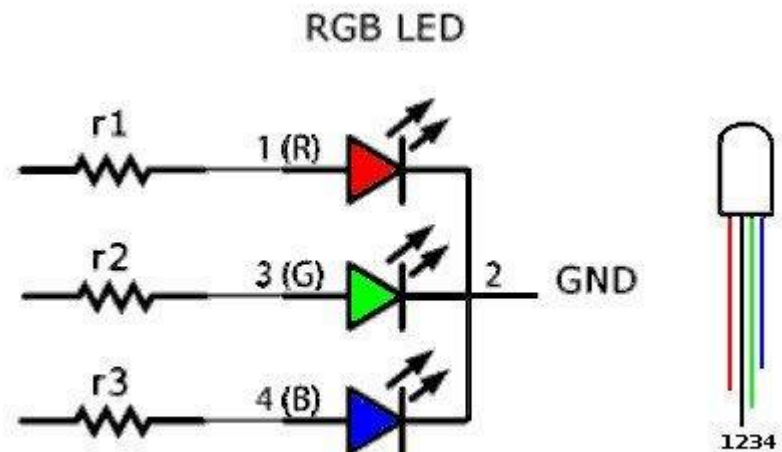
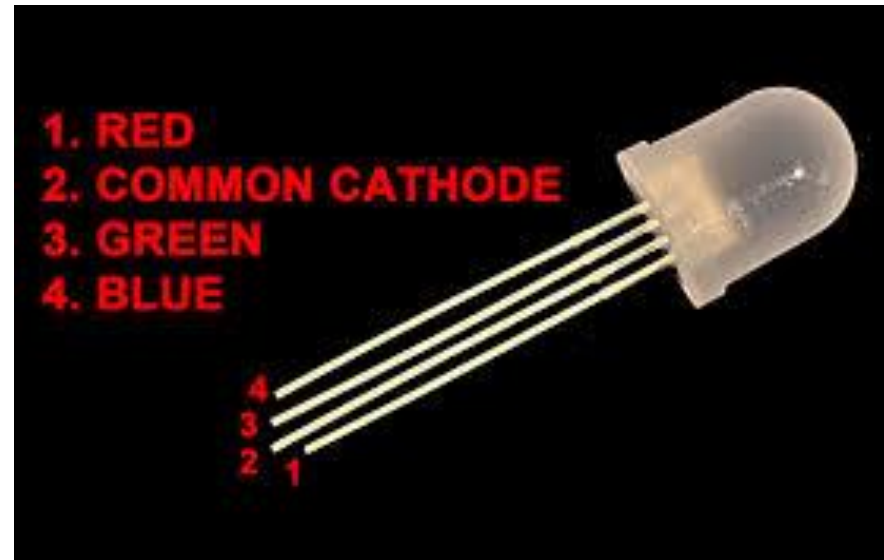
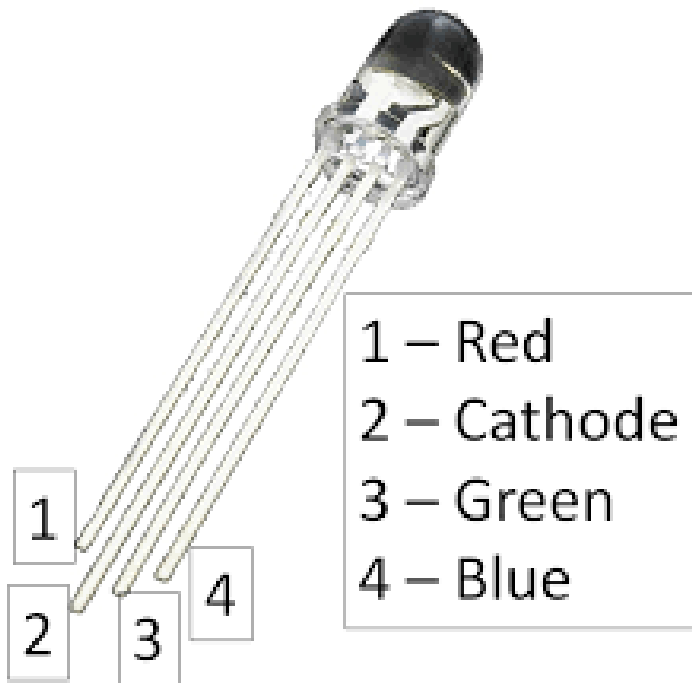


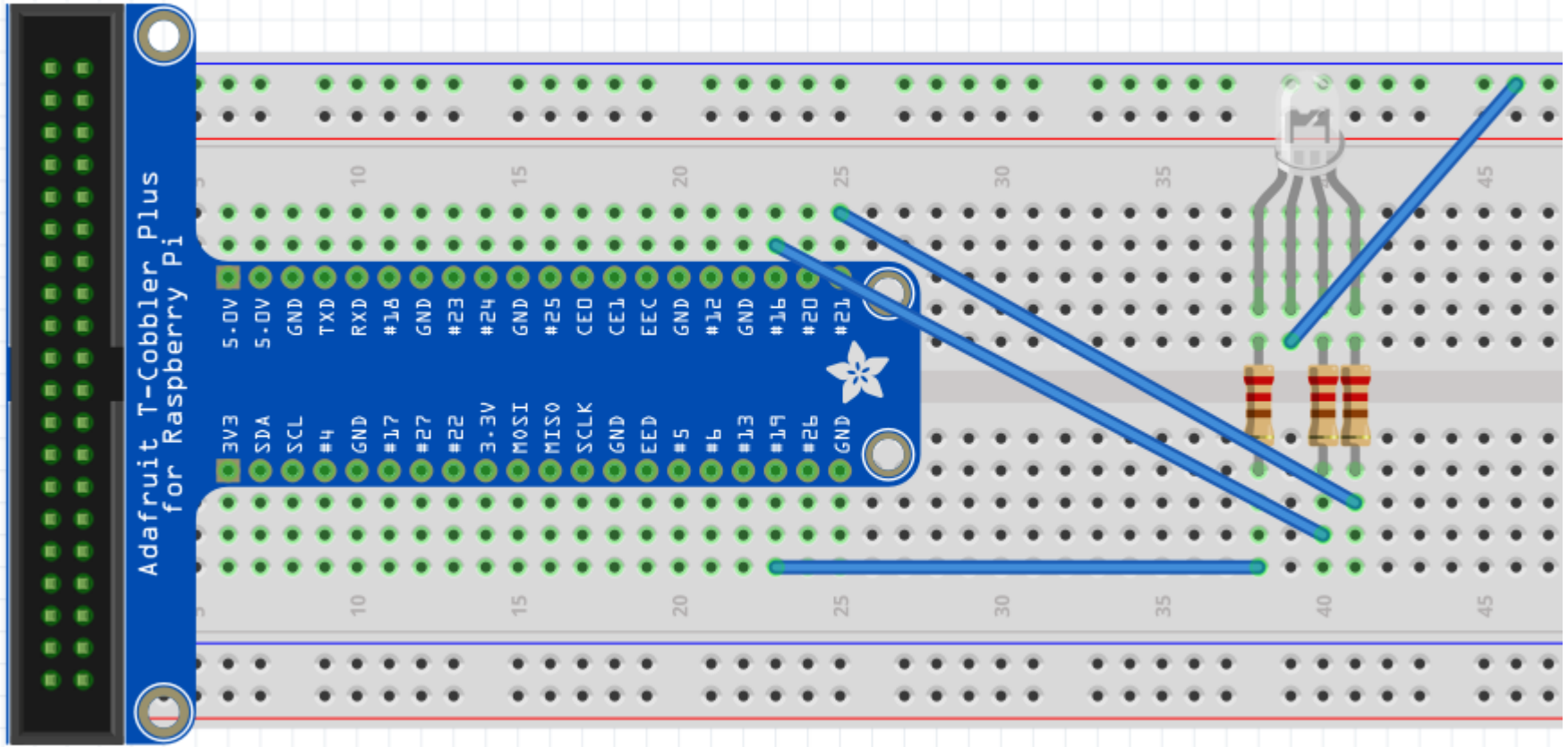
Intro to the Raspberry Pi (RPi) & The Internet of Things (IoT)

Class 4 – 2018 / 04 / 01

RGB LED – Common CATHODE



RGB LED





```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time
```

```
leds = [ 21, 16, 19 ]
```



```
def setup():
    GPIO.setmode(GPIO.BCM)
    for led_pin in leds:
        GPIO.setup(led_pin, GPIO.OUT)
```

```
def led_on(led_pin):
    GPIO.output(led_pin, GPIO.HIGH)
```

```
def led_off(led_pin):
    GPIO.output(led_pin, GPIO.LOW)
```

```
def main():
    speed = int(input("Speed in milliseconds? ")) / 1000.0
```

```
    try:
        setup()
        while True:
            for led in leds:
                led_on(led)
                time.sleep(speed)
                led_off(led)
```

```
    finally:
        for led in leds:
            led_off(led)
        GPIO.cleanup()
```

```
main()
```

Notice the use of lists:
leds = [21, 16, 19]

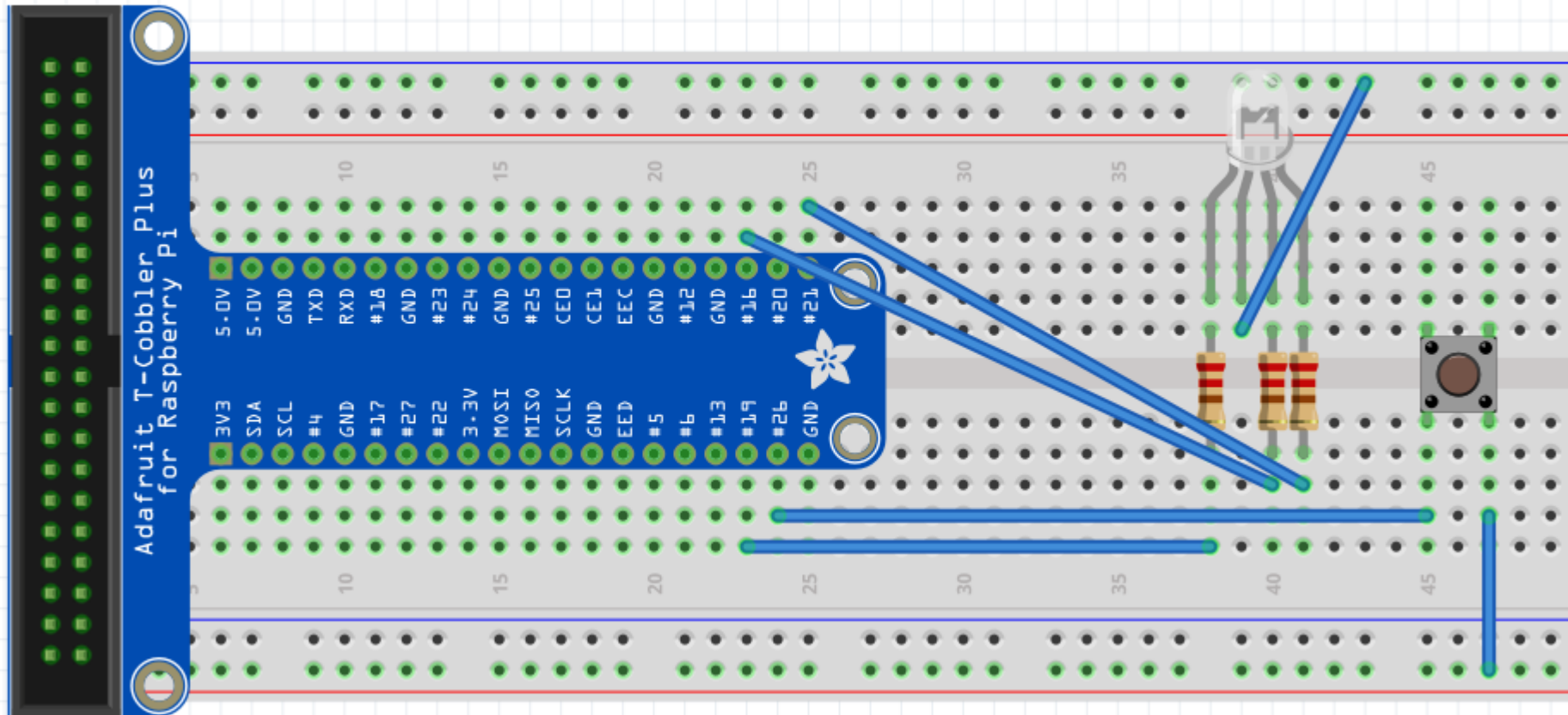
„Iteration“ through a list:
for led_pin in leds:
<do something>

**Lists save a lot
of variables
(led1, led2, led3)**

Digital Input - Button



How NOT To Connect the Button – But let's try, you will be amazed!!



```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time

leds = [ 21, 16, 19 ]
button_pin = 26

def setup():
    GPIO.setmode(GPIO.BCM)
    for led_pin in leds:
        GPIO.setup(led_pin, GPIO.OUT)
    GPIO.setup(button_pin, GPIO.IN)

def led_on(led_pin):
    GPIO.output(led_pin, GPIO.HIGH)

def led_off(led_pin):
    GPIO.output(led_pin, GPIO.LOW)

def read_button(button):
    return GPIO.input(button_pin)

def main():

    try:
        setup()
        led = 0

        while True:
            if read_button(button_pin) == GPIO.HIGH:
                led_off(leds[led])
                led += 1
                led %= len(leds)
                led_on(leds[led])

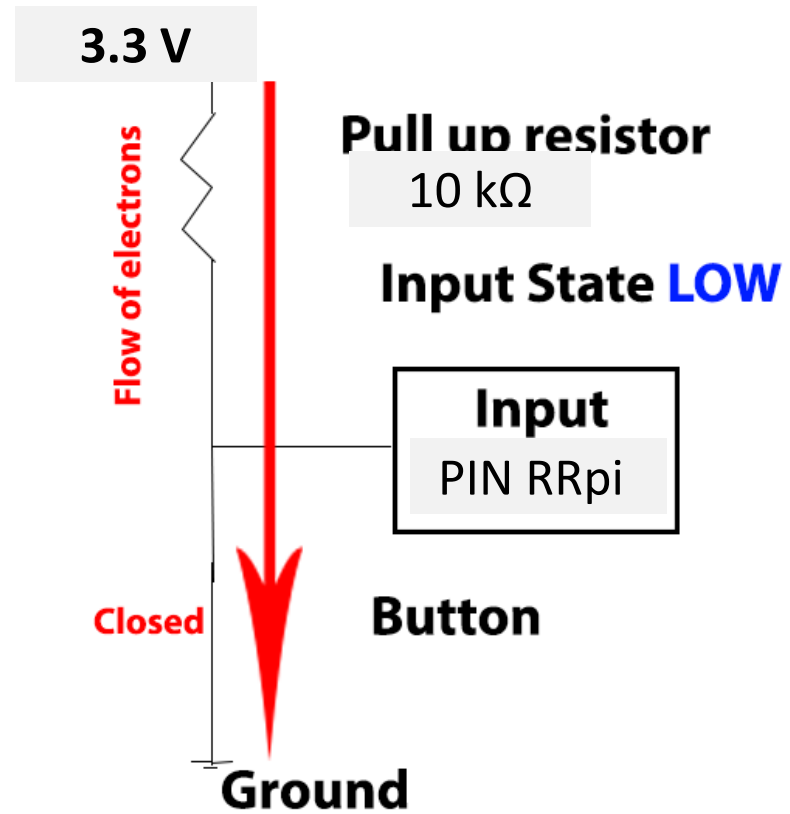
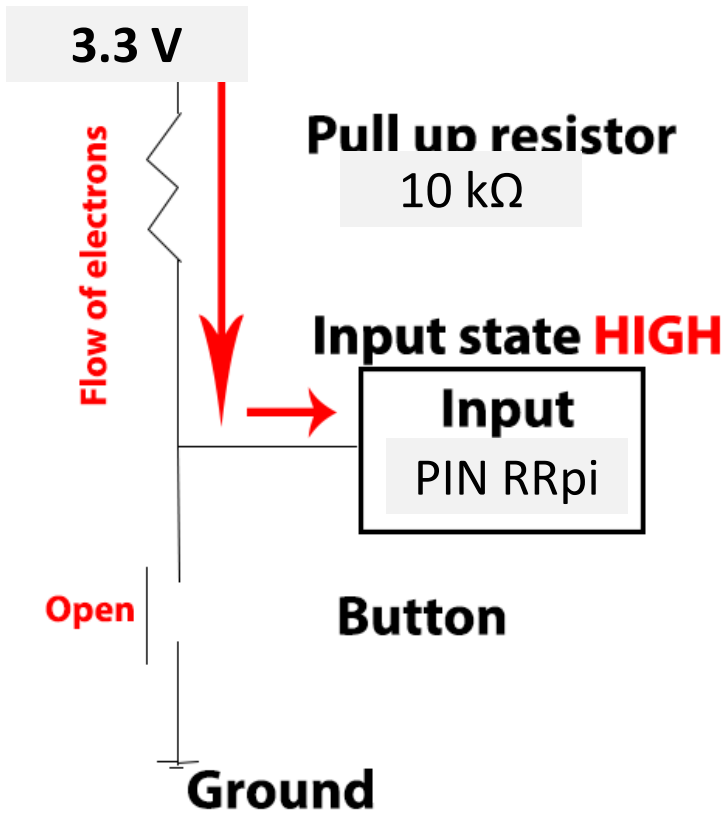
    finally:
        for led in leds:
            led_off(led)
        GPIO.cleanup()

main()
```

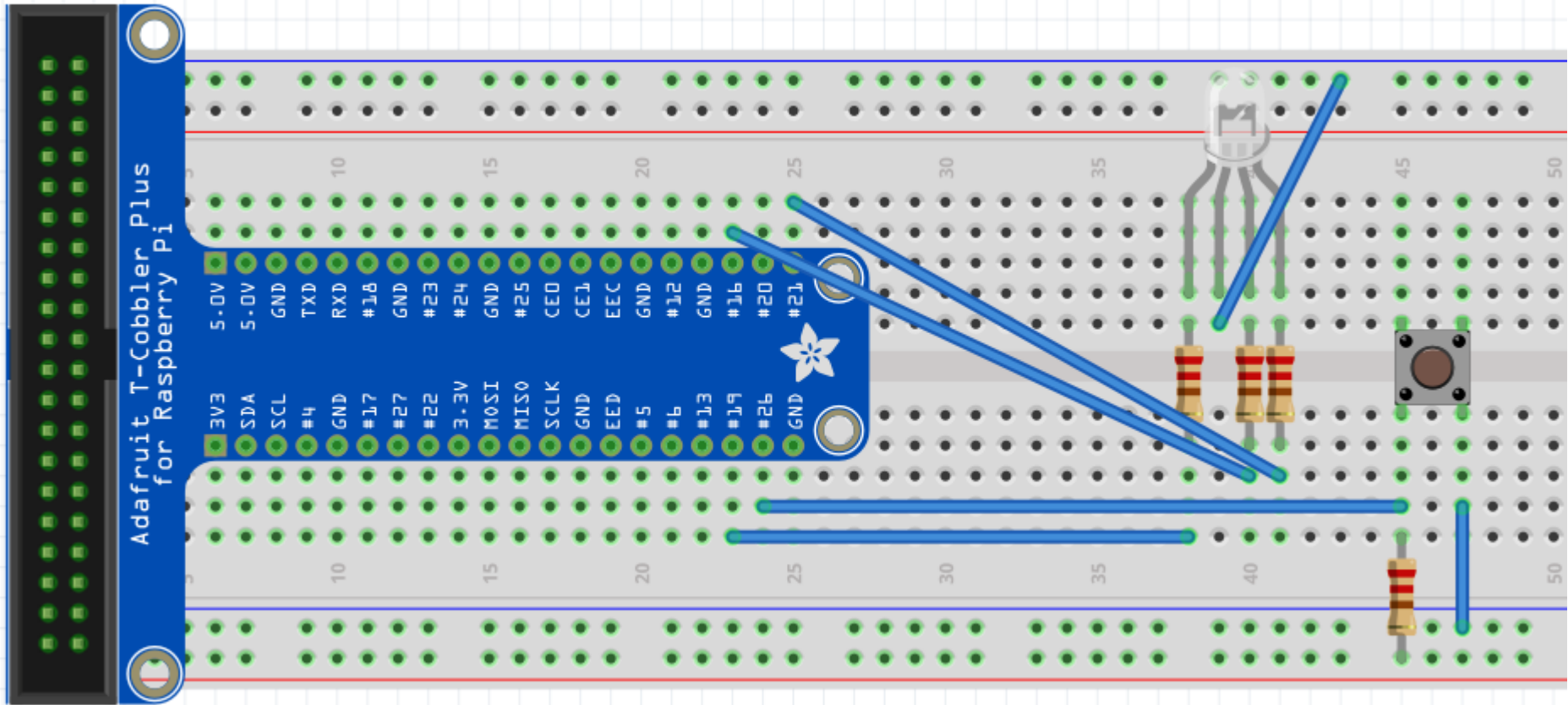
Explanation – Floating Inputs

- Unconnected Inputs do not have a well-defined logical state!
 - They are neither 0 / LOW nor 1 / HIGH
 - We need to use something more elaborate
 - **Pull Up Resistor circuit**
 - When not pressed, logic level = HIGH
 - When pressed, logic level = LOW
 - There are also **Pull Down resistors**
 - Inverse function (different circuit!)

Pull Up Resistor



Important - Use the 10 k Ω resistor !
From the smaller pack of 5 resistors



And one more problem...

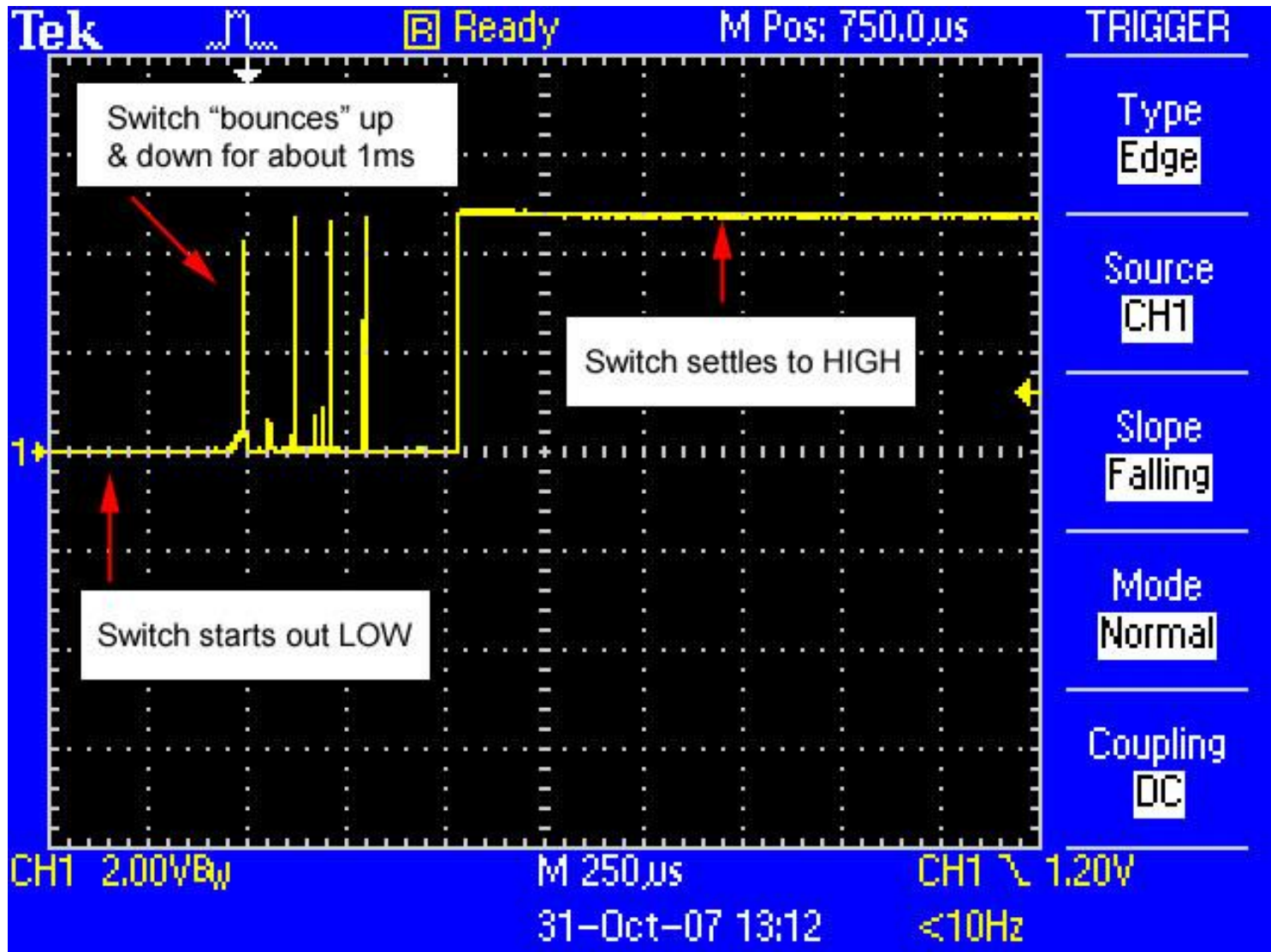
- Even if we wait until the button goes LOW again before we continue in our main loop:

```
while read_button(button_pin) == GPIO.LOW:  
    pass
```



**... then there is STILL something strange going on!
Sometimes, the LED jumps over colors, or there
is no change at all! WHY?**

Button Debouncing



```

#!/usr/bin/python3
import RPi.GPIO as GPIO
import time

leds = [ 21, 16, 19 ]

# configure as pull up - pin over 10 kOhms to 3.3 and button
# other button pin to GND
# normally, HIGH, when closed, LOW

button_pin = 26

def setup():
    GPIO.setmode(GPIO.BCM)
    for led_pin in leds:
        GPIO.setup(led_pin, GPIO.OUT)
    GPIO.setup(button_pin, GPIO.IN)

def led_on(led_pin):
    GPIO.output(led_pin, GPIO.HIGH)

def led_off(led_pin):
    GPIO.output(led_pin, GPIO.LOW)

def read_button(button):
    return GPIO.input(button_pin)

def main():
    try:
        setup()
        led = 0
        while True:
            if read_button(button_pin) == GPIO.LOW:
                led_off(leds[led])
                #led += 1
                #led %= len(leds)
                led = ( led + 1 ) % len(leds)
                led_on(leds[led])
                time.sleep(0.1)
                while read_button(button_pin) == GPIO.LOW:
                    pass
    finally:
        for led in leds:
            led_off(led)
        GPIO.cleanup()

main()

```

Projects - Level 1 Difficulty

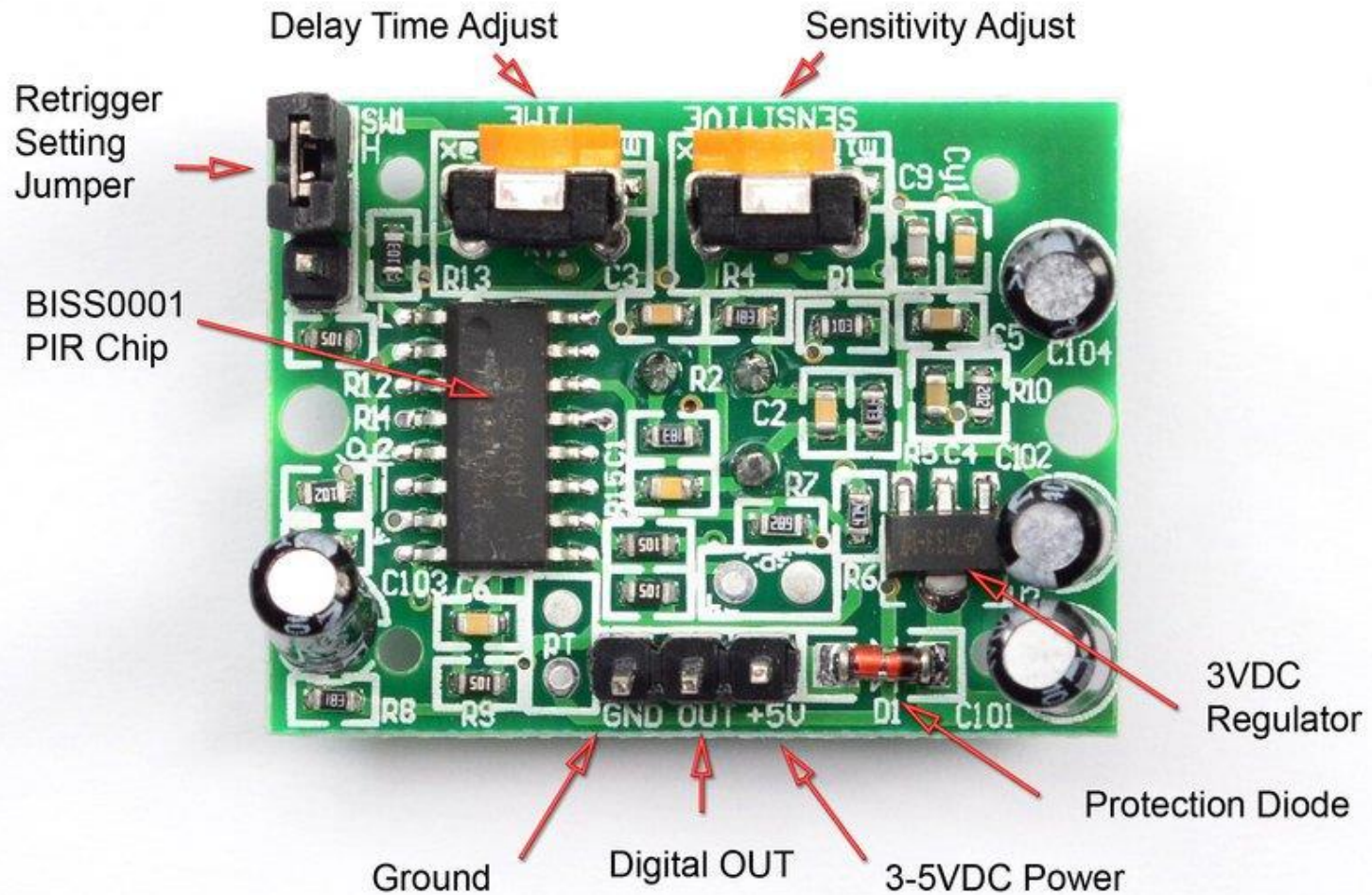
- Light mixer
 - Use 3 Buttons and RGB LED to „mix“ 7 colors
- Sound board
 - Use 3 Buttons for different sound effects
 - Have LEDs, too
- Binary Counter
 - Use 3 or 4 LEDs and one button to show binary numbers
 - number += 1 per button press
 - You will need to DEBOUNCE the button

Projects - Level 2 Difficulty

- IR Alarm
 - Get a **PIR sensor** from me (I have 10) and figure out how to use it to make an alarm system!
 - Add alarm sound
 - Add button to cancel alarm
 - Add button to arm / disarm
 - Add LEDs for armed / disarmed stat
 - Blink LEDs when alarm



PIR Sensor



Projects - Level 2 Difficulty

- „Catch the LED“ Game
 - A reaction game with 3 LEDs and 3 buttons
 - There is one button per LED
 - After a random time between 0.2 and 5 seconds, a random LED lights up
 - Use has to push the corresponding button *fast* (0.2 seconds?) to get a point
 - Don't allow to cheat!
- After 20 attempts, report points
- Game could accelerate

Projects – Level 3 Difficulty

- Make a „Simon“ game
 - 3 LEDs
 - 3 buttons
 - 3 sounds
 - ... or better, 4 😊



Programming Technique – Time Measurement

```
import time
```

```
ts = time.clock()
```

```
wait_time = 1.5    # in seconds
```

```
while True:
```

```
    if time.clock() - ts < wait_time :
```

```
        print("time passed!")
```

```
        break
```

Programming Technique – Random Integers and Choices

```
import random
```

```
number = random.randint(30,5000)
```

```
print(number)
```

```
random.choice(["Great", "Nice", "Good"])
```

Programming Technique – How to Make a Sound

```
import pygame
```

```
pygame.init()
```

```
file = '/usr/lib/libreoffice/share/gallery/sounds/laser.wav'
```

```
sound = pygame.mixer.Sound(file)
```

```
sound.play()
```

```
...
```

```
sound.stop()
```

**There are plenty of other sounds
already on the Raspberry Pi!**

In the Console, find them:

```
cd /usr/  
find . -name "*.wav"
```

Programming Technique – Break Out of a Loop

- **break** out of a loop when a condition is met:

```
for x in range(10):  
    print(x)  
    if x == 5:  
        break
```

Will print 0, 1, 2, 3, 4, 5 only

- Works for **while** and **for**

Programming Technique – Skip Remainder of Loop Body & Start Next Round

- Use **continue** to skip what's still coming and start new round of the loop:

```
for x in range(10):  
    if x < 6:  
        continue  
    print(x)
```

Will print 6, 7, 8, 9 only

- Works **for** while and **for**