Using the HC-SR501 PIR Motion Sensor – With Arduino & Raspberry Pi

Table of Contents [hide]

Motion sensing devices are commonly used in burglar alarms and automated lighting systems. **The HC-SR501 is a motion sensor module that is inexpensive and very versatile. It can be used all by itself or combined with a microcontroller or microcomputer to create a number of motion sensing products.**

In this article you'll learn how PIR (Passive Infrared) motion sensors work and how to use them with an Arduino or Raspberry Pi.

Introduction

There is no denying it – human beings are hot stuff! Humans (and animals) radiate body heat to such a degree that the temperature of a room rises when people enter it. If you've ever been crammed into a small room full of people.

That rise in heat level experienced when a person enters a room can be detected and used as a trigger for a number of devices. Burglar alarms and intruder detectors are obvious applications. Another common reason to detect the presence of people is for energy-saving lighting systems that only come on when a room is occupied.

Of course there are a number of ways to detect people. We have already seen the HC-SR04 Ultrasonic Sensor, this device can be used to detect an intruder by reflecting an ultrasonic signal back to a sensor. Infrared light can also be used either by reflecting or breaking a beam, many burglar alarm systems work this way and microwaves can also be used to detect items composed of water, and people are mostly water.

Another way of detecting people, and the one we will be focusing on today, is by using a "Passive Infrared" or "PIR" sensor. Unlike other sensors which emit and then detect infrared light a PIR sensor does not emit anything. Rather it measures infrared energy in a room and is triggered upon a sudden change in it, such as the change that occurs when a warm blooded mammal enters the room.

In this article and the accompanying video we will see **how PIR sensors work** and we'll experiment with a very useful **PIR Sensor module – the HC-SR501**

Passive Infrared Sensors

Any object with a temperature above absolute zero emits heat energy in the form of radiation. This radiation isn't visible to the human eye because it radiates at infrared wavelengths, below the spectrum that people can see. We discussed infrared light when we worked with IR Remote Controls.

A measurement of IR energy is not the same as measuring temperature. Temperature measurements rely upon thermal conductance, so when a person enters a room they don't immediately raise the room temperature (unless they happen to be on fire). They do, however, have a unique IR "signature" or "footprint" due to their body temperature and it is that signature that a PIR sensor is looking for.

The infrared "footprint" can either be detected directly or it can be reflected from a surface that reflects infrared light.

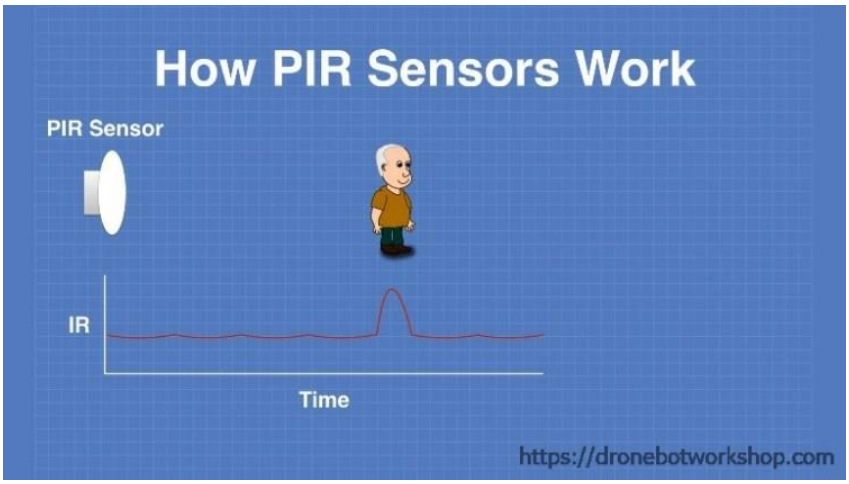
How PIR Motion Detectors work

A Passive Infrared (PIR) motion detector works by first adjusting itself to the "normal" IR heat signature within its detecting area. It then looks for abrupt changes to this IR signature, changes that occur because a living being has entered or moved within the monitored area.

To sense infrared energy, the detector makes use of a pyroelectric sensor. This is a device that generates an electrical current in response to receiving IR energy.

Because the detector does not emit a signal (such as the previously mentioned Ultrasonic sensor does) it is called "passive". It just sits there, listening for a change in the ambient IR energy level.

When a change is detected the PIR motion detector will trigger an alert by changing its output signal.



Fresnel Lenses

In order to increase the sensitivity and effectiveness of a PIR sensor a method of focusing IR energy onto the device is required. This is usually accomplished with a series of lenses that are focused upon the sensor.

The cheapest and most practical type of lenses to use are “Fresnel Lenses”.

A Fresnel Lens is generally constructed of plastic and is very compact, making it ideal for low-cost PIR sensors. The “dome” that covers most of these sensors actually consists of several small Fresnel lenses. While the plastic on the dome may appear to be translucent to the naked eye it is actually fully transparent to infrared light so it also serves as an IR filter of sorts.

Interestingly the Fresnel lens was originally developed in 1823 and was used to focus the light beams in lighthouses. It was invented by a French physicist and engineer Augustin-Jean Fresnel.

Fresnel lenses are also used in video projectors and to increase the efficiency of solar cells. These lenses also were popular in the 1970’s as flexible plastic ones were sold as a television screen enlarging device!

HC-SR501 PIR Motion Detector

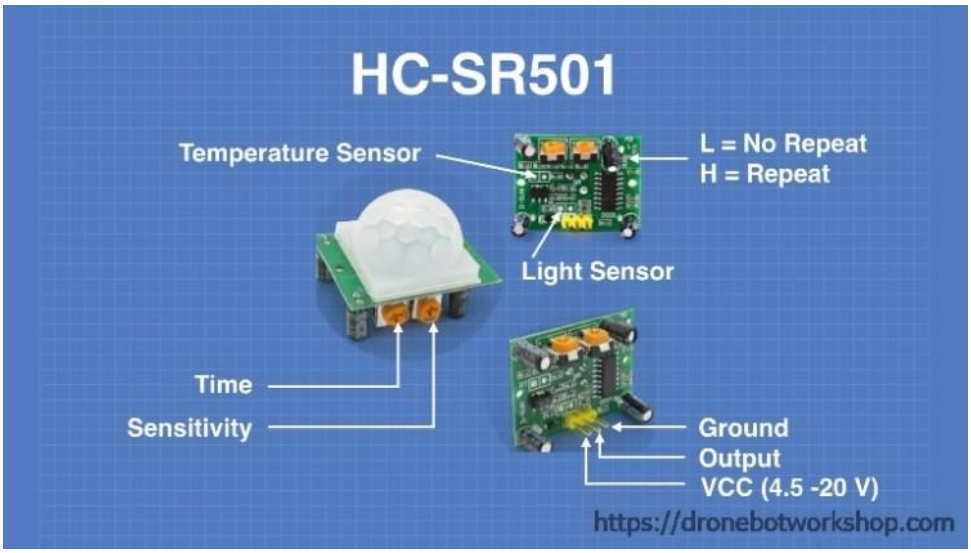
The HC-SR501 is an inexpensive PIR sensor that is readily available on eBay, Amazon and at your local electronics store. It is a completely self-contained unit capable of operating on its own or of being interfaced to a microcontroller or microcomputer.

The HC-SR501 has a sensitivity adjustment that allows it to sense movement from 3 to 7 meters away, enough to cover an average sized room. Its output can be adjusted to stay high for a period from 3 seconds to five minutes. It has a built-in voltage regulator so it can be powered by any DC voltage from 4.5 to 20 volts and it draws a minimal amount of current.

Connections and Adjustments

The HC-SR501 has a 3-pin connector that interfaces it to the outside world. The connections are as follows:

- **VCC** – This is a positive DC voltage input from 4.5 to 20 VDC.
- **OUTPUT** – This is a 3.3-volt logic output. LOW indicates no detection; HIGH means someone has been detected.
- **GND** – This is the Ground connection.
-



There are also two potentiometers on the board to adjust a couple of parameters:

- **SENSITIVITY** – This sets the maximum distance that motion can be detected. It ranges from 3 meters to approximately 7 meters. The topology of your room can affect the actual range you achieve.
- **TIME** – This sets how long that the output will remain HIGH after detection. At minimum it is 3 seconds, at maximum it is 300 seconds or 5 minutes.

Finally, the board has a jumper (on some models the jumper is not soldered in). It has two settings:

- **H** – This is the Hold or Repeat setting. In this position the HC-SR501 will continue to output a HIGH signal as long as it continues to detect movement.
- **L** – This is the Intermittent or No-Repeat setting. In this position the output will stay HIGH for the period set by the TIME potentiometer adjustment.

Operational Considerations

As with most PIR sensors the HC-SR501 requires some time to acclimatize to the infrared energy in the room. This takes from 30 to 60 seconds when the sensor is first powered up. In addition, the sensor has a “reset” period of about 5 or 6 seconds after making a reading. During this time, it will not detect any motion.

When designing a system based upon the HC-SR501 you will need to take these delay periods into account. The Arduino sketch I will be showing you later does exactly that.

Optional Components

The HC-SR501 circuit board has solder pads for two additional components. These are usually labeled, note that on some boards the labels may be covered by the "dome" lens on the side opposite the components.

- **RT** – This is meant for a thermistor or temperature-sensitive resistor. Adding this allows the HC-SR501 to be used in extreme temperatures, it also increases the accuracy of the detector to some degree.
- **RL** – This connection is for a Light Dependent Resistor (LDR) or Photo resistor. By adding this component the HC-SR501 will only operate in darkness, a common application for motion-sensitive lighting systems. I will be illustrating its use further in this article and in the accompanying video.

The additional components can be soldered directly to the board or extended to remote locations using wires and connectors.

Demo 1 – HC-SR501 on its own

For our first experiment we will use the HC-SR501 on its own to illustrate how useful it is by itself and how easily it can be interfaced to 5-volt logic despite using a 3.3-volt output.

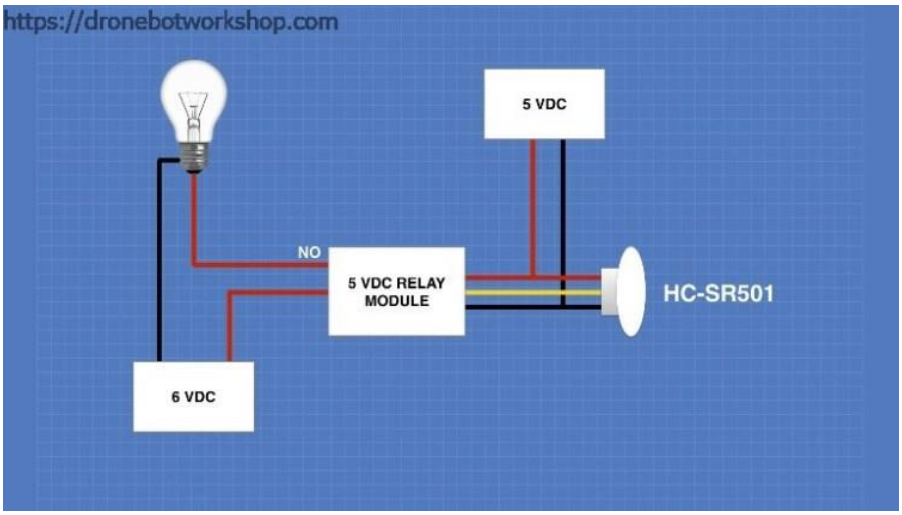
The circuit for this experiment is very simple. The HC-SR501 will be driving a 5-volt relay module, these devices are very common and consist of a relay as well as an opto-isolator that allows the relay to be driven directly from a logic circuit.

If you wanted to use a stand-alone relay you would need to place a diode across it to prevent the relays induced current from damaging the HC-SR501. Also, relays that activate on just 3.3 volts are a bit hard to find. The relay module makes things a lot easier and the opto-isolator adds an additional margin of safety should you decide to use it to control line-voltage devices.

In this demonstration I will be powering the HC-SR501 with a 5-volt bench power supply since that is also the power supply that the relay requires. As the HC-SR501 can accept any voltage from 4.5 to 20 volts you could use a higher voltage provided that you had a zener diode or voltage regulator to power the relay module. Using the 5-volt supply just seemed a lot easier!

I used a 6-volt lamp and lantern battery on the other side of the relay, note that these are connected to the NO (Normally Open) contacts on the relay so that the lamp is illuminated when the relay is closed. You could use a different load (i.e. a buzzer) and a different voltage a so long as it was within the capabilities of the relay contacts. I don't recommend experimenting with line

voltage on the workbench, however if you were to build this circuit into an insulated enclosure you could use it to control a desk lamp or other electrical appliance. Just be sure to keep safety first!
Here is the connection diagram for our first demonstration:



The operation of this circuit is very simple. After its warm-up period the HC-SR501 will start looking for motion. When it detects movement it will send its output to 3.3 volts. This is sufficient to be seen as a “digital 1” by the relay module and it will trigger, this turning on the lamp.
I urge you to experiment with the sensitivity and time adjustments as well as the position of the jumper to better understand how the affect the operation of the HC-SR501.
As you can see this is an entirely practical circuit built without the use of any microcontroller or microcomputer.

Demo 2 – Adding a Photoresistor

The next demonstration uses the same circuit as the previous one. In this experiment we will add an LDR (Light Dependent Resistor) or photoresistor to the HC-SR501 board and observe how the operation is affected.
The pads to solder in the LDR are located right next to the output connector. On some boards they are labeled “RL”.
When I performed the experiment (you can see it in the video accompanying this article) I soldered a pair of male header pins onto the HC-SR501. This allowed me to use a standard

Dupont connector to run to my LDR, which I placed on a solderless breadboard. You may also solder the LDR directly to the board or run a couple of wires to it – the choice is yours.
However, you do it make certain that the LDR is exposed to the room lighting. You'll also want to keep it shielded from the light from the lamp you are using as a load.
This diagram illustrates how the LDR is connected to the HC-SR501.

Once you have your LDR installed power up the circuit and let it get adjusted to the room. If all is working well (and if you have the lights on) nothing should happen!
The LDR prevents the HC-SR501 from triggering when the room is lighted.
Now turn of the lights and check it out. The HC-SR501 should now work, triggering whenever it spots some human or animal activity in the room.
You've just constructed a light-sensitive motion detector, perfect for illuminating those dark corners at night. It also makes an excellent emergency light in case of power failures.
And once again no microcontroller was required to build a highly useful and practical device.
However this is not to say that the HC-SR501 couldn't benefit from the addition of some external intelligence. In our next experiment we will interface it to an Arduino.

Demo 3 – HC-SR501 with Arduino

While the HC-SR501 is a useful device on its own it can also be used as an input to a microcontroller. You can then do all sorts of things when you detect motion – turn on a light, drive a robot or a myriad of other applications.

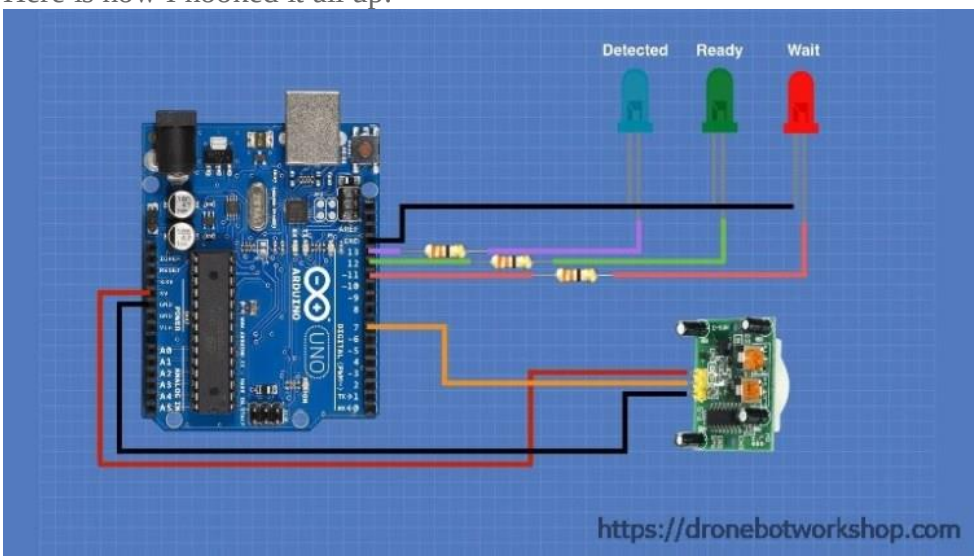
A microcontroller, like an Arduino, can also be used to allow for the acclimatization period and reset period that is inherent in the HC-SR501. This way the device can be more accurate as you won't attempt to sense fore motion when the sensor isn't ready.

You could also interface multiple HC-SR501 PIR sensors to an Arduino, allowing you to monitor for motion in different locations or from different directions.

In the following demonstration we will hook up a single HC-SR501 to an Arduino. We will also attach three LEDs to the Arduino to display the status of the motion sensor as follows:

- **Waiting** – This LED indicates that the sensor is not ready, either because has just been turned on and is acclimatizing or because it just finished triggering and is being reset. I used a red LED for this indicator.
- **Ready** – This LED indicates that the sensor is ready to detect motion. I used a green LED in my experiment for this.
- **Triggered** – This LED is illuminated for 3 seconds when the sensor has been triggered. In my experiment I used a blue LED here. You could also drive an external output (like the relay module we used earlier) instead of the LED if you wish.

Here is how I hooked it all up:



You will want to set the jumper on the HC-SR501 to the "L" (no repeat) position for this to work correctly. You'll also need to set the TIME to the minimum of 3 seconds, turn the TIME potentiometer as far counterclockwise as it will go. Set the sensitivity anywhere you like, I suggest midpoint for starters.

Now that you have it all wired up you will need some code to make it work. Here is the sketch you can use:

```
/*
  HC-SR501 Motion Sensor Demonstration 1
  HC-SR501-Demo1.ino
  Motion Sensor with Delay
  Set sensor for 3-second trigger
  DroneBot Workshop 2018
  https://dronebotworkshop.com
*/

// Define pins for LEDs
int detectedLED = 13;
int readyLED = 12;
int waitLED = 11;

// Input from Motion Sensor
int pirPin = 7;

// Variable for Motion Detected
int motionDetected = 0;

// Variable to store value from PIR
int pirValue;


void setup() {

	// Setup LEDs as Outputs
	pinMode(detectedLED, OUTPUT);
	pinMode(readyLED, OUTPUT);
	pinMode(waitLED, OUTPUT);

	// Setup PIR as Input
	pinMode(pirPin, INPUT);

	// Initial 1 Minute Delay to stabilize sensor
	digitalWrite(detectedLED, LOW);
	digitalWrite(readyLED, LOW);
	digitalWrite(waitLED, HIGH);
	delay(60000);
	digitalWrite(readyLED, HIGH);
	digitalWrite(waitLED, LOW);

}

void loop() {

	// Get value from motion sensor
	pirValue = digitalRead(pirPin);detectedPin
	// See if motion Detected
	if (pirValue == 1){
		// Display Triggered LED for 3 seconds
		digitalWrite(detectedLED, HIGH);
		motionDetected = 1;
		delay(3000);
	} else {
		digitalWrite(detectedLED, LOW);
	}
	// Add delay after triggering to reset sensor
	if (motionDetected == 1) {
		// After trigger wait 6 seconds to re-arm
		digitalWrite(detectedLED, LOW);
		digitalWrite(readyLED, LOW);
```

```
63                      digitalWrite(waitLED, HIGH);
64                      delay(6000);
65                      digitalWrite(readyLED, HIGH);
66                      digitalWrite(waitLED, LOW);
67                      motionDetected = 0;
68              }
69
70  }
```

Remember you don't need to copy this out by hand or even copy it from the code window as all of the code for this article is in the Resources box at the bottom.
Once you have the sketch compiled and loaded to your Arduino the fun starts! You should observe the red LED (the "waiting" LED) illuminating for a minute while the sensor is getting adjusted to the ambient IR in the room.
After the minute is up the "ready" LED (green in my case) will come on and the "waiting" LED will extinguish. The sensor is ready to detect motion.
As soon as motion has been detected the "triggered" LED (or whatever you are driving instead of a LED) will be activated for three seconds.
After the activation period is over the "waiting" LED will be illuminated again, the "ready" and "triggered" LEDs will be off. Once the 6 second reset period has elapsed the sensor will be ready for action again and the "ready" LED will turn back on.

Demo 4 – HC-SR501 with Raspberry Pi

We will finish our demonstrations of the HC-SR501 by using a Raspberry Pi with a camera to construct a cool intruder detector. Our little project will not only detect the presence of an intruder; it will also take either a picture or a video of the culprit!
You will, of course, need both a Raspberry Pi and a Raspberry Pi Camera to make this happen. Any type of Raspberry Pi can be used, I used a Raspberry Pi Zero W but the Zero, Model 2 or Model 3 are also suitable.

Setting up the Raspberry Pi

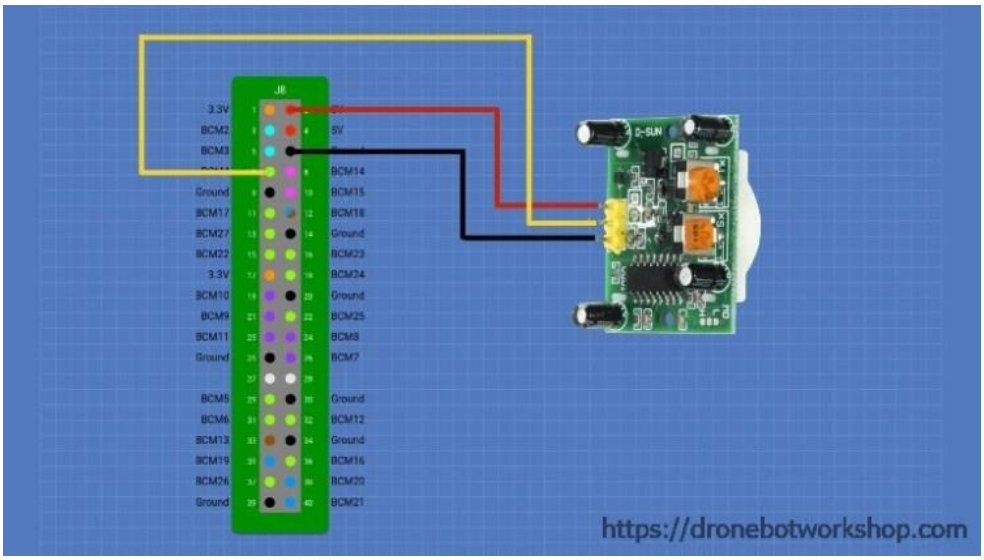
Load the latest version of Raspbian onto a microSD card and insert it into the Raspberry Pi. If you need instructions for doing this follow the excellent articles on the Raspberry Pi website. You can also get the latest version of Raspbian there, when I performed the experiment I used Raspbian Stretch.
Connect your camera (any Raspberry Pi camera will suffice) to the CSI connector on the Raspberry Pi. If you are using a Zero or Zero W the CSI connector is smaller so you'll either

need a Zero-specific camera or an adapter cable or connector. Make sure to orient the cameras ribbon cable properly.

Next you'll want to connect your HC-SR501 to the Raspberry Pi. This is pretty simple, the connections are as follows:

- VCC – Connect to the Raspberry Pi 5-volt output. This is GPIO pin 2
- OUTPUT – Connect this to pin 7 on the GPIO. This is GPIO In/Out number 4 programmatically.
- GND – Connect this to a Ground connection on the GPIO. Pin 6 is a good choice.

If your Raspberry Pi has a GPIO connector already soldered in you can use a 3-pin female to female Dupont cable to easily make these connections



Attach a keyboard, mouse and an HDMI monitor to your Raspberry Pi. Now power it up.

When you first boot up you'll be greeted by the Raspbian desktop. You'll need to set a few things here before we can begin.

Click the "Raspberry" icon on the top taskbar and choose "Preferences" Then from the sub-menu that appears choose "Raspberry Pi Configuration".

In the configuration screen go to the "Localisation" tab. In here you will want to set at minimum your Locale and Keyboard (the Raspberry Pi defaults to a UK keyboard which is not completely compatible with a US one).

Now go to the "Interfaces" tab. You will need to enable the Camera for this project.

After you finish setting your configuration click OK. You will be prompted to reboot, do that otherwise the changes won't be saved.

After you reboot and are back on the desktop you are ready to begin.

Testing the Camera

It's a good idea to test your camera to be sure it is connected properly and is working. This can easily be done from the Terminal window.

Open the Terminal by clicking the icon on the top taskbar. At the command prompt type the following command (note this is all lowercase) :

raspistill – k

Press *Enter*. If all is working you will see an image from the camera displayed on your monitor. Once you are satisfied that the camera works and are done admiring yourself press "*X*" and then "*Ente*r" on your keyboard to close the camera connection.

You can now close the terminal window if you like as we are done with it.

Testing the HC-SR501

Next we will check our connection to the HC-SR501. To do this we will run a short Python script, which you will find here (and in the Resources link at the bottom of this article).

This script loads the Motion Sensor components of the GPIO library and then tests to see if it can detect motion. If it detects movement it will display an indication on the Python Shell screen.

```
1  from gpiozero import MotionSensor
2  from picamera import PiCamera
3
4  pir = MotionSensor(4)
5  camera = PiCamera()
6
7  while True:
8      pir.wait_for_motion()
9      if pir.motion_detected:
10         print("Good Looking Person Detected")
11         camera.start_preview()
```

To run this script go to the "Raspberry" icon again and select "programming". From here open up the "Python 3 (IDLE)" application. This will open the IDLE programing environment in a "Python Shell" box.

Click "File" and then "New" to open up a blank coding window. Enter the code as it appears above paying close attention to observing proper indentation as this is critical in Python programming. Dave the file with an appropriate filename.

If you have a network-connected Raspberry Pi or a USB Flash drive you can also load the "*hc-sr50-cam-test.py*" script that I've provided in the link in the Resource box to avoid typing it in manually.

Once you have the script you can run it by pressing the F5 key. If you typed it in manually and have a typo or indentation error, you'll be informed of it. Otherwise you will see an indication in the Python Shell window that the script is running.

After a warm-up period the motion sensor should start working. When it detects any motion you'll see several lines of "Good looking person detected" printed in the shell. As we have the HC-SR501 set for a 3-second output this will continue to print for three seconds. After that the sensor will take a few seconds to reset and then start over.
To stop the script press Ctrl-F6.
Assuming everything worked we can now begin to write the script(s) for our motion sensitive intruder camera.

Alarm with Pictures

The next Python script we will be running is the one that will take a picture with our camera whenever motion is detected by the HC-SR501.
Open a new script file or load the one supplied in the resources link called "*intruder_camera.py*". The code is as follows:

```
1  from gpiozero import MotionSensor
2  from picamera import PiCamera
3  import time
4  import datetime
5
6  # Create object for PIR Sensor
7  # PIR Sensor is on GPIO-4 (Pin 7)
8  pir = MotionSensor(4)
9
10 # Create Object for Camera
11 camera = PiCamera()
12
13 # Function to create new Filename from date and time
14 def getFileName():
15              return datetime.datetime.now().strftime("%Y-%m-%d_%H.%M.%S.jpg")
16
17
18 while True:
19     # Get a Filename
20     filename = getFileName()
21     # Wait for a motion to be detected
22     pir.wait_for_motion
23     # Print text to Shell
24     print("Sneaky Person Alert!!")
25     # Preview camera on screen until picture is taken
26     camera.start_preview()
27     # Take a picture of intruder
28     camera.capture(filename)
29     camera.stop_preview()
30     # Wait 10 seconds before repeating
31     time.sleep(10)
```

This code loads a number of libraries, one for the Motion Sensor, one for the Camera and two time related libraries. It then creates objects to represent the PIR sensor and the Camera.
It defines a function called "*getFileName*" that generates a unique name for the picture file. The name is based upon the current date and time and has a "jpg" extension.

In the main code block we define a new filename and then wait for motion to be detected. When it is detected we type a “Sneaky Person Alert” to our Python shell screen and start the camera preview, which will display the cameras output on our screen. We then take a picture that is saved under the new filename we just generated and then close the preview monitor.
After doing that we wait 10 seconds before looking for motion again.
Run the program by pressing F5 and get a few pictures taken. Then stop the program by pressing Ctrl-F6.
Now open up the file manager and look in the “pi” folder (the file manager defaults to this folder). You should see a few files with a “.jpg” extension, they will be named with a timestamp. Click on one of them to open the image viewer.
If all is working, you’ll see an image of the sneaky person that the sensor detected!

Alarm with Video

We can modify the previous sketch quite easily to shoot video of the intruder we detected. The code is as follows, notice how similar it is to the previous code:

```
1  from gpiozero import MotionSensor
2  from picamera import PiCamera
3  import time
4  import datetime
5
6  # Create object for PIR Sensor
7  # PIR Sensor is on GPIO-4 (Pin 7)
8  pir = MotionSensor(4)
9
10 # Create Object for Camera
11 camera = PiCamera()
12
13 # Function to create new Filename from date and time
14 def getFileName():
15              return datetime.datetime.now().strftime("%Y-%m-%d_%H.%M.%S.h264")
16
17
18 while True:
19     # Get a Filename
20     filename = getFileName()
21     # Wait for a motion to be detected
22     pir.wait_for_motion
23     # Print text to Shell
24     print("Jellybean thief detected!")
25     # Preview camera on screen during video
26     camera.start_preview()
27     # Start recording video
28     camera.start_recording(filename)
29     # Record for 10 seconds
30     camera.wait_recording(10)
31     # Stop preview and recording
32     camera.stop_preview()
33     camera.stop_recording()
34     # Wait 25 seconds and repeat
35     time.sleep(25)
```

Note the small change to the getFileName function, the extension has been changed to ".h264". This is a format of video that the Pi Camera library defaults to and it can be played on the Raspbian system using the "Omx Player" application or on another computer using VLC video player.

In the main program section there are just a few changes to record video instead of snapping a photo. We use the "*start_recording*" function of the camera library to start recording our video with the filename we have just determined. We then use "*wait_recording*" to record for 10 seconds, you can change the length of the video here if you wish. Then finally "*stop_recording*" ends the video and saves the file.

After we finish we wait 25 seconds before we do it all again. If you like you can adjust this delay to suit your preferences.

Once again we run this script with the F5 key and stop it with Ctrl-F6.

After we finish you'll find some files in the "pi" folder that have a ".h264" extension. These can be played back a number of ways:

- Open the Terminal and type "*omxplayer*", a space and the name of the file. This works but the filenames are very long and cumbersome.
- Right-click on the file in the file explorer and choose "*Open With*". The "*Choose Application*" box will appear. Choose the "*Custom Command*" tab and type "*omxplayer*" into the "*Command Line to Execute*" box.
- If your Pi is network conected or if you have a USB flash drive copy the files and then use VLC Media Player to view them on an external computer.

Either way you should see a video of the nasty person who snuck in to pilfer your jelly beans!

Conclusion

As you can see the HC-SR501 is a very versatile sensor that is pretty capable all on its own. By interfacing it to some external electronics like an Arduino or Raspberry Pi you can expand upon its versatility even further.

So what will you construct with your motion sensor? Please let me know in the comments below. Now get moving and build something!

Parts List

Here are some components that you might need to complete the experiments in this article. Please note that some of these links may be affiliate links, and the DroneBot Workshop may receive a commission on your purchases. This does not increase the cost to you and is a method of supporting this ad-free website.

COMING SOON!

Resources

Code – All of the Arduino and Raspberry Pi code for this article
PIR Sensors – Wikipedia article on Passive Infrared sensors
Physical Computing with Raspberry Pi – Using devices on the Raspberry Pi GPIO
Raspberry Pi Downloads – Get the latest version of Raspbian here