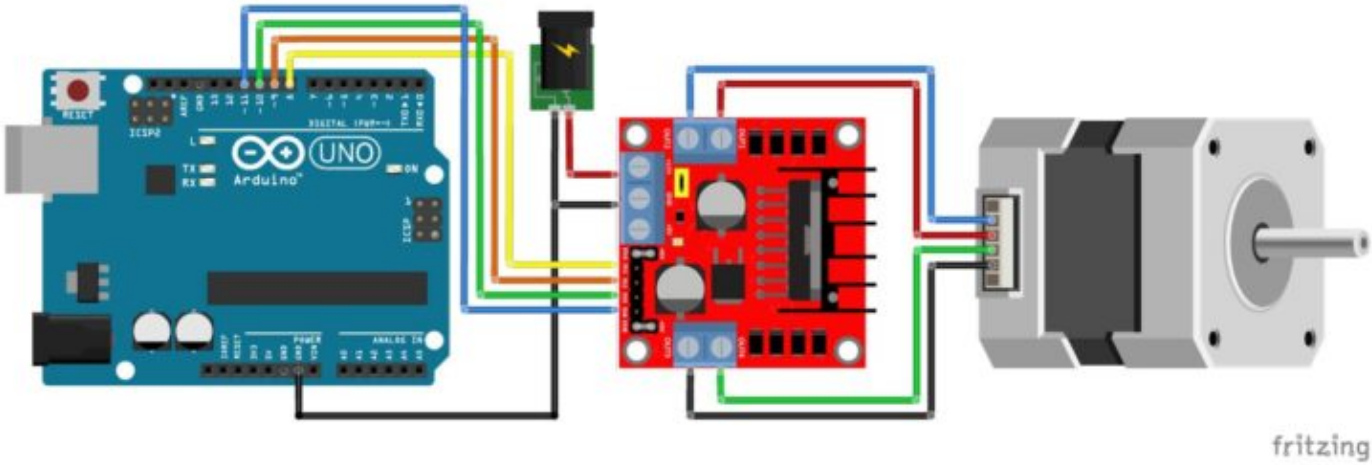
Makerguides.com

Control a stepper motor with L298N motor driver and Arduino

Written by Benne de Bakker (https://www.makerguides.com/author/benne-de-bakker/)



In this article you will learn how to control a stepper motor with the L298N motor driver (https://www.amazon.com/HiLetgo-Controller-Stepper-H-Bridge-Mega2560/dp/B07BK1QL5T/ref=as_li_ss_tl?keywords=l298n&qid=1562745291&s=gateway&sr=8-3&linkCode=ll1&tag=makerguides-20&linkId=3730d8f3d0b471b6cfb1a3acd6e96577&language=en_US). This driver board is usually used to control DC motors, but it is also an inexpensive alternative to control stepper motors! It can control both the speed and the spinning direction of most stepper motors like a NEMA 17.

I have included a wiring diagram and many example codes. In the first example we will look at the **Stepper.h** Arduino library. I highly recommend to also take a look at the example codes for the **AccelStepper** library at the end of this tutorial. This library is fairly easy to use and can greatly improve the performance of your hardware.

After each example, I break down and explain how the code works, so you should have no problems modifying it to suit your needs.

If you would like to learn more about other stepper motor drivers, then the articles below might be useful:

- How to control a stepper motor with A4988 driver and Arduino (https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/)
- 28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial (https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/)
- How to control a Stepper Motor with Arduino Motor Shield Rev3 (https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/)

The Arduino Motor Shield Rev3 (https://amzn.to/2PYs2fe) also uses a L298 driver.

Supplies

Hardware components

(https://amzn.to/35vYMU)	L298N motor d
(https://www.amazon.com/Adafruit-PID-324-Stepper-motor/dp/B01N30ISYC/ref=as_li_ss_tl?keywords=nema+17&qid=1575555491&s=electronics&sr=1-2&linkCode=ll1&tag=makerguides-20&linkId=e6745e523bbdb6d5ed267fde98509ff5&language=en_US)	NEMA 17 stepp Stepper-motor/ keywords=nem 2&linkCode=ll1 20&linkId=e67
(https://amzn.to/374aJjX)	Arduino Uno R
	12V power sup
	Jumper wires (M
	USB cable type

Software

Arduino IDE (https://www.arduino.cc/en/Main/Software)

Information about the L298N Motor Driver

The L298N Motor Driver Board (https://www.amazon.com/HiLetgo-Controller-Stepper-H-Bridge-Mega2560/dp/B07BK1QL5T/ref=as_li_ss_tl?keywords=l298n&qid=1562745291&s=gateway&sr=8-3&linkCode=ll1&tag=makerguides-20&linkId=3730d8f3d0b471b6cfb1a3acd6e96577&language=en_US) is built around the L298 dual full-bridge driver, made by STMicroelectronics. With this motor driver you can control DC motors, stepper motors, relays, and solenoids. It comes with two separate channels, called A and B, that you can use to drive 2 DC motors, or 1 stepper motor when combined.

The L298N is usually mounted on a (red) breakout board, which makes wiring a lot easier. The breakout board also includes a 78M05 5 V power regulator.

Why is my stepper motor getting HOT?

One thing that is very important to remember is that the L298 **does not have an easy way to set a current limit** unlike other stepper motor drivers like the A4988 (tutorial (https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/#current-

limit)). This means that the current draw depends on the relationship between the inductance and resistance (L/R) of the stepper motor that you connect to it. When the motor draws too much current, you can damage the driver and the motor will get hot!

What this means for you, is that you need to be careful when selecting the stepper motor and power supply to use with this motor driver. **Not all stepper motors will work!** The L298N operating voltage is between 4.8 and 46 volts (max 35 V when mounted on the breakout board). Since the driver can supply a **maximum of 2 amperes per channel**, you need to find a stepper motor that can be used in this voltage range and doesn't exceed the maximum current rating.

Check the datasheet of your stepper motor and look for the voltage/current draw of the motor. If you can't find the datasheet, you can measure the resistance of one of the windings and use the following formula to get an estimation of the current draw:

$$I = U \div R \text{ or Current draw (A) = Supply voltage (V) } \div \text{ Winding resistance } (\Omega)$$

I would try to find a motor that draws less than 2 A at the voltage that you want to use.

The motor I used for this tutorial draws around 1 A at 5 V. I also found this stepper motor from Adafruit (https://amzn.to/2YjOKkJ) that works great at 12V and only draws 350 mA.

If the motor you want to drive doesn't work with the L298N motor driver, it is best to use a chopper drive instead. I wrote tutorials for the A4988 (https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/) and DRV8825 (https://www.makerguides.com/drv8825-stepper-motor-driver-arduino-tutorial/) drivers that work great with many stepper motors.

L298N Motor Driver Specifications

Operating voltage	5 – 35 V
Logic voltage	4.5 – 7 V
Max current	2 A per channel or 4 A max
Motor controller	L298N, drives 2 DC motors or 1 stepper motor
Voltage regulator	78M05
Module dimensions	43 x 43 x 28 mm
Hole dimensions	3.2 mm, 37 mm spacing
Cost	Check price (https://amzn.to/2ZDdDba)

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

L298N Datasheet

L298N Pinout

The L298 comes in several different packages, the pinout for the L298N (Multiwatt15) is given below:

Pin no.	Name	Function
1, 15	Sense A, Sense B	The sense resistor needs to be connected between this pin and GND (not used on breakout board).
2, 3	Out 1, Out2	Outputs of the Bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	Vs	Supply Voltage for the Power Output Stages
5, 7	Input 1, Input 2	TTL Compatible Inputs of Bridge A
6, 11	Enable A, Enable B	TTL Compatible Enable Input: the LOW state disables the bridge A (enable A) and/or the bridge B (enable B).
8	GND	Ground
9	VSS	Supply Voltage for the Logic Blocks.
10, 12	Input 3, Input 4	TTL Compatible Inputs of the Bridge B.
13, 14	Out 3, Out4	Outputs of the Bridge B; the current that flows through the load connected between these two pins is monitored at pin 15.

Wiring – Connecting L298N to stepper motor and Arduino

The wiring diagram/schematic below shows you how to connect a stepper motor, power supply, and Arduino to the L298N breakout board.

L298N motor driver with stepper motor and Arduino wiring diagram.

The connections are also given in the table below:

L298N Connections

L298N	Connection
+12V	5 – 35 V power supply
GND	Power supply and Arduino ground
12 V jumper	Remove if motor power > 12 V!
5V+ (optional)	5 V Arduino if 12 V jumper is removed
IN1	Pin 8 Arduino
IN2	Pin 9 Arduino
IN3	Pin 10 Arduino
IN4	Pin 11 Arduino
ENA and ENB jumper	Leave installed
OUT1 + OUT2	Stepper motor coil A
OUT3 + OUT4	Stepper motor coil B

Important note: remove the +12V jumper if you are using a power supply higher than 12 V.

When the +12V jumper is attached, the on-board voltage regulator is enabled and it will create the 5 V logic voltage. When you remove the jumper, you need to provide the board with 5 V from the Arduino.

You also need to keep both the ENA and ENB jumpers in place so the the motor is always enabled.

How to determine the stepper motor wiring?

If you can't find the datasheet of your stepper motor, it can be difficult to figure out how to wire your motor correctly. I use the following trick to determine how to connect 4 wire bipolar stepper motors:

The only thing you need to identify is the two pairs of wires which are connected to each of the two coils. One coil gets connected to OUT1 and OUT2 and the other to OUT3 and OUT4, the polarity doesn't matter.

To find the two wires from one coil, do the following with the motor disconnected:

1. Try to spin the shaft of the stepper motor by hand and notice how hard it is to turn.
2. Now pick a random pair of wires from the motor and touch the bare ends together.
3. Next, try to spin the shaft of the stepper motor again.
4. **If you feel a lot of resistance, you have found a pair of wires from the same coil.** If you can spin the shaft freely, try another pair of wires.

Now connect the two coils to the pins shown in the wiring diagram above.

(If it is still unclear, please leave a comment below, more info can also be found on the RepRap.org wiki (https://reprap.org/wiki/Stepper_wiring))

Stepper.h library example code for L298N driver with stepper motor and Arduino

You can upload the following example code to your Arduino using the Arduino IDE (https://www.arduino.cc/en/main/software).

This example uses the **Stepper.h** library, which should come pre-installed with the Arduino IDE. You can find it by going to **Sketch > Include Library > Stepper**.

This sketch turns the stepper motor 1 revolution in one direction, pauses, and then turns 1 revolution in the other direction.

```
 /* Example sketch to control a stepper motor with L298N motor driver, Arduino UNO and Stepper.h library. More info: https://www.makerguides.com */

// Include the Stepper library:
#include <Stepper.h>

// Define number of steps per revolution:
const int stepsPerRevolution = 200;

// Initialize the stepper library on pins 8 through 11:
Stepper myStepper = Stepper(stepsPerRevolution, 8, 9, 10, 11);

void setup() {
  // Set the motor speed (RPMs):
  myStepper.setSpeed(100);
}

void loop() {
  // Step one revolution in one direction:
  myStepper.step(200);

  delay(2000);

  // Step on revolution in the other direction:
  myStepper.step(-200);

  delay(2000);
}
```

How the code works:

The sketch starts by including the Stepper.h Arduino library. More information about this library can be found on the Arduino website (https://www.arduino.cc/en/Reference/Stepper).

```
// Include the Stepper library:
#include <Stepper.h>
```

Next we need to define how many steps it takes for the motor to rotate 1 revolution. n this example we will be using the motor in **full-step mode**. This means it takes 200 steps to rotate 360 degrees. You can change this value if you want if you are using a different type of stepper motor or setup.

```
// Define number of steps per revolution:
const int stepsPerRevolution = 200;
```

After this, you need to create a new instance of the Stepper class, which represents a particular stepper motor connected to the Arduino. For this we use the function `Stepper(steps, pin1, pin2, pin3, pin4)` where steps is the number of steps per revolution and pin1 through pin4 are the pins used to drive the stepper motor. In our case these are pins 8, 9, 10 and 11.

```
// Initialize the stepper library on pins 8 through 11:
Stepper myStepper = Stepper(stepsPerRevolution, 8, 9, 10, 11);
```

In this case I called the stepper motor 'myStepper' but you can use other names as well, like 'z_motor' or 'liftmotor' etc. `Stepper liftmotor = Stepper(stepsPerRevolution, 8, 9, 10, 11);` . The name 'myStepper' will be used to set the speed and number of steps for this particular motor. Note that you can create multiple stepper objects with different names if you want to control more than one motor.

In the `setup()` we define the speed of the motor. You can set the speed of the motor in **RPM** with the function `setSpeed(rpm)` . I set it to 100, so we should see around 1.6 revolutions per second.

```
  // Set the motor speed (RPMs):
  myStepper.setSpeed(100);
```

In the loop section of code, we simply call the `step(steps)` function which turns the motor a specific number of steps at a speed determined by the `setSpeed(rpm)` function. Passing a negative number to this function reverses the spinning direction of the motor.

```
void loop() {
  // Step one revolution in one direction:
  myStepper.step(200);

  delay(2000);
```

```
22.
23.     // Step on revolution in the other direction:
24.     myStepper.step(-200);
25.
26.     delay(2000);
27.   }
```

Note that the **step(steps)** function is **blocking**, this means it will wait until the motor has finished moving to pass control to the next line in your sketch.

Installing the AccelStepper library

In the following three examples I will show you how you can control both the speed, the direction and the number of steps the stepper motor should take. In this example I will be using the **AccelStepper** library.

The AccelStepper library written by Mike McCauley is an awesome library to use for your project. One of the advantages is that it supports acceleration and deceleration, but it has a lot of other nice functions too.

You can download the latest version of this library here (https://www.airspayce.com/mikem/arduino/AccelStepper/files.html) or click the button below.

AccelStepper-1.59.zip

You can install the library by going to **Sketch > Include Library > Add .ZIP Library...** in the Arduino IDE.

Another option is to navigate to **Tools > Manage Libraries...** or type Ctrl + Shift + I on Windows. The Library Manager will open and update the list of installed libraries.

You can search for ‘accelstepper‘ and look for the library by Mike McCauley. Select the latest version and then click Install.

1. Continuous rotation AccelStepper example code

The following sketch can be used to run one or more stepper motors continuously at a constant speed. (No acceleration or deceleration is used).

```
 /* Example sketch to control a stepper motor with L298N motor driver, Arduino UNO and AccelStepper.h library. Contiuous rotation. More info: https://www.makerguides.com */

// Include the AccelStepper library:
#include <AccelStepper.h>

// Define the AccelStepper interface type:
#define MotorInterfaceType 4

// Create a new instance of the AccelStepper class:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

void setup() {
  // Set the maximum speed in steps per second:
  stepper.setMaxSpeed(1000);
}

void loop() {
```

```
18.     // Set the speed of the motor in steps per second:
19.     stepper.setSpeed(500);
20.     // Step the motor with constant speed as set by setSpeed():
21.     stepper.runSpeed();
22.   }
```

How the code works:

The first step is to include the library with `#include <AccelStepper.h>` .

```
3.  // Include the AccelStepper library:
4.  #include <AccelStepper.h>
```

The next step is to define the motor interface type. The motor interface type must be set to 4 when using a 4 wire stepper motor in full-step mode (200 steps/revolution). You can find the other interface types here (https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html#a608b2395b64ac15451d16d0371fe13ce).

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 `motorInterfaceType` , the compiler will replace it with the value 4 when the program is compiled.

```
6.  // Define the AccelStepper interface type:
7.  #define MotorInterfaceType 4
```

Next, you need to create a new instance of the AccelStepper class with the appropriate motor interface type and connections.

In this case I called the stepper motor ‘stepper’ but you can use other names as well, like ‘z_motor’ or ‘liftmotor’ etc. `AccelStepper liftmotor = AccelStepper(motorInterfaceType, 8, 9, 10, 11);` . As you saw in the previous example, the name that you give to the stepper motor will be used later to set the speed, position, and acceleration for that particular motor. You can create multiple instances of the AccelStepper class with different names and pins. This allows you to easily control 2 or more stepper motors at the same time.

```
// Create a new instance of the AccelStepper class:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);
```

In the `setup()` section of the code we define the maximum speed in steps/second. Speeds of more than 1000 steps per second can be unreliable, so I set this as the maximum. Note that I specify the name of the stepper motor ('stepper'), for which I want to define the maximum speed. If you have multiple stepper motors connected, you can specify a different speed for each motor:

```
void setup() {
  // Set the maximum speed in steps per second:
  stepper.setMaxSpeed(1000);
  stepper2.setMaxSpeed(300);
}
```

In the `loop()` we first set the speed that we want the motor to run at. For this, we use the function `setSpeed()` . (you can also place this in the setup section of the code).

`stepper.runSpeed()` polls the motor and when a step is due executes 1 step. This depends on the set speed and the time since the last step. If you want to change the direction of the motor, you can set a negative speed: `stepper.setSpeed(-400);` turns the motor the other way.

```
void loop() {
  // Set the speed of the motor in steps per second:
  stepper.setSpeed(500);
  // Step the motor with constant speed as set by setSpeed():
  stepper.runSpeed();
}
```

---

2. Example code to control number of steps or revolutions

With the following sketch you can control both the speed, direction, and the number of steps/revolutions.

In this case, the stepper motor turns 2 revolutions clockwise with 200 steps/sec, then turns 1 revolution counterclockwise at 600 steps/sec, and lastly turns 3 revolutions clockwise at 400 steps/sec.

```
 /* Example sketch to control a stepper motor with L298N motor driver, Arduino UNO and AccelStepper.h library. Number of steps or revolutions. More info: https://www.makerguides.com */

 // Include the AccelStepper library:
 #include <AccelStepper.h>

 // Define the AccelStepper interface type:
 #define MotorInterfaceType 4

 // Create a new instance of the AccelStepper class:
 AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

 void setup() {
   // Set the maximum steps per second:
   stepper.setMaxSpeed(1000);
 }

 void loop() {
   // Set the current position to 0:
   stepper.setCurrentPosition(0);

   // Run the motor forward at 200 steps/second until the motor reaches 400 steps (2 revolutions):
   while (stepper.currentPosition() != 400)  {
     stepper.setSpeed(200);
     stepper.runSpeed();
   }

   delay(1000);

   // Reset the position to 0:
   stepper.setCurrentPosition(0);

   // Run the motor backwards at 600 steps/second until the motor reaches -200 steps (1 revolution):
   while (stepper.currentPosition() != -200)   {
     stepper.setSpeed(-600);
     stepper.runSpeed();
   }

   delay(1000);

   // Reset the position to 0:
   stepper.setCurrentPosition(0);

   // Run the motor forward at 400 steps/second until the motor reaches 600 steps (3 revolutions):
   while (stepper.currentPosition() != 600)  {
     stepper.setSpeed(400);
```

```
    stepper.runSpeed();
  }

  delay(3000);
}
```

Code explanation:

The first part of the code up to the loop() section is exactly the same as in the previous example.

In the loop I make use of a while loop (https://www.arduino.cc/reference/en/language/structure/control-structure/while/) in combination with the `currentPosition()` function. First, I set the current position of the stepper motor to zero with `stepper.setCurrentPosition(0)`.

```
  // Set the current position to 0:
  stepper.setCurrentPosition(0);
```

Next we make use of the while loop. A while loop will loop continuously, and infinitely, until the expression inside the parenthesis, () becomes false. So in this case I check if the current position of the stepper motor is not equal to 200 steps (!= means: is not equal to). While this is not the case, we run the stepper motor at a constant speed as set by `setSpeed()`.

```
  // Run the motor forward at 200 steps/second until the motor reaches 400 steps (2 revolutions):
  while (stepper.currentPosition() != 400)  {
    stepper.setSpeed(200);
    stepper.runSpeed();
  }
```

In the rest of the loop we do exactly the same, just with a different speed and target position.

---

3. Acceleration and deceleration example code

In this example we will look at one of the main reasons to use the AccelStepper library.

With the following sketch you can add acceleration and deceleration to the movements of the stepper motor without any complicated coding. The first section of this sketch is the same as in example 1, but the setup and the loop are different.

The motor will run five revolutions back and forth with a speed of 200 steps per second and an acceleration of 50 steps/second^2.

```
 /* Example sketch to control a stepper motor with L298N motor driver, Arduino UNO and AccelStepper.h library. Acceleration and deceleration. More info: https://www.makerguides.com */

// Include the AccelStepper library:
#include <AccelStepper.h>

// Define the AccelStepper interface type:
#define MotorInterfaceType 4

// Create a new instance of the AccelStepper class:
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 9, 10, 11);

void setup() {
  // Set the maximum steps per second:
  stepper.setMaxSpeed(200);

  // Set the maximum acceleration in steps per second^2:
  stepper.setAcceleration(50);
}

void loop() {
  // Set target position:
  stepper.moveTo(1000);
  // Run to position with set speed and acceleration:
  stepper.runToPosition();

  delay(1000);

  // Move back to original position:
  stepper.moveTo(0);
  // Run to position with set speed and acceleration:
  stepper.runToPosition();

  delay(1000);
}
```

How the code works:

In the `setup()`, besides the maximum speed, we need to define the acceleration/deceleration. For this, we use the function `setAcceleration()`.

```
    // Set the maximum steps per second:
    stepper.setMaxSpeed(200);

    // Set the maximum acceleration in steps per second^2:
    stepper.setAcceleration(50);
```

In the loop section of the code, I used a different way to let the motor rotate a predefined number of steps. First I set the target position with the function `moveTo()`. Next, we simply use the function `runToPosition()` to let the motor run to the target position with the set speed and acceleration. The motor will decelerate before reaching the target position.

```
    // Set target position:
    stepper.moveTo(1000);
    // Run to position with set speed and acceleration:
    stepper.runToPosition();
```

Finally, we set the new target position back to the 0, so that we return to the origin.

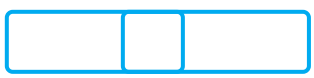
Conclusion

In this article I have shown you how you can control a stepper motor with an L298N motor driver. We have looked at 4 examples, using both the Stepper and AccelStepper libraries. I hope you found it useful and informative. If you did, **please share it with a friend that also likes electronics!**

I would love to know what projects you plan on building (or have already built) with the L298N motor driver. 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.

Beginner


What to read next?

LM35 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/lm35-arduino-tutorial/)

TMP36 analog temperature sensor with Arduino tutorial (https://www.makerguides.com/tmp36-arduino-tutorial/)

Arduino Nano Board Guide (Pinout, Specifications, Comparison) (https://www.makerguides.com/arduino-nano/)

The complete guide for DS18B20 digital temperature sensors with Arduino (https://www.makerguides.com/ds18b20-arduino-tutorial/)

How to use an IR receiver and remote with Arduino (https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/)

Comments

Norm Peacey says
December 1, 2020 at 2:11 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-5076)

I am making remote control roller blinds and have created a blend of the Accel Acceleration and Multistepper sketches.
This opens and closes the blinds successfully.
However, I want to release the stepper coils at the end of each function in order to allow the blinds to be adjusted manually if needed.
I have been looking unsuccessfully for this code.
Can you offer a suggestion?
I plan to operate the arduino with an IR remote control.

Reply

Benne de Bakker says
December 1, 2020 at 2:19 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-5078)

Hi Norm,

Cool project! I think you should be able to use the disableOutputs() and enableOutputs() functions of the AccelStepper library. You can find more information about these functions here: https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html#a3591e29a236e2935afd7f64ff6c22006 (https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html#a3591e29a236e2935afd7f64ff6c22006)

I haven't used these functions yet, but I think you can just call myStepper.disableOutputs() at the end of your functions and myStepper.enableOutputs() at the beginning.

Best regards,

Benne

Reply

Michel says
April 19, 2020 at 7:40 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-2496)

Thank you so much for this valuable course.
This help me to use the L298 module and a dvd head motor

Reply

Jacobus van Rooyen says
January 13, 2020 at 12:24 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-1574)

Good day Thank you for lots of good an interesting information
What I would like to know can I use 2 L298N drivers to control two stepper motors from the same Arduino uno board
Not sure about were the second board wiring has to go?
Please excuse me for asking stupid questions
Hope you can give me some advice I am very new to this

Reply

Trackbacks

Stepper Motor + Arduino Motor Shield Rev3 Tutorial (4 Examples) (https://www.makerguides.com/arduino-motor-shield-stepper-motor-tutorial/) says:
September 9, 2019 at 1:10 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-526)
[...] Control a stepper motor with L298N motor driver and Arduino [...]

28BYJ-48 Stepper Motor with ULN2003 and Arduino (4 Examples) (https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/) says:
September 8, 2019 at 5:50 pm (https://www.makerguides.com/l298n-stepper-motor-arduino-tutorial/#comment-519)
[...] Control a stepper motor with L298N motor driver and Arduino [...]

(https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)

How to control a character I2C LCD with Arduino (https://www.makerguides.com/character-i2c-lcd-arduino-tutorial/)

(https://www.makerguides.com/tm1637-arduino-tutorial/)

TM1637 4-digit 7-segment LED display Arduino tutorial (https://www.makerguides.com/tm1637-arduino-tutorial/)

(https://www.makerguides.com/character-lcd-arduino-tutorial/)

How to use a 16×2 character LCD with Arduino (https://www.makerguides.com/character-lcd-arduino-tutorial/)