Sensor de presencia en Raspberry Pi

13/05/2013 0:01 · 4 comentarios · manuel

Podemos hacer uso de la biblioteca de GPIO para Python, conectar sensores digitales y tomar acciones de acuerdo al estado de estos sensores.

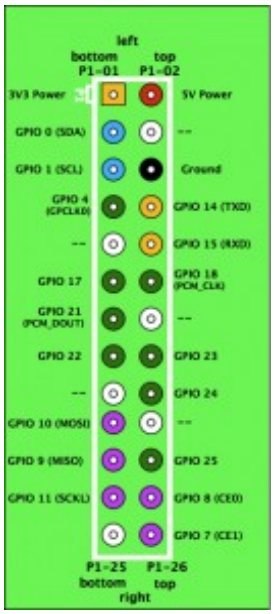
El siguiente tutorial nos muestra como instalar la biblioteca GPIO para Python y como conectar un sensor de presencia para mostrar una salida por pantalla cuando se active.

Materiales

- Fuente de poder 5V @ 2A
- Acceso a Inet
- Protoboard
- Conector para Raspberry Pi
- Sensor de presencia rev. A. mas información en este **link**

Sobre el Hardware

Los puertos GPIO (General Purpose Input Output) se encuentran junto a la conector de la tarjeta SD, y están etiquetados con P1. Es un puerto de 26 pines. La siguiente figura muestra los pines.



El sensor de presencia o sensor PIR (Passive Infra-Red) es un equipo piroeléctrico que detecta movimiento por la variación en los niveles de de infrarrojo (calor) que emiten los objetos cercanos. Cuando se detecta movimiento. el sensor produce una señal de nivel alto en el pin de salida. Este modelo cuenta con un jumper que permite cambiar la operacion del pin de salida. Nosotros dejamos el jumper en la posición H

La conexión es la siguiente, se indican los pines en la Raspberry Pi

PIN 1 a +
PIN 6 a GND
PIN 26 a Salida

El conector de la Raspberry Pi va al protoboard y se usan cables para conectar a los pines del sensor.

Instalando Software

Instalamos la biblioteca de GPIO para Python usando los siguientes comandos

```
sudo wget https://pypi.python.org/packages/source/R/RPi.GPIO/RPi.GPIO-0.5.0a.tar.gz
tar zxf RPi.GPIO-0.5.0a.tar.gz
cd RPi.GPIO-0.5.0a
sudo python setup.py install
```

Esta biblioteca ya la habíamos usado antes en el artículo de hola mundo pero esta versión de la biblioteca es más actualizada.
Para verificar usamos los siguientes comandos

```
sudo python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
```

```
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> import RPi.GPIO as GPIO
>>> GPIO.RPI_REVISION
2
>>> GPIO.VERSION
'0.5.0a'
```

Codificando en Python

Para importar la biblioteca usamos

```
import RPi.GPIO as GPIO
```

Para definir la nomenclatura a usar

```
GPIO.setmode(GPIO.BCM)
```

este comando significa que usamos la numeracion original del chip BroadCom

Como ejemplo muy básico de esto es indicar por pantalla si detecta o no una presencia. Usamos el siguiente comando

```
sudo nano PIR_demo.py
```

Para insertar el código que se muestra a continuación

```
# Basic testing on presence sensor
#
# Author : Manuel Carrasco
# Date   : 12/05/2013

# Import required Python libraries
import time
import RPi.GPIO as GPIO

# Use BCM GPIO references
# instead of physical pin numbers
GPIO.setmode(GPIO.BCM)

# Define GPIO to use on Pi
GPIO_PIR = 7

print "PIR Module basic test (CTRL-C to exit)"
```

```
GPIO.setwarnings(False)

# Set pin as input
GPIO.setup(GPIO_PIR,GPIO.IN)

try:

        print "Waiting for PIR to settle ..."
        time.sleep(1)
        print "  Ready"

    # Loop until users quits with CTRL-C
        while True :
            if GPIO.input(GPIO_PIR):
                print("Se detecta  presencia")
                time.sleep(0.5)
            else :
                print("No hay presencia")
                time.sleep(0.5)
except KeyboardInterrupt:
                print "  Quit"
    # Reset GPIO settings
GPIO.cleanup()
```

Para funcionar necesita el siguiente comando

```
sudo python PIR_demo.py
```

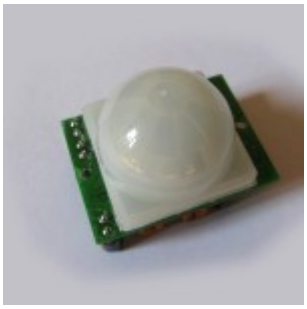
Obviamente hay muchas variaciones posibles. Se pueden tomar acciones u obtener información de otros sensores.

Bibliografia

http://www.raspberrypi-spy.co.uk/2013/01/cheap-pir-sensors-and-the-raspberry-pi-part-1/
https://code.google.com/p/raspberry-gpio-python/wiki/Main

Cheap PIR Sensors and the Raspberry Pi – Part 1

Posted on January 21, 2013 by Matt



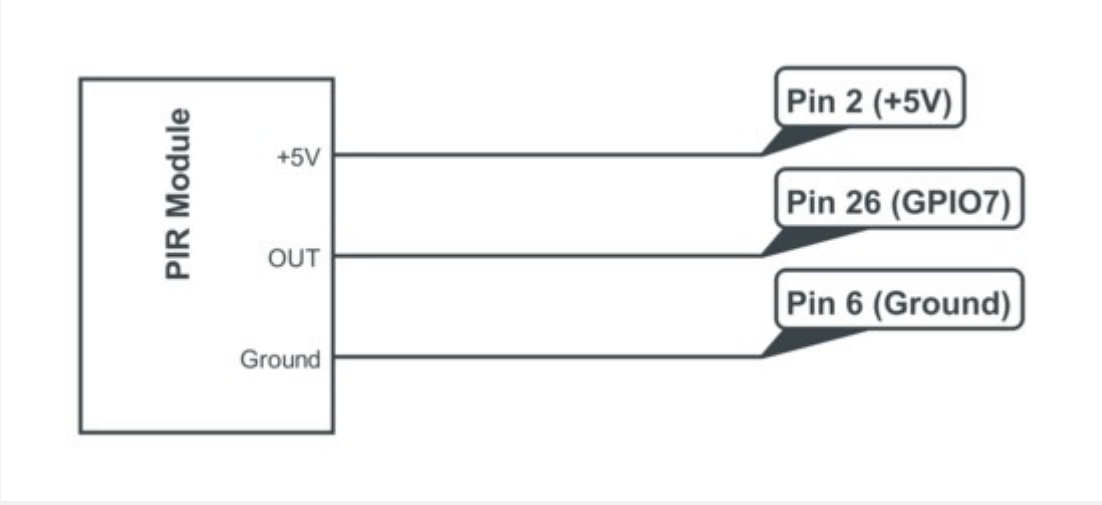
A great little sensor you can add to your Raspberry Pi projects is a PIR module. These 5V "Passive Infra Red" sensors are available for a few pounds from eBay. They can be powered from 5V and output 3V so can be connected directly to pins on the Pi's GPIO header without any other components.

The module sets a single output pin high whenever it detects movement within its field of view. It holds this pin High (3.3V) for a minimum period of time. If continuous movement is detected the output pin will stay High. When the time has elapsed and no more movement is detected the output pin returns Low (0V).

I am currently using one in an alarm system and it works great for such a small and cheap device.

PIR Connections

Here is a diagram showing the pin-out on the PIR module and how I connected it to my Raspberry Pi :

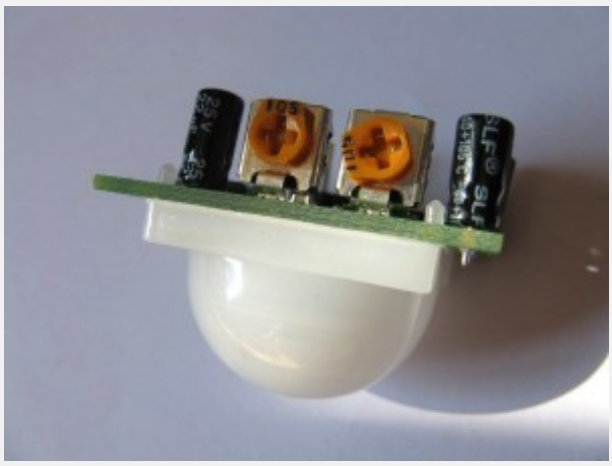


PIR Module

The device has two variable resistors that you can adjust to tweak the performance of the module.

The first one (left-hand side on the photo) determines the sensitivity of the device. The default setting is usually 50%.

The second control (right-hand side on the photo and usually marked “time” on the PCB) allows you to adjust the amount of time the output pin stays at 3V (high) when it is triggered by movement. This can be set from a few seconds to 200 seconds. The default setting is usually a few seconds.



Trimming Controls

The units available on eBay vary in specification but they are all very similar.

Python Example Script

If you connect your module as shown in the diagram above the following Python script will allow you to get started. Cut and paste the script below into a text file and transfer to the Pi or download the script directly using this link.

```
#!/usr/bin/python

#+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

#|R|a|s|p|b|e|r|r|y|P|i|-|S|p|y|.|c|o|.|u|k|

#+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

#

# pir_1.py

# Detect movement using a PIR module

#

# Author : Matt Hawkins

# Date   : 21/01/2013



# Import required Python libraries
```

```
import RPi.GPIO as GPIO

import time



# Use BCM GPIO references

# instead of physical pin numbers

GPIO.setmode(GPIO.BCM)



# Define GPIO to use on Pi

GPIO_PIR = 7



print "PIR Module Test (CTRL-C to exit)"



# Set pin as input

GPIO.setup(GPIO_PIR,GPIO.IN)      # Echo



Current_State  = 0

Previous_State = 0



try:



  print "Waiting for PIR to settle ..."
```

```
  # Loop until PIR output is 0

  while GPIO.input(GPIO_PIR)==1:

    Current_State  = 0



  print "  Ready"



  # Loop until users quits with CTRL-C

  while True :



    # Read PIR state

    Current_State = GPIO.input(GPIO_PIR)



    if Current_State==1 and Previous_State==0:

      # PIR is triggered

      print "  Motion detected!"

      # Record previous state

      Previous_State=1

    elif Current_State==0 and Previous_State==1:

      # PIR has returned to ready state

      print "  Ready"

      Previous_State=0
```

```
    # Wait for 10 milliseconds

    time.sleep(0.01)



except KeyboardInterrupt:

  print "  Quit"

  # Reset GPIO settings

  GPIO.cleanup()
```

This script can also be downloaded onto your Pi directly using this command line :

```
wget http://www.raspberrypi-spy.co.uk/archive/python/pir_1.py
```

This can then be run using :

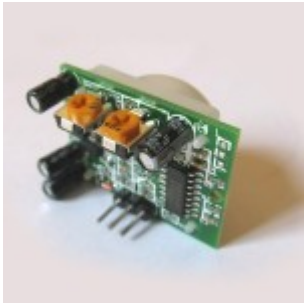
```
sudo python pir_1.py
```

When run the script waits for the output pin to go Low. It then prints a message to the screen every time the output state changes. This is either when movement is detected (output changes to High) or the device sees no movement (outout changes to Low).

Try changing the reset time by turning the “time” resistor clockwise by a few degrees. Run the script again, trigger the device and then wait to see how long it takes to go back to the ready state.

Cheap PIR Sensors and the Raspberry Pi – Part 2

Posted on February 24, 2013 by Matt



Following on from my first PIR sensor module article I thought I would create a Python script that allowed me to easily measure the reset time.
That way I could attach a module, run the script and measure the time it took for the output pin to drop back to the Low state. Then I could tweak the trimming resistor and adjust the reset time to my preferred value in a more controlled and precise way.

Example Python Script

The following script assumes you have your PIR module connected to the GPIO header as shown in Part 1. The example uses GPIO7 (Pin 26).
Cut and paste the script below into a text file and transfer to the Pi or download the script directly using this link (recommended).

```
#!/usr/bin/python

#+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

#|R|a|s|p|b|e|r|r|y|P|i|-|S|p|y|.|c|o|.|u|k|

#+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

#

# pir_2.py

# Measure the holding time of a PIR module

#

# Author : Matt Hawkins

# Date   : 20/02/2013



# Import required Python libraries

import time

import RPi.GPIO as GPIO



# Use BCM GPIO references

# instead of physical pin numbers

GPIO.setmode(GPIO.BCM)
```

```
# Define GPIO to use on Pi

GPIO_PIR = 7



print "PIR Module Holding Time Test (CTRL-C to exit)"



# Set pin as input

GPIO.setup(GPIO_PIR,GPIO.IN)      # Echo



Current_State  = 0

Previous_State = 0



try:



  print "Waiting for PIR to settle ..."



  # Loop until PIR output is 0

  while GPIO.input(GPIO_PIR)==1:

    Current_State  = 0



  print "  Ready"



  # Loop until users quits with CTRL-C
```

```
  while True :


    # Read PIR state

    Current_State = GPIO.input(GPIO_PIR)


    if Current_State==1 and Previous_State==0:

      # PIR is triggered

      start_time=time.time()

      print "  Motion detected!"

      # Record previous state

      Previous_State=1

    elif Current_State==0 and Previous_State==1:

      # PIR has returned to ready state

      stop_time=time.time()

      print "  Ready ",

      elapsed_time=int(stop_time-start_time)

      print " (Elapsed time : " + str(elapsed_time) + " secs)"

      Previous_State=0


except KeyboardInterrupt:

  print "  Quit"

  # Reset GPIO settings
```

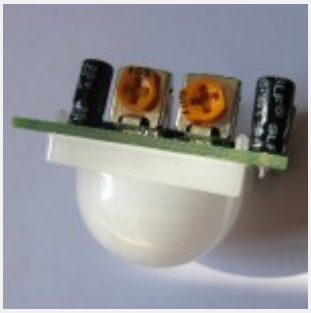
```
  GPIO.cleanup()
```

Alternatively the script can also be easily downloaded onto your Pi directly using this command line :

```
wget http://www.raspberrypi-spy.co.uk/archive/python/pir_2.py
```

The script can then be run using :

```
sudo python pir_2.py
```



PIR Module with two trimming controls

When run the script waits for the output pin to go Low. Just like in pir_1.py it prints a message to the screen when movement is detected. The main difference is that it now measures the elapse of time between the output pin going High and it returning to Low.

You can use a small screwdriver to tweak the “time” control to increase or decrease the time. Once you’ve triggered the module it is important to stay still so you don’t increase the time the output stays High which will mess up your results.

A half turn will result in the reset time being increased by a few minutes so it’s best to make adjustments in small increments.