**Parent Detector**

How to use a Raspberry Pi to find out who's been in your room: make a parent detector that use motion detection to trigger video recording via the Raspberry Pi camera module.

**About the Passive Infrared (PIR) motion sensor**

In this project, we are going to be using a Passive Infrared (PIR) motion sensor. You have probably seen these before: they are often used in burglar alarm systems (the sensors placed in the corners of rooms are typically PIR ones). All objects whose temperatures are above absolute zero emit infrared radiation. Infrared wavelengths are not visible to the human eye, but they can be detected by the electronics inside one of these modules.

The sensor is regarded as passive because it doesn't send out any signal in order to detect movement. It adjusts itself to the infra red signature of the room it's in and then watches for any changes. Any object moving through the room will disturb the infra red signature, and will cause a change to be noticed by the PIR module.



We don't need to worry about the inner workings of the motion sensor. What we're interested in are the three pins on it, which can be used to connect it to the Raspberry Pi.

**Connect the PIR motion sensor**

Before booting your Raspberry Pi, connect the PIR module to the Raspberry Pi.

Using three female-to-female jumper cables, you'll need to connect each of the PIR sensor's connectors to the appropriate pins on the Raspberry Pi. Don't rely on the diagram to identify which pin is which, though: check the labels on the PIR.

1. Connect the one labelled VCC on the PIR sensor to the 5V pin on the Raspberry Pi. This provide power to the PIR sensor.
2. Connect the one labelled GND to a ground pin on the Raspberry Pi. This completes the circuit.
3. Connect the one labelled OUT to GPIO pin 4. This pin will output a voltage when motion is detected, that can then be received by the Raspberry Pi



**Test the PIR motion sensor**

We're going to use the Python programming language to write some code that will detect movement and print out some text; we can extend the program to involve the camera board later on.

The program is pretty simple. We will first set up the Raspberry Pi GPIO pins to allow us to use pin 4 as an input; it can then detect when the PIR module detects motion. We need to check the pin continually for any changes, so we use a while True loop for this. This is an infinite loop so the program will run continuously unless we stop it manually with Ctrl + F6.

Open IDLE (Menu>Programming>Python3 (IDLE)), create a new file (by using the File>New File menu options within IDLE) and copy in the code below.

from gpiozero import MotionSensor

pir = MotionSensor(4)

while True:

 if pir.motion\_detected:

 print("Motion detected!")

Save your file and press F5 to run it.

Everytime the PIR detects motion, you should see the words Motion detected! appear in the IDLE shell. Press Ctrl + F6 when you want to exit.



On the PIR module you should see two orange components with sockets that fit a Phillips screwdriver (see above). These are called potentiometers: they allow you to adjust the sensitivity of the sensor and the detection time. You should begin by setting the sensitivity to max and the time to min, but you can vary this later if you wish.

**Setting up the camera**

Before you can begin the project, you'll need to connect the Camera Module. You should do this before you boot the Pi, or, if your Pi is running, you should stop it by entering sudo halt.

Follow the instructions [here](https://www.raspberrypi.org/help/camera-module-setup/) to set up and test the Camera Module. Stop once you have successfully used a few of the example commands.

Next, if you have it, set up the camera mount. This will enable you to aim the camera at the right part of the room.

**Program the camera to preview on movement**

Now we're ready to extend our previous program to give it the ability to control the Camera Module. To start with, let's just make our program display what the camera can see when movement is detected; we can set up recording to a file later.

We first need to add the import picamera statement at the top; this allows your program to access the pre-made code which can control the camera module. We then declare the camera object camera, which provides all the camera control functions that we need to use. Then inside the while loop we can start the camera preview each time motion is detected.

Either copy and paste the code below, or enter it manually:

from gpiozero import MotionSensor

from picamera import PiCamera

camera = PiCamera()

pir = MotionSensor(4)

while True:

 pir.wait\_for\_motion()

 camera.start\_preview()

 pir.wait\_for\_no\_motion()

 camera.stop\_preview()

**Recording to a file and playing back**

We can now add a bit more code to allow us to record to a file for playback at a later stage. If there are multiple intruders in your room, you want to be able to capture them all and not just the most recent one. To do that, we need a way to generate a new file name automatically each time movement is detected. The easiest and safest way to do this is to make a file name out of the date and time.

For example, if the time now was 11 February 2017 at 10:24 and 18 seconds, the file name would be something like this: 2017-02-11\_10.24.18.h264. This uses the format of YEAR-MONTH-DAY\_HOUR.MINUTE.SECOND.h264; h264 refers to the format the video will be recorded in. It's the same as the format used by YouTube.

We need to import the datetime Python module and write a function to generate the filename. Then you simply use the commands to start and stop the recording using the generated file name. These should happen at the same time as the preview commands respectively.

Either copy and paste the code below, or enter it manually:

from gpiozero import MotionSensor

from picamera import PiCamera

from datetime import datetime

camera = PiCamera()

pir = MotionSensor(4)

while True:

 pir.wait\_for\_motion()

 filename = datetime.now().strftime("%Y-%m-%d\_%H.%M.%S.h264")

 camera.start\_recording(filename)

 pir.wait\_for\_no\_motion()

 camera.stop\_recording()

Save and run your script. You can quit it once it is tested and has taken some video.

**Playback**

If you look inside the file browser you should see that a few files have been generated. You can play the videos in the terminal (Ctrl+Alt+t) by typing:

omxplayer <file> -o hdmi

For example: omxplayer 2017-02-11\_10.24.18.h264 -o hdmi

Or you can rename the file to an mp4 and open it in the Epiphany web browser.

**What next?**

You have completed your parent detector, but why not try taking it to another level by using it in stealth mode?

* You could disable the red LED on the camera board which comes on when you start your Python program. This can be done by editing the Raspberry Pi configuration file. Enter the command below:

sudo nano /boot/config.txt

Add the following line to the end of the file:

disable\_camera\_led=1

Press Ctrl + O to save and Ctrl + X to quit. The changes will only take effect after a reboot, so enter the following command to do this:

sudo reboot

* If you want to leave the monitor connected and turned on while the program is running, it's a good idea to edit the Python code to disable the camera preview lines. Use the # sign at the start of a line to disable it.
* You could also start your Python program under a different login. To do this press ALT + F2 before you log in; this will show you a new login prompt, so log in there and start the Python program. Now if you press ALT + F1 to go back to the usual login prompt, it will appear as though the Raspberry Pi is innocently waiting for someone to log in.

[Creative Commons](https://www.raspberrypi.org/creative-commons/)

This learning resource is provided for free by the [Raspberry Pi Foundation](https://www.raspberrypi.org/) under a [Creative Commons](https://www.raspberrypi.org/creative-commons/) licence.

Find more at [raspberrypi.org/resources](https://www.raspberrypi.org/resources/) and [github.com/raspberrypilearning](https://github.com/raspberrypilearning).