2

The Bare Essentials

2.1 ACTUATORS

A mechatronic system must "do" something, even if it is just to move a pointer or change a display. The industrial robot must have motors with which to move an end effector, perhaps a gripper, while another system's output might concern heaters.

The mechatronic engineer should not be in too much of a hurry to run to the catalog to choose an electric motor. To the electrical engineer, motors are a fascinating playground around which to debate the merits and challenges of axial flux, windage losses, rotor resistance, or commutation. The mechatronic engineer is by no means certain that the solution does not instead lie with something hydraulic or pneumatic.

This section attempts to put a selection of the vast range of actuators into some sort of perspective.

2.1.1 Choosing a Technology

The first question to ask is: "What must the output do?"

At the bottom end of the list, in terms of power, is the task of displaying a value on an indicator. Many automobile instrument panels have now been taken over by liquid crystal displays, probably putting them outside the grasp of mechatronics, but they are just the tip of the iceberg.

For many years the simplest of cheap automobile instruments, such as the fuel gauge, have been moved by a bimetal strip. Around it is wound some resistance wire. As current is passed through the wire, the temperature rises and the bimetal bends to move the pointer. A simple twist compensates for variation in ambient temperature. This old technology has been given a new lease on life by the arrival of *memory alloys* that change their shape with temperature.

For many applications, this simplicity and robustness is ruled out of question by a need for a rapid response. An electromagnetic solution might have more appeal.

When current flows in a conductor within a magnetic field, the conductor experiences a force. That more or less sums up electric motors! But the devil is in the detail.

In an electromagnetic indicator, the force is opposed by a spring, so that the deflection of the needle increases with the current.

The simplest electromagnetic actuator that can move a load is the solenoid. When current passes through the solenoid's winding, it results in a magnetic field that causes a slug of soft iron to move to close a gap in the magnetic circuit. This single action might be enough, say, to release a remote-entry door lock. But other applications demand something more versatile.

2.1.2 DC Motors

You are probably most familiar with the permanent-magnet DC motor, used in everything from toys to tape recorders. The rotor is wound in such a way that the electromagnetic force causes the rotor to rotate. If the currents in the motor's conductors were constant, the rotor would move to some stable position, swing to and fro around it a few times, and then come to rest. But the current is not allowed to be constant. Long before the stable position is reached, a commutator breaks the current to that particular coil and energizes the next one in succession. The motor continues to rotate.

The "old-fashioned" structure of the commutator used curved plates of copper with brushes, often made of carbon, that rubbed on them. The "brushless" DC motor is becoming increasingly common. Here sensors measure the rotor position, and electronic switches apply the commutation by selecting the appropriate coils. There is another important difference. Since the magnetic material usually has more mass than the rotor, in a traditional motor it is the coils that rotate. In a brushless motor the coils are fixed and the magnet rotates.

2.1.3 Stepper Motors

Now let us take away the commutation again. Energize one coil and the rotor is pulled to a particular position, requiring a fair amount of torque to deflect it. Energize another coil and the motor "steps" to another position. In other words, by selecting coils in sequence, a computer can step the motor an exact number of increments to a new position—this is a "stepper motor."

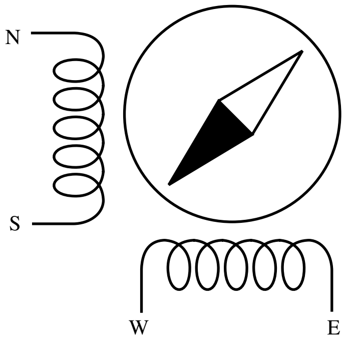

Figure 2.1 *Stepper schematic—NSEW.*

Think of it in terms of a compass needle being pulled into line by a pair of coils, arranged north–south and east–west (see Fig. 2.1). Current can be passed through these coils in either direction, so we might start with both the NS and EW coils being driven in the "positive" direction, resulting in the needle pointing northeast. Now if we reverse the drive the NS coil, the needle will move to point southeast. Reverse the EW coil, and it will rotate to point southwest. Make the drive to the NS winding positive again, and the needle moves on to point northwest. Finally reverse the EW drive to be positive and the needle completes the circle to point northeast once again. You can see an animation of this at www.essmech.com/2/1/3.htm

In practice, the magnet of a stepper motor has a large number of poles, and the windings are helped by a similar large number of salient polepieces (Fig. 2.2) in the soft iron on which they are wound. As a result, the switching sequence must be repeated 50 times for a "200-step" motor to make one complete revolution.

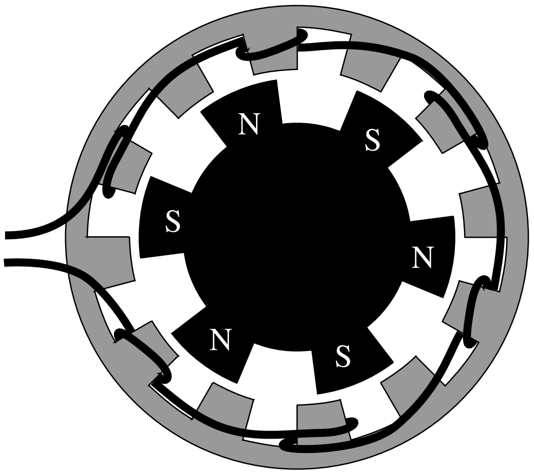

Figure 2.2 *Stepper schematic—polepieces.*

Simple software can command the motor to move to a desired position, so the stepper motor has great appeal for the amateur robotics builder. But it has a great number of shortcomings. There is a limit to the torque it can resist before it "clunks" out of the desired position and rotates to a different stable location. If a transient of excessive torque causes it to "drop out of step", then, without a separate position transducer, the slip goes unnoticed by the processor and the error remains uncorrected. What is more, this dropout torque decreases markedly with speed. An attempt to accelerate the motor too rapidly can be disastrous and the software is made more complex by the need to ramp the speedup gently.

Of course, there are other ways than the use of a permanent magnet for producing a magnetic field. More powerful DC motors, such as automobile starter motors, use current in a *field winding* to generate the stator's magnetic field. Similar motors are not restricted to using direct current. By connecting the stator and rotor windings in series, the torque will be in the same sense whether positive or negative voltage is applied across it. The motor can be driven by either an AC or DC voltage. This is the *universal motor* (Fig. 2.3), to be found in vacuum cleaners and a host of other domestic gadgets.

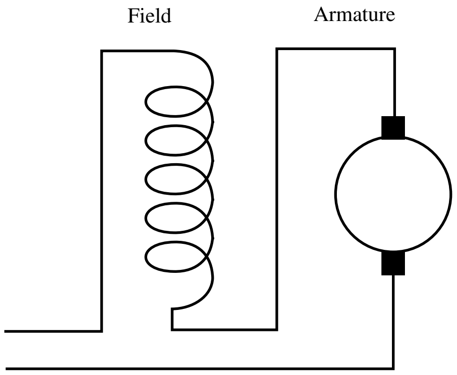

Figure 2.3 *Universal motor.*

2.1.4 AC Motors

Another family of motors depend on alternating current for their fundamental mode of operation. They use *rotating fields*. If the stator has two sets of windings at right angles and if a sine-wave current flows in one winding and a cosine-wave current flows in the other, then the result is a magnetic field that rotates at the supply frequency.

This is illustrated at www.essmech.com/2/1/4.htm.

From this one simple principle, a host of variations are possible. In one case, short-circuited coils are wound onto a soft-iron rotor. If the rotor is stationary, the rotating field induces currents in the rotor coils that in turn propel the rotor to rotate with the field. So the rotor accelerates, but cannot quite catch up with the field. If the rotor were to rotate at the supply frequency, it would experience no relative rate of change of field and no current would be induced in it.

The rotor "sees" the *slip frequency*, the amount by which the rotation falls short of the field rotation. Large industrial motors are designed to give maximum torque for a few percent of slip, thus improving their efficiency but requiring some special provision to get them up to speed (see Fig. 2.4).

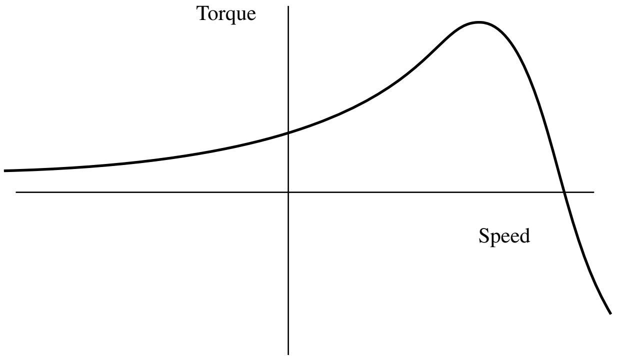

Figure 2.4 *Torque–slip curve.*

Induction motors can make useful servomotors. If the sine-wave winding is powered "at full strength" while the cosine-wave current is of a variable magnitude, then the rotating component of the field can be varied in strength, including the possibility of reversing its direction. This variable field servomotor has sufficient torque to move aircraft control surfaces. Now, however, the torque–slip characteristic must be modified so that maximum torque is generated at 100% slip—when the motor is at a standstill.

It is possible to run a two-phase induction motor from a single-phase supply. One phase is connected across the supply, while the second is energized via a series capacitor. This capacitor gives the phase shift that is needed to result in a rotating field. However, many appliances such as water pumps have a switch to disconnect the capacitor as soon as the motor is "up to speed," so that the motor continues running from a single phase alone. It works as shown in Figure 2.5.

The field from a single phase can be thought of as the result of two fields rotating in opposite directions (see curves in Fig. 2.6). The motor has a torque–slip characteristic that gives maximum torque for a small slip. Thus the torque from the field going in the "correct direction" is much greater than that of the opposite direction, so that the motor continues to rotate efficiently.

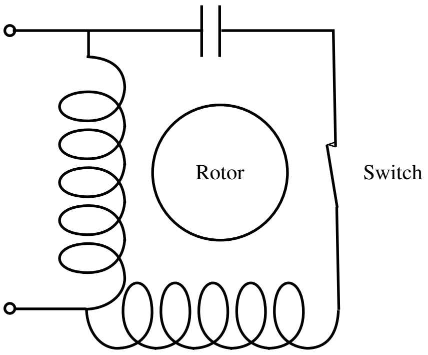

Figure 2.5 *Motor with starter capacitor and switch.*

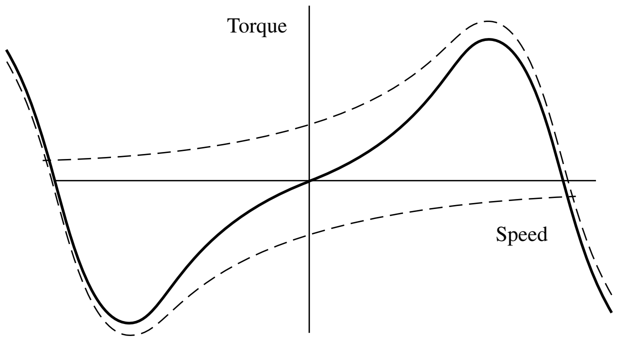

Figure 2.6 *Combination of two torque–slip curves.*

If a servomotor were designed with an "efficient" torque–slip curve, it would be in danger of running away, failing to stop when the control voltage was removed. But the conventional two-phase induction motor has some simple uses, as you will see in the section on interfacing.

Large induction motors are wound for three-phase operation, and their interfacing for control applications, such as in an elevator, presents problems all of their own.

The rotor of another variation of motor contains no iron and consists merely of a thin cylinder of copper. It will still experience rotational forces after the fashion of an induction motor. This is the "drag cup" motor, popular in servo repeater systems as used in autopilots in the 1960s. Although its torque was small, its tiny moment of inertia meant that response speeds could be very rapid.

Soft iron acquires a magnetic moment when in the presence of another magnetic field, but loses it when the field is removed. A permanent magnet is made of a "hard" material in which the magnetic moment can be self-supporting. Somewhere between these extremes is the material used in a *hysteresis motor.* The rotating field induces a magnetic moment in the rotor that remains, even as the rotating field advances. The field thus drags the rotor after it. Even when the rotor has accelerated to rotate in synchronism with the field, the residual permanent magnetism will keep the hysteresis motor rotating.

We can double the number of poles on the stator, so that the motor's basic speed of rotation is halved. The variations are endless. A motor can be constructed without bearings as a pair of rings, to be mounted on opposite sections of a robot joint. The structure can involve fields that are radial, as in a "conventional" motor, or fields that run parallel to the axis of rotation.

2.1.5 Unusual Motors

If you think that axial field motors are rare, rescue a $5\frac{1}{4}$-in. floppy disk drive from the junk heap. There is an easily recognized stepper motor for moving the head in and out, but where is the motor that rotates the floppy?

When you remove a plate from the large circuit board, you will see copper windings "stitched" to the board like the petals of a flower (see Fig. 2.7). The plate that covered it was in fact the magnetic rotor that carries the floppy round with it, while sensors on the board switch the fields to control the rotation. This axial field motor is truly "embedded" in the product, rather than being added as an identifiable component.

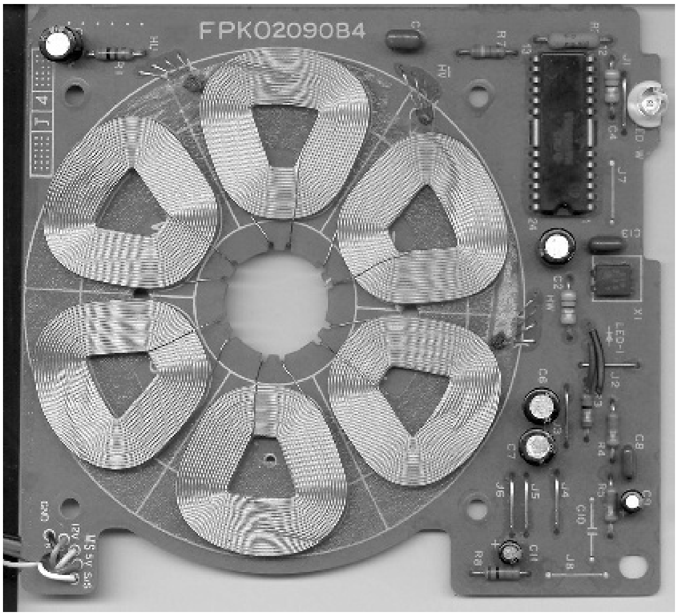

Figure 2.7 *Windings on floppy board.*

A motor can even be "rolled out flat." A linear stepper motor has its pole-pieces side-by-side. It is mounted close to a linear track stacked up from "slices" of magnetic and nonmagnetic material, along which it steps its way at considerable speed.

A linear induction motor can be propelled along a conducting or magnetic plate. This is a popular form of propulsion for hovering or magnetic levitation trains.

So, the mechatronic designer has much more to worry about than finding a motor in a catalog. Why should the motors be electrical at all? How about hydraulics and pneumatics?

2.1.6 Hydraulics and Pneumatics

The fundamental principles of these seem to be glaringly obvious. First, you must construct a cylinder and place a piston in it, maybe resulting in something not very different from a bicycle pump (see Fig. 2.8). When you pressurize the air or oil in one end of the cylinder, the piston will be forced away. Once again, the details make a simple situation very complicated.

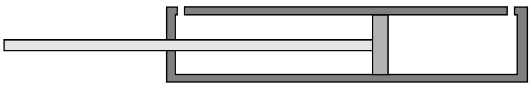

Figure 2.8 *Hydraulic/pneumatic cylinder.*

There are essential differences between hydraulics and pneumatics. Air is much more compressible than oil, but has much less inertia. Pneumatics will therefore have the edge in situations where rapid acceleration is needed, but where the power is not large. Hydraulics will flex its muscles for the heavier tasks.

But the choice of motor technology cannot be made in isolation. Power and efficiency will be just one factor, while ease of interfacing and the control dynamics will require just as much attention.

2.2 SENSORS

If the range of actuators seemed vast, it does not compare with the gamut of possibilities offered by sensors. Of course, the field is narrowed down by the nature of the quantity that is to be measured. Perhaps it is best to list some of the possibilities.

2.2.1 Position

There is a fundamental need for a feedback signal for position control, but the choices are numerous.

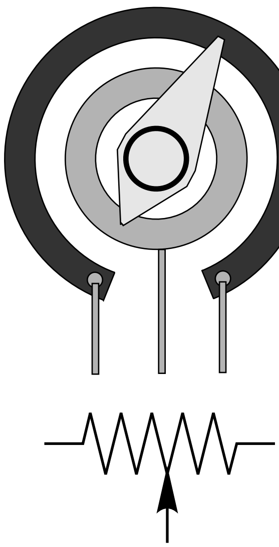

Figure 2.9 *Potentiometer.*

The “classic” sensor is the potentiometer (Fig. 2.9), where a voltage is applied across a resistive track and a moving “wiper” picks off a proportion of the voltage corresponding to its position. The track can be made from a winding of resistance wire, or a track of conducting plastic or carbon composition. The motion sensed can be linear or rotary. Prices vary by factors of hundreds, influenced by noise, reliability, and required lifetime.

The plastic potentiometer gives a signal that is not quantized—it varies smoothly and not in steps. This analog property has many advantages when the objective is simplicity, but like all analog signals, there is a question of accuracy and linearity.

Other sensors are “steppy” by nature, where the steps are absolutely defined by the construction of the sensor. One such sensor is the *incremental encoder*. Yet again, several technologies offer themselves, but the one most frequently found is optical.

Think of a sequence of “stripes” (Fig. 2.10) moving between a light source and a phototransistor. The associated electronics can “see” the difference between stripe and gap, and can count the stripes as they move past. This count is presented as the measured position. Now, this is all very well if we are certain of the direction of movement, but clockwise or anticlockwise (counterclockwise) movement will give signals that are indistinguishable from each other. So, how can they be resolved to count up or down?

The answer is to provide a second phototransistor alongside the first to render the transducer *two-phase* (see Fig. 2.11). The signals change one after the other. Now if we give the values 0 or 1 to each of the pair of signals, we might see the sequence of values 00, 10, 11, 01, 00, and so on for the pair when

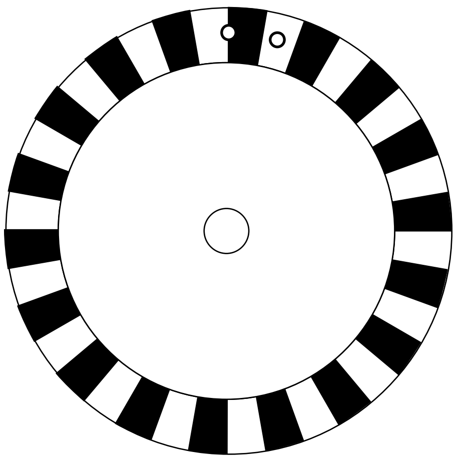

Figure 2.10 *Two optical sensors, with stripes.*

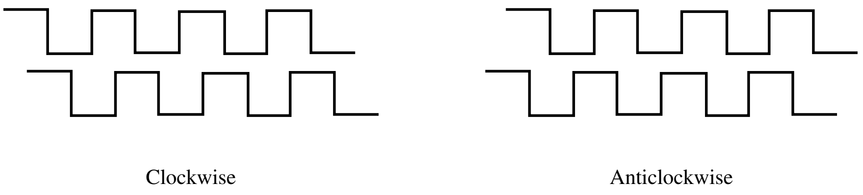

Figure 2.11 *Waveforms traveling forward and reverse.*

the rotation is clockwise. If it is rotated anticlockwise, however, it would be 00, 01, 11, 10, 00, and so on. Look closely, and you will see an essential difference, one that will cause a logic circuit or a couple of lines of software no trouble at all.

Even within optical sensors, there are many choices. The resolution can be increased to many thousands of "stripes" per revolution by passing two optical gratings across each other and observing the change in the Moiré pattern. An even finer resolution can result from mixing laser signals to produce interference fringes.

Now, this incremental technique seems an ideal solution—until you realize that it tells you only the change in position, not the absolute position at switch-on time. To find the initial position, one solution is to run the system to some hard stop and reset the counter there.

A more subtle technique was used in the Unimation Puma robot. As well as the incremental stripes, of which there were around a hundred per revolution of the geared motor, an extra stripe was added so that one specific point in the revolution could be identified. At startup, the motor rotated slowly until this stripe was detected. This still left tens of possibilities for the arm position,

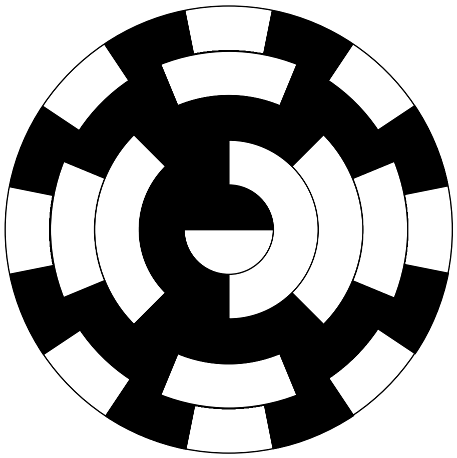

Figure 2.12 *Rotary Gray code.*

as the motor was geared to rotate many times to move the arm from one extreme to the other. But in addition to the incremental transducer, the arm carried a potentiometer. Although the potentiometer might not have the accuracy required for providing the precision demanded of the robot, it was certainly good enough to discriminate between whole revolutions.

The concept of "coarse–fine" measurement is a philosophy in itself!

Of course, there is the alternative possibility of reading 10 binary signals at once, giving a 10-bit number with an "at a glance" resolution of 0.1%. However, the 10 tracks to give these signals must be very accurately aligned, even when the output is arranged as in *rotary Gray code* (Fig. 2.12). In Gray code, the first stripe is a half-revolution sector. The next is also a half-revolution sector, displaced so that the two outputs divides the range into four quadrants. Then each successive track has double the number of stripes, with edges splitting in half the regions defined so far. The last has 512 fine stripes. These devices are not cheap!

Other sensors can be based on magnetic "Hall effect" semiconductors of the switching variety for locating an endstop or for counting teeth in a magnetic gear or track.

Alternatively, analog Hall effect sensors (see example in Fig. 2.13) can pick off the sine and cosine of a rotation angle, by detecting the component of a magnetic field perpendicular to their sensing plane. These are "noncontact" and have high reliability.

A chip is available from Phillips, the KMZ43T (the superseded version was the KMZ41), which gives sine and cosine outputs depending on the angle of the field. Provided it is strong enough, they are independent of its strength. The output gives two electrical cycles per revolution of the magnet.

A noncontact sensor that perhaps does not entirely deserve its popularity is the *linear variable differential transformer* (LVDT) (Fig. 2.14). A coil is

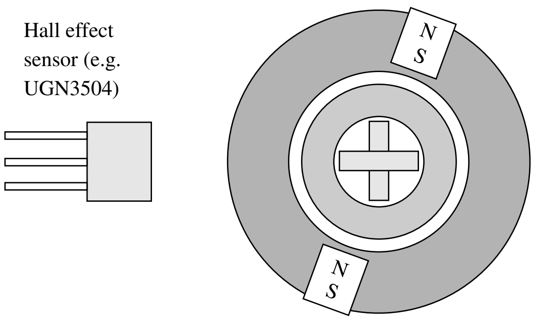

Figure 2.13 *"Crossed" Hall effect angle sensor.*

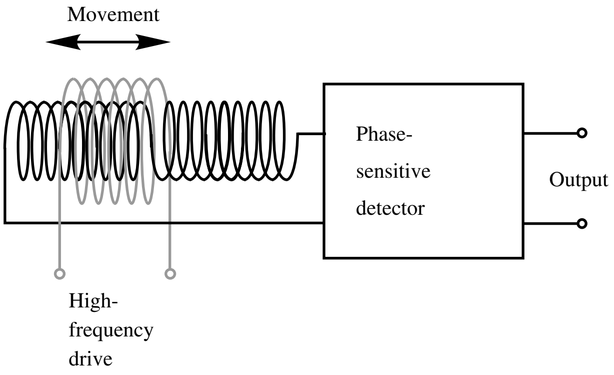

Figure 2.14 *LVDT.*

energized with a high-frequency signal, while another coil wound over it and sliding along it picks off a varying voltage by induction. The circuitry includes oscillator and detector, and the result is an output voltage that is proportional to the displacement.

There is a common displacement transducer that gives a resolution of a fraction of a millimeter in two dimensions, yet costs only $20 or less. It is used in the computer "mouse" and is based on the optical transducer described above. An alternative "optical" mouse has a simple image chip that "looks" at marks on the desk underneath the mouse and works out the movement by correlation.

If you widen your scope, the examples keep coming. GPS receivers must be included. By skillful combination of the carrier-phase measurements from two receivers, displacements can be tracked to subcentimeter accuracy. Distances can be measured with radar, with ultrasonic "estate-agent-grade" room measuring instruments or with the Polaroid sensors used in early autofocus cameras. The time of flight of laser pulses is used in another family of distance meters and in the Sick sensor.

2.2.2 Velocity

The response of most position controllers can be made more "crisp" by the addition of a velocity signal in the feedback. Now this can of course be derived by computation from a position sensor, although this "secondhand" signal is more sluggish than a direct measurement.

Any small permanent-magnet DC motor will act as a generator. When coupled to the shaft of a servomotor, it will give a voltage proportional to the velocity of rotation. The voltage will be somewhat "lumpy" as a result of the commutation, but will still be a great help to stability. The result is known as a "tacho" (*tachometer*) signal.

In an agricultural tractor, it is common to measure speed over the ground by radar. The unit is mounted below the body and detects the Doppler shift of a radar pulse directed obliquely at the ground.

Another form of velocity can be important for navigation: angular velocity. There are many forms of rate gyroscope to measure it. In classic form, these contain an inertial mass that spins at high speed. When there is a rotation about an axis perpendicular to that of the spin, it will result in substantial precession forces that are easy to measure. These rate gyroscopes have been an essential component of autopilots from early times.

More recent devices use a miniature "tuning fork" and have no rotating parts. As the fork rotates about its long axis, the to-and-fro vibration of the tines will gain a side-to-side component that can be measured. A similar device uses vibrations in a ring.

Yet another more recent device is based on the velocity of light in a coil of optical fiber.

2.2.3 Acceleration

A transducer for linear acceleration can take the form of a tilt sensor, measuring the displacement of a pendulum or of a mass mounted on leaf springs. This is really a secondhand measurement, where the acceleration is deduced from a position measurement of the movement of the pendulum. However, it is different from inferring the acceleration from measuring the position of the accelerating vehicle and computing differences.

Other sensors are designed to measure high-frequency vibrations. One range of commercial sensors is a close relative of a microphone, using either the change in capacitance between two moving plates or a piezoelectric voltage as a small mass resists the disturbing acceleration.

In a similar way, rotational acceleration can be deduced from the torque required to rotate a miniature "dumbbell," or from two linear accelerometers positioned some distance apart.

2.3 SENSORS FOR VISION

Optical sensing systems are many and various, with a variety of resolutions in both position and intensity. In order of increasing complexity, they can be ranked as follows.

2.3.1 Single Point, Binary

At the bottom of the ladder is a single *phototransistor*. When light falls on a phototransistor, it allows current to flow through it to cause a change in the output voltage.

A "pair" consisting of a single light-emitting diode (LED) and a single phototransistor can be used in a variety of ways. As a *reflective opto switch* they can detect a dark mark on a light background or vice versa. To obtain any kind of picture the sensor would have to be scanned mechanically in both directions.

As a *slotted opto switch*, the LED and the phototransistor are mounted to face each other and indicate when there is an obstruction in the slot. There are no real vision applications of this configuration, but it makes a useful detector for motion measurement (see example configuration in Fig. 2.15). Two such sensors monitor the movement in each axis of a conventional computer mouse.

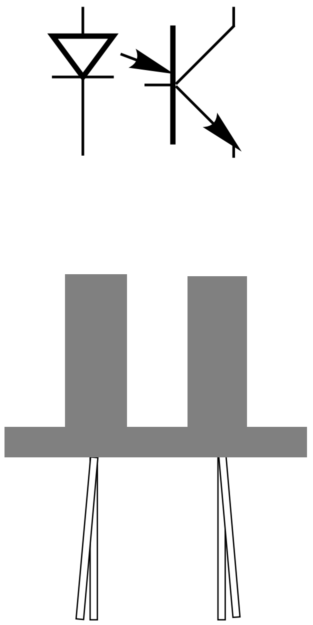

Figure 2.15 *Slotted opto switch.*

A simple "beambreak sensor" can be very versatile. It can calibrate the motion of a toolpiece moved by a robot, check the length of a drill, or verify that a hole has been punched.

2.3.2 Linescan Devices

A linescan sensor is used in every fax (facsimile) machine. The image of the page is focused onto the device, which "sees" a single scanline across the page. As the page is fed through the machine, the image is built up, line by line. This sensor could have around 2000 *pixels*, individual sensing elements, and will probably use the "bucket brigade" principle.

When light falls on the sensitive silicon sensor, a current flows that is proportional to the brightness. In the single phototransistor sensor the current is measured directly. In most camera systems, the current builds up a charge on a small embedded capacitor—part of the device. This allows the device to be much more sensitive, provided we are prepared to allow a little time to pass so that the charge can build up before we read and reset the voltage.

In a *bucket-brigade* device, we must apply a pulse sequence that shunts all the charges simultaneously into a second row of capacitors. Then a second sequence of pulses causes the charges to march along the second line of capacitors—living up to the title "bucket brigade." At the end of the chain, a sequence of voltages is seen, representing the light integrated by cell1, cell2, cell3, . . . and so on in turn. This kind of sensor needs an assortment of clock pulses to be accurately controlled in both timing and voltage. Fortunately, there are some other devices that are much easier to use.

The TSL214 and its successors use the same principle, in that the photoelectric current of each of their sensors is integrated in a capacitor. However, the method of reading the charge is much more robust and tolerant of timing and waveform variations. Each time a clock waveform rises and falls, a single "1" is moved along a shift-register. As the "1" reaches each cell, its capacitor charge is transferred to the output amplifier, so that a corresponding voltage appears on the output pin. One more input is needed, to insert the single "1" at the beginning of the sequence.

The chip runs from a single 5-V supply, and only two output bits from a microcomputer are needed to control it. It can easily be attached to the printer port of a PC, and the pulses can be generated by simple software. That leaves only the problem of inputting the brightness data to the PC.

An obvious option is to use an analog-to-digital converter (ADC)—perhaps an 8-bit device that can discriminate 256 levels of brightness. In many applications, though, the brightness needs to be compared only against a fixed level.

It is much easier in this case to perform the analog comparison in a single comparator and input a single bit per pixel. The method is a little less crude than it appears, since the sensitivity can be varied from within the software. By allowing more time to elapse between scans, a higher voltage will be obtained for a given brightness.

But we are already allowing this brief introduction to be complicated by considerations of interfacing.

2.3.3 Framescan Devices

These devices are built around a videocamera similar in principle to that of a camcorder. Some texts still refer to the vacuum-tube cameras that are now part of TV history. The "useful" technology is at present centered on a photosensitive silicon chip.

An array of photosensitive sites on the chip allow charge to build up on capacitors (also part of the chip) at a rate proportional to the light falling on each pixel, just as in the case of the linescan camera. By a variety of bucket-brigade operations, these charges are sampled in turn and appear at an output pin essentially in the form of a raster TV signal.

There is another, simpler form of camera termed a "Webcam." It usually has fewer pixels than does a videorecorder camera, typically 640 × 480. Its virtues are that it is cheap and that it comes with a complete interface to a personal computer.

A color TV display has a fine array of red, green, and blue phosphor dots, which have the combined effect of creating a full-color image. More expensive cameras use three chips, one to receive each primary color. The cheaper versions, including Webcams, use a single chip.

By printing a tinted pattern on the face of the camera chip, the manufacturers can produce an output that can be converted into a full-color picture, in exchange for some loss of resolution. Groups of 4 pixels are arranged with 2 of green and 1 each of red and blue (e.g., see Fig. 2.16). Since the eye resolves color with much less sharpness than it does brightness, the lower resolution of the color data is not a problem.

```
R G R G R G R G R G R G
G B G B G B G B G B G B
R G R G R G R G R G R G
G B G B G B G B G B G B
R G R G R G R G R G R G
G B G B G B G B G B G B
R G R G R G R G R G R G
G B G B G B G B G B G B
```

Figure 2.16 *Red-green-green-blue (RGGB) array.*

2.4 THE COMPUTER

Computers come in all shapes and sizes. Some are designed to be embedded in mechatronic products, while others are virtually complete products in themselves.

But the sophisticated PC and the simplest microprocessors have the same principles at heart. The essential components are memory and a *processor* consisting of an *arithmetic–logic unit* and a control unit. Bytes are "fetched" from the memory and treated either as data, numbers to be crunched, or instructions to be executed. The "cunning" of the computer (let us not yet call it "intelligence") lies in the ability to choose a different sequence order of instructions to obey, according to the value of the data.

So at rock bottom the program consists of a list of bytes to be executed. Switch on your Windows PC, click on START and RUN, and type "debug." Then, when a black panel appears, type

```
d f000:0
```

followed by clicking on <enter>

An array of two-digit numbers will appear, where the digits include the letters A–F. If the numbers are all zero, try inputting a different number to look at a different part of memory. When you have found something that looks interesting, try

```
u f000:0
```

or whatever number you have chosen.

Now a cryptic list of codes will appear on the screen, such as (although not the same as)

```
TEST CL,3F
JZ 0020
CMP AH,02
JZ 0025
```

You are looking at an example of *assembly code.*

The jumble of numbers was a dump of that part of memory, represented as *hexadecimal bytes*. Believe it or not, this is the most powerful sort of code. By planting the correct bytes, you can cause the computer to execute any instruction of which it is capable.

Since the very early days of computing, programmers have found it much easier to remember *mnemonics* than the raw codes. So CMP stands for "compare," representing hexadecimal 80 (in some cases).

A program called an "assembler" turns the lines of code into the corresponding bytes. It is obviously more productive for a programmer to write code using these mnemonics than to write hexadecimal bytes, but if there is an instruction that the assembler does not "know about," it is impossible to use it. (If you want to be adventurous, type **?** to see a list of instructions including an assembler, or else type **q** to quit and return to normal.)

For more substantial programming, the *compiler* is the method of choice. Code does not now correspond exactly to the *machine code* bytes, but instead represents the mathematical task being attempted. This is where the PC and the simple microprocessor start to part company.

Generations of PC software have become more and more massive, until programs of many megabytes are common. Embedded mechatronic tasks can often be performed by software consisting of only a few thousand bytes, so writing in assembly language is not out of the question. Even though C compilers are available for most microprocessors, the compactness and efficiency of code written in assembler can often be worth the extra effort.

On the other hand, there are very few embeddable microprocessors on which you would want to run the assembler or compiler software to convert your code into bytes. Instead, you will run a *cross-compiler* or *cross-assembler* on a PC to generate bytes that you will download to the microcomputer.

For your programming efforts, you need the home comforts of a keyboard, a mouse, and a screen, plus a disk on which to save the results. The embedded processor is unlikely to have any of these.

There is an alternative to downloading the code bytes to an embedded computer. They can be sent off to the "chip foundry" to be *mask-programmed* into the chip itself. Such chips might cost only a few cents each—but they must be purchased in batches of 10,000!

As I have mentioned, my first embedded computer product, many years ago, used the TMS1000, the chip that powered the very earliest pocket calculators. Its data memory consisted of just sixty-four 4-bit "nibbles"—a nibble is just half a byte. The entire program memory held just over 1000 instructions. Yet this chip had to function as the heart of a cooker clock, counting mains cycles to calculate elapsed time, switching the oven on and off in response to the set cooking time and ready time. It had to show each function on a *vacuum fluorescent display* and to respond to button presses by the user. It had to detect option switches to choose between 50- and 60-Hz mains frequency and between 12-h and 24-h displays. Would you not call that mechatronics?

There was not much fundamental difference between processors of the next generation. You might find the 6502 embedded in a simple controller or else at the heart of an early version of the personal computer such as the BBC Micro or the Apple II. Yet these machines could function as word processors and handle spreadsheets and databases at speeds that seemed hardly slower than those of the modern PC.

Over the years, the 16-bit processor took over the personal computer role, later being outstripped by 32-bit machines and more. Computing power increased by a factor of a 1000, yet software seemed to run no faster.

Most of the power was soaked up by embellishments of the graphics and by an operating system that tried to give the illusion of performing a thousand tasks at the same time. Microsoft realized that for every engineer who needed a machine for computer-aided design, there were a hundred youngsters who wanted a machine on which to play videogames. The Windows operating system has leaned more and more toward providing instant gratification as a music player, a Web-surfing machine, and as a digital television.

It's not all bad.

The move to multimedia has opened up the way to powerful videoprocessing tools. Streams of Webcam images can be captured, dissected with DirectX routines, and analyzed to extract their data. So there are applications where it is preferable to embed an entire PC than to use something simpler. Remember that the "guts" of a PC, minus disk, display, and keyboard, can be purchased as a single board for a very few hundred dollars. A Webcam interfaced to a computer board to process the signal will cost a small fraction of the price of a Pulnix camera alone.

2.5 INTERFACE ELECTRONICS FOR OUTPUT

Almost without exception, the task of getting from a computer's binary output to physical reality will involve responding to a signal that lies between 0 and 5 V. The actual range of output might be only from 0.5 to 3.5 V, and the circuit that it drives must not draw or source a current of more than 1 mA or so.

Something is needed to amplify this signal before we try to use it for moving a motor—or even lighting an LED.

For small signals, the *Darlington driver* has been popular for many years. It amplifies the computer output of a few milliamperes to 0.5 A, enough to drive a small relay or a simple stepper motor. A popular single chip contains eight such drivers, so that 8 output bits of a port such as the printer port could control two stepper motors.

2.5.1 Transistors

The theory of transistors can be ornamented with *hybrid-pi* parameters and matrices, but the essential knowledge for using them is much simpler. Let us start with the *bipolar* or *junction* transistor. This has three connections, a *collector*, a *base*, and an *emitter*. For driving a relay coil, you might connect a transistor as shown in Figure 2.17.

The emitter is connected to ground, the supply is connected to one end of the coil, while the other side of the coil is connected to the collector. As it stands, no current will flow and the coil will be off.

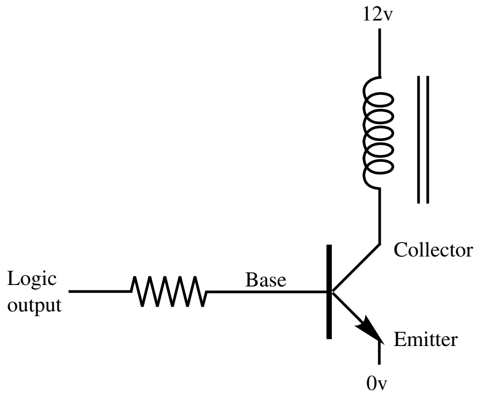

Figure 2.17 *Transistor and coil, with electrodes labeled.*

Now apply some current to the base, to flow through the transistor to the emitter. For every milliampere that is applied, the transistor will allow 100 mA to flow from the collector to the emitter. If it takes 500 mA to turn the relay coil on, we need supply only 5 mA to the base.

Of course, the factor is not exactly 100 for every transistor, but that is a ballpark figure. The actual figure is the *gain* of the transistor, defined by its parameter *beta*.

Not only do we get the current gain of this factor; we get a voltage gain as well. The current entering the base is resisted by a voltage of around 0.7 V. The voltage on the collector for some transistors can be hundreds of volts, but typically you might use 12 or 24 V to drive your relay or stepper motor.

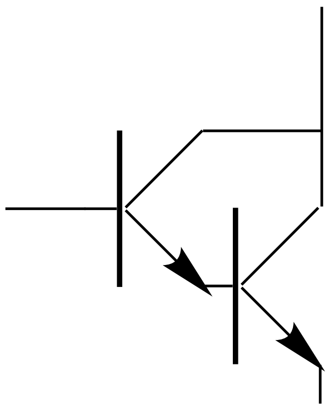

Figure 2.18 *Darlington driver transistor.*

So, if the task can be done with such a simple transistor, why do we bother to use a pair of transistors connected as a Darlington driver (Fig. 2.18)? We must look at the power applied to both the coil and the transistor. Consider a power supply of 12 V and a 24 Ω coil that will take 0.5 A when switched fully on.

When the transistor is off, there is no current and no dissipation. There will be 12 V between collector and emitter of the transistor. When the transistor allows 100 mA to flow, the voltage across the coil will be 24 × 0.1 = 2.4 V. That will leave 9.6 V across the transistor. With 100 mA passing through it, the dissipation in the transistor would be 9.6 × 0.1 = 0.96 W.

When the transistor allows 250 mA to pass, the voltages across coil and transistor will each be 6 V. Each will dissipate 1.5 W. Overheating of the transistor could certainly be a problem. But when the transistor allows the full 500 mA to flow, the voltage on the collector will have dropped to zero and the transistor dissipation will again be zero. To avoid overheating the transistor, we must be sure that it is *saturated*, that we are supplying enough current to the base to pass all the collector current available.

So, in the Darlington configuration, the emitter of the first transistor is connected to the base of the second, while both collectors are connected together. The base current of the second transistor is (1 + beta) times the current applied to the first, so the output current is some 10,000 times the input current—until the collector voltage has fallen too low to supply that much current to the second base. We can be sure that the driver will saturate for a very small input current.

But there are some transistors that can control their output current for no input current at all! These are *field effect* transistors (FETs). There are powerful ones known as *metal oxide semiconductor FETs* (MOSFETs). Once again, the mechatronic engineer need not be concerned with the physics that makes the transistor work, just with the details and pitfalls of using it.

The three connections are now known by entirely different names, *drain*, *source*, and *gate*, but conceptually we can drop the device into the same scenario as the former *junction* or *bipolar* transistor.

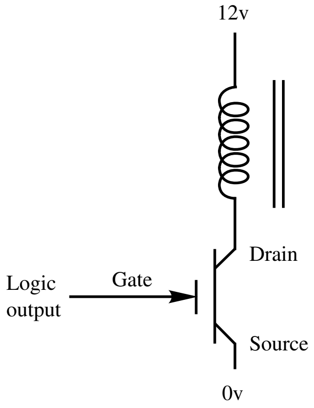

Figure 2.19 *FET driving a coil.*

Connect the source to ground and the relay coil to the drain, and we are ready to control the current with the gate. The essential difference is that when we apply 5V to the gate, no gate current flows at all! Meanwhile the drain has allowed all the available coil current to flow through the FET to ground, switching the relay coil fully on (see Fig. 2.19).

These devices make interfacing seem all too easy! They have another property that makes us spoiled for choice. Both bipolar transistors and FETs are available in *complementary* forms. N-channel FETs and NPN transistors are used with a positive voltage on the drain or collector and are turned on with a positive gate voltage or a positive base current. P-channel FETs and PNP transistors take a negative voltage on the drain or collector, while they are turned on with a negative gate voltage or base current.

There are several pitfalls to avoid. When a transistor or FET is switched from OFF to ON states, there is a brief burst of dissipation because the output voltage does not drop instantaneously. When it turns off, there is another burst of dissipation that can last somewhat longer. In general, this is not serious. If the controlling computer switches the drive on and off repeatedly at high speed, however, dissipation can become a problem.

A second problem can result from the inductance of the load, whether it is a coil or motor winding. As the current is turned off, there is a spike of voltage in a direction that tries to keep the current flowing. The use of a diode across the coil will avoid this voltage, but the current in the coil will fall more gradually.

So far we have considered simple on/off switching, but in the case of a DC motor we usually want to reverse the direction of the current in the motor. For that, a convenient solution is the H-bridge.

2.5.2 The H-Bridge

If we connect one end of the motor to ground, we would have to provide current from both a positive and negative supply to ensure two-way control of the motor. Instead, we can make do with a single supply if we can switch the connection of either end of the motor.

This results in a circuit in the form of an H, with the motor representing the crossbar and with two transistors in each of the "uprights": *A* and *B* in one and *C* and *D* in the other, as shown in Figure 2.20.

If transistors *A* and *D* are turned on, the motor will be driven one way. If, instead, *B* and *C* are turned on, the motor will be driven in the reverse direction. If *A* and *B* or *C* and *D* are turned on together, however, it will mean instant death for the transistors. Much of the detail of any H-bridge circuit (see example configuration in Fig. 2.21) will be there to prevent this from happening.

As with so many electronic circuits, the entire device can be purchased "on a chip" for a very few dollars. The L298 from ST Microelectronics (http://www.st.com, see also http://www.learn-c.com/l298.pdf) contains two

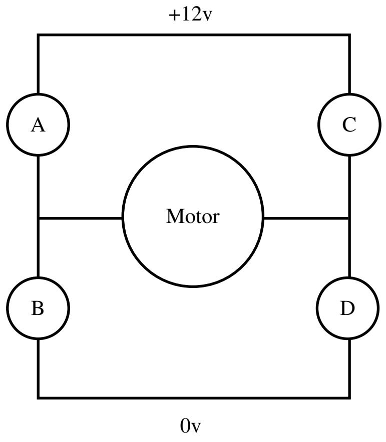

Figure 2.20 H-bridge.

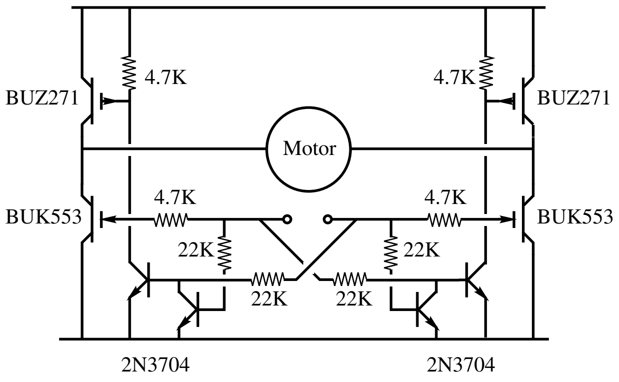

Figure 2.21 H-bridge circuit.

H-bridge circuits and will drive small motors with ease. For larger loads, however, you might need to design and build a more specialized circuit.

2.5.3 The Solid-State Relay

Another useful on-a-chip device is the solid-state relay. It contains a light-emitting diode and a photosensitive triac, with the result that a single output bit of the computer can switch an AC device on or off. The LED provides complete electrical isolation between the computer and the AC circuit.

Figure 2.22 shows an example in which a two-phase induction motor can be driven forward or in reverse. It was used with great success to automate the operation of the valves of a pilot sewage plant! Time has to be allowed for the motor to stop before reversing it, but the control was very leisurely.

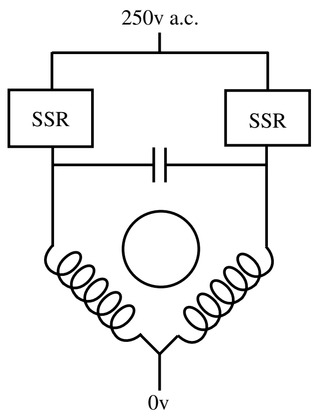

Figure 2.22 *Induction motor with two solid-state relays.*

Although undesirable, switching both drives on at the same time need not be disastrous.

2.6 INTERFACE ELECTRONICS FOR INPUT

We see that, in general, we can apply our control via one or more computer output bits. If we prefer a gradual change in drive, rather than a *bang-bang* on/off control, we can often use mark–space output. Instead of the output bit being on or off continuously, it is switched on for an interval that we can vary by software. The dynamics of the system being controlled will smooth this into a proportional signal.

For simple inputs, we can read logic values from input pins, both on the simplest of embedded microcontrollers and on a PC used as a controller.

Reading analog sensor values into the computer is a different matter. Now we can have a continuously varying voltage, such as a tachometer or potentiometer output, that we want the computer to be able to read to considerable precision. We need some means to convert this analogue signal to digital form.

2.6.1 Simple Inputs

Perhaps our inputs are merely logic signals. We can then arrange for them to pull one of our logic inputs to ground, maybe using a transistor to avoid applying the signal directly to a computer pin. If we are particularly concerned about the computer's safety, we can use an *opto isolator* (Fig. 2.23). The package, usually containing four channels or more, has a LED activated by the input signal and a phototransistor that conducts when the LED is lit. In this way, there is no electrical connection between the computer and the sensor circuit.

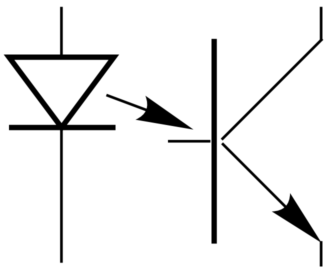

Figure 2.23 *Opto isolator.*

The printer port of the PC, in addition to its eight output data lines, has five logic inputs. Indeed, a single output command can also convert those eight output lines to inputs.

2.6.2 The Analog-to-Digital Converter

For reading an analog sensor, we need something more.

Analog-to-digital converters (ADCs) come in a variety of forms, from the very simple circuit that reads the movement of a game joystick to sophisticated systems that can "grab" waveforms at video speeds and above and pack the data into a block of memory.

The vast choice is one of the problems that beset the mechatronics system designer.

At the simplest level, a ramping voltage can be compared against the signal to be measured. The resulting number is obtained from a counter that measures the time taken for the ramp to reach the signal voltage. It is simple, but it is slow if any sort of precision is desired. It can be constructed from a few dollars' worth of components to connect to the printer port, if no other ADC can be found. A more detailed design is given in Section 5.3.5, together with simple software needed to drive it.

Many converters are built around a digital-to-analog converter (DAC), the counterpart of the ADC, that converts a number to an analog output. With this, it is possible to use a binary search method, termed *successive approximation.*

Suppose that the input can be in a range of 0–8 V and that the actual value is 6.5 V. First, the DAC output is set to 4 V, the halfway point, and a comparator circuit gives the answer to the question "Bigger or smaller?" As 6.5 V is bigger, so the next DAC value is 6 V, three-quarters or 75% of the maximum. The answer is still "bigger," so the next ADC value is 7 V, dividing the 6–8 V range in half. Now the answer is "smaller."

The sequence "bigger–bigger–smaller" becomes a binary number 110 that defines the value we are looking for. By successively dividing the range to search in half, a precision of 1 part in 4000 can be gained from just 12 tests.

In general, it is not necessary to know these details. A circuit deals with all the logic operations and presents its answer to the computer.

Many ADCs have *multiplexers* that allow us to select between a number of input channels. So we must start our software operation by outputting the desired channel number. This might also tell the device to start the conversion, or else we must issue a specific command. When the answer is ready, the status of an input bit will change and we can read the data. However this might not be as simple as it sounds.

If the ADC gives a 12-bit result, precise to 1 part in 4000, and if our input is an 8-bit byte, we have to find a way to read the answer in two parts and to combine these together. Rather than go into details here, see the code examples in Chapter 3.

For very high-speed data acquisition, there is the *flash converter*, which relies on brute strength and a large number of comparators to give an answer in a small fraction of a microsecond, so that video signals can be encoded.

Commercial interface cards are becoming more and more complicated—and more expensive, too. They could be very simple, indeed, containing little more than a single-chip ADC that can be purchased with 16 input channels for under $5. Instead, they often contain first-in first-out (FIFO) buffering for 1000 samples or offer conversion rates of many megasamples per second.

The ADCs that could be purchased in the late 1990s were simple and ideal for student interfacing, requiring no more than a dozen lines of code to drive them. There has been a more recent trend for cards to hide behind a megabyte of driver software and to expect the user to depend on some software environment such as Simulink. Nevertheless, a close look at the manual will reveal that they have the same essential structure.

It is unfortunate that the ADCs of the “sound card” cannot easily be exploited for online control. Found in just about every PC, they are designed to digitize the two channels of stereo audio at 44kHz. However, the audio signals are *AC-coupled*, meaning that steady voltages are lost.

Nevertheless, for laboratory experiments there is an alternative. The simple chip, the MCP3204 from Microchip Technology Inc., provides four channels of 12-bit ADC conversion for a few dollars. It communicates serially, in a way that can be connected directly to the parallel port of a PC with no other components whatsoever. It is described fully in Chapter 5, Section 5.3.4.

For another alternative, it is no great task to attach an embeddable microprocessor such as a PIC chip or the Motorola HC12 to the serial port of a PC. The tasks of ADC and control outputs can then be delegated to the microcontroller, at the cost of some slight delay in the data communication. Indeed, the entire control task can be performed by the microcontroller, but for software development or laboratory data logging, the home comforts of the PC remain very attractive.

Mark Phythian has designed just such a PIC-based ADC 4-bit output as well. Full details of the circuit and software are given in Section 11.3.1.

2.6.3 Other Inputs

In the case of a PC, we can, of course, use the computer's more general inputs for sensor data. The older serial computer mouse connects to a COM port that receives signals encoded in ASCII format. Inside the mouse, a chip does all the necessary decoding and analysis of the sensors that detect its movement and sends trains of 5 bytes at 1200 Hz to the serial port of the computer. Other versions of the mouse can be connected to a PS2 or USB port, but the principle is the same. The mouse chip does the hard work and sends processed data to the computer.

By reading and interpreting the mouse codes directly, we can bypass the acceleration that would prevent us from using them for absolute displacement measurement. For a few dollars, we have two channels of position sensor with resolutions of 0.1 mm.

Once we have delegated tasks to other devices, the choices become even more bewildering. Communication systems such as CANBUS are available at chip level and allow us to connect a whole network of devices together. We can use these devices as *autonomous agents*. We should not rule out Ethernet and other general networking systems, where protocols such as TCP/IP can allow devices to call each other up with all the ease of browsing internet pages.

2.6.4 Signal Conditioning and the Operational Amplifier

We may have a signal that is not suitable for connection to an ADC, possibly because the voltage is too small. The *operational amplifier* (Fig. 2.24) comes to the rescue.

The design of a linear transistor amplifier can be a tricky business, with the need to provide the correct bias currents and to match the gains of transistors to minimize the drift effects of changing temperatures. Fortunately, the makers of integrated circuits addressed this problem many decades ago. The operational amplifier sells for a few cents and can be configured to perform a wide range of tasks.

Apart from two power supply connections, in essence it has just three terminals: the output, the inverting input, and the noninverting input. It can be described by just one equation

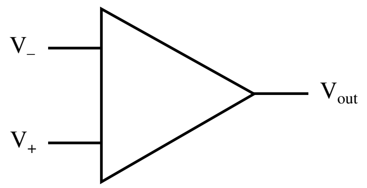

Figure 2.24 *Schematic representation of an operational amplifier.*

$$V_{\text{out}} = A(V_+ - V_-)$$

where A is a very large number, typically 100,000. Some earlier designs had additional connections for frequency compensation, but such considerations are unnecessary now except for unusual applications.

It is amazing that such a straightforward device can come in so many varieties. Some have more powerful output, some have exceptionally small drift, while others have an extended frequency response or rapid "slew rate."

A feature to look for is the ability to work between supply rails of 0 and 5V (many are at their best with +/− 15V) with an output voltage that can swing rail to rail.

Circuit design techniques are given in Chapter 5.

2.7 PRAGMATIC CONTROL

Control theory can be a mathematician's delight, blending *calculus of variations* and the *two-endpoint problem* with all things fuzzy, neural, genetic, chaotic, and catastrophic, with a generous helping of complex variables and matrices thrown in. In practice, the essence of designing a controller is an understanding of the dynamic behavior of the system to be controlled.

2.7.1 The PID Controller

The three terms in the three-term or "PID" controller are *proportional*, *integral*, and *derivative*. Over the decades its principles have been applied, starting with mechanical contraptions with ball-and-plate integrators or pneumatic devices worked by air power, all the way to the latest of the *Programmable Logic Controllers* sold as a universal cure for control problems.

We must start with the fundamental concept of *feedback*. First consider a system without it. The temperature of a directly heated bathroom shower could be controlled by a simple power controller wired to its electric heater. As the power is increased, the flowing water gets hotter. The knob of the power controller could be calibrated in degrees and the job would be done. At the mark on the knob labeled 45°C, let us suppose that 50% of full power is applied to give the desired temperature.

But what happens when a tap is turned on in the kitchen? The flow decreases by half while the same power is applied. The shower user lets out a scalded yelp. The temperature has risen to 70°C, as we will soon see.

Instead of relying on "open loop" control, we would like to measure the output temperature and adjust the power accordingly. This is feedback, and if we make the power proportional to the error, it is *proportional feedback*.

So now let us make the heater power proportional to the difference between the measured temperature and the voltage that is now given by the setting knob:

- Suppose that full power gives a rise in temperature of 50°C, so that each 1% increase in power will give us a rise of 0.5°C.
- Suppose also that the cold-water supply is at 20°C.
- Suppose that the target setting is at 45°C.
- Suppose that the *gain*, the factor by which we multiply the error to get the power setting, is G percent heating per degree of error.

Then, if the output temperature is T degrees, the percentage of power applied will be

$$(45-T)G$$

resulting in an output temperature

$$20+0.5(45-T)G$$

Now we can equate this to the temperature T, to get

$$T=20+0.5(45-T)G$$

which we can solve to get

$$T=(20+0.5G45)/(1+0.5G)$$

If $G = 2$, this gives an output temperature of (20 + 45)/2 or 32.5°C, a long way below the target. If $G = 20$, we get (20 + 450)/11 or 42.8°C, much closer.

But to see the real advantage of feedback, consider the effect of halving the flow. Now at half-flow, full power gives 100°C temperature rise. So if we relied on open-loop control applying half-power, we would get an output temperature of 70°C.

With closed-loop control and $G = 2$ we would get (20 + 90)/3 = 36.7°C. Turning on the kitchen tap has caused an increase of 4.2°C, which is much safer.

With $G = 20$, we get (20 + 900)/21 = 43.8°C, a jump of only one degree.

So, feedback can greatly reduce the effect of a disturbance, and as the feedback gain is increased, the error away from the target gets smaller.

Why not make the gain infinite? The *infinite-gain controller* can take the form of a simple thermostat switch that cuts the power as the temperature rises above 45°C. As the heater starts to cool, the temperature drops below 44°C, say, at which the system switches the power on again.

We could complain that the temperature is oscillating as the switch opens and closes in a *limit cycle*, but the one-degree variation is probably quite acceptable. Indeed, we could say with some justification that at least 90% of all control systems are unstable and oscillate, since that is probably the proportion of thermostats and similar controllers in everyday use.

Infinite gain is an option in this system, where the temperature can be measured very close to the heater, but there are many more systems where it is not. These may require a "smooth" output. Perhaps the necessary sensor signals with "immediate response" are not available. For numerous reasons, we may have to use more ingenuity in the controller.

The *proportional* feedback gain G accounts for the P in PID, but what of the other letters? If we were forced to measure the shower temperature further downstream, the oscillation might become rather serious. Oscillation in the shower temperature can often occur with human control. If the temperature is low, we increase the power. But the water, now too hot, flows through the hose and does not hit our skin sensor until a second or two later. When it does, we cut the power and suffer the further second of overly hot water, only to be hit by a chilling blast.

The solution is to reduce the size of our adjustment and inch the setting slowly up, waiting for the effect before making each new adjustment.

Consider an automatic temperature control system that oscillates unless we reduce the gain considerably, say, to 2 or below. Now the temperature error is larger than we can accept. How do we correct it? We can "integrate" the error, to get a term I that increases with time and with the error—this is *integral feedback*.

If we add this to the power drive, we will have an output temperature

$$20+0.5((45-T)G+I)$$

or after solving with $G = 2$

$$\begin{aligned}T &= (20+0.5G45+0.5I)/(1+0.5G)\\ &= (20+45+0.5{*}I)/2\end{aligned}$$

The temperature starts at first to settle at 32.5°C, but all the time it is below the target of 45°C, the value of I will increase steadily—although not so fast that it will cause oscillation. After a time (theoretically infinite), I will reach the value 50, which will cause the output temperature to be at 45°C—perfect!

But there is a catch. Suppose that the kitchen tap is turned on and the flowrate halves when the integrator has reached this value of 50. Now, with the flow halved, the 0.5 becomes 1.0, so the new equation for the temperature is

$$\begin{aligned}T &= (20+90+I)/(1+2)\\ &= (20+90+50)/3 \text{ since } I \text{ has not had time to change}\\ &= 53.3 \text{ degrees}\end{aligned}$$

which is probably still enough rise to cause a yelp of pain.

Integral control is a valuable addition for tuning a process plant to give a steady exact output, but is less useful when large disturbances can occur suddenly. That leaves us with the task of explaining *derivative feedback*.

Suppose that we are trying to control a motor by means of an input u that sets its acceleration. The differential equation that describes the motor is

$$\frac{d^2x}{dt^2} = -u$$

If we apply a feedback signal proportional to x, we have

$$\frac{d^2x}{dt^2} = -kx$$

But as we will see, this is the same equation that we get for a pendulum. For any positive value of k, the result will be an oscillation. (For any negative value, it will cause x to run away toward infinity!)

The D term in the controller estimates a derivative of x (assuming that we cannot actually measure one), and we will later see that it can make the system stable.

2.7.2 Understanding the Dynamics of a System

The theory taught in many courses is centered on the manipulation of transfer functions. But a real problem does not start with a transfer function. Instead, you are confronted by a physical system for which you have to derive the differential equations yourself before any kind of mathematical analysis can even start. What is more, the equations will certainly have some serious nonlinearities, such as the "full drive" limit of the acceleration of a motor or the "bottoming" of a spring shock absorber.

The most valuable contribution of "modern control theory" (well, it's still only half a century old) is the concept of the system's *state*. This is a set of properties such as position and velocity that define exactly what it is doing at any instant.

Let us consider a classical second-order system, a swinging pendulum. The acceleration of the bob is proportional to its displacement and is directed toward the center. In equation form, this becomes

$$\frac{d^2x}{dt^2} = -n^2x$$

If we differentiate $\cos(nt)$ twice, we get $-n^2 \cos(nt)$. Similarly, if we differentiate $\sin(nt)$ twice, we get $-n^2\sin(nt)$ The general solution is a mixture of sines and cosines of nt

$$A\cos(nt)+B\sin(nt)$$

with coefficients A and B that are determined by the *initial conditions*. These initial conditions will define the motion from that time on.

But what if we had measured the values of the initial conditions a moment later? The solution must still be the same. We can define any time throughout the swinging as our *start time* and use those initial conditions to define the future behavior of the pendulum.

So, those initial conditions are no longer quite so "initial," but carry forward all that we need to know about the past movement, in order to predict the future. Those initial conditions have become "state variables."

Each such variable has a rate of change that depends only on itself and other state variables, or on the input of the system if it has one. Our two variables in this case are the position and velocity of the bob, x and v.

The first rate-of-change equation is rather obvious:

$$\frac{dx}{dt}=v$$

The second equates the rate of change of v to the acceleration:

$$\frac{dv}{dt}=-n^2x$$

Instead of a second-order equation, we have two first-order equations.

For a simple position controller, there will similarly be two state variables, position and velocity. The first equation appears almost trivial—it just states that rate of change of position is equal to the velocity.

The second equation concerns the rate of change of the velocity, and will express it in terms of the input drive and the velocity itself if there is any kind of friction. Now we can concern ourselves with the way in which feedback will change the dynamics of the system.

Having worked out the equations, it is a matter of simplicity to write a few lines of computer code to simulate the system, nonlinearities and all. It is easy to add a few more lines to represent a control strategy and deduce the response of the closed-loop system.

If these methods look a bit cut and try, they can be given a mathematical gloss by using matrices to represent the equations when they are linear. The matrices can be manipulated to represent the closed-loop system, too. In a later section we will see that a routine mathematical operation gives an equation, the characteristic equation, from which the stability of the system can be assessed.

But that is getting ahead of the practicalities of designing a controller. Let us consider, for example, the control of the end effector of a robot.

2.7.3 Practical Design

The first task of the designer will have been to choose a motor and gearbox that will give an appropriate top speed and torque. Let us assume that sensors are in place to measure both the position and the velocity.

It can be shown that a simple second-order system such as this will be stable for any values of negative feedback of the two variables. But mere stability is not enough. We want a response that does not overshoot and a final position that is not deflected excessively if there are disturbing forces.

Linear theory assumes that if the error is doubled, the drive to the motor will also be doubled. But this is a real motor, with a real drive amplifier that has a limit on the output it can give. To be sure that the error is kept within bounds, we must ensure that this full drive is applied if the maximum acceptable error is exceeded. This error is termed the *proportional band*, the range of steady errors for which the drive will be proportional to the error. Outside this band, the drive will saturate.

How much velocity should we feed back? The simple answer is: "Enough to avoid an overshoot." From the largest likely initial error, the motor must accelerate toward the target, but start to decelerate soon enough that the load can come to rest without overshoot. As the speed builds up, the contribution of the velocity feedback must cancel out the position term by the halfway point.

Pragmatic design such as this is greatly at odds with the linear approach that is conventionally taught. But when the system is dominated by a nonlinearity such as drive saturation, the linear approach is no longer appropriate. That is not to say that we cannot apply analytic tools to the task, as you will see later in the book.

2.8 ROBOTICS AND KINEMATICS

Machine tools were traditionally built around slideways and pivots, so that in a lathe the work was spun about an axis while a cutter moved in straight lines to carve it into cylindrical sections. Generally, just one of the controlled axes would be adjusted at a time. The involvement of a computer has meant that robot manipulators could be freed from the need to move in straight lines and circles.

Now we are faced with a robot with six rotary joints, linking arm segments in series in a way that can in theory reach any point in its workspace with a toolpiece held at any angle.

The art of calculating the *end effector* position and orientation from the six angles of the axes is termed *kinematics*. Some elegant matrix methods that we will meet later have been developed to ease the calculations. But that is only half the problem—and the easy half at that.

If we know where we want to put the tool, we need a way of calculating the axis angles that will achieve it. This is a much less straightforward calculation called *inverse kinematics*. We will also be interested in the relationship between the velocities of the axis angles and the velocity of the toolpiece. This is where *dynamics* comes in.

For all except the simplest of mechanisms, these problems are likely to get mathematically intense. This is a good place at which to round off the bare essentials and dig deeper into the toolbox.