3.Button

Introduction

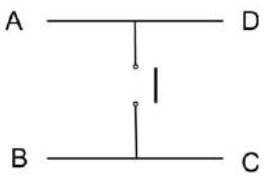
In this lesson, you will learn how to use push button with digital inputs to turn an LED on and off.

Hardware Required

- ✓ 1 * Raspberry Pi
- ✓ 1 * T-Extension Board
- ✓ 1 * LED
- ✓ 1 * Button
- ✓ 1 * 40-pin Cable
- ✓ Several Jumper Wires
- ✓ 1 * Breadboard
- ✓ 1 * Resistor(10KΩ)
- ✓ 1 * Resistor(220Ω)

Principle

Button



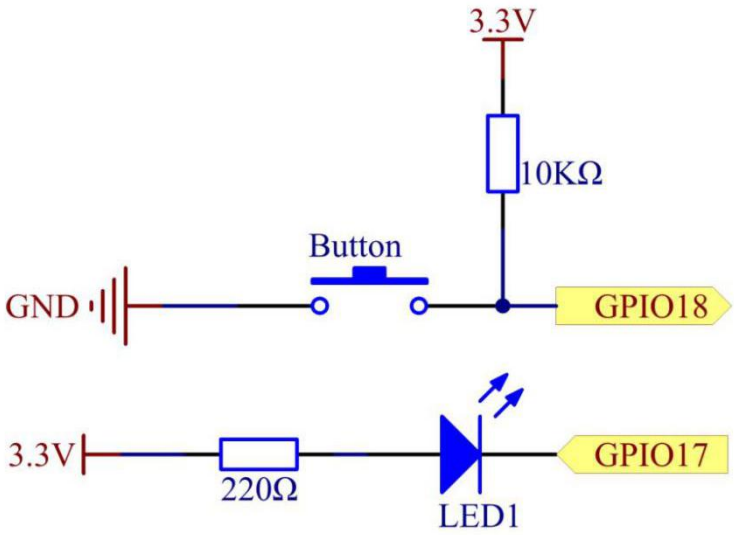
Buttons are a common component used to control electronic devices. They are usually used as switches to connect or disconnect circuits. Although buttons come in a variety of sizes and shapes, the one used here is a6mmmini-button as shown in the following pictures. Pins pointed out by the arrows of same color are meant to be connected.

Schematic Diagram

Use a normally open button as the input of Raspberry Pi, the connection is shown in the schematic diagram below. When the button is pressed, the GPIO18 will turn into low level (0V). We can detect the state of the GPIO18 through programming. That is, if the GPIO18 turns into low level, it means the button is pressed. You can run the corresponding code when the button is pressed, and then the LED will light up.

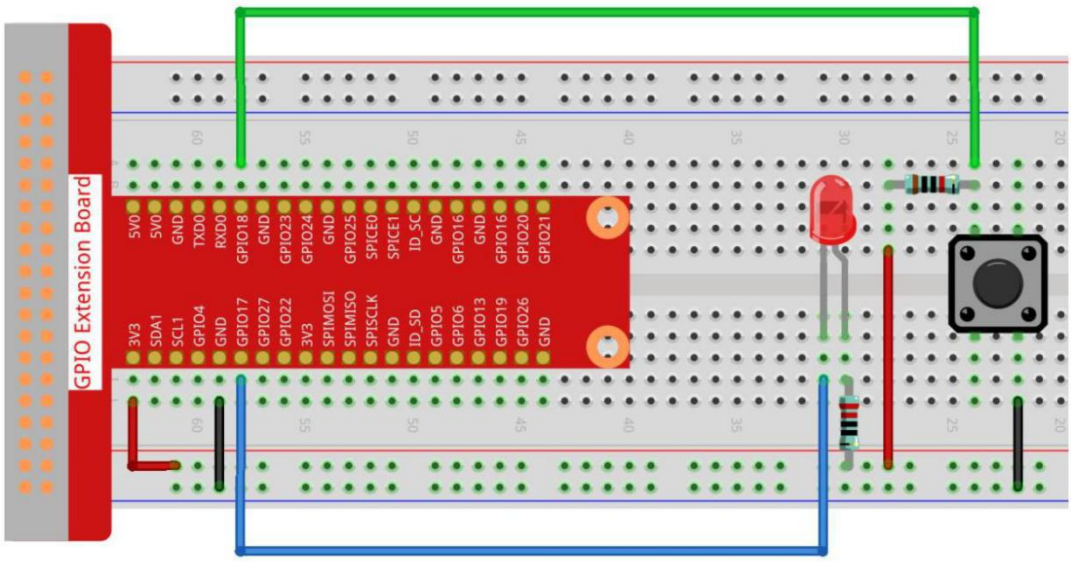
3.Button

T-Board Name	physical	wiringPi	BCM
GPIO17	Pin 11	0	17
GPIO18	Pin 12	1	18



Experimental Procedures

Step 1: Build the circuit.



For C Language Users

Step 2:Open the code file.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/C/3.Button
```

Note: Change directory to the path of the code in this experiment via cd.

Step 3: Compile the code.

```
gcc 3.Button.c -o Button.out -lwiringPi
```

Step 4: Run the executable file .

```
sudo ./Button.out
```

After the code runs, press the button, the LED lights up; otherwise, turns off.

Code

```
#include <wiringPi.h>
#include <stdio.h>

#define LedPin          0
#define ButtonPin       1

int main(void){
    // When initialize wiring failed, print message to screen
    if(wiringPiSetup() == -1){
        printf("setup wiringPi failed !");
        return 1;
    }

    pinMode(LedPin, OUTPUT);
    pinMode(ButtonPin, INPUT);
    // Pull up to 3.3V,make GPIO1 a stable level
    pullUpDnControl(ButtonPin, PUD_UP);

    digitalWrite(LedPin, HIGH);

    while(1){
        // Indicate that button has pressed down
```

```
        if(digitalRead(ButtonPin) == 0){
            // Led on
            digitalWrite(LedPin, LOW);
        //    printf("...LED on\n");
        }
        else{
            // Led off
            digitalWrite(LedPin, HIGH);
        //    printf("LED off...\n");
        }
    }
    return 0;
}
```

Code Explanation

```
#define LedPin          0
```

Pin GPIO17 in the T_Extension Board is equal to the GPIO0 in the wiringPi.

```
#define ButtonPin      1
```

ButtonPin is connected to GPIO1.

```
    pinMode(LedPin, OUTPUT);
```

Set LedPin as output to assign value to it.

```
    pinMode(ButtonPin, INPUT);
```

Set ButtonPin as input to read the value of ButtonPin.

```
    pullUpDnControl(ButtonPin, PUD_UP);
```

Set the ButtonPin as pull-up input. When the button is not pressed, the I/O port is 3.3V. When the button is pressed, the I/O port connects to GND (OV). You can judge the button status by reading the level value of the I/O port.

```
while(1){
        // Indicate that button has pressed down
        if(digitalRead(ButtonPin) == 0){
```

```
            // Led on
            digitalWrite(LedPin, LOW);
        //  printf("...LED on\n");
        }
        else{
            // Led off
            digitalWrite(LedPin, HIGH);
        //  printf("LED off...\n");
        }
```

if (digitalRead (ButtonPin) == 0: check whether the button has been pressed. Execute digitalWrite(LedPin, LOW) when button is pressed to light up LED.

For Python Language Users

Step 2: Open the code file.

```
cd /home/pi/REXQualis_Raspberry_Pi_Complete_Starter_Kit/Python
```

Step 3: Run the code.

```
sudo python3 3.Button.py
```

Now, press the button, and the LED will light up; press the button again, and the LED will go out. At the same time, the state of the LED will be printed on the screen.

Code

The code here is for Python3, if you need for Python2, please open the code with the suffix py2 in the attachment.

```
#!/usr/bin/python3

import RPi.GPIO as GPIO
import time
# Set GPIO17 as LED pin
LedPin = 17
# Set GPIO18 as button pin
```

3.Button

```
BtnPin = 18


# Set Led status to True(OFF)
Led_status = True


# Define a setup function for some setup
def setup():
    # Set the GPIO modes to BCM Numbering
    GPIO.setmode(GPIO.BCM)
    # Set LedPin's mode to output,
    # and initial level to high (3.3v)
    GPIO.setup(LedPin, GPIO.OUT, initial=GPIO.HIGH)
    # Set BtnPin's mode to input,
    # and pull up to high (3.3V)
    GPIO.setup(BtnPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    # Set up a falling detect on BtnPin,
    # and callback function to swLed
    GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed)


# Define a callback function for button callback
def swLed(ev=None):
    global Led_status
    # Switch led status(on-->off; off-->on)
    Led_status = not Led_status
    GPIO.output(LedPin, Led_status)
    #if Led_status:
    #    print 'LED OFF...'
    #else:
    #    print '...LED ON'
```

3.Button

```
# Define a main function for main process
def main():
    while True:
        # Don't do anything.
        time.sleep(1)

# Define a destroy function for clean up everything after
# the script finished
def destroy():
    # Turn off LED
    GPIO.output(LedPin, GPIO.HIGH)
    # Release resource
    GPIO.cleanup()

# If run this script directly, do:
if __name__ == '__main__':
    setup()
    try:
        main()
    # When 'Ctrl+C' is pressed, the program
    # destroy() will be    executed.
    except KeyboardInterrupt:
        destroy()
```

Code Explanation

```
LedPin = 17
```

Set GPIO17 as LED pin

```
BtnPin = 18
```

Set GPIO18 as button pin

```
GPIO.add_event_detect(BtnPin, GPIO.FALLING, callback=swLed)
```

Set up a falling detect on BtnPin, and then when the value of BtnPin changes from a high level to a low level, it means that the button is pressed. The next step is calling the function, swled.

```
def swLed(ev=None):
    global Led_status
    # Switch led status(on-->off; off-->on)
    Led_status = not Led_status
    GPIO.output(LedPin, Led_status)
```

Define a callback function as button callback. When the button is pressed at the first time,and the condition, not Led_status is false, GPIO.output() function is called to light up the LED. As the button is pressed once again, the state of LED will be converted from false to true, thus the LED will turn off.

Phenomenon Picture