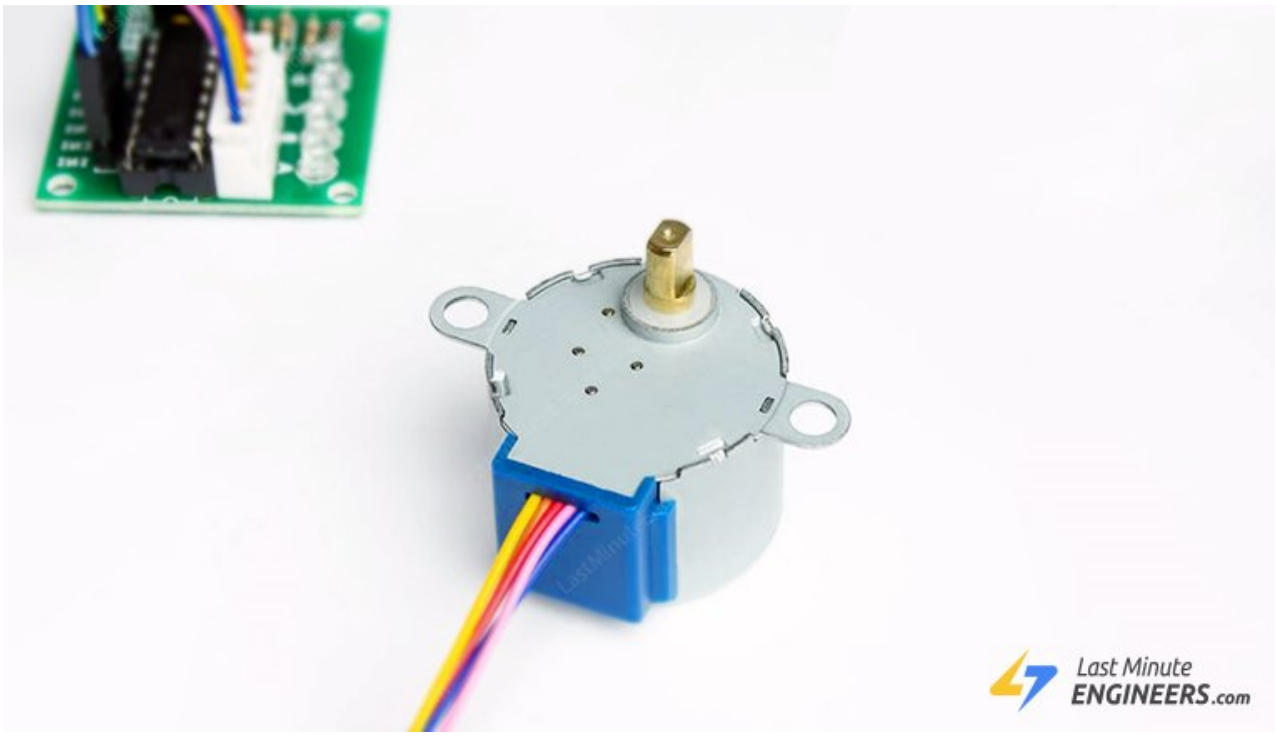
Control 28BYJ-48 Stepper Motor with ULN2003 Driver & Arduino

lastminuteengineers.com/28byj48-stepper-motor-arduino-tutorial

22 December 2019



Stepper motors are great motors for position control. They are a special type of brushless motors that divides a full rotation into a number of equal “steps”. They are usually found in desktop printers, 3D printers, CNC milling machines, and anything else that requires precise positioning control.

One of the inexpensive way to learn about stepper motors is to use **28BYJ-48** stepper motors. They usually come with a ULN2003 based driver board which makes them super easy to use.

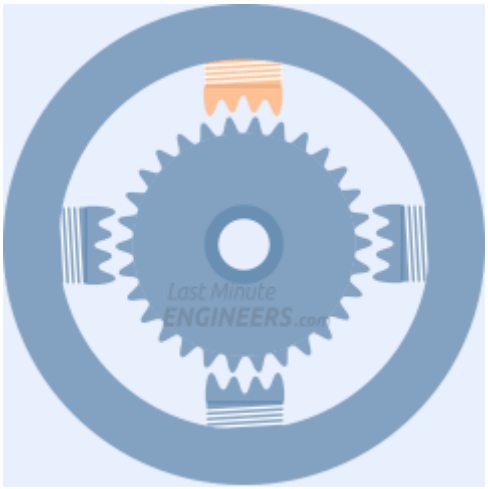
Do you know how these stepper motors work?

These stepper motors use a cogged wheel (having 32 teeth) and four electromagnets to rotate the wheel one ‘step’ at a time.

Each HIGH pulse sent, energizes the coil, attracts the nearest teeth of the cogged wheel and drives the motor one step.

The way you pulse these coils greatly affects the behavior of the motor.

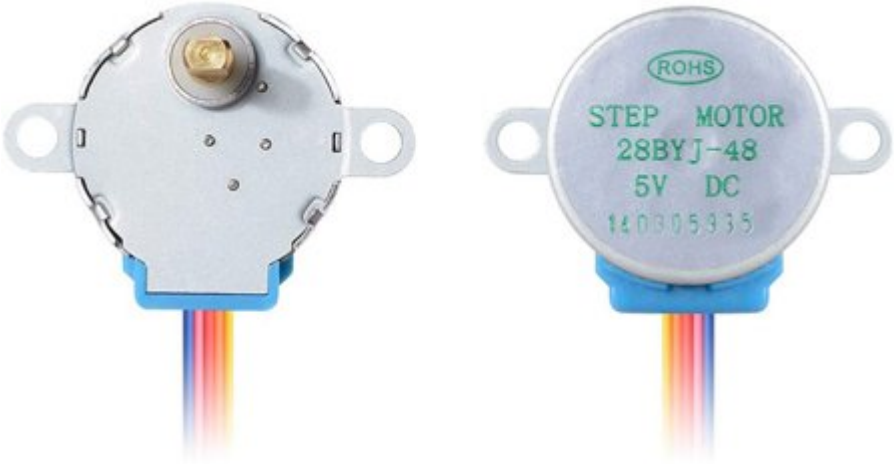
- The sequence of pulses determines the spinning direction of the motor.
- The frequency of the pulses determines the speed of the motor.
- The number of pulses determines how far the motor will turn.



The 28BYJ-48 Stepper Motor

The 28BYJ-48 is a 5-wire unipolar stepper motor that runs on 5 volts.

The interesting thing about this motor is that people have been using it in countless applications over the last few decades. It is used in air-conditioner, vending machines and many other applications.



One of the best things about these motors is that they can be positioned accurately, one 'step' at a time.

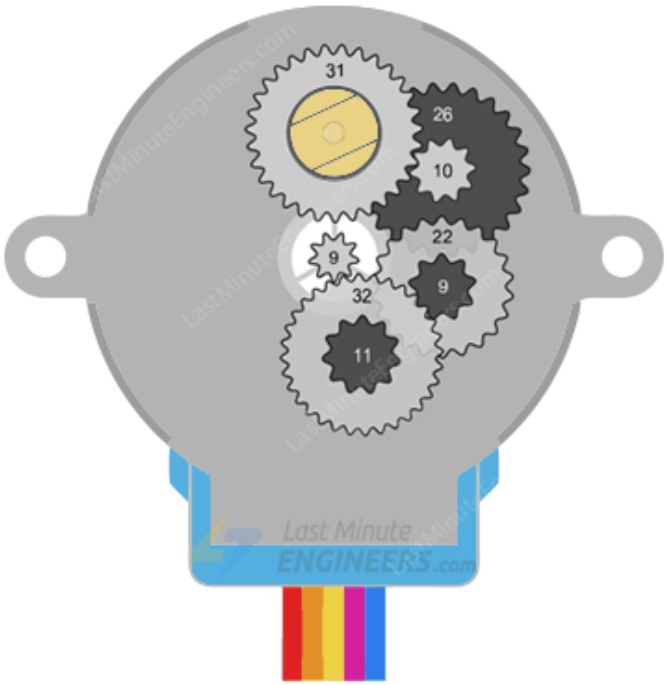
The other advantage is that they are relatively precise in their movement and they are quite reliable since the motor does not use contact brushes.

They generally give good torque even in the stand-still state which is maintained as long as power is supplied to the motor.

The only downside is that they are a bit hungry for power and consume power even when they are not moving.

28BYJ-48 Gear Reduction Ratio

According to the data sheet, when the 28BYJ-48 motor runs in full step mode, each step corresponds to a rotation of 11.25°. That 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 has a 1/64 reduction gear set. (Actually its 1/63.68395 but for most purposes 1/64 is a good enough approximation)

What this means is that there are actually 32*63.68395 steps per revolution = 2037.8864 ~ 2038 steps!

28BYJ-48 Power Consumption

The power consumption of the motor is around 240mA.

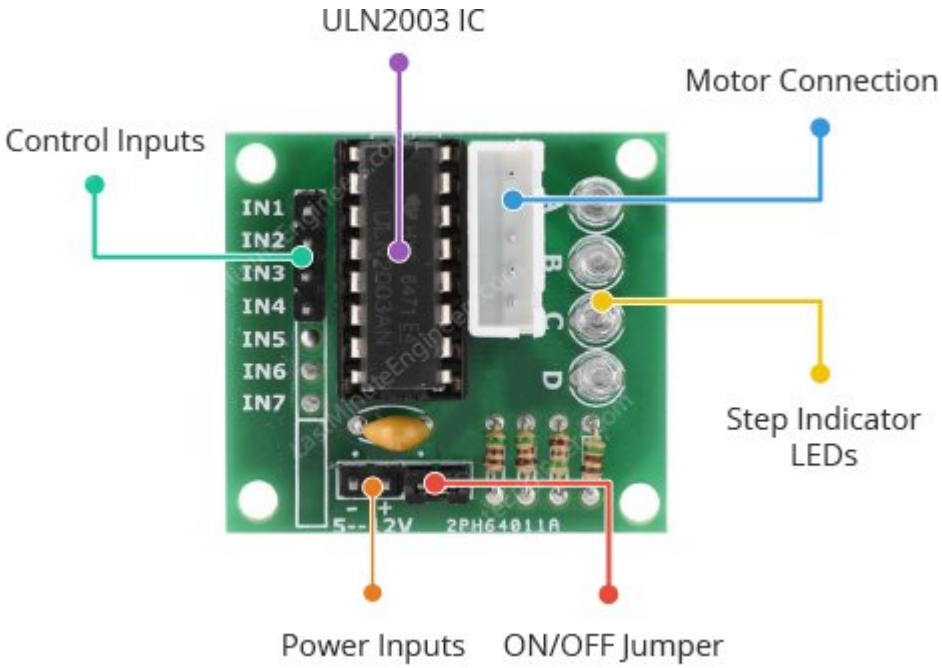
Because the motor draws too much power, it is best to power it directly from an external 5V power supply rather than drawing that power from the Arduino.

The motor consumes power, even in the stand still state, to maintain its position.

The ULN2003 Driver Board

The motor usually comes with a ULN2003 based driver board.

The ULN2003 is one of the most common motor driver ICs, consisting of an array of 7 Darlington transistor pairs, each pair is capable of driving loads of up to 500mA and 50V. Four out of seven pairs are used on this board.



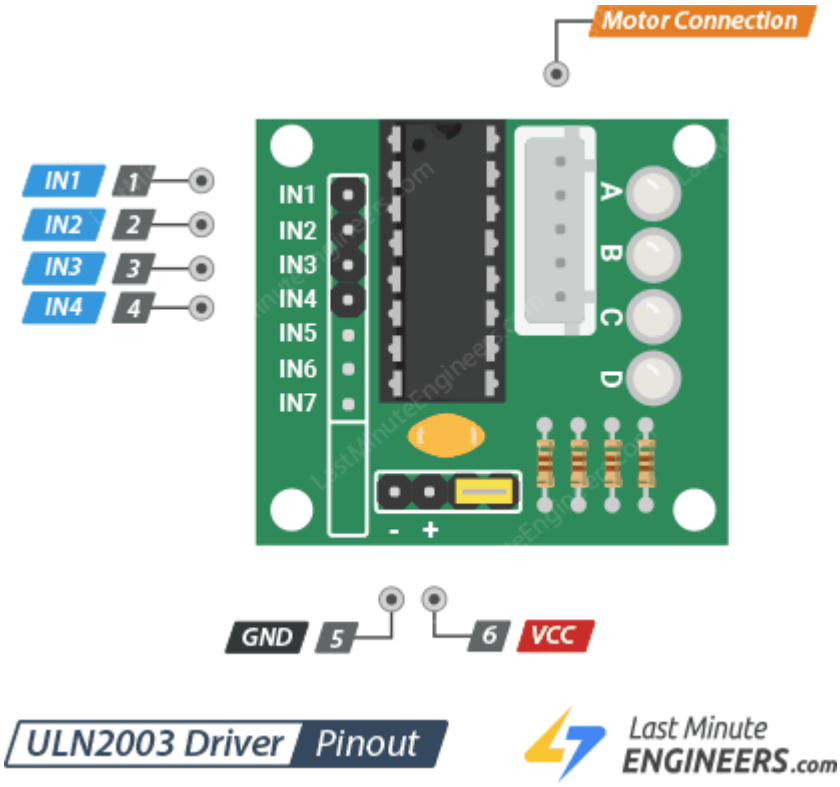
The board has a connector that mates the motor wires perfectly which makes it very easy to connect the motor to the board. There are also connections for four control inputs as well as power supply connections.

The board has four LEDs that show activity on the four control input lines (to indicate stepping state). They provide a nice visual when stepping.

The board also comes with an ON/OFF jumper to isolate power to the stepper Motor.

ULN2003 Stepper Driver Board Pinout

The pinouts of the ULN2003 stepper driver board are as follows:



IN1 – IN4 pins are used to drive the motor. Connect them to a digital output pins on the Arduino.

GND is a common ground pin.

VDD pin supplies power for the motor. Connect it to an external 5V power supply. Because the motor draws too much power, you should NEVER use the 5V power from your Arduino to power this stepper motor.

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

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

Now that we know everything about the motor, we can begin hooking it up to our Arduino!

Start by connecting the power supply up to the ULN2003 driver.

Note that it is possible to directly power the stepper motor from the Arduino. However, this is not recommended; as the motor may induce **electrical noise** onto its power supply lines and this could damage the Arduino.

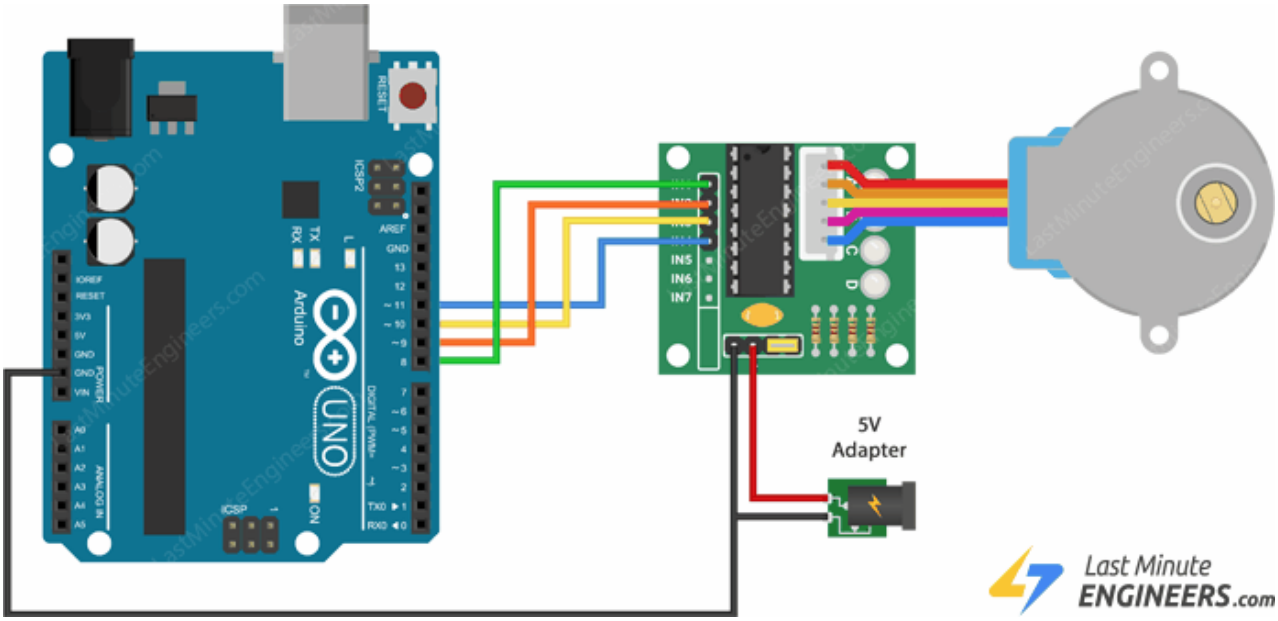
So, use a separate 5V power supply to power your stepper motors.

Next connect the ground from that power supply to the arduino's ground. This is very important so that we establish the same voltage reference between the two.

Now connect the driver board's IN1, IN2, IN3, IN4 to the Arduino digital pins 8, 9, 10, and 11 respectively.

Finally, hook the motor cable from the stepper motor up to the driver board.

When you're done you should have something that looks similar to the illustration shown below.



Arduino Code – Using Built-in Stepper Library

For our first experiment we will make use of the Arduino Stepper Library which comes packaged with your Arduino IDE.

The stepper library takes care of the stepping sequence and makes it straight forward to control a wide variety of stepper motors, both unipolar and bipolar.

Here's the simple sketch that moves the stepper motor clockwise slowly and then counterclockwise quickly.

```
//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
        myStepper.setSpeed(100);
        myStepper.step(stepsPerRevolution);
        delay(1000);

        // Rotate CCW quickly
        myStepper.setSpeed(700);
        myStepper.step(-stepsPerRevolution);
        delay(1000);
}
```

Code Explanation:

The sketch starts with including the Arduino Stepper Library.

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

Next, we define a constant `stepsPerRevolution` which holds the number of 'steps' that the motor will take to complete one revolution. In our case, it's 2038.

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

The 28BYJ-48 Unipolar stepper motor has a step sequence of IN1-IN3-IN2-IN4. We will use this information to drive the motor by creating an instance of stepper library called `myStepper` with the pin sequence of 8, 10, 9, 11.

Make sure you get this right or the motor will not operate properly.

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

There is nothing to set in the setup function as the Stepper library internally sets the four I/O pins as outputs.

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

In the loop function, we use the `setSpeed()` function to set the speed that we wish the stepper motor to move and subsequently use the `step()` function to tell it how many steps to rotate. Passing a negative number to the `step()` function reverses the spinning direction of the motor.

First code snippet will turn the motor clockwise very slowly. And the second will turn the motor counterclockwise at a much faster speed.

```
void loop() {
        // Rotate CW slowly
        myStepper.setSpeed(100);
        myStepper.step(stepsPerRevolution);
        delay(1000);

        // Rotate CCW quickly
        myStepper.setSpeed(700);
        myStepper.step(-stepsPerRevolution);
        delay(1000);
}
```

Arduino Code – Using AccelStepper library

The Arduino Stepper Library is perfectly adequate for simple, single motor applications. But when you want to control multiple steppers, you'll need a better library.

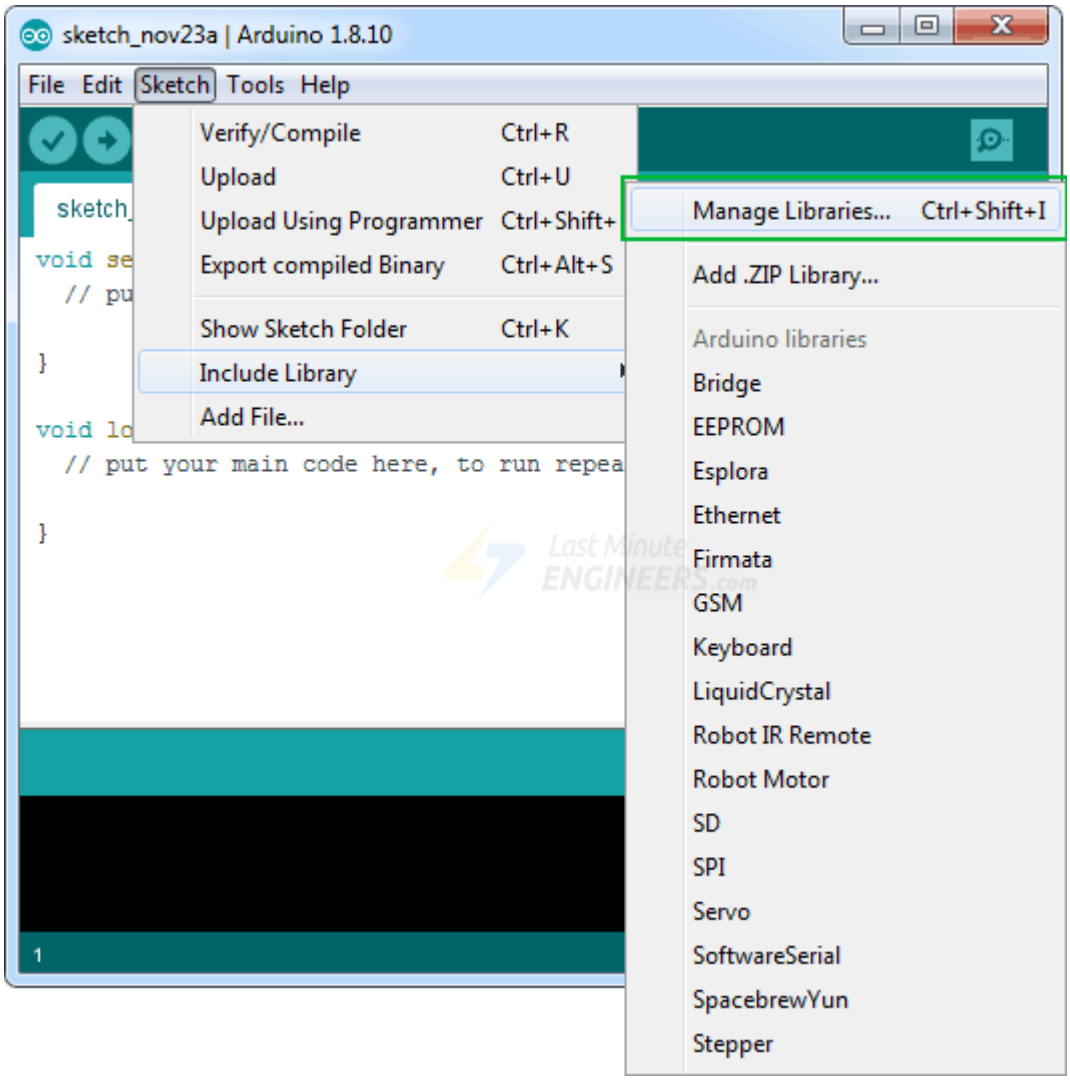
So, for our next experiment we will make use of an advanced stepper motor library called AccelStepper library. It significantly improves on the standard Arduino Stepper library in several ways:

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

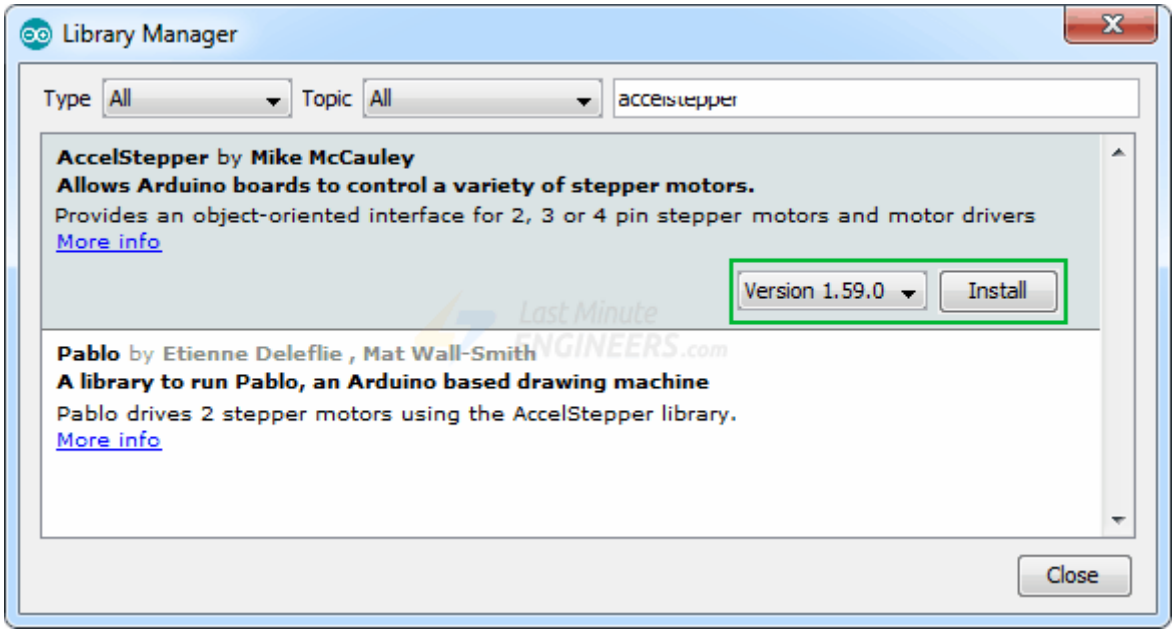
This library is not included in the Arduino IDE, so you will need to install it first.

Library Installation

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



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



Arduino Code

Here's the simple sketch that accelerates the stepper motor in one direction and then decelerates to come to rest. Once the motor makes one revolution, it changes the spinning direction. And it keeps doing that over and over again.

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

// Define step constant
#define FULLSTEP 4

// Creates an instance
// Pins entered in sequence IN1-IN3-IN2-IN4 for proper step sequence
AccelStepper myStepper(FULLSTEP, 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:

We start off by including the newly installed AccelStepper library.

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

Now as we are going to drive our motor at full steps, we will define a constant for this. If you want to drive the motor at half steps, set the constant to 8.

```
#define FULLSTEP 4
```

Next, we create an instance of stepper library called `myStepper` with the pin sequence of 8, 10, 9, 11 (Remember the step sequence for these motors is IN1-IN3-IN2-IN4).

Again, make sure you get this right or the motor will not operate properly.

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

In the setup function we first set the maximum speed of the motor to a thousand which is about as fast as these motors can go. We then set an acceleration factor for the motor to add acceleration and deceleration to the movements of the stepper motor.

Next we set the regular speed of 200 and the number of steps we're going to move it to i.e. 2038 (as you recall the 28BYJ-48 with its gearing will move 2038 steps per revolution).

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

In the loop function, we use an If statement to check how far the motor needs to travel (by reading the `distanceToGo` property) until it reaches the target position (set by `moveTo`). Once `distanceToGo` reaches zero we will move the motor in the opposite direction by changing the `moveTo` position to the negative of its current position.

Now at the bottom of the loop you'll notice we have called a `run()` function. This is the most important function, because the stepper will not run until this function is 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();
}
```

Control Two 28BYJ-48 Stepper Motors Simultaneously

For our next experiment, we will add a second 28BYJ-48 stepper and ULN2003 driver set to our Arduino to drive two motors simultaneously.

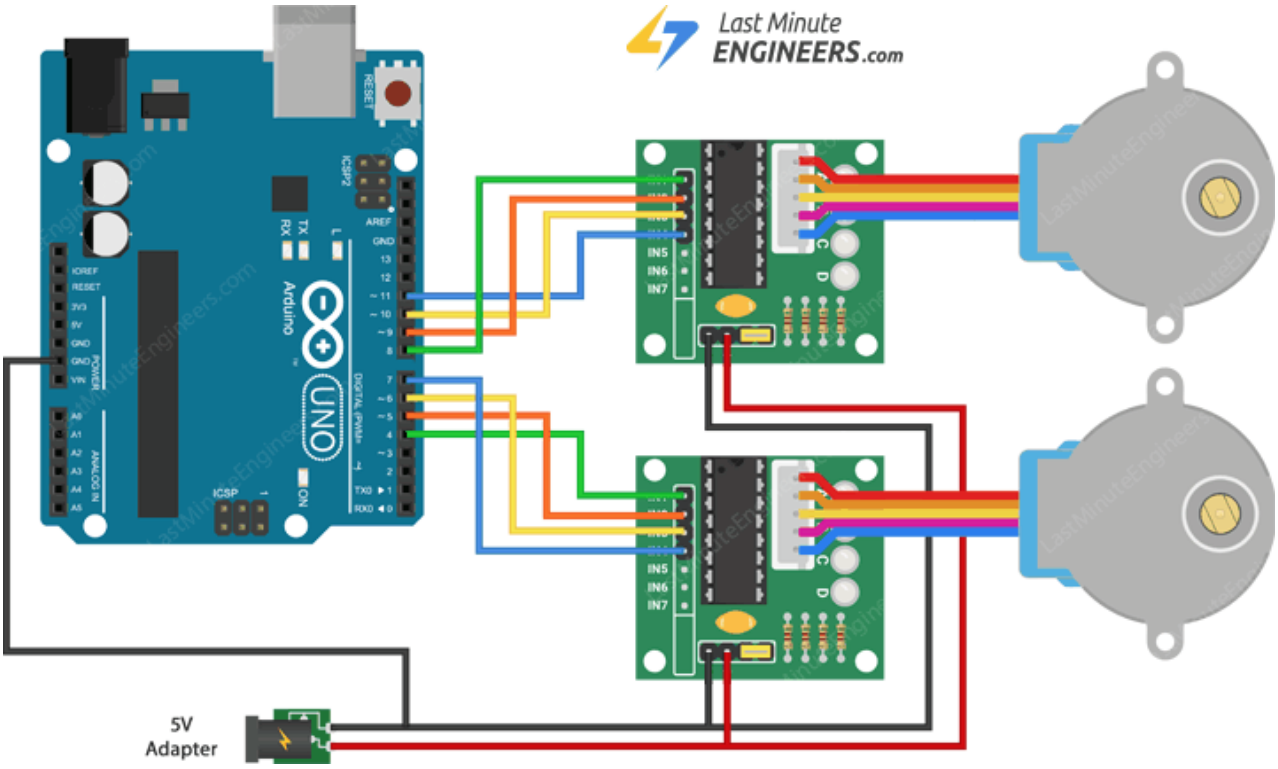
Wiring

Leave the connections you made earlier as they are and wire the new devices as follows:

Once again we will use the separate 5V power supply to supply power to the ULN2003 driver board.

Now connect the second driver board's IN1, IN2, IN3, IN4 to the Arduino digital pins 4, 5, 6, and 7 respectively.

The following illustration shows the wiring.



Arduino Code

Here's the sketch that drives one motor at full steps and the second one at half steps. When the motors make one revolution their spinning direction will change. There's some acceleration and deceleration involved as well.

```
// 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:

We begin by including the AccelStepper Library.

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

Now as we are going to drive one motor at full steps and the second one at half steps. We will define two constants for that.

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

Next, we create two motor objects, one for each motor. We use our pin definitions and the step definitions to set these up.

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

In the setup function we first set the maximum speed of the `stepper1` to a thousand. We then set an acceleration factor for the motor to add acceleration and deceleration to the movements of the stepper motor.

Next we set the regular speed of 200 and the number of steps we're going to move it to i.e. 2038 (as you recall the 28BYJ-48 with its gearing will move 2038 steps per revolution).

We're going to do exactly the same thing for `stepper2` except we'll instruct it to move to -2038 because we want it to move counterclockwise.

```
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, we use two `If statements`, one for each motor, to check how far the motors need to travel (by reading the `distanceToGo` property) until they reach their target positions (set by `moveTo`). Once `distanceToGo` reaches zero, we will change their `moveTo` position to the negative of their current position, so that they start moving in the opposite direction.

Finally we set them into motion by calling the `run()` function.

```
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();
}
```