Driving high-current loads with microcontrollers

(SWR 2 Feb 2012)

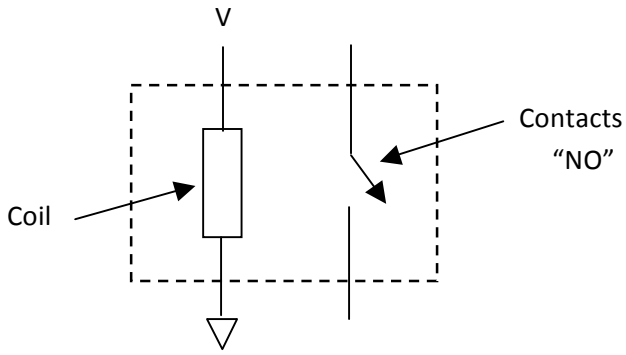
This document briefly describes some methods for using microcontrollers (MCU) like the ATmega328 device on the Arduino board, to drive high-current DC loads such as motors, solenoids, light bulbs, and other high-current devices.

The current source or sink capability for an output or input pin on the -328 is around 40 milliamps (mA). While this is sufficient to power an LED, it is not enough to power devices which require substantially more current in order to operate. The question then becomes one of deciding how to best control higher current devices with a low current microcontroller.

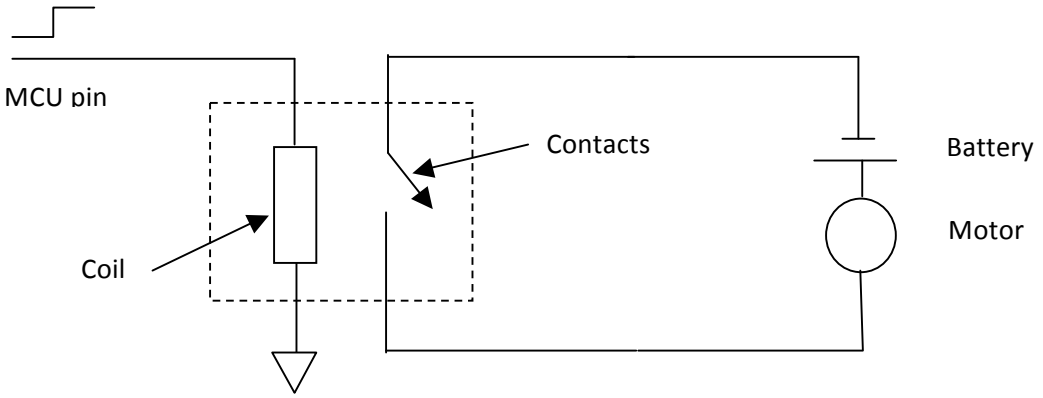
High-current devices are typically controlled using signals from the microcontroller through some sort of intermediate device like a relay, a transistor, or a more highly integrated device like a dedicated motor driver/controller board.

Relays

An electro-mechanical relay can be used to switch a high-current device on or off. Relays have a set of contacts and an actuating coil. The electromagnetic field resulting from passing a current through the coil moves the contacts to either make, or break, an electrical connection. The contacts can handle much more current than the current required through the coil to actuate them, thus providing a way for a small current from an MCU output pin to control a much larger current. In it's simplest form a single-pole single-throw relay looks like this:

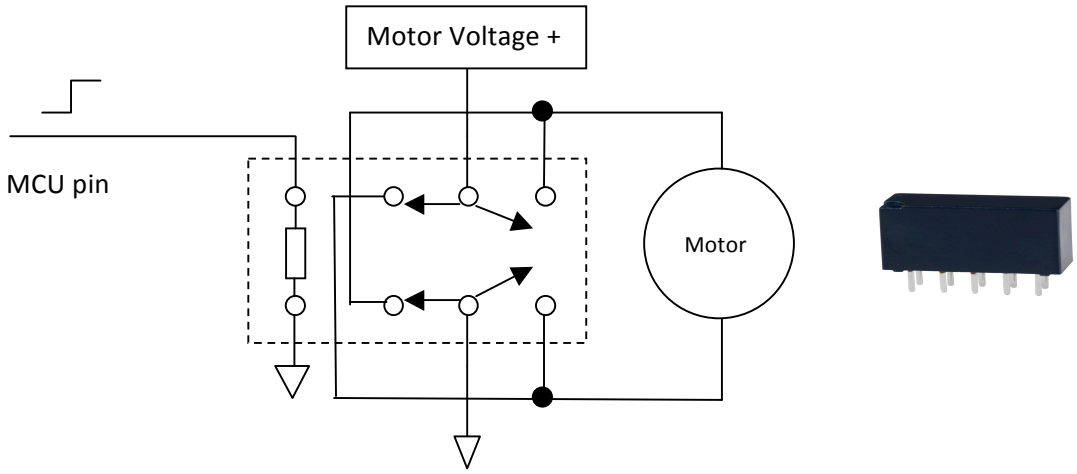


With an appropriate voltage applied to the terminal labeled “V”, a small current will flow through the coil producing an electro-magnetic field which causes the normally open (NO) contacts to close. Below is an example of how such a relay might be used to turn a motor on and off.



Selecting an appropriate relay involves reading the relay data sheet to see if the actuating current of the relay coil can be met by the current-source capability of the MCU output pin. The relay data sheet will typically specify the resistance of the coil and the design voltage for the coil. Some small relays are designed to be driven directly from an MCU. The 5 volt output of the MCU pin, combined with the coil resistance, will determine whether or not the MCU pin can adequately drive the relay. Assuming the coil drive requirements can be met, the next required specification is the current handling capability of the relay contacts. This specification is then compared to the current requirement of the device you want to control.

A SPST relay like the one above can turn a device on or off, but cannot change the direction of current flow. In the case of a solenoid or a light bulb this probably doesn’t matter. However, in the case of a motor we often want to control not only the on/off state, but the direction of rotation as well. This can be accomplished with a double pole double throw (DPDT) relay. The contacts in a DPDT setup for reversing the motor direction look like this:



With this arrangement we can control the direction of current flow. By having the MCU output pin actuate the relay, or not, we can make the double throw contacts "rock" back and forth and thus control the rotation direction of the motor.

Note however that while we can switch a motor on or off (SPST) or control the direction of motor rotation (DPDT), we cannot control the speed. Motor control with a relay only allows full On, or full Off, but no intermediate speed control. This may or may not matter. Control of low-speed gearhead motors often just requires simple stop or go control, either uni- or bidirectional.

Relay summary

- Control of high current with a small current.
- Zero leakage current in the off state.
- Very low resistance in the on state.
- Simple control requirements.
- Very good isolation between the input (control) and output (load) circuits.
- Not susceptible to damage from inductive voltage spikes.
- Relatively slow switching times.
- Cannot provide intermediate current control; full on or off only.
- Control cannot be modulated with a PWM signal.

Solid-State Relays: A special class of electronic "relays" should be mentioned. These are the so-called *solid-state relays*, which are used to switch **AC loads** often operating in the 100-230 VAC range. They take a low-current 3-32 VDC control signal, easily available from a microcontroller, and can switch AC loads up to 10's of amps. Despite the "relay" terminology these are purely electronic devices which have no contacts or other moving parts. They are designed for "zero-crossing" switching, meaning they switch at the moment where the AC voltage crosses the zero-volt point, thus reducing electrical switching noise and voltage spikes which would otherwise occur. These devices are easy to apply, can be quite useful, and can be rather expensive depending on how much current they can handle..

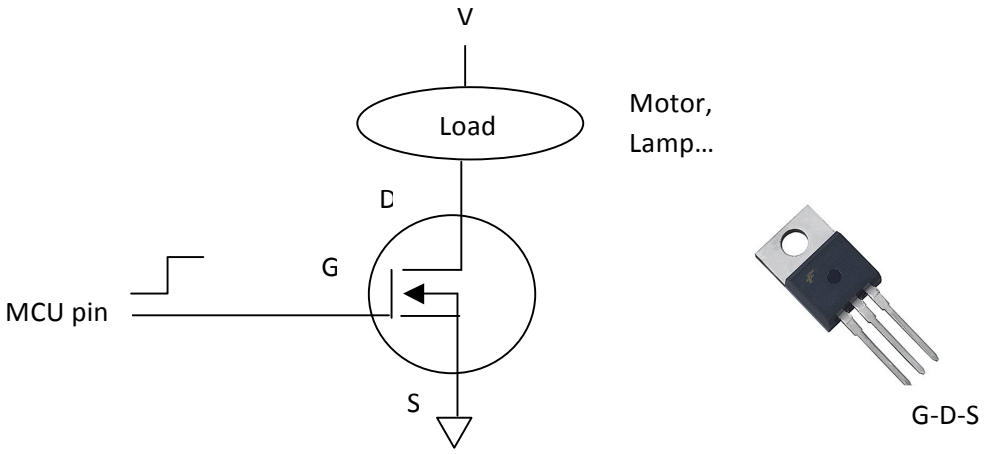
Transistors

A shortcoming of mechanical relay control is that they switch devices on or off, but cannot control intermediate currents such as required for variable motor speed control, variable lamp brightness, variable valve opening, and similar requirements. A transistor controlled by a microcontroller output pin *can* provide variable control. While there are many types of

transistors, we focus here on one of the types often used for power-switching, called a ***power MOSFET*** (Metal Oxide Semiconductor Field Effect Transistor).

A MOSFET is a three-terminal device. The terminals are denoted Gate, Source, and Drain. As MOSFET transistors were developed the nomenclature was taken from physics, so that in the circuit where they are employed *source* refers to the source of electrons (e.g. "ground") and *drain* refers to the sink for those electrons (5V, 12V…). The *gate* is the control terminal; a voltage applied to the gate governs the flow of electrons (current) from source to drain. Keep in mind that this terminology is reversed from the usual sense in electronics where we think of current flowing from a high potential toward a lower potential (often ground).

A MOSFET, controlled by an output pin on the MCU, is schematically shown below:



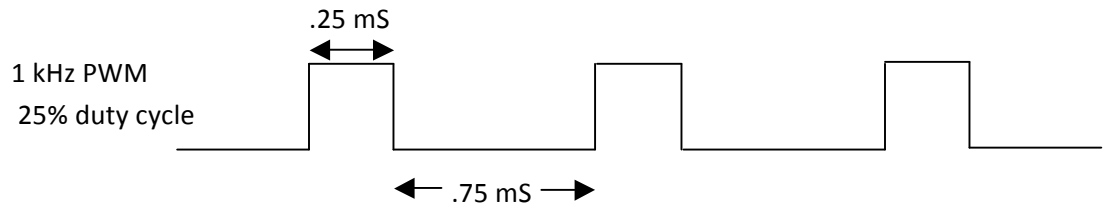
where G = gate, D = drain, and S = source. Although a number of other transistor types could be used, the MOSFET has some advantages. The first is that the gate input impedance is very high, which means it draws virtually no current from the driving circuit. Related to this input characteristic is the property that MOSFETs are controlled by an input *voltage*, not a current. In general these characteristics make MOSFETs easier to apply than current-controlled BJT (bipolar junction transistor) types. In addition, note that the voltage labelled V at the top of the load can be any DC voltage, not just the 5V of the MCU. For many of our projects the voltage to operate the load (often a motor) will be 12 volts.

In order for the MOSFET to begin conducting, the voltage applied to the *gate* terminal must be greater than the voltage at the *source* terminal by some device-specific gate-source ***threshold voltage*** (V_{GS}). For our applications we often use so-called *logic* MOSFETs, for which a fully "on" condition is guaranteed by the manufacturer for a V_{GS} of typically 3 volts or so. Some other important parameters include the effective resistance between drain and source at *saturation* (fully driven on), denoted as R_{DS}-on, which is often a small fraction of an ohm, the maximum

current handling, which can be up to 10's of amps, and the maximum voltage applied at the drain terminal. Heat sinks are often required for the MOSFET to handle continuous currents greater than 1-2 amps.

The control voltage on the gate could be a variable analog voltage supplied by a potentiometer or a digital-to-analog converter (DAC). Varying the gate voltage over the 0-5V range would produce a proportional current through the device thereby controlling the speed of a motor.

However, in the microcontroller world we usually do this job using pulse-width modulation (PWM). A PWM signal consists of a square-wave pulse train characterized by a frequency and a *duty cycle*. The duty cycle refers to the percentage of time the square wave spends at the high (usually 5V) level. For example, a PWM signal operating at 1000 Hz with a 50% duty cycle would look like a square wave with a 1 millisecond period during which the signal is at 5V for half a millisecond and at 0V for half a millisecond. By varying the duty cycle of the PWM output between 0 and 100% we can achieve the effect of a continuously varying voltage to the gate. This continuous effect results because the motor, or filament light bulb, or similar device typically cannot respond to the individual transitions of the gate drive PWM waveform, but responds rather to the average level of the waveform.



The ATmega328 MCU used on the Arduino Uno microcontroller board has six *hardware* PWM channels. The significance of doing PWM output with hardware capability is that once initialized and running the PWM output will continue autonomously until changed or stopped. The Arduino compiler instruction is: **`analogWrite`**`(pin,duty)` which specifies which PWM pin to use (pins 3,5,6,9,10,11 can be used for PWM output), and the duty cycle (0-255, which corresponds to 0-100% duty cycle). The Arduino default PWM frequency is 490 Hz for pins 9,19,11,13, which works generally well for motor control. The default frequency for pins 5 and 6 is 976 Hz. While the Arduino compiler has no instruction to change the PWM frequency, the frequency can be changed by reconfiguring the TCCRxB register, where x refers to internal timer 0,1, or 2. Further details on PWM frequency are available on the Arduino website.

MOSFET Summary

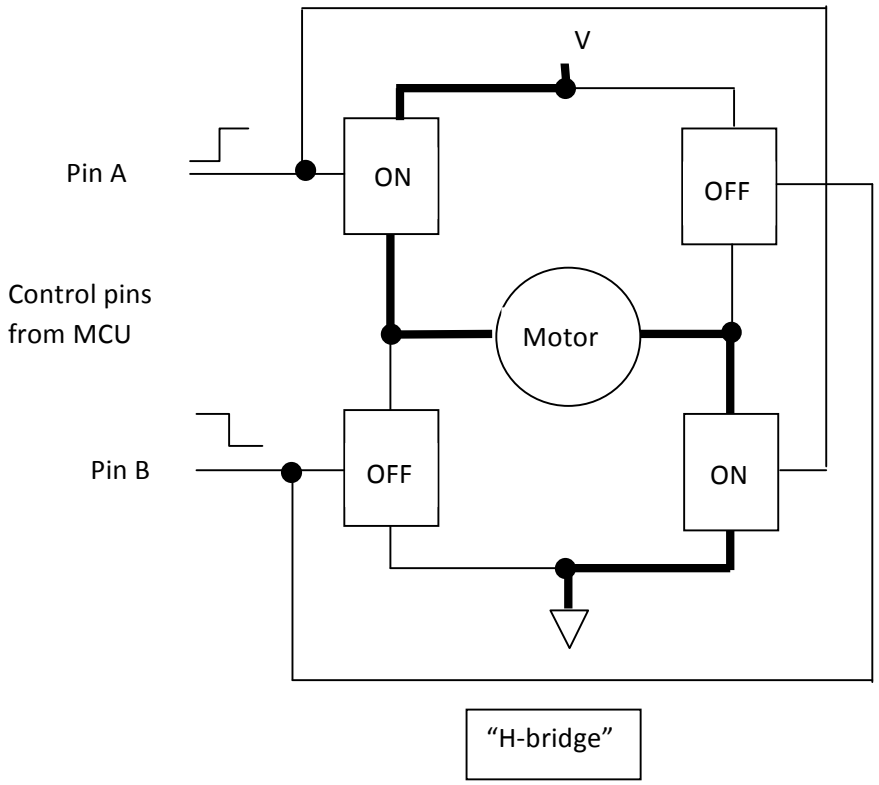
- Easily controlled from a microcontroller pin.
- High input impedance means the gate does not load the control signal.
- Can be used with PWM outputs for variable motor speed control.
- Can easily handle 10's of amps of DC current.

- Not used to control AC devices.
- Often requires a heat sink if operating at higher currents.
- Can be damaged by inductive voltage spikes or electrostatic discharge.

Motor Driver Solutions

Using a single MOSFET allows us to continuously vary the speed of a motor, but we cannot control the direction of rotation. The relay allowed us to control the direction, but not the speed. One possible solution would be to combine direction control using a relay with speed control using a MOSFET. This would work, and depending on other design constraints can often be a satisfactory solution.

More commonly however, an all-electronic approach is taken. By using four MOSFETs we can arrange them to control both speed and direction. This circuit design is called an *H-bridge*, and is conceptually diagrammed below.



The H-bridge circuit shown above takes it's name from the arrangement of the motor and the four MOSFETs labeled ON and OFF. Which MOSFETs conduct current is governed by the state of the two control terminal pins A and B. Note that each pin controls two MOSFETS. Depending on the states of pins A and B, current can flow from voltage V through the motor to ground as shown by the bold lines. Reversing the states of A and B, the current would pass in the opposite direction through the motor thereby reversing the motor. In addition, A and B can be (and usually are) PWM signals, thus giving us both direction and speed control. A key critical point is that pins A and B must never be set such that both transistors on the same side of the H-bridge are simultaneously on, as that would produce a direct short from voltage V to ground.

The next step up in integration would be to control discrete MOSFETs not directly from the MCU, but through an intermediate specialized integrated circuit (IC) which contains internal circuitry to protect against inductive voltage spikes which result from switching current through the motor, and other control circuitry. There are a number of specialized motor drive integrated circuits, each offering different combinations of features optimized for particular control situations.

Motor control circuits are frequently designed using the ideas and components mentioned above, with power MOSFETs and PWM signals from a microcontroller. The motor can be driven with discrete MOSFET components, through intermediate specialized ICs which drive the MOSFETs, or they can be driven entirely with integrated circuits containing the complete H-bridge.

However, there are times when the design engineer needs to do motor control but doesn't want to put together a motor control circuit themselves. The solution here is to purchase a motor controller board. Motor controller boards hide most of the electronic control details from the user, and typically have two control inputs: a direction control pin and a speed control pin. These are 'logic-level' control pins, each one driven directly from output pins on the microcontroller. Direction control is achieved by setting the direction pin either high (1) or low (0). Speed control is a PWM pulse train with a programmable duty cycle sent from a PWM output pin on the microcontroller.

For our projects we will be using a motor controller board designed for the Arduino microcontroller platform. Refer to the handout on "PID Control: A Brief Introduction…" for details on using the motor control board.