== Lec2

`lighten_led.py`

```
import RPi.GPIO as GPIO
from time import sleep

# GPIO.cleanup()

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)


if __name__ == '__main__':
    try:
        while True:
            print('Start')
            GPIO.output(17, GPIO.HIGH)
            print('R High')
            sleep(0.1)
            GPIO.output(17, GPIO.LOW)
            print('R Low')
            sleep(0.1)
            GPIO.output(27, GPIO.HIGH)
            print('G High')
            sleep(0.1)
            GPIO.output(27, GPIO.LOW)
            print('G Low')
            sleep(0.1)
            print('End')
    except KeyboardInterrupt:
        GPIO.cleanup()
```

== Lec3

`switch_led.py`

```
import RPi.GPIO as GPIO
from time import sleep
from threading import Thread, Lock

GPIO.setmode(GPIO.BCM)
GPIO.setup(26, GPIO.OUT)
GPIO.setup(19, GPIO.OUT)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # 使用上拉电阻

now_state = 0
lock = Lock()


press_duration = 0.05
presses_interval = 0.5
def button_pressed():
    global now_state
```

```
    while True:
        if GPIO.input(17) == GPIO.HIGH:
            sleep(press_duration) # the press shall be long enough: 0.05s
            if GPIO.input(17) == GPIO.HIGH:
                with lock:
                    now_state = (now_state + 1) % 5
                    print('Button pressed! State:', now_state)
                sleep(presses_interval) # the presses shall be far enough: 0.5s
        sleep(0.05) # reduce usage


def led_control():
    global now_state
    while True:
        with lock:
            private_now_state = now_state # Lock stops here. Avoid locking for
too long.
        if private_now_state == 0:
            GPIO.output(26, GPIO.LOW)
            GPIO.output(19, GPIO.LOW)
            sleep(0.5)

        elif private_now_state == 1:
            GPIO.output(26, GPIO.HIGH)
            sleep(0.5)

        elif private_now_state == 2:
            for _ in range(5):
                GPIO.output(26, GPIO.HIGH)
                sleep(0.05)
                GPIO.output(26, GPIO.LOW)
                sleep(0.05)

        elif private_now_state == 3:
            GPIO.output(19, GPIO.HIGH)
            sleep(0.5)

        elif private_now_state == 4:
            for _ in range(5):
                GPIO.output(19, GPIO.HIGH)
                sleep(0.05)
                GPIO.output(19, GPIO.LOW)
                sleep(0.05)


thread1 = Thread(target=button_pressed)
thread2 = Thread(target=led_control)

try:
    thread1.start()
    thread2.start()

    thread1.join()
    thread2.join()

except KeyboardInterrupt:
```

```
    GPIO.cleanup()
    print('\nProgram interrupted and GPIO cleaned up.')
```

== Lec4

`1_keyboard_control_brightness.py`

```
import smbus
import time

bus = smbus.SMBus(1)
address = 0x48  # sudo i2cdetect -y 1


def set_led_brightness(brightness):
    try:
        bus.write_byte_data(address, 0x40, brightness)
    except OSError as e:
        print(f"Error setting LED brightness: {e}")


try:
    while True:
        for i in range(256):
            set_led_brightness(i)
            time.sleep(0.01)
        for i in range(255, -1, -1):
            set_led_brightness(i)
            time.sleep(0.01)

except KeyboardInterrupt:
    set_led_brightness(0)
    print("\nTerminated")
```

`2_print_now_voltage.py`

```
import smbus2 as smbus
import time

bus = smbus.SMBus(1)
address = 0b01001000


# 读取模拟输入值
def read_analog_input():
    bus.write_byte(address, 0b00000000)  # 选择 AIN0 通道
    analog_value = bus.read_byte(address)
    return analog_value


# 计算电压
def calculate_voltage(value):
    voltage = value / 255 * 5
    return voltage
```

```
# 打印电压值
def print_voltage():
    analog_value = read_analog_input()
    voltage = calculate_voltage(analog_value)
    print(f"Voltage: {voltage:.2f} V")


# 主程序
if __name__ == "__main__":
    try:
        while True:
            print_voltage()
            time.sleep(0.2)
    except KeyboardInterrupt:
        print("\nTerminated")
```

`3_breathe_control_brightness.py`

```
import smbus
import time

bus = smbus.SMBus(1)
address = 0x48  # sudo i2cdetect -y 1


def set_led_brightness(brightness):
    try:
        bus.write_byte_data(address, 0x40, brightness)
    except OSError as e:
        print(f"Error setting LED brightness: {e}")


try:
    while True:
        brightness = int(input("Input brightness (0-255): "))
        if 0 <= brightness <= 255:
            set_led_brightness(brightness)
except KeyboardInterrupt:
    set_led_brightness(0)
    print("\nTerminated")
```

`4_potentiometer_control_brightness.py`

```
import smbus
import time

bus = smbus.SMBus(1)
address = 0x48


def set_led_brightness(brightness):
```

```
    try:
        bus.write_byte_data(address, 0x40, brightness)
    except OSError as e:
        print(f"Error setting LED brightness: {e}")


# 读取模拟输入值
def read_analog_input():
    bus.write_byte(address, 0x40)  # 选择 AIN0 通道
    analog_value = bus.read_byte(address)
    return analog_value


if __name__ == "__main__":
    try:
        while True:
            analog_value = read_analog_input()
            set_led_brightness(analog_value)
            time.sleep(0.01)
    except KeyboardInterrupt:
        set_led_brightness(0)
        print("\nTerminated")
```

S_plot_wave.py

```
import smbus2 as smbus
import time
import matplotlib.pyplot as plt
import numpy as np

bus = smbus.SMBus(1)
address = 0b01001000
in_channel = 0b00000000
bus.write_byte(address, in_channel)  # 选择 AIN0 通道

num_samples = 100
sample_rate = 50  # Hz

# 主程序
if __name__ == "__main__":
    try:

        data = np.zeros(num_samples, dtype=float)
        t = np.zeros(num_samples, dtype=float)

        basic_sleep_time = 1 / sample_rate

        start_time = time.time()
        for i in range(num_samples):
            data[i] = bus.read_byte(address) * 3 / 255
            t[i] = time.time()
            time.sleep(basic_sleep_time)
        t -= start_time
        t, data = t[100:], data[100:]  # discard the first few ones that are
unstable
```

```
        plt.subplot(211)
        plt.plot(t, data)
        plt.xlabel('Time (s)')
        plt.ylabel('Voltage (V)')
        plt.title('Voltage vs Time')

        plt.subplot(212)
        data_fft = np.fft.fft(data)
        plt.plot(data)

        # freq = np.fft.fftfreq(num_samples, d=1 / sample_rate)
        # plt.plot(freq, np.abs(data_fft))
        plt.show()

        # FFT



    except KeyboardInterrupt:
        print("\nTerminated")
```

== Lec5

`get_temperature.py`

```
import math

import smbus
import time

from thonny.plugins.microbit.api_stubs.time import sleep

bus = smbus.SMBus(1)
address = 0b01001000
R0 = 10000
B = 3950
T0 = 298.15


# 读取模拟输入值
def read_analog_input():
    bus.write_byte(address, 0b00000000)  # 选择 AIN0 通道
    analog_value = bus.read_byte(address)
    return analog_value


def calculate_temperature(analogVal):
    Vr = 5 * float(analogVal) / 255
    Rt = 10000 * Vr / (5 - Vr)
    T = 1 / (1 / T0 + 1 / B * math.log(Rt / R0))
    return T


if __name__ == "__main__":
```

```
    while True:
        try:
            print(f'Temperature:
{(calculate_temperature(read_analog_input())-273.15):.2f} °C')
            time.sleep(0.2)
        except KeyboardInterrupt:
            print("\nTerminated")
            break
```

== Lec6

`ultrasonic_ranging.py`

```
import time

import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(18, GPIO.IN)


def send_trig():
    GPIO.output(17, GPIO.HIGH)
    sleep(0.00002)  # 20us
    GPIO.output(17, GPIO.LOW)


if __name__ == '__main__':
    try:
        while True:
            send_trig()
            while GPIO.input(18) == 0:
                sleep(0.00001)

            # print("Start")
            start_time = time.time()

            while GPIO.input(18) == 1:
                if time.time() - start_time > 0.01:
                    break
                sleep(0.00001)

            end_time = time.time()
            # print("End")

            echo_time = end_time - start_time
            # print(f'Echo time: {echo_time}')
            distance = echo_time * 34320 / 2
            print(distance)

            sleep(0.5)
```

```
    except KeyboardInterrupt:
        GPIO.cleanup()
        print("\nKeyboard Interrupt")
        pass
```

== Lec7

`control_engine.py`

```
import time
import matplotlib.pyplot as plt
import numpy as np
import RPi.GPIO as GPIO
from time import sleep


def pin_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(17, GPIO.OUT)


ANGLE_INTERVAL_TABLE = {180: 2.5 * 0.001}
PERIOD = 20 * 0.001


def control_angle(angle, duration):
    interval = ANGLE_INTERVAL_TABLE.get(angle)

    start_time = time.time()
    while time.time() - start_time < duration:
        GPIO.output(17, GPIO.HIGH)
        sleep(interval)
        GPIO.output(17, GPIO.LOW)
        sleep(PERIOD - interval)


if __name__ == '__main__':
    try:
        pin_setup()
        control_angle(180, 1000)
    except Exception:
        pass
    finally:
        GPIO.cleanup()
```

`music.py`

```
import time
import RPi.GPIO as GPIO
from time import sleep


def pin_setup():
```

```
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(17, GPIO.OUT)


FREQUENCY_TABLE = {
    "C2": 65.41, "D2": 73.42, "E2": 82.41, "F2": 87.31, "G2": 98.00, "A2":
110.00, "B2": 123.47,
    "C3": 130.81, "D3": 146.83, "E3": 164.81, "F3": 174.61, "G3": 196.00, "A3":
220.00, "B3": 246.94,
    "C4": 261.63, "D4": 293.66, "E4": 329.63, "F4": 349.23, "G4": 392.00, "A4":
440.00, "B4": 493.88,
    "C5": 523.25, "D5": 587.33, "E5": 659.25, "F5": 698.46, "G5": 783.99, "A5":
880.00, "B5": 987.77,
    "C6": 1046.50, "D6": 1174.64,
    "bB4": 466.16,"bB3": 466.16/2,
}


def play_melody(melody):
    for note, duration in melody:
        if note == 'NO':
            GPIO.output(17, GPIO.HIGH)
            sleep(duration)
        else:
            frequency = FREQUENCY_TABLE.get(note)
            if frequency is None:
                print(f"Error: Note {note} is not found.")
                # raise Exception  # TODO
            sound(frequency, duration)


def sound(frequency, interval):
    start_time = time.time()
    while time.time() - start_time < interval:
        sleep(0.5 / frequency)
        GPIO.output(17, GPIO.LOW)
        sleep(0.5 / frequency)
        GPIO.output(17, GPIO.HIGH)


if __name__ == '__main__':
    try:
        GPIO.setwarnings(False)
        pin_setup()
        # melody = [('C4', 0.5), ('E4', 0.5), ('G4', 1)]
        # k7 = ('B5', .2)
        k7 = ('bB3', .2)
        k1 = ('C4', .2)
        k2 = ('D4', .2)
        k5 = ('G3', .2)
        melody1 = [k7, k1, k2, k7, k1, k5, k7, k1, k2, k7, k1, k5, k7, k1, k2,
("NO", .2)]
        melody2 = [k7, k1, k5, k7, k1, k2, k7, k1, k5, k7, k1, k2, k7, k1, k5,
("NO", .2)]
        # 715712715712715
        melody3 = [('G4', .4), ('F4', .4), k2, k1, k7, k5]
```

```
        # +5+4+2+175
        play_melody(melody1)  # 按顺序播放音符
        play_melody(melody2)
        play_melody(melody1)  # 按顺序播放音符
        play_melody(melody3)
        play_melody([('G4', .8)])
        # play_melody([('C6', .5), ('D6', .5)])

    except KeyboardInterrupt:
        pass

    finally:
        GPIO.output(17, GPIO.HIGH)

# 7 +1 +2 7 +1 5 7 +1 +2 7 +1 5 7 +1 +2
# 715712715712715
# +5+4+2+175
```

== Lec8

`joystick_control_buzzler.py`

```
#!/usr/bin/env python3
from time import sleep

import smbus2 as smbus
import time
import RPi.GPIO as GPIO


PCF8591_ADDR = 0x48
bus = smbus.SMBus(1)
AIN0_ADDR = 0x40
GPIO_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN, GPIO.OUT)


def read_channel(channel):
    bus.write_byte(PCF8591_ADDR, AIN0_ADDR + channel)
    bus.read_byte(PCF8591_ADDR)
    data = bus.read_byte(PCF8591_ADDR)
    return data


def sound():
    GPIO.output(GPIO_PIN, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(GPIO_PIN, GPIO.HIGH)


if __name__ == '__main__':
    try:
        GPIO.output(GPIO_PIN, GPIO.HIGH)
```

```
        while True:
            ain0 = read_channel(0)
            print(f"AIN0: {ain0}")
            if ain0 > 200:
                print("Low")
                sound()
            else:
                GPIO.output(GPIO_PIN, GPIO.HIGH)
                print("High")
                sleep(0.5)
    except KeyboardInterrupt:
        GPIO.output(GPIO_PIN, GPIO.HIGH)
        pass
```

joystick_control_led.py

```
import smbus2 as smbus
import RPi.GPIO as GPIO
import time
import threading

PCF8591_ADDR = 0x48
bus = smbus.SMBus(1)
AIN0_ADDR = 0x40
GPIO_PIN = 25

GPIO.setmode(GPIO.BCM)
GPIO.setup(GPIO_PIN, GPIO.OUT)
pwm = GPIO.PWM(GPIO_PIN, 1000)
pwm.start(50)  # Initial percentage 50%

# 全局变量
duty_cycle = 50  # Initial percentage 50%
lock = threading.Lock()


# 读取 AIN0 的函数
def read_ain0():
    global duty_cycle
    while True:
        try:
            bus.write_byte(PCF8591_ADDR, AIN0_ADDR)
            bus.read_byte(PCF8591_ADDR)  # Dummy read
            ain0 = bus.read_byte(PCF8591_ADDR)
            time.sleep(0.1)

            with lock:  # <78: -=5, >178: +=5
                if ain0 > 178:
                    duty_cycle = min(duty_cycle + 5, 100)  # Cannot exceed 100%
                elif ain0 < 78:
                    duty_cycle = max(duty_cycle - 5, 0)  # Cannot be less than 0

            print(f"AIN0: {ain0}, Duty Cycle: {duty_cycle}%")
```

```
        except Exception as e:
            print(f"读取 AIN0 时出错: {e}")
            break


def control_pwm():
    global duty_cycle
    while True:
        with lock:
            pwm.ChangeDutyCycle(duty_cycle)
        time.sleep(0.1)


if __name__ == '__main__':
    try:
        read_thread = threading.Thread(target=read_ain0)
        pwm_thread = threading.Thread(target=control_pwm)

        read_thread.start()
        pwm_thread.start()

        read_thread.join()
        pwm_thread.join()

    except KeyboardInterrupt:
        print("\n程序已终止")

    finally:
        pwm.stop()
        GPIO.cleanup()
```

`read_joystick_voltage.py`

```
#!/usr/bin/env python3
import smbus2 as smbus
import time

PCF8591_ADDR = 0x48
bus = smbus.SMBus(1)
AIN0_ADDR = 0x40


def read_channel(channel):
    bus.write_byte(PCF8591_ADDR, AIN0_ADDR + channel)
    bus.read_byte(PCF8591_ADDR)
    data = bus.read_byte(PCF8591_ADDR)
    return data


if __name__ == '__main__':
    try:
        while True:
            ain0 = read_channel(0)
            ain1 = read_channel(1)
```

```
            ain2 = read_channel(2)
            print(f"AIN0: {ain0}, AIN1: {ain1}, AIN2: {ain2}")
            time.sleep(0.2)
    except KeyboardInterrupt:
        pass
```

== Lec9

`IR_control_sound.py`

```
import time
from time import sleep
import RPi.GPIO as GPIO
import lirc


def pin_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(23, GPIO.OUT)  # R
    GPIO.setup(24, GPIO.OUT)  # G
    GPIO.setup(25, GPIO.OUT)  # B


def parse_code(data):  # 解析按键
    if not data:  # 如果 data 是空列表，直接返回
        return

    key = data[0]  # 获取列表中的第一个元素，例如 'echo "KEY_1"'

    state_R = GPIO.input(23)
    state_G = GPIO.input(24)
    state_B = GPIO.input(25)

    print(f'R: {state_R}; G: {state_G}, B: {state_B}')
    if key == 'echo "KEY_1"':
        print('Key 1: Change R')
        GPIO.output(23, not state_R)
    elif key == 'echo "KEY_2"':
        print('Key 2: Change G')
        GPIO.output(24, not state_G)
    elif key == 'echo "KEY_3"':
        print('Key 3: Change B')
        GPIO.output(25, not state_B)



if __name__ == '__main__':
    try:
        pin_setup()
        print()
        sockid = lirc.init("~/Code/RemoteTest/src/Lec9/IR_control_sound.py",
blocking=False)
        lirc.load_config_file("/etc/lirc/lircrc")
        print('LIRC initialized: Socket ID:', sockid)
```

```
        while True:
            codes = lirc.nextcode()
            if codes:
                print(codes)
                parse_code(codes)

    except KeyboardInterrupt:
        pass

    finally:
        # GPIO.output(23, GPIO.HIGH)
        lirc.deinit()
        GPIO.cleanup()
```

== Lec 10

`button_interrupt.py`

```
import RPi.GPIO as GPIO
from time import sleep

task = 0  # 0: task0, 1: task1, 2: task2

PIN_IN = 5

PIN_R = 12
PIN_G = 16


def button_callback(channel):
    print("Button pressed")
    global task
    task = (task + 1) % 2
    print("New Task: ", task)


def pin_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    # Input Pin
    # GPIO.setup(PIN_IN, GPIO.IN)  # Set PIN_IN as input
    GPIO.setup(PIN_IN, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # Set PIN_IN as input
    print(GPIO.input(PIN_IN))
    GPIO.add_event_detect(PIN_IN, GPIO.FALLING, callback=button_callback,
bouncetime=200)  # Set event detector

    # Output Pins
    GPIO.setup([PIN_R, PIN_G], GPIO.OUT)


def task0():
    GPIO.output(PIN_R, GPIO.HIGH)
    GPIO.output(PIN_G, GPIO.LOW)
```

```
    pass


def task1():
    GPIO.output(PIN_G, GPIO.HIGH)
    GPIO.output(PIN_R, GPIO.LOW)
    pass


def task2():
    GPIO.output(PIN_R, GPIO.LOW)
    sleep(0.5)
    GPIO.output(PIN_R, GPIO.HIGH)
    sleep(0.4)
    pass


if __name__ == '__main__':

    try:
        pin_setup()
        while True:
            match task:
                case 0:
                    task0()
                case 1:
                    task1()
                case 2:
                    task2()
            sleep(0.1)
    except KeyboardInterrupt:
        print("\nKeyboard Interrupt")
    finally:
        GPIO.cleanup()
```