Controlling Stepper/DC/Servo Motors with Arduino/NI DAQ/PMAC

1. What's the difference between DC, Servo & Stepper Motors?

1.1. DC Motors

DC (Direct Current) Motors are two wire (power & ground), continuous rotation motors. When you supply power, a DC motor will start spinning until that power is removed. Most DC motors run at a high RPM (revolutions per minute), examples being computer cooling fans, or radio controlled car wheels.

The speed of DC motors is controlled using pulse width modulation (PWM), a technique of rapidly pulsing the power on and off. The percentage of time spent cycling the on/off ratio determines the speed of the motor, e.g. if the power is cycled at 50% (half on, half off), then the motor will spin at half the speed of 100% (fully on). Each pulse is so rapid that the motor appears to be continuously spinning with no stuttering.

1.2. Servo Motors

Servo motors are generally an assembly of four things: a DC motor, a gearing set, a control circuit and a position-sensor (usually a potentiometer).

The position of servo motors can be controlled more precisely than those of standard DC motors, and they usually have three wires (power, ground & control). Power to servo motors is constantly applied, with the servo control circuit regulating the draw to drive the motor. Servo motors are designed for more specific tasks where position needs to be defined accurately such as controlling the rudder on a boat or moving a robotic arm or robot leg within a certain range.

Servo motors do not rotate freely like a standard DC motor. Instead the angle of rotation is limited to 180 Degrees (or so) back and forth. Servo motors receive a control signal that represents an output position and applies power to the DC motor until the shaft turns to the correct position, determined by the position sensor.

PWM is used for the control signal of servo motors. However, unlike DC motors it’s the duration of the positive pulse that determines the position, rather than speed, of the servo shaft. A neutral pulse value dependent on the servo (usually around 1.5ms) keeps the servo shaft in the center position. Increasing that pulse value will make the servo turn clockwise, and a shorter pulse will turn the shaft anticlockwise. The servo control pulse is usually repeated every 20 milliseconds, essentially telling the servo where to go, even if that means remaining in the same position.

When a servo is commanded to move, it will move to the position and hold that position, even if external force pushes against it. The servo will resist from moving out of that position, with the maximum amount of resistive force the servo can exert being the torque rating of that servo.
1.3. Stepper Motors

A stepper motor is essentially a servo motor that uses a different method of motorization. Where a servo motor uses a continuous rotation DC motor and integrated controller circuit, stepper motors utilize multiple toothed electromagnets arranged around a central gear to define position.

Stepper motors require an external control circuit or micro controller (e.g. a Raspberry Pi or Arduino) to individually energize each electromagnet and make the motor shaft turn. When electromagnet ‘A’ is powered it attracts the gear’s teeth and aligns them, slightly offset from the next electromagnet ‘B’. When ‘A’ is switch off, and ‘B’ switched on, the gear rotates slightly to align with ‘B’, and so on around the circle, with each electromagnet around the gear energizing and de-energizing in turn to create rotation. Each rotation from one electromagnet to the next is called a "step", and thus the motor can be turned by precise pre-defined step angles through a full 360 Degree rotation.

Stepper motors are available in two varieties; unipolar or bipolar. Bipolar motors are the strongest type of stepper motor and usually have four or eight leads. They have two sets of electromagnetic coils internally, and stepping is achieved by changing the direction of current within those coils. Unipolar motors, identifiable by having 5, 6 or even 8 wires, also have two coils, but each one has a center tap. Unipolar motors can step without having to reverse the direction of current in the coils, making the electronics simpler. However, because the center tap is used to energize only half of each coil at a time they typically have less torque than bipolar.

The design of the stepper motor provides a constant holding torque without the need for the motor to be powered and, provided that the motor is used within its limits, positioning errors don't occur, since stepper motors have physically pre-defined stations.

1.4. Summary

1.4.1. DC Motors

Fast, continuous rotation motors – Used for anything that needs to spin at a high RPM e.g. car wheels, fans etc.

1.4.2. Servo Motors

Fast, high torque, accurate rotation within a limited angle – Generally a high performance alternative to stepper motors, but more complicated setup with PWM tuning. Suited for robotic arms/legs or rudder control etc.

1.4.3. Stepper Motors
Slow, precise rotation, easy set up & control – Advantage over servo motors in positional control. Where servos require a feedback mechanism and support circuitry to drive positioning, a stepper motor has positional control via its nature of rotation by fractional increments. Suited for 3D printers and similar devices where position is fundamental.

2. Stepper motor control

2.1. About stepper motor

A stepper motor (or step motor) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any feedback sensor (an open-loop controller), as long as the motor is carefully sized to the application.

The stepper motor is known by its important property to convert a train of input pulses i.e. a square wave pulse into a precisely defined increment in the shaft position. Each pulse moves the shaft through a fixed angle. Stepper motors effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external control circuit, such as a microcontroller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

stepper motors
2.2. Two phase stepper motor

Two phase stepper motors are most commonly used. Usually such a motor has 4/5/6 wires, with different colors. In any case, we need only connect the 4 wires namely A+, A-, B+, and B-(normally responding to red, green, yellow, blue or black, green, red, blue). A+ and A- (B+ and B-) are conducted to each other, and thus make a wire pair. If we do not know which is which, we can test the conductivity between two wires using a universal meter. Interchange between A+ and A- (B+ and B-) will just change the direction of the motor motion.
2.3. Control System Overview

A stepper motor system consists of three basic elements, often combined with some type of user interface (host computer, PLC or dumb terminal):

- **Indexers** - The indexer (or controller) is a microprocessor capable of generating step pulses and direction signals for the driver. In addition, the indexer is typically required to perform many other sophisticated command functions.

- **Drivers** - The driver (or amplifier) converts the indexer command signals into the power necessary to energize the motor windings. There are numerous types of drivers, with different voltage and current ratings and construction technology. Not all drivers are suitable to run all motors, so when designing a motion control system the driver selection process is critical.

- **Stepper motors** - The stepper motor is an electromagnetic device that converts digital pulses into mechanical shaft rotation. Advantages of step motors are low cost, high reliability, high torque at low speeds and a simple, rugged construction that operates in almost any environment. The main disadvantages in using a stepper motor is the resonance effect often exhibited at low speeds and decreasing torque with increasing speed.
2.4. Drivers for stepper motors

The driver (or amplifier) converts the indexer command signals into the power necessary to energize the motor windings. Usually the driver requires two/three signals: pulse, direction, and enable control (this signal can be left unconnected if not used). Pulse signal is a sequence of pulses whose frequency determines the moving speed of the motor. The relationship is

\[
\text{motion speed} = \text{pulse frequency} \times 60 / (200 \times \text{subdivision number}) \text{ (r/min)}.
\]

Direction signal is a level signal usually at either +5V or 0V. Change of this signal will turn the moving direction of the motor.

A typical control system is shown as below.

2.5. Controlling stepper motor with Arduino
Based on the introduction above, we can see to control a stepper motor with a driver, we need only provide a pulse signal and a direction signal. These can be simply generated by the output pins of Arduino.

Example Codes:

```c
void setup()
{
    //set pin 8 and pin 9 as output channel
    pinMode(8,OUTPUT);  //pulse
    pinMode(9,OUTPUT);  //direction
    //set begining direction
    digitalWrite(8,HIGH);
}

int t=1;
int i;

void loop()
{
    if(t)  //repeat only if t!=0
    {
        for(i=1;i<=50;i++)  //rotate for 90 degrees (50 intervals)
    
```
{  
    // giving one pulse for every 20 microseconds  
    digitalWrite(9,HIGH);  
    delay(10);  
    digitalWrite(9,LOW);  
    delay(10);  
};

digitalWrite(8,LOW); //change direction

for(i=1;i<=50;i++)  //rotate for 90 degrees (50 intervals)
{  
    // giving one pulse for every 20 microseconds  
    digitalWrite(9,HIGH);  
    delay(10);  
    digitalWrite(9,LOW);  
    delay(10);  
}
}

t=0; //set t=0 so it only runs for once

2.6. Controlling stepper motor with NI DAQ

Combined with LabVIEW, NI data acquisition system also provides a highly convenient solution of stepper motor control. The data acquisition card (DAQ) in this lab is the NI DAQ 6218 which has 8 digital output ports able to produce both pulse sequence (for pulse control) and level signal (for direction control).
2.7. Controlling stepper motor with PMAC

The Turbo PMAC2-Eth-Lite controller ("Clipper") from Delta Tau provides a very powerful, but compact and cost-effective, multi-axis controller for cost-sensitive applications. It has a full Turbo PMAC2 CPU section and provides a minimum of 4 axes of servo or stepper control with 32 general-purpose digital I/O points. It provides both Ethernet and RS-232 communications links.

To control a stepper motor with Clipper, we need only connect the system as follows:
3. DC motor control

3.1. About DC motor

DC motor relies on the fact that like magnet poles repel and unlike magnetic poles attract each other. A coil of wire with a current running through it generates an electromagnetic field aligned with the center of the coil. By switching the current on or off in a coil its magnet field can be switched on or off or by switching the direction of the current in the coil the direction of the generated magnetic field can be switched 180°. A simple DC motor typically has a stationary set of magnets in the stator and an armature with a series of two or more windings of wire wrapped in insulated stack slots around iron pole pieces (called stack teeth) with the ends of the wires terminating on a commutator. The armature includes the mounting bearings that keep it in the center of the motor and the power shaft of the motor and the commutator connections. The winding in the armature continues to loop all the way around the armature and uses either single or parallel conductors (wires), and can circle several times around the stack teeth. The total amount of current sent to the coil, the coil's size and what it's wrapped around dictate the strength of the electromagnetic field created. The sequence of turning a particular coil on or off dictates what direction the effective electromagnetic fields are pointed. By turning on and off coils in sequence a rotating magnetic field can be created. These rotating magnetic fields interact with the magnetic fields of the magnets (permanent or electromagnets) in the stationary part of the motor (stator) to create a force on the armature which causes it to
rotate. In some DC motor designs the stator fields use electromagnets to create their magnetic fields which allow greater control over the motor. At high power levels, DC motors are almost always cooled using forced air.

The commutator allows each armature coil to be activated in turn. The current in the coil is typically supplied via two brushes that make moving contact with the commutator. Now, some brushless DC motors have electronics that switch the DC current to each coil on and off and have no brushes to wear out or create sparks.

Different number of stator and armature fields as well as how they are connected provide different inherent speed/torque regulation characteristics. The speed of a DC motor can be controlled by changing the voltage applied to the armature. The introduction of variable resistance in the armature circuit or field circuit allowed speed control. Modern DC motors are often controlled by power electronics systems which adjust the voltage by "chopping" the DC current into on and off cycles which have an effective lower voltage.

Since the series-wound DC motor develops its highest torque at low speed, it is often used in traction applications such as electric locomotives, and trams. The DC motor was the mainstay of electric traction drives on both electric and diesel-electric locomotives, street-cars/trams and diesel electric drilling rigs for many years. The introduction of DC motors and an electrical grid system to run machinery starting in the 1870s started a new second Industrial Revolution. DC motors can operate directly from rechargeable batteries, providing the motive power for the first electric vehicles and today's hybrid cars and electric cars as well as driving a host of cordless tools. Today DC motors are still found in applications as small as toys and disk drives, or in large sizes to operate steel rolling mills and paper machines.

If external power is applied to a DC motor it acts as a DC generator, a dynamo. This feature is used to slow down and recharge batteries on hybrid car and electric cars or to return electricity back to the electric grid used on a street car or electric powered train line when they slow down. This process is called regenerative braking on hybrid and electric cars. In diesel electric locomotives they also use their DC motors as generators to slow down but dissipate the energy in resistor stacks. Newer designs are adding large battery packs to recapture some of this energy.

3.2. Controlling DC motor with Arduino
To control a DC motor is relatively simple. Since the motion of a DC motor is totally determined by the voltage provided (the speed is proportional to the voltage level and the direction is controlled by the polarity), we need only supply such a voltage signal to the motor, which can be generated by many controllers. The problem is power; most motors require a driving current as large as several amperes, while most controllers can only provide voltage signals at mA level. Thus we need a driver to amplify the signal.

The driver shown in the following picture is a compact MC33926 motor driver carrier that can be found from [http://www.pololu.com/product/1212](http://www.pololu.com/product/1212). This breakout board for Freescale’s MC33926 full H-bridge has an operating range of 5 – 28 V and can deliver almost 3 A continuously (5 A peak) to a DC motor. The MC33926 works with 3 – 5 V logic levels, supports ultrasonic (up to 20 kHz) PWM, and features current feedback, under-voltage protection, over-current protection, and over-temperature protection.

In a typical application, five I/O lines are used to connect the motor driver to a microcontroller: the two input lines, IN1 and IN2, for direction control, one of the disable lines, D1 or D2, for PWM speed control, the status flag, SF, for monitoring motor driver errors, and the current sense output, FB, for monitoring motor current draw (connected to an analog-to-digital converter input). The control lines can be reduced to two pins if PWM signals are applied directly to the two input pins with both disable pins held inactive. A two-pin interface can also be achieved using one of the disable lines for PWM speed control and the INV input for direction control with IN1 and IN2 held at different values (i.e. one set HIGH and the other set LOW). In each of these cases, the other unused lines must be set to enable proper operation. For example, if D2 is used for the PWM input (as is typically the case), D1 must be held low to prevent it from disabling the motor driver. The circuit board provides convenient jumper points for overriding the motor driver defaults without having to connect extra wires to the module.

A possible connection is shown as below:

![controlling DC motor with Arduino](image)

3.3. Controlling DC motor with PMAC
When controlling DC motors, we often use the differential DAC analog outputs of PMAC. It is a differential signal that varies between -10V to +10V (the maximum output voltage can be configured less than 10V by software). Similarly, we need another motor driver to amplify the signal. The following is a capable one which can be found at http://item.taobao.com/item.htm?spm=a1z09.2.9.56.D2tWcM&id=19396268376.

The following shows how to connect the system using this driver and PMAC clipper card to control DC motors.

4. Servo motor control

4.1. About servo motor

A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term servomotor is often used to refer to a motor suitable for use in a closed-loop control system. A DC motor with encoder can
be classified as a servomotor; to control this kind of servomotors is similar to control DC motors, except that the feedback should be processed. Another kind of servomotors integrate the feedback processing circuit in themselves, leaving the outer control system much simpler.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

4.2. Controlling DC motor with encoder using PMAC clipper

To control a DC motor with encoder using PMAC clipper is quite simple. Based on the system we have introduced in Section 3.3, we need only additionally bring the feedback signal of the encoder to the PMAC. The connection method is shown as below.

One thing we must notice is that when we use the encoder feedback, we are going to control the motor in close loop. A PID control system is utilized by the PMAC. The PID parameters
should be carefully tuned in order to match the property of the motor and achieve optimal control.

4.3. Controlling servo using Arduino

Servos integrating feedback processing circuits usually have three leads: power (+5V), ground, and control. It is controlled by PWM signal. The duty cycle of the PWM determines the angular position of the servo. Usually the servo motor has a motion range, for example, -90°~90°. A 50% duty cycle corresponds to 0°; 0% corresponds to -90°; and 100% corresponds to 90°. Therefore we need only program the PWM output to control the accurate position of the servo.

Example Codes:
#include<Servo.h>

Servo servo;
int servoPosition = 0;
int t=1;

void setup()
{
  servo.attach(9); //attaches the servo on pin 9 to the servo object
  servo.write(servoPosition); //initialize the servo position to 0
}

void loop()
{
  if(t)
  {
    servoPosition+=90;
    servo.write(servoPosition); // rotate 90 degrees
    delay(1000);
  
}
servoPosition = 90;
servo.write(servoPosition); // rotate -90 degrees to initial position
delay(1000);
}
t = 0; // set t = 0 so it only runs for once
}