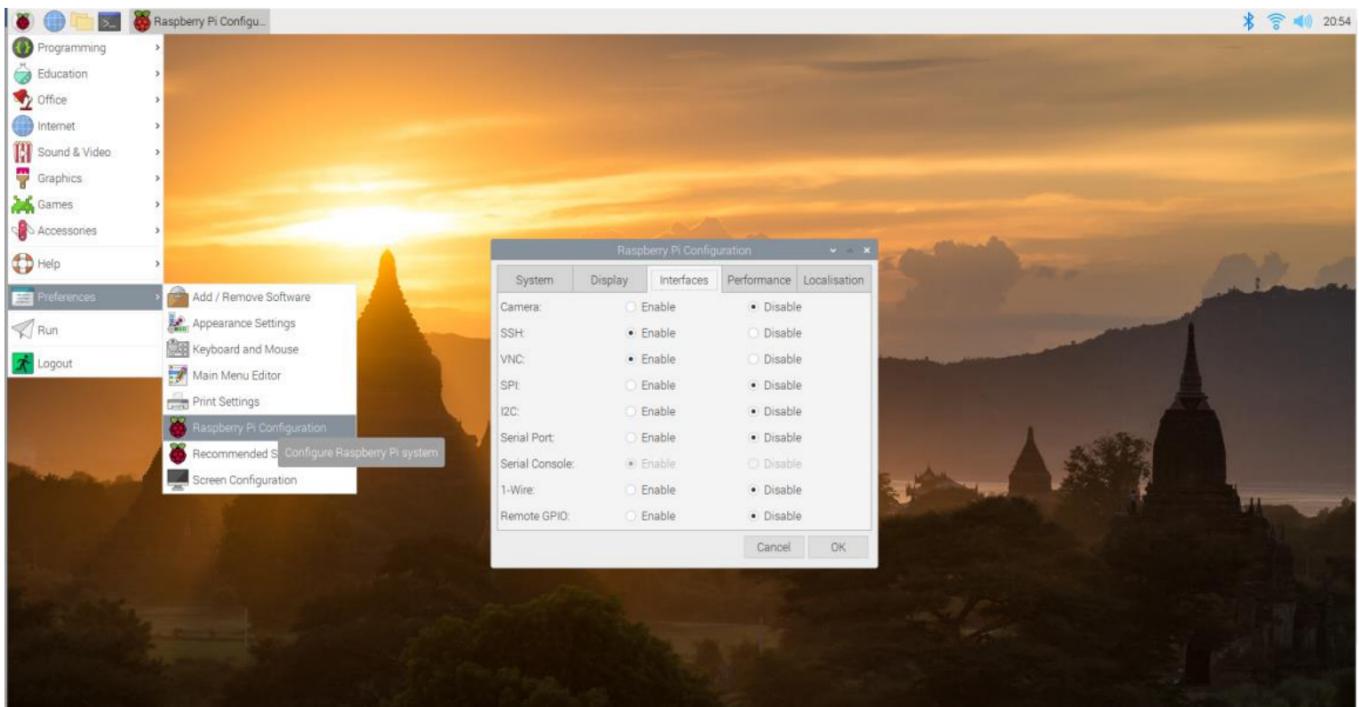
17. VSCode Raspberry Pi

18. ssh-rsa ~/.ssh/authorized\_keys

19. :

20. [VNC Viewer](#)

## 21. VNC Server



22. Raspberry

Raspberry Pi ip

23.

Raspberry Pi

Raspberry Pi

## 2.2

---

### Lab2 LED

Raspberry Pi IO

wiringPi RPi.GPIO

LED

#### 1. Raspberry Pi IO

2. Raspberry Pi 40 GPIO wiringPi BCM2837 SOC

3. BCM

#### 4. wiringPi

5. wiringPi C/C++ GPIO Raspberry Pi GPIO GPIO

#### 6. RPi.GPIO

7. RPi.GPIO Python Python Raspberry Pi GPIO Raspbian API

#### 8. Mu Geany IDE

9. Mu Python IDE

10. Geany IDE C/C++

11. IDE VSCode

#### 12. LED

13. LED LED

1.

2. LED S Raspberry Pi GPIO GND Raspberry Pi GPIO BCM GPIO19, GPIO20, GND

3.

4. Mu Python LED GPIO / LED

5. C/C++ Geany GPIO wiringPi

6.

7. Python C/C++ LED

8. T

9.

10. LED

## PYTHON

Python

LED

```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pins for the LED (BCM numbering)
5 RED_PIN = 19 # Red part of the dual-color LED
6 GREEN_PIN = 20 # Green part of the dual-color LED
7
8 # Setup GPIO mode and pin directions
9 GPIO.setmode(GPIO.BCM)
10 GPIO.setup(RED_PIN, GPIO.OUT)
11 GPIO.setup(GREEN_PIN, GPIO.OUT)
12
13 try:
14     while True:
15         # Turn on red LED
16         GPIO.output(RED_PIN, GPIO.HIGH)
17         GPIO.output(GREEN_PIN, GPIO.LOW)
18         print("Red LED is ON")
19         time.sleep(1) # Wait for 1 second
20
21         # Turn on green LED
22         GPIO.output(RED_PIN, GPIO.LOW)
23         GPIO.output(GREEN_PIN, GPIO.HIGH)
24         print("Green LED is ON")
25         time.sleep(1) # Wait for 1 second
26
27 except KeyboardInterrupt:
28     print("Program stopped by user")
29
30 finally:
31     # Clean up GPIO settings before exiting
32     GPIO.cleanup()

```

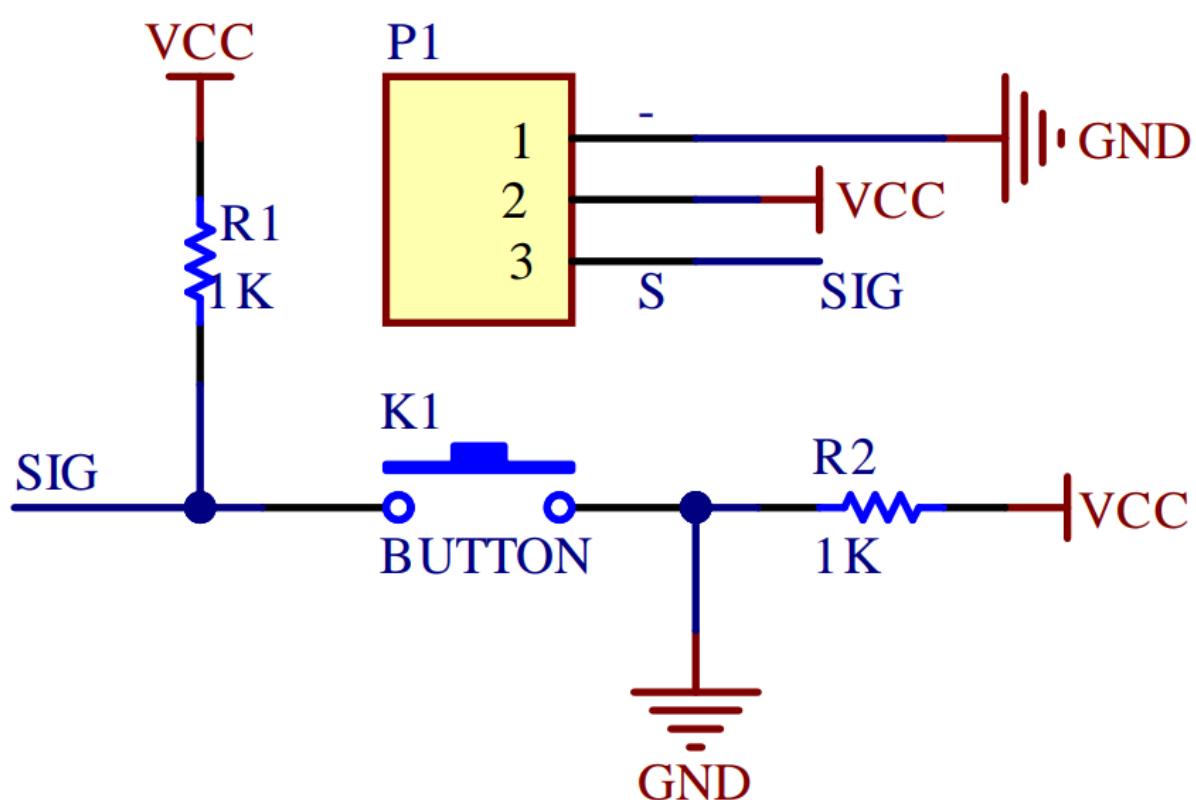
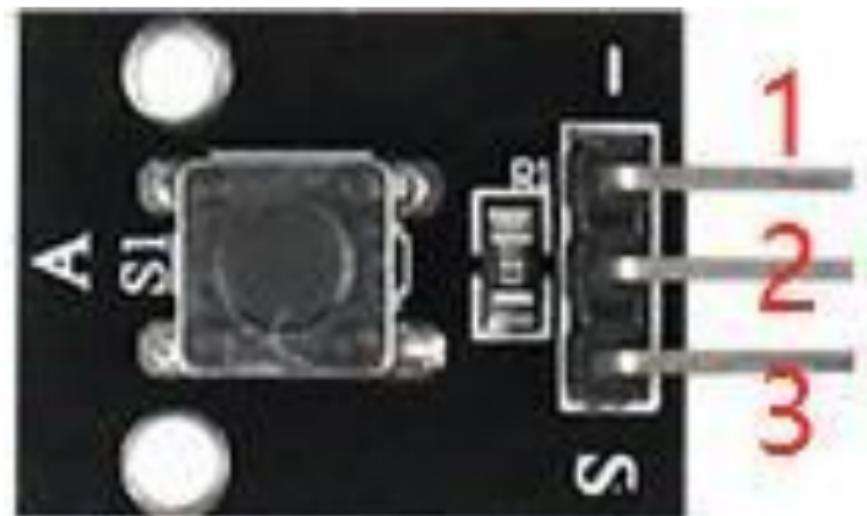
LED - - -

1

2.3

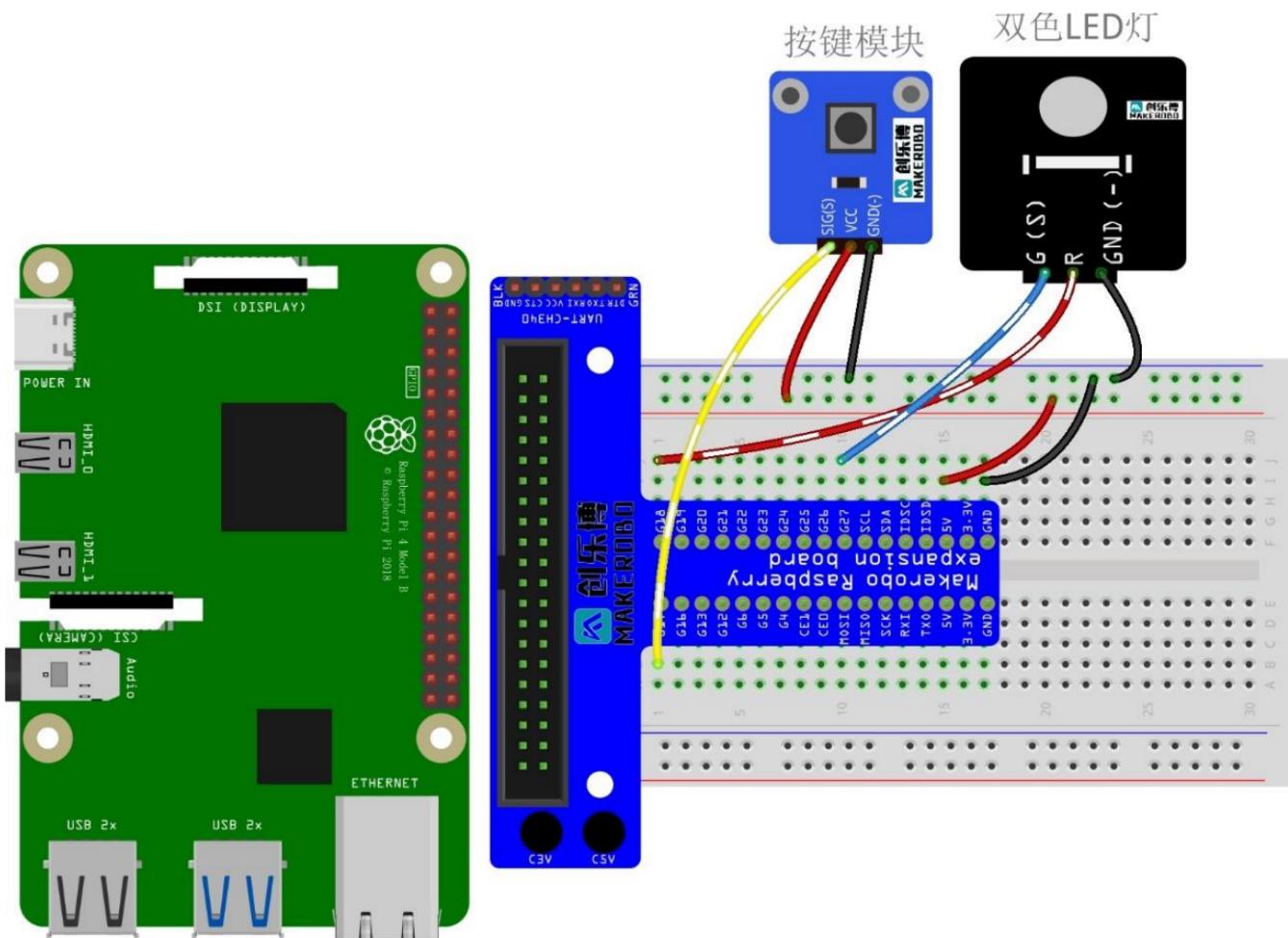
Lab3

- VCC S S



LED                    GPIO                    LED                    S                    S                    S                    LED  
 - Python    `time.sleep()`

1.



2. Mu      VSCode      Python      LED

3. GPIO      /

4. LED

5.

```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pins for the LED (BCM numbering)
5 RED_PIN = 19 # Red part of the dual-color LED
6 GREEN_PIN = 20 # Green part of the dual-color LED
7 SWITCH_PIN = 21 # GPIO pin for the tactile switch
8
9 # Setup GPIO mode and pin directions
10 GPIO.setmode(GPIO.BCM)
11 GPIO.setup(RED_PIN, GPIO.OUT)
12 GPIO.setup(GREEN_PIN, GPIO.OUT)
13 GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
14
15 def switch_with_delay(pin, delay=0.1):
16     state = GPIO.input(pin)
17     time.sleep(delay)
18     return state == GPIO.input(pin)
19
20 try:
21     while True:
22         if switch_with_delay(SWITCH_PIN):
23             GPIO.output(RED_PIN, GPIO.HIGH)
24             GPIO.output(GREEN_PIN, GPIO.LOW)
25         else:
26             GPIO.output(RED_PIN, GPIO.LOW)
27             GPIO.output(GREEN_PIN, GPIO.HIGH)
28
29 except KeyboardInterrupt:
30     print("Exiting...")
31 finally:
32     GPIO.cleanup()

```

1.

LED	1.	LED	2.	LED	3.	LED	4.	LED	.....
-----	----	-----	----	-----	----	-----	----	-----	-------

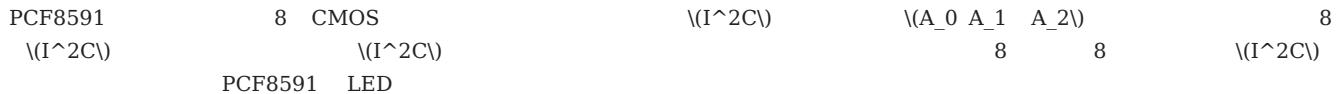
```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pins for the LED (BCM numbering)
5 RED_PIN = 19 # Red part of the dual-color LED
6 GREEN_PIN = 20 # Green part of the dual-color LED
7 SWITCH_PIN = 21 # GPIO pin for the tactile switch
8
9 # Setup GPIO mode and pin directions
10 GPIO.setmode(GPIO.BCM)
11 GPIO.setup(RED_PIN, GPIO.OUT)
12 GPIO.setup(GREEN_PIN, GPIO.OUT)
13 GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
14
15 def switch_with_delay(pin, delay=0.1):
16     state = GPIO.input(pin)
17     time.sleep(delay)
18     return state == GPIO.input(pin)
19
20 def toggle_led(pin):
21     GPIO.output(pin, not GPIO.input(pin))
22
23 try:
24     while True:
25         if switch_with_delay(SWITCH_PIN):
26             toggle_led(RED_PIN)
27             time.sleep(0.5)
28             toggle_led(RED_PIN)
29         else:
30             toggle_led(GREEN_PIN)
31             time.sleep(0.5)
32             toggle_led(GREEN_PIN)
33
34 except KeyboardInterrupt:
35     print("Exiting...")
36 finally:
37     GPIO.cleanup()

```

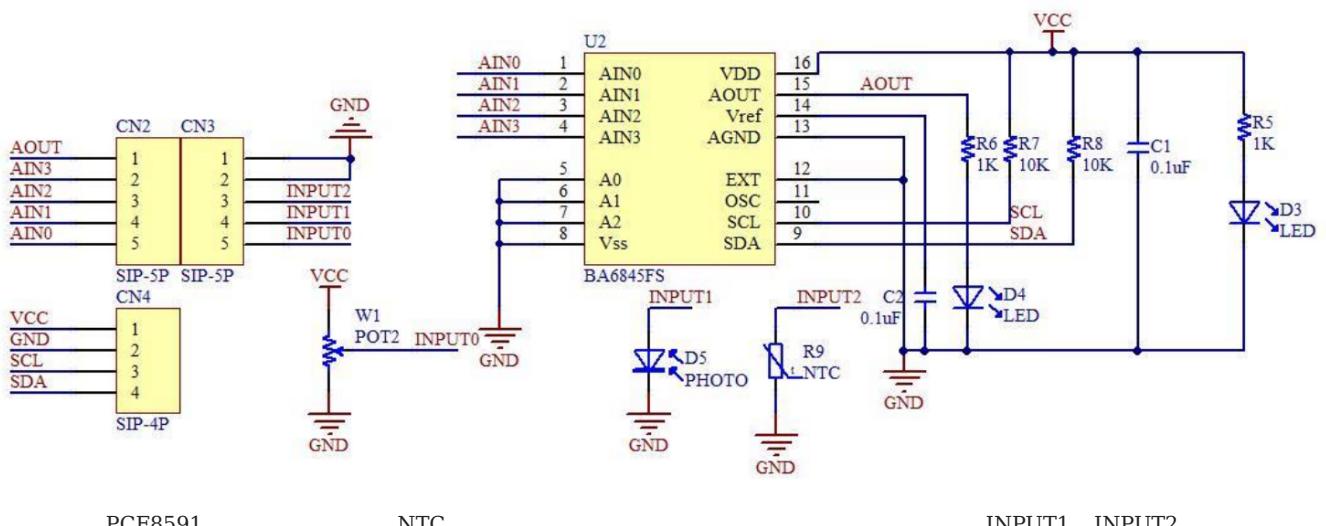
## 2.4 PCF8691

Lab5 PCF8591



### 1. PCF8591

2. PCF8591      CMOS      8 A/D      8 D/A
3. I2C      0x48      A0, A1, A2      8       $\setminus(I^2C)$
4. PCF8591      A/D
5. AINO( 0 )      AOUT( )      LED      LED



PCF8591

NTC

INPUT1    INPUT2

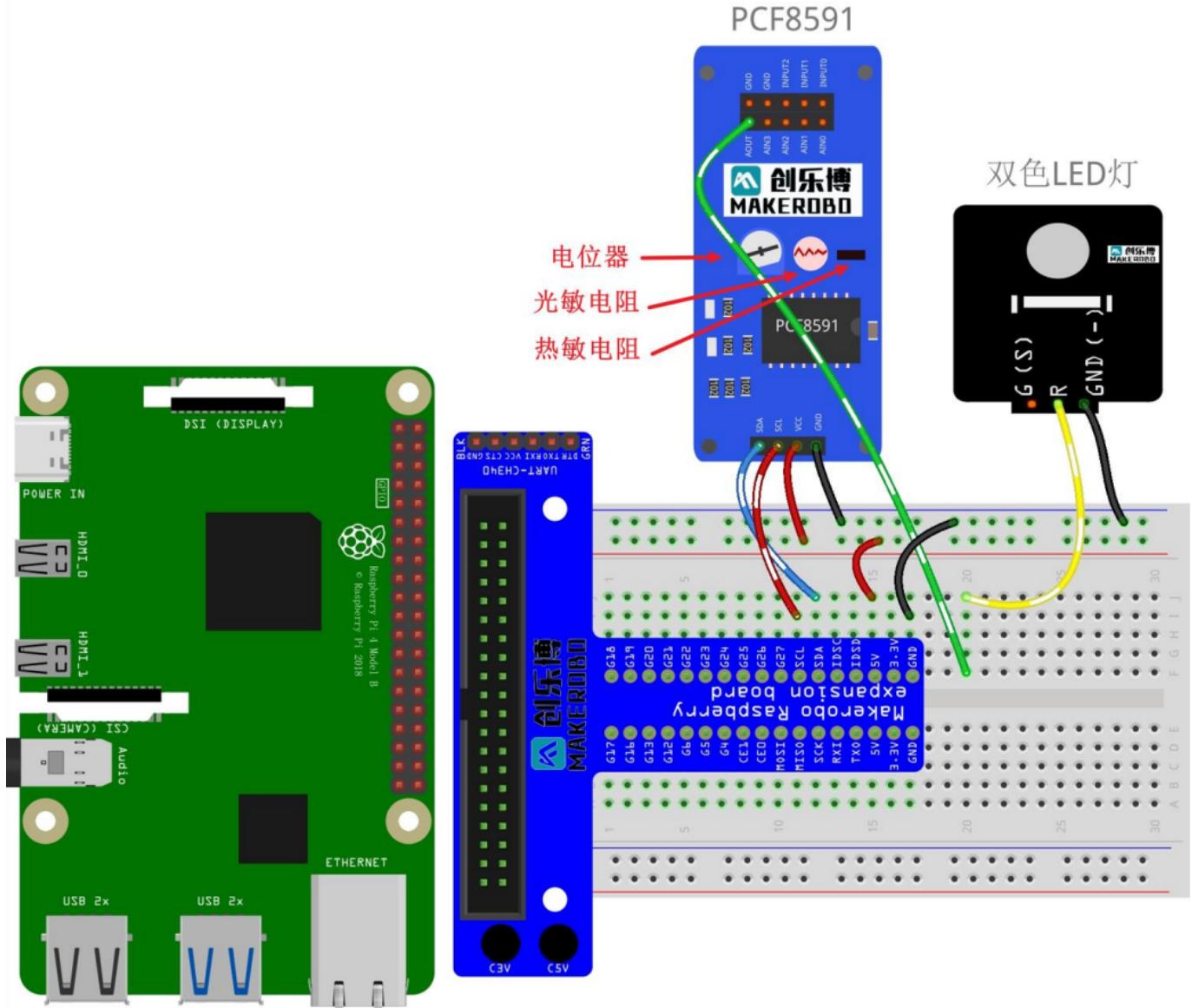
### 6. I2C

7.  $\setminus(I^2C)$       SDA      SCL      Raspberry Pi      PCF8591
- 8.
9. AIN0      AOUT      LED      LED
10. NTC

1.

2. Raspberry Pi T      PCF8591      SDA SCL VCC GND

3. LED                   PCF8591 AOUT    GND

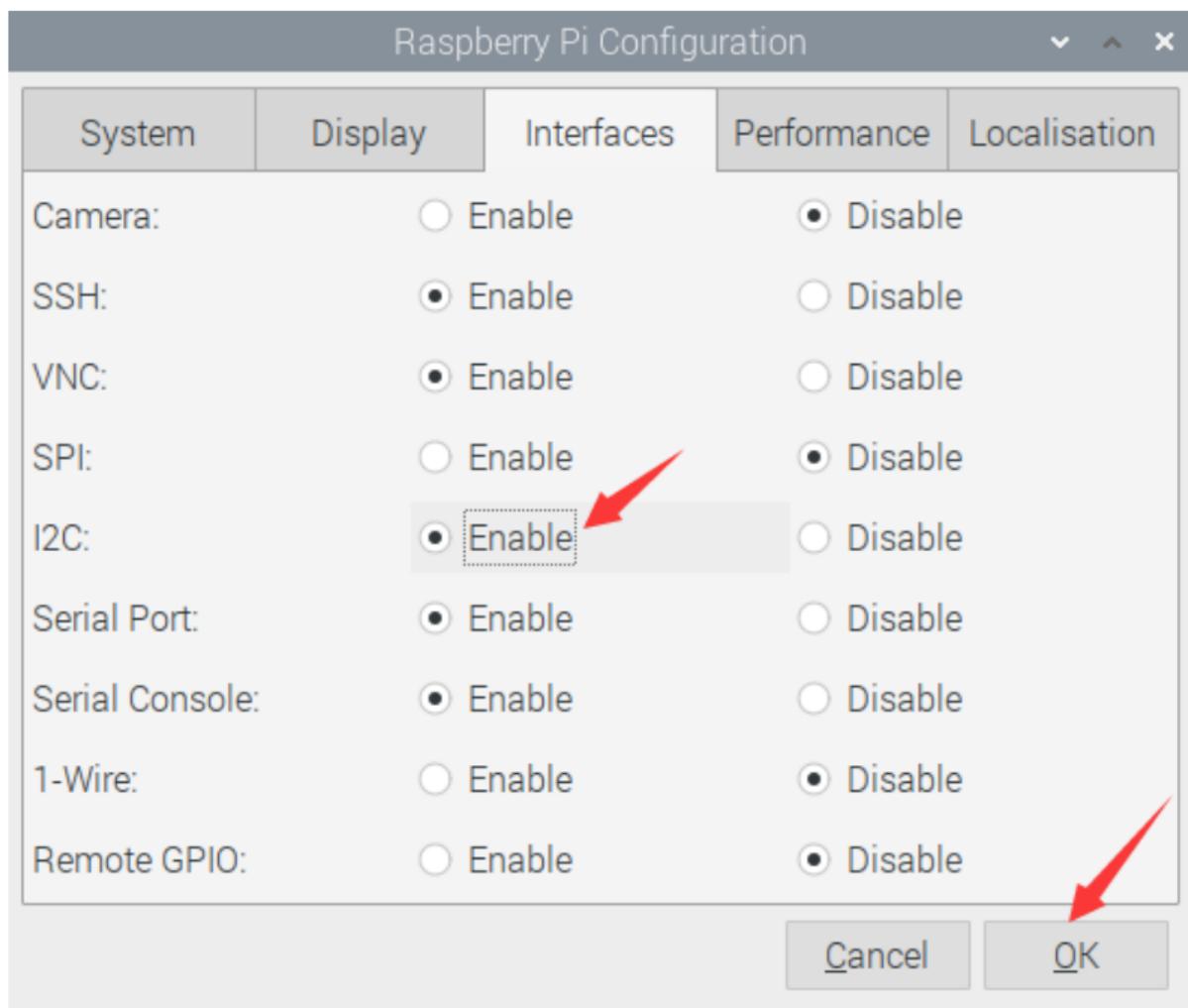


4. I2C

5. Raspberry Pi

Preferences -> Raspberry Pi Configuration

6. Interfaces I2C OK



7.

8. sudo i2cdetect -y 0 I2C

9. PCF8591 0x48

10.

11. Python smbus I2C

12.

SMBus PCF8591 AIN0 AOUT LED

```

1 import smbus
2 import time
3
4 # Define the I2C address of the PCF8591 and control bits
5 address = 0x48 # Default address for PCF8591
6 control_bit = 0x40 # Command to start conversion on channel 0 (AIN0)
7
8 # Initialize the SMBus library
9 bus = smbus.SMBus(1) # Use I2C bus 1
10
11 try:
12     while True:
13         # Write the control byte to initiate an A/D conversion on channel 0
14         bus.write_byte(address, control_bit)
15
16         # Read back the converted value from the PCF8591
17         analog_value = bus.read_byte(address)
18
19         # Print out the raw analog value
20         print("Analog Value:", analog_value)
21
22         # Map the analog value to a range suitable for controlling LED brightness
23         led_brightness = int((analog_value / 255.0) * 100)
24
25         print("LED Brightness (%):", led_brightness)
26
27         time.sleep(0.1) # Small delay between readings
28
29 except KeyboardInterrupt:
30     print("Exiting...")

```

## 2.5

## Lab5

NTC                          NTC  
 NTC                          ( )                  /

## 1. NTC

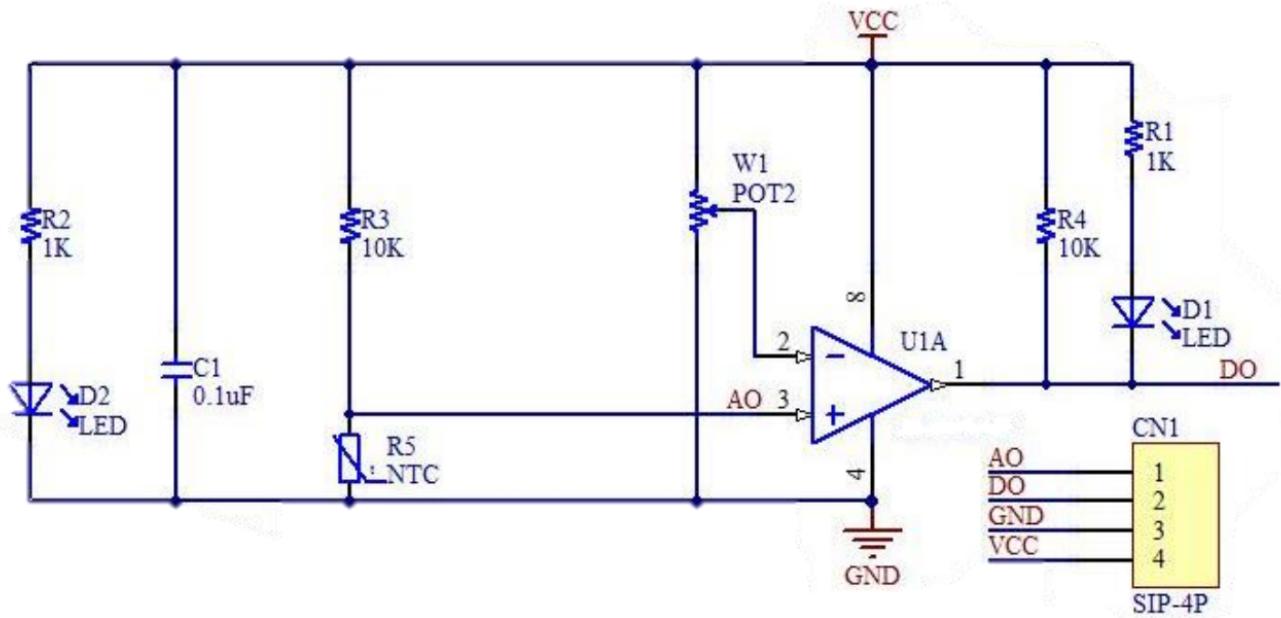
2.                          Steinhart-Hart

-                          Steinhart-Hart

3. Steinhart-Hart       $\frac{1}{R} = A + B \ln(T) + C(\ln(T))^3$        $\ln(R)$        $\ln(A), \ln(B), \ln(C)$   
 $R_0 = 10k\Omega$        $B = 3950K$

4.

5.                          NTC



6.                          PCF8591      AIN0

7.

8.                          PCF8591      A/D      5V ADC      8      0<sub>255</sub>      0<sup>5V</sup>

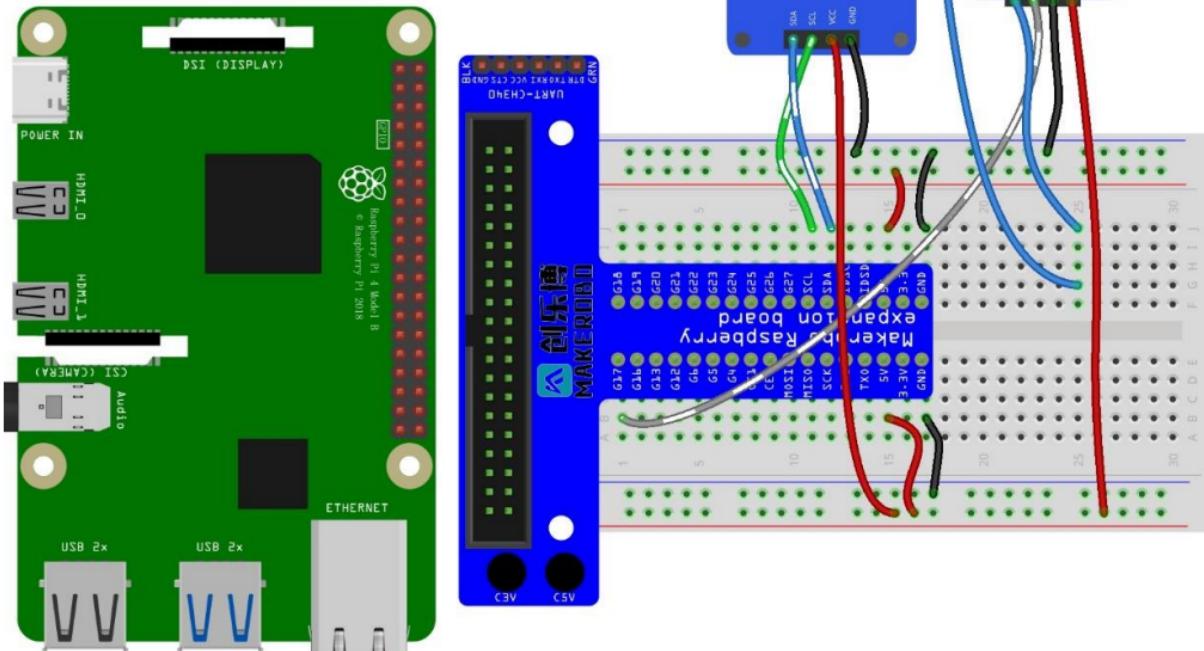
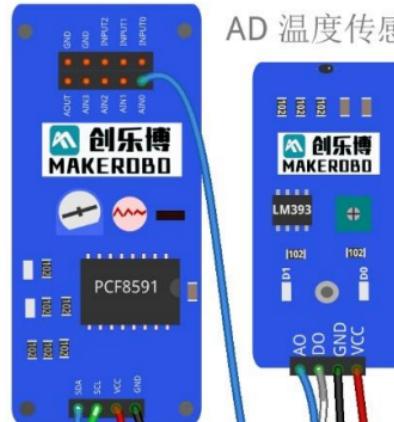
9.                          Steinhart-Hart      T

1.

2. Raspberry Pi T PCF8591 SDA SCL VCC GND

PCF8591

AD 温度传感器模块



3. AO PCF8591 AIN0 DO VCC 5V GND

4. I2C Lab4 2

5.

6.

## 7. SMBus PCF8591 AIN0

```

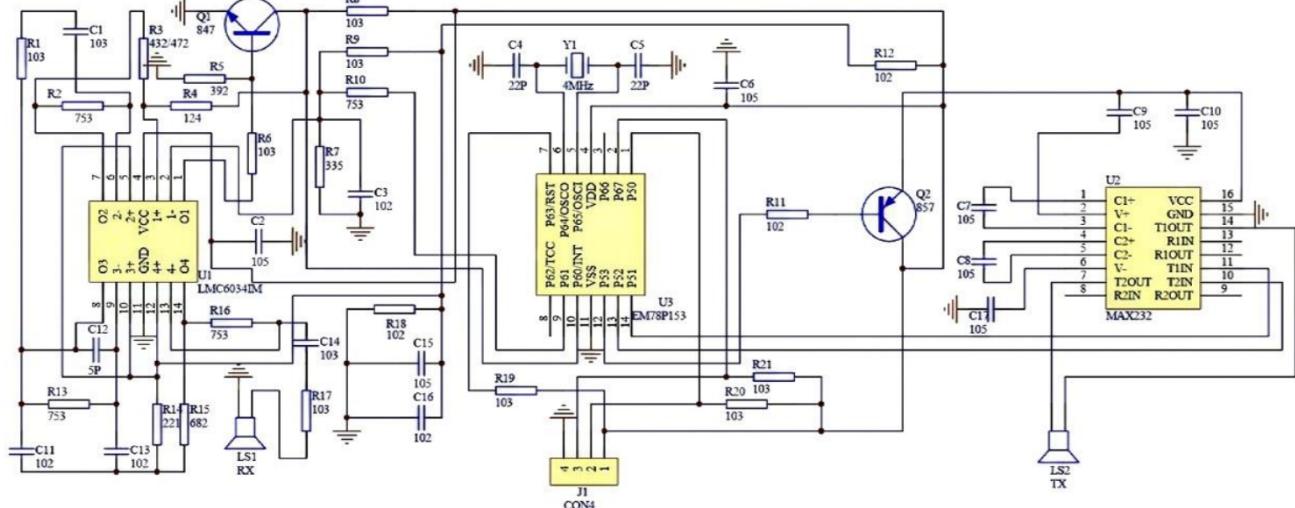
1 import smbus
2 import math
3 import time
4
5 # Define the I2C address of the PCF8591 and control bits
6 address = 0x48 # Default address for PCF8591
7 control_bit = 0x40 # Command to start conversion on channel 0 (AIN0)
8
9 # Constants for the thermistor calculation
10 R0 = 10000 # Resistance at 25°C in ohms
11 B = 3950 # Thermistor constant in Kelvin
12 T0 = 298.15 # Standard temperature in Kelvin (25°C)
13 Vcc = 5.0 # Supply voltage in volts
14
15 # Initialize the SMBus library
16 bus = smbus.SMBus(1) # Use I2C bus 1
17
18 def read_temperature():
19     try:
20         # Write the control byte to initiate an A/D conversion on channel 0
21         bus.write_byte(address, control_bit)
22
23         # Read back the converted value from the PCF8591
24         analog_value = bus.read_byte(address)
25
26         # Calculate the analog voltage
27         Vr = (analog_value / 255.0) * Vcc
28
29         # Calculate the resistance of the thermistor
30         Rt = R0 * Vr / (Vcc - Vr)
31
32         # Apply the Steinhart-Hart equation to calculate temperature
33         temp_kelvin = 1 / (math.log(Rt / R0) / B + 1 / T0)
34         temp_celsius = temp_kelvin - 273.15
35
36         return round(temp_celsius, 2)
37
38     except Exception as e:
39         print("Error reading temperature:", str(e))
40         return None
41
42     try:
43         while True:
44             temperature = read_temperature()
45             if temperature is not None:
46                 print(f"Temperature: {temperature}°C")
47             else:
48                 print("Failed to read temperature.")
49
50             time.sleep(1) # Small delay between readings
51
52     except KeyboardInterrupt:
53         pass # Allow the program to exit cleanly with Ctrl+C

```

## 2.6 Lab6

40KHz  
 2cm 400cm 3mm 5 Raspberry 5V 4 VCC 5V - Trig  
**- Echo** **-GND**

R3=392/432: 测试距离最大4M左右  
 R3=472: 测试距离最大7M左右

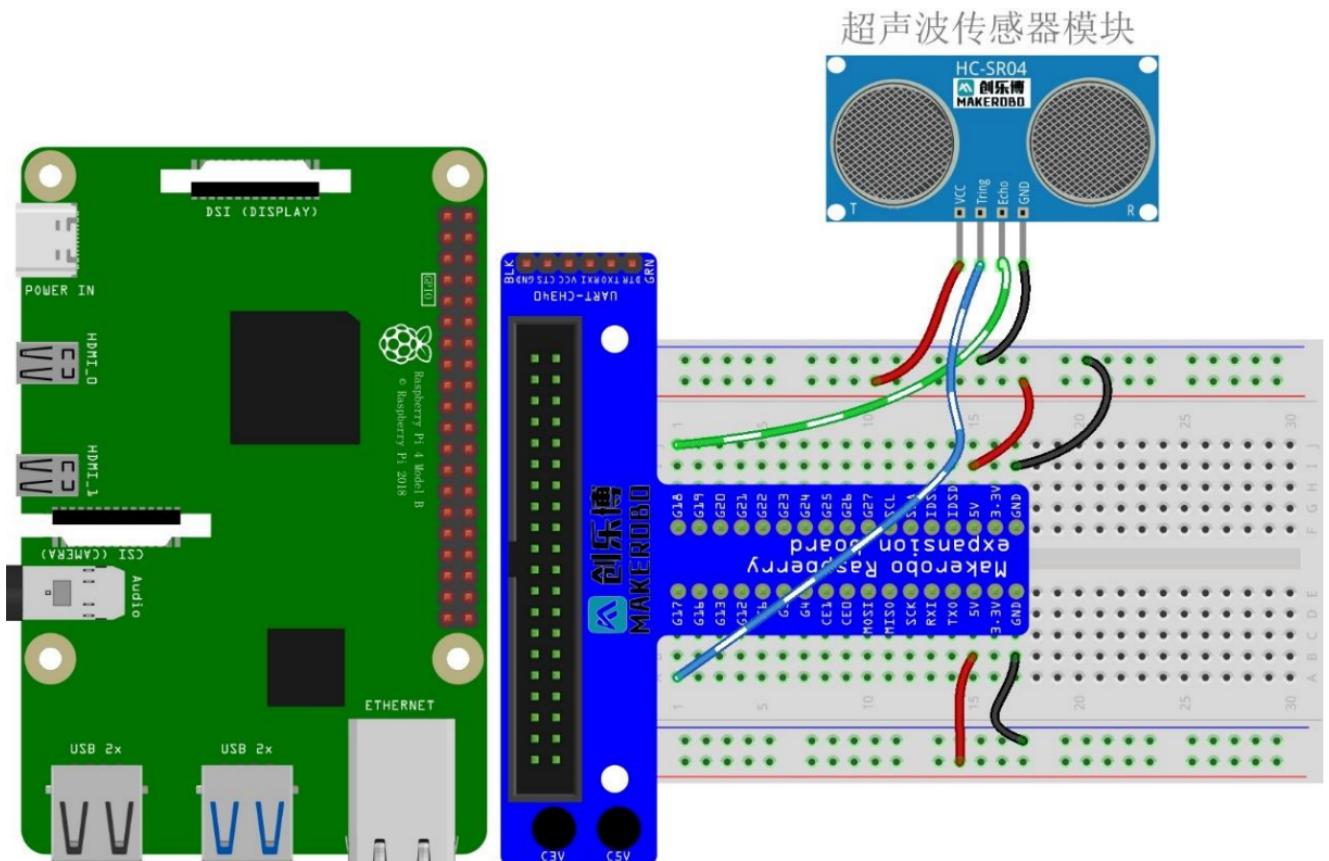


- 1.
2. Trig 10 8 40kHz
3. Echo Echo
4. 343 / 20 Echo
5. = Echo \* 34300 / 2
- 6.
7. Echo 5V GPIO 3.3V Echo

- 1.

2. Raspberry Pi T

VCC Trig Echo GND



3. Trig Raspberry Pi GPIO17 BCM Echo GPIO18 BCM VCC 5V GND

4.

5. RPi.GPIO Trig Echo BCM

6.	get_distance()	Trig	10	Echo	Echo
----	----------------	------	----	------	------

```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pins for the ultrasonic sensor
5 TRIG = 17 # BCM numbering
6 ECHO = 18 # BCM numbering
7
8 # Setup GPIO mode and pin directions
9 GPIO.setmode(GPIO.BCM)
10 GPIO.setup(TRIG, GPIO.OUT)
11 GPIO.setup(ECHO, GPIO.IN)
12
13 def get_distance():
14     # Ensure TRIG is low initially
15     GPIO.output(TRIG, False)
16     time.sleep(0.2)
17
18     # Send a 10us pulse to TRIG
19     GPIO.output(TRIG, True)
20     time.sleep(0.0001)
21     GPIO.output(TRIG, False)
22
23     # Wait for ECHO to go high
24     while GPIO.input(ECHO) == 0:
25         pulse_start = time.time()
26
27     # Wait for ECHO to go low again
28     while GPIO.input(ECHO) == 1:
29         pulse_end = time.time()
30
31     # Calculate the duration of the pulse
32     pulse_duration = pulse_end - pulse_start
33
34     # Convert pulse duration to distance in centimeters
35     distance = pulse_duration * 17150 # Speed of sound in cm/s divided by 2 (round trip)
36     distance = round(distance, 2)
37
38     return distance
39
40 try:
41     print("Measuring distance...")
42     while True:
43         dist = get_distance()
44         print(f"Distance: {dist} cm")
45         time.sleep(1)
46
47 except KeyboardInterrupt:
48     print("Measurement stopped by user")
49
50 finally:
51     GPIO.cleanup() # Clean up GPIO settings before exiting

```

## 2.7

---

### Lab7

Pulse-Frequency Modulation PFM

- 1.
- 2.
3.                  GPIO
- 4.
- 5.
6. PFM Pulse-Frequency Modulation                  /
7. PWM Pulse-Width Modulation
- 8.
9.                  GPIO
10.                 PWM

1

- 1.
2.                Raspberry Pi T                  I/O VCC GND
3.                3.3V
- 4.
5. Python                  RPi.GPIO        GPIO
6. play\_tone()                  GPIO

7.

```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pin for the buzzer (BCM numbering)
5 BUZZER_PIN = 17 # BCM 17, physical pin 11
6
7 # Setup GPIO mode and pin direction
8 GPIO.setmode(GPIO.BCM)
9 GPIO.setup(BUZZER_PIN, GPIO.OUT)
10
11 def play_tone(duration=0.5):
12     """Play a tone using the active buzzer."""
13     try:
14         # Turn on the buzzer (low level trigger)
15         GPIO.output(BUZZER_PIN, GPIO.LOW)
16         time.sleep(duration)
17
18         # Turn off the buzzer
19         GPIO.output(BUZZER_PIN, GPIO.HIGH)
20         time.sleep(0.1) # Short pause between tones
21
22     except KeyboardInterrupt:
23         print("Stopped by user")
24
25 finally:
26     GPIO.cleanup() # Clean up GPIO settings before exiting
27
28 if __name__ == "__main__":
29     print("Playing tone...")
30     while True:
31         play_tone()

```

2

1.

2.                   Raspberry Pi T	I/O VCC GND
3.        PWM    GPIO    GPIO18    BCM	
4.	
5.    Python           RPi.GPIO    pigpio            PWM	
6.       play_music()	

7.

```

1 import RPi.GPIO as GPIO
2 import pigpio
3 import time
4
5 # Define GPIO pin for the passive buzzer (BCM numbering)
6 BUZZER_PIN = 18 # BCM 18, physical pin 12
7
8 # Initialize pigpio library
9 pi = pigpio.pi()
10
11 # Notes and their frequencies in Hz
12 NOTES = {
13     'C4': 262, 'D4': 294, 'E4': 330, 'F4': 349, 'G4': 392, 'A4': 440, 'B4': 494,
14     'C5': 523, 'D5': 587, 'E5': 659, 'F5': 698, 'G5': 784, 'A5': 880, 'B5': 988,
15 }
16
17 # A simple melody to play
18 MELODY = ['C4', 'D4', 'E4', 'C4', 'E4', 'D4', 'C4']
19
20 # Function to set frequency of the passive buzzer
21 def set_frequency(freq):
22     pi.hardware_PWM(BUZZER_PIN, freq, 500000) # Frequency, Duty cycle (50%)
23
24 def play_music(melody):
25     try:
26         for note in melody:
27             if note in NOTES:
28                 set_frequency(NOTES[note])
29                 time.sleep(0.5) # Duration of each note
30                 set_frequency(0) # Stop sound between notes
31                 time.sleep(0.1) # Short pause between notes
32
33     except KeyboardInterrupt:
34         print("Music stopped by user")
35
36 finally:
37     pi.stop() # Clean up pigpio resources
38     GPIO.cleanup() # Clean up GPIO settings before exiting
39
40 if __name__ == "__main__":
41     print("Playing music...")
42     play_music(MELODY)

```

1.

2. Python

3.

4.

5.

## 2.8 PS2

---

**Lab8 PS2**

PS2	Raspberry Pi	LED	PS2	X Y	Z
	PCF8591	LED			

### 1. PS2

2. PS2 X Y 0V 5V 2.5V
3. SW 0V
- 4.
5. PS2 X VRX Y VRY PCF8591 AIN0 AIN1 SW GPIO
6. PCF8591 Raspberry Pi
- 7.
8. PCF8591 X Y
9. PCF8591 AOUT LED LED LED

1.

2. Raspberry Pi T PCF8591 PS2 SDA SCL VCC GND VRX VRY SW
3. PS2 VRX PCF8591 AIN0 VRY AIN1 SW GPIO VCC 5V GND

### 4. I2C

5. Raspberry Pi Preferences -> Raspberry Pi Configuration
6. Interfaces I2C OK
- 7.
8. Python smbus I2C
9. SMBus PCF8591 AIN0 AIN1 LED

10.

```

1 import smbus
2 import time
3
4 # Define the I2C address of the PCF8591 and control bits
5 address = 0x48 # Default address for PCF8591
6 control_bit_x = 0x40 # Command to start conversion on channel 0 (AIN0, X-axis)
7 control_bit_y = 0x41 # Command to start conversion on channel 1 (AIN1, Y-axis)
8
9 # Initialize the SMBus library
10 bus = smbus.SMBus(1) # Use I2C bus 1
11
12 def read_joystick(axis='x'):
13     """Read joystick position from specified axis."""
14     if axis.lower() == 'x':
15         control_bit = control_bit_x
16     elif axis.lower() == 'y':
17         control_bit = control_bit_y
18     else:
19         raise ValueError("Invalid axis. Choose 'x' or 'y'.")
20
21 try:
22     # Write the control byte to initiate an A/D conversion on selected channel
23     bus.write_byte(address, control_bit)
24
25     # Read back the converted value from the PCF8591
26     analog_value = bus.read_byte(address)
27
28     return analog_value
29
30 except Exception as e:
31     print(f"Error reading {axis}-axis: {str(e)}")
32     return None
33
34 def map_to_brightness(value, in_min=0, in_max=255, out_min=0, out_max=100):
35     """Map joystick value to LED brightness percentage."""
36     return int((value - in_min) * (out_max - out_min) / (in_max - in_min) + out_min)
37
38 try:
39     while True:
40         x_value = read_joystick('x')
41         y_value = read_joystick('y')
42
43         if x_value is not None and y_value is not None:
44             print(f"X-axis: {x_value}, Y-axis: {y_value}")
45
46             # Calculate LED brightness based on joystick position
47             led_brightness_x = map_to_brightness(x_value)
48             led_brightness_y = map_to_brightness(y_value)
49
50             # Here you would add code to set the LED brightness using PWM or similar method.
51             # For demonstration purposes, we'll just print the calculated brightness.
52             print(f"LED Brightness X (%): {led_brightness_x}, Y (%): {led_brightness_y}")
53
54         time.sleep(0.1) # Small delay between readings
55
56 except KeyboardInterrupt:
57     pass # Allow the program to exit cleanly with Ctrl+C

```

1.

2. Python LED

3.

4.

5.

## 2.9

---

### Lab9

LIRC    Raspberry Pi  
Raspberry Pi

- 1.
2.         LED    PIN
- 3.
4.    PIN
- 5.

### 6. LIRC

7. LIRC Linux Infrared Remote Control    LIRC

#### 1. LIRC

2.         LIRC

```
1 sudo apt-get update
2 sudo apt-get install lirc
```

3. /boot/config.txt    GPIO                  22              23

```
1 dtoverlay= gpio-ir, gpio_pin=22
2 dtoverlay= gpio-ir-tx, gpio_pin=23
```

- 4.
5. /etc/lirc/lirc\_options.conf    LIRC

```
1 sudo nano /etc/lirc/lirc_options.conf
```

```
1 driver = default
2 device = /dev/lirc0
```

- 6.
7.         Raspberry Pi

```
1 sudo reboot
```

#### 8. IR

9.         irw

```
1 irw
```

- 10.
- 11.
12.         Python    LIRC

## 13. Python

```
1 import subprocess
2
3 def listen_to_remote():
4     try:
5         process = subprocess.Popen(['irw'], stdout=subprocess.PIPE)
6
7         while True:
8             line = process.stdout.readline().decode('utf-8').strip()
9             if not line:
10                 break
11
12             print("Received IR command:", line)
13
14     except KeyboardInterrupt:
15         print("\nListening stopped.")
16
17 if __name__ == "__main__":
18     print("Listening for IR commands...")
19     listen_to_remote()
```

1.

2. Python

3.

4.

5.

## 2.10

---

### Lab10

Raspberry Pi                                  GPIO  
LED

1.

2. `GPIO.add_event_detect()`              GPIO

- `channel`              GPIO
- `edge`              GPIO.RISING      GPIO.FALLING      GPIO.BOTH
- `callback`
- `bouncetime`

3.

4.            `GPIO.wait_for_edge()`              CPU

5.

6.            "      "

1.

2.            Raspberry Pi T              SIG(S) VCC GND

3.            SIG(S)      Raspberry Pi    GPIO23    BCM      VCC    5V    GND

4.            LED              LED              GPIO17      LED      GPIO27

5.

6. Python              RPi.GPIO      GPIO

7.            `setup_gpio()`      GPIO

8.            `button_pressed_callback()`      LED

9.

```

1 import RPi.GPIO as GPIO
2 import time
3
4 # Define GPIO pins for the LED and button (BCM numbering)
5 RED_LED_PIN = 17 # BCM 17, physical pin 11
6 GREEN_LED_PIN = 27 # BCM 27, physical pin 13
7 BUTTON_PIN = 23 # BCM 23, physical pin 16
8
9 def setup_gpio():
10     """Setup GPIO mode and pin directions."""
11     GPIO.setmode(GPIO.BCM)
12
13     # Setup LEDs as output
14     GPIO.setup(RED_LED_PIN, GPIO.OUT)
15     GPIO.setup(GREEN_LED_PIN, GPIO.OUT)
16
17     # Setup button as input with pull-up resistor
18     GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
19
20 def button_pressed_callback(channel):
21     """Callback function called when the button is pressed."""
22     if channel == BUTTON_PIN:
23         print("Button pressed!")
24
25     # Toggle between red and green LED
26     if GPIO.input(RED_LED_PIN):
27         GPIO.output(RED_LED_PIN, GPIO.LOW)
28         GPIO.output(GREEN_LED_PIN, GPIO.HIGH)
29     else:
30         GPIO.output(RED_LED_PIN, GPIO.HIGH)
31         GPIO.output(GREEN_LED_PIN, GPIO.LOW)
32
33 try:
34     setup_gpio()
35
36     # Add event detection on the button pin with debouncing
37     GPIO.add_event_detect(BUTTON_PIN, GPIO.FALLING, callback=button_pressed_callback, bouncetime=200)
38
39     print("Waiting for button press...")
40     while True:
41         time.sleep(1) # Keep script running to allow callbacks to work
42
43 except KeyboardInterrupt:
44     print("\nProgram stopped by user")
45
46 finally:
47     GPIO.cleanup() # Clean up GPIO settings before exiting

```

1.

2. Python

LED

3. bouncetime

4.

5. ...

6.

7.

8.