In Chapter 10, we interfaced a stepper motor. Stepper motors are appropriate when we wish to control the angle of the shaft. For example when moving paper through a printer we wish to move paper in small and accurate amounts. On the other hand if we wish to rotate the shaft quickly with lots of power we will use a **DC motor**. We rate motors according to their **torque**, which is force times distance.

The DC motor has a **frame** that remains motionless (called the **stator**), and an **armature** that moves (called the **rotor**). A **brushed DC motor** has an electromagnetic coil as well, located on the rotor, and the rotor is positioned inside the stator. In Figure 12.7, North and South refer to a permanent magnet, generating a constant *B* field from left to right. In this case, the rotor moves in a circular manner. When current flows through the coil, a magnetic force is created causing a rotation of the shaft. A brushed DC motor uses commutators to flip the direction of the current in the coil. In this way, the coil on the right always has an up force, and the one on the left always has a down force. Hence, a constant current generates a continuous rotation of the shaft. When the current is removed, the magnetic force stops, and the shaft is free to rotate. In a pulse-width modulated DC motor, the computer activates the coil with a current of fixed period and fixed magnitude but varies the duty cycle in order to adjust the power delivered to the motor.

VIDEO 12.4A. WORKING OF A DC MOTOR

C12 4a DC motor interface YouTube

JONATHAN VALVANO: Let's look at how the DC motor works.

We have a north magnet, and a south magnet.

And this will produce a B field, and this will create a magnetic field from

north to south.

Next we're going to have a wire

8:48 / 8:48 1.0x

This is a course on robots and embedded systems, so we're going to have wires.

And through this wire, a current is going to flow.

So if current flows through this wire, and this wire

is perpendicular to the magnet, then what will happen

is a force will be induced on the wire such that the magnetic field--

the electrical current and the force-- are all orthogonal to each other.

The actual interface to create these magnets, these electromagnets,

will be to have a winding-- and we saw that in our stepper motor--

such that if I were to drive current through this circular path,

that will create a magnetic field in this direction.

From an electrical standpoint, we will see the motor has three parts.

The resistance of the windings, the inductance of the coil, and then we're

going to have this magic electromotive force.

Interestingly enough, this conversion is two directional.

So mechanical force can be converted to electrical energy,

and electrical energy can be converted to mechanical energy.

So the micro-controller, in order to generate electrical energy,

has to produce current through the coil.

And this current, times the voltage across the coil, as you know is power.

And if we take power and we do it for a certain amount of time,

we're going to have energy.

So this is the essential conversion here, that occurs in the interface.

So let's look at this interface.

We're going to have an output port of the microcontroller.

This will be a digital output as you know, zero or 3.3 volts.

And it's going to go through a 1K resistor.

The currents through these motors are quite large.

Help

The one on our stepper motor car will vary from either 100 milliamps

all the way up to one amp depending upon how much load there is.

And these currents are way too large to connect to a microcontroller.

So we're going to use a current amplifier, or transistor.

And so this is the coil, which as you saw contains some R some L and EMF.

And to prevent things from exploding, I'm going add a snubber diode.

Right there, that's a 1N914.

And because this is a one AMP motor, I'm going

to use a transistor which can drive lots more than one amp.

I'm going to use a TIP120.

This is an NPN Darlington which essentially amplifies current.

And so I'm going to have current on the order of milliamps driving out

of the microcontroller, into the base, down to the emitter.

And this current is going to be amplified such

that I can have up to an amp of current flowing out of my battery,

across the motor, across the collector, down to the emitter,

and down to ground.

Here's our circuit again.

Let me remind you how to connect it up to the microcontroller,

without exploding the microcontroller.

Because now we have lots and lots of current.

The battery here, it's going to control the robot.

And it's is important to connect the battery such

that the current flow comes out of the battery, across the motor,

and it will flow across the collector of this transistor,

and back to the battery.

And the current-- this one amp of current

is not going to flow into the micro-controller.

Help

Now we are going to need to power the microcontroller.

So we're going to take this 8.4 into a regulator--

just like we did with the stepper motor car-- to get a 5 volt output,

and tie this into VBUS.

But these currents here are very tiny.

So the microcontroller current, the lower current--

which is on the order of 10 to 50 milliamps--

to control the board, that's going to flow through this direction

and back to the battery.

And so we see we've got two separate current paths.

One for the motor, and one to drive the board.

And so we have our LaunchPad powered in this way.

OK, now it's time to write some software.

We saw that the energy applied to the motor is a function of the voltage.

Which in our case, is fixed at 8.4 volts.

The current, which is also fixed, from 100 milliamps to one

amp depending upon the load and time.

And it's the time parameter that we're going to control.

So I'm going to connect the circuit to PA5.

I could've used any output pin.

And I'm going to create what's called a pulse width modulated signal.

Basically I'm going to adjust the parameter time.

And the way I'm going to do a pulse width

modulated signal is going to create a pattern, such

that the period-- or the frequency of this wave-- is fixed.

But the duty cycle is going to vary.

So if I labeled this time H for high, and this time L for low,

that's the time.

And there's an H and there's an L and there's

an H. I'm going to fix the period by forcing H plus L to be a constant.

Help

And I'm going to control the duty cycle, by having H over H plus L vary.

And so the microcontroller can make the motor spin faster

by increasing the duty cycle.

Or even-- or even faster.

And we can see here that the period of each of these waves, which is H plus L,

is fixed.

But the duty cycle will vary.

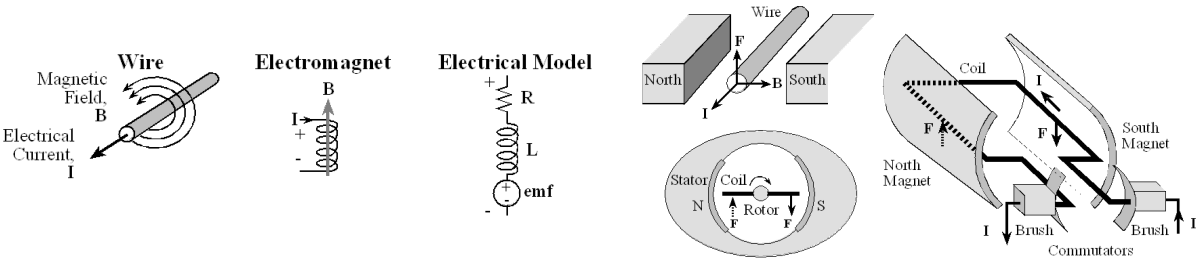

Figure 12.7. A brushed DC motor uses a commutator to flip the coil current.

In the LED interface the microcontroller was able to control electrical power to the LED in a binary fashion: either all on or all off. Sometimes it is desirable for the microcontroller to be able to vary the delivered power in a variable manner. One effective way to do this is to use pulse width modulation (PWM). The basic idea of PWM is to create a digital output wave of fixed frequency, but allow the microcontroller to vary its duty cycle. The system is designed in such a way that **High+Low** is constant (meaning the frequency is fixed). The **duty cycle** is defined as the fraction of time the signal is high:

$$duty\ cycle = \frac{High}{High + Low}$$

Hence, duty cycle varies from 0 to 1. We interface this digital output wave to an external actuator (like a DC motor), such that power is applied to the motor when the signal is high, and no power is applied when the signal is low. We purposely select a frequency high enough so the DC motor does not start/stop with each individual pulse, but rather responds to the overall average value of the wave. The average value of a PWM signal is linearly related to its duty cycle and is independent of its frequency. Let *P*(*P=V*I*) be the power to the DC motor, shown in Figure 12.8, when the PA5 signal is high. Under conditions of constant speed and constant load, the delivered power to the motor is linearly related to duty cycle.

Delivered Power = $duty\ cycle * P = \frac{High}{High + Low} * P$

Unfortunately, as speed and torque vary, the developed emf will affect delivered power. Nevertheless, PWM is a very effective mechanism, allowing the microcontroller to adjust delivered power.

The resistance in the coil (*R*) comes from the long wire that goes from the + terminal to the – terminal of the motor, see Figure 12.8. The inductance in the coil (*L*) arises from the fact that the wire is wound into coils to create the electromagnetics. The coil itself can generate its own voltage (emf) because of the interaction between the electric and magnetic fields. If the coil is a DC motor, then the emf is a function of both the speed of the motor and the developed torque (which in turn is a function of the applied load on the motor.) Because of the internal emf of the coil, the current will depend on the mechanical load. For example, a DC motor running with no load might draw 100 mA, but under load (friction) the current may jump to 1 A.

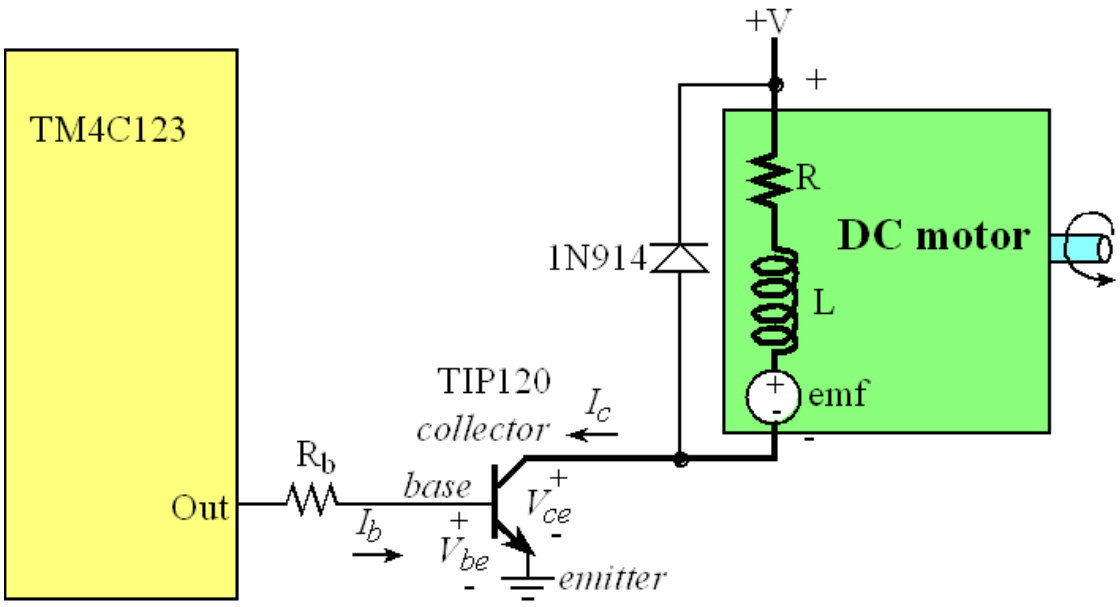

Figure 12.8. The system uses a transistor to turn the motor on and off.

There are lots of motor driver chips, but we will use an NPN Darlington transistor, e.g., TIP120, to allow the software to turn a motor on and off. If the port output is low, no current can flow into the base, so the TIP120 transistor is off, and the collector current, I_C, will be zero. If the port output is high, current does flow into the base and V_{BE} goes above V_{BEsat} turning on the TIP120 transistor. The transistor is in the linear range if $V_{BE} \leq V_{BEsat}$ and $I_c = h_{fe} \cdot I_b$. The transistor is in the saturated mode if $V_{BE} \geq V_{BEsat}$, V_{CE} = 0.7V and $I_c < h_{fe} \cdot I_b$. We select the value of the R_b resistor to operate the transistor in its saturated mode. We start with the desired motor current, which will be the collector current I_c. The voltage across the coil will be the $V - V_{CE}$. Next, we calculate the needed base current (I_b) given the current gain of the NPN

$I_b = I_c / h_{fe}$

knowing the current gain of the NPN (h_{fe}), see Table 12.7. Finally, given the output high voltage of the microcontroller (V_{OH} is about 3.3 V) and base-emitter voltage of the NPN (V_{BEsat}) needed to activate the transistor, we can calculate the desired interface resistor.

$R_b \leq (V_{OH} - V_{BEsat})/I_b = h_{fe} * (V_{OH} - V_{BEsat})/I_c$

The inequality means we can choose a smaller resistor, creating a larger I_b. Because the h_{fe} of the transistors can vary a lot, it is a good design practice to make the R_b resistor 0.1 to 0.5 times the value shown in the above equation. Since the transistor is saturated, the increased base current produces the same V_{CE} and thus the same coil current.

Parameter	TIP120 (at I_{CE}=200mA)
h_{fe}	900
V_{BEsat}	1.3 V
V_{CE}	0.7 V
I_{CE}	Up to 5A

Table 12.7. Design parameters for the TIP120.