Chapter 4

Reading Sensors

Solutions in this chapter:

Introduction

Motors, through gears and pulleys, provide motion to your robot; they are the muscles that move its legs and arms. The time has come to equip your creature with *sensors*, which will act as its eyes, ears, and fingers.

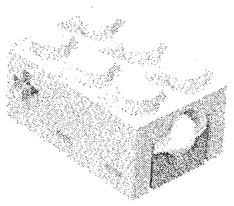
The MINDSTORMS box contains two types of sensors: the *touch sensor* (two of them) and the *light sensor*. In this chapter, we'll describe their peculiarities, and those of the optional sensors that you can buy separately: the *rotation sensor* and the *temperature sensor*. All these devices have been designed for a specific purpose, but you'll be surprised at their versatility and the wide range of situations they can manage. We will also cover the cases where one type of sensor can *emulate* another, which will help you replace those that aren't available. Using a little trick that takes advantage of the infrared (IR) light on the RCX, you will also discover how to turn your light sensor into a sort of radar.

We invite you to keep your MINDSTORMS set by your side while reading the chapter, so you can play with the real thing and replicate our experiments. For the sake of completeness, we'll describe some parts that come from MINDSTORMS expansion sets or TECHNIC sets. Don't worry if you don't have them now; this won't compromise your chances to build great robots.

Touch Sensor

The *touch* sensor (Figure 4.1) is probably the simplest and most intuitive member of the LEGO sensor family. It works more or less like the push button portion of your doorbell: when you press it, a circuit is completed and electricity flows through it. The RCX is able to detect this flow, and your program can read the state of the touch sensor, **on** or **off**.

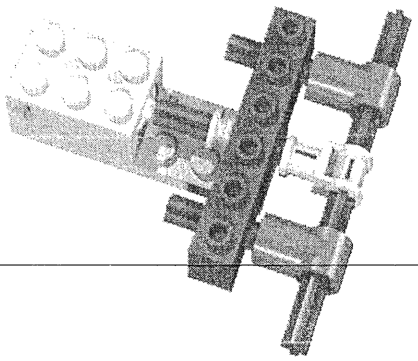
Figure 4.1 The Touch Sensor



If you have already played with your RIS, read the Constructopedia, and built some of the models, you're probably familiar with the sensors' most common application, as *bumpers*. Bumpers are a simple way of interacting with the environment; they allow your robot to detect obstacles when it hits them, and to change its behavior accordingly.

A bumper typically is a lightweight mobile structure that actually hits the obstacles and transmits this impact to a touch sensor, closing it. You can invent many types of bumpers, but their structure should reflect both the shape of your robot as well as the shape of the obstacles it will meet in its environment. A very simple bumper, like the one in Figure 4.2, could be perfectly okay for detecting walls, but might not work as expected in a room with complex obstacles, like chairs. In such cases, we suggest you proceed by experimenting. Design a tentative bumper for your robot and move it around your room at the proper height from the floor, checking to see if it's able to detect all the possible collisions. If your bumper has a large structure, don't take it for granted that it will impact the obstacle in its optimal position to press the sensor. Our example in Figure 4.2 is actually a bad bumper, because when contact occurs, it hardly closes the touch sensors at the very end of the traverse axle. It's also a bad bumper because it transmits the entire force of the collision straight to the switch, meaning an extremely solid bracing would be necessary to keep the sensor mounted on the robot.

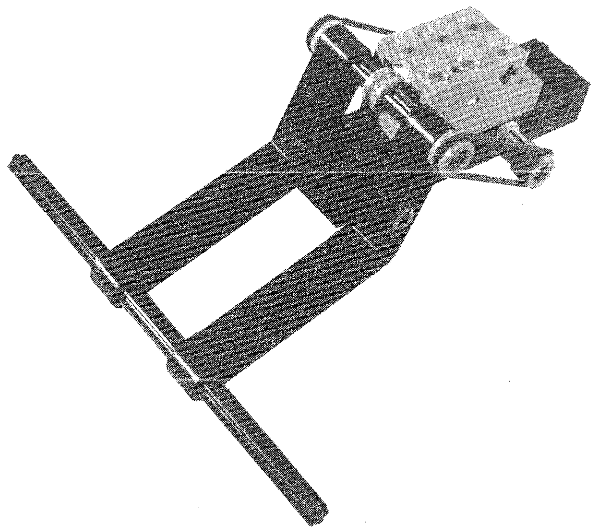
Figure 4.2 A Simple Bumper



Be empirical, try different possible collisions to see if your bumper works properly in any situation. You can write a very short program that loops forever, producing a beep when the sensor closes, and use it to test your bumper.

When talking of bumpers, people tend to think they should *press* the switch when an obstacle gets hit. But this is not necessarily true. They could also *release* the switch during a collision. Look at Figure 4.3, the rubber bands keep the bumper gently pressed against the sensor; when the front part of the bumper touches something, the switch gets released.

Figure 4.3 A Normally Closed Bumper



Actually, there are some important reasons to prefer this kind of bumper:

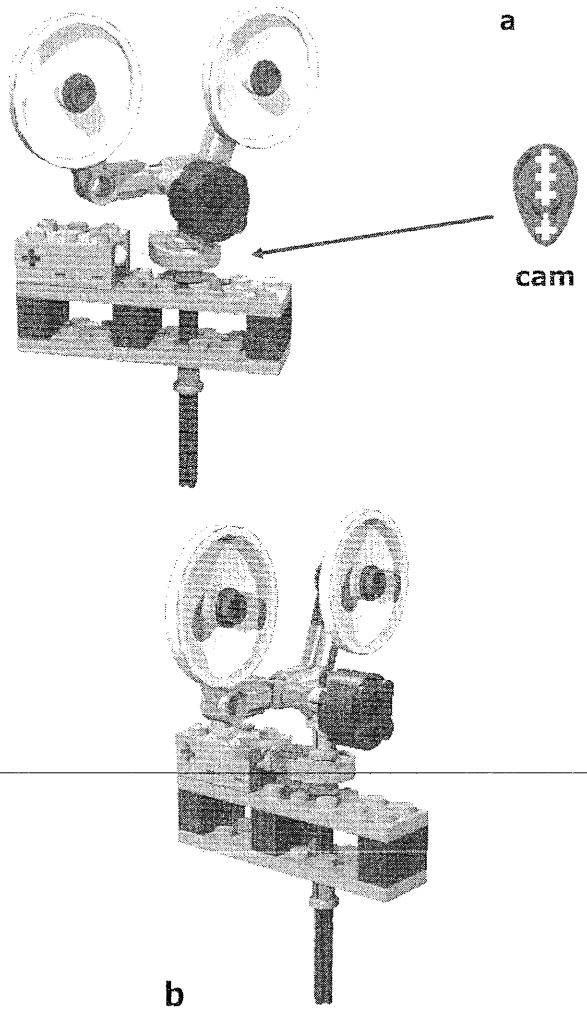
- The impact force doesn't transfer to the sensors itself. Sensors are a bit more delicate than standard LEGO bricks and you should avoid shocking them unnecessarily.
- The rubber bands absorbing the force of the impact preserve not only your sensor but the whole body of your robot. This is especially important when your robot is very fast, very heavy, very slow in reacting, or possesses a combination of these factors.

Bumpers are a very important topic, but touch sensors have an incredible range of other applications. You can use them like buttons to be pushed manually when you want to inform your RCX of a particular event. Can you think of a possible case? Actually, there are many. For example, you could push a button to order your RCX to "read the value of the light sensor *now*," and thus calibrate readings (we will discuss this topic later). Or you could use two buttons to give feedback to a learning robot about its behavior, *good* or *bad*. The list could be long.

Another very common task you'll demand from your switch sensors is *position control*. You see an example of this in Figure 4.4. The rotating head of our robot

(Figure 4.4a) mounts a switch sensor that closes when the head looks straight ahead (Figure 4.4b). Your software can rely on timing to rotate the head at some level (right or left), but it can always drive back the head precisely in the center simply waiting for the sensor to close. By the way, the *cam* piece we used in this example is really useful when working with touch sensors, as its three half-spaced crossed holes allow you to set the proper distance to close the sensor.

Figure 4.4 Position Control with a Touch Sensor



There would be many other possible applications in regards to position control. We'll meet some of them in the third part of this book. What matters here is to invite you to explore many different approaches before actually building your

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase® Smarter legal research.