So far in our discussion of programming the Arduino microcontroller, we have generally glossed over the physical interactions of turning things on and off and sensing the world around us that make programming microcontrollers so much fun. Now it’s time to fix that. Programming a microcontroller can be rather unique because of its ability to easily turn on and off motors, lights, and other actuators, as well as read from switches and sensors. This is possible on the Arduino Uno by using its 20 input and output (I/O) pins, each having different functions that can be configured in software—including input, output, analog to digital conversion (ADC), and analog output, more accurately called pulse width modulation (PWM)). This chapter will take as its focus the various digital input and output functions available and how to interface the needed hardware to make them work.

What unifies the functions and code in this chapter as being digital is that they all have only two different states: on or off. We have actually already used the Arduino’s digital I/O from the very beginning of this book and briefly introduced digital input in the last chapter using a very simple form of a digital sensor in the basic rolling ball tilt switch. Other forms of digital input include pushbuttons, mat switches, magnetic or reed switches, thermostats, passive infrared motion detectors, and oh so many more. We can use digital output to turn on not only LEDs but also to make sound, create movement with motors or solenoids, and even control larger devices using transistors and relays.

We will start with an introduction of the electrical properties of the Arduino’s I/O pins, followed by a new project called Noisy Cricket, and then we will move into a thorough examination of the digital functions and how we can incorporate them into our code.

What’s needed for this chapter:

- Arduino Uno
- Passive infrared motion detector
- Piezoelectric speaker
- Momentary pushbutton or switch
- 10 kilohm ¼ watt resistor or similar
- Hookup wires
- Solderless breadboard
Arduino I/O Demystified

The Arduino interface board has a single row of connectors or pins on each side of the board. These are used mostly for inputs, outputs, and power. Figure 5-1 shows a simple illustration of the Arduino Uno interface board. Your board might be a little different.

As shown in Figure 5-1 and printed on the interface board, there are 14 pins on one side of the board marked Digital that are numbered from 0 to 13. Of these, two are used for serial communications through USB for programming, debugging, and other forms of communication. These pins are numbered 0 and 1, and are marked RX <- and TX -> respectively. Anything connected to these pins may affect or be affected by serial communications. Pin 13 has a resistor and LED (marked L) connected to it on board so that it can be used for testing, diagnostics, and other forms of blinking. Six of the digital pins can also act as analog outputs using pulse width modulation (PWM). These are pins 3, 5, 6, 9, 10, and 11 on the Arduino Uno and are marked on the interface board with the tilde symbol (~). On the other side of the board are six additional I/O pins labeled as Analog In, numbering A0 through A5. These pins are connected to the microcontroller’s analog to digital convertor (ADC) for interpreting incoming analog signals, although they can also be used as additional digital inputs and outputs.

Each I/O pin operates on a range of 0 to +5 VDC, or volts direct current. A range of 0 to +2V is said to be off or LOW while anything over about +3V is said to be on or HIGH. Generally, a circuit connected to one of the Arduino I/O pins needs to have a high or positive side connected to power (+5 VDC), some form of load, and a low or negative side connected to ground (GND). How a circuit is physically connected to the Arduino or in which direction determines how the I/O pin can be used and which side of the circuit is connected to the Arduino. Each pin can either source, or provide a positive biased current, or sink, to provide a negative biased current, up to 40 milliamps each. Typically, most Arduino circuits are designed so that they source current, as shown in Figure 5-2, although there are times where sinking current is necessary, as shown in Figure 5-3.