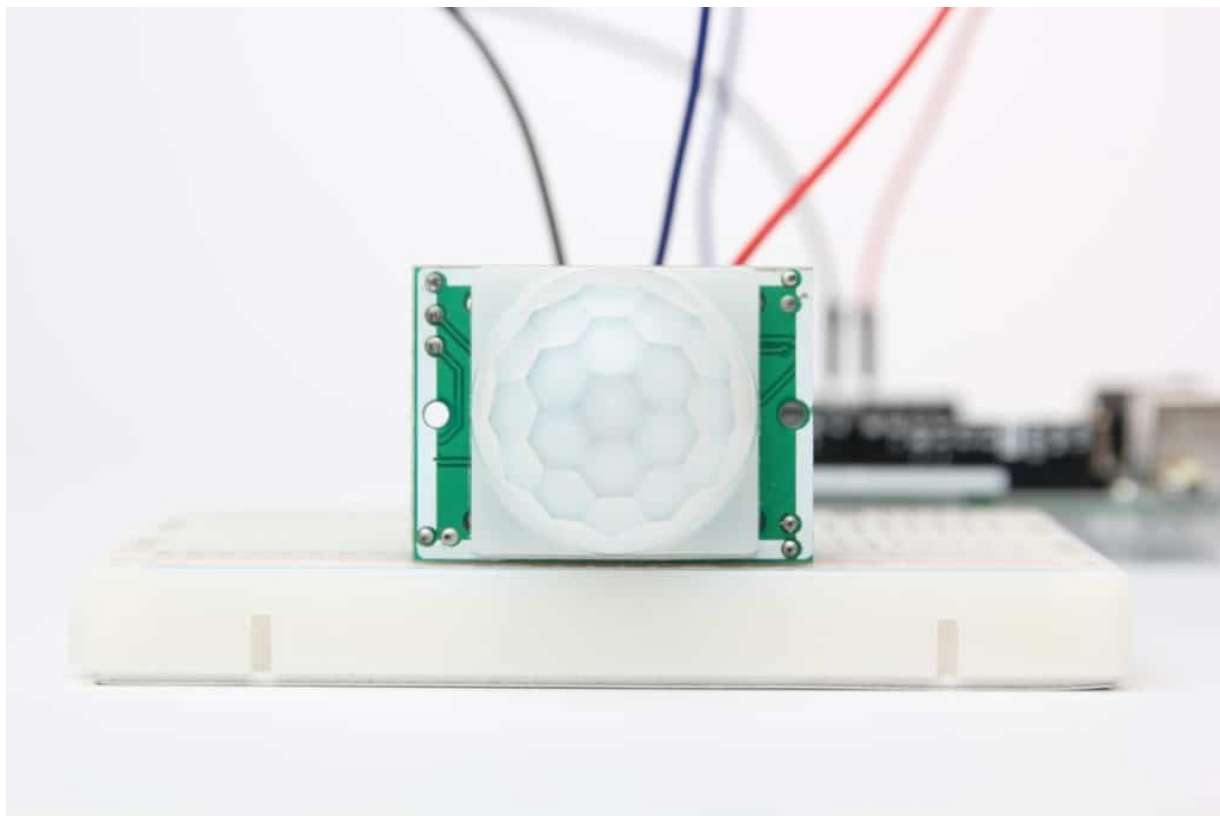


Low Cost Coding Service &amp; Quick Delivery

# How to use HC-SR501 PIR Motion Sensor with Arduino

 8 Comments

In this tutorial, you will learn how the HC-SR501 PIR motion sensor works and how you can use it with Arduino. You can find Passive Infrared (PIR) sensors all around you, they are not only used for security purposes, but also in most automatically-activated lighting systems.

In this article, I have included a wiring diagram and example codes so you can start experimenting with your sensor. After each example, I break down and explain how the code works, so you should have no problems modifying it to suit your needs.






First I will show you how you can use the HC-SR501 as a standalone unit. Next, we will connect it to an Arduino UNO and I will show you how to use it as a simple alarm system.

This tutorial focuses on the HC-SR501 sensor, but you can also use the provided code for similar sensors like the [HC-SR505](#) or [AM312](#). The main difference is that these cheaper sensors have a smaller detection range and don't have a potentiometer to adjust the sensitivity and time delay.

---

## Supplies

### Hardware components

	HC-SR501 PIR motion sensor	× 1	<a href="#">Amazon</a>
	Arduino Uno Rev3	× 1	<a href="#">Amazon</a>
	Breadboard	× 1	<a href="#">Amazon</a>
	Jumper wires	× 10	<a href="#">Amazon</a>
	Resistor assortment	× 1	<a href="#">Amazon</a>
	LEDs	× 1	<a href="#">Amazon</a>
	Passive buzzer	× 1	<a href="#">Amazon</a>
	USB cable type A/B	× 1	<a href="#">Amazon</a>

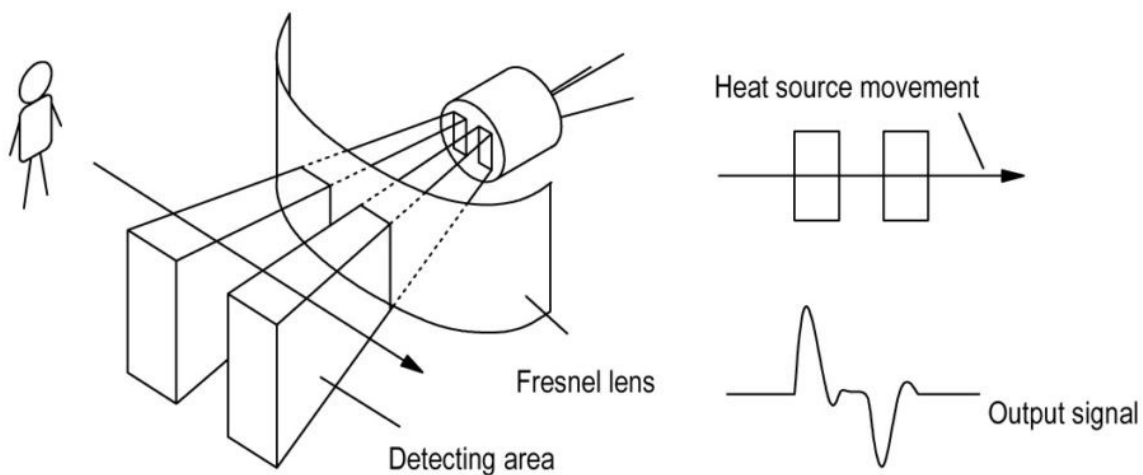
### Software



Makerguides.com is a participant in the Amazon Services LLC Associates Program, an affiliate advertising program designed to provide a means for sites to earn advertising fees by advertising and linking to products on Amazon.com.

## How does a PIR Motion Sensor work?

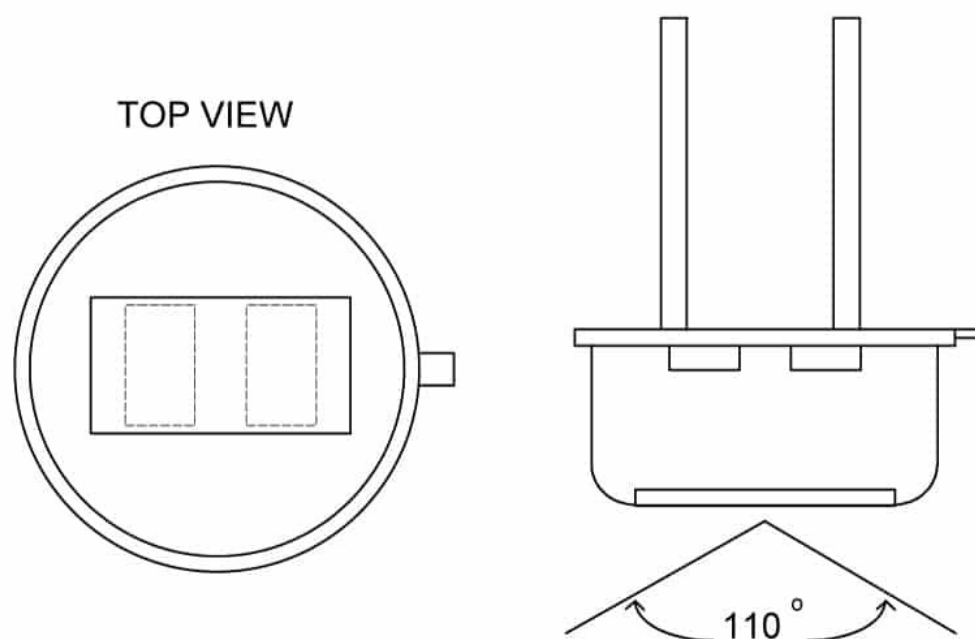
PIR motion sensors consist of two main parts: a **pyroelectric sensing element** and a **fresnel lens**. The pyroelectric sensing element can detect infrared radiation. All objects with a temperature above absolute zero (0 Kelvin / -273.15 °C) emit heat energy in the form of infrared radiation, including human bodies.



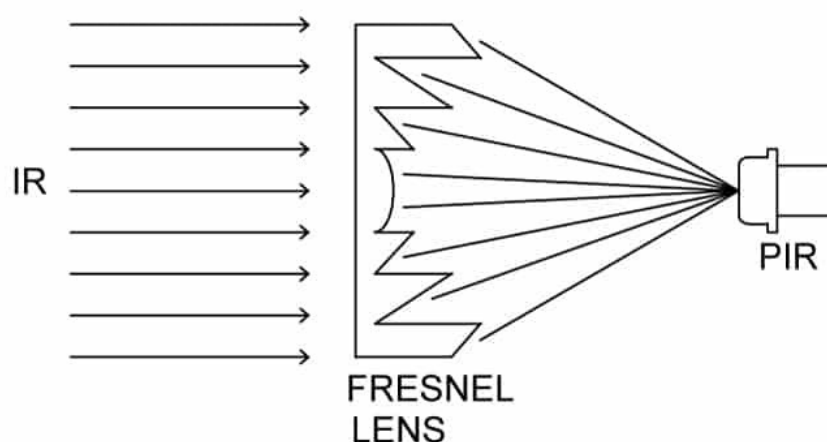
A pyroelectric sensor has two rectangular slots in it made of a material that allows the infrared radiation to pass. Behind these, there are two separate infrared sensor electrodes, one responsible for producing a positive output and the other a negative output. The reason for that is that we are looking for a change in IR levels and not ambient IR levels.

The two electrodes 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.

The on-board signal processing IC processes this signal and turns the output pin of the sensor HIGH or LOW accordingly.



The white dome in front of the sensing element is a fresnel lens. This lens focuses the infrared radiation onto the sensor.



## HC-SR501 PIR Motion Sensor

The HC-SR501 PIR motion sensor is built around the BISS0001 Micro Power PIR Motion Detector IC. This IC is specifically developed to process the signal from PIR motion sensors.

If you remove the fresnel lens, you will see the RE200B pyroelectric sensing element. On the PCB you can also find a built-in voltage regulator. This means you can power the board with a large DC voltage range, typically 5 V is used.

The specifications of the HC-SR501 are given in the table below, note that there might be small differences between manufacturers.

## HC-SR501 Specifications

Operating voltage	4.5 – 20 V
Quiescent current	50 $\mu$ A
Level output	HIGH 3.3 V / LOW 0 V
Trigger	L single trigger / H repeating trigger
Delay time	3 – 300 s
Blocking time	2.5 s (default)
Trigger	L single trigger / H repeating trigger
Measuring range	3 – 7 m maximum
	2 mm
Measuring angle	< 110° cone angle
PCB dimensions	32.5 x 24 mm
Mounting holes	2 mm, 28.5 mm spacing

Fresnel lens dimensions	15 mm x 23 mm diameter
Operating temperature	-15 – 70 °C
Cost	<a href="#">Check price</a>

---

For more information, you can check out the datasheets below:

HC-SR501 Datasheet

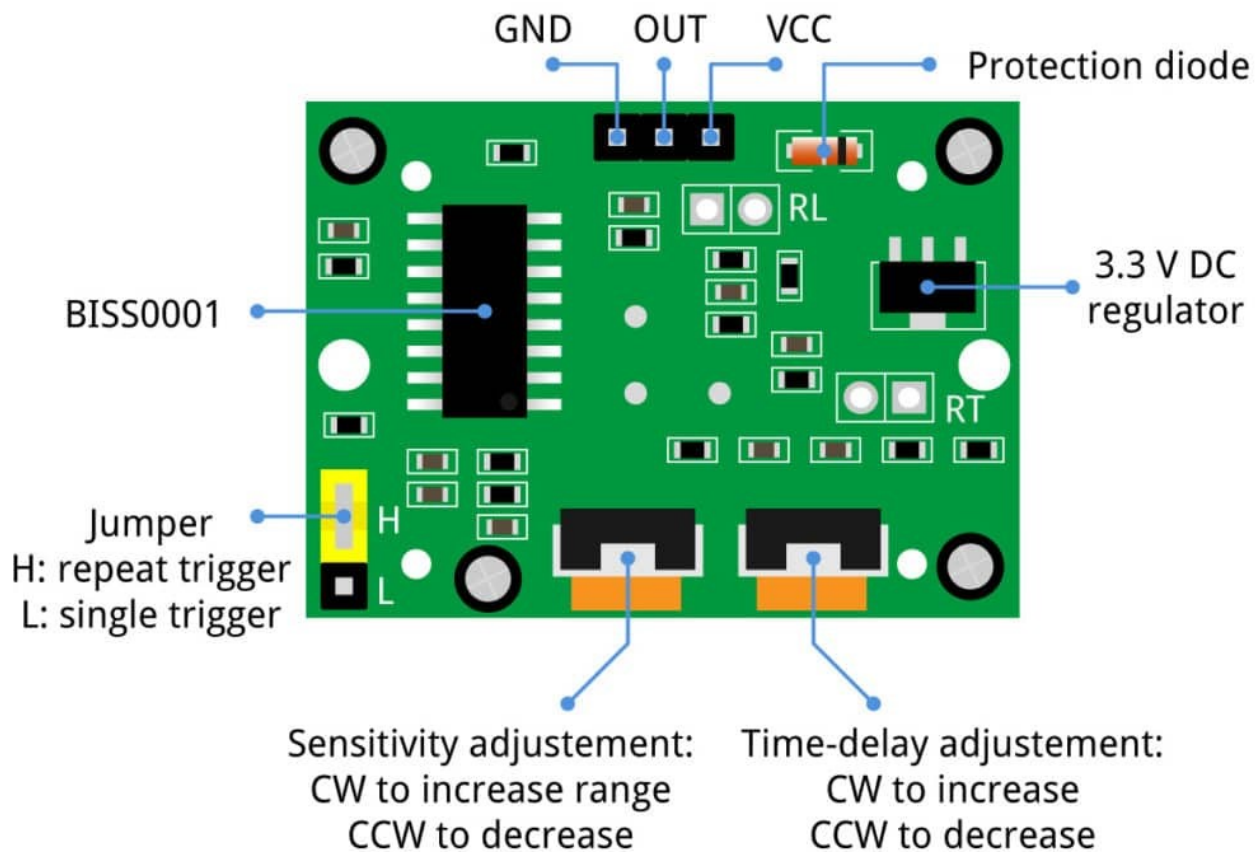
BISS0001 Datasheet

RE200B Datasheet

---

## Adjusting the HC-SR501

On the back of the board you will find two potentiometers and a jumper, which you can use to adjust several parameters:



## Sensitivity (range) adjustment

The HC-SR501 has a maximum sensing distance (detection range) of 7 meters. You can adjust the sensing distance by rotating the sensitivity potentiometer CW or CCW (see picture above). Rotating the potentiometer clockwise increases the sensing distance to a maximum of 7 meters. Rotating it counterclockwise decreases the sensing distance to a minimum of 3 meters.

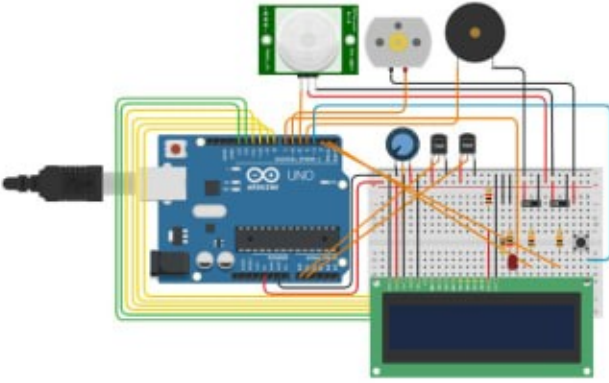
## Time-delay adjustment (Tx)


This potentiometer can be used to adjust the time that the output stays HIGH for after motion is detected. At a minimum, the delay is 3 seconds and at a maximum, it is 300 seconds or 5 minutes. Turn the potentiometer clockwise to increase the delay and counterclockwise to decrease the delay.

## Trigger selection jumper

The (yellow) jumper can be used to select one of the two trigger modes. It can be set to either **L** (single trigger) or **H** (repeating trigger):

## Cheap Arduino Coding Services Done For You:



 **alithethird**

I will code an arduino esp32 esp8266 project

★★★★★ 5.0 (28)

STARTING AT \$10



**Arduino Projects**

Programming  
Circuit Design  
Automation  
IOT & More...

 **ahmadraza566**

I will do arduino programming, projects and circuit designing

★★★★★ 5.0 (138)

STARTING AT \$30

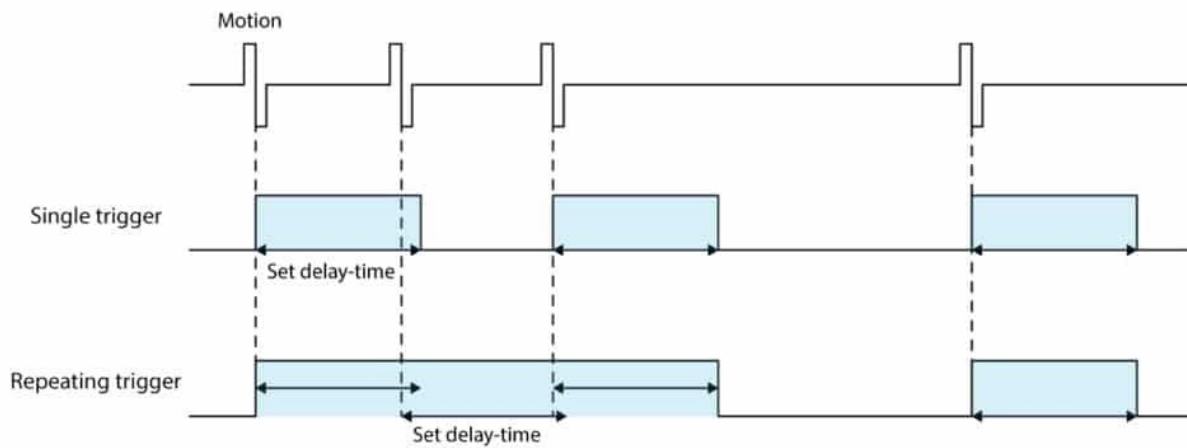
[Explore more Arduino services](#)

Services on **fiverr**.

- **Single trigger** – The output will turn HIGH as soon as motion is detected. It will stay HIGH for the time set by the potentiometer. Any movement during this period is not processed and does not restart the timer.
- **Repeating trigger** – Every time motion is detected, the delay timer is restarted.

The difference between the single and repeating trigger mode is shown in the figure below.

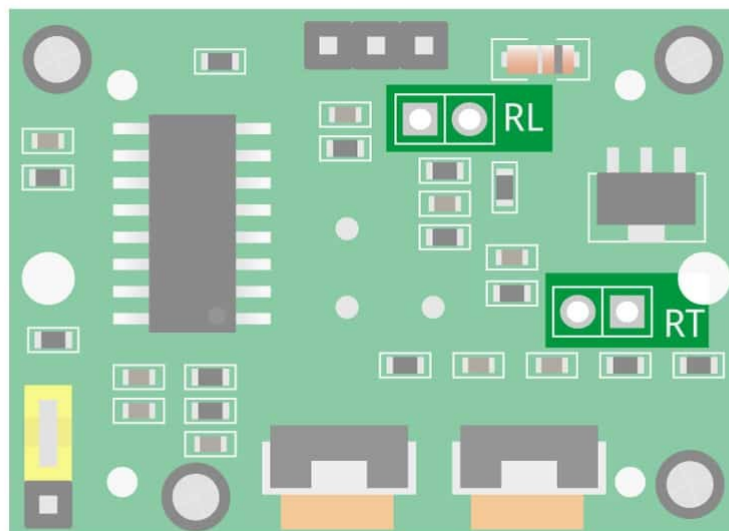




Difference between single and repeating trigger mode. The arrows indicate the set delay-time.

## Adding a thermistor and/or LDR to the HC-SR501

As can be seen in the image below, the HC-SR501 has solder pads for two additional components. These pads are typically labeled 'RL' and 'RT'.



- **RL** – Here you can add a [light dependent resistor](#) (LDR) or photoresistor which has a low resistance under strong ambient light. This causes the detector to be operational only when the detection area is sufficiently dark.
- **RT** – This pad is meant for a [thermistor](#). Adding this makes the sensitivity of the sensor less dependent on the ambient temperature

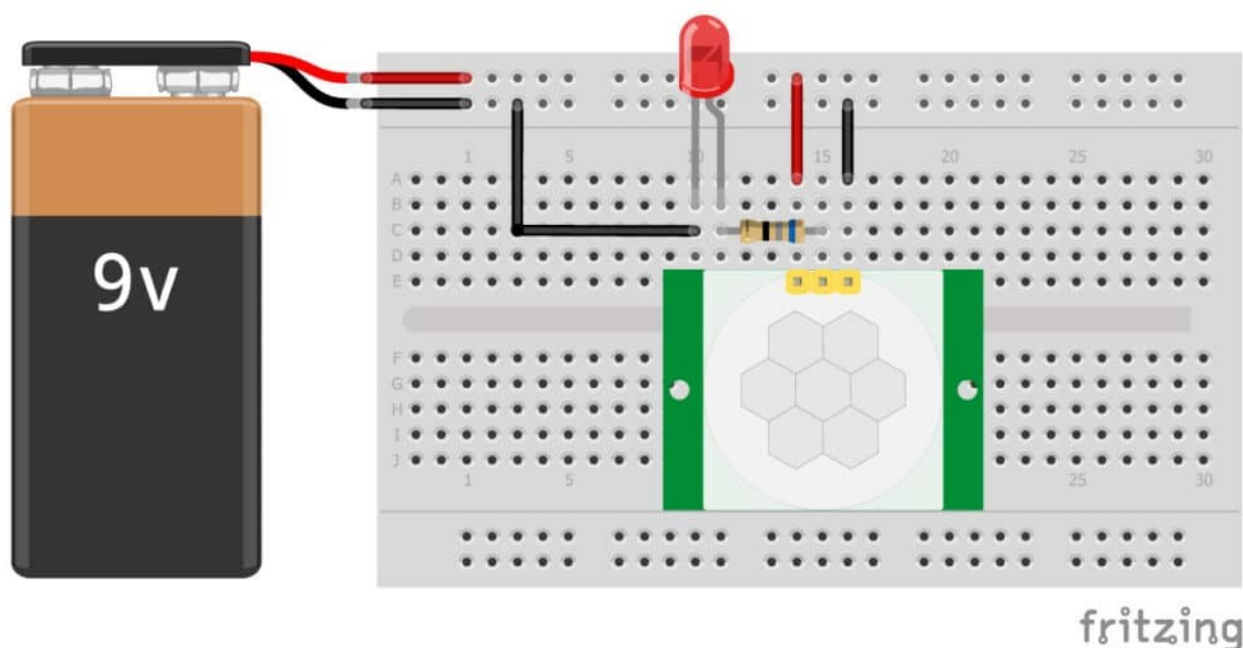
Unfortunately, no additional information is provided in the datasheets, so I am not entirely sure about what component values you should use.

---

## Using the HC-SR501 PIR motion sensor as a standalone unit

For most applications, you can just use the HC-SR501 as a standalone unit. You can use the output signal to trigger things like relays and LEDs.

The wiring is very simple as can be seen in the picture below. Simply connect VCC and GND to a battery and a red LED between the output pin and ground. The output voltage is 3.3 V, so I added a 68  $\Omega$  current limiting resistor in series with the LED.



HC-SR501 PIR motion sensor with LED wiring diagram

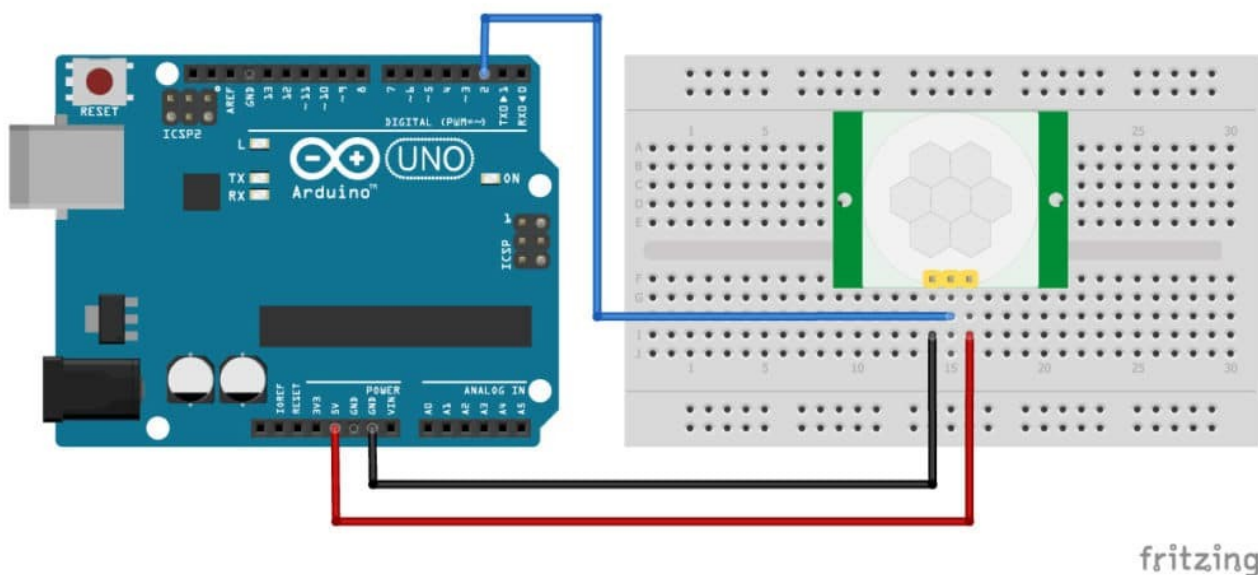
Note that after powering up the sensor, you need to wait 30 – 60 seconds for the sensor to initialize. During this period, the LED might blink a couple of times. After waiting for a minute, you can wave your hand in front of the sensor and you should be able to see the LED light up.

With this setup, it is easy to test the functionality of the sensor. This is also a good moment to play around with the sensitivity and time-delay settings, as well as the two different trigger modes.

## Wiring – Connecting HC-SR501 PIR motion sensor to Arduino UNO

By connecting the motion sensor to a microcontroller like the Arduino UNO, you can use it to control all kinds of things: LEDs, relays, motors, buzzers etc.

In the wiring diagram below, you can see how to hook it up to the Arduino. You can read the sensor with one of the general-purpose input/output (GPIO) pins of the Arduino. In this example, I connected it to digital pin 2. The VCC and GND pins are connected to 5 V and GND respectively.



HC-SR501 PIR motion sensor with Arduino wiring diagram

The connections are also given in the table below:

### HC-SR501 PIR Motion Sensor Connections

HC-SR501 PIR Motion Sensor	Arduino
VCC	5 V
OUT	Pin 2
GND	GND

Once you have wired up the sensor, the next step is to upload some example code.

## HC-SR501 PIR motion sensor with Arduino UNO example code

With the following example code, you can read out the sensor and control the on-board LED of the Arduino (connected to pin 13). This code can also be used to control simple relays to turn a bigger light on or off.

You can upload the example code with the [Arduino IDE](#).

For this code to work properly, it is best to set the trigger mode jumper to 'H' (repeat trigger mode). Also adjust the time-delay potentiometer to the lowest value. Turn it counterclockwise as far as it will go.

The code will read the state of the sensor (HIGH or LOW) and turn on or off the on-board LED accordingly. It will also print a message to the Serial Monitor, which you can access under Tools or type (Ctrl+Shift+M).

```
/* Example code for HC-SR501 PIR motion sensor with Arduino. More i

// Define connection pins:
#define pirPin 2
#define ledPin 13

// Create variables:
```

```
int val = 0;
bool motionState = false; // We start with no motion detected.

void setup() {
  // Configure the pins as input or output:
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);

  // Begin serial communication at a baud rate of 9600:
  Serial.begin(9600);
}

void loop() {
  // Read out the pirPin and store as val:
  val = digitalRead(pirPin);

  // If motion is detected (pirPin = HIGH), do the following:
  if (val == HIGH) {
    digitalWrite(ledPin, HIGH); // Turn on the on-board LED.

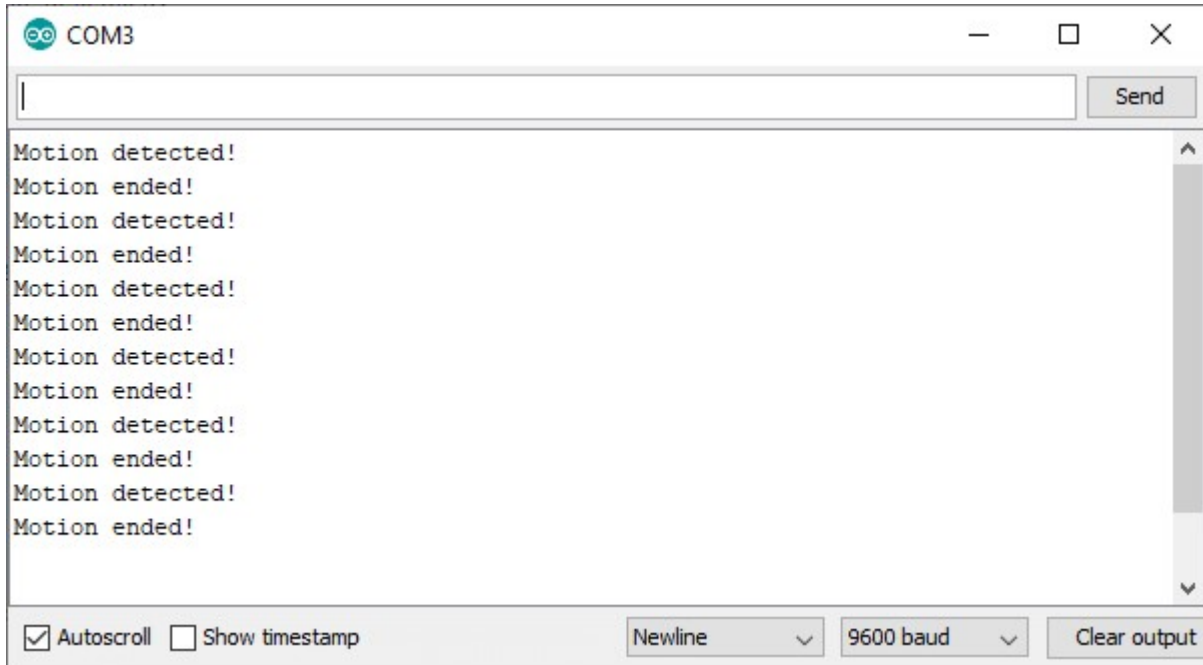
    // Change the motion state to true (motion detected):
    if (motionState == false) {
      Serial.println("Motion detected!");
      motionState = true;
    }
  }

  // If no motion is detected (pirPin = LOW), do the following:
  else {
    digitalWrite(ledPin, LOW); // Turn off the on-board LED.

    // Change the motion state to false (no motion):
    if (motionState == true) {
      Serial.println("Motion ended!");
      motionState = false;
    }
  }
}
```

```
}  
}
```

You should see the following output in the serial monitor:



PIR Sensor output on serial monitor

## Code explanation:

The code is quite simple and you don't need any Arduino libraries to use this sensor.

The sketch starts with defining the PIR sensor pin and LED pin. I connected them to Arduino pin 2 and 13 (on-board LED).

## The statement

```
#define
```

is used to give a name to a constant value. The compiler will replace any references to this constant with the defined value when the program is compiled. So everywhere you mention

```
pirPin
```

, the compiler will replace it with the value 2 when the program is compiled.

I also created two variables,

```
val
```

and

```
motionState
```

, which are an **integer** and **boolean** (true/false) respectively. The variable

```
val
```

is used to store the output of the PIR sensor (HIGH or LOW) and

```
motionState
```

will turn to true while motion is detected and to false when there is no motion.

```
// Define connection pins:  
#define pirPin 2  
#define ledPin 13
```

```
// Create variables:  
int val = 0;  
bool motionState = false; // We start with no motion detected.
```

In the

```
setup()
```

, we set the pins as input or output with the function

```
pinMode(pin,mode)
```

. The pirPin is an input and the ledPin is an output. We also begin serial communication at a baud rate of 9600. Make sure that the Serial Monitor is also set to 9600.

```
void setup() {  
  // Configure the pins as input or output:  
  pinMode(ledPin, OUTPUT);  
  pinMode(pirPin, INPUT);  
  
  // Begin serial communication at a baud rate of 9600:  
  Serial.begin(9600);  
}
```

In the

```
loop()
```

, I first read out the sensor with the function

```
digitalRead(pin)
```

. This function returns HIGH or LOW.



```
// Read out the pirPin and store as val:  
val = digitalRead(pirPin);
```

When the sensor output/val is HIGH, I turn the LED on with the function

```
digitalWrite(pin,value)
```

.

```
// If motion is detected (pirPin = HIGH), do the following:  
if (val == HIGH) {  
    digitalWrite(ledPin, HIGH); // Turn on the on-board LED.  
  
    // Change the motion state to true (motion detected):  
    if (motionState == false) {  
        Serial.println("Motion detected!");  
        motionState = true;  
    }  
}
```

Next, the motionState is changed to true and the message 'Motion detected!' is printed to the Serial Monitor. Note that I first check the current motionState, this makes sure that the message is only printed once per motion event.

If there is no longer motion in front of the sensor,

```
val
```

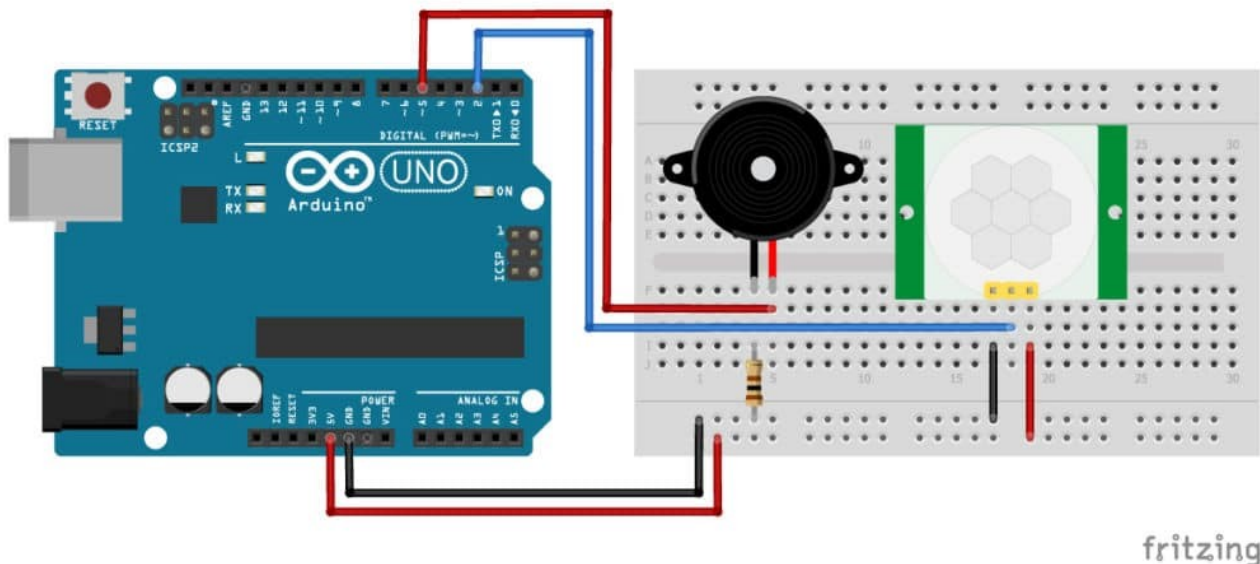
will change to LOW and the LED is turned off and the message 'Motion ended!' is printed to the serial monitor.

---

## Create an alarm system with PIR motion

## sensor and buzzer

With some simple changes, you can create an alarm system with the HC-SR501 and a [piezoelectric buzzer](#). I connected the buzzer with a 100  $\Omega$  resistor between digital pin 5 and GND. You can probably use the buzzer without a resistor (this makes it louder), but it will not sound as nice.



HC-SR501 PIR motion sensor with Arduino UNO and buzzer wiring diagram.

The code below is mostly the same as the previous example. I only added a function to create the beeping alarm sound. You can change the pitch of the buzzer by changing the input parameter of the

```
alarm(duration, frequency)
```

function.

```
/* Example code to create an alarm system with HC-SR501 PIR motion

// Define connection pins:
#define buzzerPin 5
#define pirPin 2
#define ledPin 13
```

```
// Create variables:
int val = 0;
bool motionState = false; // We start with no motion detected.

void setup() {
  // Configure the pins as input or output:
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);

  // Begin serial communication at a baud rate of 9600:
  Serial.begin(9600);
}

void loop() {
  // Read out the pirPin and store as val:
  val = digitalRead(pirPin);

  // If motion is detected (pirPin = HIGH), do the following:
  if (val == HIGH) {
    digitalWrite(ledPin, HIGH); // Turn on the on-board LED.
    alarm(500, 1000); // Call the alarm(duration, frequency) function
    delay(150);

    // Change the motion state to true (motion detected):
    if (motionState == false) {
      Serial.println("Motion detected!");
      motionState = true;
    }
  }

  // If no motion is detected (pirPin = LOW), do the following:
  else {
    digitalWrite(ledPin, LOW); // Turn off the on-board LED.
    noTone(buzzerPin); // Make sure no tone is played when no motion is detected
    delay(150);
  }
}
```

```
// Change the motion state to false (no motion):
if (motionState == true) {
    Serial.println("Motion ended!");
    motionState = false;
}
}

// Function to create a tone with parameters duration and frequency
void alarm(long duration, int freq) {
    tone(buzzerPin, freq);
    delay(duration);
    noTone(buzzerPin);
}
```

---

## Things to consider when designing a PIR sensor system

Just like other PIR sensors, the HC-SR501 needs some time to initialize and adjust to the infrared levels in the room. This takes approximately 1 minute when it is first powered up. Try to eliminate any motion in front of the sensor during this period.

Wind and a light source close to the sensor can cause interference, so try to adjust your setup to avoid this. Also, note that you must mount the sensor horizontally since most motion will happen in the horizontal plane (e.g. walking).

Besides the delay-time ( $T_x$ ), the sensor also has a 'blocking-time' ( $T_i$ ). By default, the blocking time is 2.5 seconds and it is not very easy to change (see BISS0001 datasheet). Each time the output goes from HIGH to LOW, the blocking period start. During this time period, the sensor will not

detect any motion.

When designing a system based on the HC-SR501, you will need to take these delay periods into account.

---

## Conclusion

In this article, I have shown you how the HC-SR501 PIR motion sensor works and how you can use it with Arduino. I hope you found it useful and informative. If you did, please **share it with a friend** who also likes electronics and making things!

I would love to know what projects you plan on building (or have already built) with this sensor. If you have any questions, suggestions, or if you think that things are missing in this tutorial, **please leave a comment down below**.

Note that comments are held for moderation to prevent spam.



By Benne

Published: July 12, 2019 - Last updated: March 2, 2022

[Arduino](#), [Other Tutorials](#), [Tutorials](#)

[Home](#) > [Tutorials](#) > How to use HC-SR501 PIR Motion Sensor with Arduino

---

Arduino    HC-SR501    HC-SR505    motion sensor    PIR    Sensor

Tutorial

---

← **Control a stepper motor with L298N motor driver and Arduino**

## TB6560 Stepper Motor Driver with Arduino Tutorial →

### 8 COMMENTS

#### Comment

Write your comment...

Name \*

Email \*

COMMENT



**NEIL IVES**

March 21, 2021 at 08:28 PM

To avoid cold surprises, I'm planning a system to shut the hot water taps in the house when someone is in the shower. This IR detector will be mounted so it detects a person standing under the shower head at the end of the bath. The output will trigger CMOS switches to close water valves in the house. I need sensitivity to be adjustable so a person taking a bath does not trigger the circuit. I also want the valves to stay closed for a while in the case of the showerer standing still long enough for the circuit release; the delay adjuster will be

important.

[↩ Reply](#)



**Ron K**

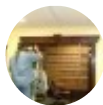
January 23, 2021 at 04:45 PM

To me, the data sheet doesn't provide critical information. Does the minimum range change with the maximum range? I did find it on another site. No. The minimum remains a constant zero.

Full CW, the range of sensing distance increases, and the maximum sensing distance range is about 0-7 meters.

Full CCW, the range of sensing distance is reduced, and the minimum sensing distance range is about 0-3 meters.

[↩ Reply](#)



**Scotty**

September 7, 2020 at 03:46 AM

thank you. Everyone makes building and using these look so simple.

[↩ Reply](#)



**Tim**

July 12, 2020 at 03:47 AM

Thanks for posting this. What I want to do is have four PIR's controlling four Halloween props in my yard. What I need to do is have the PIR sense movement and trigger the prop but I need to build in a delay so people can't just stand there and retrigger the prop. The delay would be for each PIR circuit. I figure four PIR's on one uno to control four props. Some might need to trigger a relay depending on the device

[↩ Reply](#)



**Ray**

November 15, 2020 at 12:13 AM

If you're using an Arduino, you could set the output pin to have a delay on/off for a given amount of time for each sensor. There are also relays that have timers built in as well that may help.

[↩ Reply](#)



**Christian H.**

June 12, 2020 at 07:32 PM

Hello,

Thanks for your tuto.

Do you know if there is a way to expand the "Block time" to 1 minute ?

Christian H.



[↩ Reply](#)**tony**

March 8, 2020 at 11:02 AM

re LDR, I added a 5516, 100k LDR and it worked fine, only triggering when relatively dark.

[↩ Reply](#)**bummi**

January 12, 2020 at 06:23 PM

hallo

habe die schaltung mit buzzer aufgebaut und den dazugehörigen code hochgeladen.

jedoch habe ich das problem, das der buzzer an 5 ohne bewegung durchgehend piept und mit bewegung pulsiert. (er geht nicht aus). die led an 13 ist jedoch ohne bewegung aus und schaltet bei bewegung ein (so wie es sein soll). der serielle monitor zeigt ebenfalls on und off korrekt an.  
wo liegt der fehler?

danke

[↩ Reply](#)



Guides, Tutorials & Projects For The Maker Community



## Navigation

- ▶ [Arduino Displays](#)
- ▶ [Distance Sensors](#)
- ▶ [Motor Controls](#)
- ▶ [Temperature & Humidity Sensors](#)
- ▶ [Other Tutorials](#)
- ▶ [Projects](#)
- ▶ [FAQs](#)
- ▶ [Contact](#)
- ▶ [Disclaimer](#)
- ▶ [Privacy Policy](#)
- ▶ [Terms & Conditions](#)

COPYRIGHT © 2021 MAKERGUIDES.COM