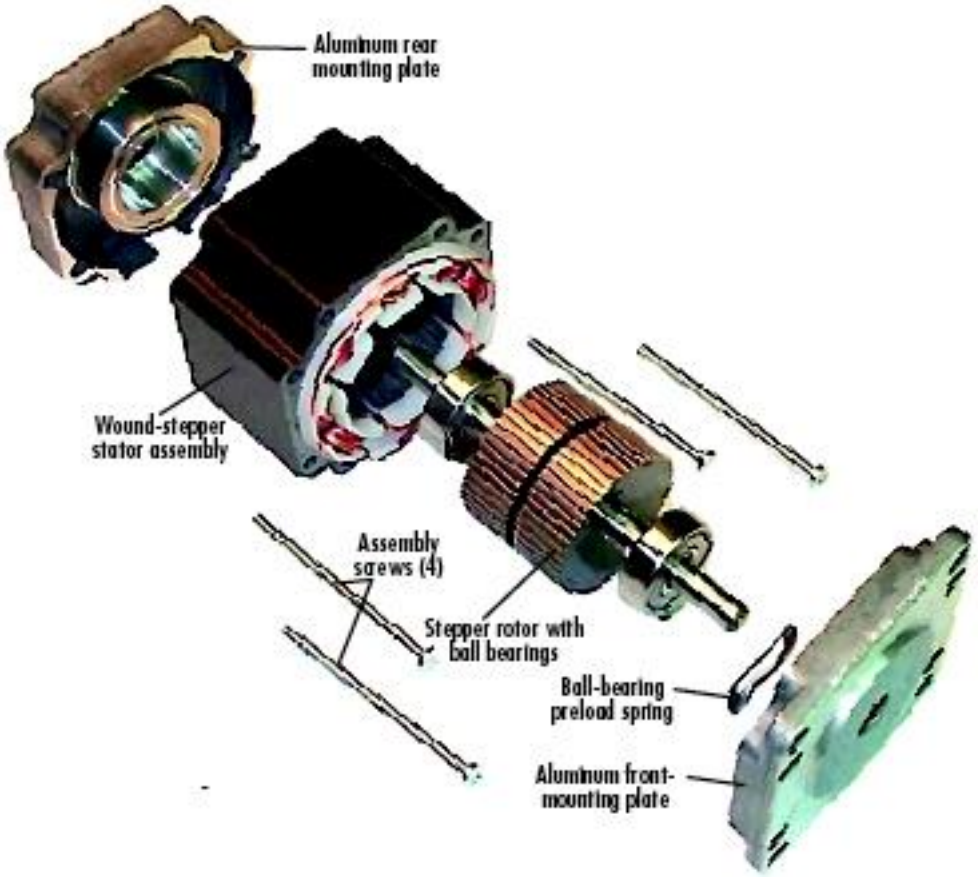
Stepper Motors:

This document was written for the purpose of familiarizing you with stepper motors and subsequently how to program their operation into Arduino. Arduino is a prototyping program that allows the hobbyist or student level user to perform operations of more advanced circuits or microchips through the use of an easy to use and modular computer code. Though the code is simple it can still be challenging to many who are not familiar with computer code or the devices that a computer code can control.

A stepper motor, much like a DC motor has a rotating permanent magnet propelled by stationary electrical magnets, however the motion is divided into a number of steps around the rotation of the rotating magnet. It does so by having several teeth on the rotating magnet that line up with specific locations around the stationary charged magnets. When voltage is supplied to a specific magnet or specific sequence of magnets the motor will rotate, or step to that position and hold.

Stepper motors can spin like a regular DC motor, however they can also stop on a position like a servo motor. This gives them many uses. Some basic places where you will find stepper motors are in disc drives, printers, faxes, slot machines, clocks, intelligent lighting, and automotive gauges.

There are also some differences you will see internally on a stepper motor as opposed to a regular DC brushless motor. One main feature is the stepped rotor as opposed to a smooth magnetic rotor as you can see in the picture:



This is what actually gives the motor its 'steps'. By charging and releasing magnets on the stator in sequence you can get the position of the rotor to line up on the individual steps around a single revolution of the motors movement. Each ridge on the stepper rotor is an individual step and the number of steps a motor is capable of is the number of ridges around the circumference of the rotor.

This creates a difference in how a stepper motor is controlled vs a DC motor. While a stepper motor can accept straight current to turn in either direction this is not its most effective mode. If all you need is for the motor to turn a DC motor would be better as the steps on a stepper motors rotor give less magnetic area and therefore a stepper motor of similar size to a DC motor will have less power. But as stated a stepper motor can stop and hold a position.

There are a few more things that need to be known about stepper motors before we can jump into the programming side. A stepper motors measurement is not only is torq production or max RPM:

1. Pull in Torq: This is the torq provided by the motor when it is holding a position while power is being applied. This is how powerfully the motor can hold a step while it us under power. This is useful when a stepper motor is holding a spring loaded device like a small catapult or a an arm on a robot when it is pre-loaded.
2. Pull out Torq: This is the amount of torq that is needed to make the motor stop or miss steps when it is rotating under power. This is useful to know if you plan on using the motion of this motor against a force. A good example of this would be in a CNC machine or on a large tooling lathe.
3. Detent Torq: This is the resistance torq provided by the motor when it is not under any charge. This is a measure of the magnetic force inherent in the motor and not a measure of friction of any bearings in the device or the force it takes to move the mass of the rotor alone. This is important to know, however know that the detent torq will be negligible on motors with soft iron cores. This knowledge is good to have so you are aware of how much power it will take to move the motor when it is not under power. This is important in devices where several motors will be connected a single device and movement will be based on which motor is running. Or in any spring loaded device where you depend on a spring working against the motor to make the device return to its original position.

As stated above, because a stepper motor is not most efficient running off of a standard motor code other commands have been built into programs in order to operate a stepper motor properly. Arduino has special commands to operate stepper motors. On the following page is an example of a stepper motor program that can be found on Arduinos website here: http://www.arduino.cc/en/Tutorial/Stepper .

Read over this code. Once you have read over it a few times and have familiarized yourself with some of the functions I will go into some of the individual codes and explain how they are implemented into the operation of a stepper motor

```
/*

 Stepper Motor Controller
 language: Wiring/Arduino

 This program drives a unipolar or bipolar stepper motor.
 The motor is attached to digital pins 8 and 9 of the Arduino.

 The motor moves 100 steps in one direction, then 100 in the other.

 Created 11 Mar. 2007
 Modified 7 Apr. 2007
 by Tom Igoe

 */

// define the pins that the motor is attached to. You can use
// any digital I/O pins.

#include <Stepper.h>

#define motorSteps 200
#define motorPin1 8
#define motorPin2 9
#define ledPin 13

// initialize of the Stepper library:
Stepper myStepper(motorSteps, motorPin1,motorPin2);

void setup() {
  // set the motor speed at 60 RPMS:
  myStepper.setSpeed(60);

  // Initialize the Serial port:
  Serial.begin(9600);

  // set up the LED pin:
  pinMode(ledPin, OUTPUT);
  // blink the LED:
  blink(3);
}

void loop() {
  // Step forward 100 steps:
  Serial.println("Forward");
  myStepper.step(100);
  delay(500);

  // Step backward 100 steps:
  Serial.println("Backward");
  myStepper.step(-100);
  delay(500);

}


// Blink the reset LED:
```

```
void blink(int howManyTimes) {
  int i;
  for (i=0; i< howManyTimes; i++) {
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
    delay(200);
  }
}
```

The first place you can see a definite signal of servo motor usage is in the first pin command lines:

```
#include <Stepper.h>

#define motorSteps 200
#define motorPin1 8
#define motorPin2 9
#define ledPin 13
```

 # include <Stepper.h> initializes the stepper library in the Arduino.

#define motorSteps 200 is what tells the Arduino how many steps to expect out of the stepper motor. This is where you would input how many steps your specific stepper motor has.

The next three lines specify where the signal will be going on the Arduino and *ledPin* is an output for an indicator LED that will be specified later in the code. The LED acts as an indicator only and is not a function of the stepper motor.

```
void setup() {
  // set the motor speed at 60 RPMS:
  myStepper.setSpeed(60);

  // Initialize the Serial port:
  Serial.begin(9600);

  // set up the LED pin:
  pinMode(ledPin, OUTPUT);
  // blink the LED:
  blink(3);
}
```

void setup () then initializes the code that will control the speed of the motor, serial port, and the blinking of the LED.

As you can see in this case *myStepper.setSpeed(60) ;* controls the speed at which the motor will operate in RPMs or Revolutions Per Minute. In this portion of the code you can control the speed of the motor.

From there a serial port is initialized so that outputs of the function can be read and the indicator LED is set up.

```
void loop() {
  // Step forward 100 steps:
  Serial.println("Forward");
  myStepper.step(100);
  delay(500);

  // Step backward 100 steps:
  Serial.println("Backward");
  myStepper.step(-100);
  delay(500);

}
```

In this loop we begin to see the stepper motor receiving commands. *myStepper.step(100) ;* commands the Arduino to make the motor move 100 steps forward. In this line you can change the number of steps or enter a variable based on a sensor input. The following command *delay(500) ;* Then tells the system to wait 500 milliseconds (or .5 seconds) until initiating the next command. From the following command you can now see that making the stepper motor rotate in the opposite direction requires entering a minus sign before the number. i.e. *(-100)*. The purpose of the line *Serial.println( )* is to print out on the serial monitor what the motor is doing at that particular time.

Next is a less necessary function but it can lead into how to form output command from stepper motor operation.

```
void blink(int howManyTimes) {
  int i;
  for (i=0; i< howManyTimes; i++) {
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
    delay(200);
  }
}
```

In this code an intelligent command or *i* is used to indicate how many times the motor has rotated back and forth. I will go further into these commands at a later date as I am still learning the definitions of the i and i++ commands.