

AZ-Delivery

Welcome!

Thank you for purchasing our *AZ-Delivery 4 Relays Module*. On the following pages, we will introduce you to how to use and set-up this handy device.

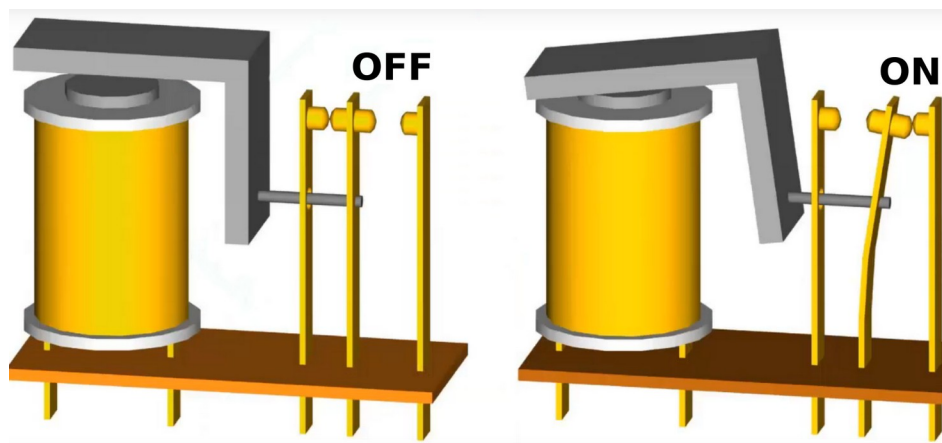
Have fun!



Az-Delivery

Relays are used to control AC circuits, switching them *ON* or *OFF*. The relay is one of the most important control elements. It is an electrical switch that responds to a signal received from the microcontroller (like in Arduino or Raspberry Pi). Relays are widely used in remote control, communications, mechatronics devices, power electronic devices, etc. They also can be used to separate powerful voltage/current electronics (like AC or DC motors, or any AC device, etc.), from microelectronics (like microcontrollers, sensors, etc.).

Inside relay, there is one mechanical switch (three yellow metal rods, with one in the middle that is bent to one side, and is movable), which is controlled by the second element of the electromagnet (yellow cylinder), as shown on the image below:



In the non active state, the mechanical switch is in the *OFF* state, NC pin is connected with common pin, and NO is unconnected. When the power is being connected to the electromagnet (via transistor and rectifier diode), this moves the switch to the active state, thus connecting the common pin to NO pin.



SAFETY WARNING!

When doing projects that are connected to mains voltage, misuse may lead to serious electrical shock!

For the sake of your own safety, be 100% sure what you are doing! Otherwise, ask someone who knows!

According to Current regulations, working with mains Voltage is reserved for qualified electricians only!

The 4 relays module consists of four relays capable of handling up to 5A 50V AC. For every relay, there are also a LED, two resistors, a NPN transistor, a rectifier diode and optocoupler.

On the DC side of the board there are six pins, four input pins for four relays, one for power supply (VCC) and one for ground (GND). There is also a two pin jumper for selecting power supply (external or VCC power supply). On the AC side there is three pin screw terminal header, where pins are labeled as: Normally Closed - NC pin, Common pin and Normally Open - NO pin.

Specifications:

TTL Control Signal: 3.3V to 12V DC

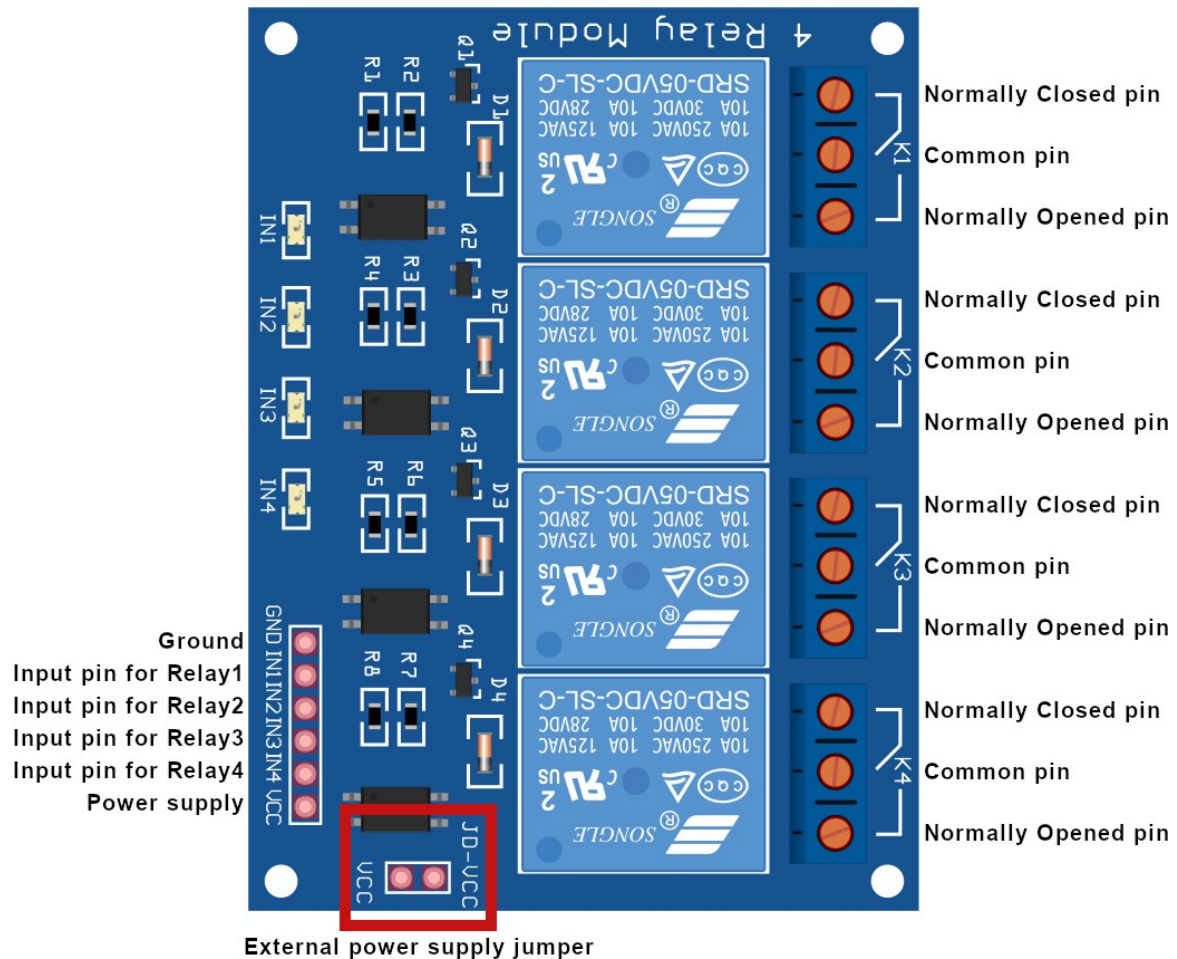
Maximum AC: 5A 50V

Maximum DC: 5A 30V

Contact Type: Both: Normally Closed – NC, Normally Opened - NO

Dimensions: 55 x 75mm [2.2 x 2.95in]

The pinout



External power supply jumper is used for selecting the power supply input. If it is left unconnected, the relays will not be powered up at all, but the LEDs on-board the module will still blink. If you connect the JD-VCC pin and VCC pin together (with two pin jumper), the module will be powered up from the VCC pin.

If you want to use external power supply, remove two pin jumper, and connect the positive side of external power supply to JD-VCC pin, and ground pin of external power supply with ground pin of the module.



External power supply

Why is there a need for external power supply?

Firstly, it is needed because sometimes a voltage regulator on-board Arduino is not powerful enough to drive the Arduino and the module.

Secondly, it is better for relays and microcontroller power supplies to be separated. Because relays are used to control the AC or powerful DC devices, and having to control electronic and powerful electronic circuits separated is a safety precautionary measure. One of the good sides of using relays is to protect the microelectronic circuits from powerful electronic circuits.

Az-Delivery

How to set-up the Arduino IDE

If you did not install Arduino IDE already, this is how to do it. Go to the link: <https://www.arduino.cc/en/Main/Software> and download installation file for your operating system platform.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle with a white infinity symbol containing a minus and a plus sign. To its right, the text reads: **ARDUINO 1.8.9**, followed by a description of the IDE as open-source software written in Java, and a note that it can be used with any Arduino board. On the right side of the page, there are links for downloading the IDE for various operating systems: Windows (installer and ZIP), Windows app (with a 'Get' button), Mac OS X, and Linux (32 bits, 64 bits, and ARM 32/64 bits). At the bottom right, there are links for Release Notes, Source Code, and Checksums.

ARDUINO 1.8.9
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

For Windows, double click on downloaded ".exe" file and follow instructions in installation window.

Az-Delivery

For Linux, download file with extension *".tar.xz"*, which then you need to extract. When you extract it, go to the extracted directory, and open terminal in that directory. You need to run two *".sh"* scripts, first called *"arduino-linux-setup.sh"*, and second called *"install.sh"*.

To run first script in terminal, run the following command:

```
sh arduino-linux-setup.sh user_name
```

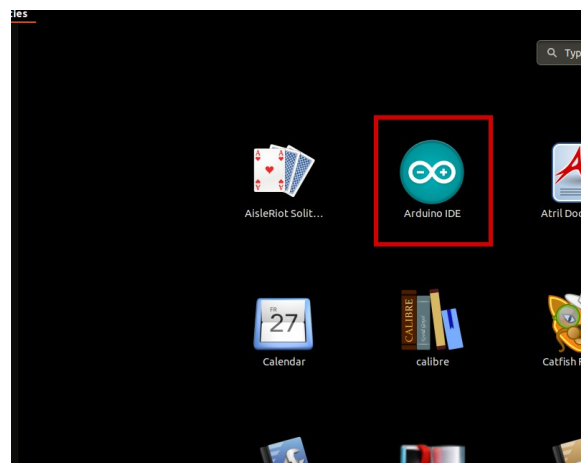
user_name - is the name of super user in the Linux operating system.

After this, you will be prompted to provide password for the super user. Wait for a few minutes for script to complete everything.

After installation of the first script, run the second called *"install.sh"* script. In terminal, run the following command:

```
sh install.sh
```

After the installation of these scripts, go to the *All Apps* to find the *Arduino IDE* installed.

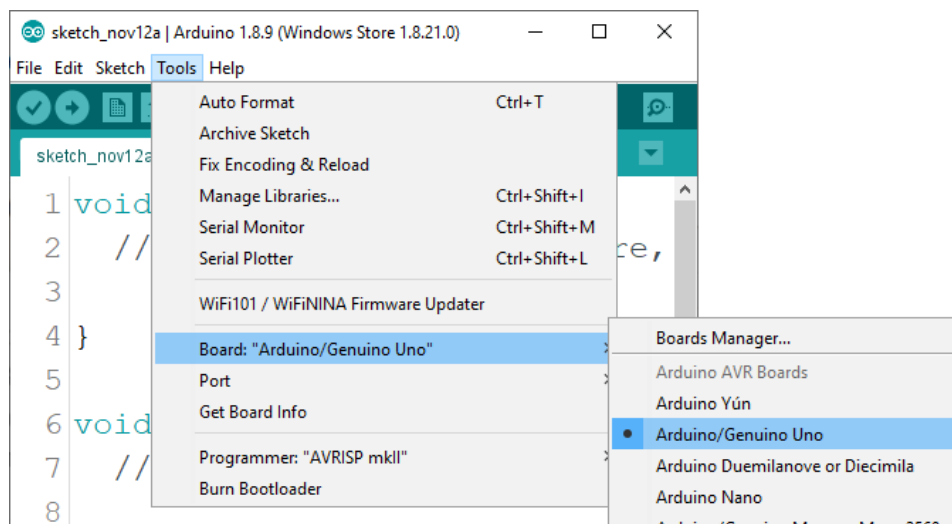


Az-Delivery

Next thing is to check if your PC can detect the Uno board. Open freshly installed *Arduino IDE*, and go to:

Tools > Board > {your board name here}

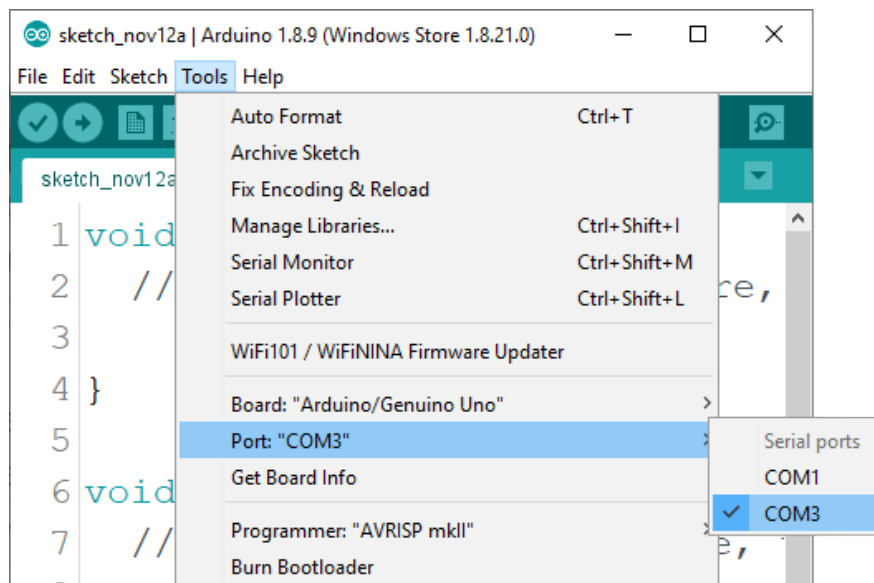
{your board name here} should be the *Arduino/Genuino Uno*, as you can see on the image below:



Az-Delivery

After this you need to select the port on which the Arduino board is connected. Go to: *Tools > Port > {port name goes here}*

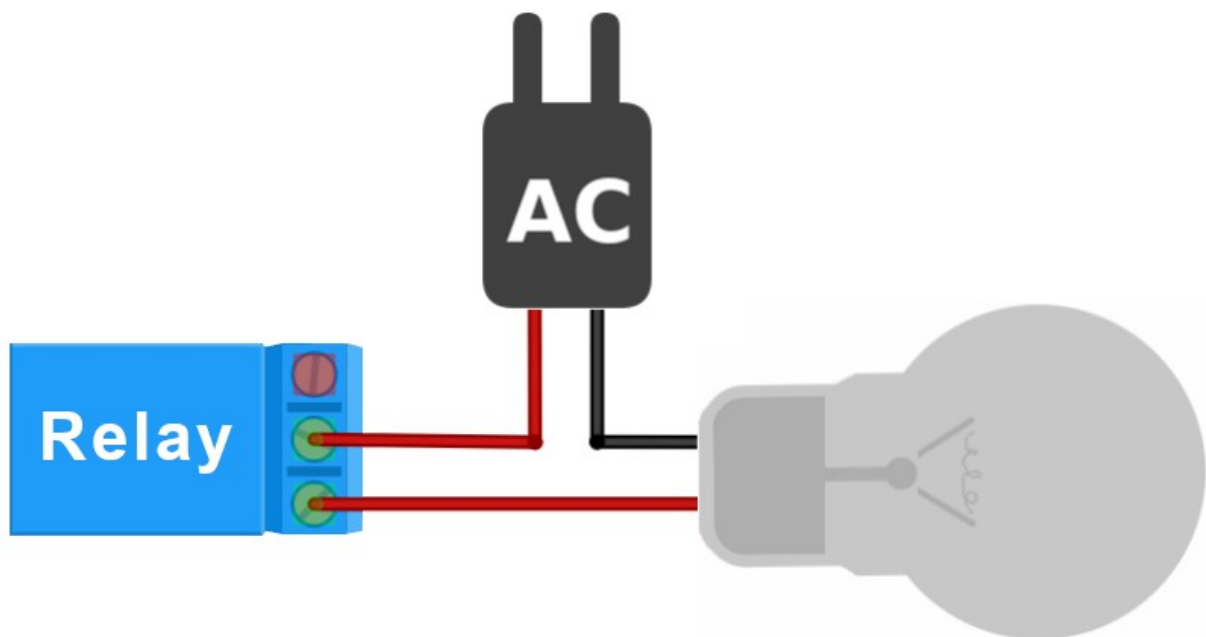
If you connected the Uno board on the usb port, there should be several port names. Because we are using *Arduino IDE* on *Windows*, port names are like on image below.



For Linux users, port name is “*/dev/ttyUSBx*” for example, where “*x*” represents specific integer number between 0 and 9, for instance.

Connecting the AC side of the relay

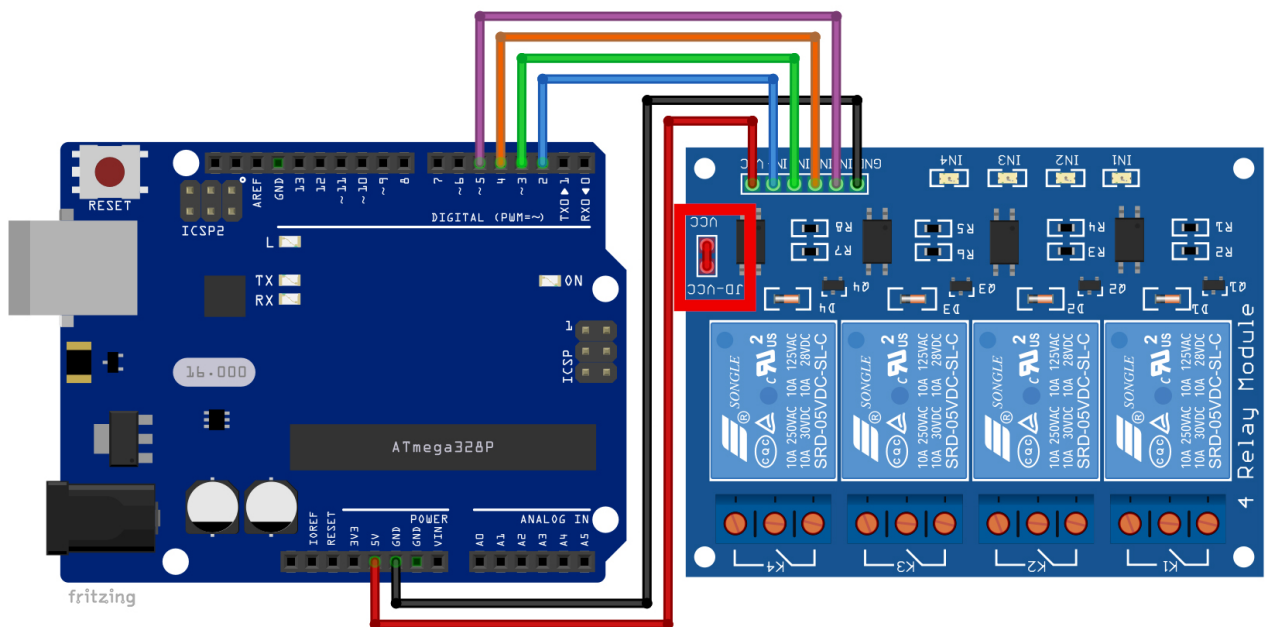
For this purpose we will be using one light bulb, a two wire cable and an AC power plug. The AC part of the connection diagram is the same for all four relays on-board 4 relays module. Connect the relay with light bulb and power plug as shown on the connection diagram below:



Module pin	>	Power plug, light bulb	
Common pin	>	One side of power plug	Red wire
Normally opened pin	>	One side of light bulb	Red wire
Light bulb	>	Power plug	
The other side of light bulb	>	The other side of power plug	Black wire

Connecting the module with Uno

Connect the 4 relays module with the Uno as shown on the following connection diagram:



Module pin > Uno pin

IN1 > D2

Purple wire

IN2 > D3

Orange wire

IN3 > D4

Green wire

IN4 > D5

Blue wire

GND > GND

Black wire

VCC > 5V

Red wire

NOTE: As you can see in the red rectangle on the connection diagram, external power supply jumper is connected, connecting JD-VCC pin with VCC pin. This means that the 4 relays module will be powered up from Arduino board via VCC pin.

Az-Delivery

Sketch example:

```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(3, OUTPUT);  
    pinMode(4, OUTPUT);  
    pinMode(5, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(2, HIGH);  
    digitalWrite(3, HIGH);  
    digitalWrite(4, HIGH);  
    digitalWrite(5, HIGH);  
    delay(1000);  
    digitalWrite(2, LOW);  
    digitalWrite(3, LOW);  
    digitalWrite(4, LOW);  
    digitalWrite(5, LOW);  
    delay(1000);  
}
```

Az-Delivery

When you upload the sketch to the Uno, you should hear clicks from relays. When the relay changes state from active to rest and vice versa, you can hear switching clicks.

Both light bulbs connected to relays should blink every second.

We can change NO/NC pin states by these lines of the code:

`digitalWrite(2, HIGH);` - **NC** pin is not connected to the common pin

NO pin is connected to the common pin

`digitalWrite(2, LOW);` - **NC** pin is connected to the common pin

NO pin is not connected to the common pin



How to set-up Raspberry Pi and Python

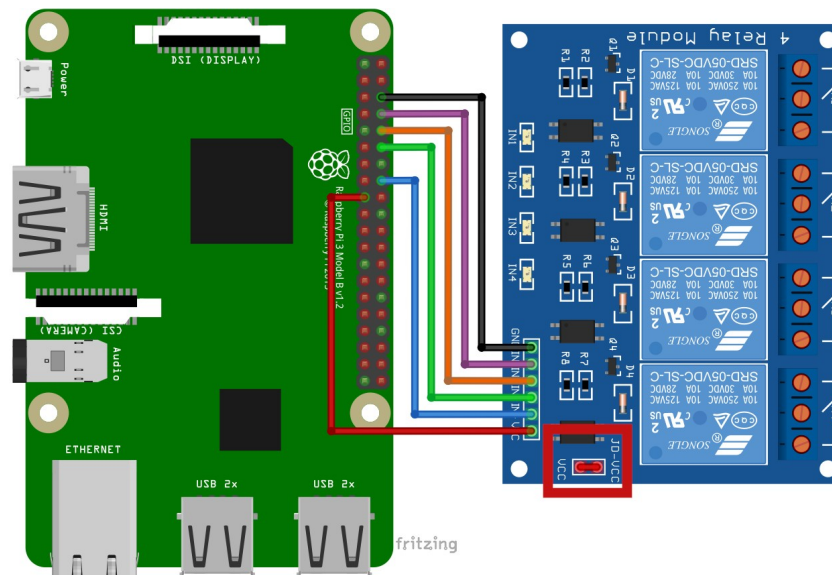
First you have to install operating system on the Raspberry Pi, then to set it up so that you can use it in the “*headless*” mode. *Headless* mode enables you to remotely connect to the Raspberry Pi, without the need for *PC* screen Monitor, mouse and keyboard. You can find detailed explanation in the free eBook "*Raspberry Pi Quick Startup Guide*", which can be found on our site:

<https://www.az-delivery.de/products/raspberry-pi-kostenfreies-e-book?ls=en>

The *Raspbian* operating system comes with the Python preinstalled.

Connecting the module with Raspberry Pi

Connect the module with Raspberry Pi as shown on the following connection diagram:



Module pin > Raspberry Pi pin

GND	>	GND	[pin 6]
IN1	>	GPIO14	[pin 8]
IN2	>	GPIO15	[pin 10]
IN3	>	GPIO18	[pin 12]
IN4	>	GPIO23	[pin 16]
VCC	>	3V3	[pin 17]

Black wire

Purple wire

Orange wire

Green wire

Blue wire

Red wire

NOTE: As you can see in the red rectangle on the connection diagram, external power supply jumper is connected, connecting JD-VCC pin with VCC pin. This means that the 4 relays module will be powered up from Raspberry Pi board via VCC pin.



Python script:

```
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
Relay1_PIN = 14
Relay2_PIN = 15
Relay3_PIN = 18
Relay4_PIN = 23
GPIO.setup(Relay1_PIN, GPIO.OUT)
GPIO.setup(Relay2_PIN, GPIO.OUT)
GPIO.setup(Relay3_PIN, GPIO.OUT)
GPIO.setup(Relay4_PIN, GPIO.OUT)
print('[press ctrl+c to end the script]')
try: # Main program loop
    while True:
        GPIO.output(Relay1_PIN, GPIO.HIGH)
        GPIO.output(Relay2_PIN, GPIO.HIGH)
        GPIO.output(Relay3_PIN, GPIO.HIGH)
        GPIO.output(Relay4_PIN, GPIO.HIGH)
        print('Normally opened pin is HIGH')
        sleep(1) # Waitmode for 1 second
        GPIO.output(Relay1_PIN, GPIO.LOW)
        GPIO.output(Relay2_PIN, GPIO.LOW)
        GPIO.output(Relay3_PIN, GPIO.LOW)
        GPIO.output(Relay4_PIN, GPIO.LOW)
        print('Normally opened pin is LOW')
        sleep(1) # Waitmode for 1 second

# Scavenging work after the end of the program
except KeyboardInterrupt:
    print('Script end!')

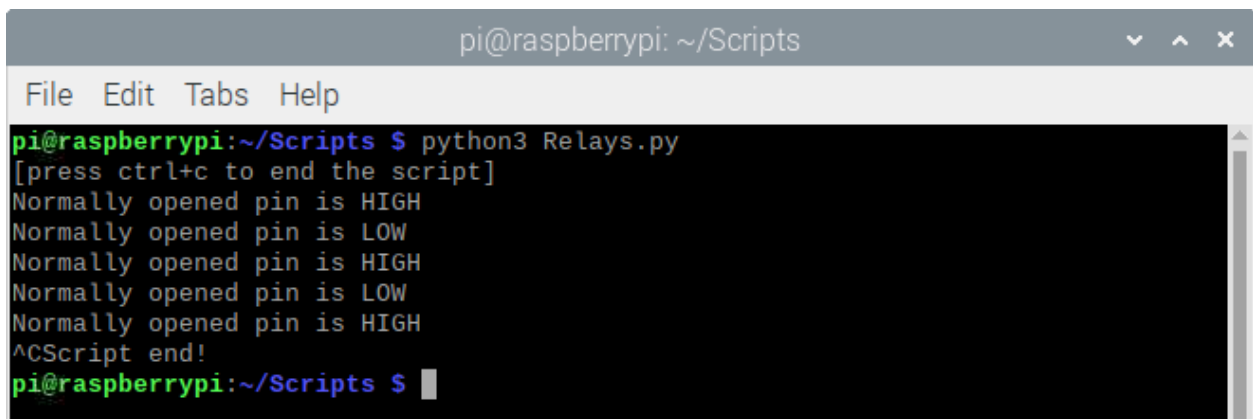
finally:
    GPIO.cleanup()
```

Az-Delivery

Save the script by the name "*Relays.py*" into default script directory. To run the script open terminal in the directory where you saved the script and run the following command:

python3 Relays.py

The output should look like the output on the image below:

A screenshot of a terminal window titled "pi@raspberrypi: ~/Scripts". The window has a menu bar with "File", "Edit", "Tabs", and "Help". The terminal output shows the command "python3 Relays.py" being executed. The output includes a prompt "[press ctrl+c to end the script]", followed by five lines of status messages: "Normally opened pin is HIGH", "Normally opened pin is LOW", "Normally opened pin is HIGH", "Normally opened pin is LOW", and "Normally opened pin is HIGH". The output ends with "^CScript end!". The prompt "pi@raspberrypi:~/Scripts \$" is visible at the bottom of the terminal window.

```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 Relays.py
[press ctrl+c to end the script]
Normally opened pin is HIGH
Normally opened pin is LOW
Normally opened pin is HIGH
Normally opened pin is LOW
Normally opened pin is HIGH
^CScript end!
pi@raspberrypi:~/Scripts $
```

To end the script press "*CTRL + C*".

The script is self explanatory.

You've done it!

Now you can use your module for various projects.



Now is the time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.

<https://az-delivery.de>

Have Fun!

Impressum

<https://az-delivery.de/pages/about-us>