

## 28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial

[27 Comments](#) / [Arduino](#), [Motor Controls](#), [Tutorials](#)

This article includes everything you need to know about controlling a [28BYJ-48 stepper motor](#) with the ULN2003 driver board and Arduino. I have included datasheets, a wiring diagram, and many example codes!

First we take a look at the easy to use Arduino **Stepper library**. This library is great when you are just starting out, but doesn't have many extra features.

An advertisement for PCBWay featuring a green background with a grid of dots. On the right, there is a photograph of a complex printed circuit board (PCB) with various electronic components. On the left, the text is arranged as follows: the PCBWay logo in green and orange; two price points, 'Only \$5' and 'Only \$30', in large pink font, with 'for 10 PCBs' and 'for PCB Assembly' in smaller white text below them; a bullet point stating 'High-Quality PCB Production & Assembly With the most options to customize your products.'; and a white button with the text 'Order Now' in green.

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 want to learn more about controlling larger stepper motors with more torque and more speed, take a look at the articles below. In these articles I teach you how to control NEMA 17 stepper motors, with drivers like the A4988.

Other stepper motor tutorials:



- [Control a stepper motor with L298N motor driver and Arduino](#)
- [How to control a Stepper Motor with Arduino Motor Shield Rev3](#)
- [How to control a stepper motor with A4988 driver and Arduino](#)
- [How to control a stepper motor with DRV8825 driver and Arduino](#)

If you have any questions, please leave a comment below.


---

# Supplies

## Hardware components

	28BYJ-48 stepper motor	× 1	<a href="#">Amazon</a>
	ULN2003 driver board	× 1	<a href="#">Amazon</a>
	Arduino Uno Rev3	× 1	<a href="#">Amazon</a>
	Jumper wires (male to female)	× 10	<a href="#">Amazon</a>
	Breadboard (optional, makes wiring easier)	× 1	<a href="#">Amazon</a>
	USB cable type A/B	× 1	<a href="#">Amazon</a>
	5V power supply (powering the stepper motor directly from the Arduino can damage it!)	× 1	<a href="#">Amazon</a>

## Software

	Arduino IDE		
---	-------------	--	--

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.

---

## Information about the 28BYJ-48 stepper motor and ULN2003 driver board

The 28BYJ-48 is [one of the cheapest stepper motors you can find](#). Although it is not super accurate or powerful, it is a great motor to use for smaller projects or if you just want to learn about stepper motors.

This motor is often used to automatically adjust the vanes of an air conditioner unit. It has a built-in gearbox, which gives it some extra torque and reduces the speed drastically.

Below you can find the specifications for both the stepper motor and driver that are used in this tutorial.

## 28BYJ-48 Stepper Motor Specifications

Rated voltage	5 V
Coil Resistance	50 Ohms
Coil Type	Unipolar
Diameter – shaft	0.197" (5.00 mm)
Length – shaft and bearing	0.394" (10 mm)
Features	Flatted shaft
Size/dimension	Round – 1.100" dia (28.00 mm)
Mounting hole spacing	Flatted Shaft
Gear reduction	1/64 ( <b>see note</b> )
Step angle	Half step mode (recommended): 0.0879° Full step mode: 0.176°
Steps per revolution	Half step mode: 4096 ( <b>see note</b> ) Full step mode: 2048
Termination style	Wire leads with connector
Motor type	Permanent Magnet Gear Motor

Number of phases	4
Cost	<a href="#">Check price</a>

---

For more information you can check out the datasheet [here](#).

[28BYJ-48 Datasheet](#)

**Important note:** Manufacturers usually specify that the motors have a 64:1 gear reduction. Some members of the [Arduino Forums](#) noticed that this wasn't correct and so they took some motors apart to check the actual gear ratio. They determined that the exact gear ratio is in fact **63.68395:1**, which results in approximately **4076 steps per full revolution** (in half step mode).

I am not sure if all manufacturers use the exact same gearbox, but you can just adjust the steps per revolution in the code, to match your model.

The [Adafruit Industries Small Reduction Stepper Motor](#) uses the same form factor as the 28BYJ-48, but does have a different gear ratio. It has a roughly 1/16 reduction gear set, which results in 513 steps per revolution (in full-step mode). You can download the datasheet for it [here](#).

For more information about the driver you can check out the datasheet [below](#).

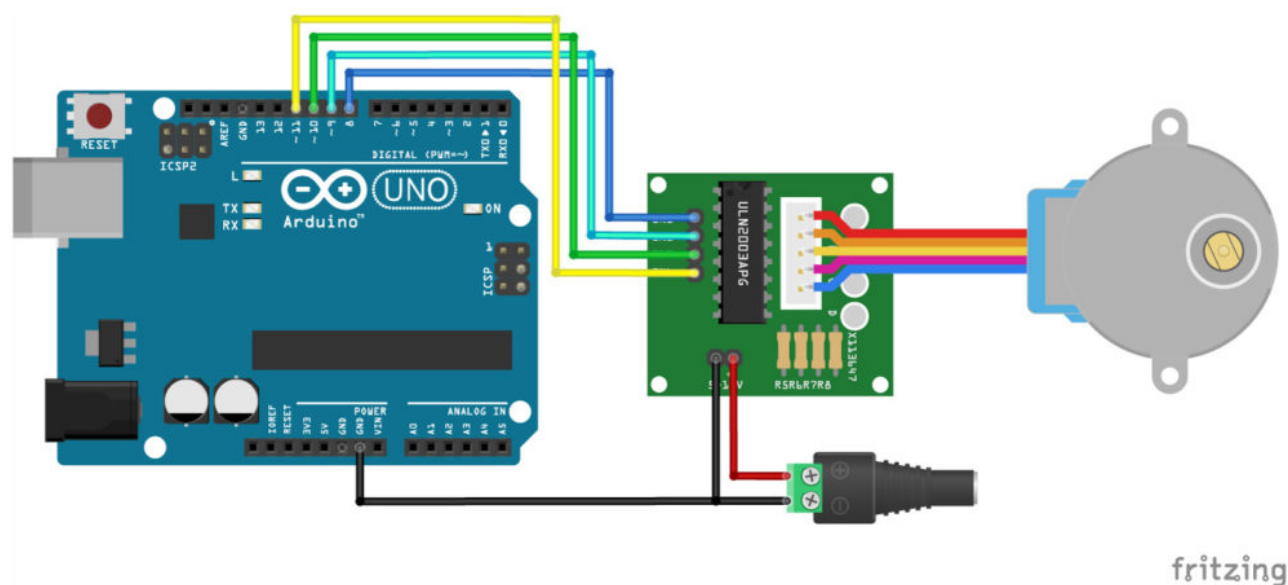
[ULN2003 Datasheet](#)

[ULN2003 Driver PCB](#)

---

## Wiring – Connecting 28BYJ-48 stepper motor and ULN2003 driver board to Arduino UNO

The wiring diagram/schematic below shows you how to connect the ULN2003 driver board to the 28BYJ-48 stepper motor and the Arduino. The connections are also given in the table below.



Wiring diagram for ULN2003 driver with 28BYJ-48 stepper motor and Arduino.

I used a breadboard and some jumper wires to connect the driver board to an external power supply.

ULN2003 and 28BYJ-48 to Arduino Connections

ULN2003 Driver Board	Connection
IN1	Pin 8 Arduino
IN2	Pin 9 Arduino
IN3	Pin 10 Arduino
IN4	Pin 11 Arduino
—	Logic GND Arduino
—	GND power supply
+	5 V power supply

---

**Please note:** It is possible to directly power the stepper motor from the 5 V output of the Arduino. This however, is not recommended. When the stepper motor draws too much current you can **damage the Arduino**. I also found that when powering the Arduino with USB power only, I would get inconsistent behavior and bad performance of the stepper motor.

I recommend to power the driver board/stepper motor with [an external 5 V power supply, like this one](#). It should come with a female DC connector, so you can easily connect it to some (jumper) wires. Note that you also need to connect the GND of the Arduino to the – pin on the ULN2003 driver board.

After uploading the code **you also need to power the Arduino**, either with a USB type-B cable or via the 5.5 mm power jack.

The jumper next to power connections on the driver board can be used to disconnect power to the stepper motor.

---

## Basic Arduino example code to control a 28BYJ-48 stepper motor

You can upload the following example code to your Arduino using the [Arduino IDE](#).

This example uses the **Stepper.h library**, which should come pre-installed with the Arduino IDE. This sketch turns the stepper motor 1 revolution in one direction, pauses, and then turns 1 revolution in the other direction.

```
1.  /* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board
    2.  and Arduino UNO. More info: https://www.makerguides.com */
    3.
    4.  // Include the Arduino Stepper.h library:
    5.  #include <Stepper.h>
    6.
    7.  // Define number of steps per rotation:
```

```

7.  const int stepsPerRevolution = 2048;
8.
9.  // Wiring:
10. // Pin 8 to IN1 on the ULN2003 driver
11. // Pin 9 to IN2 on the ULN2003 driver
12. // Pin 10 to IN3 on the ULN2003 driver
13. // Pin 11 to IN4 on the ULN2003 driver
14.
15. // Create stepper object called 'myStepper', note the pin order:
16. Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
17.
18. void setup() {
19.     // Set the speed to 5 rpm:
20.     myStepper.setSpeed(5);
21.
22.     // Begin Serial communication at a baud rate of 9600:
23.     Serial.begin(9600);
24. }
25.
26. void loop() {
27.     // Step one revolution in one direction:
28.     Serial.println("clockwise");
29.     myStepper.step(stepsPerRevolution);
30.     delay(500);
31.
32.     // Step one revolution in the other direction:
33.     Serial.println("counterclockwise");
34.     myStepper.step(-stepsPerRevolution);
35.     delay(500);
36. }

```

## Code explanation:

The sketch starts by including the Stepper.h Arduino library. More information about this library can be found on the [Arduino website](#).

```

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

```

Next, I defined how many steps the motor takes to rotate 1 revolution. In this example we will be using the motor in **full-step mode**. This means it **takes 2048 steps to rotate 360 degrees** (see motor specifications above).

```

6.  // Define number of steps per rotation:
7.  const int stepsPerRevolution = 2048;

```

Next, 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 to which the motor is



connected. To get the correct step sequence, we need to set the pins in the following order: 8, 10, 9, 11.

```
15. // Create stepper object called 'myStepper', note the pin order:
16. Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 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, 10, 9, 11);`. You can create multiple stepper motor objects with different names and pins. This allows you to easily control 2 or more stepper motors at the same time.

In the setup, you can set the speed in rpm with the function `setSpeed(rpm)`.

The **maximum speed** for a 28byj-48 stepper motor is roughly **10-15 rpm at 5 V**.

```
18. void setup() {
19.     // Set the speed to 5 rpm:
20.     myStepper.setSpeed(5);
21.
22.     // Begin Serial communication at a baud rate of 9600:
23.     Serial.begin(9600);
24. }
```

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.

```
26. void loop() {
27.     // Step one revolution in one direction:
28.     Serial.println("clockwise");
29.     myStepper.step(stepsPerRevolution);
30.     delay(500);
31.
32.     // Step one revolution in the other direction:
33.     Serial.println("counterclockwise");
34.     myStepper.step(-stepsPerRevolution);
35.     delay(500);
36. }
```

---

## Example codes for 28BYJ-48 stepper motor with Arduino and 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 these examples I will be using the **AccelStepper library**.

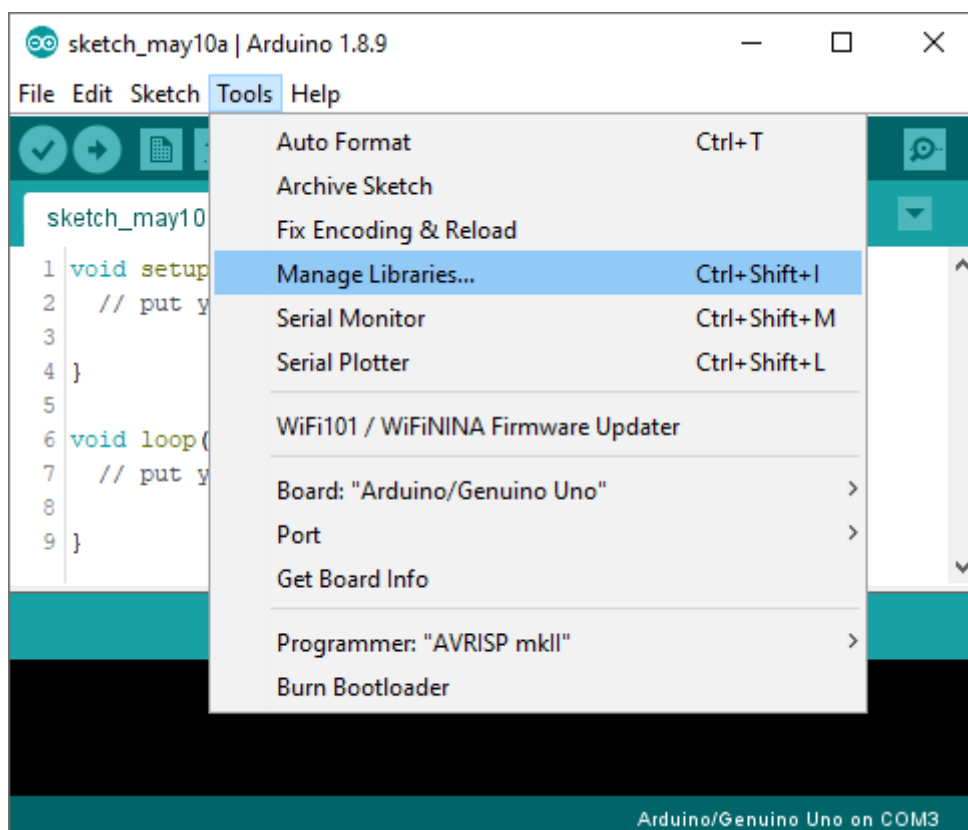
The AccelStepper library written by Mike McCauley is an awesome library to use for your project. One of the advantages is that it supports acceleration and deceleration, but it has a lot of other nice functions too.

You can download the latest version of this library [here](#) or click the button below.

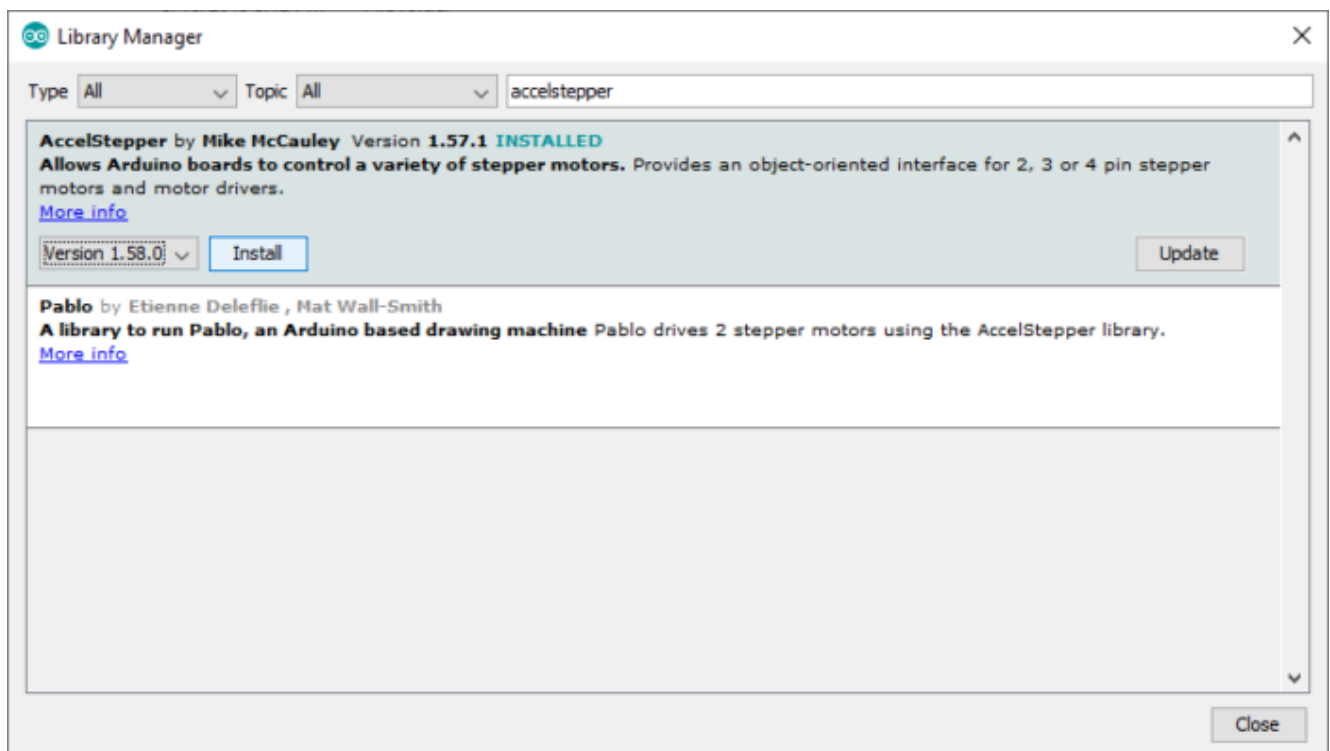
AccelStepper-1.59.zip

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 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).

You can copy the code by clicking on the button in the top right corner of the code field.

```
1.  /* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board,
2.   AccelStepper and Arduino UNO: continuous rotation. More info:
3.   https://www.makerguides.com */
4.
5.   // Include the AccelStepper library:
6.   #include <AccelStepper.h>
7.
8.   // Motor pin definitions:
9.   #define motorPin1  8      // IN1 on the ULN2003 driver
10.  #define motorPin2  9      // IN2 on the ULN2003 driver
11.  #define motorPin3  10     // IN3 on the ULN2003 driver
12.  #define motorPin4  11     // IN4 on the ULN2003 driver
13.
14.  // Define the AccelStepper interface type; 4 wire motor in half step mode:
15.  #define MotorInterfaceType 8
16.
17.  // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper library
18.  with 28BYJ-48 stepper motor:
```

```

16. AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3,
    motorPin2, motorPin4);
17.
18. void setup() {
19.     // Set the maximum steps per second:
20.     stepper.setMaxSpeed(1000);
21. }
22.
23. void loop() {
24.     // Set the speed of the motor in steps per second:
25.     stepper.setSpeed(500);
26.     // Step the motor with constant speed as set by setSpeed():
27.     stepper.runSpeed();
28. }

```

## How the code works:

Again 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 ULN2003 to Arduino connections.

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 `motorPin1`, the compiler will replace it with the value 8 when the program is compiled.

```

6. // Motor pin definitions:
7. #define motorPin1 8 // IN1 on the ULN2003 driver
8. #define motorPin2 9 // IN2 on the ULN2003 driver
9. #define motorPin3 10 // IN3 on the ULN2003 driver
10. #define motorPin4 11 // IN4 on the ULN2003 driver

```

The next step is to specify the motor interface type for the AccelStepper library. In this case we will be driving a 4 wire stepper motor in **half step mode**, so we set the interface type to '8'. You can find the other interface types [here](#). If you want to run the motor in full-step mode (fewer steps per revolution), just change the 8 to 4.

```

12. // Define the AccelStepper interface type; 4 wire motor in half step mode:
13. #define MotorInterfaceType 8

```

Next, you need to create a new instance of the AccelStepper class with the appropriate motor interface type and connections. To get the correct step sequence, we need to set the pins in the following order: **motorPin1, motorPin3, motorPin2, motorPin4**.

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, motorPin1, motorPin3, motorPin2, motorPin4);`. You can create multiple stepper motor objects with different names and pins. This allows you to easily control 2 or more stepper motors at the same time.

```
15. // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper library
    with 28BYJ-48 stepper motor:
16. AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3,
    motorPin2, motorPin4);
```

In the setup section of the code, we define the maximum speed in steps/second with the function `setMaxSpeed()`. 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: `stepper2.setMaxSpeed(500);`.

```
18. void setup() {
19.     // Set the maximum steps per second:
20.     stepper.setMaxSpeed(1000);
21. }
```

In the loop, we first set the speed that we want the motor to run at with 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 it 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.

```
23. void loop() {
24.     // Set the speed of the motor in steps per second:
```

```
25.     stepper.setSpeed(500);
26.     // Step the motor with constant speed as set by setSpeed():
27.     stepper.runSpeed();
28. }
```

In half step mode, one revolution takes 4096 steps, so 500 steps/sec results in roughly **7 rpm**.

---

## 2. Sketch 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 1 revolution clockwise with 500 steps/sec, then turns 1 revolution counterclockwise at 1000 steps/sec, and lastly turns 2 revolutions clockwise at 1000 steps/sec.

```
1.  /* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board,
2.     AccelStepper and Arduino UNO: number of steps/revolutions. More info:
3.     https://www.makerguides.com */
4.
5.
6.     // Include the AccelStepper library:
7.     #include <AccelStepper.h>
8.
9.     // Motor pin definitions:
10.    #define motorPin1  8      // IN1 on the ULN2003 driver
11.    #define motorPin2  9      // IN2 on the ULN2003 driver
12.    #define motorPin3 10      // IN3 on the ULN2003 driver
13.    #define motorPin4 11      // IN4 on the ULN2003 driver
14.
15.    // Define the AccelStepper interface type; 4 wire motor in half step mode:
16.    #define MotorInterfaceType 8
17.
18.    // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper library
19.    // with 28BYJ-48 stepper motor:
20.    AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3,
21.    motorPin2, motorPin4);
22.
23.    void setup() {
24.        // Set the maximum steps per second:
25.        stepper.setMaxSpeed(1000);
26.    }
27.
28.    void loop() {
29.        // Set the current position to 0:
30.        stepper.setCurrentPosition(0);
31.
32.        // Run the motor forward at 500 steps/second until the motor reaches 4096 steps
33.        // (1 revolution):
34.        while (stepper.currentPosition() != 4096) {
35.            stepper.setSpeed(500);
36.            stepper.runSpeed();
37.        }
38.
39.        // Run the motor backward at 1000 steps/second until the motor reaches -4096 steps
40.        // (-1 revolution):
41.        while (stepper.currentPosition() != -4096) {
42.            stepper.setSpeed(-1000);
43.            stepper.runSpeed();
44.        }
45.
46.        // Run the motor forward at 1000 steps/second until the motor reaches 8192 steps
47.        // (2 revolutions):
48.        while (stepper.currentPosition() != 8192) {
49.            stepper.setSpeed(1000);
50.            stepper.runSpeed();
51.        }
52.    }
```

```

31.     }
32.     delay(1000);
33.
34.     // Reset the position to 0:
35.     stepper.setCurrentPosition(0);
36.
37.     // Run the motor backwards at 1000 steps/second until the motor reaches -4096
steps (1 revolution):
38.     while (stepper.currentPosition() != -4096) {
39.         stepper.setSpeed(-1000);
40.         stepper.runSpeed();
41.     }
42.
43.     delay(1000);
44.
45.     // Reset the position to 0:
46.     stepper.setCurrentPosition(0);
47.     // Run the motor forward at 1000 steps/second until the motor reaches 8192 steps
(2 revolutions):
48.     while (stepper.currentPosition() != 8192) {
49.         stepper.setSpeed(1000);
50.         stepper.runSpeed();
51.     }
52.
53.     delay(3000);
54. }

```

## 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](#) in combination with the `currentPosition()` function. First, I set the current position of the stepper motor to zero with `stepper.setCurrentPosition(0)`.

```

24.     // Set the current position to 0:
25.     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 4096 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()`.

```

27.     // Run the motor forward at 500 steps/second until the motor reaches 4096 steps
(1 revolution):
28.     while (stepper.currentPosition() != 4096) {
29.         stepper.setSpeed(500);
30.         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

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 two revolutions back and forth with a speed of 1000 steps per second and an acceleration of 200 steps/second<sup>2</sup>.

```

1.  /* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board,
    AccelStepper and Arduino UNO: acceleration and deceleration. More info:
    https://www.makerguides.com */

2.
3.  // Include the AccelStepper library:
4.  #include <AccelStepper.h>
5.
6.  // Motor pin definitions:
7.  #define motorPin1  8      // IN1 on the ULN2003 driver
8.  #define motorPin2  9      // IN2 on the ULN2003 driver
9.  #define motorPin3 10      // IN3 on the ULN2003 driver
10. #define motorPin4 11      // IN4 on the ULN2003 driver
11.
12. // Define the AccelStepper interface type; 4 wire motor in half step mode:
13. #define MotorInterfaceType 8
14.
15. // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper library
    with 28BYJ-48 stepper motor:
16. AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1, motorPin3,
    motorPin2, motorPin4);
17.
18. void setup() {
19.     // Set the maximum steps per second:
20.     stepper.setMaxSpeed(1000);
21.     // Set the maximum acceleration in steps per second^2:
22.     stepper.setAcceleration(200);
23. }
24.
25. void loop() {
26.     // Set target position:
27.     stepper.moveTo(8192);
28.     // Run to position with set speed and acceleration:
29.     stepper.runToPosition();
30.
31.     delay(1000);
32.

```



```
33. // Move back to original position:
34. stepper.moveTo(0);
35. // Run to position with set speed and acceleration:
36. stepper.runToPosition();
37.
38. delay(1000);
39. }
```

## How the code works:

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

```
18. void setup() {
19.     // Set the maximum steps per second:
20.     stepper.setMaxSpeed(1000);
21.     // Set the maximum acceleration in steps per second^2:
22.     stepper.setAcceleration(200);
23. }
```

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.

```
25. void loop() {
26.     // Set target position:
27.     stepper.moveTo(8192);
28.     // Run to position with set speed and acceleration:
29.     stepper.runToPosition();
30.
31.     delay(1000);
32.
33.     // Move back to original position:
34.     stepper.moveTo(0);
35.     // Run to position with set speed and acceleration:
36.     stepper.runToPosition();
37.
38.     delay(1000);
39. }
```

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 28BYJ-48 stepper motor with a ULN2003 driver and Arduino. 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 this stepper motor. 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.

[← Previous Post](#)

[Next Post →](#)

## 27 thoughts on “28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial”

**R4Y**

MARCH 30, 2021 AT 12:40 PM

Hi, it kinda works but the stepper motor doesnt rotate. ot vibrates littlebit tho.  
how can i fix this?

**R4Y**

MARCH 30, 2021 AT 12:44 PM

im using arduino mega 2560 BTW

**DAVE PRESTON**

NOVEMBER 17, 2020 AT 1:52 PM

I use this configuration with 1 driver/motor to rotate my Sony camera on a tripod mounted bracket. I use another driver/motor to tilt the bracket via a toothed belt and pulleys. This reduces the torque on the motor and gives me very smooth movement.

These are connected to a UNO3 board and I wrote an Android app to remotely control them via Bluetooth, and it all works well BUT with the lens at full zoom the motor isn't quite powerful enough to tilt the bracket up, so I sometimes have to zoom out a bit.

Would it be possible/advisable to connect another motor to the driver that controls the tilt? I'm hoping this would give me twice the torque? If feasible is it just a case of splicing the cables or is there a neater way?

**BENNE DE BAKKER**

NOVEMBER 21, 2020 AT 9:16 AM

Hi Dave,

Sounds like an awesome project! I wouldn't recommend connecting two stepper motors to the same driver.

You could simply add another 28BYJ-48 stepper motor and ULN2003 driver or upgrade to a geared-down NEMA 17 stepper motor with an A4988 or DRV8825.

Note that you can also use the analog pins of the Arduino as digital outputs if you do not have enough pins available.

Benne

**MAX**

OCTOBER 26, 2020 AT 3:18 AM

Can you explain or provide any information on how you set up the 5V power supply. Ultimately I'd like to plug in one cable and power both the board and the motor, but I'll first start with just the power supply's wiring.

**JJFF**

OCTOBER 11, 2020 AT 9:14 AM

One of the guides on the 28BYJ-48 I could read till now. And I'm following this since some times. Wished I could have started with this documentation. ????

Important links (to consult the background information) are present, mainly topics nicely described.

**BENNE DE BAKKER**

OCTOBER 11, 2020 AT 12:11 PM

Thanks!

**KEITH L**

AUGUST 11, 2020 AT 6:21 PM

My stepper will not run at a speed of 1000. I set the speed to 800 and it works fine. I will try increasing it incrementally beyond 800 to see what it's maximum speed is.

**KEITH L**

AUGUST 11, 2020 AT 6:53 PM

It works at 1000 with acceleration. I guess it cannot start at 1000, but will run at that speed. Thanks for a great tutorial.

**CASPER VAN ENGELENBURG**

JULY 9, 2020 AT 3:13 PM

Hey,

About the motordriver: is it possible to increase the amount of micro-steps up to say 1/16th of a full-step? The Allegro A4988 works for example fine up

to such small steps but I am not sure if they are easily compatible with these amazing stepper motors.

Thanks,  
Casper

**BENNE DE BAKKER**

JULY 9, 2020 AT 3:35 PM

Hi Casper,

The 28BYJ-48 is a unipolar stepper motor and is not directly compatible with the Allegro A4988 which is designed for bipolar motors. It is possible to modify the 28byj-48 to turn it into a bipolar motor, but I don't have experience with that myself.

You can find some sources about this conversion online, for example:  
<https://coeleveld.com/wp-content/uploads/2016/10/Modifying-a-28BYJ-48-step-motor-from-unipolar-to-bipolar.pdf>

Additionally, I don't think microstepping for the 28byj-48 stepper motors is very useful. These stepper motors are already geared down a lot, so each step of the motor results in a very small rotation of the output shaft. Due to the gear reduction, there is quite a lot of backlash. This means that you will never get very high precision out of these motors, even with microstepping.

Best regards,

Benne

JOHN

JULY 5, 2020 AT 8:32 AM

@Ben Draper

try

D8 -> In1 (connects to blue)

D9 -> In3 (yellow)

D10 -> In2 (pink)

D11 -> In4 (orange)

Stepper.h gives this table:

\* Step C0 C1 C2 C3

\* 1 1 0 1 0

\* 2 0 1 1 0

\* 3 0 1 0 1

\* 4 1 0 0 1

The Spec sheet for 28BYJ-48 gives pink and orange as one winding, and blue and yellow as the other. For (each) one winding the two ends should be driven anti-phase, that is, when one end is on, the other is off. From the table, you can see that C0 and C1 are driven anti-phase, so should be connected to the opposite ends of the same winding.

The wire order on my 28BYJ-48 plug is blue, pink, yellow, orange, red. This works out to blue -> In1, pink -> In2, yellow -> In3, orange -> In4.

HTH

**BEN DRAPER**

MAY 15, 2020 AT 9:27 AM

Great tutorial so thanks for all the effort that you've gone to on this.

I'm tearing my hair out with these stepper motors. Every library I use and every code example I use will only turn the motor one way. When it reaches the "reverse" section of the code, the motor does a small step in the reverse direction and then immediately back to the forward direction.

One of the suggestions that I've tried (and failed) is to reverse the middle two wires (<https://arduino.stackexchange.com/questions/57089/stepper-motor-wont-reverse-turn-ccw>) I've also tried the code on <https://www.instructables.com/id/BYJ48-Stepper-Motor/> to drive the motor with specific code, however this just makes the motor travel only in the reverse direction.

Speed settings in the AccelStepper library are 400 which is the highest it can get to without failing. I've tried slower speeds without any change in the result. Best result is from the example code in the above "2. Sketch to control number of steps or revolutions".

I've tried with 4 different controllers and motors and all produce the same result.

I've come to the conclusion that I've got a bad batch of these, but this is my last throw of the dice to see if anyone has had and fixed this problem before I move onto buying some better quality steppers!

Thanks in advance.



**ANDY**

JUNE 13, 2020 AT 9:43 AM

I have had a similar issue, but found the best solution was to change the steps out of sequence, so instead of using pins 8,9,10,11, try 8,10,9,11, that was the only way I could get it to reverse.

Andy

**MAX**

APRIL 12, 2020 AT 7:56 PM

Wow really nice !

Since a single motor occupies 4 pins, it seems difficult to have a lot of step motors like this right ?

Is it possible to 'extends' the number of pins of the arduino ? If yes what is the max number of step motors I can attach to arduino ? (I am a real starter at arduino stuff so any explanation or links to useful resource would be awesome)

Thanks !

**BENNE DE BAKKER**

MAY 22, 2020 AT 1:38 PM

Hi Max, you could use an I/O expander but I recommend using an Arduino Mega instead. The Mega has 54 digital pins, which should be enough for most projects.

**NIKOS**

JUNE 19, 2020 AT 1:39 PM

No need for Arduino Mega. Good old Shift Registers will do the trick. With 4 pins you can have as many motors as you like.

I made a video using this method but its in Greek. Not sure it will be of any assistance but here it is:

<https://youtu.be/tLNTftE25QE>

**THE BDOUILLEUR**

MARCH 26, 2020 AT 4:26 PM

Hi,

I have a similar ULN2003 driver, but it is wired differently:

- it has only one GND port
- it has a VM port instead
- it has a switch labeled VCC and VM

How should I connect it to the Arduino ?

Any help very welcome 😊

**MURAT A.**

FEBRUARY 9, 2020 AT 2:51 PM

Tnx a lot. I found it very useful for my DIY automatic fish feeder project.

**GWR**

NOVEMBER 15, 2019 AT 3:02 PM

Excellent. Just what I needed. Nice, clear detail. Good post, thank you. Now, I just need to find a reasonably rapid stepper ... off to the shed ...

**EDWARD AZABLE**

SEPTEMBER 21, 2019 AT 11:00 PM

While I can get one motor to spin I cannot get two to spin. can you help?

/\* Example sketch to control a 28BYJ-48 stepper motor with ULN2003 driver board, AccelStepper and Arduino UNO: continuous rotation. More info:

<https://www.makerguides.com> \*/

// Include the AccelStepper library:

#include

```
// Motor pin definitions:

#define motorPin1 8 // IN1 on the ULN2003 driver
#define motorPin2 9 // IN2 on the ULN2003 driver
#define motorPin3 10 // IN3 on the ULN2003 driver
#define motorPin4 11 // IN4 on the ULN2003 driver


#define motorPin1 3 // IN1 on the ULN2003 driver2
#define motorPin2 4 // IN2 on the ULN2003 driver2
#define motorPin3 5 // IN3 on the ULN2003 driver2
#define motorPin4 6 // IN4 on the ULN2003 driver2

// Define the AccelStepper interface type; 4 wire motor in half step mode:
#define MotorInterfaceType 8

// Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper
library with 28BYJ-48 stepper motor:
AccelStepper stepper = AccelStepper(MotorInterfaceType, motorPin1,
motorPin3, motorPin2, motorPin4);
AccelStepper stepper2 = AccelStepper(MotorInterfaceType, motorPin1,
motorPin3, motorPin2, motorPin4);
void setup() {
// Set the maximum steps per second:
stepper.setMaxSpeed(1000);
stepper2.setMaxSpeed(750);
}
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();
}
```

Hi Edward,

Thank you for your comment. It looks like you didn't specify different pin names for the second stepper motor. Both use motorPin1 etc. You don't necessarily have to define the pins first, you can also just put them in the AccelStepper function:

```
AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 10, 9, 11);  
AccelStepper stepper2 = AccelStepper(MotorInterfaceType, 3, 5, 4, 6);
```

I created and tested the following example:

<https://www.makerguides.com/wp-content/uploads/2019/09/28BYJ-48-2-Stepper-Motors.pdf>

```
1.  #include <AccelStepper.h>  
2.  
3.  // Connections:  
4.  // Stepper 1  
5.  // IN1 to pin 8  
6.  // IN2 to pin 9  
7.  // IN3 to pin 10  
8.  // IN4 to pin 11  
9.  // Stepper 2  
10. // IN1 to pin 3  
11. // IN2 to pin 4  
12. // IN3 to pin 5  
13. // IN4 to pin 6  
14.  
15. // Define the AccelStepper interface type; 4 wire motor in half step  
    mode:  
16. #define MotorInterfaceType 8  
17.  
18. // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the  
    AccelStepper library with 28BYJ-48 stepper motor:  
19. AccelStepper stepper = AccelStepper(MotorInterfaceType, 8, 10, 9, 11);  
20. AccelStepper stepper2 = AccelStepper(MotorInterfaceType, 3, 5, 4, 6);  
21.  
22. void setup() {  
23.     // Set the maximum steps per second:  
24.     stepper.setMaxSpeed(1000);  
25.     stepper2.setMaxSpeed(750);  
26. }
```

```
27.  
28. void loop() {  
29.     // Set the speed of the motor in steps per second:  
30.     stepper.setSpeed(500);  
31.     stepper2.setSpeed(250);  
32.     // Step the motor with constant speed as set by setSpeed():  
33.     stepper.runSpeed();  
34.     stepper2.runSpeed();  
35. }
```

Hope this helps,

Benne

**EDWARD AZABLE**

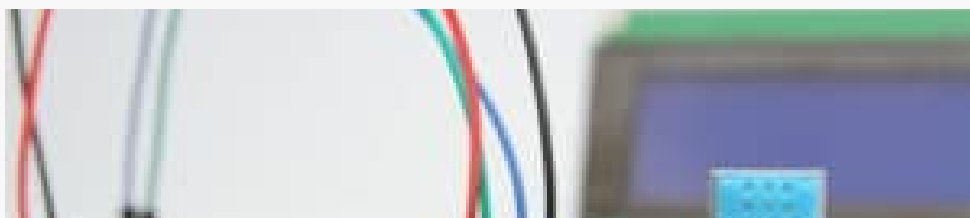
SEPTEMBER 22, 2019 AT 9:54 AM

Thank you for the swift reply. I have a hard time figuring out C. At least my modification of your code didn't throw any error messages at me so that is progress. 😊

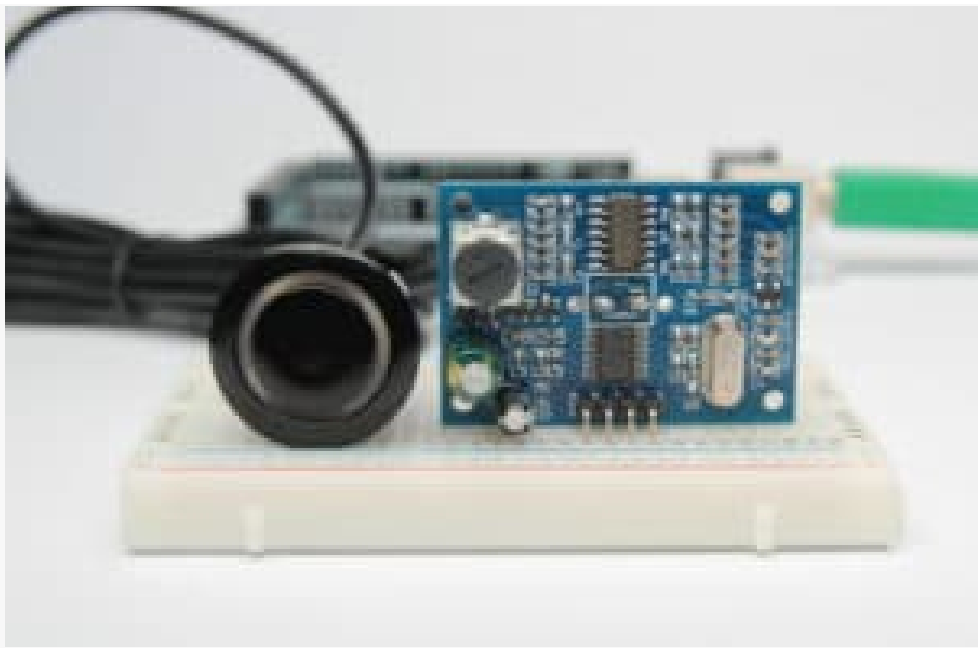
Comments are closed.



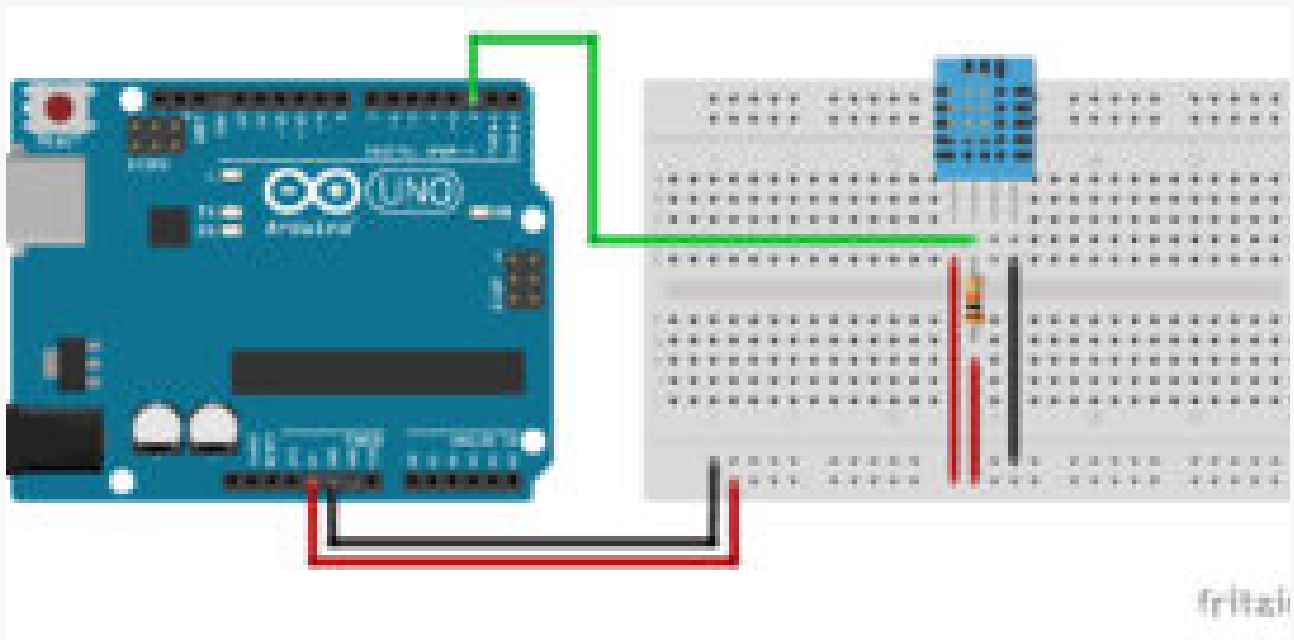
## Popular Posts

















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](#)