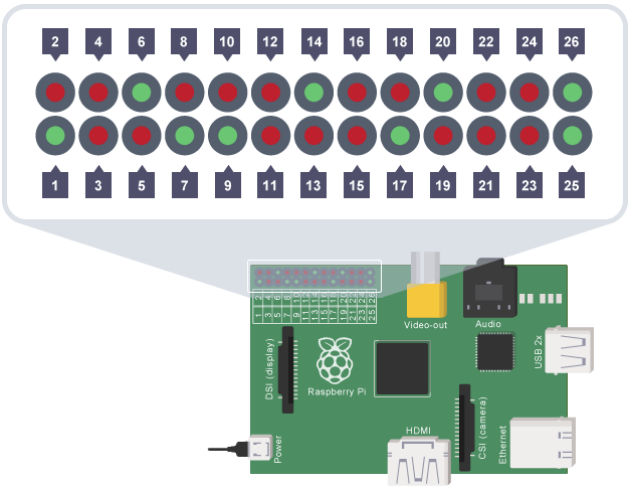
Bitesize

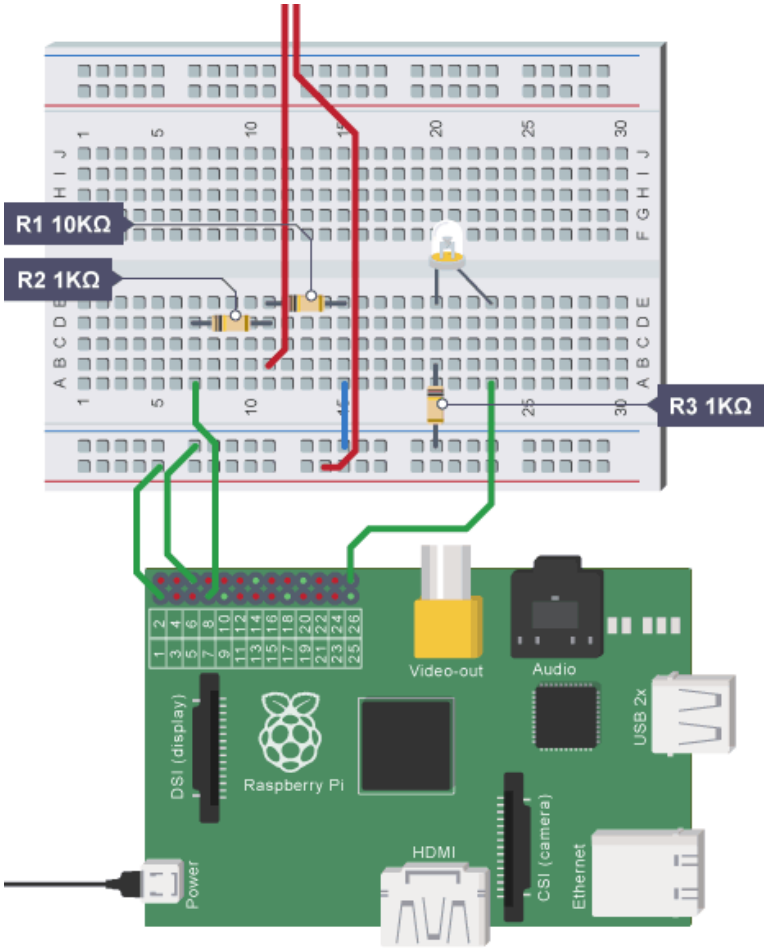
GCSE Computer Science

Raspberry Pi programming exercise: LED light button

This guide will show how to hook up a simple push button switch and an LED to the GPIO pins on a Raspberry Pi. The button is going to toggle the LED on and off.



The Raspberry Pi needs to be connected as shown in the diagram below:



Equipment

- a Raspberry Pi already loaded with an operating system and connected to a monitor (via the HDMI port), keyboard and mouse (through the USB port)
- a breadboard (the white rectangle board shown in the diagram above)
- four male-to-female jumper wires and one male to male jumper wire
    - one male to female jumper wire of length approximately 10 cm (green wires in the diagram)
    - three male to female jumper wires of length approximately 5 cm (green wires in the diagram)
    - one male to male jumper wire of length approximately 1 cm (blue wire in the diagram)
- two 1 K ohm resistors
- one 10 K ohm resistor
- one push button switch – (if you don't have a switch, two wires can be used instead, as shown by the red wires in the diagram)
- one LED

Method

1. Connect the Raspberry Pi to the breadboard.
    a. Connect the Raspberry Pi to the breadboard using the wires, as follows:

Wire	**Raspberry pin**	**Breadboard point (column, row)**
10 cm jumper	GPIO 1	Positive, 5
5 cm jumper	GPIO 6	Negative, 7
5 cm jumper	GPIO 26	A,23
5 cm jumper	GPIO 7	A,7

    b. Connect the resistors to the breadboard, as follows:

Resistor	**Breadboard point A (column, row)**	**Breadboard point B (column, row)**
10 k ohm	E,11	E,15
1 k ohm	D,7	D,11
1 k ohm	B,20	Negative, 20

    c. Connect the red switch wires to the breadboard:
        i. one in the positive column, row 14, the other end left unconnected.
        ii. one in B,11, the other end left unconnected.

    d. Connect the LED to the breadboard:
        i. the shortest straight leg in E,20.
        ii. the longer leg in E,23.
    e. Connect the 1 cm male to male jumper wire from A,15 to negative, 15.

2. Turn on the computer:
    a. Plug in the power supply to switch on the Raspberry Pi.
    b. You might need to type **startx** at the Command Line to start a Graphical User Interface or it might do so automatically.

3. Write the program code:
    a. Once you can see the desktop, double-click on the LXTerminal icon to start a Command Line (or Terminal).
    b. At the flashing cursor type in: **sudo idle3 &**
    c. Hit the enter key.

       This uses the **sudo** command to tell the computer we want to do this using the special **root** user's permissions. Only the root user has access to the GPIO pins from inside the Operating System. **Idle3** is the name of the Python Shell tool and the ampersand **&** is used so that once we have started idle3 we can safely close the Terminal window without it also closing idle3.

    d. When we see the Python Shell, at the command prompt type **import RPi.GPIO as GPIO**
    e. Hit the enter key.

       If the command works and you are returned to the >>> prompt without an error message, then you have succeeded in starting idle 3 correctly.

    f. Go to the **File Menu** at the top left of the Python Shell's window and select **New Window**.

       This will open a new window which is an editor window where we can edit the code. We are going to enter and save it.

    g. Enter the following Python Code, exactly as shown, and save it in your home directory as switched.py:

       NOTE: Any line that begins with **#** is a comment to explain to you what is going on and does not need to be typed in.

File Edit Format Run Options Windows Help

```
#Get access to the GPIO pins
import RPi.GPIO as GPIO

#Tell Python which numbering scheme we want to use
GPIO.setmode(GPIO.BOARD)

# Set pin 7 to be an input and pin 26 to be an output
GPIO.setup(7, GPIO.IN)
GPIO.setup(26, GPIO.OUT)

#Tell the user what to do on the first go
print("Push the button")

#Create a variable to hold the curent state of the LED
lightOn = False

#Begin an infinite loop which will contain most of our program
while True:
    #Every time we loop we check if the button has been pressed
    #if it has then Pin 7 will go "high" which gives us the True response
    if GPIO.input(7) == True:
        #Tell us it has been pushed
        print("Button pressed")
        #Change the value of lightOn to the opposite of its' current value
        lightOn = not lightOn
        #and apply it to the physical LED
        GPIO.output(26, lightOn)
        #Now check whether we want to continue or quit
        command = input("Press enter to go again or q to quit: ")
        #If we entered q then we want to quit
        if command == "q":
            #If we are quitting then break out of the infinite loop
            break
        else:
            #Tell the user what to do
            print("Push the button")

#When we quit it is good practice to clean up after ourselves
GPIO.cleanup()
```

4. You can run the program in two ways - either go to the **Run Menu** and select **Run Module** or hit **F5**. If anything still needs saving it will ask you to do so, then it will bring up the Python Shell with the message: **Push the button**.

5. Pushing the button should turn the LED on, and print a new message. To continue switching the LED on and off keep pushing the button, otherwise type **q** and hit **Enter** to quit the program.