Raspberry Pi®

CoderDojo Belfast

Python Pi Project #5:Big Red Button

Level: Middling

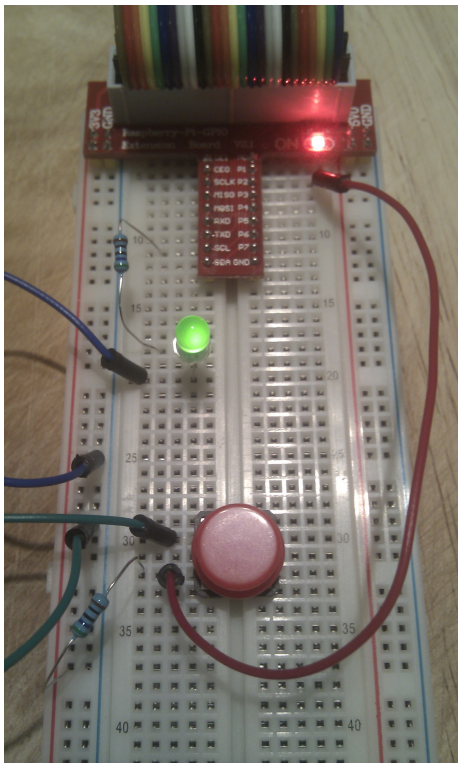
In our “LED” project we used the Pi to make the LED flash – that's a real-world *output* from the Pi. Let's add some *input* – we'll press a Big Red Button to make the LED flash.

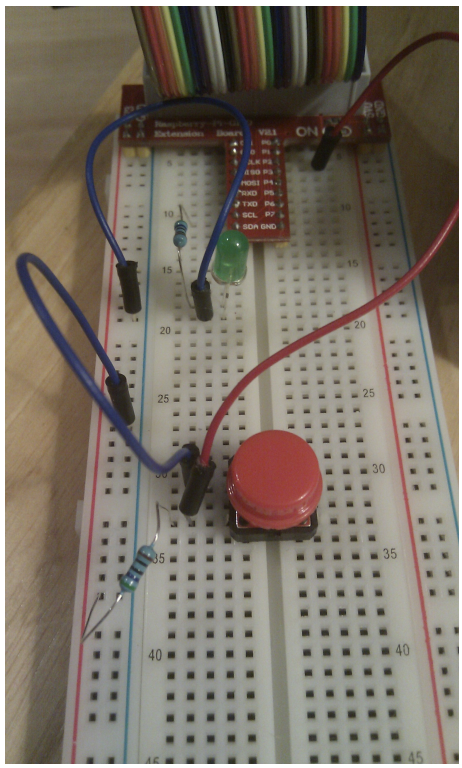
Hacking skills needed

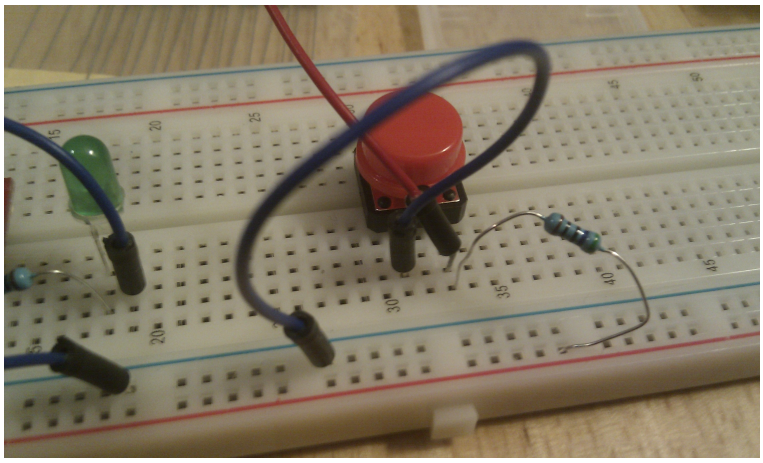
- ➔ wiring components on a breadboard
- ➔ using the GPIO module

We'll need to add some more bits to our electronic circuit. The photos below show how to wire it up.

We'll connect pin 12 from the Pi (P1 on the T connector) to the button, and we'll also connect the same foot of the button to 'live' (the red row on the board), but through a 'pull-up' resistor. (You need the resistor if you want the button to give a clear 'off/on' signal.) Then we connect the other foot of the button to 'ground' (the blue row on the board).







Now, how does the code work? The Pi gives us a way to ask the button whether it's up or down:

```
GPIO.input(pin_button)
```

The result will be 1 or 0. (Challenge: which means up and which means down? Can you write a little test with "while True" to find out?)

But we want the button to act like a light switch – one press for 'on' and then another for 'off'.

What we'll do is write a loop that constantly asks the button whether it's up or down, and remembers the answer in a variable called 'prev' (for 'previous').

Also, each time round the loop, we'll check whether the button is now 'up', but was 'down' the last time we looked. That will only happen when we've just finished pressing the button:

```
    if (0 == prev) and (1 == state) :
```

If we see that happening, we'll switch the light on (if it's off) or off (if it's on). The jargon for this is to say we'll 'toggle' the light – switch it to the opposite state from what it's in now.

We'll use some functions to help keep this tidy – remember you saw functions in Project 2, to build a house.

We'll have one function to set the light either on or off:

```
def setLight(state):
    GPIO.output(pin_led, state)
    global led_state
    led_state = state
```

This uses a global variable 'led_state' to remember whether the light is on or off. (Global variables tend to mess things up and there are better ways to do this, but that's for another time.) The function also shows how to do output, by using GPIO.output and telling it the pin number to send the output on, and what value to send.

Now we can write our 'toggle' function very simply:

```
def toggle() :
    setLight(not(led_state))
```

The 'not()' is a logic operation, which gives the opposite logical value to its input – 1 for 0, or 0 for 1. (Python thinks 0 is false, and other numbers are true!) In other words our toggler just switches the light to do the opposite of what it's doing now.

We'll also use a function to set up the GPIO board, and a few more global variables.

The complete program - press.py

```
#!/usr/bin/env python

import RPi.GPIO as GPIO
from time import *

ON = True
OFF = False
pin_button = 12
pin_led = 8
led_state = None

def setLight(state):
    GPIO.output(pin_led, state)
    global led_state
    led_state = state

def setup() :
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(pin_button, GPIO.IN)
    GPIO.setup(pin_led, GPIO.OUT)
    setLight(OFF)

def toggle() :
    setLight(not(led_state))


setup()
prev = -1

while True:
    state = GPIO.input(pin_button)
    if (0 == prev) and (1 == state) :
        toggle()
    prev = state
    sleep(0.1)
```

Dojo Challenge:

What happens if you don't have the resistor? (Hint: take away the resistor and use the little 'challenge' test I mentioned above to see what the button does now.) Can you see why the resistor is needed?

Now you can do anything you can think of at the press of a button. How about modifying the 'Teleport' program from Project 1 to teleport you at the press of a button?