

# Lab 7: Microcontroller Motor Interfacing

## BMEEn 2151 “Introductory Medical Device Prototyping”

Prof. Steven S. Saliterman

### Exercise 7.1: Using the Adafruit Motor Shield Operate Different Motors

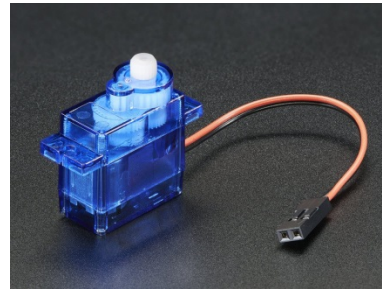
Objective: To become acquainted with DC, stepping and servo motors.

#### Items Used in this Lab

Here are some additional items to be used for this lab. This lab will demonstrate interfacing Arduino Uno with three different kinds of motors using the Adafruit motor shield.



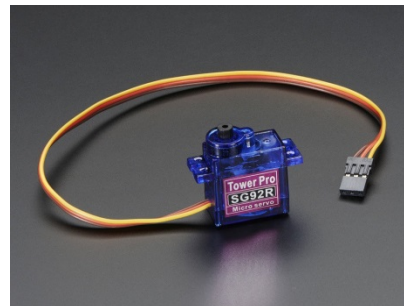
Motor Shield



DC Motor with Gear



Stepper Motor



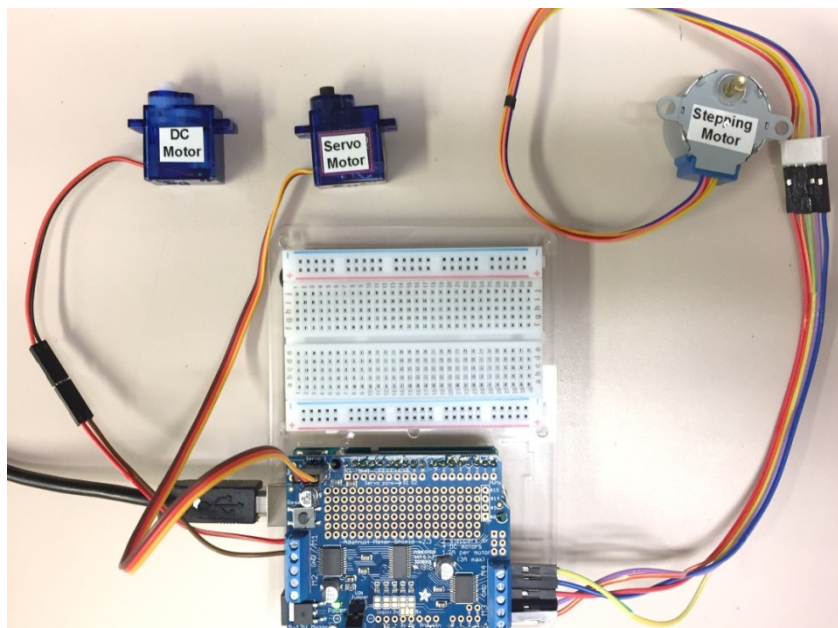
Servo Motor

#### Task

1. Obtain the Adafruit Motor Shield from the Project Box. This board contains the necessary PWM chip and motor drivers to operate a variety of motors. You can learn more about this board at the following link: [Adafruit MotorShield](#)

These are the features of the Adafruit Motor Shield:

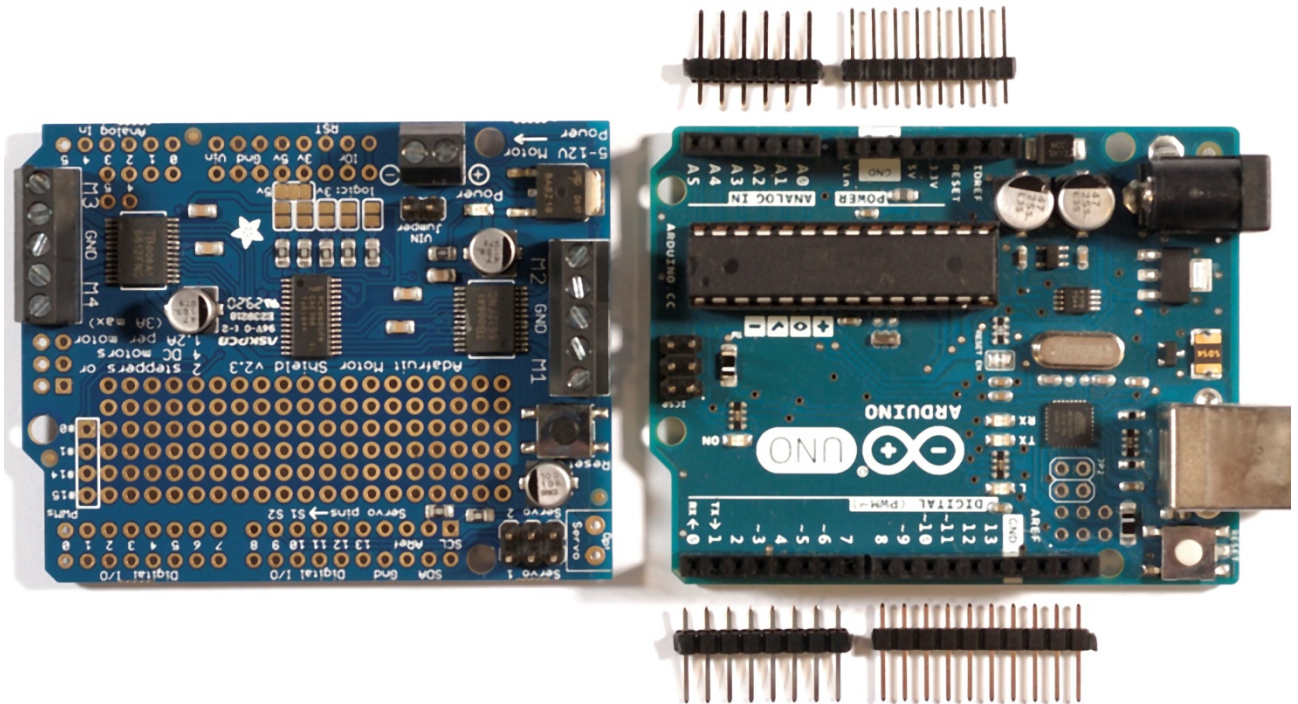
- Dedicated PWM driver chip onboard. This chip handles all the motor and speed controls over I<sup>2</sup>C.
- Only two pins (SDA & SCL) are required to drive the multiple motors, and since it's I<sup>2</sup>C you can also connect any other I<sup>2</sup>C devices or shields to the same pins. This also makes it drop-in compatible with any Arduino, such as the Uno, Due, Leonardo and Mega R3.
- Completely stackable design: 5 address-select pins means up to 32 stackable shields: that's 64 steppers or 128 DC motors.
- 4 H-Bridges: TB6612 chipset provides 1.2A per bridge (3A for brief 20ms peaks) with thermal shutdown protection, internal kickback protection diodes. Can run motors on 4.5VDC to 13.5VDC.
- Up to 4 bi-directional DC motors with individual 8-bit speed selection (so, about 0.5% resolution).
- Up to 2 stepper motors (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping.
- Motors automatically disabled on power-up.
- Big terminal block connectors to easily hook up wires (18-26AWG) and power
- Arduino reset button brought up top.
- Polarity protected 2-pin terminal block and jumper to connect external power, for separate logic/motor supplies.



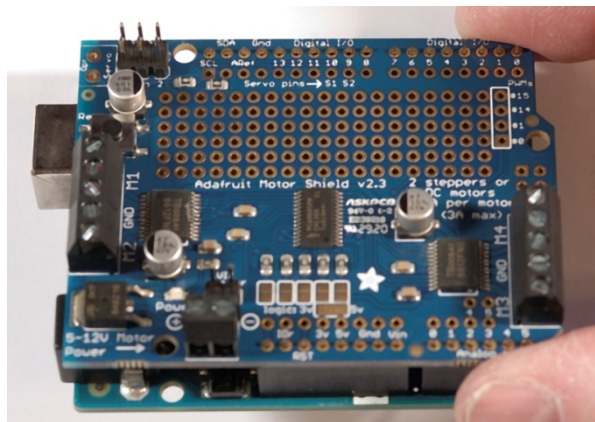
Motor connections. A DC motor, servo motor and stepping motor will all be connected to the motor shield stacked with the Arduino

## 2. Assemble the motor shield interconnecting pins.

If the Adafruit motor Shield is still in its pink bubble wrap unassembled, it will be necessary to solder to it the Arduino Uno interconnecting pins.



Using your wire cutter, snip the header pins into sections as shown above – just enough pins to match the corresponding female connectors on the Arduino Uno. Next place long end of the header pins into the matching Arduino Uno female connector. Place the motor shield on top of the short pins, squeeze together, and solder the short pins to the motor shield. The two boards may be pulled apart now – or continue to the following exercise.



## 3. Take photographs of your setup.

## Exercise 7.2 Connect Boards and Install Software

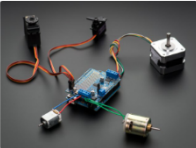
1. If not already done, carefully stack your Arduino to the motor shield. You will be powering the motors from your computer USB port. You will next need to download the Adafruit Motor Shield library (these procedures might change over time).

2. Go to the following website: <https://learn.adafruit.com/adafruit-motor-shield-v2-for-arduino/install-software> and review the page.

Next follow the link to GitHub:

[https://github.com/adafruit/Adafruit\\_Motor\\_Shield\\_V2\\_Library](https://github.com/adafruit/Adafruit_Motor_Shield_V2_Library).

Adafruit Motor Shield V2 > Install Software



### Install Software

Install Adafruit Motor Shield V2 library

To use the shield on an Arduino, you'll need to install the Adafruit Motorshield v2 library. **This library is not compatible with the older AF\_Motor library used for v1 shields.** However, if you have code for the older shield, adapting the code to use the new shield isn't difficult. We had to change the interface a little to support shield stacking, & we think it's worth it!

To begin controlling motors, you will need to [install the Adafruit\\_Motor\\_Shield\\_V2\\_Library library \(code on our github repository\)](#). It is available from the Arduino library manager so we recommend using that.

From the IDE open up the library manager...

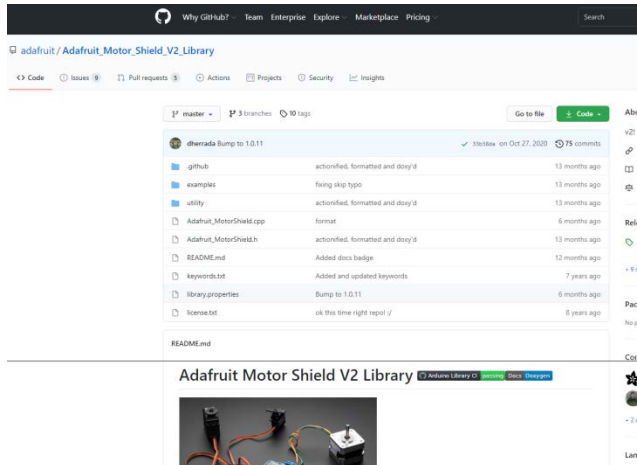


And type in **adafruit motor** to locate the library. Click **Install**

Stackable, high current DC and Stepper motor shield

By lady\_ada

Overview  
FAQ  
Install Headers & Terminals  
- Install Software  
Library Reference  
Arduino Library Docs



adafruit / Adafruit\_Motor\_Shield\_V2\_Library

Code Issues Pull requests Actions Projects Security Insights


master 3 branches 10 tags

ghemada Bump to 1.0.11 ✓ Merged on Oct 27, 2020 75 commits

File	Commit	Age
github	actionified, formatted and dosy'd	13 months ago
examples	fixing skip typo	13 months ago
utility	actionified, formatted and dosy'd	13 months ago
Adafruit_MotorShield.cpp	format	6 months ago
Adafruit_MotorShield.h	actionified, formatted and dosy'd	13 months ago
README.md	Added docs badge	13 months ago
keywords.txt	Added and updated keywords	7 years ago
library.properties	Bump to 1.0.11	6 months ago
license.txt	ok this time right lol! /	8 years ago

README.md

### Adafruit Motor Shield V2 Library



Select the green “download code” button (upper right), select “download ZIP” and save to your desktop.

3. Open your Arduino IDE. Under the menu item “Sketch”, select “Include Library”, then “Add .ZIP Library...”. Find your downloaded ZIP file from your desktop and select it.

Update April 14, 2022:

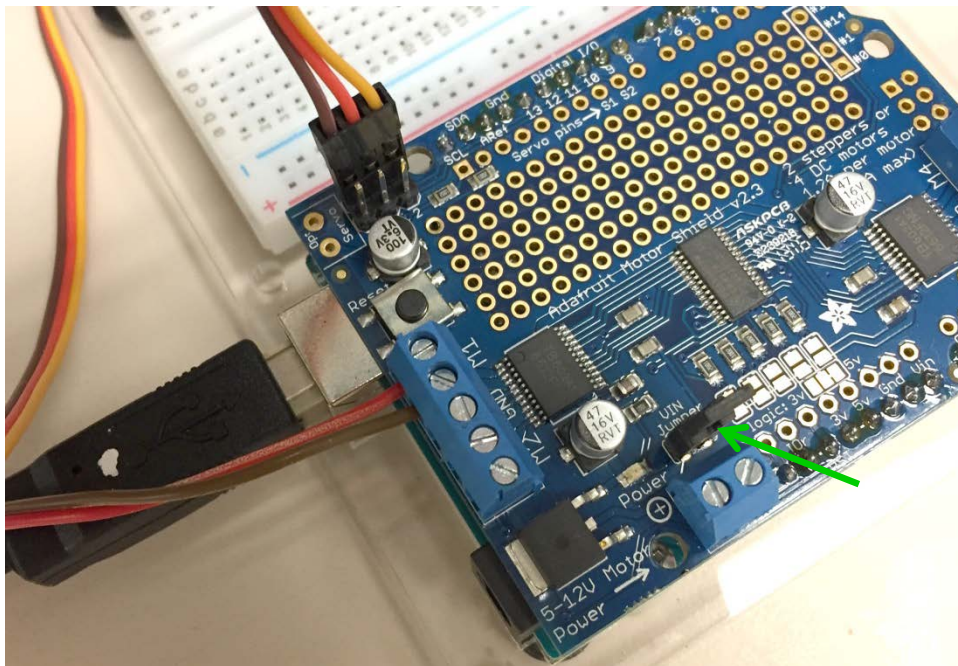
- Go to the menu item “Tools” and select “Manage Libraries.”
- In the dialog that appears, type in the search box: “Adafruit V2”.
- Scroll through the list to “Adafruit Motor Shield V2 Library” and click “Install All.”

If you now go back to “Include Library”, and scroll to the bottom, you should see the file “Adafruit Motor Shield V2 Library” listed.

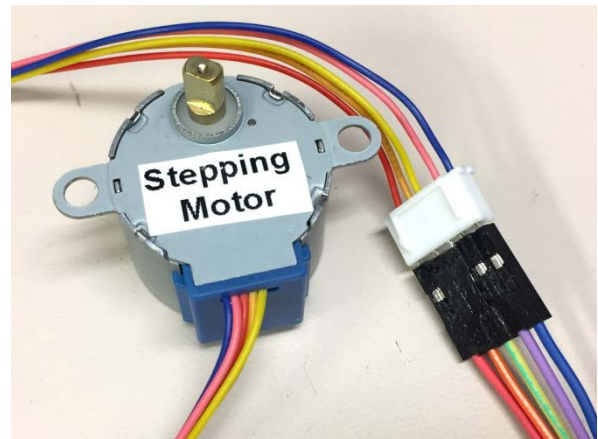
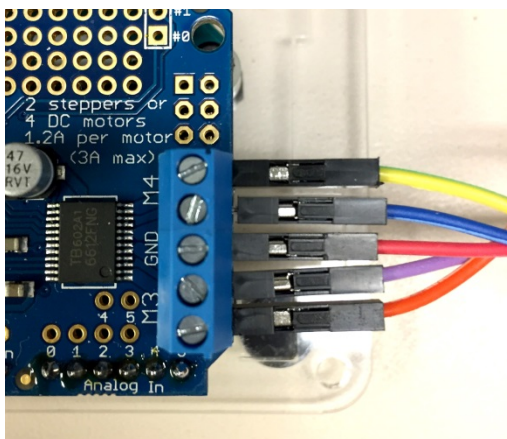


4. Copy and paste the code in your Arduino IDE for Exercise 7.2 from the Word document available on our course website under “Additional Lab Items/Program Code for Labs 6 & 7”: [Motor Party Code](#).

5. From past experience it is better to connect the DC motor first then servo motor and then the stepping motor, testing each motor first as it is added. Disconnect the USB cable from the computer when attaching each motor. You may need to connect only one motor at a time because of limited power available from your computer. You may also be able to quickly change from the computer USB power to the Arduino power adaptor, then reset the Arduino, and all the motors should run.



Connection of the DC motor (foreground), servo motor (upper left), USB A-B computer cable (left) and jumper cap (green power light should be ON).



Connection of the servo motor using Dupont male-male jumper leads. Order is important: Orange – Purple (for Pink) - Red – Blue - Yellow at the Arduino.

6. Study the following Adafruit “MotorParty” sketch until you understand each function (see Appendix: Library Interface):

```

/*
This is a test sketch for the Adafruit assembled Motor Shield for Arduino v2
It won't work with v1.x motor shields! Only for the v2's with built in PWM
control

For use with the Adafruit Motor Shield v2
---->      http://www.adafruit.com/products/1438

This sketch creates a fun motor party on your desk *whiirrr*
Connect a unipolar/bipolar stepper to M3/M4
Connect a DC motor to M1
Connect a hobby servo to SERVO1
*/

#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWM_ServoDriver.h"
#include <Servo.h>

// Create the motor shield object with the default I2C address
Adafruit_MotorShield AFMS = Adafruit_MotorShield();
// Or, create it with a different I2C address (say for stacking)
// Adafruit_MotorShield AFMS = Adafruit_MotorShield(0x61);

// Connect a stepper motor with 200 steps per revolution (1.8 degree)
// to motor port #2 (M3 and M4)
Adafruit_StepperMotor *myStepper = AFMS.getStepper(200, 2);
// And connect a DC motor to port M1
Adafruit_DCMotor *myMotor = AFMS.getMotor(1);

// We'll also test out the built in Arduino Servo library
Servo servo1;

void setup() {
  Serial.begin(9600);      // set up Serial library at 9600 bps
  Serial.println("MMMMotor party!");

  AFMS.begin(); // create with the default frequency 1.6KHz
  //AFMS.begin(1000); // OR with a different frequency, say 1KHz

```

```

// Attach a servo to pin #10
servo1.attach(10);

// turn on motor M1
myMotor->setSpeed(200);
myMotor->run(RELEASE);

// setup the stepper
myStepper->setSpeed(10); // 10 rpm
}

int i;
void loop() {
  myMotor->run(FORWARD);
  for (i=0; i<255; i++) {
    servo1.write(map(i, 0, 255, 0, 180));
    myMotor->setSpeed(i);
    myStepper->step(1, FORWARD, INTERLEAVE);
    delay(3);
  }

  for (i=255; i!=0; i--) {
    servo1.write(map(i, 0, 255, 0, 180));
    myMotor->setSpeed(i);
    myStepper->step(1, BACKWARD, INTERLEAVE);
    delay(3);
  }

  myMotor->run(BACKWARD);
  for (i=0; i<255; i++) {
    servo1.write(map(i, 0, 255, 0, 180));
    myMotor->setSpeed(i);
    myStepper->step(1, FORWARD, DOUBLE);
    delay(3);
  }

  for (i=255; i!=0; i--) {
    servo1.write(map(i, 0, 255, 0, 180));
    myMotor->setSpeed(i);
    myStepper->step(1, BACKWARD, DOUBLE);
    delay(3);
  }
}
}

```

7. Take photographs of your setup.





Here is a solution to (1) above - using a potentiometer to change the speed of a DC motor (thank you Theodore Kakacek).

```
#include <Adafruit_MotorShield.h>
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_MS_PWMServoDriver.h"
#include <Servo.h>
const int analogPin = A0;

Adafruit_MotorShield AFMS = Adafruit_MotorShield(); // motor shield object
Adafruit_DCMotor *myMotor = AFMS.getMotor(1); // DC motor

void setup()
{
  AFMS.begin();
  myMotor -> setSpeed(200);
  myMotor -> run(RELEASE);
}

void loop()
{
  myMotor->run(FORWARD);
  int analogValue = analogRead(analogPin);
  analogValue = map(analogValue, 0, 1023, 0, 255);
  myMotor->setSpeed(analogValue);
  delay(3);
}
```

## Background Information on the Library Interface (Courtesy of Adafruit)

### **class Adafruit\_MotorShield;**

The Adafruit\_MotorShield class represents a motor shield and must be instantiated before any DCMotors or StepperMotors can be used. You will need to declare one Adafruit\_MotorShield for each shield in your system.

### **Adafruit\_MotorShield(uint8\_t addr = 0x60);**

The constructor takes one optional parameter to specify the i2c address of the shield. The default address of the constructor (0x60) matches the default address of the boards as shipped. If you have more than one shield in your system, each shield must have a unique address.

### **void begin(uint16\_t freq = 1600);**

begin() must be called in setup() to initialize the shield. An optional frequency parameter can be used to specify something other than the default maximum: 1.6KHz PWM frequency.

### **Adafruit\_DCMotor \*getMotor(uint8\_t n);**

This function returns one of 4 pre-defined DC motor objects controlled by the shield. The parameter specifies the associated motor channel: 1-4.

### **Adafruit\_StepperMotor \*getStepper(uint16\_t steps, uint8\_t n);**

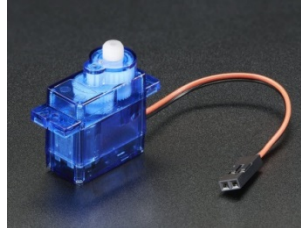
This function returns one of 2 pre-defined stepper motor objects controlled by the shield. The first parameter specifies the number of steps per revolution.

The second parameter specifies the associated stepper channel: 1-2.

### **void setPWM(uint8\_t pin, uint16\_t val);**

### **void setPin(uint8\_t pin, boolean val);**

These are low-level functions to control pins on the on-board PWM driver chip. These functions are intended for internal use only.



### **class Adafruit\_DCMotor**

The Adafruit\_DCMotor class represents a DC motor attached to the shield. You must declare an Adafruit\_DCMotor for each motor in your system.

#### **Adafruit\_DCMotor(void);**

The constructor takes no arguments. The motor object is typically initialized by assigning a motor object retrieved from the shield class as below:

- // Create the motor shield object with the default I2C address  
Adafruit\_MotorShield AFMS = Adafruit\_MotorShield();
- // Select which 'port' M1, M2, M3 or M4. In this case, M1  
Adafruit\_DCMotor \*myMotor = AFMS.getMotor(1);
- // You can also make another motor on port M2  
Adafruit\_DCMotor \*myOtherMotor = AFMS.getMotor(2);

#### **void run(uint8\_t);**

The run() function controls the motor state. The parameter can have one of 3 values:

- FORWARD - Rotate in a forward direction
- REVERSE - Rotate in the reverse direction
- RELEASE - Stop rotation

Note that the "FORWARD" and "REVERSE" directions are arbitrary. If they do not match the actual direction of your vehicle or robot, simply swap the motor leads.

Also note that "RELEASE" simply cuts power to the motor. It does not apply any braking.

#### **void setSpeed(uint8\_t);**

The setSpeed() function controls the power level delivered to the motor. The speed parameter is a value between 0 and 255.

Note that setSpeed just controls the power delivered to the motor. The actual speed of the motor will depend on several factors, including: The motor, the power supply and the load.



### **class Adafruit\_StepperMotor**

The Adafruit\_StepperMotor class represents a stepper motor attached to the shield. You must declare an Adafruit\_StepperMotor for each stepper motor in your system.

#### **Adafruit\_StepperMotor(void);**

The constructor takes no arguments. The stepper motor is typically initialized by assigning a stepper object retrieved from the shield as below:

- `// Create the motor shield object with the default I2C address`  
`Adafruit_MotorShield AFMS = Adafruit_MotorShield();`
- `// Connect a stepper motor with 200 steps per revolution (1.8 degree)`
- `// to motor port #2 (M3 and M4)`  
`Adafruit_StepperMotor *myMotor = AFMS.getStepper(200, 2);`

#### **void step(uint16\_t steps, uint8\_t dir, uint8\_t style = SINGLE);**

The step() function controls stepper motion.

- The first parameter specifies how many steps to move.
- The second parameter specifies the direction: FORWARD or BACKWARD
- The last parameter specifies the stepping style: SINGLE, DOUBLE, INTERLEAVED or MICROSTEP

The ste() function is synchronous and does not return until all steps are complete. When complete the motor remains powered to apply "holding torque" to maintain position.

#### **void setSpeed(uint16\_t);**

The setSpeed() function controls the speed of the stepper motor rotation. Speed is specified in RPM.

#### **uint8\_t onestep(uint8\_t dir, uint8\_t style);**

The oneStep() function is a low-level internal function called by step(). But it can be useful to call on its own to implement more advanced functions such as acceleration or coordinating simultaneous movement of multiple stepper motors. The direction and style parameters are the same as for step(), but onestep() steps exactly once.

Note: Calling step() with a step count of 1 is not the same as calling onestep(). The step function has a delay based on the speed set in setSpeed(). onestep() has no delay.

#### **void release(void);**

The release() function removes all power from the motor. Call this function to reduce power requirements if holding torque is not required to maintain position.

