Basic DC Motor Circuits

Living with the Lab
Gerald Recktenwald
Portland State University
gerry@pdx.edu

DC Motor Learning Objectives

• Explain the role of a snubber diode
• Describe how PWM controls DC motor speed
• Implement a transistor circuit and Arduino program for PWM control of the DC motor
• Use a potentiometer as input to a program that controls fan speed
What is a snubber diode and why should I care?

Simplest DC Motor Circuit

Connect the motor to a DC power supply

Switch open

Switch closed
Current continues after switch is opened

Opening the switch does not immediately stop current in the motor windings.

Inductive behavior of the motor causes current to continue to flow when the switch is opened suddenly.

Charge builds up on what was the negative terminal of the motor.

Reverse current

Charge build-up can cause damage

Reverse current surge through the voltage supply

Arc across the switch and discharge to ground
Motor Model

Simple model of a DC motor:
- Windings have inductance and resistance
- Inductor stores electrical energy in the windings
- We need to provide a way to safely dissipate electrical energy when the switch is opened

Flyback diode or snubber diode

Adding a diode in parallel with the motor provides a path for dissipation of stored energy when the switch is opened.

The flyback diode allows charge to dissipate without arcing across the switch, or without flowing back to ground through the +5V voltage supply.
Pulse-width modulation (PWM) for DC motor speed control

Controlling DC Motor Speed

The voltage supplied to a DC motor controls its speed

Arduino cannot supply variable DC output
- Arduino lacks a true analog output
- Use Pulse-width modulation (PWM) to simulate a variable DC supply voltage
- PWM is a common technique for supplying variable power levels to “slow” electrical devices such as resistive loads, LEDs, and DC motors
- Arduino Uno has 6 PWM pins: Digital I/O pins 3, 5, 6, 9, 10, and 11
Arduino Uno has 6 PWM pins

Look for the ~ prefix on the digital pin label, e.g. ~3

PWM: Pulsed with modulation

PWM simulates DC voltage control for slow loads

The effective voltage is

\[ V_{\text{eff}} = V_o \frac{\tau_o}{\tau_c} \]

\( \frac{\tau_o}{\tau_c} \) is called the duty cycle
Arduino PWM commands

Configure the output pin:

```c
PWM_pin = ... ;  // one of 3, 5, 6, 9, 10, 11
void setup() {
    pinMode( PWM_pin, OUTPUT);
}
```

Set the duty cycle

```c
void loop() {
    int duty_cycle = 150;  // between 0 and 255
    analogWrite( PWM_pin, duty_cycle );
}
```

The duty cycle is an 8 bit value:

\[ 0 \leq \text{duty\_cycle} \leq 255 \]

Using a transistor to switch the load
Transistor as the switching device

- Each Arduino output channel has a 40 mA limit
- The maximum current draw for an Arduino is 200 mA
- Use Arduino as the brain
- Let another switching element be the brawn

Use an NPN Transistor as a switch

This device is designed for use as a medium power amplifier and switch requiring collector currents up to 500 mA

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCEO</td>
<td>Collector-Emitter Voltage</td>
<td>40 V</td>
<td></td>
</tr>
<tr>
<td>VCEO</td>
<td>Collector-Base Voltage</td>
<td>60 V</td>
<td></td>
</tr>
<tr>
<td>I_Co</td>
<td>Collector Current</td>
<td>600 mA</td>
<td></td>
</tr>
<tr>
<td>T_J, Tst</td>
<td>Operating and Storage Junction Temperature Range</td>
<td>-55 to +150 °C</td>
<td></td>
</tr>
</tbody>
</table>

Thermal Characteristics

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Characteristics</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>R!JC</td>
<td>Thermal Resistance, Junction to Case</td>
<td>83.3 °C/W</td>
<td></td>
</tr>
<tr>
<td>R!JA</td>
<td>Thermal Resistance, Junction to Ambient</td>
<td>200 357 °C/W</td>
<td></td>
</tr>
</tbody>
</table>

This device is designed for use as a medium power amplifier and switch requiring collector currents up to 500 mA.
Electronic components in the fan kit

Replace the Switch with a Transistor

A transistor allows on/off control to be automated and it allows switching of more current than an Arduino digital pin can supply.

Pin 9 or another PWM pin drives the transistor base
Alternative locations for the transistor

Moving the transistor (and any switch) between the power supply and the motor adds a bit of safety by tying the motor to ground when the system is idle.

Diode and transistor orientation

Orient the diode so that the silver stripe is at the same voltage as the positive motor terminal.

Connector: Connect to +5V
Base: Connect to motor control pin on Arduino
Emitter: Connect to positive terminal of motor
Arduino Uno has 5 PWM pins

Look for the ~ prefix on the digital pin label, e.g. ~3

DC Motor Circuit on tiny breadboard
+5V connections

Positive motor lead (brown)  Negative motor lead (ivory)

PWM signal is connected to transistor base

Transistor base (middle pin)  PWM signal
Arduino program to spin the DC Motor

Code is in spin_DC_motor.ino

// spin_DC_motor.ino Use PWM to control DC motor speed
int motorPin = 3; // Pin 3 has PWM, connected it to the DC motor

void setup() {
    pinMode(motorPin, OUTPUT); // Set motor pin to output mode
}

void loop() {
    analogWrite(motorPin, 150); // Motor at 150/255 of full speed
    delay(1000);
    analogWrite(motorPin, 250); // Motor at 250/255 of full speed
    delay(1000);
}

User input to control fan speed
**Adjust fan speed with potentiometer input**

Use the potentiometer circuit from the earlier analog input exercise

---

**Code is in DC_motor_speed_control.ino**

```cpp
// File: DC_motor_speed_control.pde
//
// Use potentiometer input to set the speed of a DC motor
// Output to the motor is PWM

int motorPin = 3;  // pin connected to the DC motor
int potPin = 1;    // analog input connected to the potentiometer

void setup()
{
  pinMode(motorPin, OUTPUT);
}

void loop()
{
  int PWMoutput, potReading;

  potReading = analogRead(potPin);
  PWMoutput = map(potReading, 0, 1023, 0, 255);
  analogWrite(motorPin, PWMoutput);
}
```

---
Adjust fan speed with potentiometer input

```cpp
void loop() {
  int PWMoutput, potReading;
  potReading = analogRead(potPin);
  PWMoutput = map(potReading, 0, 1023, 0, 255);
  analogWrite(motorPin, PWMoutput);
}
```

Each time through the loop:
- Read the voltage at the potentiometer wiper
  - Input value is a 10-bit integer: \(0 \leq \text{potReading} \leq 1023\)
- Scale the 10-bit value (max 1023) to an 8-bit value (max 255)
  - \(\text{PWMoutput} = \text{map}(\text{potReading}, 0, 1023, 0, 255)\)
- Update the PWM signal
  - \(\text{analogWrite}(	ext{motorPin}, \text{PWMoutput})\)