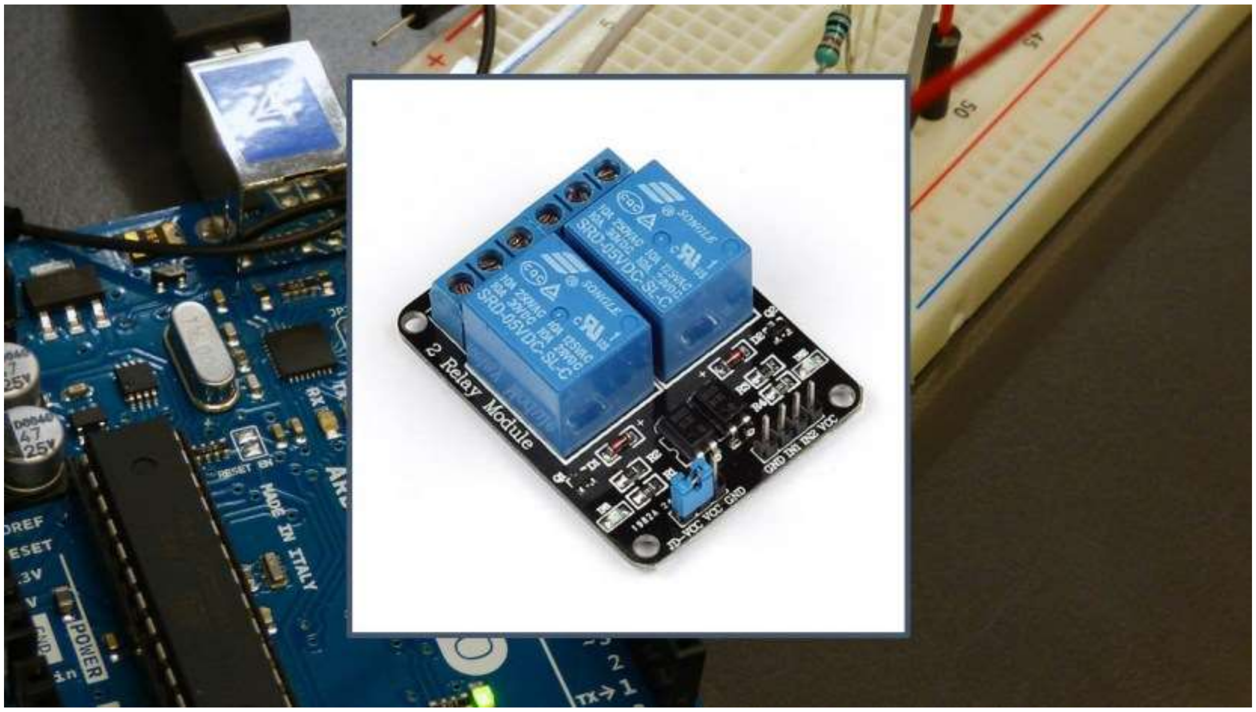
Guide for Relay Module with Arduino

This article shows how to control mains voltage with the Arduino using a relay module. We make a brief introduction to the relay module and build a simple project example with the Arduino. The example we'll build shows how to control a relay module with an Arduino and a PIR motion sensor.



By the end of this tutorial, you should be able to control any electronics appliances with your Arduino using a relay module.

Introducing the Relay Module

A relay is an electrically operated switch that can be turned on or off, letting the current go through or not, and can be controlled with low voltages, like the 5V provided by the Arduino pins.

Controlling a relay module with the Arduino is as simple as controlling any other output as we'll see later on.

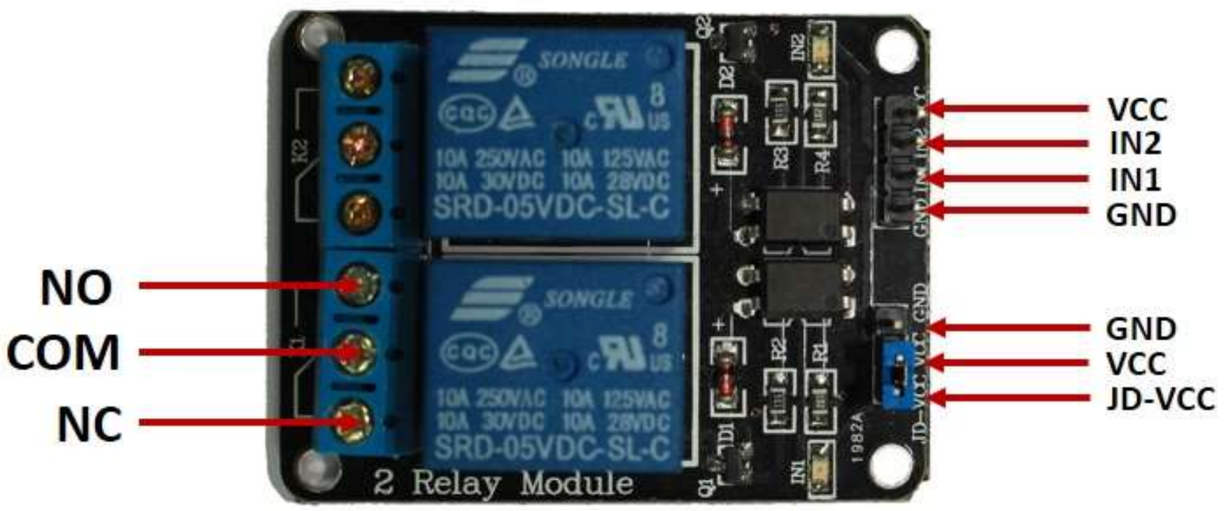
This relay module has two channels (those blue cubes). There are other models with one, four and eight channels. This module should be powered with 5V, which is appropriate to use with an Arduino. There are other relay modules that are powered using 3.3V, which is ideal for ESP32, ESP8266, and other microcontrollers.

Get a relay module:

- 5V 2-channel relay module
- 5V 1-channel relay module
- 5V 8-channel relay module
- 3.3V 1-channel relay module

Relay Pinout

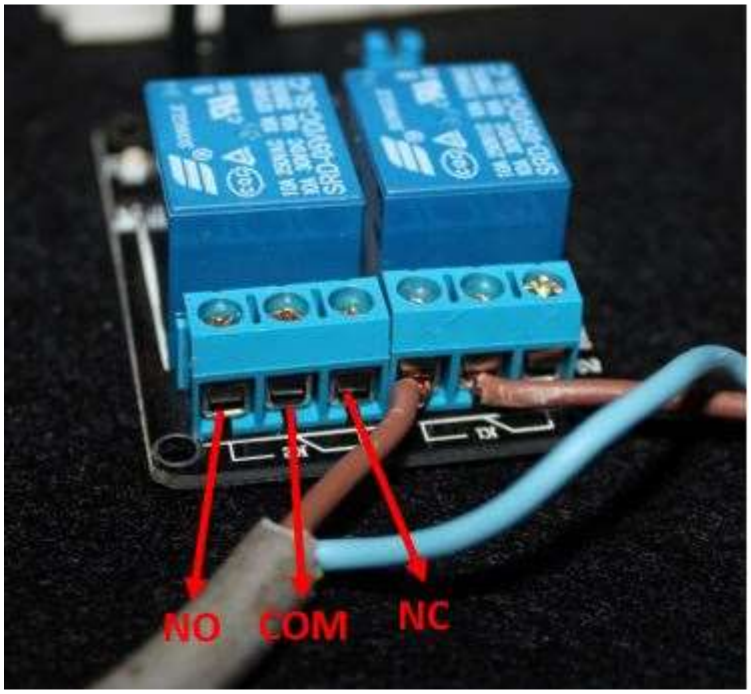
The following figure shows the relay module pinout.



The six pins on the left side of the relay module connect high voltage, and the pins on the right side connect the component that requires low voltage—the Arduino pins.

Mains voltage connections

The high-voltage side has two connectors, each with three sockets: common (COM), normally closed (NC), and normally open (NO).

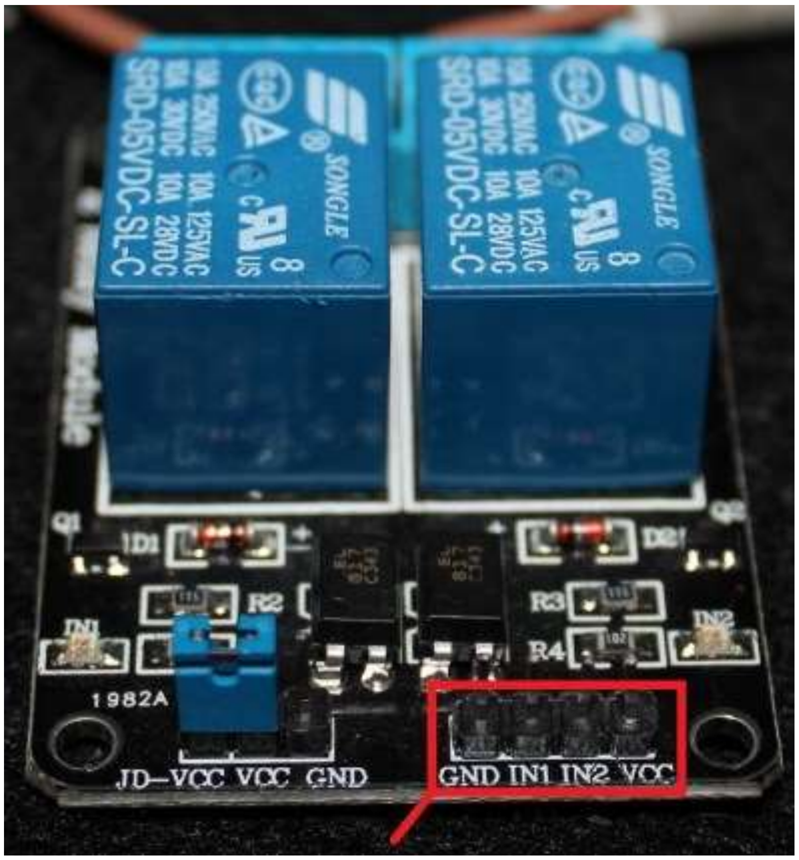


- **COM**: common pin
- **NC (Normally Closed):** the normally closed configuration is used when you want the relay to be closed by default, meaning the current is flowing unless you send a signal from the Arduino to the relay module to open the circuit and stop the current.
- **NO (Normally Open):** the normally open configuration works the other way around: the relay is always open, so the circuit is broken unless you send a signal from the Arduino to close the circuit.

If you just want to light up a lamp occasionally, it is better to use a normally-open circuit configuration.

Pin wiring

The low-voltage side has a set of four pins and a set of three pins.



The set at the right consists of **VCC** and **GND** to power up the module, and input 1 (**IN1**) and input 2 (**IN2**) to control the bottom and top relays, respectively.

The second set of pins consists of **GND**, **VCC**, and **JD-VCC** pins. The JD-VCC pin powers the electromagnet of the relay.

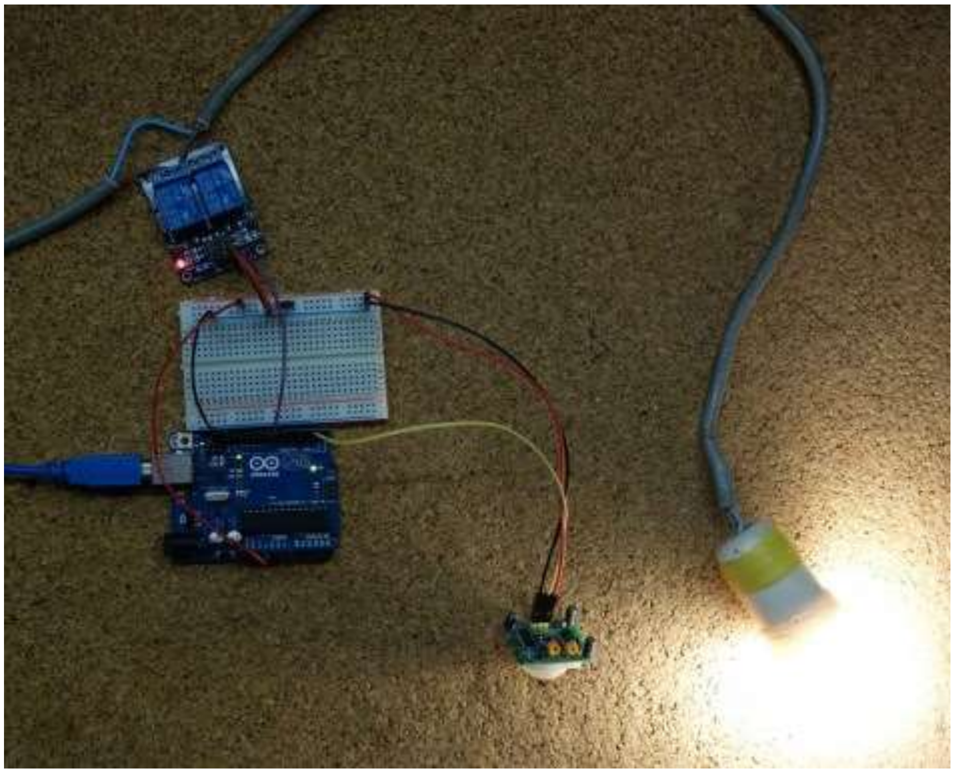
Note: notice that the module has a jumper cap connecting the VCC and JD-VCC pins; the one shown here is blue, but yours may be a different color. The jumper cap allows you to choose whether the circuit is physically connected to the Arduino circuit or not, and you can choose to have it on or not. With the jumper cap on, the VCC and JD-VCC pins are connected. That means the relay electromagnet is directly powered from the Arduino's power pin, so the relay module and the Arduino circuits are not physically isolated from each other (this is the configuration we'll use). Without the jumper cap, you need to provide an independent power source to power up the relay's electromagnet through the JD-VCC pin. That configuration physically isolates the relays from the Arduino with the module's built-in optocoupler.

The connections between the relay module and the Arduino are really simple:

- **GND**: goes to ground
- **IN1**: controls the first relay (it will be connected to an Arduino digital pin)
- **IN2**: controls the second relay (it should be connected to an Arduino digital pin if you are using this second relay. Otherwise, you don't need to connect it)

- **VCC**: goes to 5V

Example: Controlling a Lamp with a Relay Module and PIR Motion Sensor



In this example, we create a motion sensitive lamp. A lamp lights up for 10 seconds every time motion is detected.

Motion will be detected using a PIR motion sensor. If you are not familiar with the PIR motion sensor, you can read the following post:

- Arduino with PIR Motion Sensor

To control the lamp with mains voltage we'll use a relay module in normally-open configuration.

Safety warning

Before proceeding with this project, I want to let you know that you're dealing with mains voltage. Please read the safety warning below carefully.

Warning: when you are making projects that are connected to mains voltage, you really need to know what you are doing, otherwise you may shock yourself. This is a serious topic, and we want you to be safe. If you're not 100% sure what you are doing, do yourself a favor and don't touch anything. Ask someone who knows!

Parts required

Here's the needed parts for this example:

- Relay Module
- Arduino UNO – read Best Arduino Starter Kits
- PIR Motion Sensor
- Lamp Cord Set (view on eBay)

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Code

Copy the following code to your Arduino IDE and upload it to your Arduino board.

Warning: you shouldn't upload new code while your Arduino is connected to the relay.

```
/*********

  Rui Santos

  Complete project details at https://randomnerdtutorials.com

*********/



// Relay pin is controlled with D8. The active wire is connected to Normally
Closed and common

int relay = 8;

volatile byte relayState = LOW;



// PIR Motion Sensor is connected to D2.

int PIRInterrupt = 2;
```

```
// Timer Variables

long lastDebounceTime = 0;

long debounceDelay = 10000;



void setup() {

  // Pin for relay module set as output

  pinMode(relay, OUTPUT);

  digitalWrite(relay, HIGH);

  // PIR motion sensor set as an input

  pinMode(PIRInterrupt, INPUT);

  // Triggers detectMotion function on rising mode to turn the relay on, if
the condition is met

  attachInterrupt(digitalPinToInterrupt(PIRInterrupt), detectMotion, RISING);

  // Serial communication for debugging purposes

  Serial.begin(9600);

}



void loop() {

  // If 10 seconds have passed, the relay is turned off

  if((millis() - lastDebounceTime) > debounceDelay && relayState == HIGH){

    digitalWrite(relay, HIGH);

    relayState = LOW;

    Serial.println("OFF");

  }
  delay(50);

}



void detectMotion() {

  Serial.println("Motion");

  if(relayState == LOW){

    digitalWrite(relay, LOW);

  }

  relayState = HIGH;

  Serial.println("ON");

  lastDebounceTime = millis();

}
```

View raw code

How the code works

First, we create variables to hold the pin the relay IN1 pin is connected to and to save the relay state:

```
int relay = 8;
volatile byte relayState = LOW;
```

The PIR motion sensor is connected to pin 2:

```
int PIRInterrupt = 2;
```

We need to create some auxiliary variables to handle timers with the PIR motion sensor. The lastDebounceTime variable saves the last time motion was detected. The debounceDelay saves how much time the lamp should remain on after motion is detected (here we're setting 10 seconds = 10000 milliseconds)

```
long lastDebounceTime = 0;
long debounceDelay = 10000;
```

In the setup(), we set the relay as an OUTPUT and turn it off by default:

```
pinMode(relay, OUTPUT);
```

```
digitalWrite(relay, HIGH);
```

Because we're using a normally open configuration, there is no contact between the COM and NO sockets unless you trigger the relay. The relay is triggered when the input goes below about 2 V. That means if you send a LOW signal from the Arduino, the relay turns on, and if you send a HIGH signal, the relay turns off; it works with inverted logic.

Set the PIR motion sensor as an interrupt:

```
pinMode(PIRInterrupt, INPUT);
// Triggers detectMotion function on rising mode to turn the relay on, if the
condition is met
attachInterrupt(digitalPinToInterrupt(PIRInterrupt), detectMotion, RISING);
```

Whenever the PIR motion sensor is triggered, it calls the detectMotion() function declared at the end of the code to turn the relay on:

```
void detectMotion() {
    Serial.println("Motion");
    if(relayState == LOW){
        digitalWrite(relay, LOW);
    }
    relayState = HIGH;
    Serial.println("ON");
    lastDebounceTime = millis();
}
```

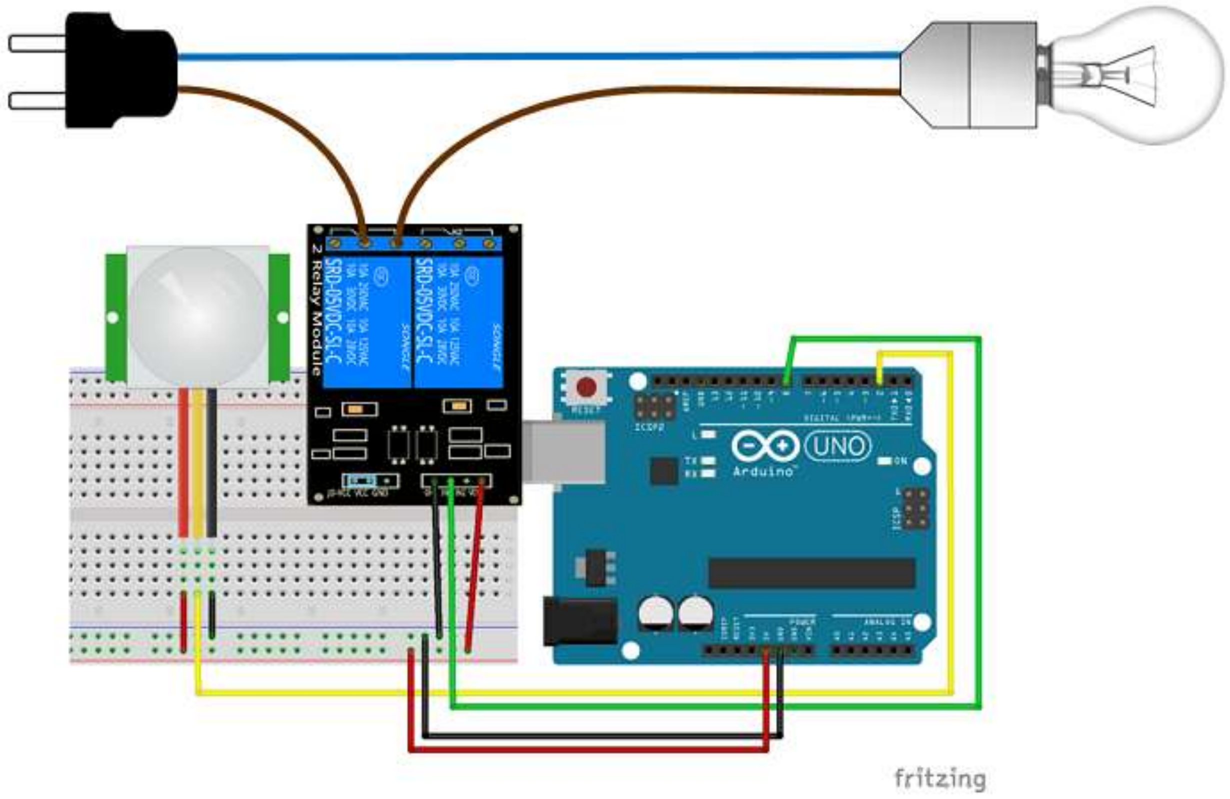
In the loop(), we check whether 10 seconds have passed since the relay is on. If that condition is true, we can turn the relay off.

```
if((millis() - lastDebounceTime) > debounceDelay && relayState == HIGH){
    digitalWrite(relay, HIGH);
    relayState = LOW;
    Serial.println("OFF");
}
```

Schematic

Assemble all the parts as shown in the schematic diagram.

Warning: do not touch any wires that are connected to mains voltage. Also make sure you have tighten all screws of the relay module.



The lamp is connected to the relay using a normally open configuration. The Arduino controls the relay through pin 8 (pin 8 is connected to the relay IN1 pin). Finally, the PIR motion sensor is connected to pin 2.

Demonstration

After uploading the code and wiring the circuit, you can test your setup.

When motion is detected, your lamp lights up. If there isn’t motion for 10 seconds, the lamp turns off.

Wrapping Up

Controlling a relay module with the Arduino is as simple as controlling an output – you just need to send HIGH or LOW signals using an Arduino digital pin. With the relay module you can control almost any AC electronics appliances (not just lamps).

We hope you've found this guide useful. If you like this project, you may also like our premium Arduino course:

- Arduino Step-by-step Projects: Build 25 Projects

We have more than 60 free tutorials and projects with the Arduino. If you're looking for a guide for a specific module, we probably have what you're looking for.

- 60+ Arduino Projects and Tutorials

Finally, you can also get access to our FREE resources here.

Thanks for reading.

January 15, 2019