**PIR Motion Sensor Interfacing with**

**Raspberry Pi using Python.**

* PIR sensor is used for detecting infrared heat radiations. This makes them useful in the detection of moving living objects that emit infrared heat radiations.
* The output (in terms of voltage) of PIR sensor is high when it senses motion; whereas it is low when there is no motion (stationary object or no object).
* PIR sensors are used in many applications like for room light control using human detection, human motion detection for security purpose at home, etc.

**How a PIR Motion Sensor Works**

* A PIR motion sensor consists of a lens over a pair of pyroelectric sensors: a crystal material that produces a voltage when it receives infrared light.
* But objects that contain any amount of heat emit some infrared. That means a single pyroelectric sensor would always produce a voltage, even when everything in view is totally still.
* So how do you turn that into a motion sensor? It’s simple yet clever. The two pyroelectric sensors are connected in such a way that the signals from the ambient infrared cancel each other out.
* When something large enough and warm enough moves in front of the lens, the two signals stop canceling out, and the sensor reports that something’s moving.

**PIR Motion Sensors Can Give False Positives**

* When you connect a PIR motion sensor to power, it needs 30 to 60 seconds to stabilize. Otherwise, it can fire false positives in this time, and this can be a headache in some circuits. With a Raspberry Pi project, it’s easy to write software that tells it to wait a short while after booting.
* The greater problem is with false positives from animals, insects, or even gusts of wind triggering the sensor.
* You can usually calibrate the component to be less sensitive, and this will stop your cat or small dog from triggering it. It’s less useful for insects flying directly in front of it
* Don’t let these false positives be a dealbreaker, though; just think about how to handle them gracefully.

**Handling a PIR Motion Sensor in Python**

* There are many ways to interface a PIR motion sensor with your Raspberry Pi, from visual programming tools like Scratch down to close-to-the-metal code like C.
* Here, we’ll use Python with the GPIO Zero library, which sorts out most of the small details for us.
* If you’ve never written a line of Python before, you might want to try a few tutorials from Automate the Boring Stuff With Python. If you don’t, the code below will still work; you’ll just get more out of this if you can follow along.
* This project also assumes a basic understanding of the [GPIO pins](https://www.deviceplus.com/raspberry-pi/an-introduction-to-raspberry-pi-gpio-pins/). If they’re brand new to you, check out the past guides. Otherwise, there’s a diagram below that should remind you of the basics.

### Connecting a PIR Motion Sensor to the

### Raspberry Pi

To get started building your [Raspberry Pi Motion](https://www.deviceplus.com/raspberry-pi/how-to-create-a-motion-control-lamp-with-raspberry-pi/) sensor, you’ll need:

|  |  |
| --- | --- |
| **A**[**Raspberry Pi**](https://thepihut.com/products/raspberry-pi-4-model-b?variant=20064052674622&currency=GBP&utm_medium=product_sync&utm_source=google&utm_content=sag_organic&utm_campaign=sag_organic&gclid=Cj0KCQjwlN32BRCCARIsADZ-J4s5VAXINFtbw9Of96pQszrsr3iA4kdZung_1ozb7Qjq6Kgl6Rku8h0aAsfUEALw_wcB)**with power supply**  **an**[**SD card with Raspbian**](https://shop.pimoroni.com/products/noobs-32gb-microsd-card-3-1?variant=31703694245971)**installed** |  |
| **A**[**breadboard**](https://uk.rs-online.com/web/p/breadboards/1029147/) |  |
| **A**[**GPIO extension board**](https://cpc.farnell.com/pro-signal/ctlcobblerassembled/cobbler-assembled/dp/SC13285?mckv=sEV9OyYS1_dc|pcrid|224679642167|kword||match||plid||slid||product|SC13285|pgrid|47129782516|ptaid|pla-558431019053|&CMP=KNC-GUK-CPC-SHOPPING&gclid=Cj0KCQjwlN32BRCCARIsADZ-J4vR71czEfw4bB-nG9xvxmNcNZUQoeLChcTMYnOoWWlZ2QVZ38WiZugaAvJREALw_wcB) |  |
| **A**[**PIR motion sensor**](https://www.jaycar.co.uk/arduino-compatible-pir-motion-detector-module/p/XC4444)  **Code**  import RPi.GPIO as GPIO  import time  GPIO.setmode(GPIO.BCM)  GPIO.setup(23, GPIO.IN) #PIR  GPIO.setup(24, GPIO.OUT) #BUzzer  try:  time.sleep(2) # to stabilize sensor  while True:  if GPIO.input(23):  GPIO.output(24, True)  time.sleep(0.5) #Buzzer turns on for 0.5 sec  GPIO.output(24, False)  print("Motion Detected...")  time.sleep(5) #to avoid multiple detection  time.sleep(0.1) #loop delay, should be less than detection delay  except:  GPIO.cleanup() |  |