Raspberry Pi GPIO Zero Reaction Timer

Tutorial by Andrew Oakley

www.cotswoldjam.org

Introduction

This Python programming tutorial, shows you how simple it is to use an LED light and a button, with the new GPIO Zero library.

Getting started

Conventions

```
Words you will see on the screen, or that you need to type in, are highlighted like this
```

At the end of each line, you will usually have to press the Enter key.

Your tutor should have prepared your Raspberry Pi for you and given you a bag of components. If not, see the section “Preparation” at the end.

The electronics

Components

An Light Emitting Diode (LED) is a light bulb with a short leg and a long leg. If you feel around the rim, you'll also find a flat edge. The short leg and flat edge always connect to negative (ground).

LEDs always need to be connected with a resistor. If you connect them without a resistor, they will burn out and probably won't work again.

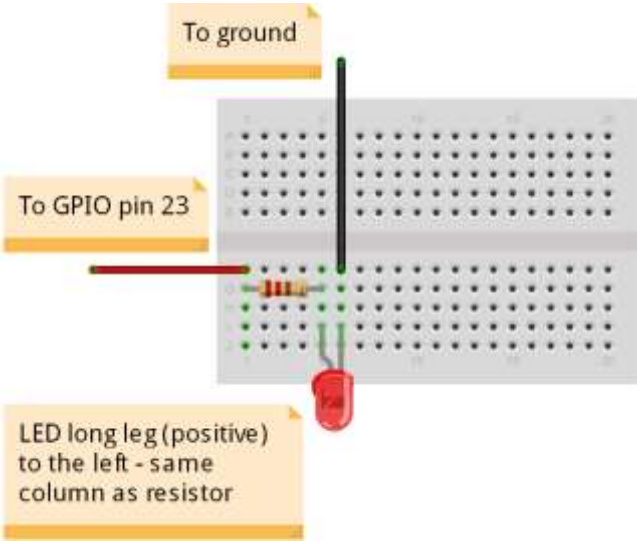
The resistor can be connected any way around. Lower value (lower Ohm) resistors will allow more current through and will make the LED brighter; higher values will let less current through and make it dimmer. We're using a 270 Ohm resistor but anything between 220-470 Ohms will work fine.

A switch can be connected any way around. When pressed, it allows electricity to flow; when released, the electricity stops. This kind of push-button is known as a “momentary press-to-make switch”.

You should also have a breadboard and some jumper wires. You should have three long male-to-female wires (with a sticky-out bit at one end, and a sticky-in bit at the other end), and one short male-to-male wire (with sticky-out bits at both ends).

Lighting the LED

The circuit



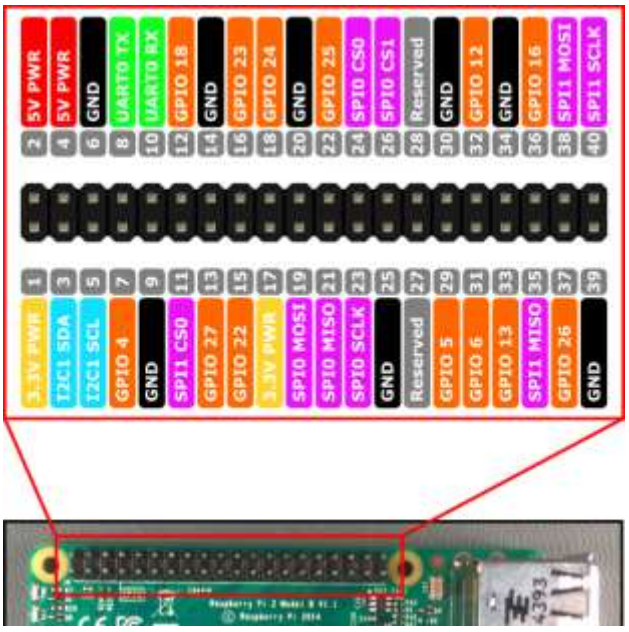
Connect up the breadboard as shown.

Breadboards allow us to quickly make electronic circuits. The board has a top section and a bottom section. All the vertical holes in each section are connected together, so if you plug things in in a vertical column, they'll make a connection.

Don’t worry about the colours of the wires – the author of this tutorial is colour-blind.

The Raspberry Pi uses two sets of numbers to refer to the General Purpose Input Output pins (GPIO pins).

"Board numbering" just starts from the bottom-left at 1, and works its way up and right, through to 40.



"BCM numbering" (Broadcom numbering) is the way the Raspberry Pi's processor sees the pin connections.

GPIO Zero uses BCM numbering.

Connect the short leg of the LED to a Ground connection.

Connect the resistor to the long leg of the LED.

Connect the other end of the resistor to GPIO 23

The program

Power up your Raspberry Pi and go to the desktop.

From the menu, select Programming - Python 3. Then use File, New Window to create a new program.

Type in the following program, or alternatively you can use File, Open to open the led.py program in the python/reaction folder.

```
from gpiozero import LED
from time import sleep

led=LED(23)
led.on()
sleep(2)
led.off()
```

Use File, Save to save this program as led.py and the run it by selecting Run menu, Run Module.

You should see the LED light up for two seconds, and then turn off.

Not working? Check you've got:

- The correct GPIO pins
- The LED is the correct way round; short leg to ground (GND)
- Components lined up on the correct vertical columns on the breadboard
- No spelling mistakes, no missing brackets, and you've used round brackets instead of pointy, curly or square ones.

What is this program doing?

```
from gpiozero import LED
from time import sleep
```

The first two lines tell the computer to learn about a new thing. Computers can learn from programs that other people have written, and we call these other programs "libraries". Our instructions tell the computer to learn about LEDs from the GPIOZero library, and learn about sleeping from the time library.

```
led=LED(23)
```

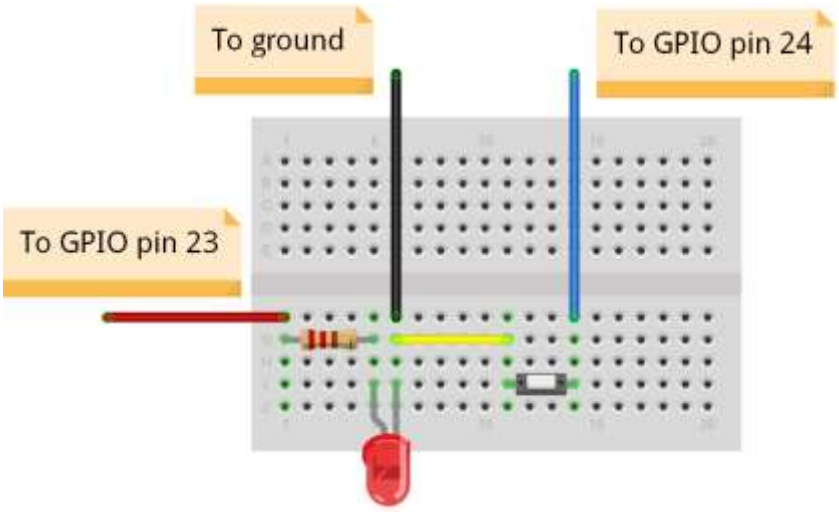
We then create an LED object. We say that the LED is connected to GPIO 23.

```
led.on()
sleep(2)
led.off()
```

Turn the LED on, wait for 2 seconds, then turn it off.

Detecting the button

The circuit



Now add the button and connecting wires.

The short jumper wire should go from the same column as the existing ground wire, to the column connecting to one of the button's legs.

The remaining long jumper wire should go from the column with the other leg of the button, to GPIO 24 on the Raspberry Pi.

The program

In Python 3, use File, New Window to create a new program. Type in the following program, or alternatively you can use File, Open to open the button.py program in the python/reaction folder.

```
from gpiozero import Button
from signal import pause

def press():
  print ( "Button PRESSED!" )

def unpress():
  print ( "Button unpressed!" )

button = Button(24)
button.when_released = unpress
button.when_pressed = press

pause()
```

Indentation matters in Python. For example, you must type the two spaces before the word “print”. Indentations like these *define* a block of code– hence the word “def”.

Use File, Save to save this program as button.py and the run it by selecting Run menu, Run Module.

Hold down the button. It should say "Button PRESSED!" in the Python 3 shell window.

Let go of the button. It should say "Button unpressed!".You can press and release the button many times.

Halt the program by selecting Shell menu, Restart shell.

What is this program doing?

```
def press():
  print ( "Button PRESSED!" )

def unpress():
  print ( "Button unpressed!" )
```

These are called functions. They define (def) something that should happen, and they give that action a name. The code is indented to show it belongs to the function. The press() function says "Button PRESSED!" and the unpress() function says "Button unpressed!".

```
button = Button(24)
button.when_released = unpress
button.when_pressed = press
```

Here we set up a button object and tell the computer that it is connected to GPIO 24.

We also tell the computer what to do, when the button is pressed or released.

```
pause()
```

This tells the computer to wait for events to happen, such as button presses.

Button and LED together

In Python 3, use File, New Window to create a new program.

Type in the following program, or alternatively you can use File, Open to open the led-button.py program in the python/reaction folder.

```
from gpiozero import Button, LED
from signal import pause

def press():
  print ( "Button PRESSED!" )
  led.on()

def unpress():
  print ( "Button unpressed!" )
  led.off()

led=LED(23)
led.off()
button=Button(24)
button.when_released=unpress
button.when_pressed=press

pause()
```

Use File, Save to save this program as led-button.py and the run it by selecting Run menu, Run Module.

Press the button and the LED should light up; release the button and it should turn off.

Halt the program by selecting Shell menu, Restart shell.

Reaction timer

In Python 3, use File, New Window to create a new program.

Type in the following program, or alternatively you can use File, Open to open the reaction.py program in the python/reaction folder.

```
from gpiozero import LED, Button
from signal import pause
from random import uniform
from time import time, sleep
timestarted=None

def press():
  global timestarted,led,button
  print ( "Reaction time: %.3f" % (time()-timestarted) )
  led.off()
  sleep(2)
  reset()

def reset():
  global timestarted,led
  sleep(uniform(0.5,3))
  led.blink(0.1,0.4,4,False)
  led.on()
  timestarted=time()

led=LED(23)
led.off()
button=Button(24)
button.when_pressed=press
reset()

pause()
```

Use File, Save to save this program as reaction.py and the run it by selecting Run menu, Run Module.

The LED will flash four times, then on the fifth time it will stay on until you press the button.

Try to press the button as quickly as possible when it comes on for the fifth time.

You will be shown your reaction time in seconds.

You can have as many goes as you like - try challenging a friend to see who has the fastest reactions. When you've had enough, halt the program by selecting Shell menu, Restart shell.

What is this program doing?

We've introduced several new concepts in the reaction program.

We're using a value which we call `timestarted` to record when the fifth flash starts. We can then compare it to the time when the user presses the button, and that will tell us the reaction time.

Values which are given a name, like `timestarted` , are called **variables**, because the actual value they contain can vary.

Some variables are **global**. This means they can be used in any part of the program.

```
  print ( "Reaction time: %.3f" % (time()-timestarted) )
```

%.3f tells the computer to print a decimal fraction with 3 decimal places after the decimal point.

The number we're going to print is the value of time now, minus the value of time when we started the final flash. This is measured in seconds.

```
  sleep(uniform(0.5,3))
```

The `uniform` command is used here to pick a random number between 0.5 and 3. It will probably pick a decimal fraction rather than a whole number. The command is called uniform because it you call it enough times, you will get uniform random distributon of numbers - you'll eventually get every possible number in that range.

```
  led.blink(0.1,0.4,4,False)
```

This command tells the LED to blink for 0.1 of a second, then wait for 0.4 of a second, then repeat that 4 times.

The last value, False, tells it to wait until it has finished before moving on to the next command. If you put True then it would start executing the next commands whilst carrying on blinking! We don't want that here.

You can find more about GPIO Zero commands at:

http://pythonhosted.org/gpiozero

Preparation

You will need the following electronic components for this tutorial:

1x Breadboard – mini 170 point
1x LED – 5mm diffused through-hole, red
1x Resistor – carbon film 270 Ohm through-hole
1x Switch – 2 pin tactile push-button through-hole

3x Male to Female jumper wires - 20cm
1x Male to Male jumper wire – 10cm

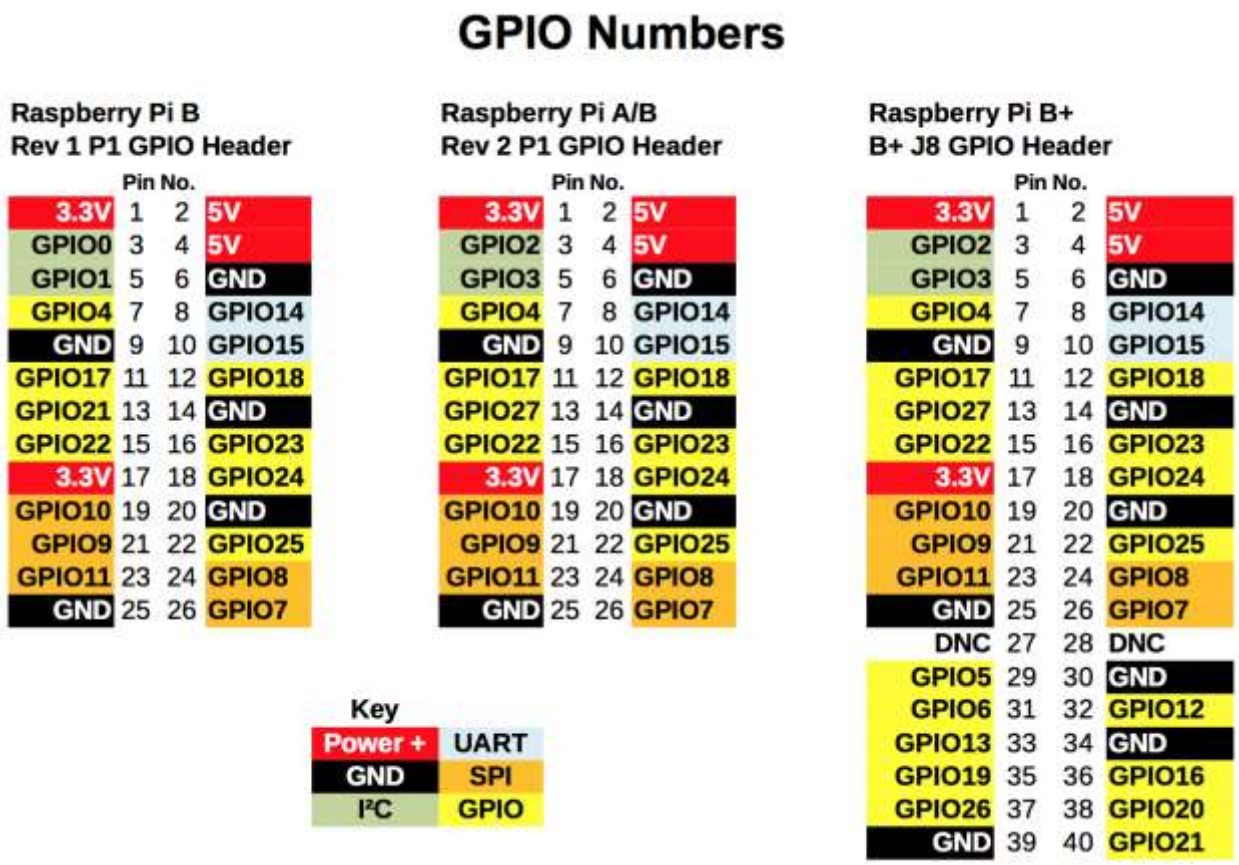
The files for this tutorial can be found at:

http://www.cotswoldjam.org/downloads/2016-01/

You will need Raspbian Jessie with Python 3, IDLE and GPIO Zero. Once you have Raspbian Jessie and an internet connection working, you can ensure all the other files are installed by going to Menu, Accessories, Terminal and typing:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo apt-get install python3-gpiozero wget
mkdir -p ~/python
cd python
wget http://www.cotswoldjam.org/downloads/2016-01/reaction-timer-code.zip
unzip reaction-timer-code.zip
exit
```

The dist-upgrade may take a while – maybe half an hour if your system hasn't been updated recently. Don't worry if apt-get tells you that you already have packages installed.



This useful GPIO guide is courtesy of Alex Eames http://raspi.tv/rpi-gpio