Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino

We are surrounded by stepper motors without even realizing it, as they are used in so many everyday items, including window blinds, 3D printers, DVD players, security cameras, and CNC machines. We’re a lot closer to stepper motors than you think.

Stepper motors fall somewhere between a conventional DC motor and a servo motor. They can rotate continuously like DC motors and be positioned precisely (in discrete steps) like servo motors.

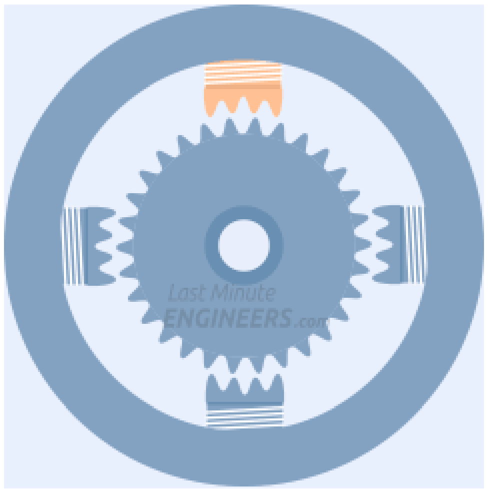
If you’re just getting started with stepper motors, the 28BYJ-48 is a great choice. They typically come with a ULN2003-based driver board, making them very simple to use.

Do you know how these stepper motors work?

Stepper motors use a cogged wheel and electromagnets to nudge the wheel round a ‘step’ at a time.

Each high pulse sent energizes the coil, attracting the teeth closest to the cogged wheel and rotating the motor in precise and fixed angle increments known as steps.

The number of steps that the stepper motor has in a 360 degree rotation is actually the number of teeth on the cog.



The way you pulse these coils determines how the motor operates.

- The sequence of pulses determines the spinning direction of the motor.
- The frequency of the pulses determines the speed of the motor.
- The number of pulses determines how far the motor will turn.

By energizing the coils in the correct sequence, the motor is rotated.

The 28BYJ-48 Stepper Motor

The 28BYJ-48 is a 5-wire unipolar stepper motor that runs on 5V. It’s perfect for projects that require precise positioning, like opening and closing a vent.

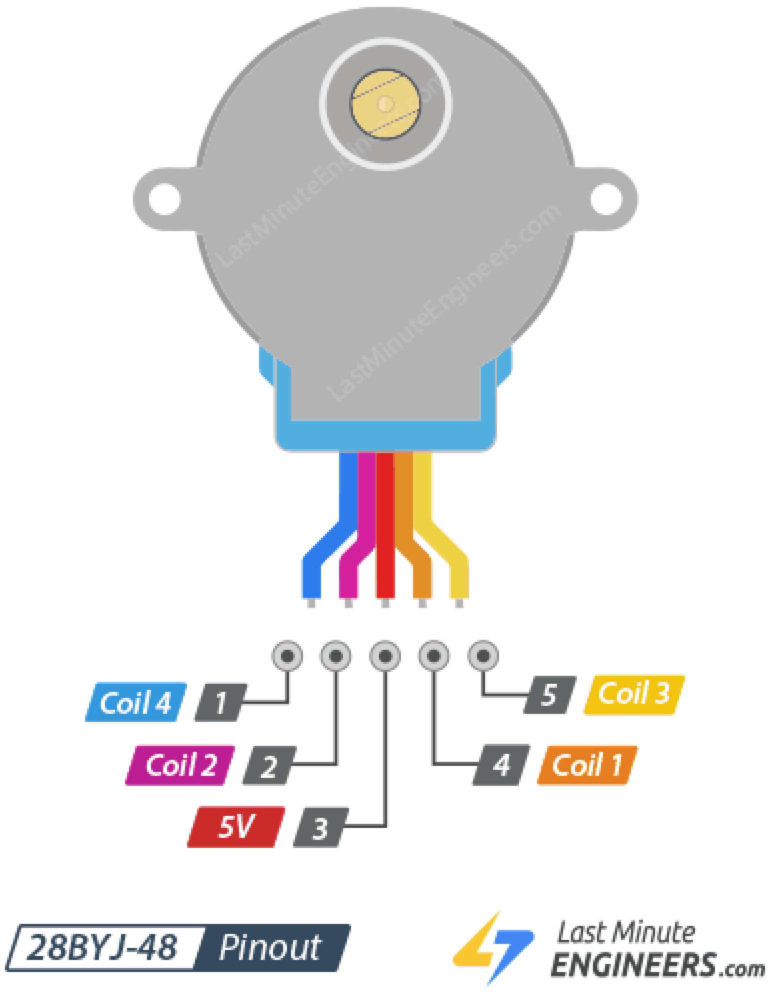
Because the motor does not use contact brushes, it has a relatively precise movement and is quite reliable.

Despite its small size, the motor delivers a decent torque of 34.3 mN.m at a speed of around 15 RPM. It provides good torque even at a standstill and maintains it as long as the motor receives power.

The only drawback is that it is somewhat power-hungry and consumes energy even when it is stationary.

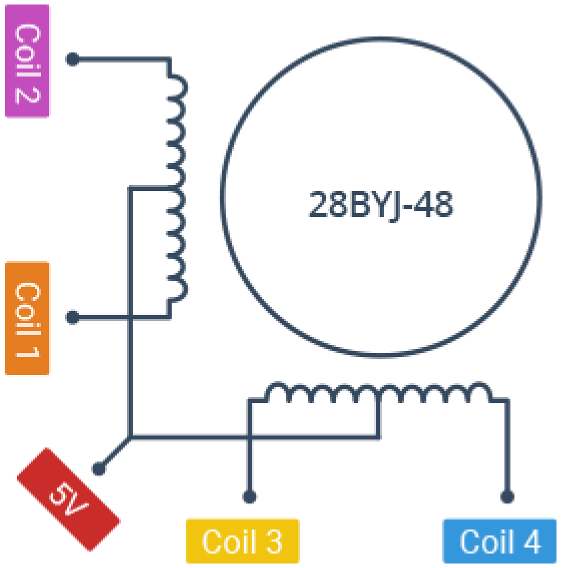
Pinout

The 28BYJ-48 stepper motor has five wires. The pinout is as follows:



The 28BYJ-48 has two coils, each of which has a center tap. These two center taps are connected internally and brought out as the 5th wire (red wire).

Together, one end of the coil and the center tap form a **Phase**. Thus, 28BYJ-48 has a total of four phases.

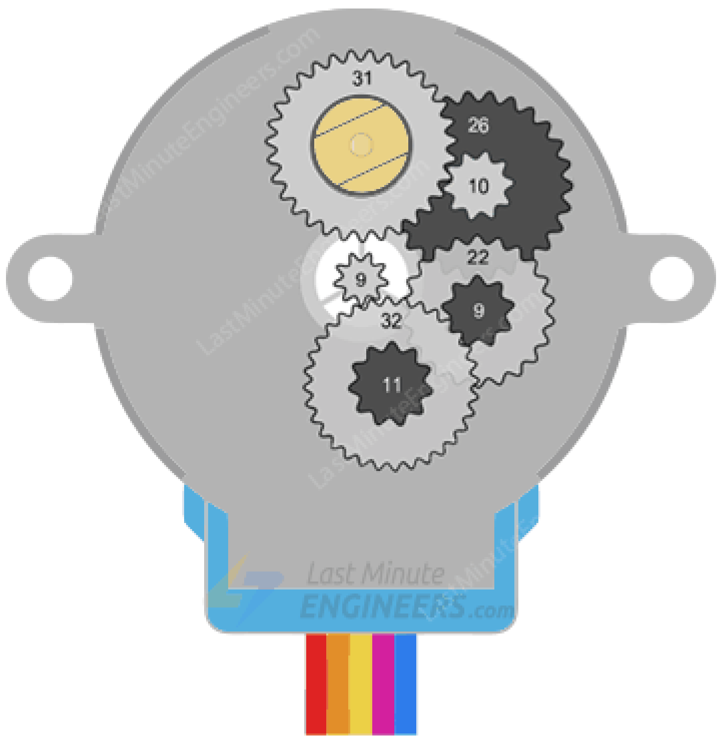


The red wire is always pulled HIGH, so when the other lead is pulled LOW, the phase is energized.

The stepper motor rotates only when the phases are energized in a logical sequence known as a **step sequence**.

Gear Reduction Ratio

According to the data sheet, when the 28BYJ-48 motor is operated in full-step mode, each step corresponds to a rotation of 11.25°. This means there are 32 steps per revolution (360°/11.25° = 32).



Gear Ratios:

- 32 / 9
- 22 / 11
- 26 / 9
- 31 / 10

Multiplying the gear ratios:

$$\frac{32}{9} \times \frac{22}{11} \times \frac{26}{9} \times \frac{31}{10} = 63.68395$$

Round 63.68395 up: 64

This gives us a 64:1 gear ratio over all

In addition, the motor features a 1/64 reduction gear set (actually, it is 1/63.68395, but 1/64 is a good enough approximation for most purposes).

This means that there are in fact 2038 steps (32*63.68395 steps per revolution = 825344 steps for 405 revolutions, approximately 2038 steps per revolution).

Power Consumption

The 28BYJ-48 typically draws about 240 mA.

Because the motor consumes a significant amount of power, it is preferable to power it directly from an external 5V power supply rather than from the Arduino.

It is worth noting that the motor consumes power even when it is at rest in order to maintain its position.

Technical Specifications

Here are the specifications:

Operating Voltage	5VDC
Operating Current	240mA (typical)
Number of phases	4
Gear Reduction Ratio	64:1 (25792: 405)
Step Angle	5.625° per half step
Frequency	100Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm

For more information, please refer to the datasheet below.

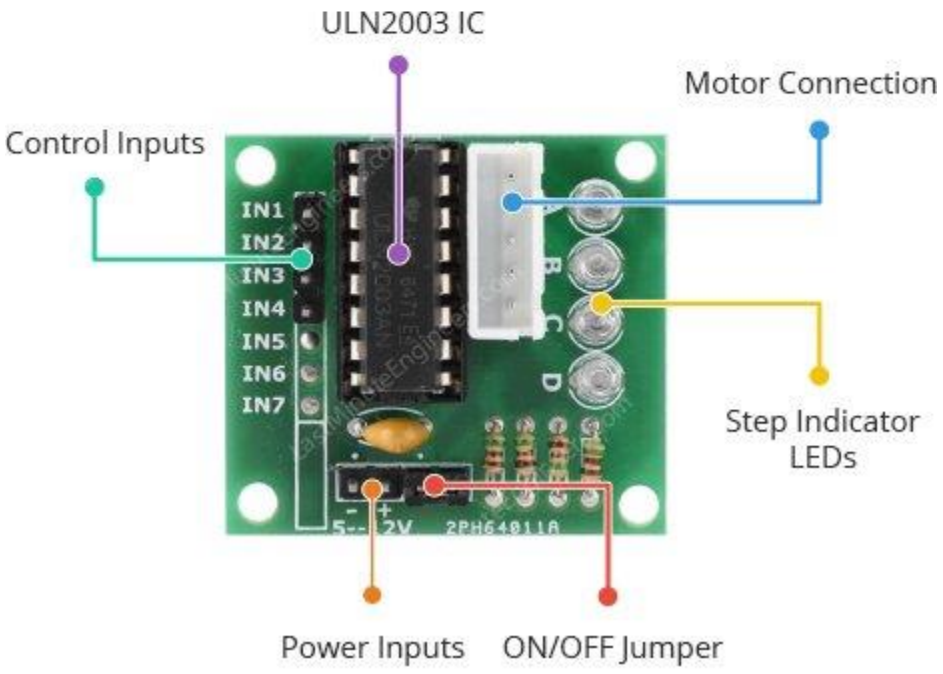
28BYJ-48 Datasheet

The ULN2003 Driver Board

Because the 28BYJ-48 stepper motor consumes a significant amount of power, it cannot be controlled directly by a microcontroller such as Arduino. To control the motor, a driver IC such as the ULN2003 is required; therefore, this motor typically comes with a ULN2003-based driver board.

The ULN2003, known for its high current and high voltage capability, provides a higher current gain than a single transistor and allows a microcontroller's low voltage low current output to drive a high current stepper motor.

The ULN2003 consists of an array of seven Darlington transistor pairs, each of which can drive a load of up to 500mA and 50V. This board utilizes four of the seven pairs.



The board has four control inputs and a power supply connection.

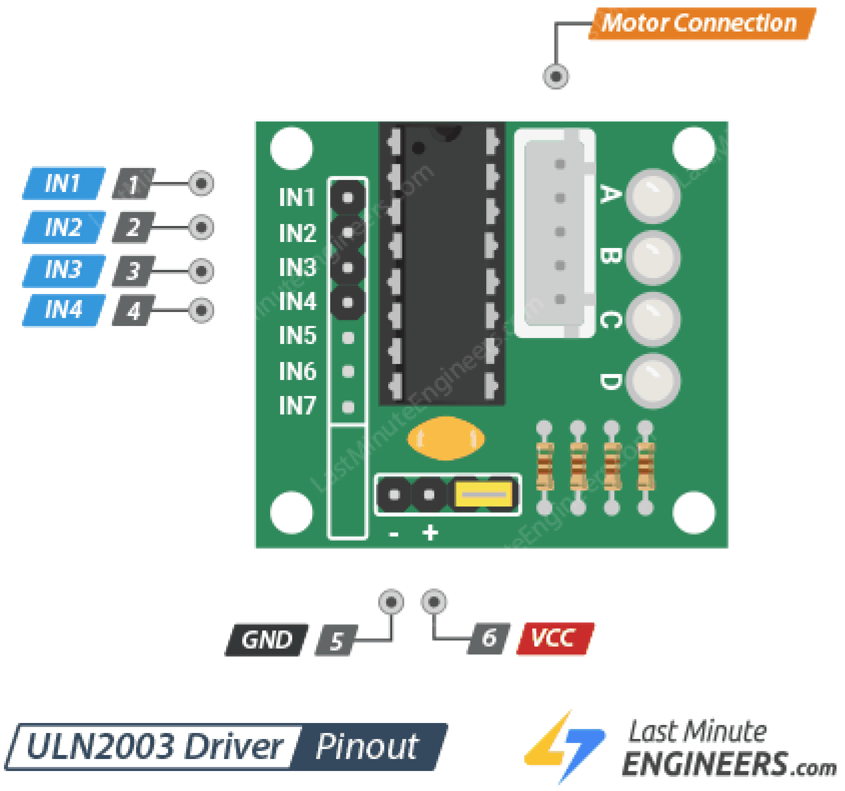
Additionally, there is a Molex connector that is compatible with the connector on the motor, allowing you to plug the motor directly into it.

The board includes four LEDs that indicate activity on the four control input lines. They provide a good visual indication while stepping.

There is an ON/OFF jumper on the board for disabling the stepper motor if needed.

ULN2003 Stepper Driver Board Pinout

The ULN2003 stepper driver board has the following pinout:



IN1 – IN4 are motor control input pins. Connect them to the Arduino's digital output pins.

GND is the ground pin.

VCC pin powers the motor. Because the motor consumes a significant amount of power, it is preferable to use an external 5V power supply rather than from the Arduino.

Motor Connector This is where the motor plugs in. The connector is keyed, so it will only go in one way.

Wiring 28BYJ-48 Stepper Motor and ULN2003 Driver to an Arduino

Let's get the motor connected to our Arduino!

The connections are straightforward. Begin by connecting an external 5V power supply to the ULN2003 driver.

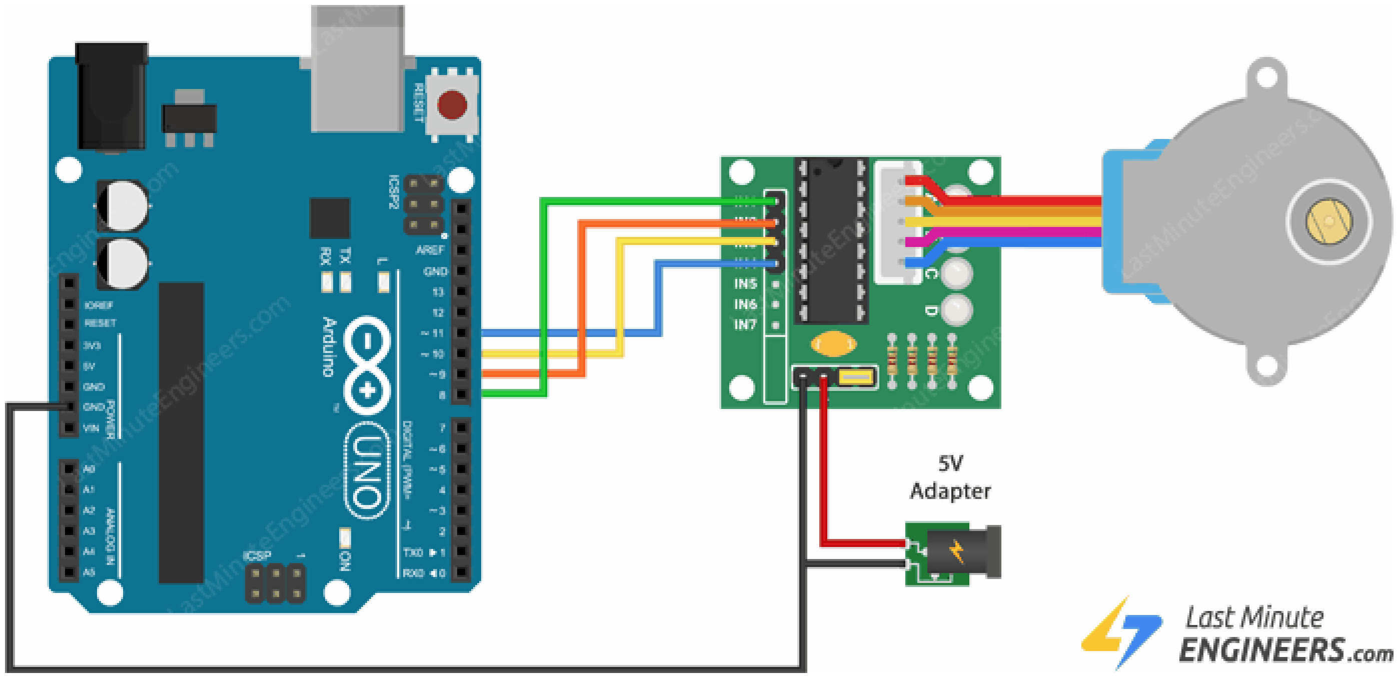
Warning:

The stepper motor can be powered directly from the Arduino, but this is not recommended because the motor can generate electrical noise on its power supply lines, which may damage the Arduino.

Connect the driver board's IN1, IN2, IN3, and IN4 to Arduino digital pins 8, 9, 10, and 11, respectively. Then connect the stepper motor to the ULN2003 driver.

Finally, make sure your circuit and Arduino share a common ground.

The following diagram shows you how to wire everything up.



IN1
IN2
IN3
IN4
IN5
IN6
IN7
5V
Adapter
Last Minute ENGINEERS.com

Arduino Example Code 1 – Using Built-in Stepper Library

For our first experiment, we'll use the Arduino Stepper Library, which comes with the Arduino IDE.

The Arduino stepper library handles the stepping sequence and allows you to control a wide range of unipolar and bipolar stepper motors.

Here is a simple sketch that turns the motor slowly in one direction, then rapidly in the opposite direction.

```
//Includes the Arduino Stepper Library
#include <Stepper.h>

// Defines the number of steps per rotation
const int stepsPerRevolution = 2038;

// Creates an instance of stepper class
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
    // Nothing to do (Stepper Library sets pins as outputs)
}

void loop() {
    // Rotate CW slowly at 5 RPM
    myStepper.setSpeed(5);
    myStepper.step(stepsPerRevolution);
    delay(1000);

    // Rotate CCW quickly at 10 RPM
    myStepper.setSpeed(10);
    myStepper.step(-stepsPerRevolution);
    delay(1000);
}
```

Code Explanation:

The sketch starts by including the built-in stepper library.

```
#include <Stepper.h>
```

Next, a constant stepsPerRevolution is defined, which contains the number of 'steps' the motor takes to complete one revolution. In our case, it is 2038.

```
const int stepsPerRevolution = 2038;
```

The step sequence of the 28BYJ-48 unipolar stepper motor is IN1-IN3-IN2-IN4. We will use this information to control the motor by creating an instance of the stepper library myStepper with the pin sequence 8, 10, 9, 11.

Make sure you do it right; otherwise, the motor won't work properly.

```
Stepper myStepper = Stepper(stepsPerRevolution, 8, 10, 9, 11);
```

Since the stepper library internally configures the four control pins as outputs, there is nothing to configure in the setup function, so it is left empty.

```
void setup() {
}
```

In the loop function, we use the setSpeed() function to specify the speed at which the stepper motor should move and the step() function to specify how many steps to take.

Passing a negative number to the step() function causes the motor to spin in the opposite direction.

The first code snippet causes the motor to spin very slowly clockwise, while the second causes it to spin very quickly counter-clockwise.

```
void loop() {
    // Rotate CW slowly at 5 RPM
    myStepper.setSpeed(5);
    myStepper.step(stepsPerRevolution);
    delay(1000);

    // Rotate CCW quickly at 10 RPM
    myStepper.setSpeed(10);
    myStepper.step(-stepsPerRevolution);
    delay(1000);
}
```

Please keep in mind that step() is a blocking function. This means that the Arduino will wait until the motor stops running before proceeding to the next line in your sketch. For example, on a 100-step motor, if you set the speed to 1 RPM and call step(100), this function will take a full minute to finish.

Arduino Example Code 2 – Using AccelStepper library

The Arduino Stepper Library is perfect for simple, single-motor applications. However, if you want to control multiple steppers, you'll need a more powerful library.

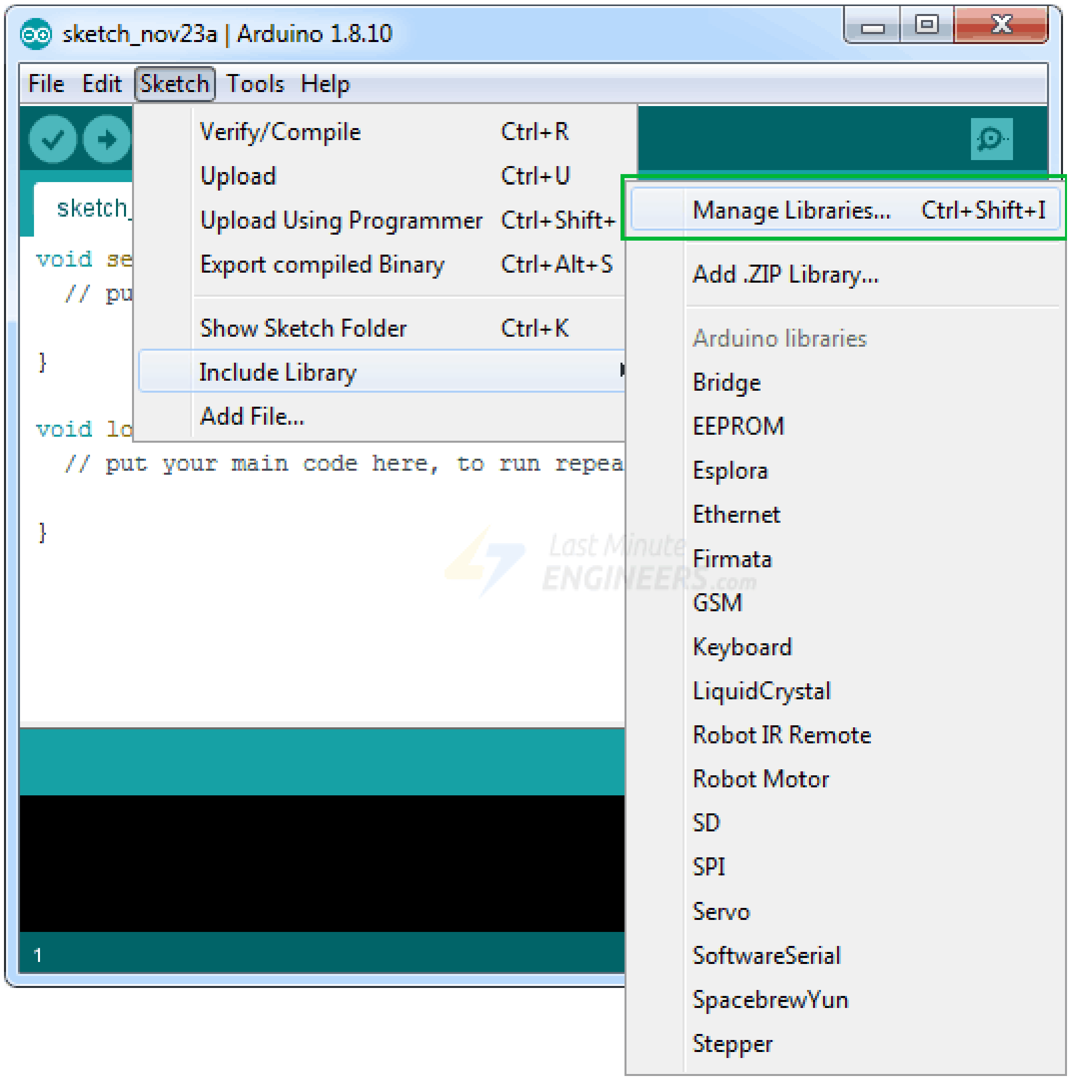
So, for our next experiment, we will use an advanced stepper motor library – the AccelStepper library. It outperforms the standard Arduino Stepper library in the following ways:

- It supports acceleration and deceleration.
- It supports half-step driving.
- It supports multiple steppers at once, with independent concurrent stepping on each stepper.

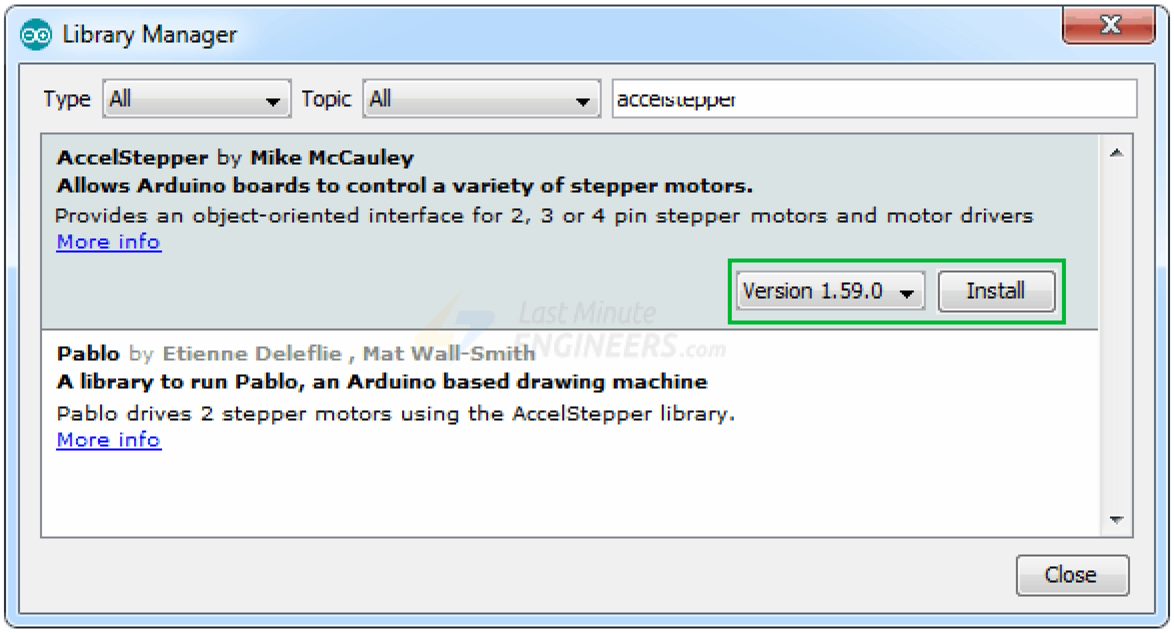
This library is not included with the Arduino IDE, so you'll need to install it first.

Library Installation

To install the library, navigate to **Sketch > Include Library > Manage Libraries...** Wait for the Library Manager to download the libraries index and update the list of installed libraries.



Filter your search by entering '**accelstepper**'. Click on the first entry, and then choose Install.



AccelStepper by **Mike McCauley**
Allows Arduino boards to control a variety of stepper motors.
Provides an object-oriented interface for 2, 3 or 4 pin stepper motors and motor drivers
More info

Pablo by Etienne Deleflie , Mat Wall-Smith
A library to run Pablo, an Arduino based drawing machine
Pablo drives 2 stepper motors using the AccelStepper library.
More info

Arduino Code

Here is a simple sketch that accelerates the stepper motor in one direction and then decelerates to come to rest. After one revolution, the motor reverses its spinning direction and repeats the process.

```
// Include the AccelStepper Library
#include <AccelStepper.h>

// Define step constant
#define MotorInterfaceType 4

// Creates an instance
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
AccelStepper myStepper(MotorInterfaceType, 8, 10, 9, 11);

void setup() {
    // set the maximum speed, acceleration factor,
    // initial speed and the target position
    myStepper.setMaxSpeed(1000.0);
    myStepper.setAcceleration(50.0);
    myStepper.setSpeed(200);
    myStepper.moveTo(2038);
}

void loop() {
    // Change direction once the motor reaches target position
    if (myStepper.distanceToGo() == 0)
        myStepper.moveTo(-myStepper.currentPosition());

    // Move the motor one step
    myStepper.run();
}
```

Code Explanation:

The sketch begins by including the newly installed AccelStepper library.

```
#include <AccelStepper.h>
```

Following that, we specify the motor interface type for the AccelStepper library. In this case, we will be driving a four-wire stepper motor in full-step mode, so we set the MotorInterfaceType constant to 4. If you want to run the motor in half-step mode, set the constant to 8.

```
#define MotorInterfaceType 4
```

Next, we create an instance of the stepper library called myStepper with the appropriate motor interface type and a pin sequence of 8, 10, 9, 11. (Keep in mind that the step sequence for these motors is IN1-IN3-IN2-IN4).

```
AccelStepper myStepper(MotorInterfaceType, 8, 10, 9, 11);
```

In the setup function, the maximum permitted speed of the motor is set to 1000 (the motor will accelerate up to this speed when we run it). The acceleration/deceleration rate is then set to add acceleration and deceleration to the stepper motor's movements.

The constant speed is set to 200. And, because the 28BYJ-48 takes 2038 steps per turn, the target position is also set to 2038.

```
void setup() {
    myStepper.setMaxSpeed(1000.0);
    myStepper.setAcceleration(50.0);
    myStepper.setSpeed(200);
    myStepper.moveTo(2038);
}
```

In the loop function, an if statement is used to determine how far the motor needs to travel (by reading the distanceToGo property) before reaching the target position (set by moveTo). When the distanceToGo reaches zero, the motor is rotated in the opposite direction by setting the moveTo position to negative of its current position.

At the bottom of the loop, you'll notice that the run() function is called. This is the most important function, as the stepper will not move if it is not executed.

```
void loop() {
    // Change direction once the motor reaches target position
    if (myStepper.distanceToGo() == 0)
        myStepper.moveTo(-myStepper.currentPosition());

    // Move the motor one step
    myStepper.run();
}
```

Arduino Example Code 3 – Control Two 28BYJ-48 Stepper Motors Simultaneously

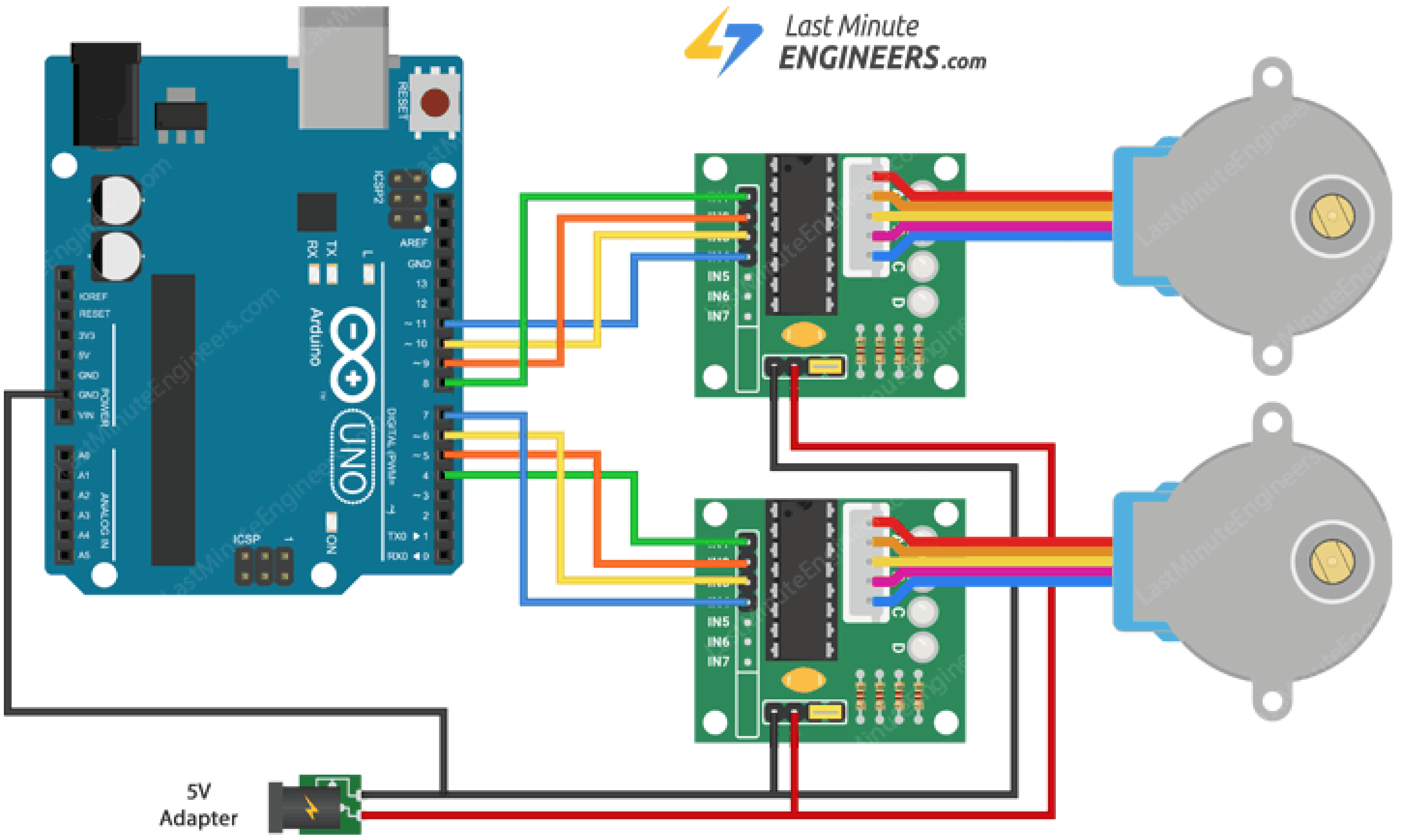
In our next experiment, we will attempt to run two 28BYJ-48 stepper motors simultaneously.

Wiring

Let's add a second 28BYJ-48 stepper motor to the setup.

Connect the ULN2003 driver's VCC to 5V and GND to ground. Now connect IN1, IN2, IN3, and IN4 to Arduino digital pins 4, 5, 6, and 7 respectively.

The following diagram shows the wiring.



Arduino Code

Here's a sketch that drives one motor in full-step mode and the other in half-step mode while accelerating and decelerating. After one revolution, their spinning direction changes.

```
// Include the AccelStepper Library
#include <AccelStepper.h>

// Define step constants
#define FULLSTEP 4
#define HALFSTEP 8
```

```
// Creates two instances
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
AccelStepper stepper1(HALFSTEP, 8, 10, 9, 11);
AccelStepper stepper2(FULLSTEP, 4, 6, 5, 7);

void setup() {
    // set the maximum speed, acceleration factor,
    // initial speed and the target position for motor 1
    stepper1.setMaxSpeed(1000.0);
    stepper1.setAcceleration(50.0);
    stepper1.setSpeed(200);
    stepper1.moveTo(2038);

    // set the same for motor 2
    stepper2.setMaxSpeed(1000.0);
    stepper2.setAcceleration(50.0);
    stepper2.setSpeed(200);
    stepper2.moveTo(-2038);
}

void loop() {
    // Change direction once the motor reaches target position
    if (stepper1.distanceToGo() == 0)
        stepper1.moveTo(-stepper1.currentPosition());
    if (stepper2.distanceToGo() == 0)
        stepper2.moveTo(-stepper2.currentPosition());

    // Move the motor one step
    stepper1.run();
    stepper2.run();
}
```

Code Explanation:

The sketch begins by including the AccelStepper library.

```
#include <AccelStepper.h>
```

As one motor will be driven in full-step mode and the other in half-step mode, the following two constants are defined.

```
#define FULLSTEP 4
#define HALFSTEP 8
```

Following that, two AccelStepper objects are created, one for each motor.

```
AccelStepper stepper1(HALFSTEP, 8, 10, 9, 11);
AccelStepper stepper2(FULLSTEP, 4, 6, 5, 7);
```

In the setup function, the maximum permitted speed for the first motor is set to 1000. The acceleration/deceleration rate is then set to add acceleration and deceleration to the stepper motor's movements. The constant speed is set to 200. And, because the 28BYJ-48 takes 2038 steps per turn, the target position is set to 2038.

To make the second stepper motor rotate in the opposite direction, we followed the exact same procedure as for the first stepper motor, with the exception of setting the target position to -2038.

```
void setup() {
    // settings for motor 1
    stepper1.setMaxSpeed(1000.0);
    stepper1.setAcceleration(50.0);
    stepper1.setSpeed(200);
    stepper1.moveTo(2038);

    // settings for motor 2
    stepper2.setMaxSpeed(1000.0);
    stepper2.setAcceleration(50.0);
    stepper2.setSpeed(200);
    stepper2.moveTo(-2038);
}
```

In the loop function, two if statements are used, one for each motor, to determine how far the motors must travel (by reading the distanceToGo property) to reach their target position (set by moveTo). When the distanceToGo reaches zero, the motors are instructed to rotate in the opposite direction by setting their moveTo position to a negative value.

Finally, the run() function is called to get the motors spinning.

```
void loop() {
    // Change direction once the motor reaches target position
    if (stepper1.distanceToGo() == 0)
        stepper1.moveTo(-stepper1.currentPosition());
    if (stepper2.distanceToGo() == 0)
        stepper2.moveTo(-stepper2.currentPosition());

    // Move the motor one step
    stepper1.run();
    stepper2.run();
}
```