In this chapter you build a robot (the Bug-Bot) that uses *bump sensors* to navigate around objects and explore a room, much like an insect would. The bump sensor is nothing more than two wires pressed together when the bot bumps into something, to let the Arduino know that it needs to find a new path (Figure 7-1). These sensors are commonly used for object detection but require that the bot have physical contact with the object to detect it. Some robots incorporate several types of sensors for object avoidance and autonomous navigation; these usually include a bump sensor on the front and/or rear of the bot to stop it if the other sensors do not.

*Figure 7-1. The visible components of the Bug-bot: three bump sensors on the left, a battery and Arduino clone in the center, two antennae sensors on the right, and two hobby Servo motors with wheels attached beneath a simple, homemade plexiglass frame.*
The simplest of bump sensors can be constructed using a piece of wire, a nut, and a bolt. To illustrate this, I decided to build the bump sensors for the front of the Bug-bot from scratch. I then used SPST switches for the rear sensors, which have aluminum bumpers attached to them.

**Reading a Switch with Arduino**

The majority of the code for this project involves reading the digital state of an Input pin. Recall from Chapter 1 that a digital Input can be read only as either HIGH (+5v) or LOW (GND). You can read the state of a digital switch from any Arduino pin using the `digitalRead(pin)` command. The default state of each pin can be either HIGH or LOW, depending on how you want to interface the switch. In this project, I focused on using as few parts as possible and keeping the design simple.

Connecting each bump sensor switch is simple: one pole of the switch connects to the Arduino Input pin; the other connects to GND. The Arduino has internal 20k ohm pull-up resistors on each pin that can be activated in the code if needed. To minimize the amount of parts needed (no extra resistors to buy), you can utilize these internal resistors to keep the Input pin HIGH unless the bump switch connects it to GND. Although it might make more sense (to me at least) to keep the input LOW using a pull-down resistor and only read HIGH (+5v) if the switch is activated because it would require slightly less power (but the Atmega168 pins are capable of holding only an input pin HIGH and not LOW), you must utilize the Arduino’s internal pull-up resistors to avoid using external resistors.

To use the Arduino’s internal pull-up resistors, you simply declare the pin as an INPUT using `pinMode()` in the setup() function; then `digitalWrite` the pin HIGH. With the pull-up resistor enabled, digital pin 2 in Listing 7-1 is HIGH by default unless driven LOW by some other source (that is, the bump switch). You can test the following example by running it on your Arduino and viewing the Serial monitor. Connect pin 2 to GND to see a 0 reading; otherwise, it reads 1.

**Listing 7-1. Reading a Contact Switch**

```cpp
// Code Example 7-1: Reading a contact switch
// Switch wires should connect to GND and Arduino digital pin 2
// The LED on pin 13 will stay On, unless pin 2 is connected to GND
// The variable "button_state" will hold the value of the button
// The button value can be either HIGH (1) or LOW (0)

int LED = 13;  // use LED connected to pin 13
int button = 2; // use pin 2 to read the button switch
int button_state; // use this variable to hold the value of "button" (pin 2).

void setup(){
    Serial.begin(9600);  // Start serial monitor at 9600 bps.
    pinMode(LED, OUTPUT);   // declare "LED" (pin 13) as an OUTPUT.
    pinMode(button, INPUT);   // declare “switch” (pin 2) as an INPUT.
    digitalWrite(button, HIGH);   // Enable the internal Pull-up resistor on Pin 2.
}

void loop(){
    button_state = digitalRead(button); // read the button

    if (button_state == 0){       // if the button_state is equal to 0 (LOW),
        Serial.println("LOW");     // then serial print the word "LOW"
        digitalWrite(LED, LOW);     // and turn the LED on pin 13 Off.
    }
}
```