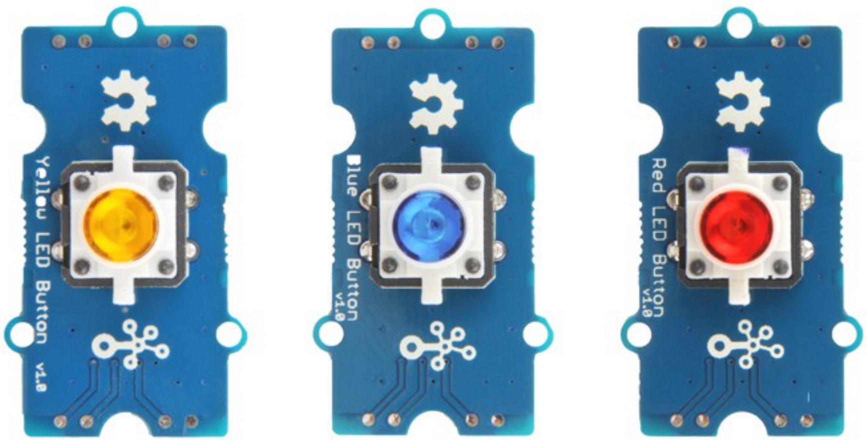
Grove-LED Button SKU：111020045



The Grove - LED Button is composed of Grove - Yellow Button, Grove - Blue LED Button and Grove - Red LED Button. This button is stable and reliable with a 100 000 times long life. With the build-in LED, you can apply it to many interesting projects, it is really useful to use the LED to show the status of the button. We use a high-quality N-Channel MOSFET to control the LED to ensure the high swithching speed and a low consumption.All in all, you want some relly awesome button? Here you go ...

Version

Product Version	Changes	Released Date
Grove-LED Button	Initial	Jun 2018

Features

- Long operating life
- Easy to use
- Grove Digital interface

Specification

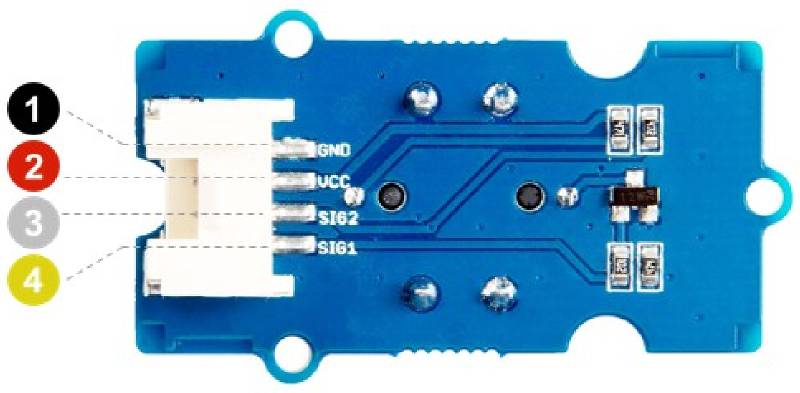
Item	Value
Working voltage	3.3V/5V
Operating Life without Load	100 000 times
LED rated current	50mA
Press Resistance^1^	<100mΩ

Item	Value
Release Resistance^2^	>100MΩ
Size	L: 40mm W: 20mm H: 13mm
Weight	4.3g
Package size	L: 140mm W: 90mm H: 10mm
Gross Weight	11g

!!!Tips 1,2- If you want to measure the resistance, please take the key cap off the board. Otherwise you will get the value of the equivalent resistance of the board instead of the true resistance of the key cap.

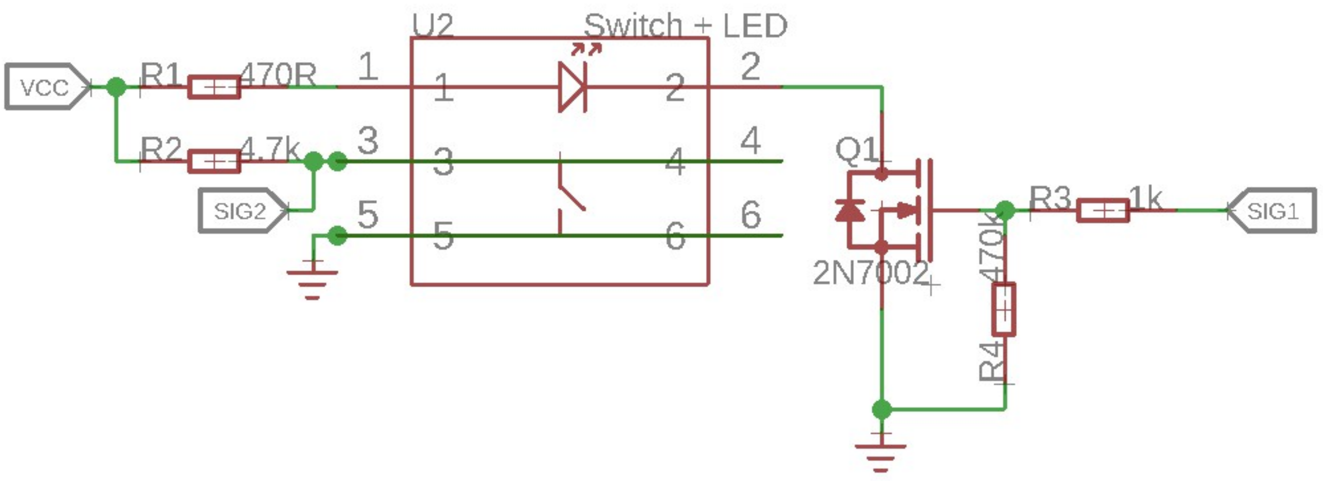
Hardware Overview

Pin Map



1. GND: connect this module to the system GND
2. VCC: you can use 5V or 3.3V for this module
3. SIG2: output the button status, default high
4. SIG1: input the LED control signal, default low

Schematic



SIG1 is the the LED control signal, the default value is low, so the N-Channel MOSFET is off , the LED is off too. When SIG1 becomes high, the N-Channel MOSFET trun on, and the LED light on.

SIG2 connect to the button pin. With a pull-up resistance, the default value of **SIG2** is high. When you press the button, the voltage is pulled low, the **SIG2** becomes to low.

Platforms Supported

Arduino	Raspberry Pi	BeagleBone	Wio	LinkIt ONE

Getting Started

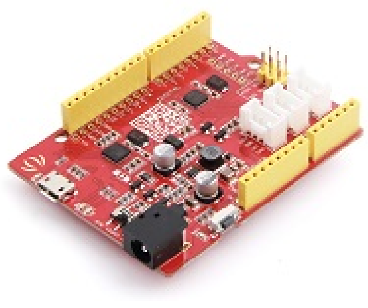
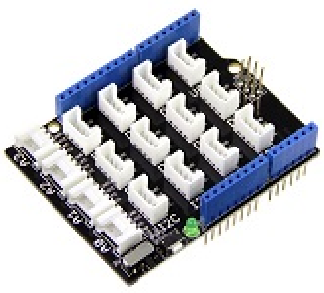
!!!Tips In this part, we use the Grove - Red LED Button as an example. The following parts also apply to Yellow and Blue.

Play With Arduino

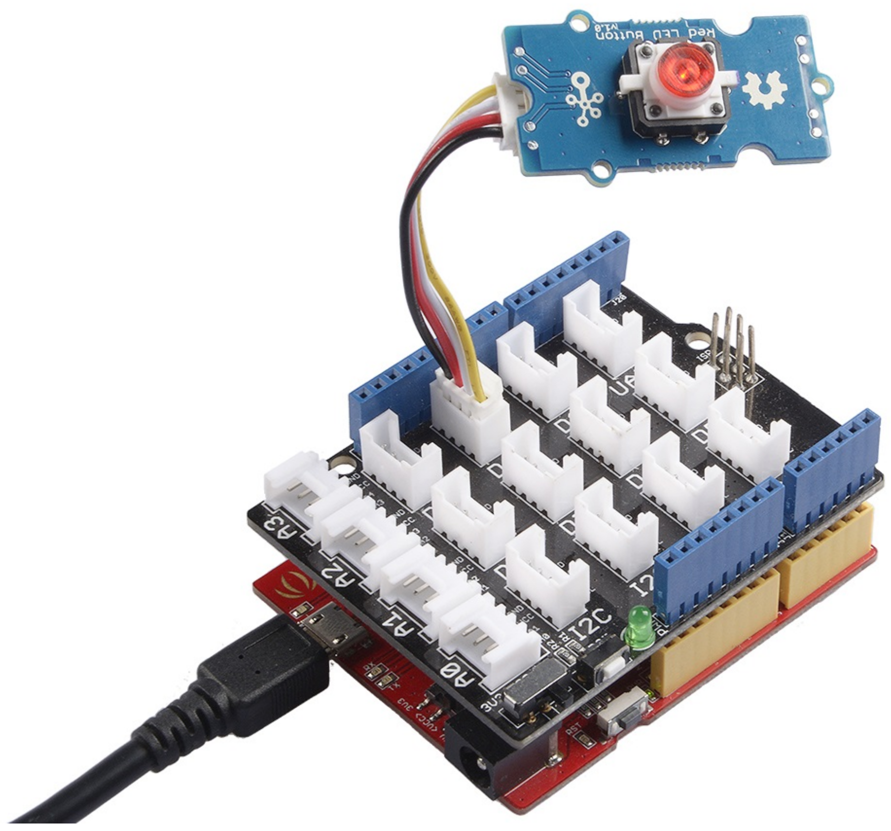
Hardware

Materials required

Seeeduino V4.2	Base Shield	Grove- Red LED Button

Seeeduino V4.2	Base Shield	Grove- Red LED Button
		

- **Step 1.** Grove- Red LED Button to port **D3** of Grove-Base Shield.
- **Step 2.** Plug Grove - Base Shield into Seeeduino.
- **Step 3.** Connect Seeeduino to PC via a USB cable.



!!!Note If we don't have Grove Base Shield, We also can directly connect this module to Seeeduino as below.

Seeeduino	Grove- Red LED Button
5V	Red
GND	Black
SIG2	White
SIG1	Yellow

Software

!!!Note If this is the first time you work with Arduino, we strongly recommend you to see Getting Started with Arduino before the start.

- **Step 1.** Open the Arduino IDE and create a new file, then copy the following code into the new file.

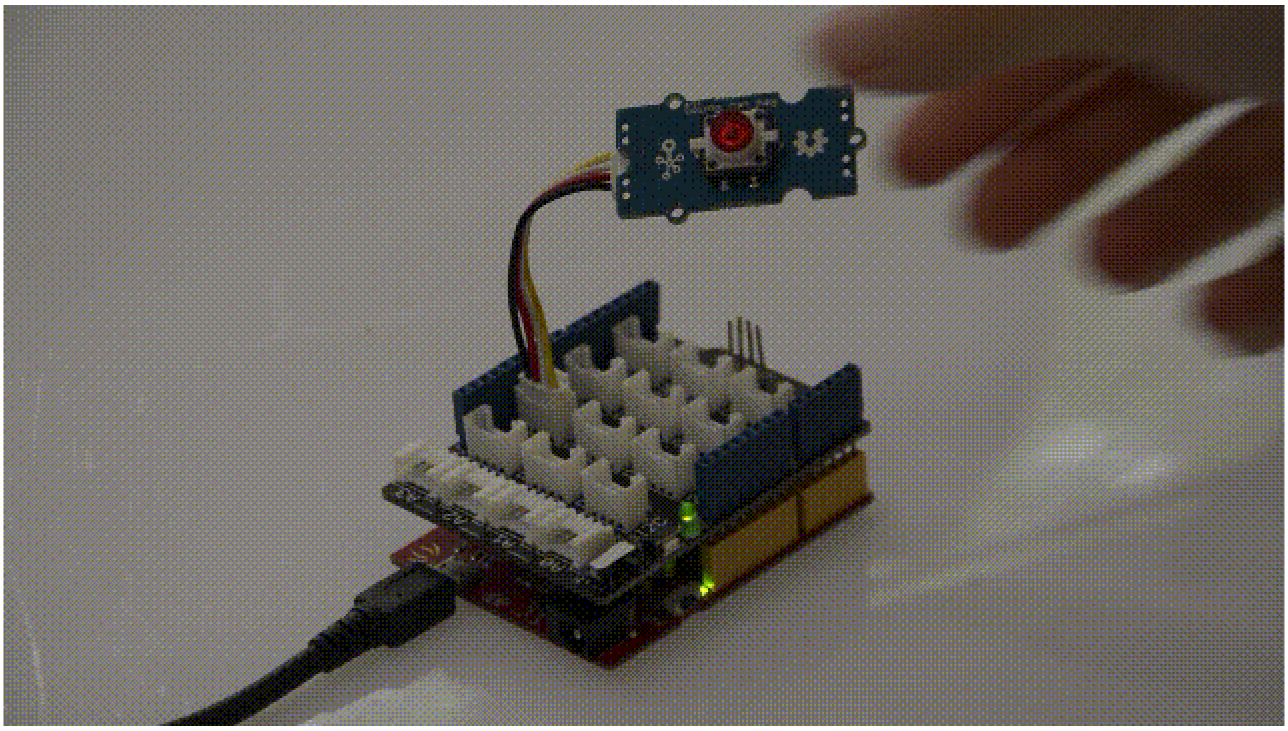
```
#include "Arduino.h"

//1: toggle mode, 2: follow mode
#define LED_MODE   1

const int ledPin = 3;      // the number of the LED pin, D3
const int buttonPin = 4;    // the number of the pushbutton pin, D4
const boolean breathMode = true;  // if or not the led lights as breath mode when
it's on

// Variables will change:
int ledState = LOW;         // the current state of the output pin
int ledFadeValue = 0;
int ledFadeStep = 5;
int ledFadeInterval = 20;   //milliseconds
int buttonState;             // the current reading from the input pin
int lastButtonState = HIGH;   // the previous reading from the input pin

unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output
flickers
unsigned long lastLedFadeTime = 0;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, ledState);
}

void loop() {
  int reading = digitalRead(buttonPin);

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer
    // than the debounce delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

#if LED_MODE == 1
```

```
        if (buttonState == LOW) {  //button is pressed
            ledState = !ledState;
            ledFadeValue = 0;
            lastLedFadeTime = millis();
        }
#else
        if (buttonState == LOW) {  //button is pressed
          ledState = HIGH;
          ledFadeValue = 0;
          lastLedFadeTime = millis();
        } else {                   //button is released
          ledState = LOW;
        }
#endif
      }
    }

    // set the LED:
    if (breathMode && ledState != LOW) {
      if (millis() - lastLedFadeTime > ledFadeInterval) {
        lastLedFadeTime = millis();
        analogWrite(ledPin, ledFadeValue);
        ledFadeValue += ledFadeStep;
        if (ledFadeValue > 255){
          ledFadeValue = 255 - ledFadeStep;
          ledFadeStep = -ledFadeStep;
        } else if (ledFadeValue < 0) {
          ledFadeValue = 0;
          ledFadeStep = -ledFadeStep;
        }
      }
    } else {
      digitalWrite(ledPin, ledState);
    }

    lastButtonState = reading;
  }
```

!!!Tip In this demo, we choose mode 1 which is the toggle mode, you can change the line 4 #define LED_MODE 1 into #define LED_MODE 2 to use the follow mode.

- **Step 2.** Upload the demo. If you do not know how to upload the code, please check How to upload code.
- **Step 3.** Now, try to press you button, you will see the LED light on with a fade on/fade off effect.

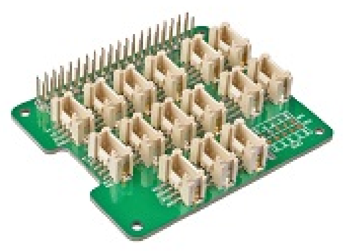
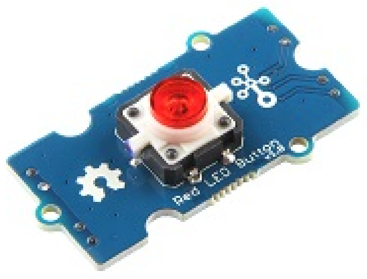
It should be like:



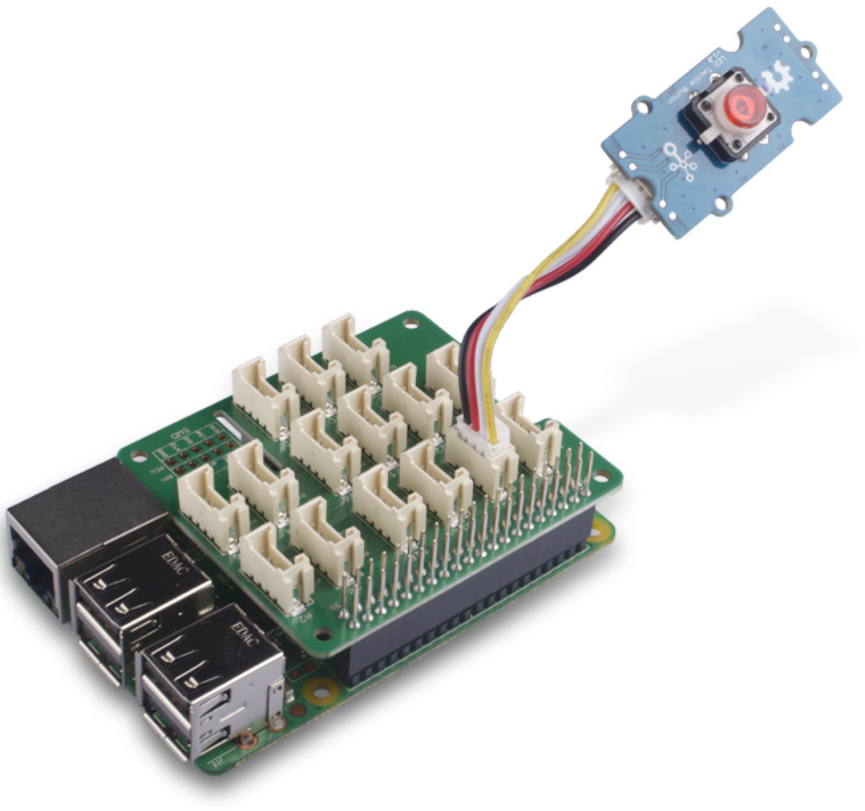
Play With Raspberry Pi

Hardware

- **Step 1**. Things used in this project:

Raspberry pi	Grove Base Hat for RasPi	Grove - Red LED Button
		

- **Step 2**. Plug the Grove Base Hat into Raspberry.
- **Step 3**. Connect the red LED button to D5 port of the Base Hat.
- **Step 4**. Connect the Raspberry Pi to PC through USB cable.



!!! Note For step 3 you are able to connect the LED button to **any GPIO Port** but make sure you change the command with the corresponding port number.

Software

- **Step 1**. Follow Setting Software to configure the development environment.
- **Step 2**. Download the source file by cloning the grove.py library.

```
cd ~
git clone https://github.com/Seeed-Studio/grove.py
```

- **Step 3**. Excute below commands to run the code.

```
cd grove.py/grove
python grove_ryb_led_button.py 5
```

Following is the grove_ryb_led_button.py code.

```
import time
from grove.button import Button
from grove.factory import Factory

class GroveLedButton(object):
    def __init__(self, pin):
        # High = light on
        self.__led = Factory.getOneLed("GPIO-HIGH", pin)
        # Low = pressed
        self.__btn = Factory.getButton("GPIO-LOW", pin + 1)
        self.__on_event = None
        self.__btn.on_event(self, GroveLedButton.__handle_event)

    @property
    def on_event(self):
        return self.__on_event

    @on_event.setter
    def on_event(self, callback):
        if not callable(callback):
            return
        self.__on_event = callback

    def __handle_event(self, evt):
        # print("event index:{} event:{} pressed:{}".format(evt['index'],
evt['code'], evt['presesed']))
        if callable(self.__on_event):
            self.__on_event(evt['index'], evt['code'], evt['time'])
            return

        self.__led.brightness = self.__led.MAX_BRIGHT
        event = evt['code']
        if event & Button.EV_SINGLE_CLICK:
            self.__led.light(True)
            print("turn on  LED")
        elif event & Button.EV_DOUBLE_CLICK:
            self.__led.blink()
            print("blink    LED")
        elif event & Button.EV_LONG_PRESS:
            self.__led.light(False)
            print("turn off LED")


Grove = GroveLedButton

def main():
    from grove.helper import SlotHelper
    sh = SlotHelper(SlotHelper.GPIO)
    pin = sh.argv2pin()

    ledbtn = GroveLedButton(pin)
    # remove ''' pairs below to begin your experiment
    '''
    # define a customized event handle your self
    def cust_on_event(index, event, tm):
        print("event with code {}, time {}".format(event, tm))

    ledbtn.on_event = cust_on_event
    '''
    while True:
        time.sleep(1)


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

!!!success If everything goes well, you will be able to see the LED turns on if you press it and turns off if you long press it. If you double click the LED button, the LED will blink.

```
pi@raspberrypi:~/grove.py/grove $ python grove_ryb_led_button.py 5
Hat Name = 'Grove Base Hat RPi'
turn on  LED
turn on  LED
blink    LED
turn on  LED
turn off LED
^CTraceback (most recent call last):
  File "grove_ryb_led_button.py", line 101, in <module>
    main()
  File "grove_ryb_led_button.py", line 97, in main
    time.sleep(1)
KeyboardInterrupt
```

You can quit this program by simply press ++ctrl+c++.

Resources

- **[Zip]** Grove-LED Button Eagle file

Tech Support

Please do not hesitate to submit the issue into our forum.