

YouTube

Workshop Kit v2.0

Tutorial 2/9: LED On & Off

Contents

Things you will need	3
Prerequisites	3
Wiring Diagram	4
Introduction	5
Getting Started.....	5
Save the Program.....	6
Writing the Code	7
Display Variables	7
Setting up the GPIO.....	7
Update Display Method	7
The Action Code – LED On	8
The Action Code – LED Off	8
Running the Program	8
Results	9
Code on GitHub.....	9
Thanks	9

Things you will need

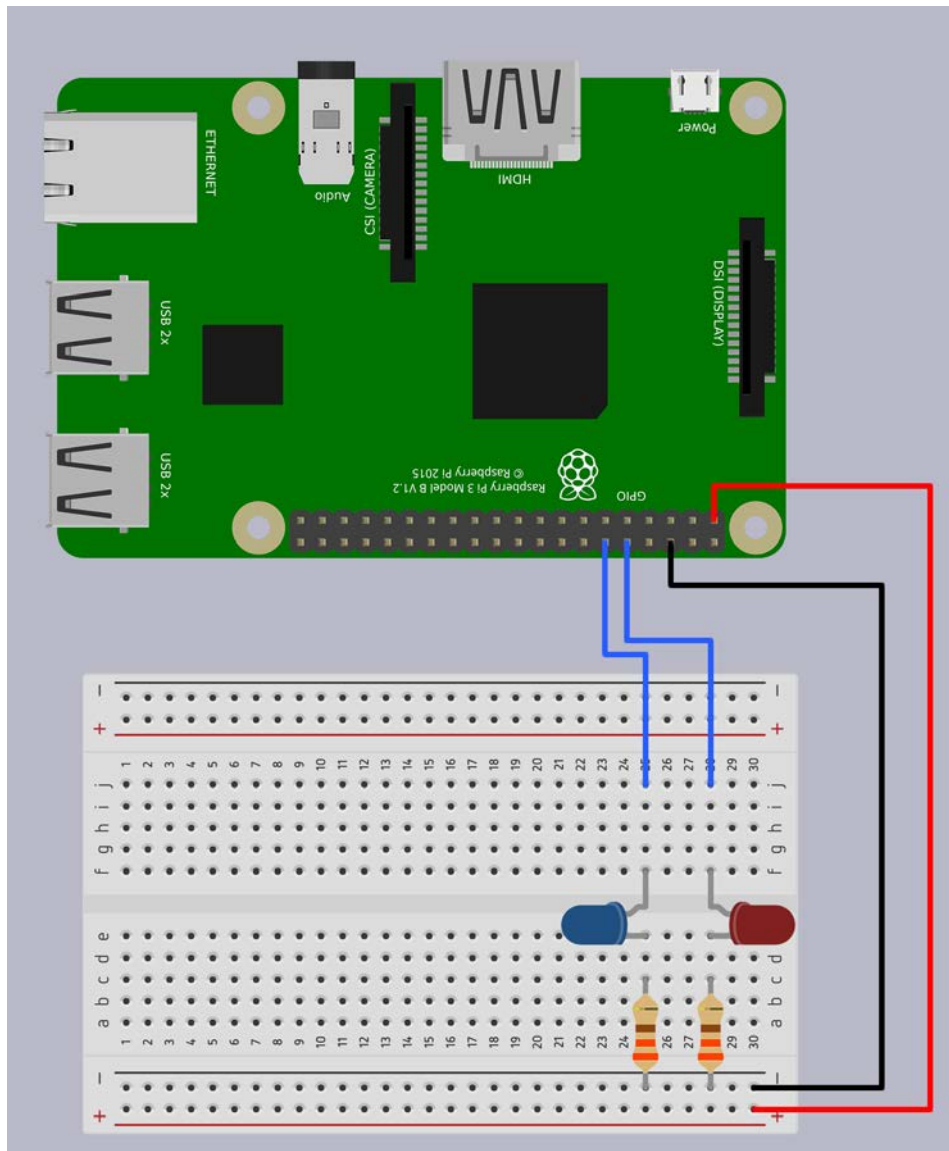
Raspberry Pi 3 Model B
Class 10 Micro SD Card
Keyboard + Mouse
Monitor + HDMI Cable
Power Supply (Recommended: 5V 2.5A)
Breadboard
1x Red LED
1x Blue LED
2x 330Ω Resistor
3x M/F Jumper Wire

If you are connecting to your Raspberry Pi remotely using VNC or other means then Keyboard, mouse Monitor and HDMI cable are optional.

Prerequisites

You will need to install the latest version of Raspbian on to your Micro SD Card. Initial setup will require a keyboard, mouse, HDMI cable and Monitor/TV.

Wiring Diagram

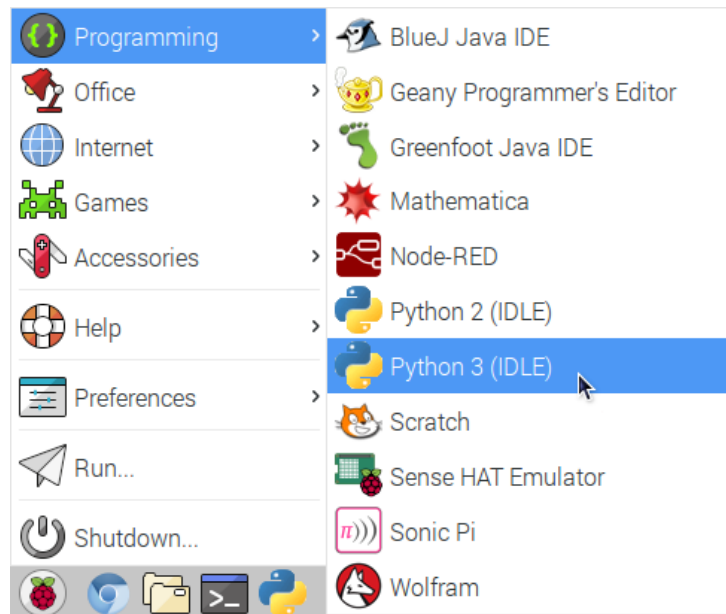


Introduction

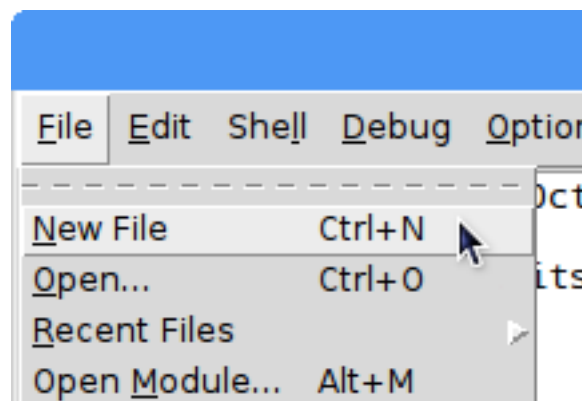
In this tutorial we will be making a Light Emitting Diode (LED) turn on and off. These tutorials are written on a Raspberry Pi 3 Model B with a clean install of Rasbian 4.4

Getting Started

To get started, first you need to open Python 3 (IDLE). To do this click on the Raspberry Pi icon on the task bar, highlight “Programming” then click on “Python 3 (IDLE)”



When IDLE has loaded, you will want to start working on a new file. You can do this by clicking on File and select “New File” or by pressing Ctrl+N

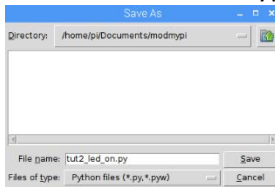


Save the Program

Now that we have IDLE running, first save a new file. First open up the File Manager by clicking on this icon on the taskbar and open the Documents folder.



In here, create a new folder, right click select “Create New” and click Folder, or press Ctrl+Shift+N. Call the folder modmypi and click ok.



Go back to the IDLE window and click on File and select “Save As”. Navigate to /home/pi/Documents/modmypi and enter tut2_led_on.py for the filename then click Save.

Writing the Code

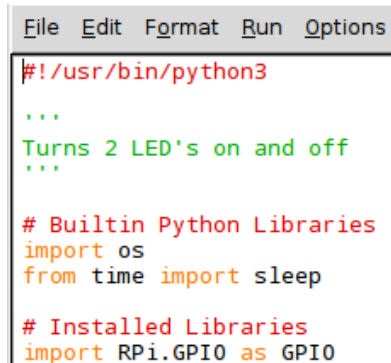
The first thing you should type is a shebang line, docstring and import your modules

```
#!/usr/bin/python3

'''
Turns 2 LED's on and off
'''

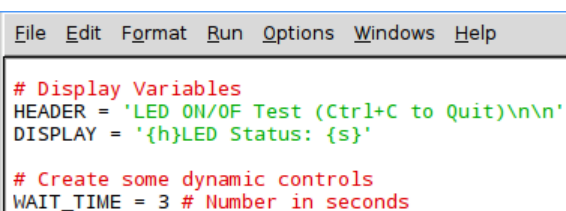
# Builtin Python Libraries
import os
from time import sleep

# Installed Libraries
import RPi.GPIO as GPIO
```

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options) and a text area containing the same code as the previous block, from the shebang line to the GPIO import.

Display Variables

Now you need to create some variables, 2 string variables and 1 integer variable. The string variables will be used to display text in the terminal to let us know what the program is doing. The integer variable will be used for a sleep timer.

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a text area showing the variable definitions for display and wait time.

```
# Display Variables
HEADER = 'LED ON/OFF Test (Ctrl+C to
Quit)\n\n'
DISPLAY = '{h}LED Status: {s}'

# Create some dynamic controls
WAIT_TIME = 3 # Number in seconds
```

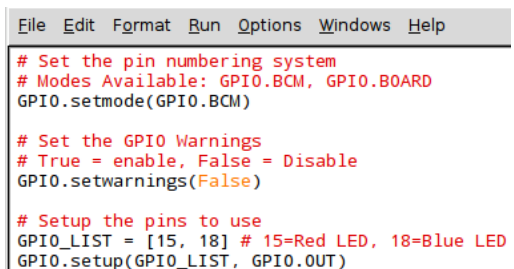
Setting up the GPIO

Time to setup the GPIO. We will set mode and warnings and also which pins that you will be using.

```
# Set the pin numbering system
# Modes Available: GPIO.BCM, GPIO.BOARD
GPIO.setmode(GPIO.BCM)

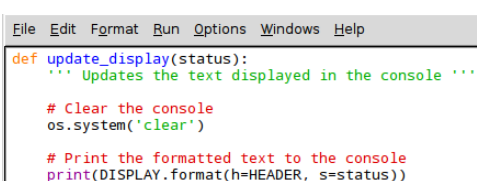
# Set the GPIO Warnings
# True = enable, False = Disable
GPIO.setwarnings(False)

# Setup the pins to use
GPIO_LIST = [15, 18] # 15=Red LED, 18=Blue LED
GPIO.setup(GPIO_LIST, GPIO.OUT)
```

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a text area showing the GPIO setup code.

Update Display Method

Next, it is time to create a Method that will update the text that will be displayed in the console. This will be called when you want to update the information in the console.

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options, Windows, Help) and a text area showing the definition of the update_display method.

```
def update_display(status):
    ''' Updates the text displayed in the console '''

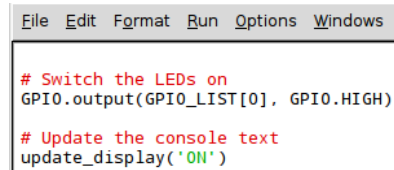
    # Clear the console
    os.system('clear')
    # Print the formatted text to the console
    print(DISPLAY.format(h=HEADER, s=status))
```

The Action Code – LED On

You will now need to add is to tell the program to turn on an LED and update the console text.

```
# Switch the LEDs on
GPIO.output(GPIO_LIST[0], GPIO.HIGH)

# Update the console text
update_display('ON')
```

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options, Windows). The code inside is:

```
# Switch the LEDs on
GPIO.output(GPIO_LIST[0], GPIO.HIGH)

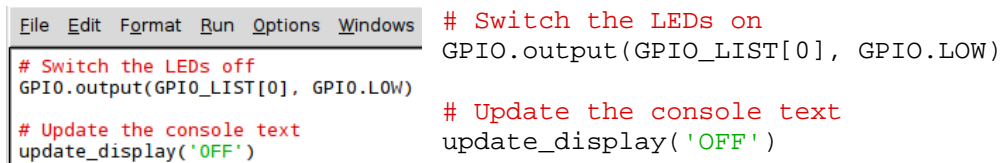
# Update the console text
update_display('ON')
```

If you would like to turn on the Blue LED instead of the red replace GPIO_LIST[0] with GPIO_LIST[1].
If you would like to turn on both LED's at the same time, replace GPIO_LIST[0] with GPIO_LIST

Make sure to save your work by clicking File and select Save, or press Ctrl+S

The Action Code – LED Off

The last thing you will need to do is save a new file like before and call it tut2_led_on.py. For this it is a simple modification to turn the LED off and update the console text.

A screenshot of a code editor window with a menu bar (File, Edit, Format, Run, Options, Windows). The code inside is:

```
# Switch the LEDs off
GPIO.output(GPIO_LIST[0], GPIO.LOW)

# Update the console text
update_display('OFF')
```

If you would like to turn on the Blue LED instead of the red replace GPIO_LIST[0] with GPIO_LIST[1].
If you would like to turn on both LED's at the same time, replace GPIO_LIST[0] with GPIO_LIST

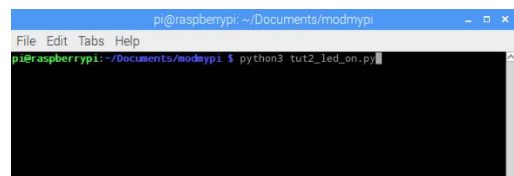
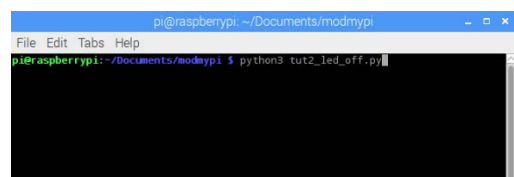
Note: make sure you use the same LED to turn off. If you try turn off the Blue LED (GPIO_LIST[1]) when you have turned on the Red LED (GPIO_LIST[0]) then when you run the LED Off program, it will appear to do nothing.

Running the Program

Save your work and it is time to run it so that you can make sure that it works as it should. Go back to the File Manager and open the modmypi folder you created. Next click on tools and select "Open Current Folder in Terminal" or press F4.

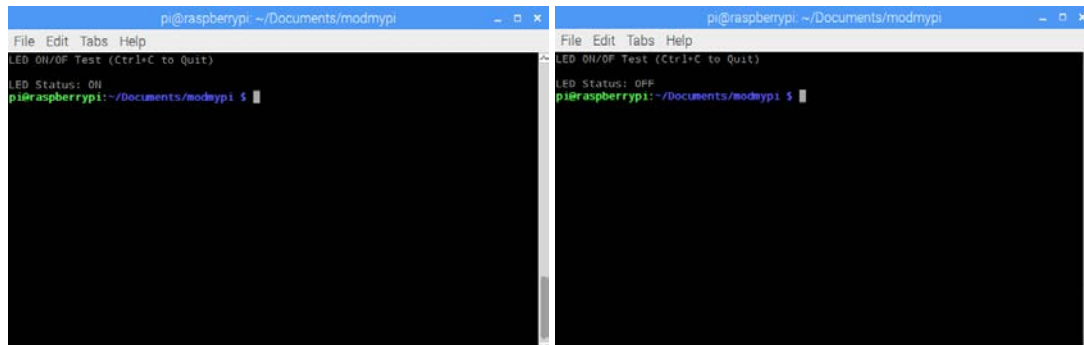
In the terminal, type python3 tut2_led_on.py and press enter

When that has complete, run the other file
tut2_led_off.py

A screenshot of a terminal window titled 'pi@raspberrypi: ~/Documents/modmypi'. The command 'python3 tut2_led_on.py' has been entered at the prompt.A screenshot of a terminal window titled 'pi@raspberrypi: ~/Documents/modmypi'. The command 'python3 tut2_led_off.py' has been entered at the prompt.

Results

If everything is working correctly you should see the LED(s) turn on when you run `tut2_led_on.py` and the LED will turn off when you run `tut2_led_off.py`. In the console you should see something like



The image shows two terminal windows side-by-side. Both windows have a title bar that reads 'pi@raspberrypi: ~/Documents/modmypi'. The left window's content shows 'LED ON/OFF Test (Ctrl+C to Quit)' followed by 'LED Status: ON' and a prompt 'pi@raspberrypi:~/Documents/modmypi \$'. The right window's content shows 'LED ON/OFF Test (Ctrl+C to Quit)' followed by 'LED Status: OFF' and a prompt 'pi@raspberrypi:~/Documents/modmypi \$'.

Code on GitHub

If you would like to download a copy of the code, you can download it from along with all the other tutorials, code and wiring diagrams from [GitHub here](#)

Thanks

Thank you for taking the time to follow this tutorial and hope that you have found this useful. Please feel free to follow the other tutorials that have been created for the ModMyPi YouTube Workshop Kit.