MODMYPI

YouTube

Workshop Kit v2.0

Tutorial 4/9: Push Button

Contents

Things you will need

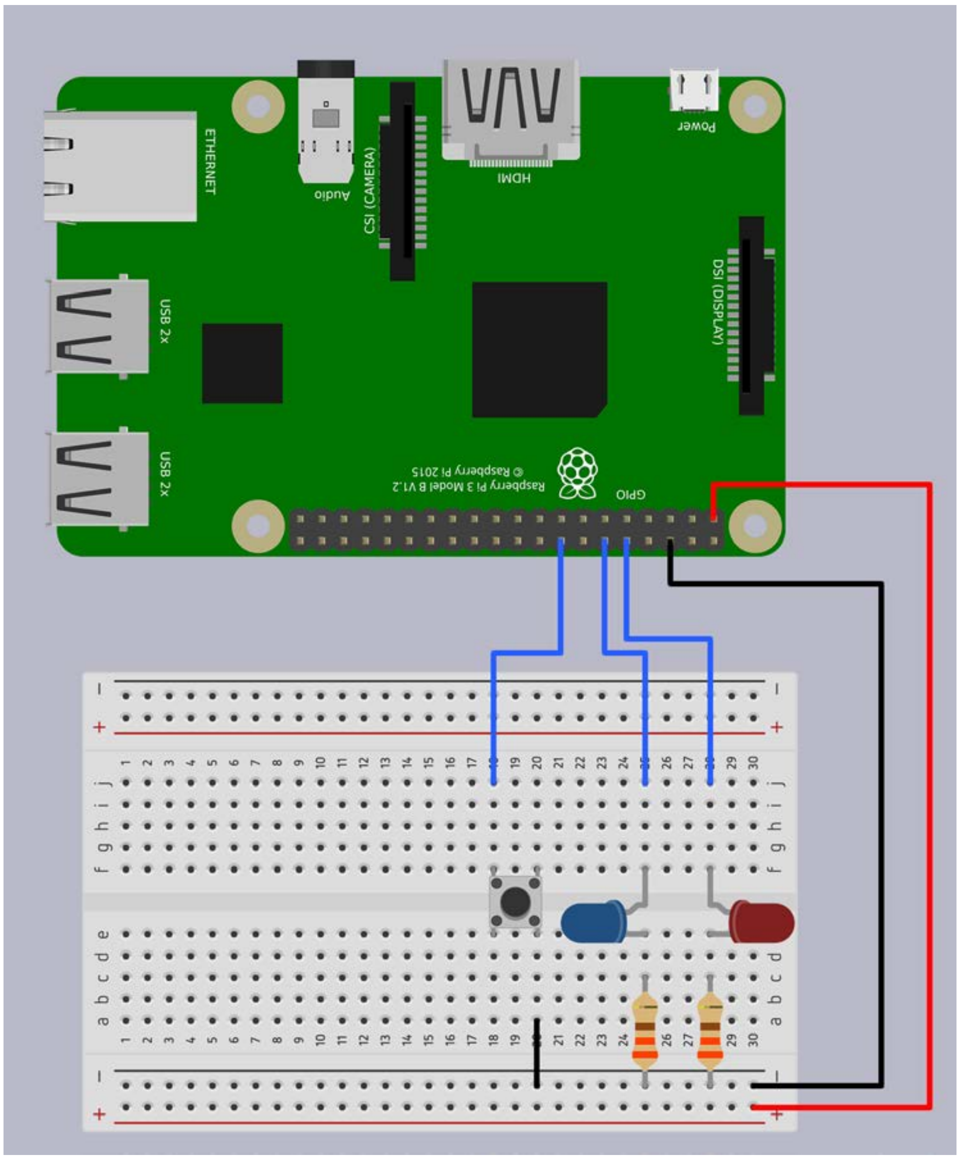
Raspberry Pi 3 Model B
Class 10 Micro SD Card
Keyboard + Mouse
Monitor + HDMI Cable
Power Supply (Recommended: 5V 2.5A)
Breadboard
1x Red LED
1x Blue LED
2x 330Ω Resistor
1x M/M Jumper Wire
4x M/F Jumper Wire
1x Button

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



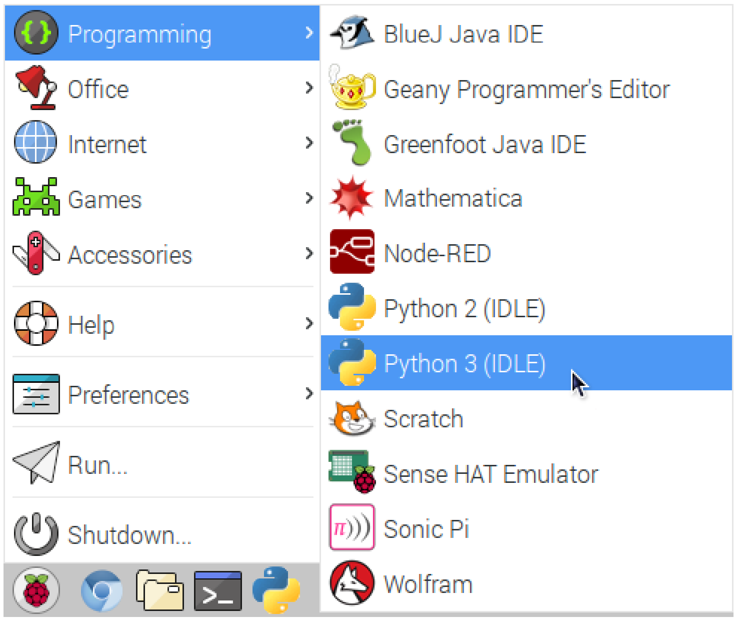
Introduction

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

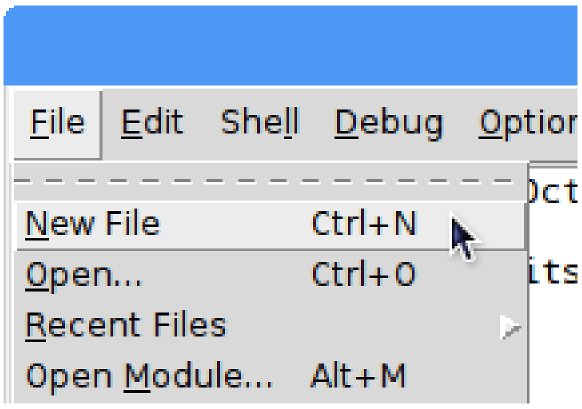
Some parts of the tutorials may look familiar as the code examples are written in a way that there is very little work needed and minor modifications to the previous code to get you up and running faster.

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)"

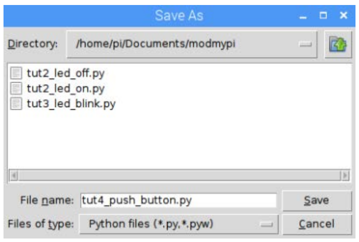


When IDLE has loaded, you will want to start working on a new file. You can do this by clicking on File and select "New File" or by pressing Ctrl+N



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.



Directory: /home/pi/Documents/modmypi

tut2_led_off.py
tut2_led_on.py
tut3_led_blink.py

File name: tut4_push_button.py

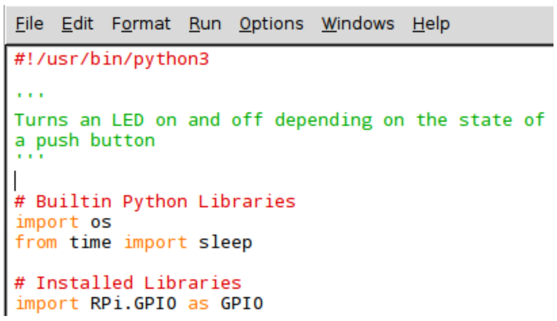
Files of type: Python files (*.py,*.pyw)

Go back to the IDLE and click on File and select “Save As”. Navigate to /home/pi/Documents/modmypi and enter tut4_push_button.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

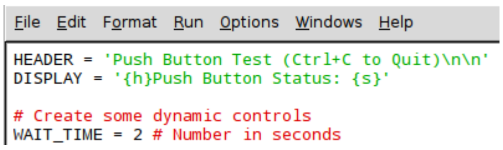
```
#!/usr/bin/python3

'''
Turns an LED on and off depending on the state of
a push button
'''

# Builtin Python Libraries
import os
from time import sleep

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



```
#!/usr/bin/python3

'''
Turns an LED on and off depending on the state of
a push button
'''

# Builtin Python Libraries
import os
from time import sleep

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

Display Variables

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



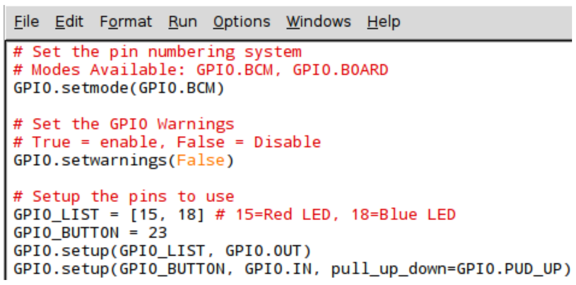
```
HEADER = 'Push Button Test (Ctrl+C to Quit)\n\n'
DISPLAY = '{h}Push Button Status: {s}'

# Create some dynamic controls
WAIT_TIME = 2 # Number in seconds
```

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

# Create some dynamic controls
WAIT_TIME = 2 # 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_BUTTON = 23
GPIO.setup(GPIO_LIST, GPIO.OUT)
GPIO.setup(GPIO_BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

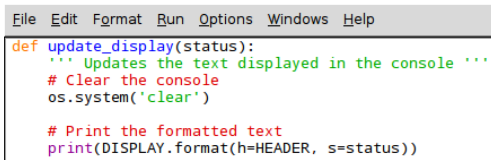


```
# Set the pin numbering system
# Modes Available: GPIO.BCM, GPIO.BOARD
GPIO.setmode(GPIO.BCM)

# 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_BUTTON = 23
GPIO.setup(GPIO_LIST, GPIO.OUT)
GPIO.setup(GPIO_BUTTON, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

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.

```
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))
```



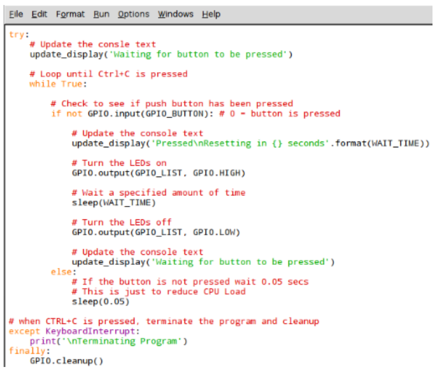
```
def update_display(status):
    ''' Updates the text displayed in the console '''
    # Clear the console
    os.system('clear')

    # Print the formatted text
    print(DISPLAY.format(h=HEADER, s=status))
```

The Action Code – Handling the button states

You will now need to add code to tell the program to turn on an LED when the push button is pressed and then reset after 2 seconds. There is the added if/else added to this code mixed with the previous tutorial.

The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed.

```
try:
    # Update the console text
    update_display('Waiting for button to be pressed')

    # Loop until Ctrl+C is pressed
    while True:

        # Check to see if push button has been pressed
        if not GPIO.input(GPIO_BUTTON): # 0 = button is pressed

            # Update the console text
            update_display('Pressed\nResetting in {} seconds'.format(WAIT_TIME))

            # Turn the LEDs on
            GPIO.output(GPIO_LIST, GPIO.HIGH)

            # Wait a specified amount of time
            sleep(WAIT_TIME)

            # Turn the LEDs off
            GPIO.output(GPIO_LIST, GPIO.LOW)

            # Update the console text
            update_display('Waiting for button to be pressed')
        else:
            # If the button is not pressed wait 0.05 secs
            # This is just to reduce CPU Load
            sleep(0.05)

# when CTRL+C is pressed, terminate the program and clean-up
except KeyboardInterrupt:
    print('\nTerminating Program')
finally:
    GPIO.cleanup()
```



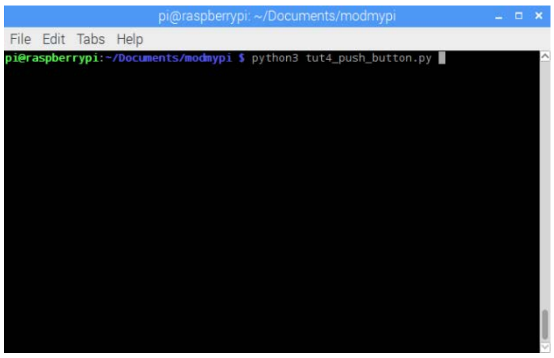
```
try:
    # Update the console text
    update_display('Waiting for button to be pressed')

    # Loop until Ctrl+C is pressed
    while True:

        # Check to see if push button has been pressed
        if not GPIO.input(GPIO_BUTTON): # 0 = button is pressed

            # Update the console text
            update_display('Pressed\nResetting in {} seconds'.format(WAIT_TIME))

            # Turn the LEDs on
            GPIO.output(GPIO_LIST, GPIO.HIGH)

            # Wait a specified amount of time
            sleep(WAIT_TIME)

            # Turn the LEDs off
            GPIO.output(GPIO_LIST, GPIO.LOW)

            # Update the console text
            update_display('Waiting for button to be pressed')
        else:
            # If the button is not pressed wait 0.05 secs
            # This is just to reduce CPU Load
            sleep(0.05)

# when CTRL+C is pressed, terminate the program and cleanup
except KeyboardInterrupt:
    print('\nTerminating Program')
finally:
    GPIO.cleanup()
```

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

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.

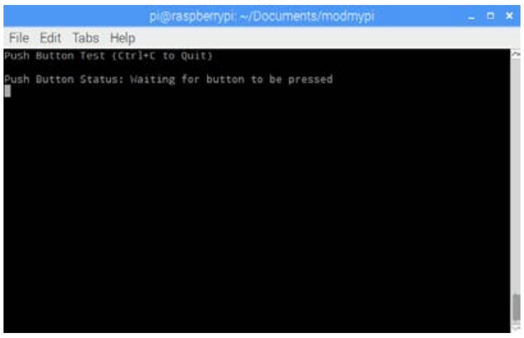
In the terminal, type python3 tut4_push_button.py and press enter



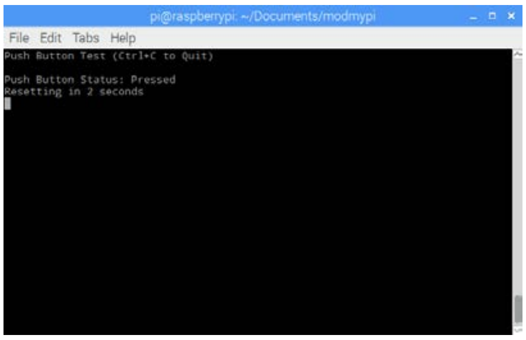
```
pi@raspberrypi:~/Documents/modmypi $ python3 tut4_push_button.py
```

Results

If everything is working correctly you should see in the console that it is waiting for the push button to be pressed. When the button is pressed the console text is updated and both LEDs turn on for 2 seconds before everything resets. In the console you should see something similar to this when running:



```
Push Button Test (Ctrl+C to Quit)

Push Button Status: Waiting for button to be pressed
```



```
Push Button Test (Ctrl+C to Quit)

Push Button Status: Pressed
Resetting in 2 seconds
```

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.