

8chSSR

ebook

8 Channel Solid State Relay

Low level Trigger

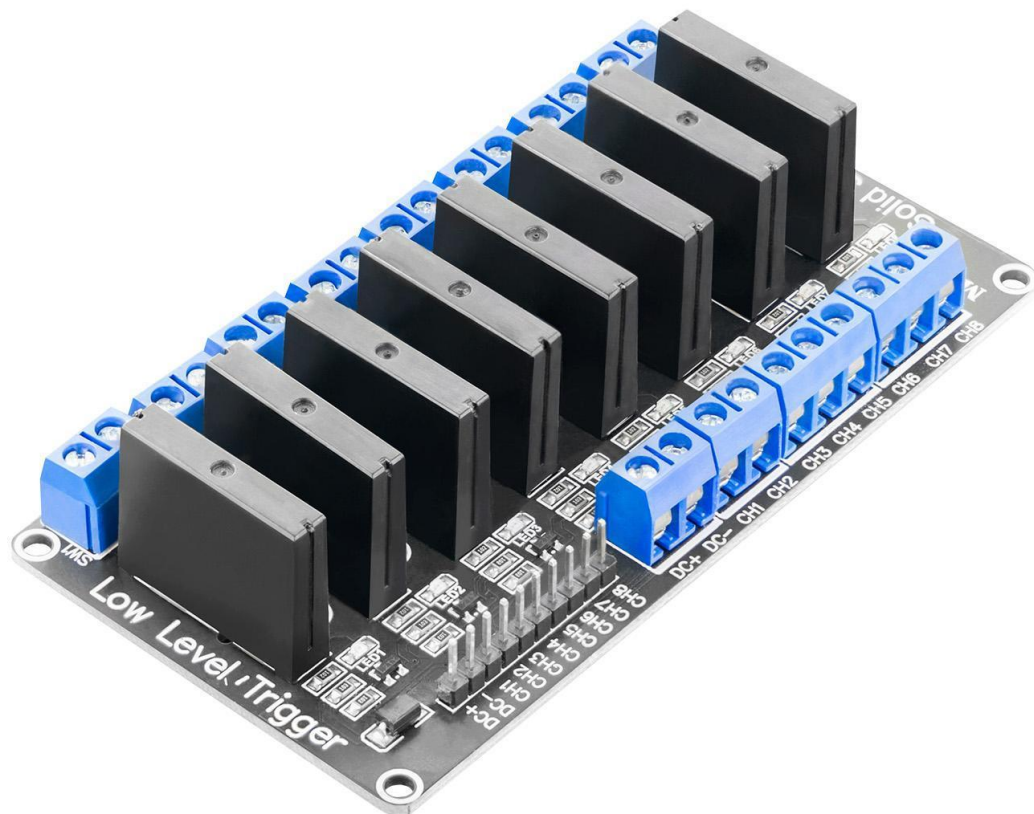


Table of Contents

Introduction	3
Features	4
Applications Examples	5
Hardware Overview	6
Board Schematic:	7
How to Use a 8 channel Solid state relay	8
Connection diagram	10
Software installation	11
Test the Arduino Code	16
Test with Raspberry pi3	21
Setting up the Raspberry Pi and Python	21
Connection diagram	22
Python example	23
Test the code	24

Introduction

This is a 5V, 8 channel low-level solid state relay module. This module is capable of switching AC voltages between 100 and 240V at up to a 2A current.

A solid state relay is similar to a mechanical relay, but it is a low trip relay. The relay therefore activates at a LOW level input. And so it can be controlled by a digital signal. This relay module generates no noise and has a much longer life. When the input logic voltage is applied to the coil, the NC disconnects from the COM by breaking the conductivity between the two. At the same time, the NO connects to the COM, allowing conductivity between the two. Depending on your wiring, the connected load will be turned on or off.

On the following pages, we will introduce you to how to use and how to set-up this handy device.

Features

Product	Solid-State Relay
Number of Channel	8
Size of the board	Size: 105 x 55 x 23mm
Weight	69.7 g
Rated Input Voltage	5VDC
Rated Load Voltage	100 to 240 VAC 50/60 Hz
Load current	2A per switch
Maximum operating current	48.8mA
Trigger mode	Low level trigger
Operating Temperature	-20 Degree C ~ + 80 Degree C
Storage Temperature	-40 Degree C ~ + 100 Degree C
Input Interface	Digital

Applications Examples

Solid-state relays are the semiconductor equivalents of the electromechanical relays, and therefore can be used to control electrical loads.

The solid-state relay is a fairly complex device, but it has a simple purpose – to activate a single output load when energized. They are efficient and effective at the job, allowing in some cases, extremely large loads to operate with very precise control voltages.

- Operations that require low-latency switching, e.g. stage light control
- Devices that require high stability, e.g. medical devices, traffic signals
- Situations that require explosion-proof, anticorrosion, moisture-proof, e.g. coal, chemical industries.
- Motor control applications :Pumps ,Compressors ,Fans ,Elevators ,Hoists, Conveyor systems.

Hardware Overview



DC+ : connected to the positive pole of the power supply (according to the voltage of the relay)

DC- : connects to the negative power pole

CH1: 1 relay module signal trigger end (low-level trigger effective)

CH2: 2 relay module signal trigger end (low-level trigger effective)

CH3: 3 relay module signal trigger end (low-level trigger effective)

CH4: 4 relay module signal trigger end (low-level trigger effective)

CH1: 5 relay module signal trigger end (low-level trigger effective)

CH2: 6 relay module signal trigger end (low-level trigger effective)

CH3: 7 relay module signal trigger end (low-level trigger effective)

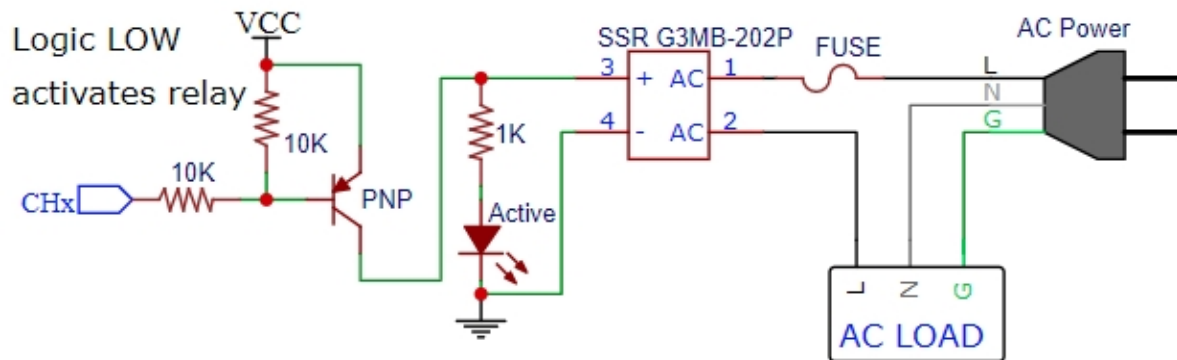
CH4: 8 relay module signal trigger end (low-level trigger effective)

High And Low Meaning:

- High-level trigger refers to the signal trigger end (IN) had a positive voltage and the negative pole of the power supply is usually between, and the triggering end of a trigger connected with the positive pole of a power supply, when the trigger end has a positive voltage or reached the trigger voltage, the relay is attracted.
- Low-level trigger refers to the signal triggering voltage between the end and the negative electrode of the power supply is 0V, or trigger terminal voltage lower voltage than the positive power supply voltage, low to can trigger, make the relay, is usually the cathode of the power supply and the triggering end of a trigger mode connection, so that the relay is attracted.

Board Schematic:

SSR component schematic



How to Use a 8 channel Solid state relay

Warning : in this example, we're going to manipulate AC voltage. Misuse can result in serious injuries. If you're not familiar with AC voltage, please ask someone who can help you. While wiring your circuit verify that everything is disconnected from AC voltage.

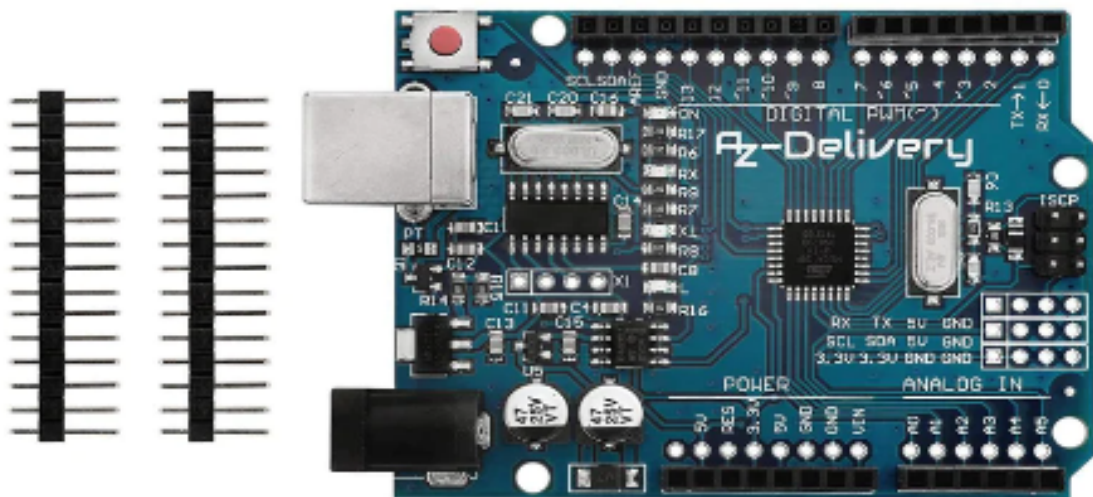
In this section of this article, we will discuss how we can hook up a module and work with Uno and raspberry board, so first of all we need a setup which described below:

NB : this sample can work with all relay channels, but in this part, we will connect only one output to the module.

Test with Microcontroller :

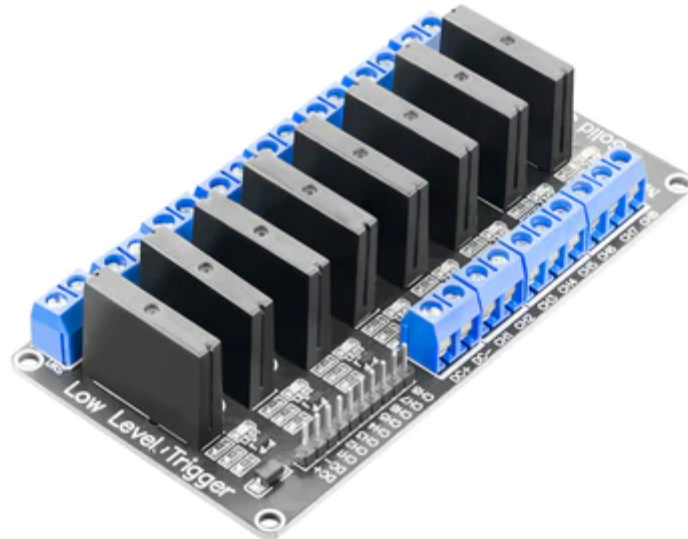
Component needed :

[-Microcontroller Board](#)



8chSSR

-8 Solid Rate Channel

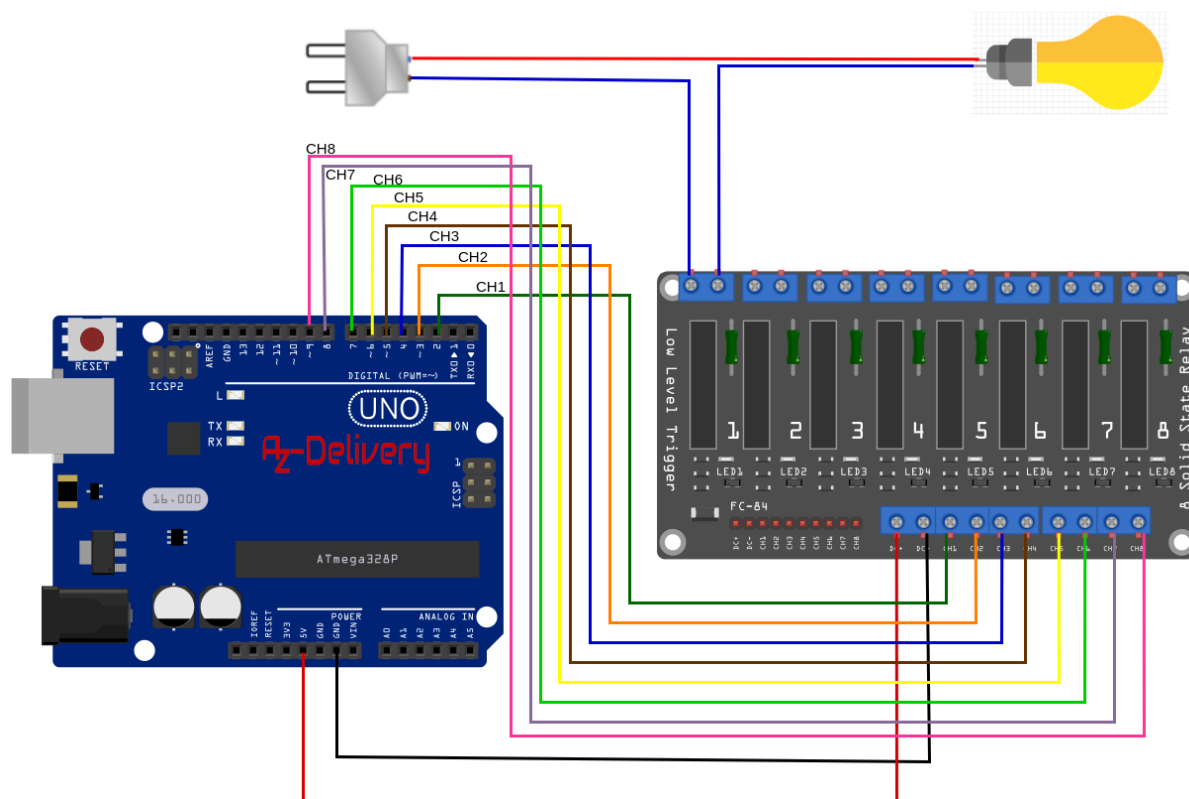


-jumper Wire



3- 220V Lamp and plug.

Connection diagram



Microcontroller	2 Solid State Relay
GND	DC-
VCC	DC+
PIN 2	CH1
PIN 3	CH2
PIN 4	CH3
PIN 5	CH4
PIN 6	CH5
PIN 7	CH6
PIN 8	CH7
PIN 9	CH8

Software installation

Download the latest version of Arduino IDE here:

<https://www.arduino.cc/en/software>

Downloads



Arduino IDE 2.0.0

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

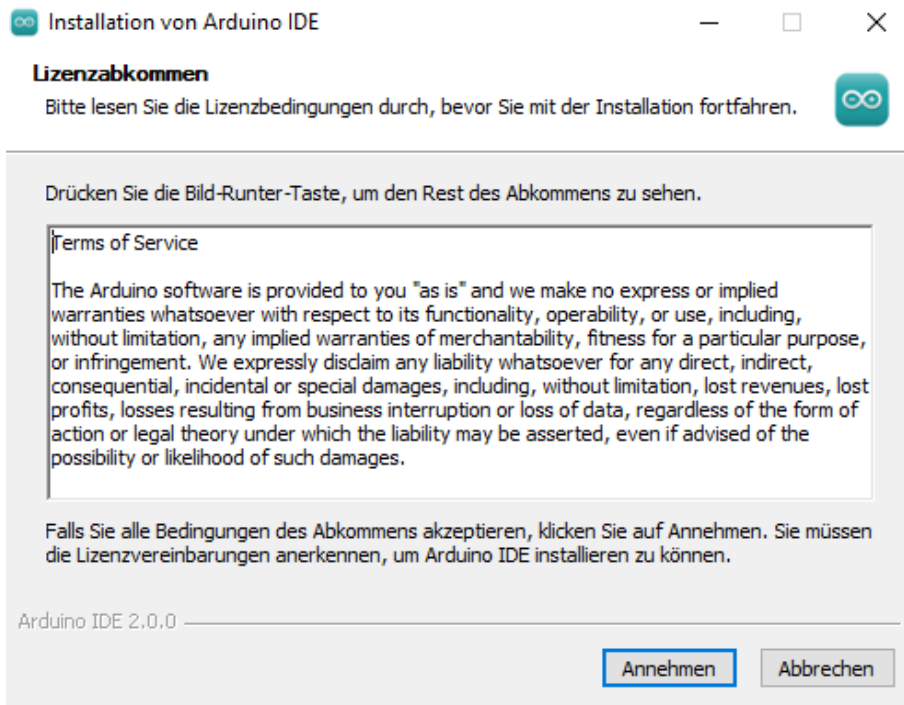
Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS 10.14: "Mojave" or newer, 64 bits

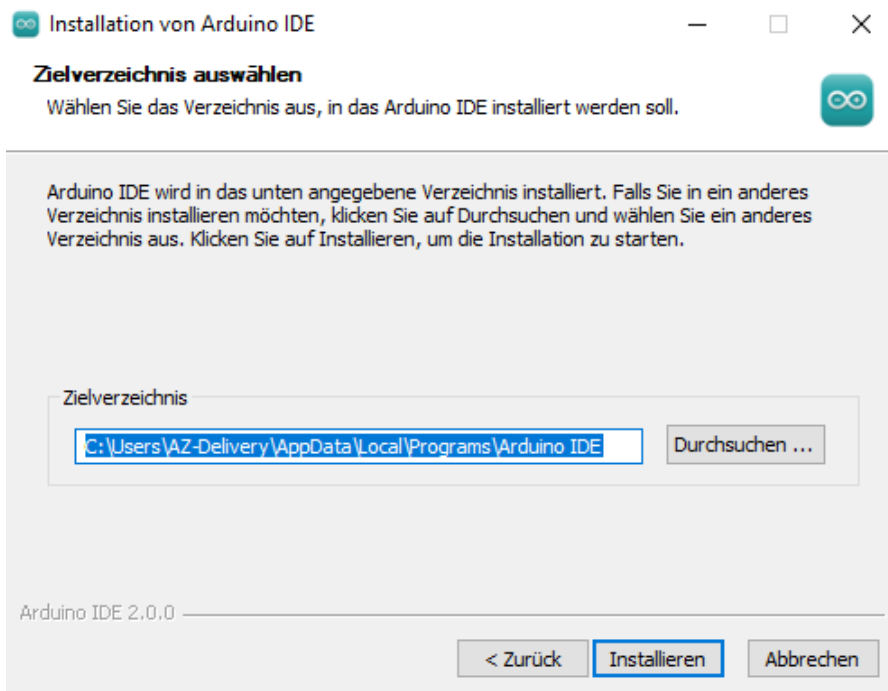
After starting the Arduino IDE installation file

"arduino-ide_2.0.0_Windows_64bit.exe" the license conditions of the software must be read and accepted.

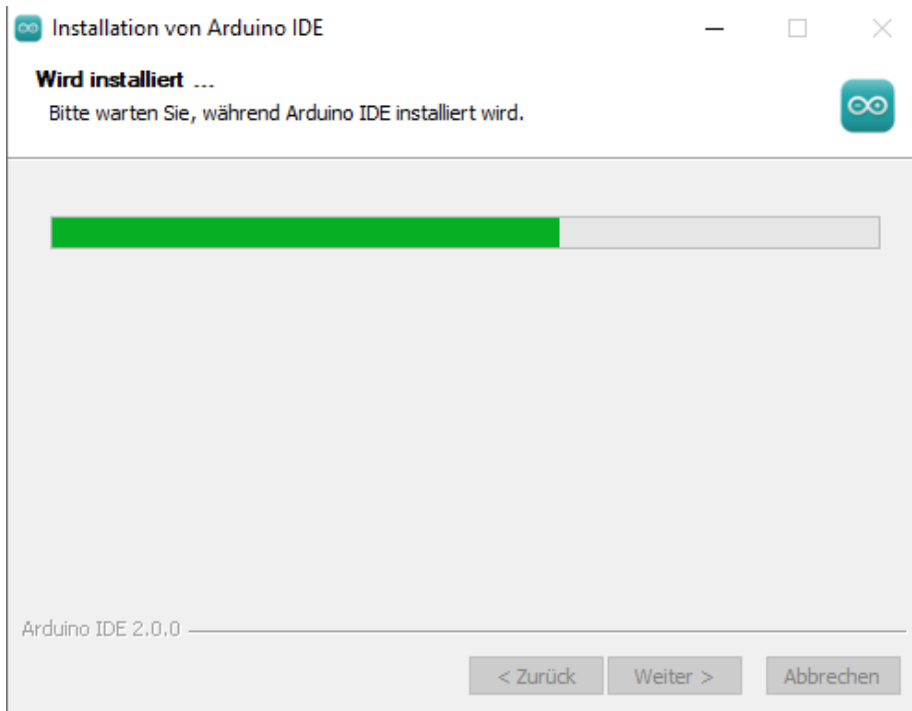


In the next step, different options can be selected for installation.

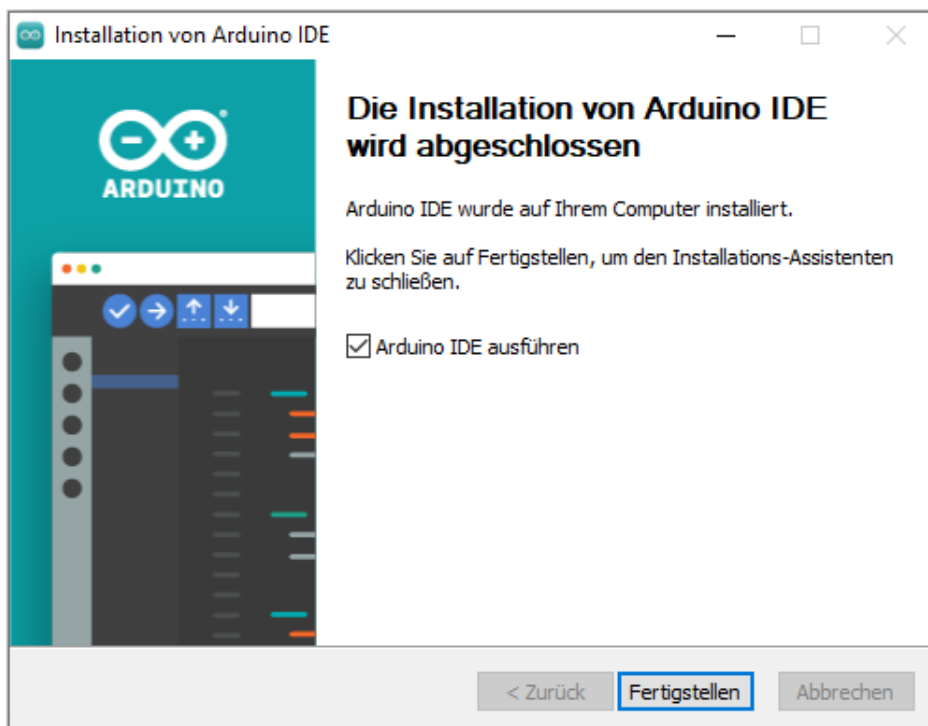
Finally, the destination folder must be specified. The installation requires approx. 500MB of free disk space.



Click on "Install" to start the installation.



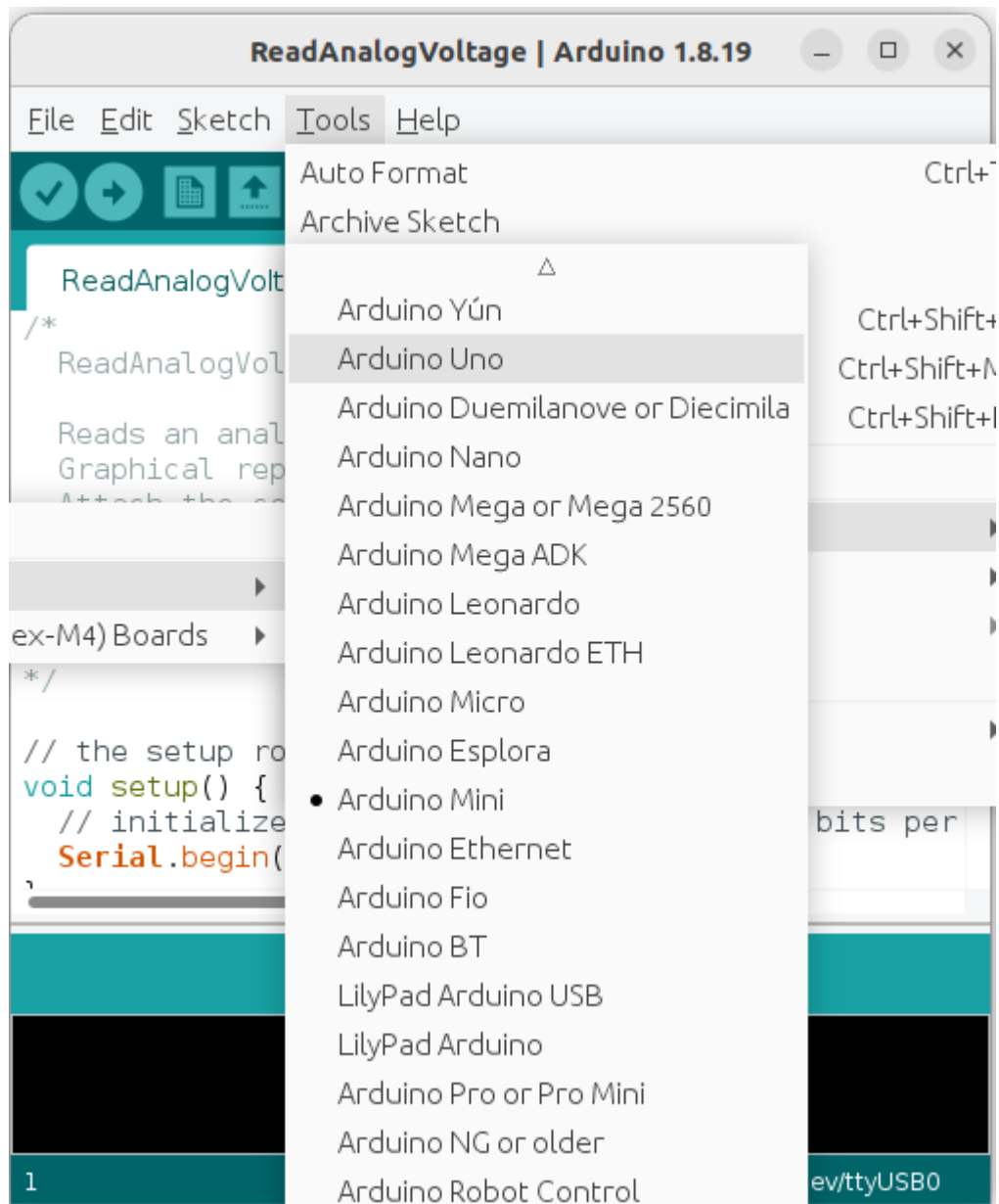
After successful installation, the installation programme can be terminated via the "**Finish**" button.



The starting window:

Select the Microcontroller Board:

Tools -> Board -> Arduino Uno



Test the Arduino Code

=> Upload the following code to the board

```
// for incoming command
int command = 0;
// type LOW if low trigger and HIGH if high trigger SSR is used
int triggerType = LOW;
int RelayOn = HIGH;
int Channels_pin_cmd[8][3] = {{2,10,11}, {3,20,21}, {4,30,31},
{5,40,41}, {6,50,51}, {7,60,61}, {8,70,71}, {9,80,81}};
void setup()
{
  Serial.begin(9600); // opens serial port, sets data rate to 9600 bps
  for (int i=0; i<8; i++)
  {
    pinMode(Channels_pin_cmd[i][0], OUTPUT);
  }
  if(triggerType == LOW)
  {
    RelayOn = LOW;
  }
  for (int i=0; i<8; i++)
  {
    digitalWrite(Channels_pin_cmd[i][0], HIGH); // turn OFF all channels
  }
  Menu();
}
void loop() {
  // send data only when you receive data:
  if (Serial.available() > 0)
  {
    // read the incoming byte:
    command = Serial.parseInt();
    if (command == 1)
    {
      for (int i=0; i<8; i++)
      {
        digitalWrite(Channels_pin_cmd[i][0], RelayOn);
      }
    }
  }
}
```



```
        Serial.println("turn ON all channel");
    }
    else if (command == 800)
    {
        for (int i=0; i<8; i++)
        {
            digitalWrite(Channels_pin_cmd[i][0], !RelayOn);
        }
        Serial.println("turn OFF all channel");
    }
    else
    {
        for (int i=0; i<8; i++)
        {
            if (command == Channels_pin_cmd[i][1])
            {
                Serial.print("turn OFF channel : ");
                Serial.println(i+1);
                digitalWrite(Channels_pin_cmd[i][0], !RelayOn);
                break;
            }
            else if (command == Channels_pin_cmd[i][2])
            {
                Serial.print("turn ON channel : ");
                Serial.println(i+1);
                digitalWrite(Channels_pin_cmd[i][0], RelayOn);
                break;
            }
        }
    }
}
}
}
}

void Menu()
{
    delay(1000);
    Serial.println("-----Solid state Relay demo
    -----");
    Serial.println("Menu :");
    Serial.println("Enter X0 : close relay channel X ");
    Serial.println("Enter X1 : open  relay channel X ");
    Serial.println("X is from 1 to 8");
    Serial.println("Enter 800  : close all relay channels");
}
```

```
Serial.println("Enter 1 : open all relay channels");  
}
```

How the code Works :

```
// initialize configuration matrix as {pinNumber, close  
command, openCommand} for the 8 output  
int Channels_pin_cmd[8][3] = {{2,10,11}, {3,20,21}, {4,30,31},  
{5,40,41}, {6,50,51}, {7,60,61}, {8,70,71}, {9,80,81}};
```

```
// Set Trigger type to Low, as described in feature table the  
trigger mode is low level
```

```
int triggerType = LOW;
```

```
// Initialize relay on signal to High  
int RelayOn = HIGH;
```

```
// Configure relay pins as output  
for (int i=0; i<8; i++)  
{  
    pinMode(Channels_pin_cmd[i][0], OUTPUT);  
}  
// Trigger mode is low level, Set on signal to Low  
if(triggerType == LOW)  
{  
    RelayOn = LOW  
}
```

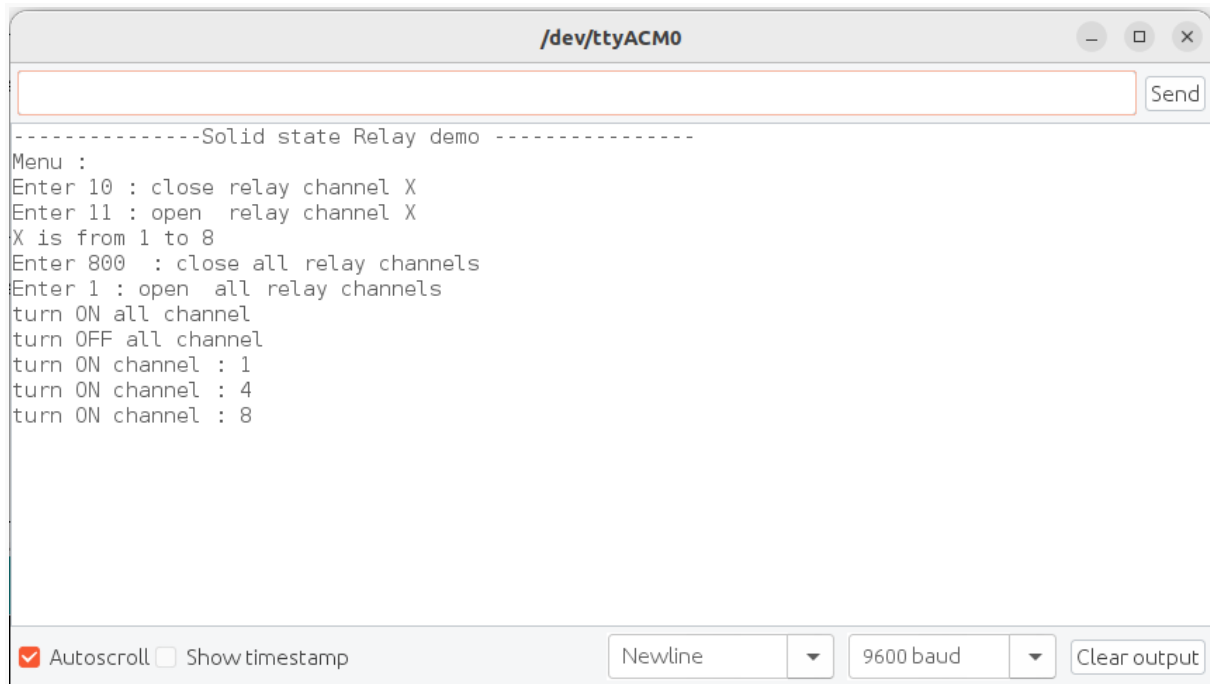
```
if (Serial.available() > 0)  
{  
    // read command entry from serial as int  
    command = Serial.parseInt();
```

```
/ if received command equal to open cmd in conf array then  
turn on the channel  
else if (command == Channels_pin_cmd[i][2])  
{  
    Serial.print("turn ON channel : ");  
    Serial.println(i+1);  
    digitalWrite(Channels_pin_cmd[i][0], RelayOn);
```

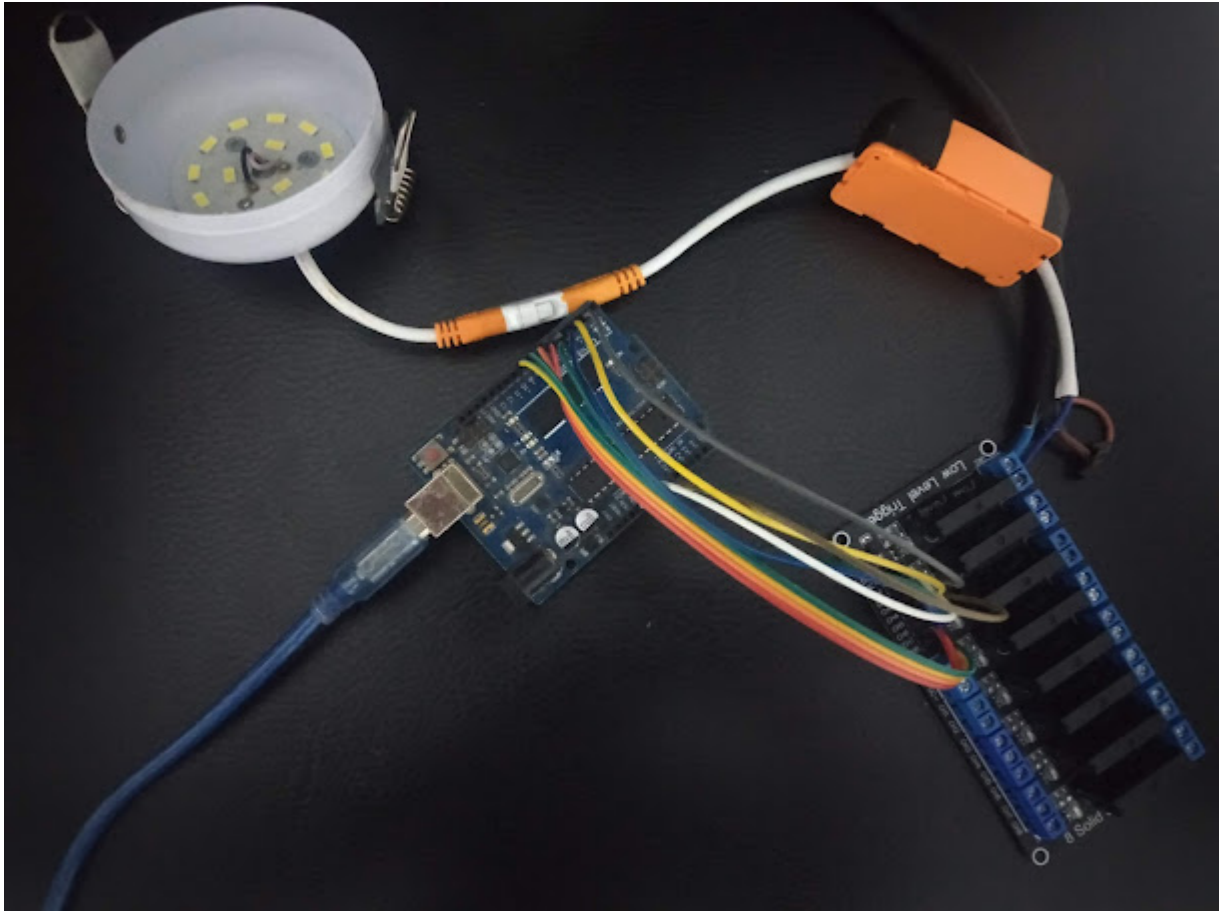
```
break;  
}
```

Upload the Code and open the serial.

If everything is fine, you should see something similar on the serial monitor



```
-----Solid state Relay demo -----  
Menu :  
Enter 10 : close relay channel X  
Enter 11 : open  relay channel X  
X is from 1 to 8  
Enter 800 : close all relay channels  
Enter 1 : open  all relay channels  
turn ON all channel  
turn OFF all channel  
turn ON channel : 1  
turn ON channel : 4  
turn ON channel : 8
```



Test with Raspberry pi3

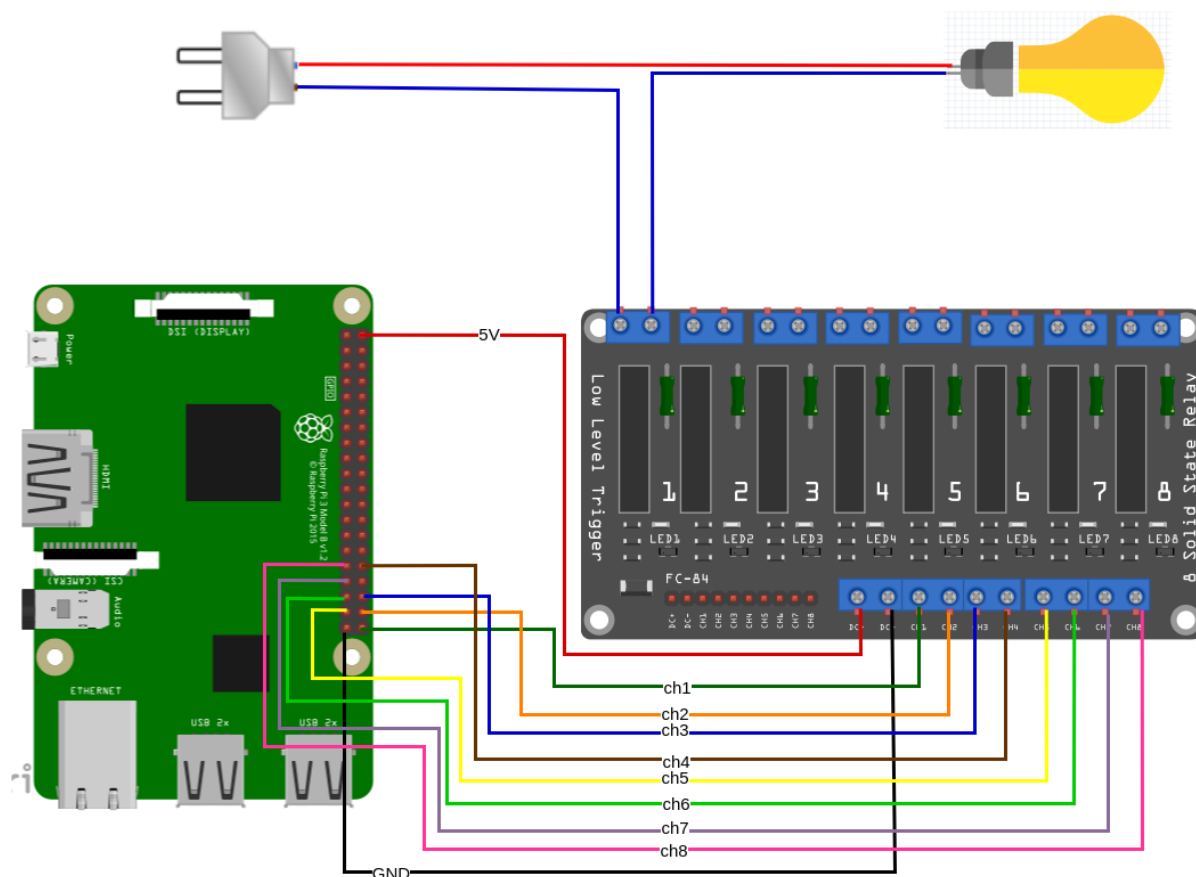
Setting up the Raspberry Pi and Python

For the Raspberry Pi, the operating system must first be installed, then everything must be set up so that it can be used in headless mode. Headless mode allows remote connection to the Raspberry Pi without the need for a PC screen monitor, mouse or keyboard. The only things used in this mode are the Raspberry Pi itself, power supply and internet connection. All this is explained in detail in the free eBook:

[Raspberry Pi Quick Startup Guide](#)

Python is preinstalled on the Raspberry Pi OS.

Connection diagram



Raspberry	8-Solid State Relay
GND	DC-
VCC	DC+
GPIO21	CH1
GPIO20	CH2
GPIO16	CH3
GPIO12	CH4
GPIO26	CH5
GPIO19	CH6
GPIO13	CH7
GPIO6	CH8

Python example

```
import RPi.GPIO as GPIO
import time
#GPIO SETUP
#GPIO21  => CH1      #GPIO20  => CH2
#GPIO16  => CH3      #GPIO12  => CH4
#PGPIO26 => CH5      #GPIO19  => CH6
#GPIO13  => CH7      #GPIO6   => CH8
Channels_pin_cmd = [[21,10,11], [20,20,21], [16,30,31], [12,40,41],
                    [26,50,51], [19,60,61], [13,70,71], [6,80,81]]
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

for i in range(8) :
    GPIO.setup(Channels_pin_cmd[i][0], GPIO.OUT)

for i in range(8) :
    GPIO.output(Channels_pin_cmd[i][0], True)

time.sleep(1)
print("-----Solid state Relay demo -----")
print("Menu :")
print("Enter X0 : close relay channel X ")
print("Enter X1 : open  relay channel X ")
print("X is from 1 to 8")
print("Enter 1 to open all relay channels")
print("Enter 800 to close all relay channels")
while True:
    cmd = int(input("Please enter your command "))
    if cmd == 1:
        for i in range(8):
            GPIO.output(Channels_pin_cmd[i][0], False)
        print("turn ON all channel")

    elif cmd == 800:
        for i in range(8):
            GPIO.output(Channels_pin_cmd[i][0], True)
        print("turn OFF all channel")

    else:
        for i in range(8):
```

```
if cmd == Channels_pin_cmd[i][1]:
    GPIO.output(Channels_pin_cmd[i][0], True)
    print("turn off channel", i+1)
    break
elif cmd == Channels_pin_cmd[i][2]:
    GPIO.output(Channels_pin_cmd[i][0], False)
    print("turn ON channel", i+1)
    break
```

Test the code

After connecting the raspberry, open the terminal and create a new .py file.
Copy the content of the code and run the sample by executing python3 filename.py

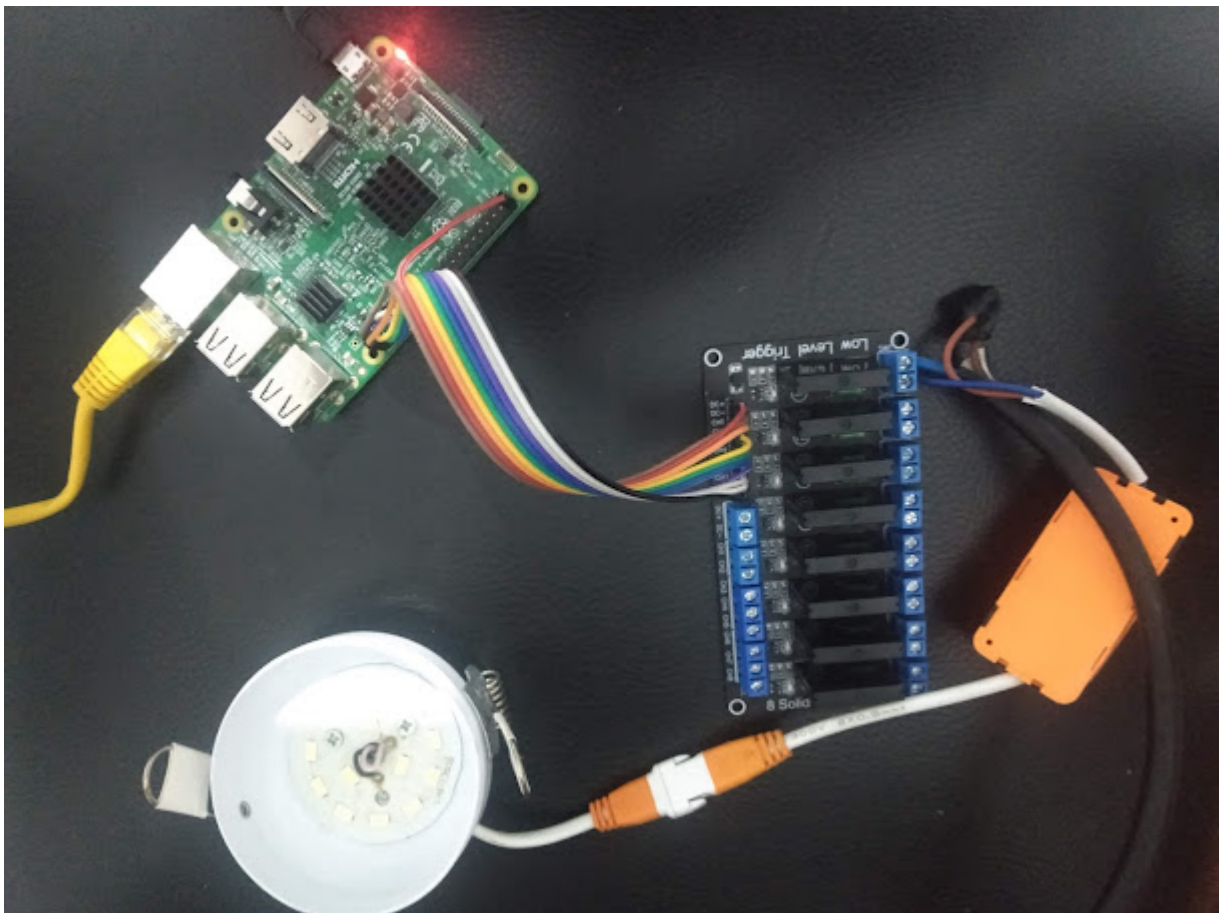
A short menu will be displayed to show you the different commands to open or close a channel.

```
pi@raspberrypi:~$ touch SolidStateRelay.py
// copy code content into SolidStateRelay.py

pi@raspberrypi:~$ python3 SolidStateRelay.py
-----Solid state Relay demo -----
Menu :
Enter X0 : close relay channel X
Enter X1 : open relay channel X
X is from 1 to 8
Enter 1 to open all relay channels
Enter 800 to close all relay channels
Please enter your command
1
turn ON all channel
Please enter your command
800
turn OFF all channel
Please enter your command
11
turn ON channel 1
Please enter your command
21
turn ON channel 2
```


8chSSR

Please enter your command
31
turn ON channel 3
Please enter your command
41
turn ON channel 4
Please enter your command
51
turn ON channel 5
Please enter your command
81
turn ON channel 8
Please enter your command
800
turn OFF all channel
Please enter your command



You've done it, you can now use your module for your projects :)

Now it is 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 microelectronics and accessories, 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>