https://raspi.tv/2014/rpi-gpio-update-and-detecting-both-rising-and-falling-edges

A few days ago, Ben Croston updated the RPi.GPIO Python library so that the 'board' pin numbering system would work with all of the pins on the B+. It doesn't really affect the way I work, as I always use BCM port numbers. They already worked on the B+ and the compute module.

While he was at it, he made a couple of bug-fix tweaks. Looking at what was tweaked, I realised there are a couple of features of RPi.GPIO that I hadn't yet documented. In this blog post I hope to update my RPi.GPIO documentation. RPi.GPIO 0.5.6 is already in the Raspbian repository, so to update to 0.5.6, just…

```
sudo apt-get update
sudo apt-get upgrade
y
```

## Edge Detection Addition

In my previous tutorials on threaded callbacks we used edge detection of RISING and FALLING edges. Somewhere along the line, Ben introduced an option to test for BOTH falling and rising edges. I knew about this, but hadn't tried it before.
So, where previously we had the code…

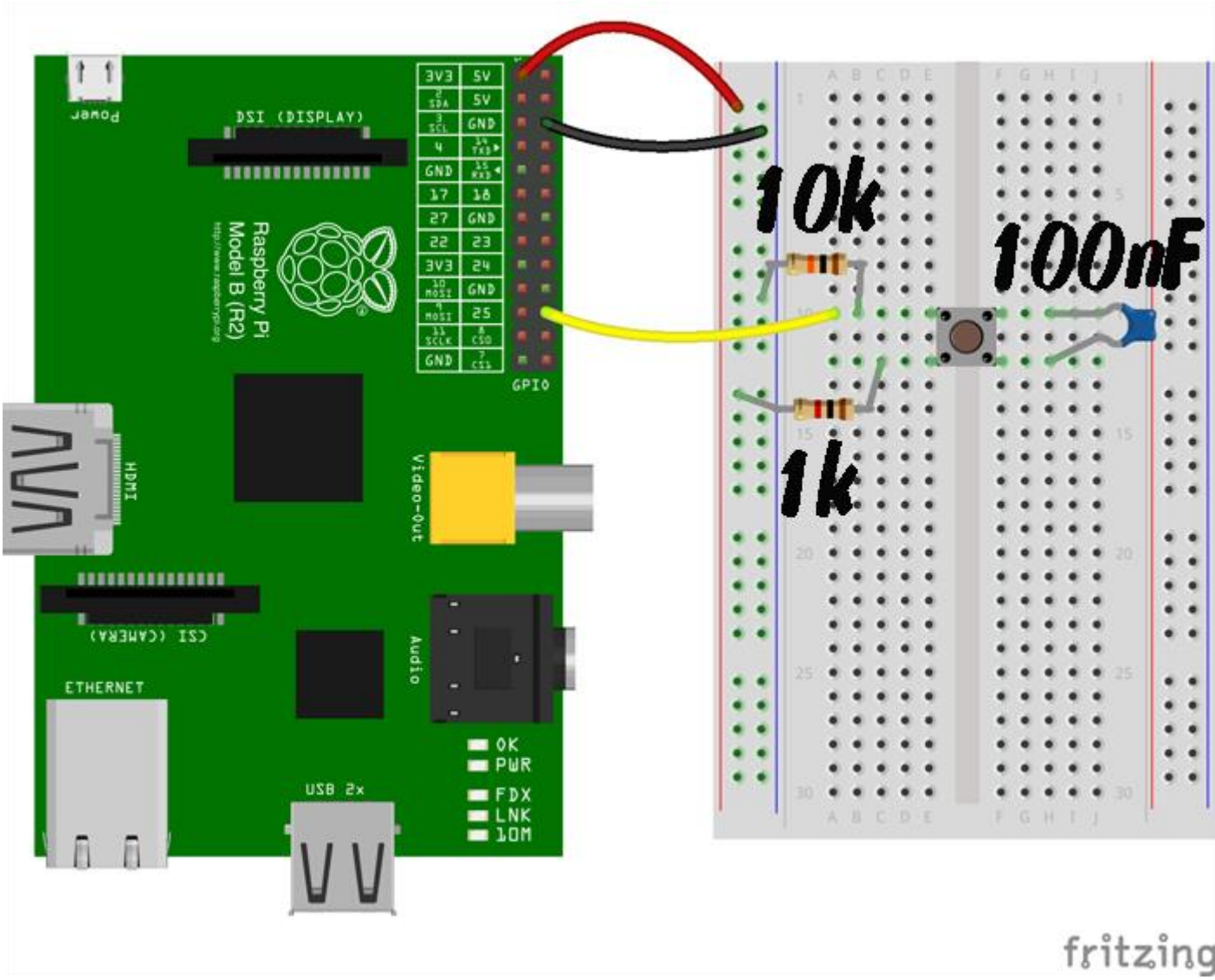
`GPIO.add_event_detect(25, GPIO.RISING, callback=my_callback)` for a rising edge
OR
`GPIO.add_event_detect(25, GPIO.FALLING, callback=my_callback)` for a falling edge
Now we can also detect edges in either changing direction, using BOTH…
`GPIO.add_event_detect(25, GPIO.BOTH, callback=my_callback)` for either rising or falling edges.

"Hold on", you say. "That's all very well, but how do I know whether it's FALLING or RISING?"

It's easy, when you think about it. Just read the port using GPIO.input(25) in the callback function. If it's 0/LOW/False, it was FALLING, and if it's 1/HIGH/True, it was RISING. As Andrew correctly points out in the comments, it's a good idea to have as little code as possible in your callback function to avoid missing any transitions while the code is executing.

That's a bit abstract, so to test it, we'll need a little circuit and some code.

## Let's Make a Circuit



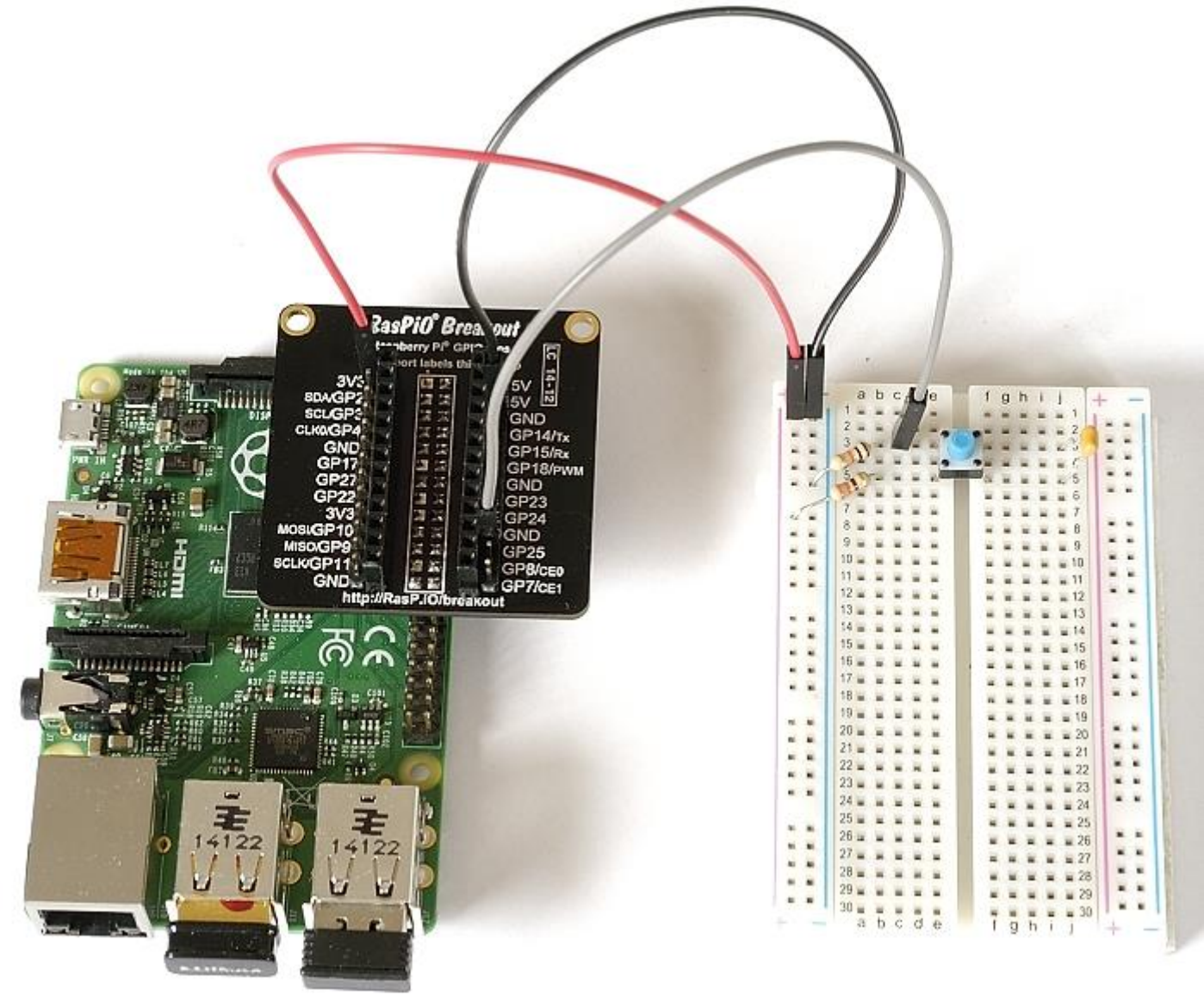

Circuit for testing GPIO.BOTH

The 10k resistor is a pull-down, to give the port a default status of 0/LOW/False.
The 1k resistor is to protect the port.
The 100 nF capacitor is needed in order to see the effect we're looking for. It debounces the switch in hardware because we've removed the 'bouncetime=300' from the 'GPIO.add_event_detect'. We did that so we can see what's happening in real-time.

When you press the button, 3V3 connects to GPIO 25 (through the 1k resistor) and causes a Rising edge on the port. If you keep the button pressed, nothing should happen. When you release the button, the 10k pull-down resistor will pull GPIO 25 back to GND and a Falling edge will result.

The program ends after 30 seconds.

Here's what my circuit looked like (I used a 330R instead of a 1k)…




Circuit for testing “BOTH”

```
1. #!/usr/bin/env python2.7
2. # demo of "BOTH" bi-directional edge detection
3. # script by Alex Eames https://raspi.tv
4. # https://raspi.tv/?p=6791
5.
6. import RPi.GPIO as GPIO
7. from time import sleep     # this lets us have a time delay (see line
    12)
8.
9. GPIO.setmode(GPIO.BCM)     # set up BCM GPIO numbering
10.      GPIO.setup(25, GPIO.IN)    # set GPIO25 as input (button)
11.
12.      # Define a threaded callback function to run in another thread
   when events are detected
13.      def my_callback(channel):
14.          if GPIO.input(25):     # if port 25 == 1
```

```
15.             print "Rising edge detected on 25"
16.         else:                   # if port 25 != 1
17.             print "Falling edge detected on 25"
18.
19.     # when a changing edge is detected on port 25, regardless of wh
   atever
20.     # else is happening in the program, the function my_callback wi
   ll be run
21.     GPIO.add_event_detect(25, GPIO.BOTH, callback=my_callback)
22.
23.     print "Program will finish after 30 seconds or if you press CTR
   L+C\n"
24.     print "Make sure you have a button connected, pulled down throu
   gh 10k resistor"
25.     print "to GND and wired so that when pressed it connects"
26.     print "GPIO port 25 (pin 22) to GND (pin 6) through a ~1k resis
   tor\n"
27.
28.     print "Also put a 100 nF capacitor across your switch for hardw
   are debouncing"
29.     print "This is necessary to see the effect we're looking for"
30.     raw_input("Press Enter when ready\n>")
31.
32.     try:
33.         print "When pressed, you'll see: Rising Edge detected on 25
   "
34.         print "When released, you'll see: Falling Edge detected on
   25"
35.         sleep(30)         # wait 30 seconds
36.         print "Time's up. Finished!"
37.
38.     finally:                   # this block will run no matter how
   the try block exits
39.         GPIO.cleanup()         # clean up after yourself
```