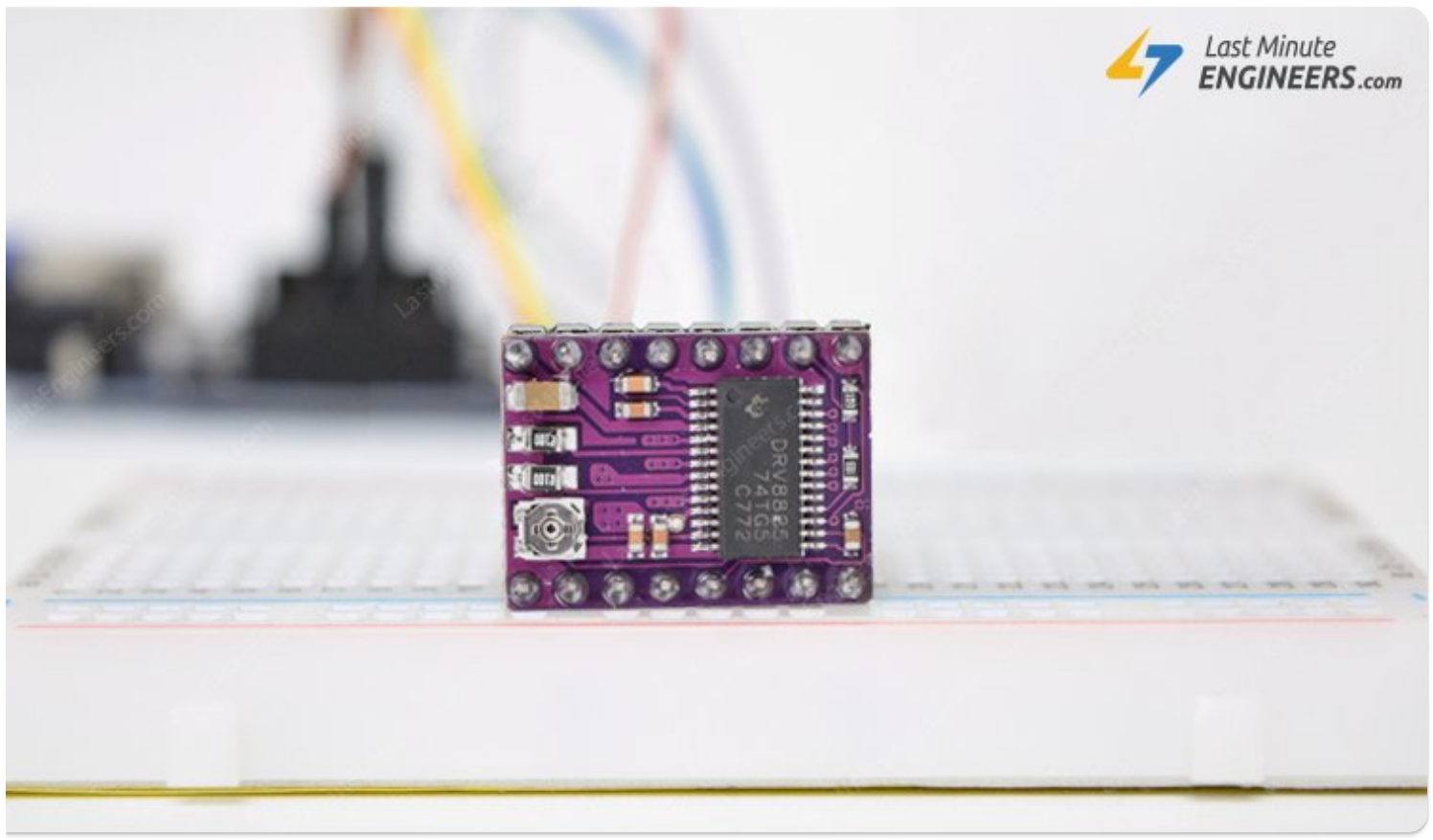
Control Stepper Motor with DRV8825 Driver Module & Arduino



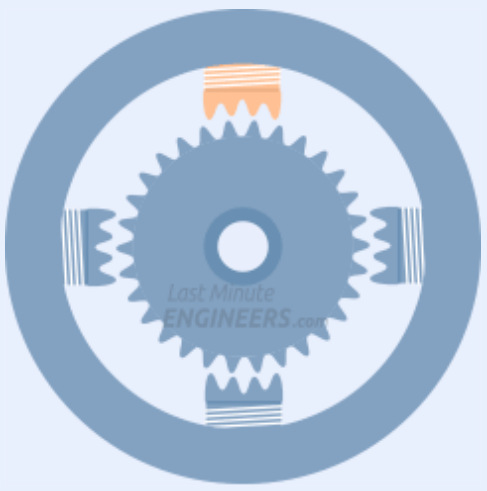
If you are planning on building your own 3D printer or a CNC machine, you will need to control a bunch of stepper motors. And having one Arduino control all of them can take up a lot of the processing and not leave you a lot of room to do anything else; unless you use a self-contained dedicated stepper motor driver – DRV8825.

It can control both speed and spinning direction of a bipolar stepper motor like NEMA 17 with just two pins. How cool is that!

Do you know how stepper motors work?

The stepper motors use a cogged wheel and 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.

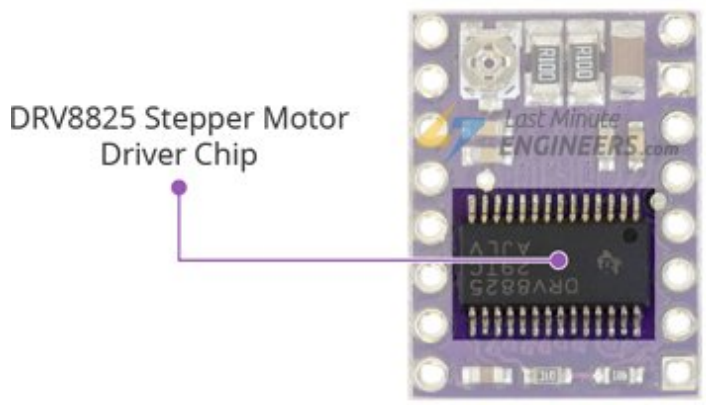


The way you pulse these coils greatly affects the behavior of the motor.

- 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.

DRV8825 Stepper Motor Driver Chip

At the heart of the module is a Microstepping Driver from Texas Instruments – DRV8825. It's small in stature (only 0.8″ × 0.6″) but still packs a punch.



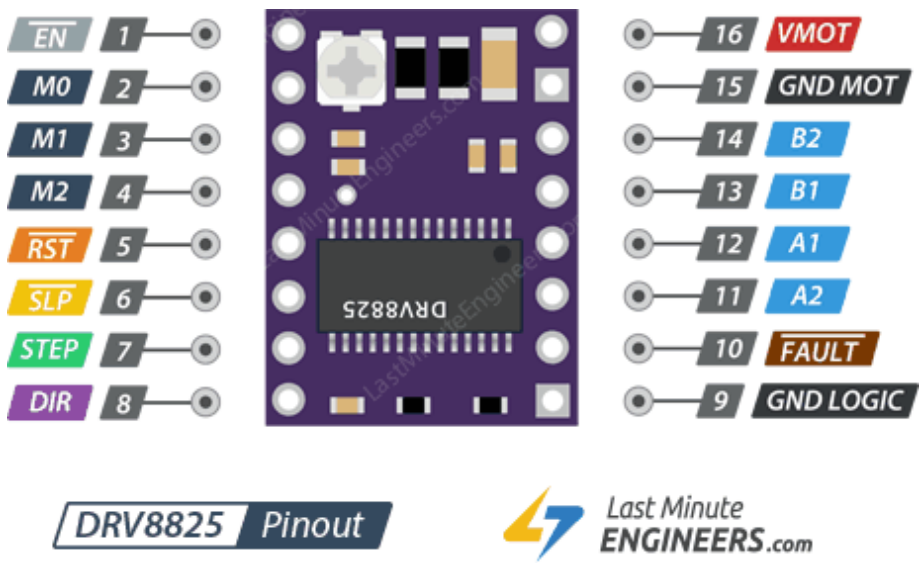
The DRV8825 stepper motor driver has output drive capacity of up to 45V and lets you control one bipolar stepper motor at up to 2.2A output current per coil.

The driver has built-in translator for easy operation. This reduces the number of control pins to just 2, one for controlling the steps and other for controlling spinning direction.

The driver offers 6 different step resolutions viz. full-step, half-step, quarter-step, eighth-step, sixteenth-step and thirty-second-step.

DRV8825 Motor Driver Pinout

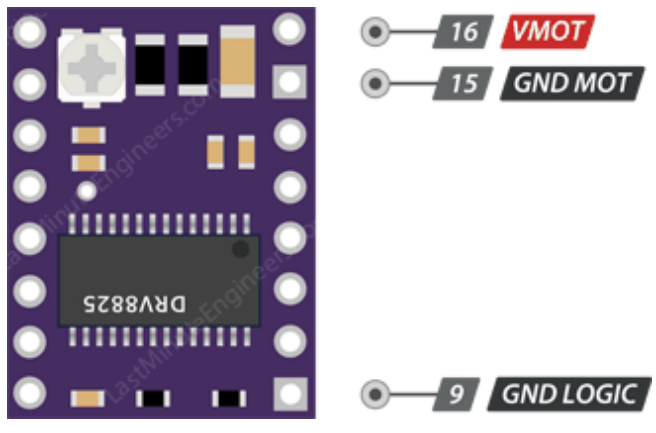
The DRV8825 driver has total 16 pins that interface it to the outside world. The connections are as follows:



Let's familiarize ourselves with all the pins one by one.

Power Connection Pins

Unlike other typical stepper motor drivers, the DRV8825 has only one power supply connection.



VMOT & GND MOT supplies power for the motor which can be 8.2V to 45V.

The module does not have any logic supply pin as DRV8825 gets its power from the internal 3V3 voltage regulator.

However, you should common your microcontroller's ground with `GND LOGIC` pin.

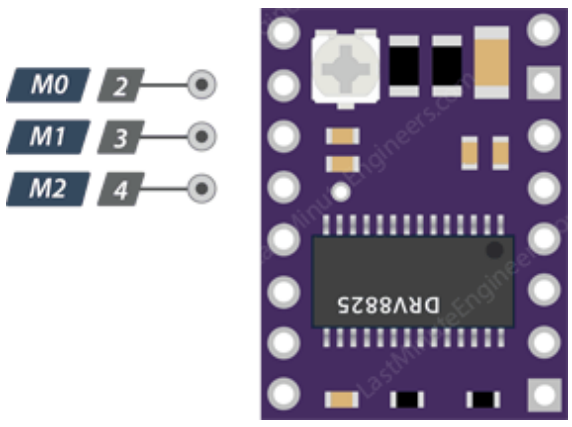
According to datasheet, the motor supply requires appropriate decoupling capacitor close to the board, capable of sustaining 4A.

> Warning:
>
> This driver has low-ESR ceramic capacitors on board, which makes it vulnerable to voltage spikes. In some cases, these spikes can exceed the 45V(maximum voltage rating of DRV8825), potentially permanently damaging the board and even the motor.
>
> One way to protect the driver from such spikes is to put a large 100µF (at least 47µF) electrolytic capacitor across motor power supply pins.

Microstep Selection Pins

The DRV8825 driver allows microstepping by allowing intermediate step locations. This is achieved by energizing the coils with intermediate current levels.

For example, if you choose to drive NEMA 17 having 1.8° or 200 steps per revolution in quarter-step mode, the motor will give 800 microsteps per revolution.



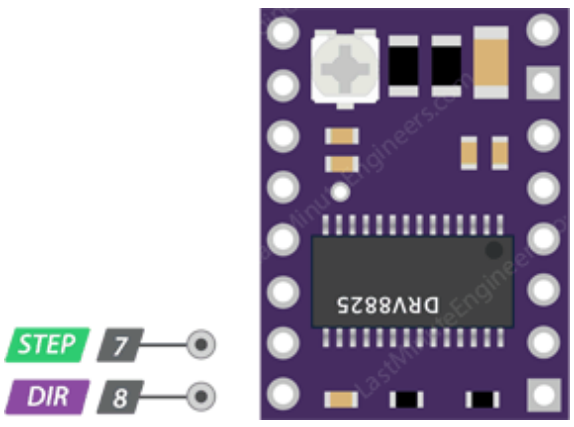
The DRV8825 driver has three step size(resolution) selector inputs viz. `M0, M1 & M2` . By setting appropriate logic levels to these pins we can set the motors to one of the six step resolutions.

M0	M1	M2	Microstep Resolution
Low	Low	Low	Full step
High	Low	Low	Half step
Low	High	Low	1/4 step
High	High	Low	1/8 step
Low	Low	High	1/16 step
High	Low	High	1/32 step
Low	High	High	1/32 step
High	High	High	1/32 step

These three microstep selection pins are pulled LOW by internal pull-down resistors, so if we leave them disconnected, the motor will operate in full step mode.

Control Input Pins

The DRV8825 has two control inputs viz. STEP and DIR.



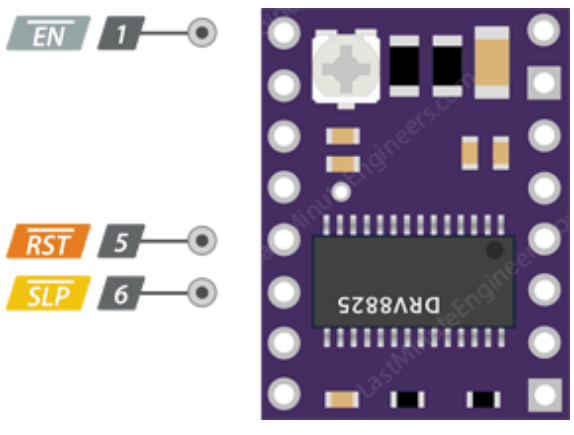
STEP input controls the mirosteps of the motor. Each HIGH pulse sent to this pin steps the motor by number of microsteps set by Microstep Selection Pins. The faster the pulses, the faster the motor will rotate.

DIR input controls the spinning direction of the motor. Pulling it HIGH drives the motor clockwise and pulling it LOW drives the motor counterclockwise.

If you just want the motor to rotate in a single direction, you can tie DIR directly to VCC or GND accordingly.

Pins For Controlling Power States

The DRV8825 has three different inputs for controlling its power states viz. EN, RST, and SLP.



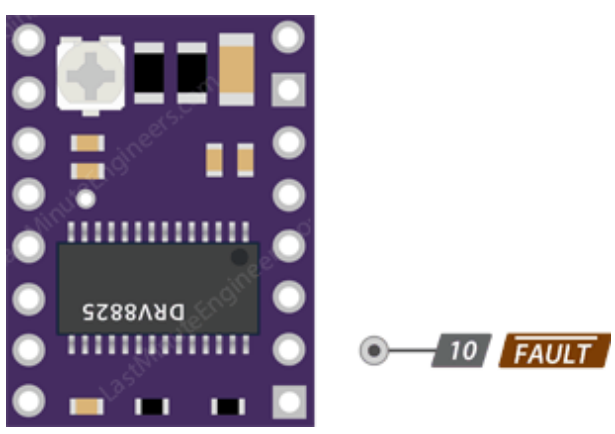
EN Pin is active low input, when pulled LOW(logic 0) the DRV8825 driver is enabled. By default this pin is pulled low so the driver is always enabled, unless you pull it HIGH.

SLP Pin is active low input. Meaning, pulling this pin LOW puts the driver in sleep mode, minimizing the power consumption. You can invoke this especially when the motor is not in use to conserve power.

RST is also an active low input. When pulled LOW, all STEP inputs are ignored, until you pull it HIGH. It also resets the driver by setting the internal translator to a predefined Home state. Home state is basically the initial position from where the motor starts and it's different depending upon the microstep resolution.

Fault Detection Pin

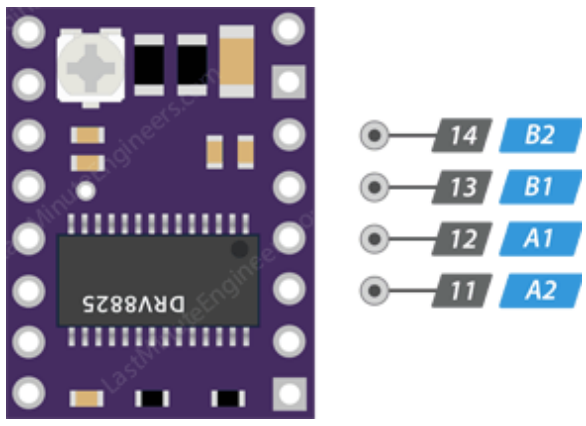
The DRV8825 also features a *FAULT* output that drives LOW whenever the H-bridge FETs are disabled as the result of over-current protection or thermal shutdown.



Actually the Fault pin is shorted to SLEEP pin so, whenever the Fault pin is driven LOW, the whole chip is disabled. And it remains disabled until either RESET, or Motor Voltage VMOT is removed and reapplied.

Output Pins

The DRV8825 motor driver's output channels are broken out to the edge of the module with *B2, B1, A1 & A2* pins.



You can connect any bipolar stepper motor having voltages between 8.2V to 45 V to these pins.

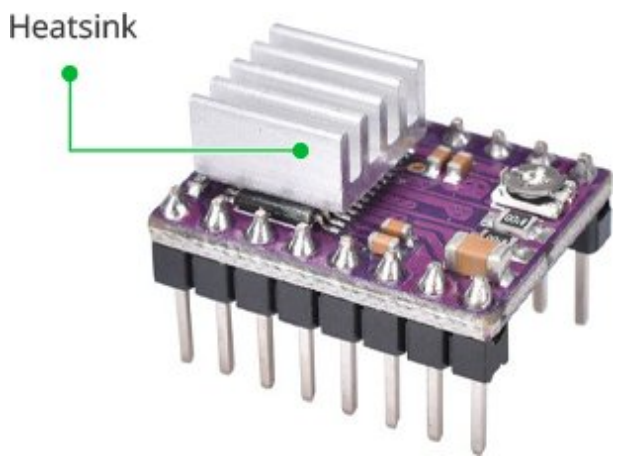
Each output pin on the module can deliver up to 2.2A to the motor. However, the amount of current supplied to the motor depends on system's power supply, cooling system & current limiting setting.

Cooling System – Heatsink

Excessive power dissipation of the DRV8825 driver IC results in the rise of temperature that can go beyond the capacity of IC, probably damaging itself.

Even if the DRV8825 driver IC has a maximum current rating of 2.2A per coil, the chip can only supply approximately 1.5A per coil without getting overheated.

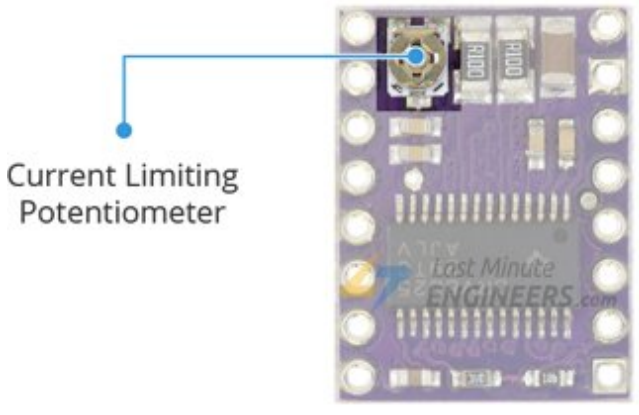
For achieving more than 1.5A per coil, a heat sink or other cooling method is required.



The DRV8825 driver usually comes with a heatsink. It is advisable to install it before you use the driver.

Current limiting

Before using the motor, there's a small adjustment that we need to make. We need to limit the maximum amount of current flowing through the stepper coils and prevent it from exceeding the motor's rated current.



There's a small trimmer potentiometer on the DRV8825 driver that can be used to set the current limit. You should set the current limit to be at or lower than the current rating of the motor.

To make this adjustment there are two methods:

Method 1:

In this method we are going to set the current limit by measuring the voltage (Vref) on the “ref” pin.

. Take a look at the datasheet for your stepper motor. Note down it’s rated current. In our case we are using NEMA 17 200steps/rev, 12V 350mA.

. Put the driver into full-step mode by leaving the three microstep selection pins disconnected.

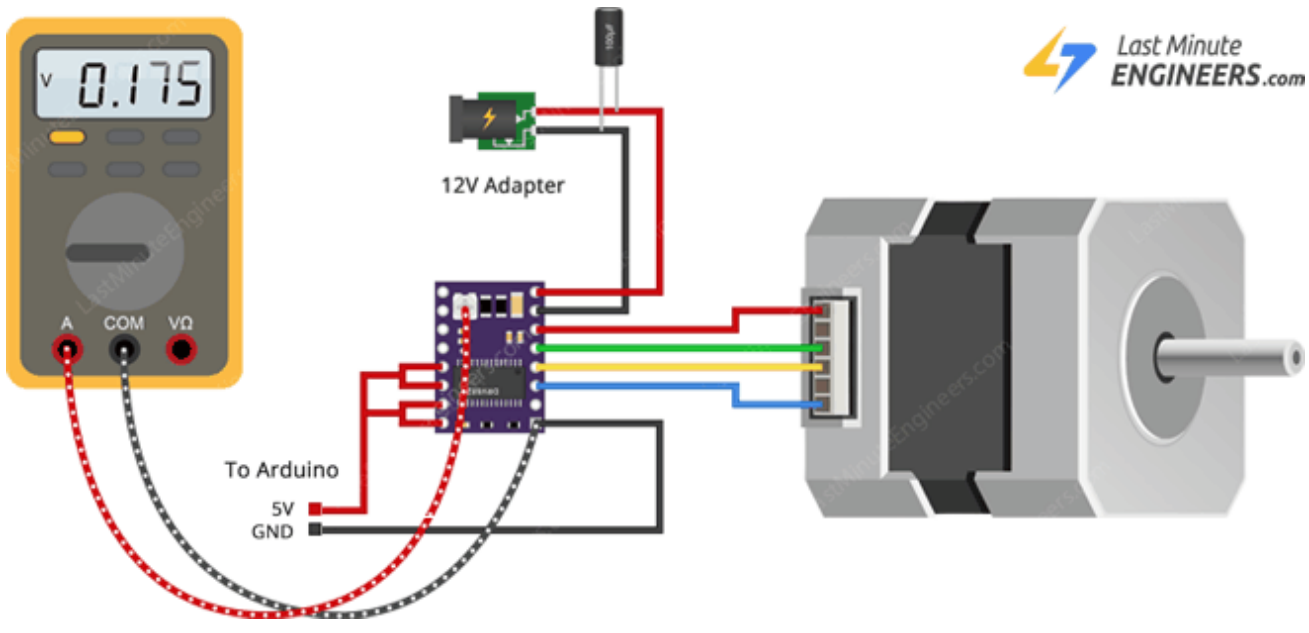
. Hold the motor at a fixed position by not clocking the STEP input.

. Measure the voltage (Vref) on the metal trimmer pot itself while you adjust it.

. Adjust the Vref voltage using the formula

$$\text{Current Limit} = \text{Vref} \times 2$$

For example, if your motor is rated for 350mA, you would adjust the reference voltage to 0.175V.



> Tip:
>
> An easy way to make adjustments is to use an alligator clip on the shaft of a metal screwdriver and attach that to your multimeter so that you can measure and adjust the voltage with the screwdriver at the same time.

Method 2:

In this method we are going to set the current limit by measuring the current running through the coil.

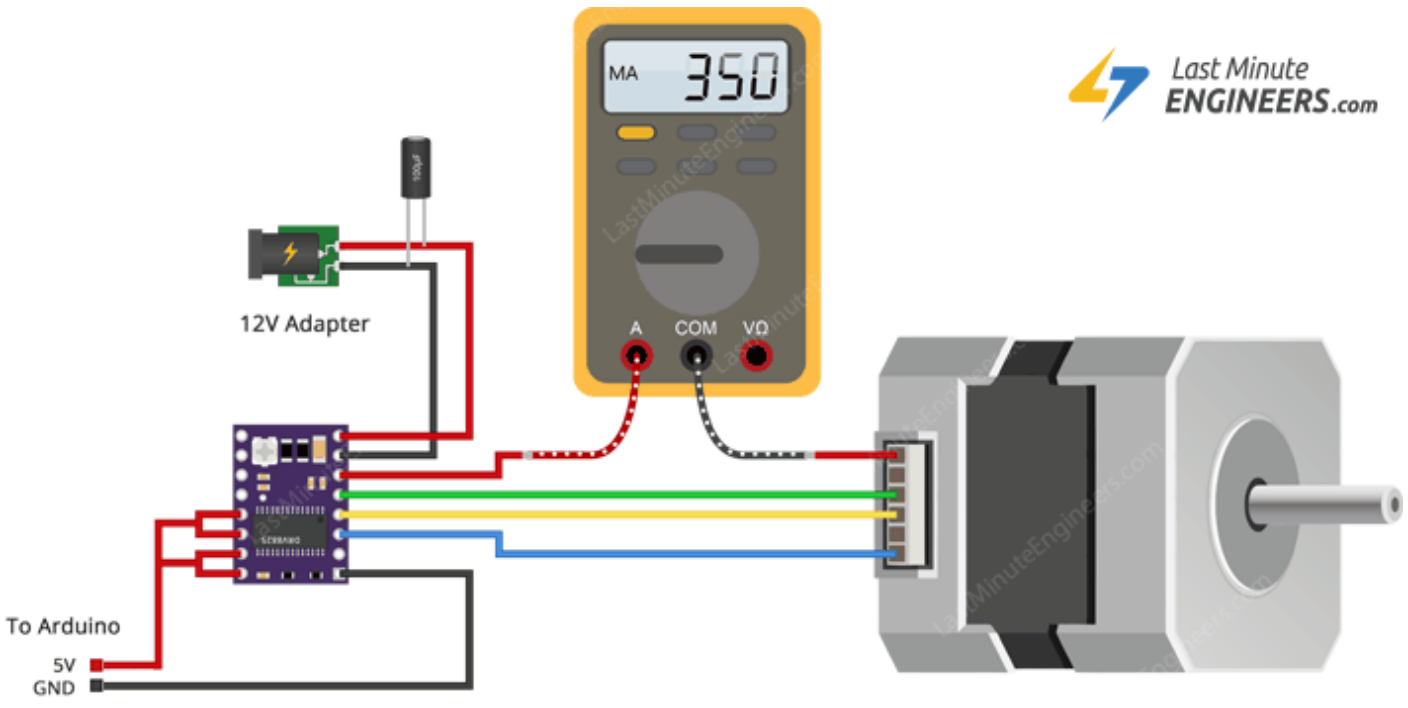
. Take a look at the datasheet for your stepper motor. Note down it’s rated current. In our case we are using NEMA 17 200steps/rev, 12V 350mA.

. Put the driver into full-step mode by leaving the three microstep selection pins disconnected.

. Hold the motor at a fixed position by not clocking the STEP input.

. Place the ammeter in series with one of the coils on your stepper motor and measure the actual current flowing.

. Take a small screwdriver and adjust the current limit potentiometer until you reach rated current.



You will need to perform this adjustment again if you ever change the logic voltage(VDD)

Wiring DRV8825 stepper motor driver with Arduino UNO

Now that we know everything about the driver, we will connect it to our Arduino.

Connections are fairly simple. Start by connecting RST pin to the adjacent SLP/SLEEP pin and both to the 5V on the Arduino to keep the driver enabled.

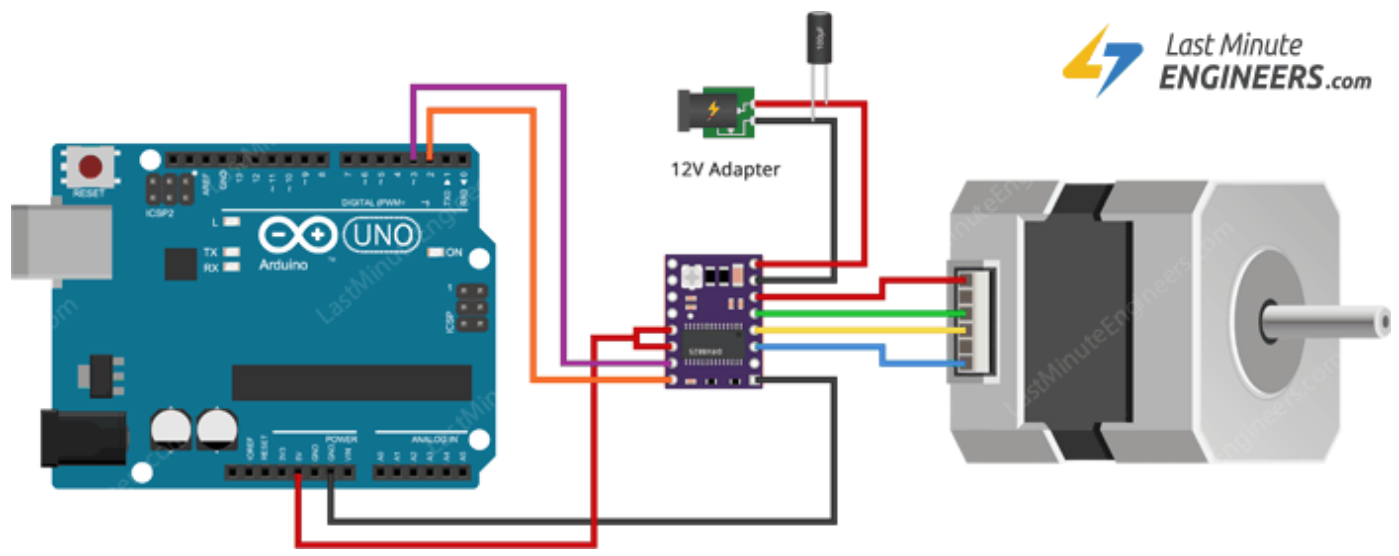
Connect GND LOGIC pin to the ground pin on the Arduino. DIR and STEP input pins are connected to #2 & #3 digital output pins on Arduino respectively.

connect the stepper motor to the B2, B1, A1 & A2 pins. Actually DRV8825 is conveniently laid out to match the 4-pin connector on several bipolar motors so, that shouldn’t be a problem.

> Warning:
>
> Connecting or disconnecting a stepper motor while the driver is powered can destroy the driver.

Remember to keep the microstep selection pins disconnected to operate the motor in full step mode.

Finally, connect the motor power supply to the VMOT and GND MOT pins. Remember to put a large 100µF decoupling electrolytic capacitor across motor power supply pins, close to the board.



Wiring Nema 17 Stepper Motor to DRV8825 driver & Arduino

Arduino Code – Basic Example

The following sketch will give you complete understanding on how to control speed and spinning direction of a bipolar stepper motor with DRV8825 stepper motor driver and can serve as the basis for more practical experiments and projects.

```
// Define pin connections & motor's steps per revolution
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 200;

void setup()
{
    // Declare pins as Outputs
    pinMode(stepPin, OUTPUT);
    pinMode(dirPin, OUTPUT);
}
void loop()
{
    // Set motor direction clockwise
    digitalWrite(dirPin, HIGH);

    // Spin motor slowly
    for(int x = 0; x < stepsPerRevolution; x++)
    {
        digitalWrite(stepPin, HIGH);
        delayMicroseconds(2000);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(2000);
    }
    delay(1000); // Wait a second

    // Set motor direction counterclockwise
```

Code Explanation:

The sketch starts with defining Arduino pins to which DRV8825's STEP & DIR pins are connected. We also define `stepsPerRevolution`. Set this to match your stepper motor specifications.

```
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 200;
```

In setup section of code, all the motor control pins are declared as digital OUTPUT.

```
pinMode(stepPin, OUTPUT);
pinMode(dirPin, OUTPUT);
```

In loop section we spin the motor clockwise slowly and then spin it counterclockwise quickly at an interval of a second.

Control Spinning Direction: To control the spinning direction of a motor we set the DIR pin either HIGH or LOW. A HIGH input spins the motor clockwise and a LOW will spin it counterclockwise.

```
digitalWrite(dirPin, HIGH);
```

Control Speed: The speed of a motor is determined by the frequency of the pulses we send to the STEP pin. The higher the pulses, the faster the motor runs. A pulses is nothing but pulling the output HIGH, waiting a bit then pulling it LOW and waiting again. By changing the delay between two pulses, you change the frequency of those pulses and hence the speed of a motor.

```
for(int x = 0; x < stepsPerRevolution; x++) {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin, LOW);
```

```
    delayMicroseconds(1000);
}
```

Arduino Code – Using AccelStepper library

Controlling the stepper without a library is perfectly fine for simple, single motor applications. But when you want to control multiple steppers, you'll need a library.

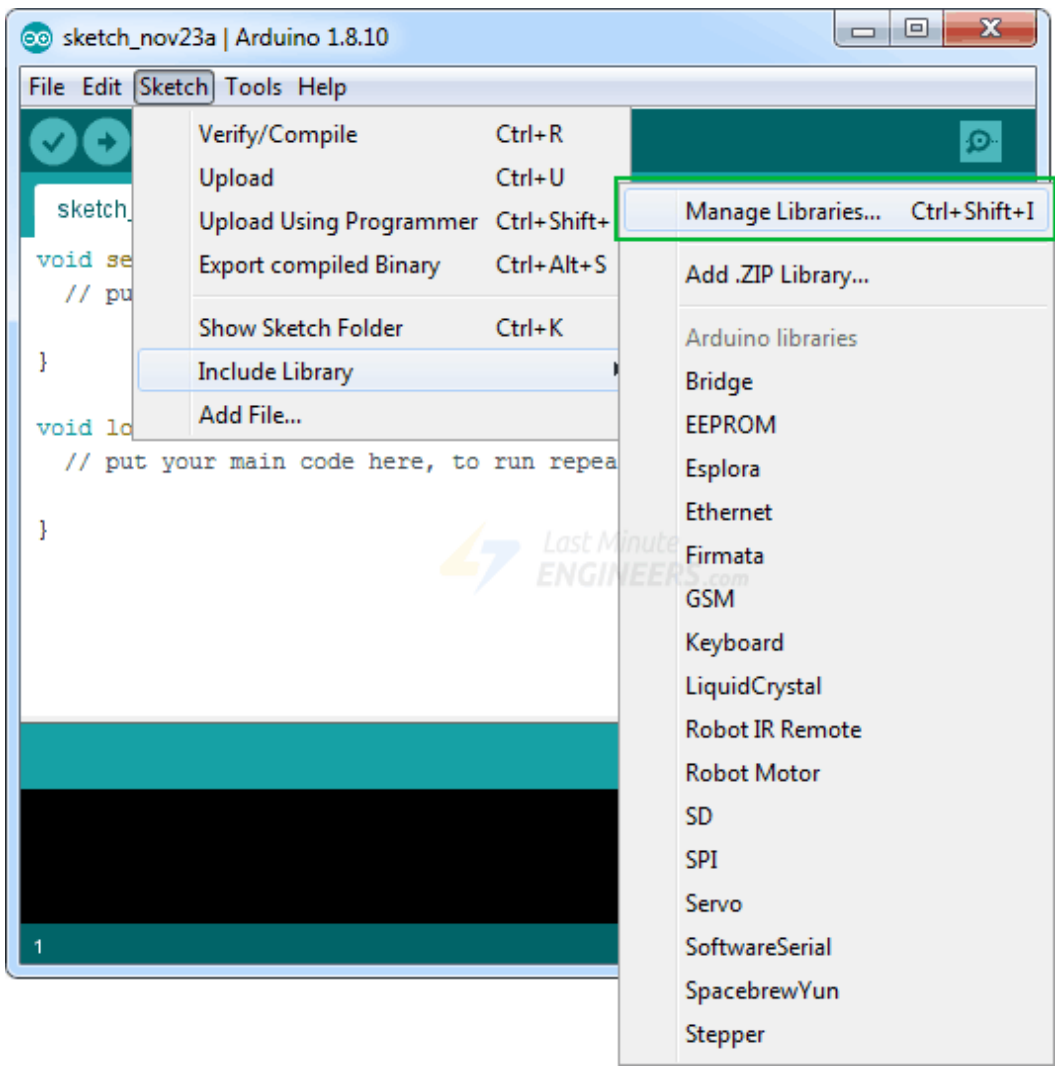
So, for our next experiment we will make use of an advanced stepper motor library called AccelStepper library. It supports:

- Acceleration and deceleration.
- Multiple simultaneous steppers, with independent concurrent stepping on each stepper.

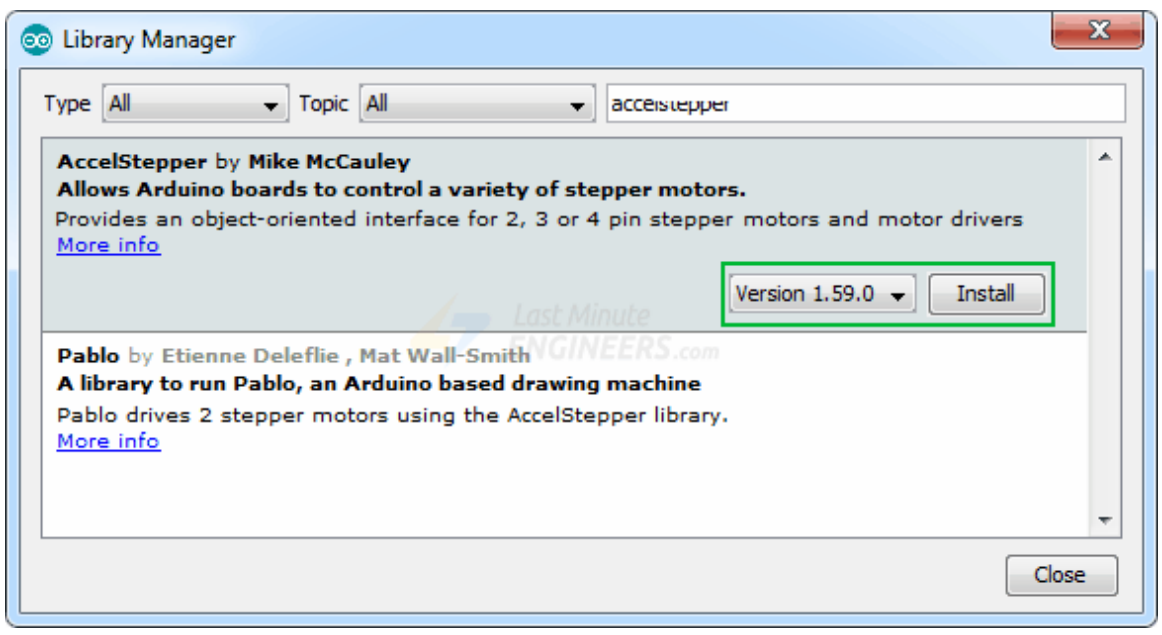
This library is not included in the Arduino IDE, so you will need to install it first.

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 pin connections
const int dirPin = 2;
const int stepPin = 3;

// Define motor interface type
#define motorInterfaceType 1

// Creates an instance
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);

void setup() {
    // set the maximum speed, acceleration factor,
    // initial speed and the target position
    myStepper.setMaxSpeed(1000);
    myStepper.setAcceleration(50);
    myStepper.setSpeed(200);
    myStepper.moveTo(200);
}

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

Code Explanation:

We start off by including the newly installed AccelStepper library.

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

We define Arduino pins to which DRV8825's STEP & DIR pins are connected. We also set `motorInterfaceType` to 1. (1 means an external stepper driver with Step and Direction pins)

```
// Define pin connections
const int dirPin = 2;
const int stepPin = 3;

// Define motor interface type
#define motorInterfaceType 1
```

Next, we create an instance of stepper library called `myStepper`.

```
AccelStepper myStepper(motorInterfaceType, stepPin, dirPin);
```

In the setup function we first set the maximum speed of the motor 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. 200 (as NEMA 17 moves 200 steps per revolution).

```
void setup() {
    myStepper.setMaxSpeed(1000);
    myStepper.setAcceleration(50);
    myStepper.setSpeed(200);
    myStepper.moveTo(200);
}
```

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() {
    if (myStepper.distanceToGo() == 0)
        myStepper.moveTo(-myStepper.currentPosition());

    myStepper.run();
}
```