GPIO Inputs - LED Control via a Button and Interrupt

The previous example of using a button to light an LED, everything worked fine, but our processor was at 100%, this needs fixing, so in this example instead of constantly polling and setting GPIO pins, we are using a interrupt where by we do nothing until a button is actually pressed.

Create a new file in the normal way (you can download the python file

here

to save time)
<sxh [py][; options for SyntaxHighlighter]>

```
sudo nano LED-Button-i.py
```

```
#import modules
import RPi.GPIO as GPIO    # This imports the GPIO library that allows the use of the GPIO pins,
import time                # This imports the time library (for delays among other things)
                           # These libraries are built in to Raspbian.
GPIO.setmode (GPIO.BOARD)  # This sets the GPIO pin numbering. Our LED is connected to Pin 12,
                           # so we can reference it by using BOARD as pin 12. However there is
                           # another option (BCM) where we can reference a pin by it's name, pin
                           # 12 is called GPIO18 (a reference to its place on the chip).
GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) # setup GPIO Pin 11 as an input, and set
                                                    # the resistor to Pull Down (PUD_DOWN)
                                                    # this is the pin the button is connected to
                                                    # button is connected from pin 11 to the
                                                    # +3.3v pin on the GPIO
# this def buttonPressed needs to be defined before it can be reference in the GPIO.add_event_detect
def buttonPressed(channel):    # this is where our code will look when button is pressed
    print "Button is Pressed"  # print something to the screen
    GPIO.output(12,1)          # set GPIO pin 12 to high (3.3v) so LED will come on
# this is where we setup the GPIO input to use the event buttonPressed that was
# defined previously. bouncetime is a simple switch debouncer in mS.
GPIO.add_event_detect(11, GPIO.RISING, callback=buttonPressed, bouncetime=500)
GPIO.setup(12, GPIO.OUT)   # Sets the GPIO pin as output. This is connected to the LED, then
                           # from the LED to 0v via a 330 Ohm resistor.
GPIO.output(12, 0)         # sets the GPIO Pin 12 to low (so 0v)
try:
        while True:                             # start a loop
                time.sleep(1)                   # 1 second delay (or LED wont stay illuminated
                GPIO.output(12,0)               # Set PIN 12 to 0v so LED is off
except KeyboardInterrupt:                       # if Ctrl-C is pressed, exit loop
        GPIO.cleanup()                          # reset GPIO pins to default state
#End
```

</sxh>
Once you have your file saved, you can run it by using:

```
sudo python LED-Button-i.py
```

Example Output

Here you can see the code running.

640x360|autoplay,loop

The main advantage using the **interrupt** is that my processor occupancy on the Pi Zero is now around 10% normal, and 14% when I press the button. That's much better than the previous constant 100%.