PIR Motion Sensor

Created by lady ada

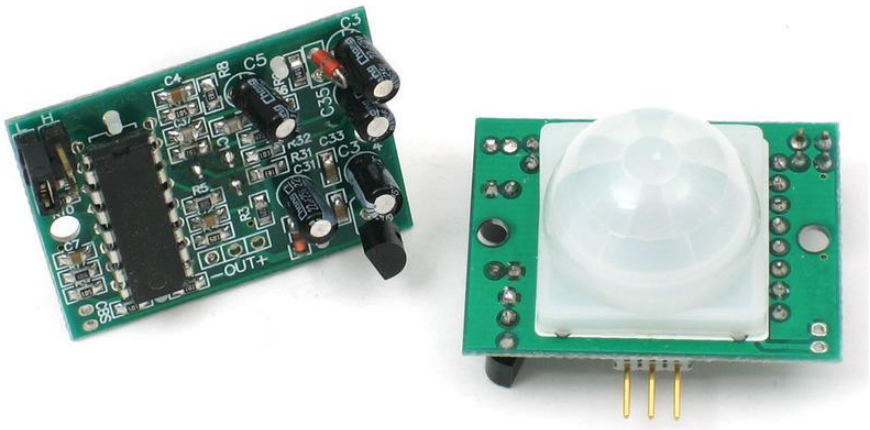
https://learn.adafruit.com/pir-passive-infrared-proximity-motion-sensor

Last updated on 2022-12-01 01:49:11 PM EST

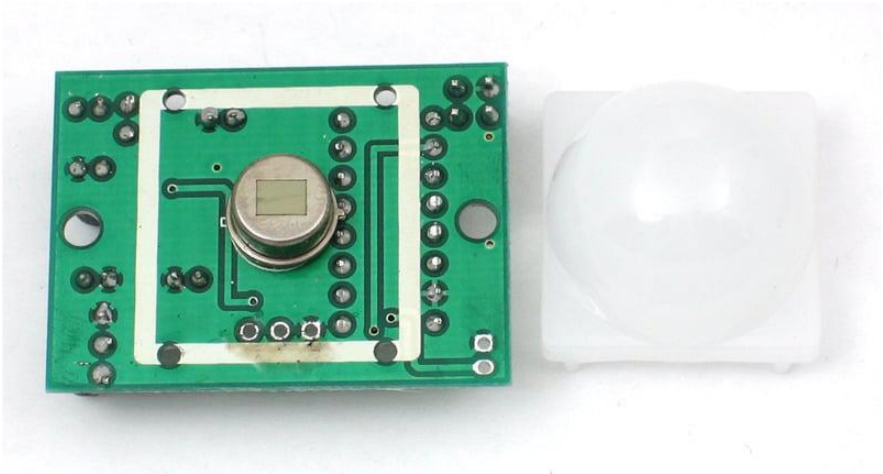
Table of Contents

Overview

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

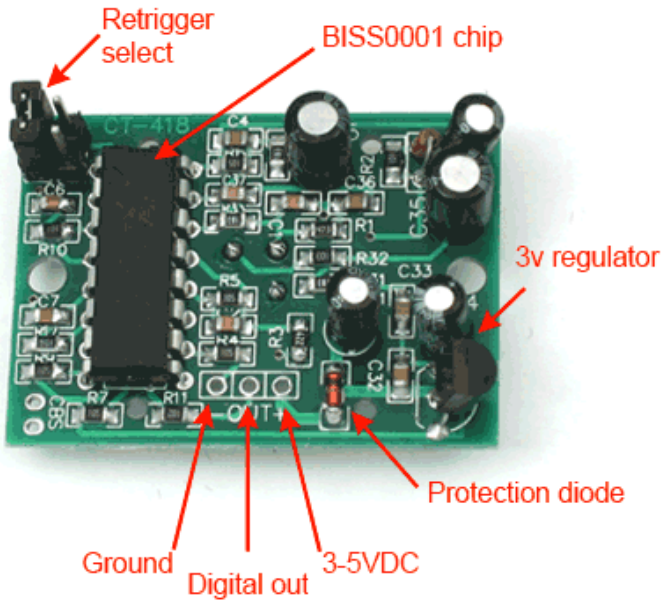


PIRs are basically made of a pyroelectric sensor () (which you can see below as the round metal can with a rectangular crystal in the center), which can detect levels of infrared radiation. Everything emits some low level radiation, and the hotter something is, the more radiation is emitted. The sensor in a motion detector is actually split in two halves. The reason for that is that we are looking to detect motion (change) not average IR levels. The two halves are wired up so that they cancel each other out. If one half sees more or less IR radiation than the other, the output will swing high or low.

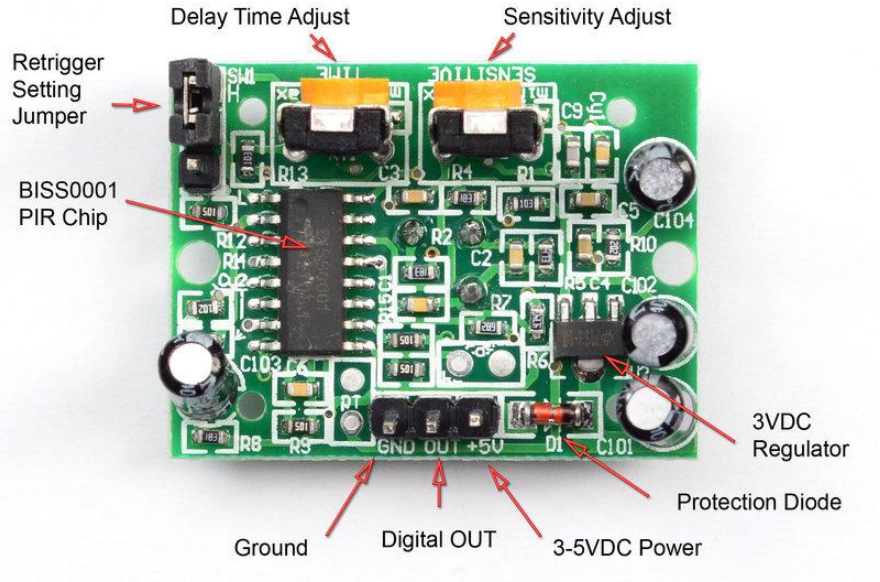


Along with the pyroelectic sensor is a bunch of supporting circuitry, resistors and capacitors. It seems that most small hobbyist sensors use the BISS0001 ("Micro Power PIR Motion Detector IC") (), undoubtedly a very inexpensive chip. This chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor.

Our older PIRs looked like this:



Our new PIRs have more adjustable settings and have a header installed in the 3-pin ground/out/power pads



For many basic projects or products that need to detect when a person has left or entered the area, or has approached, PIR sensors are great. They are low power and

low cost, pretty rugged, have a wide lens range, and are easy to interface with. Note that PIRs won't tell you how many people are around or how close they are to the sensor, the lens is often fixed to a certain sweep and distance (although it can be hacked somewhere) and they are also sometimes set off by housepets. Experimentation is key!

Some Basic Stats

These stats are for the PIR sensor in the Adafruit shop which is very much like the Parallax one () . Nearly all PIRs will have slightly different specifications, although they all pretty much work the same. If there's a datasheet, you'll want to refer to it

- Size: Rectangular
- Price: $10.00 at the Adafruit shop ()
- Output: Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor.
- Sensitivity range: up to 20 feet (6 meters) 110° x 70° detection range
- Power supply: 5V-12V input voltage for most modules (they have a 3.3V regulator), but 5V is ideal in case the regulator has different specs
- BIS0001 Datasheet () (the decoder chip used)
- RE200B datasheet () (most likely the PIR sensing element used)
- NL11NH datasheet () (equivalent lens used)
- Parallax Datasheet on their version of the sensor ()

More links!

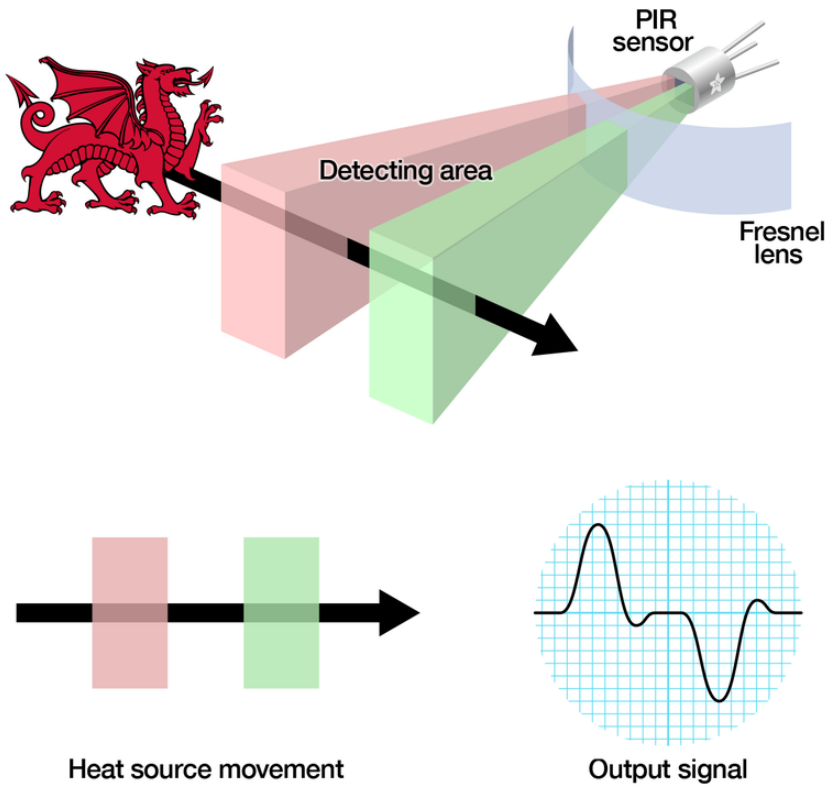
- A great page on PIR sensors from GLOLAB \\ ()

How PIRs Work

PIR sensors are more complicated than many of the other sensors explained in these tutorials (like photocells, FSRs and tilt switches) because there are multiple variables that affect the sensors input and output. To begin explaining how a basic sensor works, we'll use this rather nice diagram

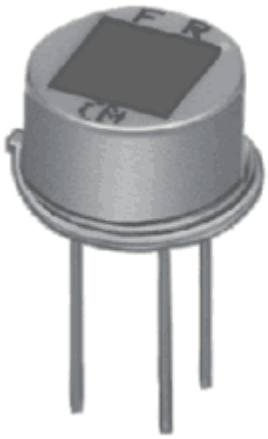
The PIR sensor itself has two slots in it, each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much and so we see that the two slots can 'see' out past some distance (basically the sensitivity of the sensor).

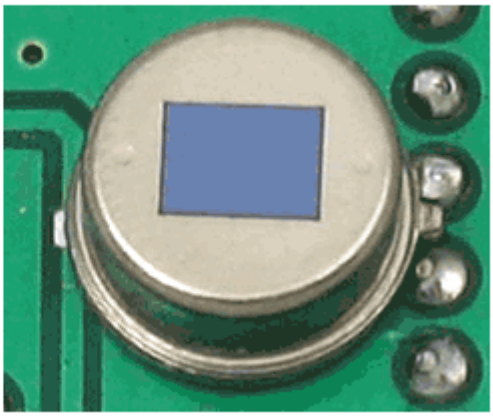
When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.



The PIR Sensor

The IR sensor itself is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity. There is a window made of IR-transmissive material (typically coated silicon since that is very easy to come by) that protects the sensing element. Behind the window are the two balanced sensors.





Left image from Murata datasheet ()

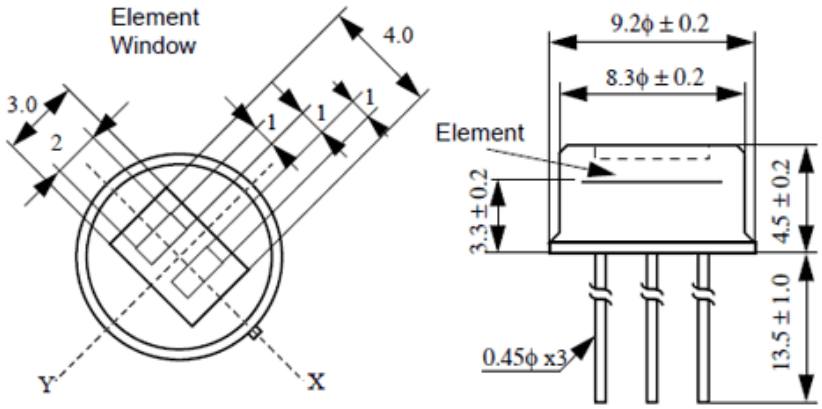


Image from RE200B datasheet ()

You can see above the diagram showing the element window, the two pieces of sensing material

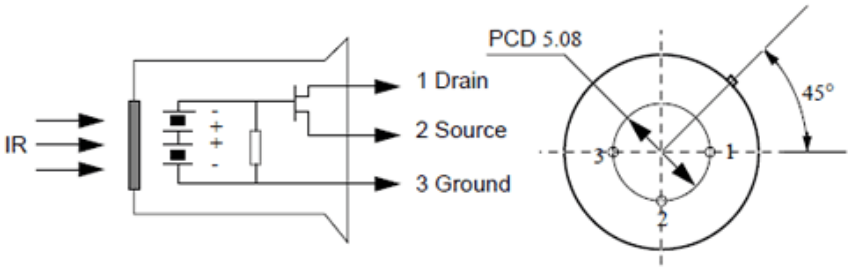


Image from RE200B datasheet ()

This image shows the internal schematic. There is actually a JFET inside (a type of transistor) which is very low-noise and buffers the extremely high impedence of the sensors into something a low-cost chip (like the BIS0001) can sense.

Lenses

PIR sensors are rather generic and for the most part vary only in price and sensitivity. Most of the real magic happens with the optics. This is a pretty good idea for manufacturing: the PIR sensor and circuitry is fixed and costs a few dollars. The lens costs only a few cents and can change the breadth, range, sensing pattern, very easily.

In the diagram up top, the lens is just a piece of plastic, but that means that the detection area is just two rectangles. Usually we'd like to have a detection area that is much larger. To do that, we use a simple lens () such as those found in a camera: they condenses a large area (such as a landscape) into a small one (on film or a CCD sensor). For reasons that will be apparent soon, we would like to make the PIR lenses small and thin and moldable from cheap plastic, even though it may add distortion. For this reason the sensors are actually Fresnel lenses ():

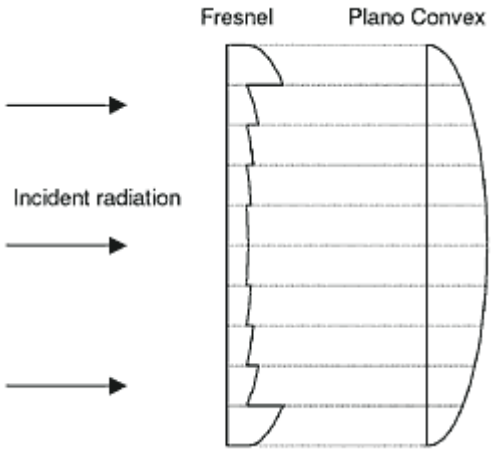


Image from Sensors Magazine ()

The Fresnel lens condenses light, providing a larger range of IR to the sensor.

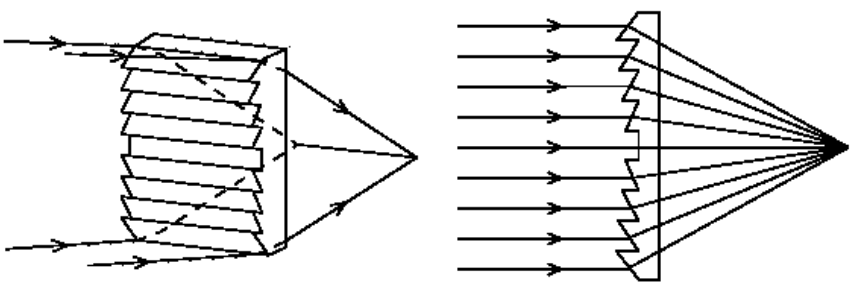


Image from BHlens.com ()

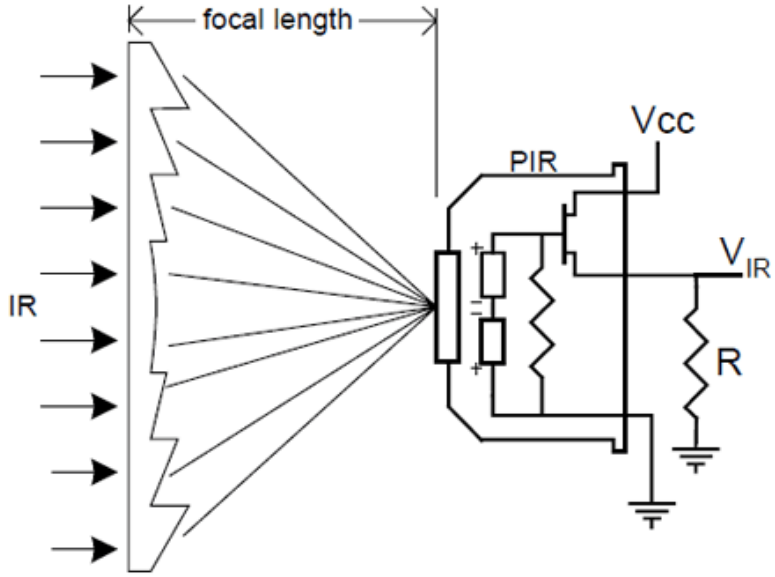
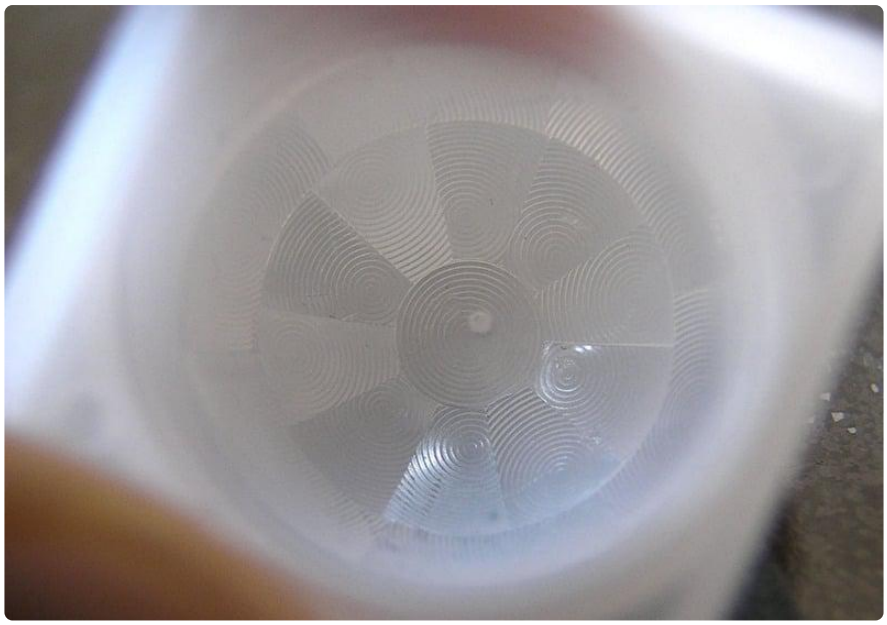


Image from Cypress appnote 2105 ()

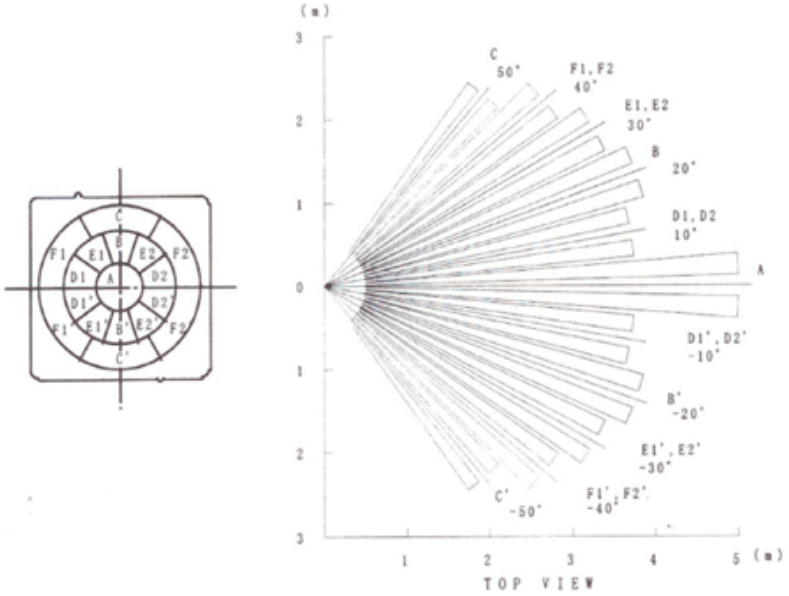
OK, so now we have a much larger range. However, remember that we actually have two sensors, and more importantly we dont want two really big sensing-area rectangles, but rather a scattering of multiple small areas. So what we do is split up the lens into multiple section, each section of which is a fresnel lens.

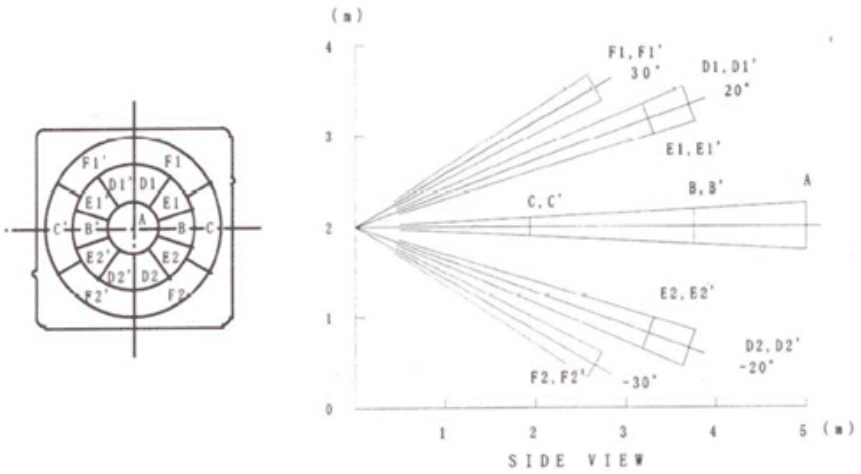
Here you can see the multiple facet-sections



This macro shot shows the different Fresnel lenses in each facet!

The different faceting and sub-lenses create a range of detection areas, interleaved with each other. That's why the lens centers in the facets above are 'inconsistent' - every other one points to a different half of the PIR sensing element





Images from NL11NH datasheet ()

Here is another image, more qualitative but not as quantitative. (Note that the sensor in the Adafruit shop is 110° not 90°)

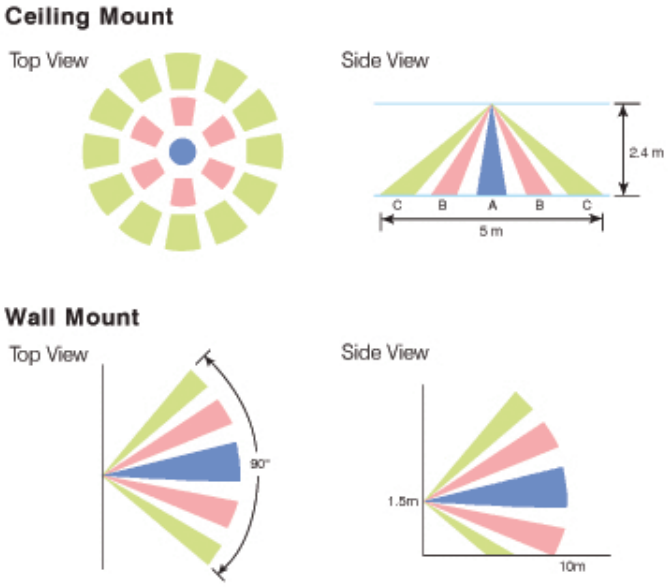
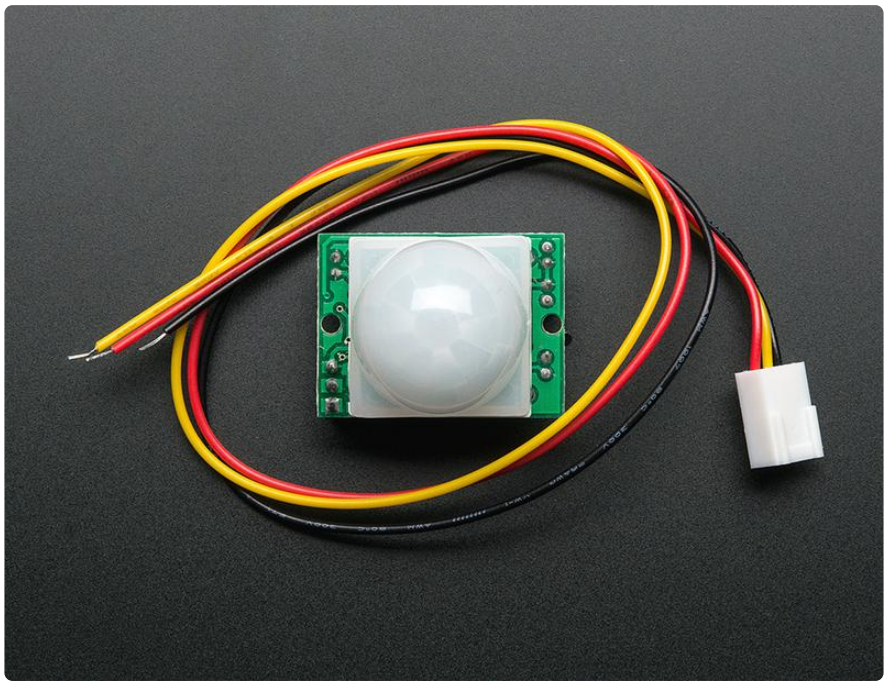


Image from IR-TEC ()

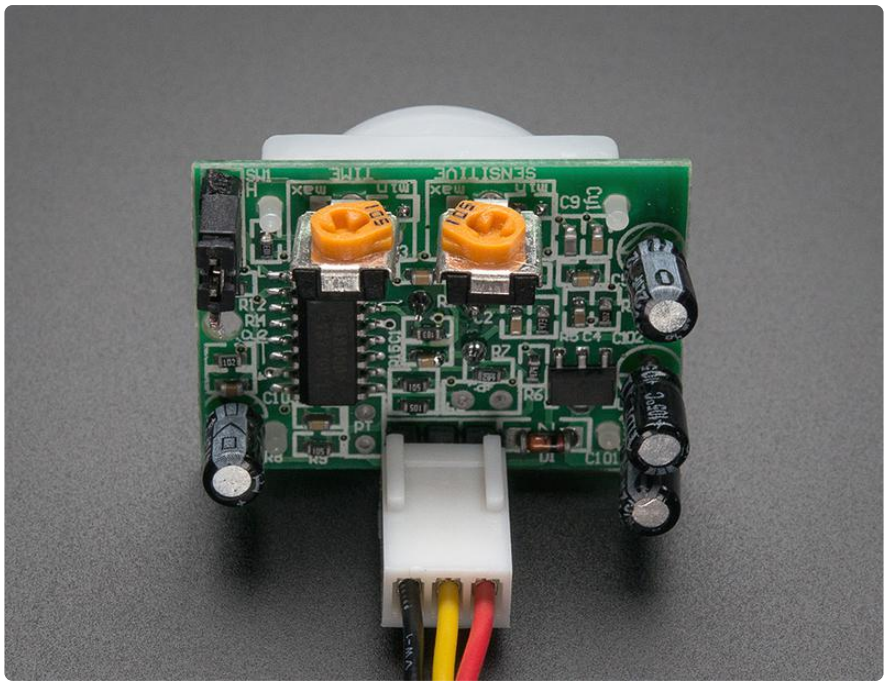
Connecting to a PIR



Most PIR modules have a 3-pin connection at the side or bottom. The pinout may vary between modules so triple-check the pinout! It's often silkscreened on right next to the connection (at least, ours is!) One pin will be ground, another will be signal and the final one will be power. Power is usually 3-5VDC input but may be as high as 12V. Sometimes larger modules don't have direct output and instead just operate a relay in which case there is ground, power and the two switch connections.

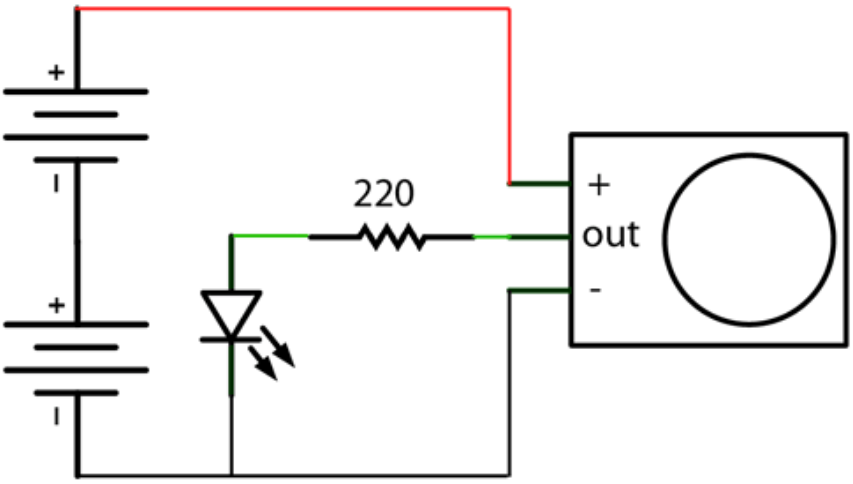
The output of some relays may be 'open collector' - that means it requires a pullup resistor. If you're not getting a variable output be sure to try attaching a 10K pullup between the signal and power pins.

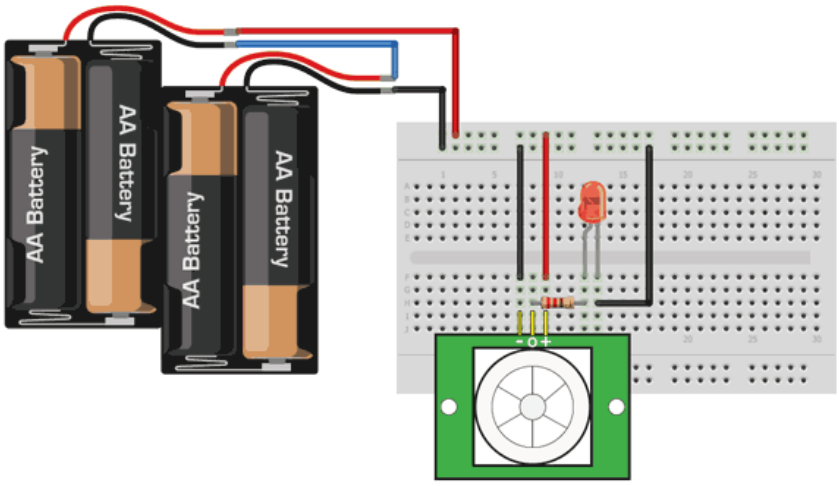
An easy way of prototyping with PIR sensors is to connect it to a breadboard since the connection port is 0.1" spacing. Some PIRs come with header on them already, the one's from adafruit have a straight 3-pin header on them for connecting a cable



For our PIR's the red cable is + voltage power, black cable is - ground power and yellow is the signal out. Just make sure you plug the cable in as shown above! If you get it backwards you won't damage the PIR but it won't work.

Testing a PIR





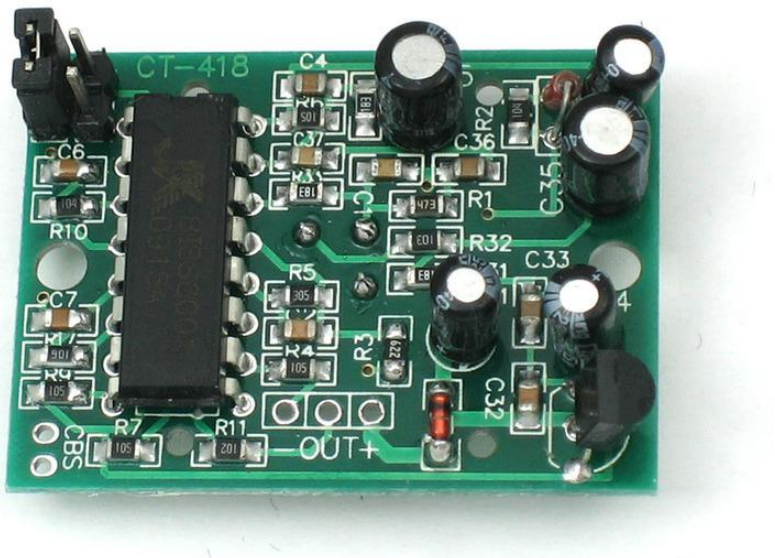
Now when the PIR detects motion, the output pin will go "high" to 3.3V and light up the LED!

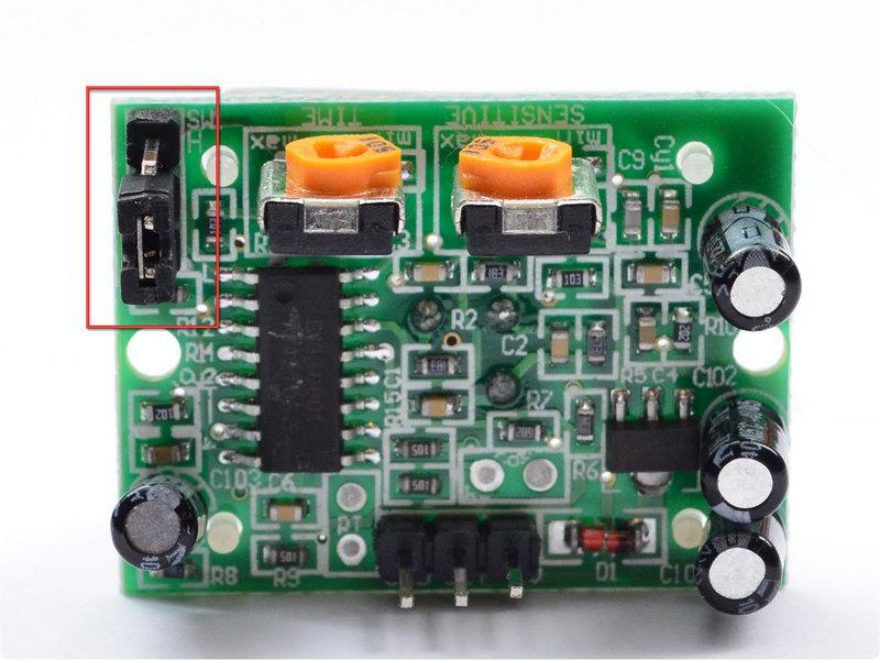
Once you have the breadboard wired up, insert batteries and wait 30-60 seconds for the PIR to 'stabilize'. During that time the LED may blink a little. Wait until the LED is off and then move around in front of it, waving a hand, etc, to see the LED light up!

Retriggering

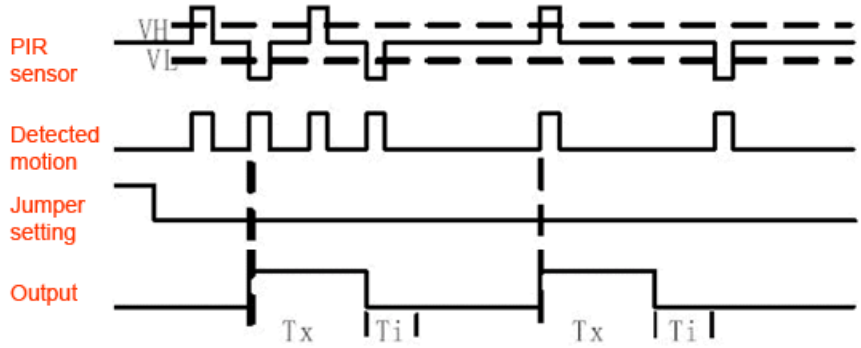
There's a couple options you may have with your PIR. First up we'll explore the 'Retriggering' option.

Once you have the LED blinking, look on the back of the PIR sensor and make sure that the jumper is placed in the L position as shown below.

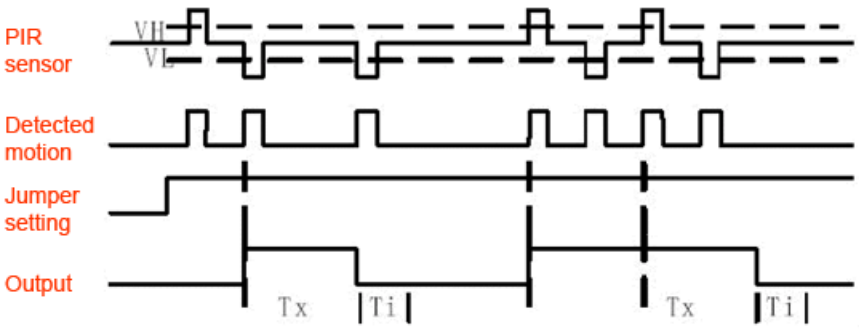




Now set up the testing board again. You may notice that when connecting up the PIR sensor as above, the LED does not stay on when moving in front of it but actually turns on and off every second or so. That is called "non-retriggering".

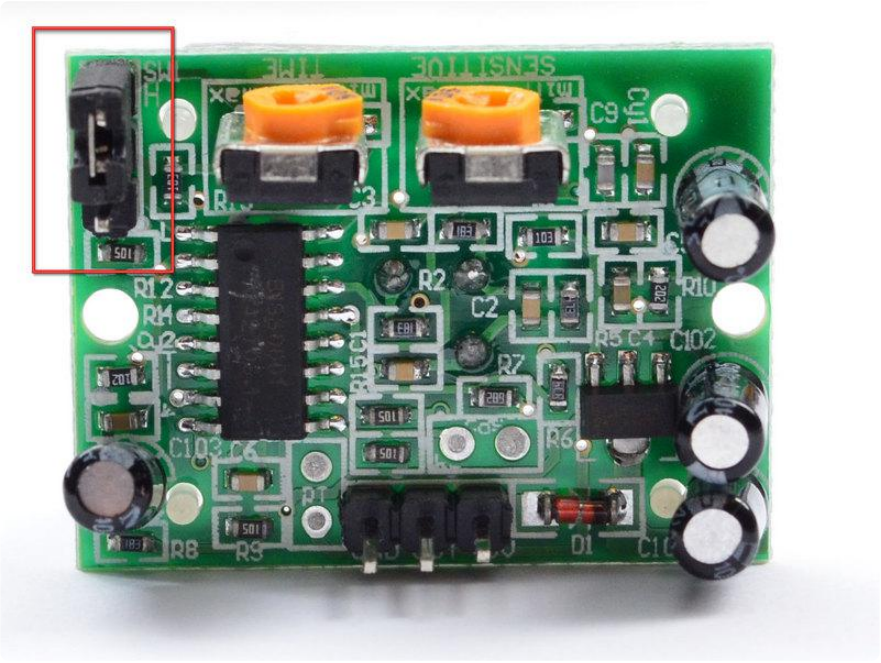


Now change the jumper so that it is in the H position. If you set up the test, you will notice that now the LED does stay on the entire time that something is moving. That is called "retriggering".



(The graphs above are from the BISS0001 datasheet, they kinda suck)

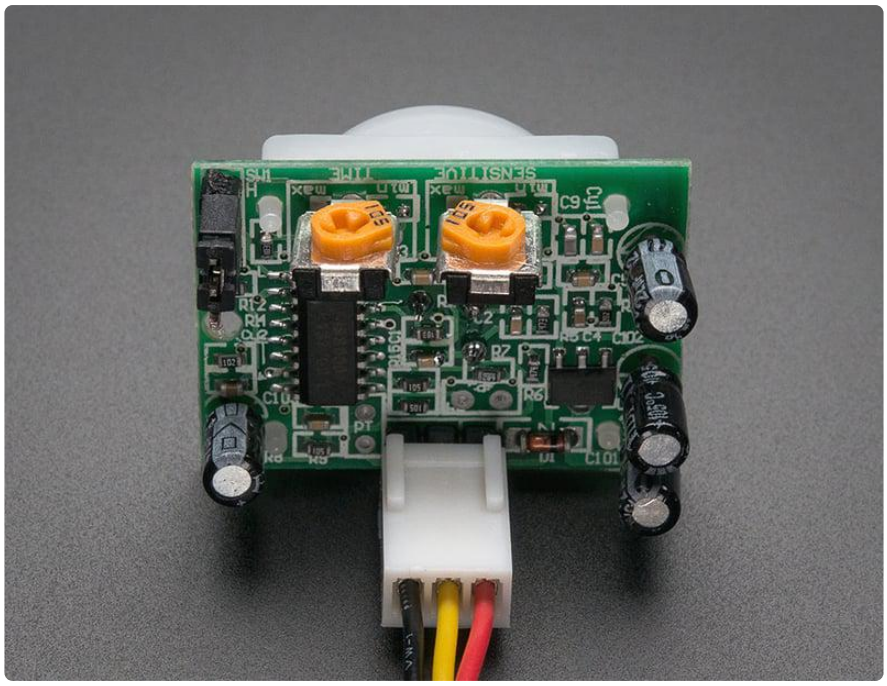
For most applications, "retriggering" (jumper in H position as shown below) mode is a little nicer.



If you need to connect the sensor to something edge-triggered, you'll want to set it to "non-retriggering" (jumper in L position).

Changing sensitivity

The Adafruit PIR has a trimpot on the back for adjusting sensitivity. You can adjust this if your PIR is too sensitive or not sensitive enough - clockwise makes it more sensitive.



Changing Pulse Time and Timeout Length

There are two 'timeouts' associated with the PIR sensor. One is the "Tx" timeout: how long the LED is lit after it detects movement - this is easy to adjust on Adafruit PIR's because there's a potentiometer.

The second is the "Ti" timeout which is how long the LED is guaranteed to be off when there is no movement. This one is not easily changed but if you're handy with a soldering iron it is within reason.

First, lets take a look at the BISS datasheet again

Tx = The time duration during which the output pin (Vo) remains high after triggering.
Ti = During this time period, triggering is inhibited. See timing charts for details.

Tx ≈ 24576 xR10 x C6; **Ti** ≈ 24 x R9 x C7。 (ref to schematic)

On Adafruit PIR sensors, there's a little trim potentiometer labeled TIME. This is a 1 Megaohm adjustable resistor which is added to a 10K series resistor. And C6 is 0.01uF so

Tx = 24576 x (10K + Rtime) x 0.01uF

If the Rtime potentiometer is turned all the way down counter-clockwise (to 0 ohms) then

Tx = 24576 x (10K) x 0.01uF = 2.5 seconds (approx)

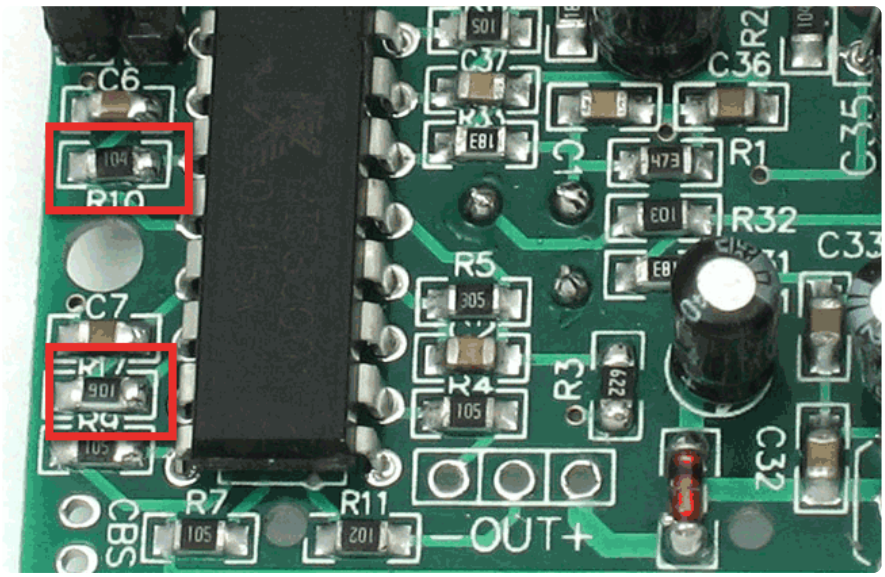
If the Rtime potentiometer is turned all the way up clockwise to 1 Megaohm then

Tx = 24576 x (1010K) x 0.01uF = 250 seconds (approx)

If RTime is in the middle, that'd be about 120 seconds (two minutes) so you can tweak it as necessary. For example if you want motion from someone to turn on a fan for a minimum of 1 minute, set the Rtime potentiometer to about 1/4 the way around.

For older/other PIR sensors

If you have a PIR sensor from somewhere else that does not have a potentiometer adjust, you can trace out the adjustment resistors this way:



Determining R10 and R9 isnt too tough. Unfortunately this PIR sensor is mislabeled (it looks like they swapped R9 R17). You can trace the pins by looking at the BISS001 datasheet and figuring out what pins they are - R10 connects to pin 3 and R9 connects to pin 7. the capacitors are a little tougher to determine, but you can 'reverse engineer' them from timing the sensor and solving!

For example:

Tx is = 24576 * R10 * C6 = ~1.2 seconds
R10 = 4.7K and C6 = 10nF

Likewise,

Ti = 24 * R9 * C7 = ~1.2 seconds
R9 = 470K and C7 = 0.1uF

You can change the timing by swapping different resistors or capacitors. For a nice tutorial on this, see Keith's PIR hacking page ().

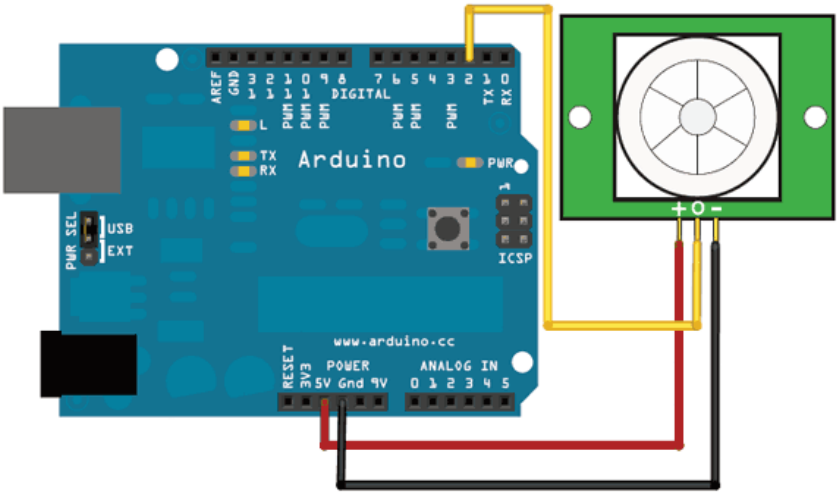
Using a PIR w/Arduino

Reading PIR Sensors

Connecting PIR sensors to a microcontroller is really simple. The PIR acts as a digital output, it can be high voltage or low voltage, so all you need to do is listen for the pin to flip high (detected) or low (not detected) by listening on a digital input on your Arduino

Its likely that you'll want retriggering, so be sure to put the jumper in the H position!

Power the PIR with 5V and connect ground to ground. Then connect the output to a digital pin. In this example we'll use pin 2.



The code is very simple, and is basically just keeps track of whether the input to pin 2 is high or low. It also tracks the state of the pin, so that it prints out a message when motion has started and stopped.

```
/*
 * PIR sensor tester
 */

int ledPin = 13;                // choose the pin for the LED
int inputPin = 2;               // choose the input pin (for PIR sensor)
int pirState = LOW;             // we start, assuming no motion detected
int val = 0;                    // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(inputPin, INPUT);     // declare sensor as input
```

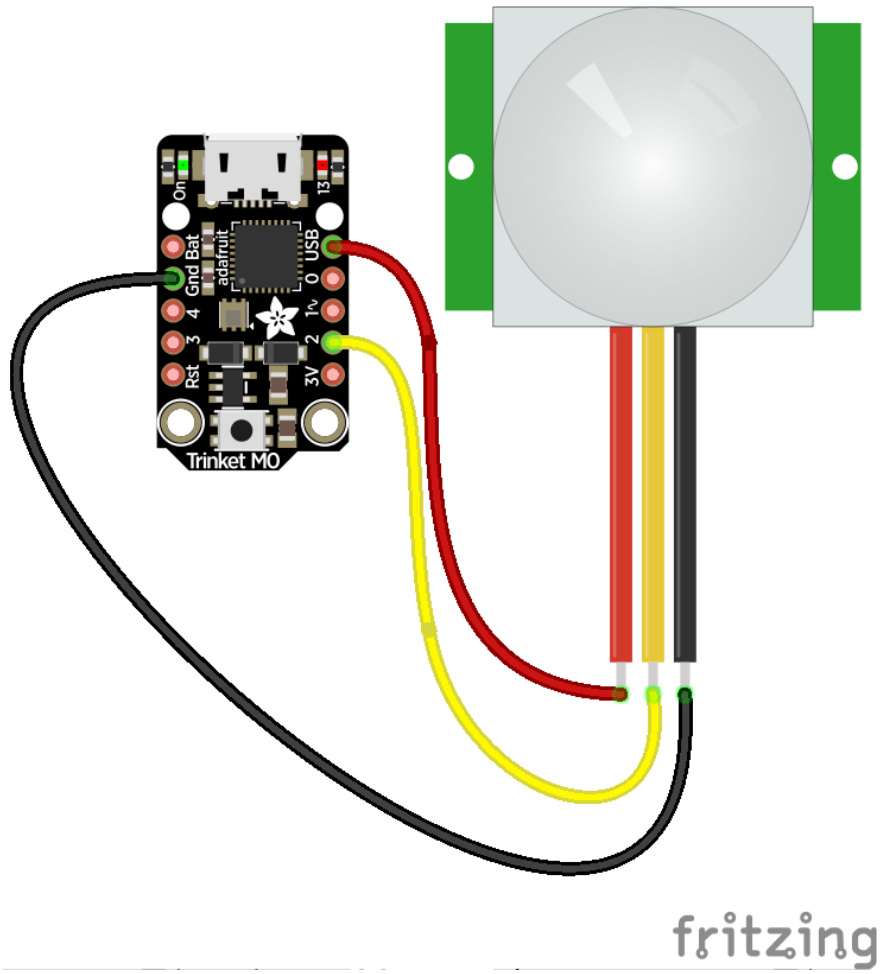
```
  Serial.begin(9600);
}

void loop(){
  val = digitalRead(inputPin);  // read input value
  if (val == HIGH) {            // check if the input is HIGH
    digitalWrite(ledPin, HIGH);  // turn LED ON
    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  } else {
    digitalWrite(ledPin, LOW); // turn LED OFF
    if (pirState == HIGH){
      // we have just turned of
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      pirState = LOW;
    }
  }
}
```

Don't forget that there are some times when you don't need a microcontroller. A PIR sensor can be connected to a relay (perhaps with a transistor buffer) without a micro!

CircuitPython Code

It's easy to use a PIR sensor with CircuitPython using simple digital inputs. The PIR sensor looks and acts kind of like a button or switch, i.e. it's only ever a high or low logic level, so you don't need any special libraries or other code to read one from Python. It will help to familiarize yourself with CircuitPython digital inputs and outputs () before continuing though!

First make sure your PIR sensor is wired to your board as shown in the previous page. There's no difference wiring a PIR sensor to an Arduino vs. CircuitPython board. You must connect the power, ground, and sensor output to your board. The sensor output should be connected to any digital I/O line on your board. In this example we'll use pin D2 on a Trinket M0.



Fritzing Source

Next connect to the board's serial REPL ()so you are at the CircuitPython >>> prompt.

Run the following code to import the board and digitalio modules which lets you read digital inputs:

```
import board
import digitalio
```

Then create a simple digital input for the PIR. Remember to use the right board pin for how you've wired your sensor to your board. This example is using pin D2 on a Trinket M0:

```
pir = digitalio.DigitalInOut(board.D2)
pir.direction = digitalio.Direction.INPUT
```

At this point you can read the state of the sensor by reading the value property. If the value is at a low logic level, or False, the sensor sees no movement. If it's at a high logic level, or True, the sensor is detecting movement!

Note you'll likely want the sensor's jumper in the H position for retriggering mode as mentioned on the previous page.

For example with no movement in front of the sensor you might see:

```
pir.value
```

```
>>> pir.value
False
>>>
```

Then wave your hand in front of the sensor, and as you wave it run the same command again. Notice you get a True result!

```
pir.value
```

```
>>> pir.value
True
>>>
```

That's all there is to using a PIR sensor with CircuitPython!

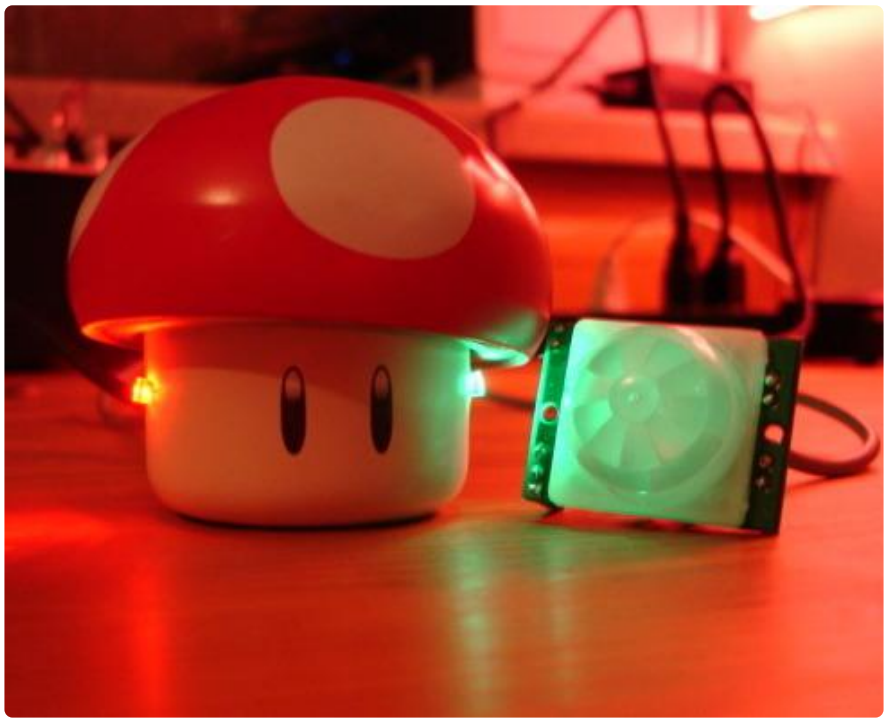
Here's a complete example just like from the previous page where movement from the PIR sensor will turn on the board's LED and print a message. This is a direct port of the previous page's Arduino example to CircuitPython. Try saving it as a main.py on your board and connecting to the serial terminal to see the output as it runs! (be sure to change the board pin numbers to your sensor and LED wiring!)

```
import board
import digitalio

LED_PIN = board.D13  # Pin number for the board's built in LED.
PIR_PIN = board.D2   # Pin number connected to PIR sensor output wire.

# Setup digital input for PIR sensor:
pir = digitalio.DigitalInOut(PIR_PIN)
pir.direction = digitalio.Direction.INPUT

# Setup digital output for LED:
led = digitalio.DigitalInOut(LED_PIN)
led.direction = digitalio.Direction.OUTPUT
```

```
# Main loop that will run forever:
old_value = pir.value
while True:
    pir_value = pir.value
    if pir_value:
        # PIR is detecting movement! Turn on LED.
        led.value = True
        # Check if this is the first time movement was
        # detected and print a message!
        if not old_value:
            print('Motion detected!')
    else:
        # PIR is not detecting movement. Turn off LED.
        led.value = False
        # Again check if this is the first time movement
        # stopped and print a message.
        if old_value:
            print('Motion ended!')
    old_value = pir_value
```

Example Projects

A simple room greeter that plays the super mario brothers theme music when triggered by a PIR in a hacked airwick freshener unit.



A USB-powered singing and blinking Mario mushroom (there's a video on the site!) ()

Rain Umbrellas ()

A home-made security system using PIR sensors (which is built into a Start Trek panel!)

PIR sensor + Arduino + Servo = automatic cat door!