

## CamJam EduKit Worksheet Five

**Project** Push button for physical input.

**Description** In this project, you will learn how to wire and program a push button for physical input.

*NOTE: This worksheet can use a Raspberry Pi Model A, B or B+. The first 26 pins on the B+ (when looking at the B+ with the pins in the top left) are exactly the same as the Model A and Model B.*

## Equipment Required

The circuit built in CamJam EduKit Worksheet Two, plus the following:

☐ Push button      ☐ 1 x M/F Jumper Wires      ☐ 1 x M/M Jumper Wires      ☐ 4.7k  $\Omega$  Resistor

## Additional Parts

You will be adding a switch to the LED circuit that you made in CamJam EduKit Worksheet Two. Here are the additional components. You may skip this section if you already know about these components.

### Push Button



A push button will complete a circuit when the button is pressed. What that means is that a current will not flow across the button until it is pressed. When it is released, the circuit will be 'broken'.

### 4.7k $\Omega$ Resistor



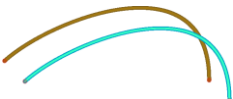
The resistor in this circuit has a special purpose.

You are going to use one of the GPIO pins as an input pin, meaning that it will react to the outside world. If you try to detect the state of one of the GPIO input pins it will randomly be either on or off, so you must have a way to ensure that it is on or off in a reliable way.

For this, you are going to use what is called a 'pull-up resistor'. When the button is not pressed, there will be a current flowing from the 3.3v power supplied by the Pi to the input pin of the Pi. When read, the input pin will be seen as being 'on'.

When you press the switch, another circuit is made which will make the current 'flow to ground', which means that the input pin will be seen as being 'off'. This is how you will detect the switch.

### Jumpers



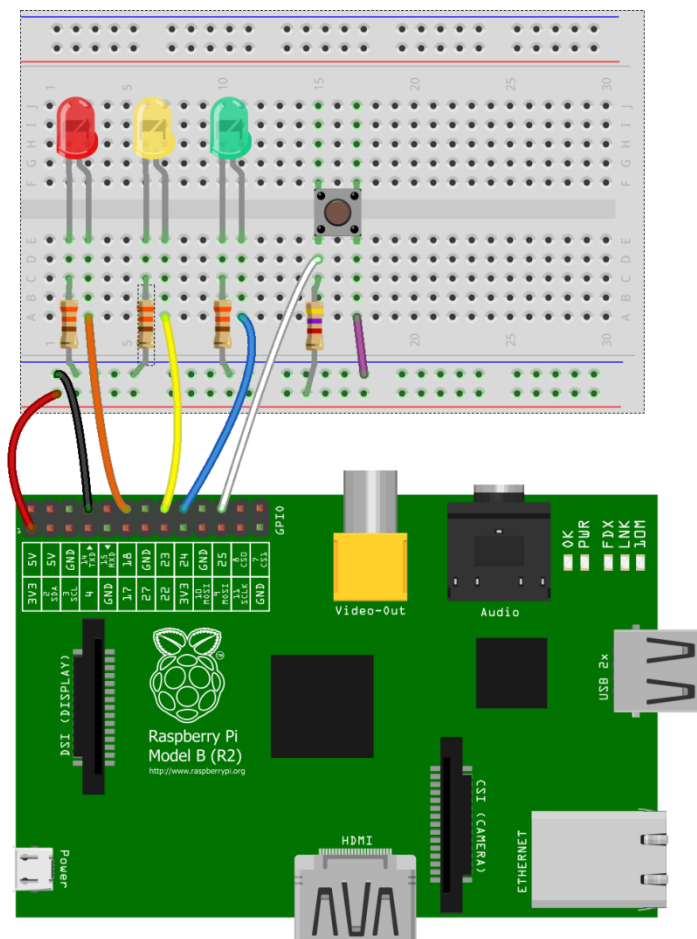
You will be adding one more Male/Female jumper wire and one Male/Male (M/M) jumper, which have a pin on each end. The M/M jumper will be used for joining two rows or columns on the breadboard together.

## Additional Parts

The colour jumper wires supplied in the EduKit will vary, and are unlikely to match the colours used in the diagrams.

## Building the Circuit

Before you connect additional components to your circuit, you should turn off your Pi. Leave the LED circuit from the previous two worksheets in place.



fritzing

Add the button to the breadboard, with the pins straddling the split in the middle of the board. It will only fit one-way round. Push down hard to ensure that the pins are inserted fully into the breadboard.

Connect one of the jumpers (shown in red here) from the 3.3v pin of the Raspberry Pi to the horizontal rail just above the red line.

Connect the 4.7kΩ resistor between the same horizontal rail and one side of the button.

Then connect another jumper (shown in white) from the same column of the breadboard as the resistor to the input pin you are going to use on the Raspberry Pi. You are using pin 25.

Finally, connect the 'ground rail' to the other side of the switch, shown here by the purple wire.

So, how does the switch work? When the switch is not being pressed, a current will flow between the 3.3v pin of the Pi, through the red wire, then the resistor and to the input GPIO pin of the Pi through the white wire.

When the switch is pressed, the current is able to take another route as well. It also goes

from the resistor, through the switch and back to ground (0v). This splits enough of the current off that the voltage across the white wire is much lower, and no longer high enough for the input GPIO pin to be seen as 'on'.

## Code

You are going to save this code into the 'EduKit' directory again.

1. Change to the directory:

```
cd ~/EduKit/
```

2. Create a new file by typing

```
nano 5-button.py
```

3. Type in the following code exactly as seen. In Python, the indentation is very important.

```
# Import Libraries
import os                                #Gives Python access to Linux commands
import time                              #Proves time related commands
import RPi.GPIO as GPIO                 #Gives Python access to the GPIO pins

GPIO.setmode(GPIO.BCM)                  #Set the GPIO pin naming mode
GPIO.setwarnings(False)                 #Supress warnings

# Set pin 25 as an input pin
ButtonPin = 25
GPIO.setup(ButtonPin, GPIO.IN)

print "-----"
print " Button + GPIO "
print "-----"

print GPIO.input(ButtonPin)

# The commands indented after this 'while' will be repeated
# forever or until 'Ctrl+c' is pressed.
while True:
    #If the button is pressed, ButtonPin will be 'false'
    if GPIO.input(ButtonPin) == False:
        print "Button Pressed"
        print GPIO.input(ButtonPin)
        time.sleep(1) #Sleep for 1 second
    else:
        os.system('clear') #Clears the screen
        print "Waiting for you to press a button"

        time.sleep(0.5) #Sleep for 0.5 seconds
```

Once complete use "Ctrl + x" then "y" then "enter" to save the file.

## Explanation

What is happening in the code? Let us go through some of the important concepts before looking at the code:

- A 'variable' is a name that contains a value. That value can be changed within your code at any time. It is often easier to use a variable to contain a number because it is easier to remember the name.
- 'Constants' are similar to variables, except that they cannot be changed within your code.
- In Python there are two special 'constants' called 'True' and 'False'. They have the values '-1' and '0'.
- Indentation is very important in Python. It is used to group commands together after certain other commands, like 'if' and 'while'. Everything at the same indentation distance after these commands will be grouped together.

```
#Set pin 25 as an input pin
ButtonPin = 25
GPIO.setup(ButtonPin, GPIO.IN)
```

Sets up a 'variable' that holds the pin number (25) and then sets it as an 'input' pin. This means that the status of the pin can be read.

When the pin is receiving a voltage nearing 3.3v, it is read as 'true', or 'on'. If the voltage nears 0v, then it is 'false', or 'off'.

```
while True:
```

The commands indented after this 'while' will be repeated forever, or until 'Ctrl+c' is pressed.

```
    if GPIO.input(ButtonPin) == False:
```

The 'if' statement gets the value of the input pin. The value is either:

- 'True', meaning 'on', or
- 'False', meaning 'off'

If it is 'False', then the button is being pressed. Everything indented after the 'if' statement will be run.

```
    else:
```

If the value of the input button is not 'false', then different code is run.

## Running the Code

To run this code type:

```
sudo python 5-button.py
```

The program will continue to run waiting for the button to be pressed and reporting the status to the screen. This will continue until you press 'Ctrl + c' to exit Python.

Remember to use `nano` if you need to edit the code.