

Experiment-4

4. Saving electricity is one of the most important responsibility of today's citizens.

Use the devices and sensors to develop an efficient IOT model for smart lighting system

Aim: To develop a iot model for lighting system to save electricity

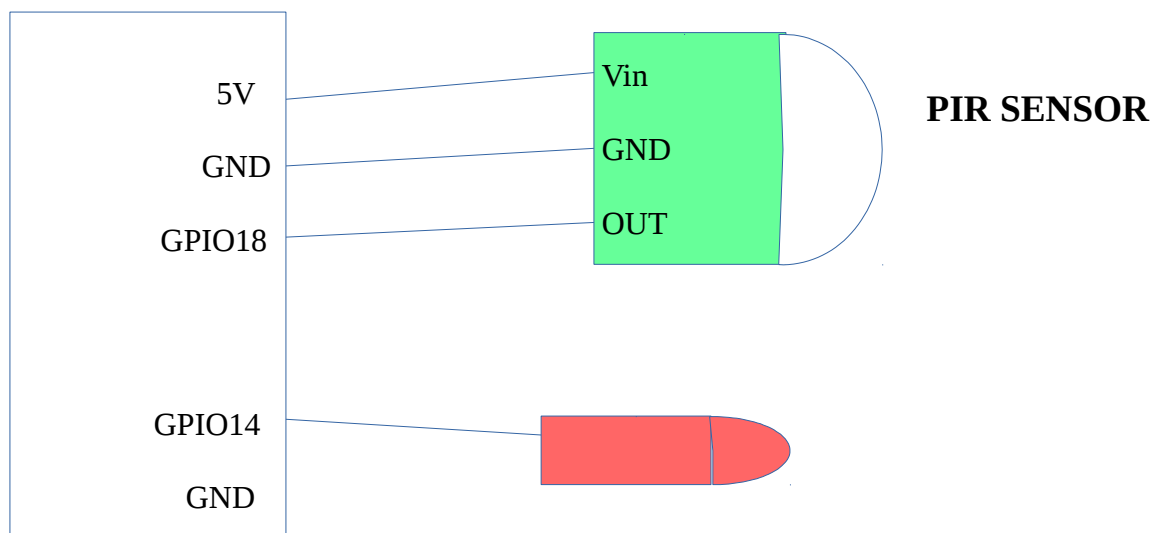
Objective: Smart lighting is used for energy saving which can be achieved by adapting lighting to the ambient conditions and by switching on/off or dimming of lights according to user needs thus reducing the unnecessary use of energy. Saving energy also helps in reducing cost.

The smart lighting can be implemented with Solid State lighting (LEDs) or IP-enabled lights (Internet or wireless controlled). The smart lighting works by sensing the occupancy, temperature/humidity and level in the environment.

HARDWARE REQUIREMENTS:

- Raspberry pi
- Jumper Wires
- Bread Board
- HDMI Cable
- LED
- PIR Sensor

Circuit Diagram:



RASPBERRYPI

LED

Working :

Connect all the wires to raspberry pi. Connect Monitor to Raspberry Pi using HDMI cable. Connect the 5V pin to VIN pin in PIR sensor, connect the ground pin to ground, and the OUT pin to the GPIO18 PIN, the LED pins are connected to GPIO14 and GND as shown in circuit diagram. By running the python code given below motion will be detected, and LED will get on and off while motion is detected the LED light gets ON.

PIR sensor: _

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

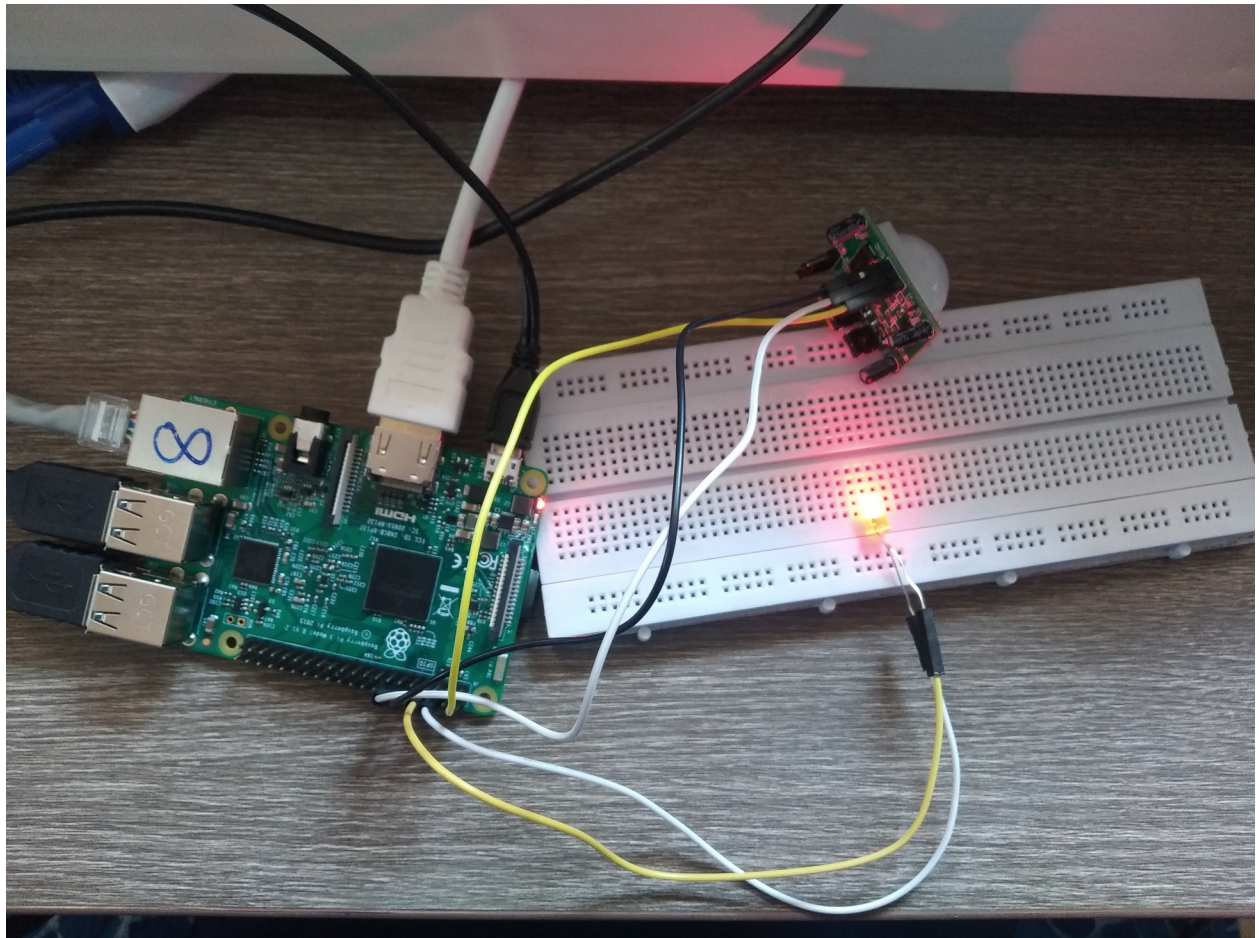
Source code:

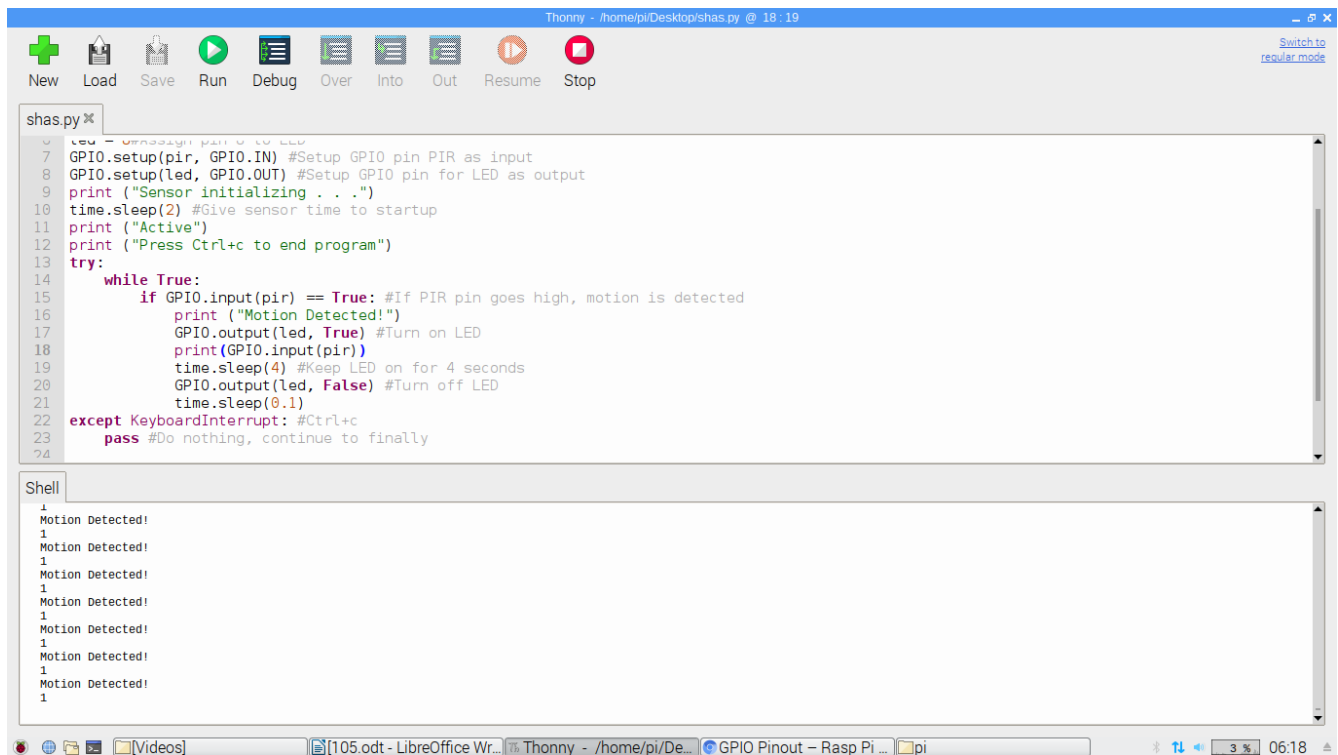
```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD) #Set GPIO to pin numbering
pir = 12 #Assign pin 8 to PIR
led = 8 #Assign pin 10 to LED
GPIO.setup(pir, GPIO.IN) #Setup GPIO pin PIR as input
GPIO.setup(led, GPIO.OUT) #Setup GPIO pin for LED as output
print ("Sensor initializing . . .")
time.sleep(2) #Give sensor time to startup
print ("Active")
print ("Press Ctrl+c to end program")
try:
    while True:
        if GPIO.input(pir) == True: #If PIR pin goes high, motion is detected
            print ("Motion Detected!")
            GPIO.output(led, True) #Turn on LED
```

```
time.sleep(4) #Keep LED on for 4 seconds  
GPIO.output(led, False) #Turn off LED  
time.sleep(0.1)
```

```
except KeyboardInterrupt: #Ctrl+c  
    pass #Do nothing, continue to finally  
finally:  
    GPIO.output(led, False) #Turn off LED in case left on  
    GPIO.cleanup() #reset all GPIO  
    print ("Program ended")
```

OUTPUT:





The screenshot shows the Thonny IDE interface on a Raspberry Pi. The top toolbar includes icons for New, Load, Save, Run, Debug, Over, Into, Out, Resume, and Stop. The main editor window displays a Python script named 'shas.py' with the following code:

```
6 # Setup GPIO pin PIR as input
7 GPIO.setup(pir, GPIO.IN) #Setup GPIO pin PIR as input
8 GPIO.setup(led, GPIO.OUT) #Setup GPIO pin for LED as output
9 print ("Sensor initializing . . .")
10 time.sleep(2) #Give sensor time to startup
11 print ("Active")
12 print ("Press Ctrl+c to end program")
13 try:
14     while True:
15         if GPIO.input(pir) == True: #If PIR pin goes high, motion is detected
16             print ("Motion Detected!")
17             GPIO.output(led, True) #Turn on LED
18             print(GPIO.input(pir))
19             time.sleep(4) #Keep LED on for 4 seconds
20             GPIO.output(led, False) #Turn off LED
21             time.sleep(0.1)
22 except KeyboardInterrupt: #Ctrl+c
23     pass #Do nothing, continue to finally
24
```

The bottom shell window shows the output of the program, displaying 'Motion Detected!' multiple times, indicating successful detection of motion events.

The taskbar at the bottom shows the following open applications: [Videos], [105.odt - LibreOffice Wr...], Thonny - /home/pi/De..., GPIO Pinout - Rasp Pi ..., and pi.