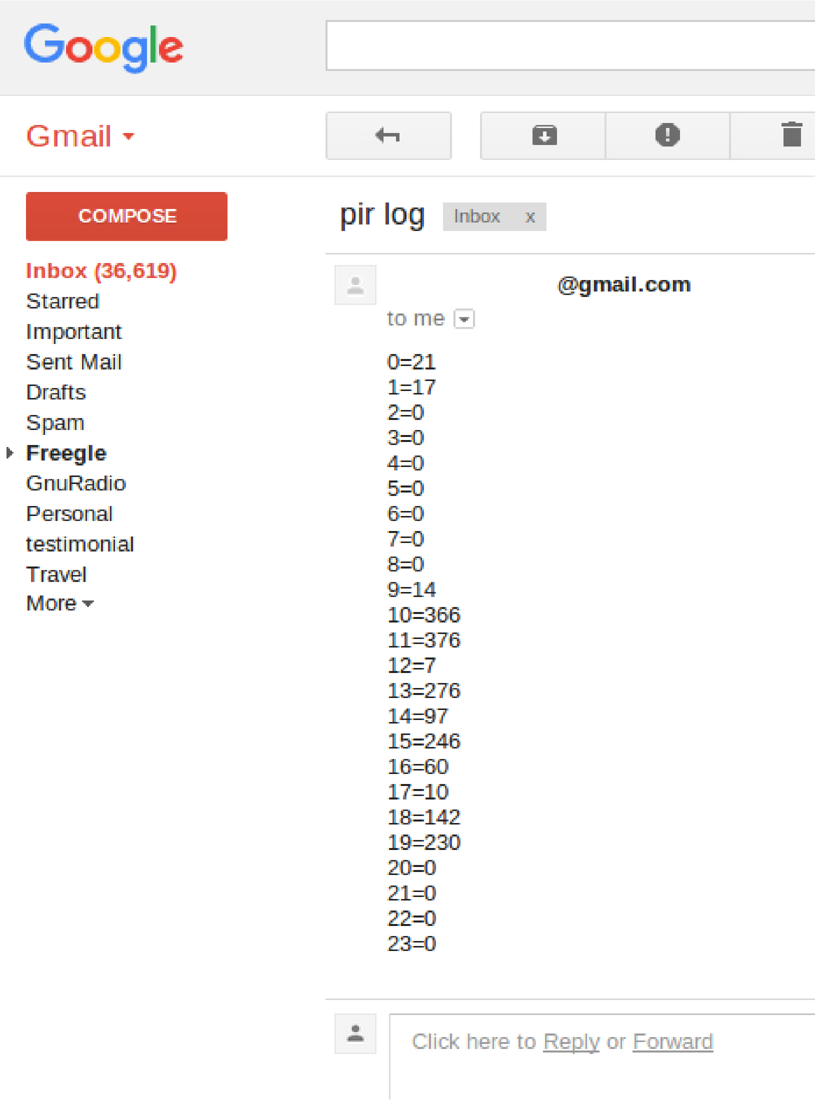
Senior Guardian PIR Project

The idea of the Senior Guardian project is to be able to discretely keep an eye on an older vulnerable relative who lives alone, without having a camera pointing at them.

The PIR sensor passively records how many movement events happen each hour, by detecting changes of body heat movement in front of the sensor. If the PIR is positioned in a thoroughfare such as a hallway or landing, then the person living in the house will perhaps pass the sensor several times each hour, and the Raspberry Pi will keep a running total of each movement detection, and then email you today's totals each hour. What you receive on your PC or phone looks like this:



pir log

@gmail.com

to me

```
0=21
1=17
2=0
3=0
4=0
5=0
6=0
7=0
8=0
9=14
10=366
11=376
12=7
13=276
14=97
15=246
16=60
17=10
18=142
19=230
20=0
21=0
22=0
23=0
```

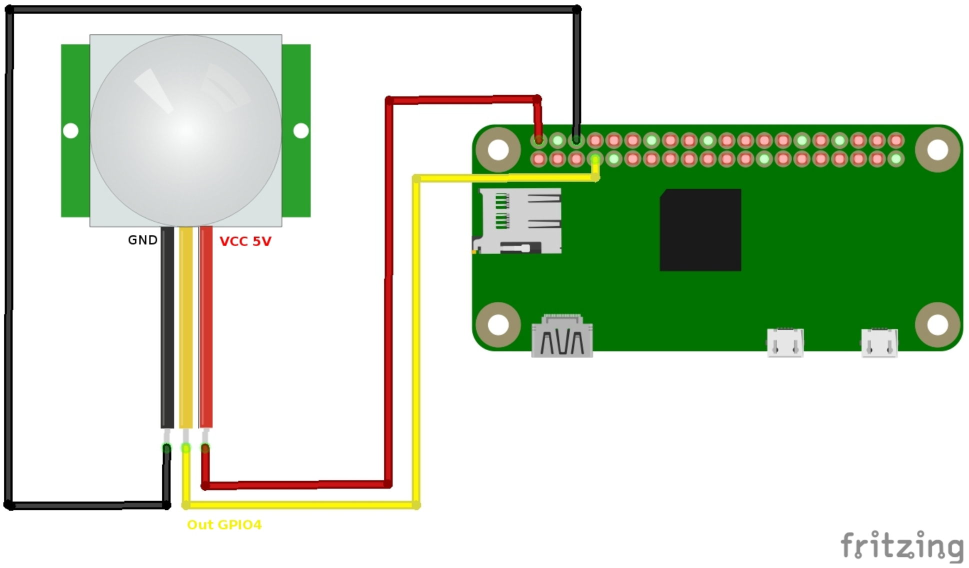
As you can see from the picture above, our demo person didn't trigger the PIR between sometime after 1am until sometime after 9am. Also, they didn't trigger the sensor between 8pm and midnight.

If they hadn't triggered the PIR at all in a 24 hour period, we'd be legitimately concerned that they might have had a fall and not be able to get to the phone.

Alternatively, a PIR system with a camera could also be used in different situations, like to keep an eye on a stockroom or building entrance. https://www.amazon.co.uk/dp/B00GOPJJWU

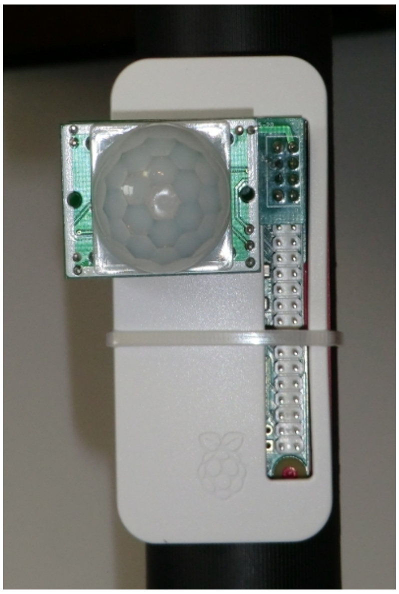
Connecting the PIR sensor to a Raspberry Pi

The PIR sensor connects to a Raspberry Pi Zero W or Pi 3 Model B using three wires.



The wires you use may vary in colour from those shown above. Connect VCC to 5V, OUT to GPIO4 and GND to GND. Pin labels are screen printed on the PIR's circuit board – lift the plastic dome off if you don't see them. The two trimmers on the top of the PIR allow you to change the sensitivity & dwell time of the PIR, leave them set at defaults for now.

When the PIR unit detects motion GPIO pin 4 reads 1, otherwise it reads 0. We can use that in a Python script to keep a log of movement events, and email them to a phone.



We also sell an adapter PCB board & PIR for £5.99, if you prefer not to use cables.
https://www.amazon.co.uk/dp/B00T4EFQBG

Configure Pi to run headless & configure WiFi.

As we'll be running the Pi headless there's little point having it load up the Desktop each time.

If you're using a Raspberry Pi Zero W or Pi 3 over a wireless connection, it’s a good idea to do the following at the Desktop -> Preferences ->Raspberry Pi Configuration:

- Enable SSH
- Change passwords for Pi and Root users to something secure and hard to guess.
- Localisation options → Change Timezone → Europe → London
- Enable Boot to CLI mode
- Reboot the Pi and login to the terminal

Now you should reboot to a terminal window.

If you're connecting your Pi to the internet using a wired connection skip these steps. If you're using WiFi enter commands:

```
sudo iwlist wlan0 scan
```

will show WiFi routers near your Pi

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

and add these lines at the bottom, using your routers SSID name and password

```
network={
ssid="testing"
psk="testingPassword"
}
```

Save the file, exit and reboot your Pi. It should now auto-attach to your wireless network when it boots up.

You can check your Pi has received an IP address from the router by entering

```
ifconfig
```

If you want to re-enable the Desktop environment later, you can run

```
sudo raspi-config
```

and choose option 3. Boot Options → Desktop.

Setting up a spare Gmail account, just for your Pi to use.

The simplest way of sending emails from your Pi, is to setup a new Gmail account for the Pi to use, even if you already have an existing Gmail account you use on your phone or PC. For one thing, it gets you 15GB of new cloud storage for your alerts & secondly it removes the complication of generating application specific passwords for other apps on your existing Gmail account.

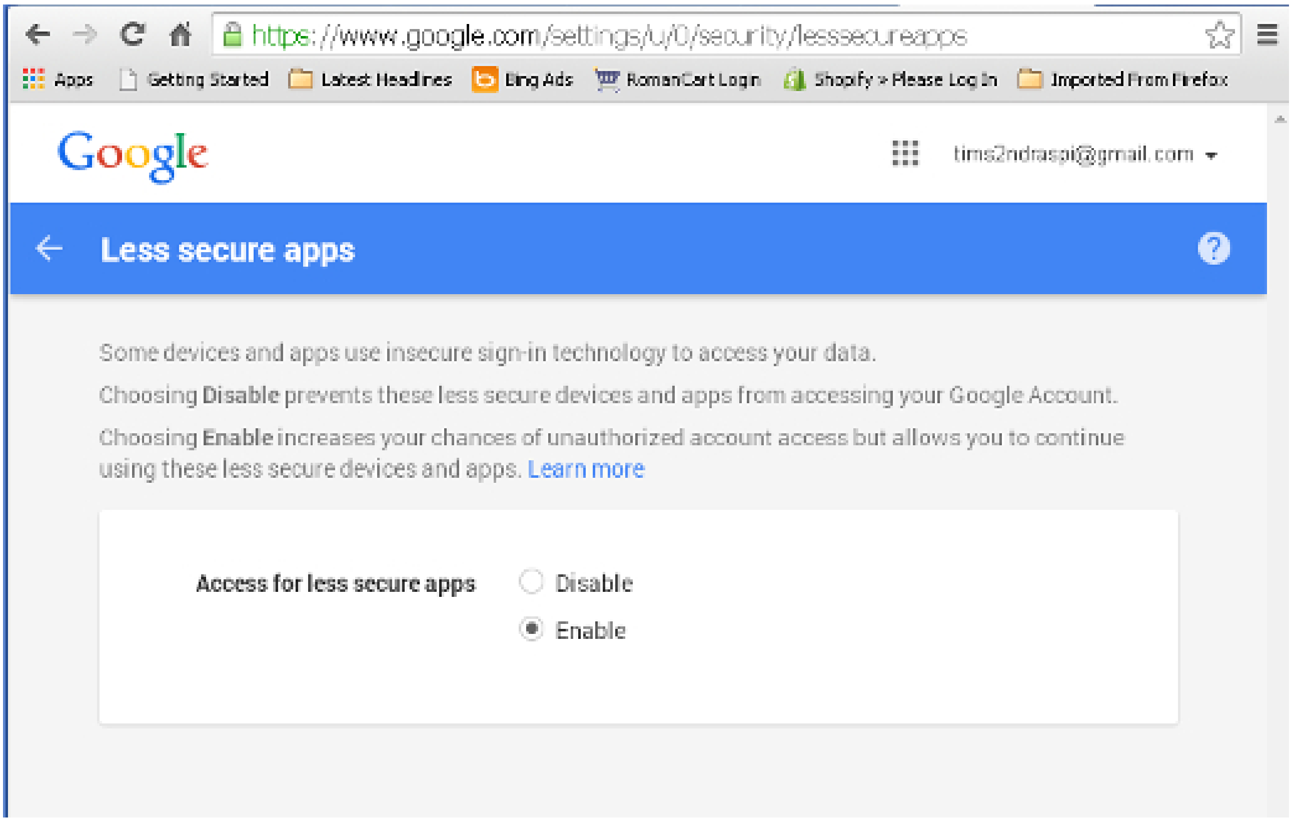
You need to create the new Gmail account in the web browser on your PC or Mac @

https://accounts.google.com/SignUp?service=mail

and note down the login & password for later. Don't use the # symbol in your password as it causes the Pi problems. Lower & upper-case letters & number combinations are always fine though.

Next, you need to set the new Gmail account to Enable “less secure apps”. While logged in go to:

https://www.google.com/settings/u/0/security/lesssecureapps



Less secure apps

Some devices and apps use insecure sign-in technology to access your data.

Choosing **Disable** prevents these less secure devices and apps from accessing your Google Account.

Choosing **Enable** increases your chances of unauthorized account access but allows you to continue using these less secure devices and apps. Learn more

Access for less secure apps ○ Disable ● Enable

Now we have a working email account, just for the Pi to use when sending emails. Any emails sent from the Pi will be backed-up in the Sent folder & you only need delete old emails if you get near to the 15GB limit. Emails from the Pi can be sent to any other email address on your phone or PC.

Setup and test the email utilities

We need to install & configure several mail applications from the internet. You also need to have made a new Gmail account, just for the Pi to use and enable “less secure app” access @ https://www.google.com/settings/u/0/security/lesssecureapps

```
sudo apt-get install ssmtp
sudo apt-get install mailutils
sudo apt-get install mpack
```

Setup default settings SSMTP.

```
sudo nano /etc/ssmtp/ssmtp.conf
```

```
AuthUser=your-gmail-account@gmail.com
AuthPass=your-user-password
FromLineOverride=YES
mailhub=smtp.gmail.com:587
UseSTARTTLS=YES
```

save file & exit. Make sure you changed the AuthUser and AuthPass variables to your own.

Send a test email with this command, substituting the email address below with your own.

```
echo "testing 1 2 3" | mail -s "Subject" you@yourdomain.co.uk
```

Test the PIR module

```
nano pir.py
```

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
Pir_Pin = 4
GPIO.setup(Pir_Pin, GPIO.IN)
while True:
   if GPIO.input(Pir_Pin):
      print("1")
      time.sleep(1)
   print("0")
   time.sleep(1)
```

```
python pir.py
```

If the PIR is wired up correctly and working, you'll see a 1 when the PIR is triggered, otherwise a 0.

Python script that emails each hour

The Python script below will keep a running total of PIR detections each hour, and email the full list to you each hour.

```
nano pir3.py
```

```
import time
import datetime
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
Pir_Pin = 4
GPIO.setup(Pir_Pin, GPIO.IN)

hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

while True:
   now = datetime.datetime.now()
   ch = now.hour
   if GPIO.input(Pir_Pin):
      print(ch)
      (hours[ch]) += 1
      print(hours[ch])
      time.sleep(5)
   if now.minute == 00 and now.second == 00:
      file = open("pirlog.txt","w")
      for i in range(0, 24):
         print(str(i) + '=' + str(hours[i]))
         file.write(str(i) + '=' + str(hours[i]) + '\n')
      file.close()
      os.system("cat pirlog.txt | mail -s 'pir log' me@gmail.com")
      time.sleep(2)
   if now.hour == 00 and now.minute == 00 and now.second == 30:
      hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      time.sleep(3)
```

Then save and exit. Remembering to change the email address. Run the script:

```
python pir3.py
```

You can make the script run automatically each time the Pi powers up by editing rc.local.

```
sudo nano /etc/rc.local
```

add these 2 lines before the exit 0 line at the end

```
cd /home/pi
python pir3.py
```

Python script that emails once a day

The Python script below will keep a running total of PIR detections each hour, and email the full list to you just before midnight. Change now.hour and now.minute values to suit.

```
nano pir4.py
```

```
import time
import datetime
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
Pir_Pin = 4
GPIO.setup(Pir_Pin, GPIO.IN)

hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

while True:
   now = datetime.datetime.now()
   ch = now.hour
   if GPIO.input(Pir_Pin):
      print(ch)
      (hours[ch]) += 1
      print(hours[ch])
      time.sleep(5)
   if now.hour == 23 and now.minute == 59 and now.second == 00:
      file = open("pirlog.txt","w")
      for i in range(0, 24):
         print(str(i) + '=' + str(hours[i]))
         file.write(str(i) + '=' + str(hours[i]) + '\n')
      file.close()
      os.system("cat pirlog.txt | mail -s 'pir log' me@gmail.com")
      time.sleep(2)
   if now.hour == 00 and now.minute == 00 and now.second == 30:
      hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      time.sleep(3)
```

Then save and exit. Remembering to change the email address. Run the script:

```
python pir4.py
```

You can make the script run automatically each time the Pi powers up by editing rc.local.

```
sudo nano /etc/rc.local
```

add these 2 lines before the exit 0 line at the end

```
cd /home/pi
python pir4.py
```

Python script that emails two people each hour

The Python script below will keep a running total of PIR detections each hour, and email the full list to two different email addresses each hour.

```
nano pir5.py
```

```
import time
import datetime
import os
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
Pir_Pin = 4
GPIO.setup(Pir_Pin, GPIO.IN)

hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

while True:
   now = datetime.datetime.now()
   ch = now.hour
   if GPIO.input(Pir_Pin):
      print(ch)
      (hours[ch]) += 1
      print(hours[ch])
      time.sleep(5)
   if now.minute == 00 and now.second == 00:
      file = open("pirlog.txt","w")
      for i in range(0, 24):
         print(str(i) + '=' + str(hours[i]))
         file.write(str(i) + '=' + str(hours[i]) + '\n')
      file.close()
      os.system("cat pirlog.txt | mail -s 'pir log' me@gmail.com")
      os.system("cat pirlog.txt | mail -s 'pir log' u@gmail.com")
      time.sleep(2)
   if now.hour == 00 and now.minute == 00 and now.second == 30:
      hours=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
      time.sleep(3)
```

Then save and exit. Remembering to change the email address. Run the script:

```
python pir5.py
```

You can make the script run automatically each time the Pi powers up by editing rc.local.

```
sudo nano /etc/rc.local
```

add these 2 lines before the exit 0 line at the end

```
cd /home/pi
python pir5.py
```

Where to buy the PIR sensors.

We supply PIR sensors on their own, with cables, and as full motion-detection kits with drill bits & nylon M2 nuts and bolts and software.

PIR sensor & cables £4.95 :	https://www.amazon.co.uk/dp/B00NFXBPU8
PIR sensor & PCB adapter £5.95	https://www.amazon.co.uk/dp/B00T4EFQBG
3x PIR sensors £8.99	https://www.amazon.co.uk/dp/B00NFWMW4C
5x PIR sensors £12.99	https://www.amazon.co.uk/dp/B00NFWB2E8
40 f-f cables £5.99	https://www.amazon.co.uk/dp/B00UUGVC9W
PIR Motion Alarm kit £11.99	https://www.amazon.co.uk/dp/B00GOPJJWU

It's also possible to re-purpose our £19 Wireless Internet Doorbell kit for Raspberry Pi as a portable panic button: https://www.amazon.co.uk/dp/B00UL3FRGS

We also supply these items in our eBay shop http://stores.ebay.co.uk/ConvertStuffUK

Buy any of these items from us and you'll also get full email support with the scripts in this document.

You can pull all the scripts down from our server to your Pi with

```
wget www.securipi.co.uk/sg.zip
unzip sg.zip
```

If you found these scripts useful, please follow @securipi on Twitter.