

YouTube

Workshop Kit v2.0

Tutorial 9/9: PIR Motion Sensor

Contents

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

Things you will need

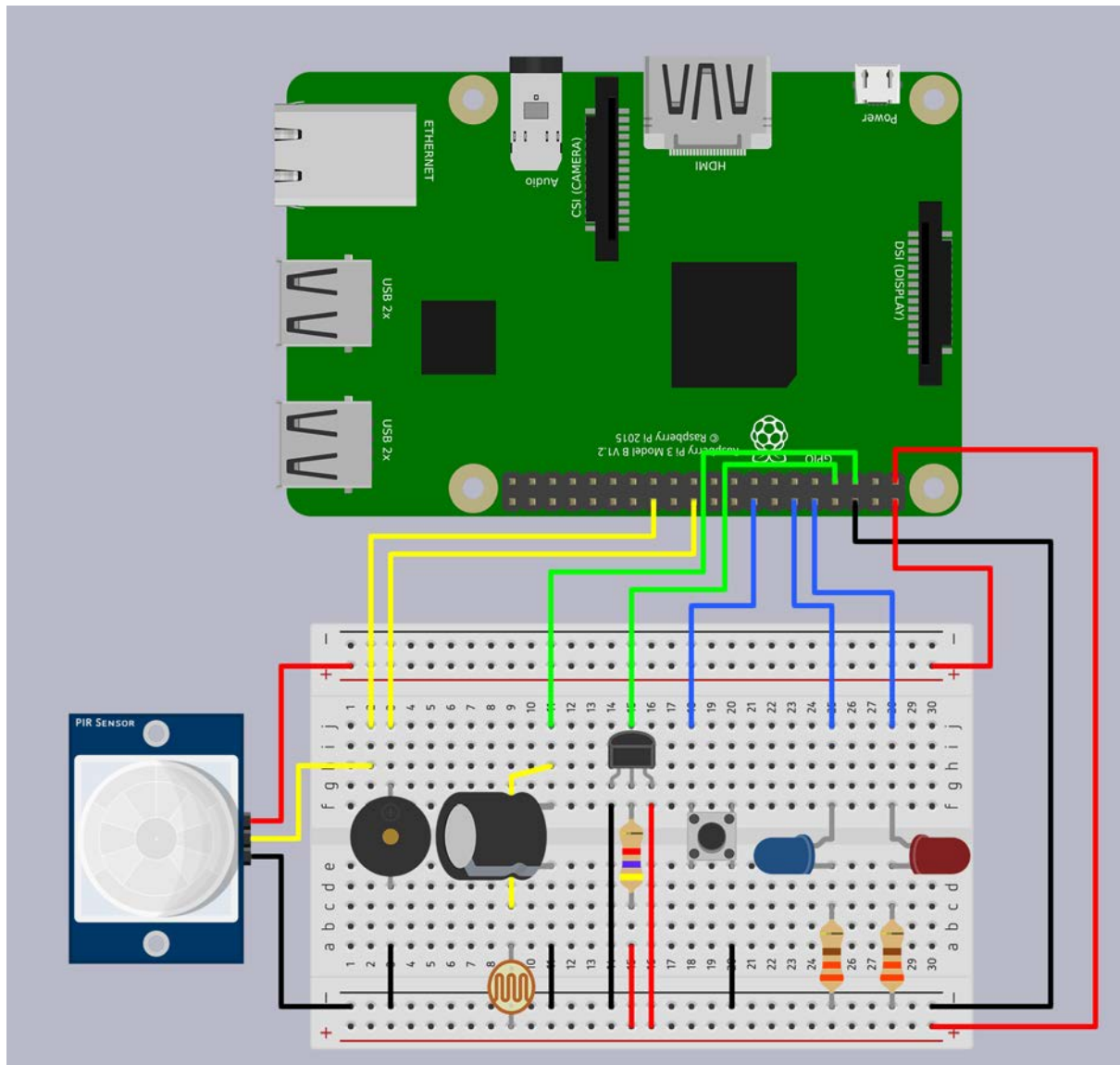
Raspberry Pi 3 Model B
Class 10 Micro SD Card
Keyboard + Mouse
Monitor + HDMI Cable
Power Supply (Recommended: 5V 2.5A)
Breadboard
1x Red LED
1x Blue LED
2x 330 Ω Resistor
5x M/M Jumper Wire
12x M/F Jumper Wire
1x Button
1x Buzzer
1x DS18B20 Temperature Sensor
1x 4k7 Ω Resistor
1x 1uF Capacitor
1x Light Dependent Resistor (LDR)
1x PIR Sensor

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 detect motion from a Passive Infrared Motion Sensor.

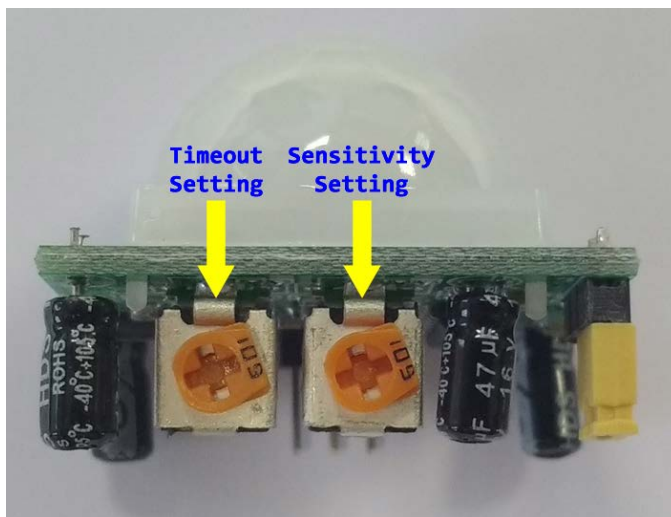
These tutorials are written on a Raspberry Pi 3 Model B with a clean install of Raspbian 4.4.

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.

PIR Information

Before we get started with anything, there is a little hardware tweaking that is required. This time we will be making tweaks on the PIR Motion Sensor.

On the PIR there are 2 Potentiometers that control the Sensitivity and Timeout settings, using a small Phillips screwdriver, adjust the potentiometers as shown below:

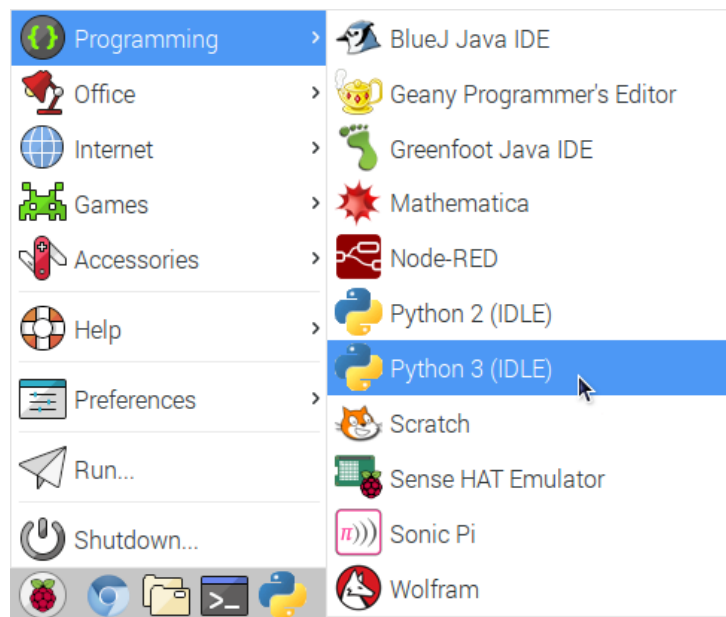


The potentiometer on the right controls the sensitivity, and the potentiometer on the left controls the timeout. Here, both are turned fully anti-clockwise, meaning that the sensitivity and timeout are at their lowest.

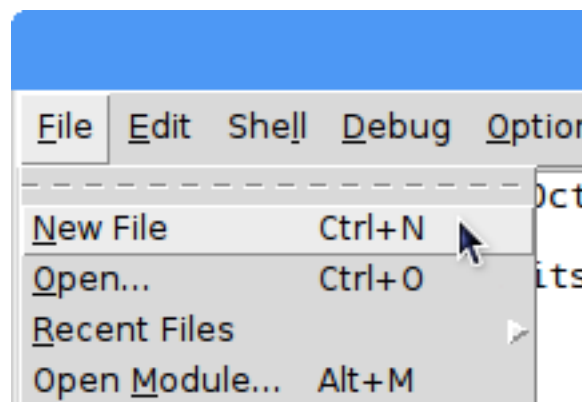
When the timeout is turned fully anti-clockwise, the PIR will output a signal for about 2.5 seconds, whenever motion is detected. If the potentiometer is turned fully clockwise, the output signal will last for around 250 seconds. When tuning the sensitivity, it is best to have the timeout set as low as possible.

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)”

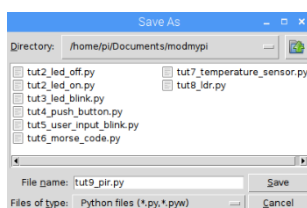


When IDLE has loaded, you will want to start working on a new file. You can do this by clicking on File and select “New File” or by pressing Ctrl+N



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.



Go back to the IDLE and click on File and select “Save As”. Navigate to /home/pi/Documents/modmypi and enter tut9_pir.py for the filename then click Save.

Writing the Code

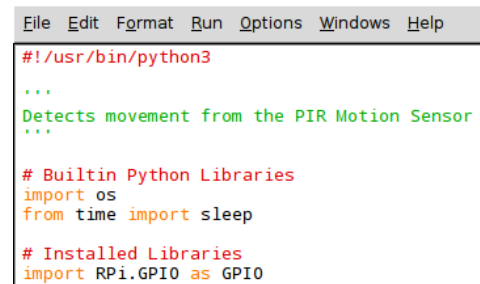
The first thing you should type is a shebang line, docstring and import your modules

```
#!/usr/bin/python3

'''
Detects movement from the PIR Motion Sensor
'''

# Builtin Python Libraries
import os
from time import sleep

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



```
File Edit Format Run Options Windows Help

#!/usr/bin/python3

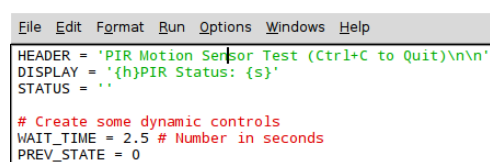
'''
Detects movement from the PIR Motion Sensor
'''

# 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, 3 string variables and 2 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 and a control variable to store the previous state of the PIR.



```
File Edit Format Run Options Windows Help

HEADER = 'PIR Motion Sensor Test (Ctrl+C to Quit)\n\n'
DISPLAY = '{h}PIR Status: {s}'
STATUS = ''

# Create some dynamic controls
WAIT_TIME = 2.5 # Number in seconds
PREV_STATE = 0
```

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

# Create some dynamic controls
WAIT_TIME = 2.5 # Number in seconds
PREV_STATE = 0
```

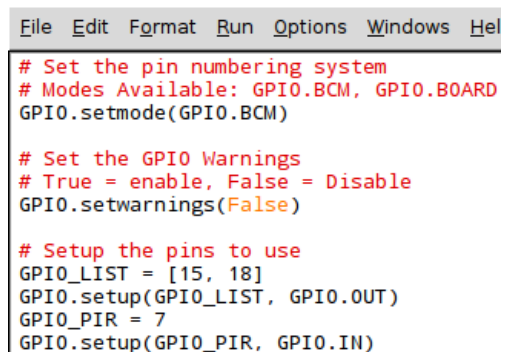
Setting up the GPIO

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]
GPIO.setup(GPIO_LIST, GPIO.OUT)
GPIO_PIR = 7
GPIO.setup(GPIO_PIR, GPIO.IN)
```



```
File Edit Format Run Options Windows Help

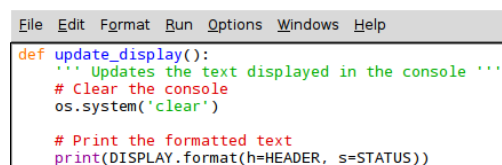
# Set the pin numbering system
# Modes Available: GPIO.BCM, GPIO.BOARD
GPIO.setmode(GPIO.BCM)

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

# Setup the pins to use
GPIO_LIST = [15, 18]
GPIO.setup(GPIO_LIST, GPIO.OUT)
GPIO_PIR = 7
GPIO.setup(GPIO_PIR, GPIO.IN)
```

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.



```
File Edit Format Run Options Windows Help

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

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

```
def update_display():
    ''' 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 – Detect Movement

This code will get detect if there is any movement and update the console text with the current state. If motion is detected it will also turn on the LEDs.

```
try:
    while True:
        # Loop until PIR output is 0
        while GPIO.input(GPIO_PIR) == 1:
            STATUS = 'Trying to detect Motion'

        # Loop until users quits with CTRL-C
        while True:
            # Read PIR state
            if GPIO.input(GPIO_PIR) == 1 and PREV_STATE == 0:
                # Set Status
                STATUS = 'Motion detected!'

                # Update outputs
                GPIO.output(GPIO_LIST, GPIO.HIGH)

                # Set PREV_STATE
                PREV_STATE = 1

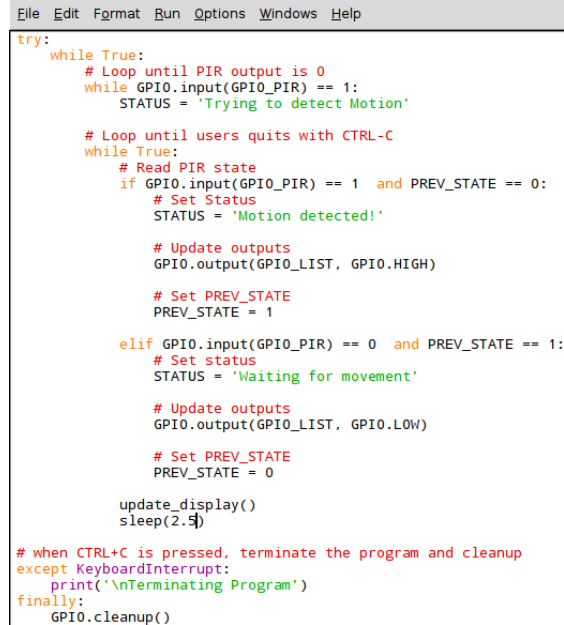
            elif GPIO.input(GPIO_PIR) == 0 and PREV_STATE == 1:
                # Set status
                STATUS = 'Waiting for movement'

                # Update outputs
                GPIO.output(GPIO_LIST, GPIO.LOW)

                # Set PREV_STATE
                PREV_STATE = 0

            update_display()
            sleep(2.5)

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



```
File Edit Format Run Options Windows Help
try:
    while True:
        # Loop until PIR output is 0
        while GPIO.input(GPIO_PIR) == 1:
            STATUS = 'Trying to detect Motion'

        # Loop until users quits with CTRL-C
        while True:
            # Read PIR state
            if GPIO.input(GPIO_PIR) == 1 and PREV_STATE == 0:
                # Set Status
                STATUS = 'Motion detected!'

                # Update outputs
                GPIO.output(GPIO_LIST, GPIO.HIGH)

                # Set PREV_STATE
                PREV_STATE = 1

            elif GPIO.input(GPIO_PIR) == 0 and PREV_STATE == 1:
                # Set status
                STATUS = 'Waiting for movement'

                # Update outputs
                GPIO.output(GPIO_LIST, GPIO.LOW)

                # Set PREV_STATE
                PREV_STATE = 0

            update_display()
            sleep(2.5)

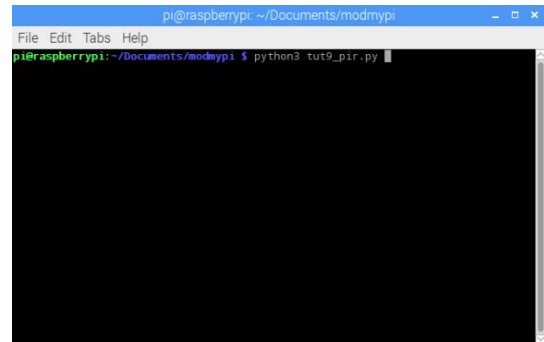
        # when CTRL+C is pressed, terminate the program and cleanup
    except KeyboardInterrupt:
        print('\nTerminating Program')
    finally:
        GPIO.cleanup()
```

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 tut8_ldr.py and press enter

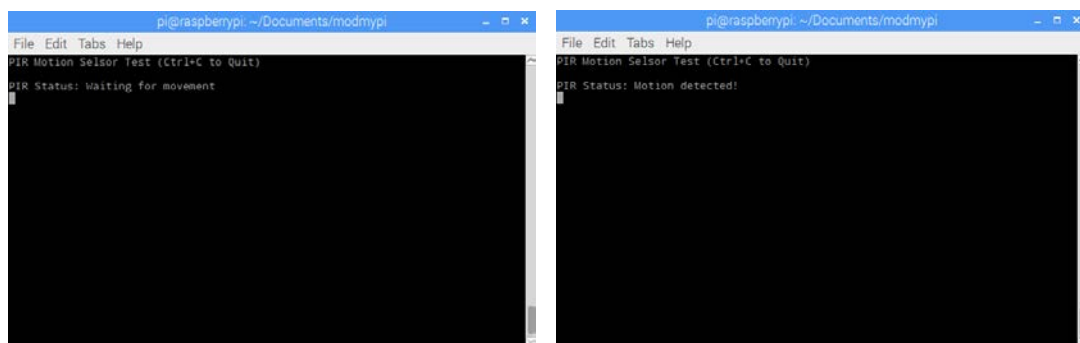


Results

If everything is working correctly you should see the console showing that it is waiting to detect movement. When movement is detected it will light up the LEDs and display that movement has been detected.

With the configuration of the motion sensor when movement is detected, the sensor will produce the signal of 1 for 2.5 seconds, and when reset it will send a signal of 0 for 2.5 seconds.

In the console you should see something similar to this when running:



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.