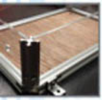
ARDUINO™ for BEGINNERS

ESSENTIAL SKILLS EVERY MAKER NEEDS

Loaded with full-color step-by-step illustrations!



JOHN BAICHTAL

que

ARDUINO FOR BEGINNERS

ESSENTIAL SKILLS EVERY MAKER NEEDS

John Baichtal

que®

800 East 96th Street,
Indianapolis, Indiana 46240 USA

Arduino for Beginners: Essential Skills Every Maker Needs

ISBN-13: 978-0-7897-4883-6
ISBN-10: 0-7897-4883-5

Library of Congress Control Number: 2013946136

Printed in the United States of America

First Printing: November 2013

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Arduino is a registered trademark of Arduino, www.arduino.cc/.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearsoned.com

Editor-in-Chief
Greg Wiegand

Executive Editor
Rick Kughen

Development Editor
Rick Kughen

Managing Editor
Sandra Schroeder

Senior Project Editor
Tonya Simpson

Copy Editor
Paula Lowell

Indexer
Lisa Stumpf

Proofreader
Sarah Kearns

Technical Editor
Pete Prodoehl

Publishing Coordinator
Kristen Watterson

Book Designer
Mark Shirar

Compositor
Mary Sudul

Contents at a Glance

Table of Contents

About the Author

John Baichtal got his start writing blog posts for *Wired*'s legendary GeekDad blog as well as the DIYer's bible *MAKE Magazine*. From there, he branched out into authoring books about toys, tools, robots, and hobby electronics. He is the co-author of *The Cult of LEGO* (No Starch) and author of *Hack This: 24 Incredible Hackerspace Projects from the DIY Movement* as well as *Basic Robot Building with LEGO Mindstorm's NXT 2.0* (both from Que). Most recently he wrote *Make: LEGO and Arduino Projects* for MAKE, collaborating with Adam Wolf and Matthew Beckler. He lives in Minneapolis, MN, with his wife and three children.

Dedication

For Harold Baichtal
1939–2013

Acknowledgments

I want to thank my loving wife, Elise, for her patience and support; all my hacker friends, for answering my endless questions; my mother, Barbara, for working on the glossary; and my children, Eileen Arden, Rosemary, and Jack, for their curiosity and interest.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Que Publishing
ATTN: Reader Feedback
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at quepublishing.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

Introduction

When you go to a store and buy an electronic gizmo, does it ever occur to you that you could make one yourself? Or even that it would be FUN to make one yourself?

The learning curve can be intimidating. You have to study electronics, learn what all the components do, and how to control them with a microcontroller. To put the components together, you'll have to learn how to solder. To program the microcontroller, you'll have to learn how to code. To make a cool container that holds the electronics, you'll have to master certain workshop skills.

Sound intimidating?

One bit of technology that makes these dreams not only achievable but enjoyable is the Arduino, a small microcontroller board designed to be easy to learn and a breeze to program. It lets you operate motors and take input from sensors, allowing you to build the project you want to!

The goal of this book is to help you create those projects—not just the gizmo, but the enclosure as well. You'll learn workshop skills, familiarize yourself with a ton of tools, build stuff. All of these projects use the easy-to-learn Arduino UNO microcontroller.

This book has been a huge learning experience for me, and I hope it is for you as well. You can build a lot of cool things with an Arduino, and the projects in this book are just the beginning. Good luck and have fun!

What's in This Book

This book is designed to take an absolute beginner and bring him or her up to speed on a large number of topics related to electronics, tools, and programming.

- Chapter 1 is called "Arduino Cram Session" because it drops a bunch of information on Arduinos and electronics—just what you need to start creating!
- Chapter 2, "Breadboarding," walks you through actually creating an electronics project—a laser trip beam!—using a handy piece of equipment called a solderless breadboard.
- Chapter 3, "How to Solder," teaches you how to use a soldering iron to connect electronic components. The chapter's project involves adding an LED light strip to a coffee table.
- Chapter 4, "Setting Up Wireless Connections," introduces you to three different ways that you can control a project with wireless signals. When you're finished learning about that, you can tackle the chapter's project, creating a wireless doorbell.
- Chapter 5, "Programming Arduino," shows you the basics of controlling your Arduino with programs you upload to the board. I'll take you line by line through an Arduino program so you can learn how it works.

- Chapter 6, "Sensing the World," describes a variety of sensors and explains the difference between digital and analog sensors. Chapter 6's project is a mood lamp that changes its colors depending on the environment around it.
- Chapter 7, "Controlling Liquid," shows readers three ways to pump liquid, and then puts one of these techniques to the test by showing how to build a plant-watering robot.
- Chapter 8, "Tool Bin," is a crash course on tools, everything from the ultimate toolbox to what to stock a wood or metal shop.
- Chapter 9, "Ultrasonic Detection," talks about using pulses of inaudible sound to map out obstructions and measure distances. The chapter's project, a cat toy, waggles a pompom intriguingly above your cat's nose when the sensor detects her.
- Chapter 10, "Making Noise," shows you how to make delightful electronic music (also known as noise!) generated by your Arduino. The project shows you how to build a hand-held noisemaker of your very own.
- Chapter 11, "Measuring Time," explains three ways in which the Arduino can keep track of time. Then I show you how to build an "indoor wind chime" that strikes on the hour.
- Chapter 12, "Safely Working with High Voltage," shows you three ways to deal with wall current safely. You'll build a sweet lava lamp controller that starts and stops the lamp on a schedule, plus you can trigger it with a remote control.
- Chapter 13, "Controlling Motors," explains motor control options for the Arduino. Then you'll build a bubble-blowing robot with your newfound skills!

Who Can Use This Book

This book is intended for persons new to making Arduinos. It assumes very little knowledge on the part of the reader; the only mental attributes needed are a sense of curiosity and a desire to tackle new challenges.

How to Use This Book

I hope this book is easy enough to read that you don't need instructions. That said, a few elements bear explaining.

Tip

Tips are helpful bits of advice that will save you time and/or headaches.

TIP

This is a Tip that provides helpful advice that I have learned along the way.

Note

Notes are tidbits of useful information that are helpful, but not mission critical.

NOTE

This is a Note that provides information that's useful, even if it is somewhat ancillary.

Caution

Cautions point out pitfalls and dangers. Don't skip these unless you like breaking things and spending time in the ER.

CAUTION

This is a Caution. You shouldn't skip these! The safety of your hardware, tools, and possibly your flesh depend on it.

Parts Lists

For each project in this book, I provide a shopping list of parts, such as the following, that you'll need to complete it.

PARTS LIST

- Arduino
- Servo (I used a HiTec HS-322HD servo, Jameco P/N 33322.)
- Servo horns (A number of horns come with the HiTec; these should be fine.)
- Chronodot RTC Module
- 1/4" dowel (You'll need about 8" to a foot.)
- Wind chime (I used a Gregorian Chimes Soprano wind chime, SKU 28375-00651.)
- 5mm plywood for the enclosure

- 1" pine board for the support blocks
- Eye bolt and nut (The Home Depot P/N 217445)
- #8 × 1/2" wood screws
- #6 × 2" wood screws
- #4 × 1/2" wood screws
- 24 1/4" × 1 1/2" bolts with locking washers and nuts
- 12 1/4" × 1" bolts with locking washers and nuts
- Drill press and a variety of drill bits
- Chop saw
- Table saw

Code

When a project requires code—or a sketch—I list it exactly as you should type it. However, unless you just like typing, you don't need to re-key the code found in this book. I've placed the code online so that you can easily download it, and then copy and paste it. Chapter 5 will get you up to speed on programming your Arduino.

Go to https://github.com/n1/Arduino-For-Beginners to download this code and other files associated with this book.

Here is a sample code listing:

```
int valve = 13; // renames Pin 13 “valve”

int offhours = 0; // how many hours before the water dispenses?
int offmins = 1; // how many minutes before the water dispenses?
int spray = 10; // number of seconds the water sprays

void setup() {
  pinMode(valve, OUTPUT);  // designates the valve pin as “output”
  Serial.begin(115200);
}

void loop() {
  int wait = (offmins * 60000) + (offhours * 3600000); // computes milliseconds

  digitalWrite(valve, HIGH);
  delay(spray * 1000); // water stays on this number of milliseconds
  Serial.println(offmins * 60000); // I used this when debugging
  digitalWrite(valve, LOW);
  delay(offmins * 60000); // water stays off this number of milliseconds
}
```

4

Setting Up Wireless Connections

This chapter explores the wireless networking tools that enable two or more Arduinos to talk together. Chief among these is the XBee, an Arduino-friendly wireless module capable of connecting a whole network of microcontrollers. In Figure 4.1, you can see one of my own projects, a LEGO robot controlled with Wii nunchucks connected to XBee-equipped Arduinos. You can learn how to build it in my book, *Make: Lego and Arduino Projects* (ISBN 978-1449321062). After you get up to speed on the XBee, you will tackle the third project, a wireless doorbell!

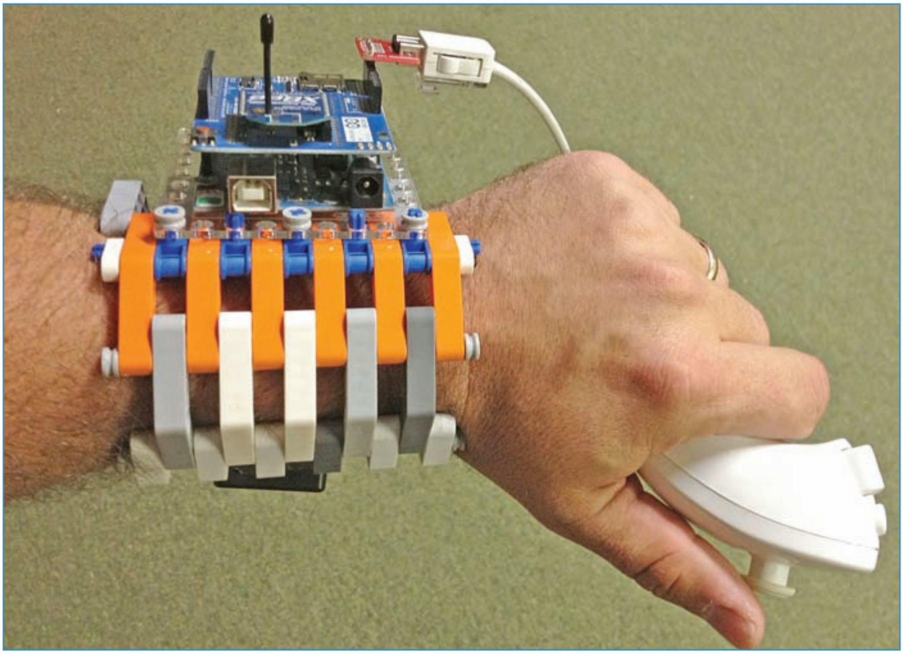


FIGURE 4.1 This XBee-equipped bracer enables you to control a robot wirelessly.

XBee Wireless Modules

XBee modules (see Figure 4.2) are based on ZigBee, which is an industry standard protocol that creates networks of multiple wireless nodes via serial data transmission, meaning only one bit (0 or 1) is sent at a time, making it slow but easy to configure. ZigBee is the default protocol used in home automation, so learning the platform's ins and outs could aid you in creating your own curtain-puller or light-switcher!

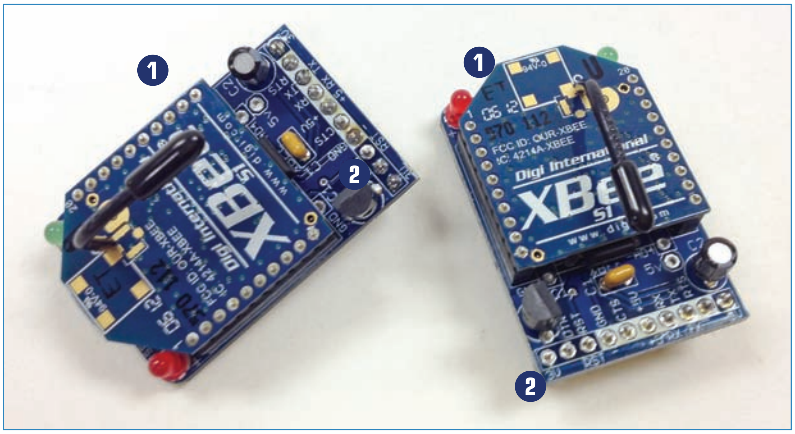


FIGURE 4.2 Two Series 1 XBee modules attached to Adafruit breakout boards.

XBee also happens to be the default communication method used by Arduino, enabling them to work together nicely. However, a wide assortment of XBee flavors are available, and you must sure to get the right one. Let's focus on just four of those XBee flavors in this chapter:

- XBee
- XBee Pro
- XBee Series 1
- XBee Series 2

XBee Versus XBee Pro

You first need to choose between XBee "regular" and "professional"—the distinction is purely about radio power. Ordinary XBees feature 1mW (one thousandth of a watt) power, whereas Pros are rated at 63mW, giving you a much greater range. What kind of range exactly? It depends on a complicated array of factors, including electromagnetic interference, antenna type, and physical obstructions.

That said, Digi International, the maker of XBee products, issues range estimates for the various models. The regular 1mW XBee is rated for 80 feet indoors and 300 feet outdoors, and the company claims the Pro model is good for 140 feet indoors and an impressive 4,000 feet—almost a mile—outdoors. Of course, for that last number, you would need the most ideal circumstances, like beaming from one hilltop to another. Any sort of obstruction will reduce the effective range of your radio.

If you don't need 4,000 feet, you might be better off skipping the Pro model because it costs more.

Series 1 Versus Series 2

The second consideration in choosing an XBee is what sort of networking you would like to configure. Digi International sells what it describes as Series 1 and Series 2 XBees.

- **Series 1**—Series 1 offers the simplest networking setup in that you don't have to set it up. Basically, every Series 1 module talks to every other Series 1 module within range—a configuration known as the mesh network. It's an easy way to get started playing around with wireless technology.

  If you want to direct data to a single module, you have to use software to set an identifier during both transmission and reception. This sounds intimidating, but it can be as simple as adding a single digit. Say you want to send data to Node 5; you can add a 5 to the beginning of your stream of data and the other nodes will ignore it.
- **Series 2**—Series 2 is more robust, offering—in addition to the settings of the Series 1—the ability to
  - Create more intricate networks with nodes being designated as "coordinators," able to issue commands.
  - Create "routers" that send and receive data.
  - Create end devices that may only receive.

  On the downside, having all these features means that you can't plug-and-play, because you must configure the modules before using them, unlike Series 1, which you can use right out of the box! More technically, the Series 2 use a different wireless protocol that makes them incompatible with Series 1 modules, so don't even try!

XBee Breakout Boards

XBee modules are easy to use, but they require a little love before they will fit into a typical Arduino project because their pin spacing is 2mm instead of Arduino-compatible 0.1". The solution is a small PCB called a *breakout board*, a way of creating a tiny circuit that can be plugged in to an Arduino.

The wimpiest of these is simply a PCB (printed circuit board) equipped with pins with the right spacing for breadboarding. However, more robust breakout boards, such as Adafruit's (P/N 126, previously shown in Figure 4.2), have a voltage regulator and status LEDs to keep your radio from getting fried.

Anatomy of the XBee

If you look at an XBee module, shown in Figure 4.3, it looks like a blue plate the size of a postage stamp, with a number of metal pins sticking out underneath. The top features an antenna. Adding it to a breakout board makes for more detail, so let's go through the XBee's various features.

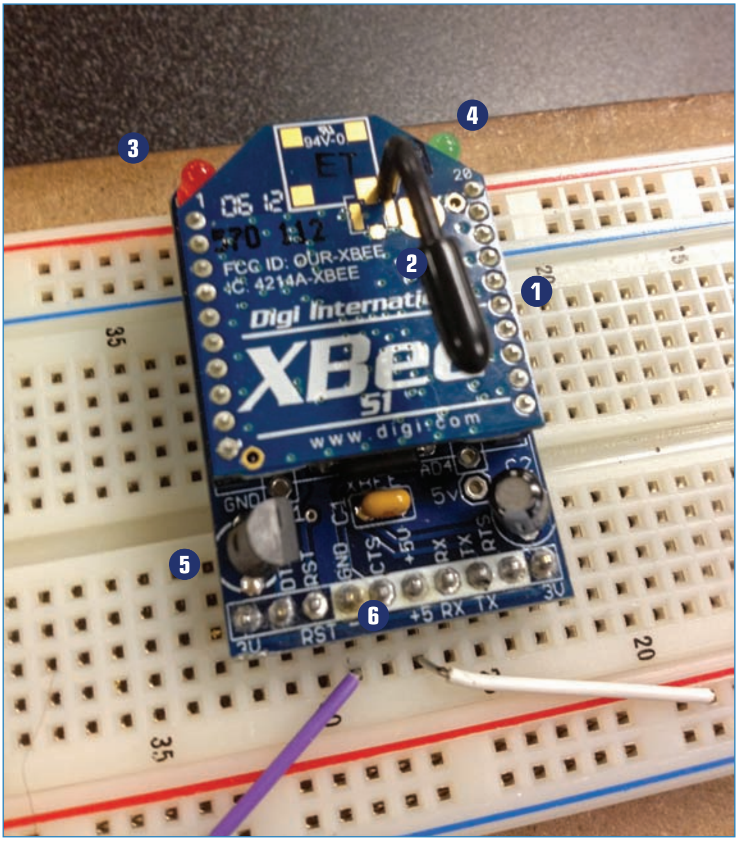


FIGURE 4.3 The XBee and its breakout board breadboarded up. Note that the 5V and GND pins are already connected to the proper terminal buses.

1. **Pins**—You can see the tops of the XBee's pins. They control the board, bringing in power and sending and receiving data from the Arduino. The pins plug into headers on the breakout board. Note that these pins have the wrong spacing for breadboards.
2. **Antenna**—You have multiple antenna options depending on the XBee, but I think this wire antenna is the best for what it does, because it's tough and can take a modest amount of abuse without bending.
3. **Power LED**—This lights when the board powers up.

4. **Data LED**—This flashes to let you know that data is passing through the XBee.
5. **Power regulator**—These capacitors and the transistor manage the power going into the XBee. Unfortunately, frying a radio by using too much power is easy to do. The good news is that the regulator keeps the power flowing at just the right voltage.
6. **Breadboard pins**—Unlike the pins that connect the XBee to the breakout board, these pins are spaced correctly for a breadboard. Just as good, they are labeled so you can see which pin does what!

Competing Wireless Modules

It probably doesn't surprise you that the XBee isn't the only party in town. Here are a couple of cool alternatives that you can purchase for use in a project.

Freakduino Chibi

Created by Tokyo-based hacker Akiba (a.k.a. Chris Wang), the Chibi (see Figure 4.4) does away with the separate boards for the microcontroller and wireless module—Akiba has combined them into a single board. The Chibi is Arduino compatible and uses the same wireless band as the XBee. You can buy it at www.freaklabsstore.com.

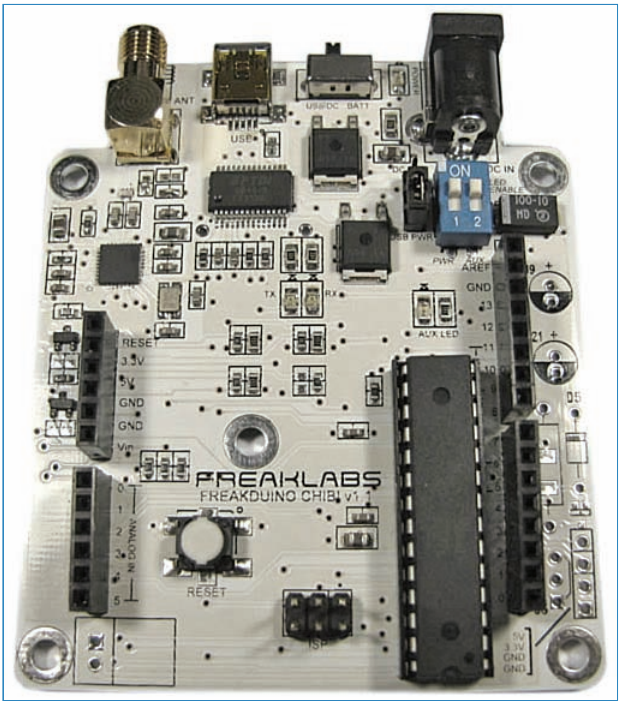


FIGURE 4.4 Freaklabs' Freakduino Chibi is essentially an Arduino with built-in wireless capability.

JeeLabs JeeNode

A similar concept to the Chibi, the JeeNode consists of an ATmega328p, which is the same microchip that serves as the mind of the Arduino, along with a built-in wireless module. JeeNodes are very small and have fewer capabilities than the Chibi, but have many fans due to the JeeNodes' small form factor and their ease of use. You can purchase them at http://jeelabs.com/products/jeenode.

TIP

Just Use Series 1

There is so much more to learn about radios, and you might already be overwhelmed! I suggest just limiting yourself to the XBee, non-Pro, Series 1. It's a wonderfully simple way to add wireless to your projects without spending too much money or frustrating yourself by taking on too complicated a radio before you need to.

Project: Wireless LED Activation

Oooh, wireless radios! Working with them sounds kind of intimidating. It's actually not, and I'll prove it. Let's create a simple network (see Figure 4.5) that lets two Arduinos communicate. In this mini-project, you'll create two identical assemblies, each consisting of an Arduino and XBee, along with a button and a LED. When you press the button on one assembly, the LED on the other one lights up, and vice versa! You can see how this project will give you a nice start toward building a wireless doorbell, which is the main project for this chapter.

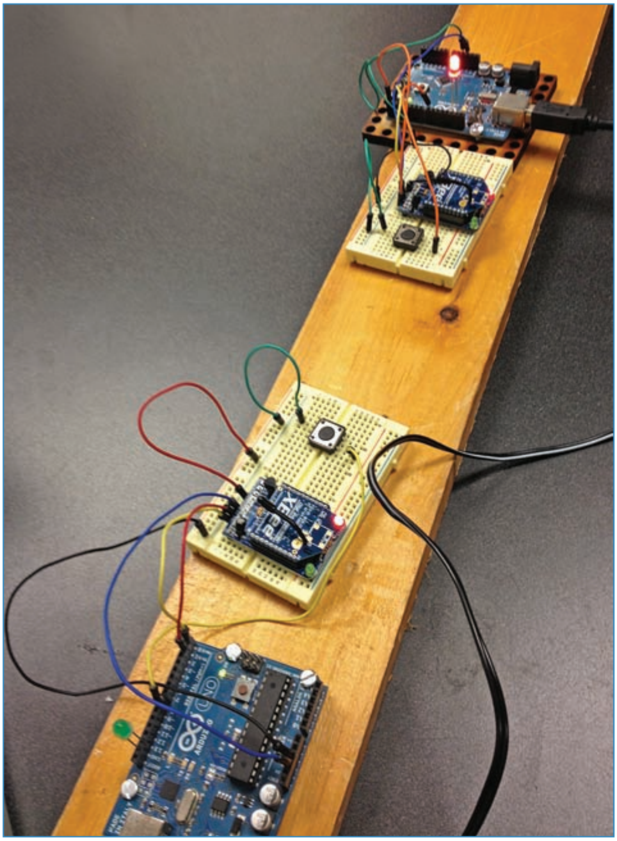


FIGURE 4.5 Control LEDs with XBee-equipped Arduinos.

PARTS LIST

You'll be making two assemblies, so you need two of everything!

- Arduinos (x2)
- XBees (x2)
- Breakout boards (x2)
- Pushbuttons (x2)
- Breadboards (x2)
- LEDs (x2)
- Jumpers

Follow these steps to assemble the XBee test platform:

1. **Solder the breakout boards**—Solder up your XBee breakout boards if you haven't already. Depending on your kit, this could mean simply soldering in some header pins. On other kits, however, you must solder in LEDs, capacitors, and so on.
2. **Connect the XBees to the breakout boards**—Attach the XBees to their respective breakout boards. This typically involves simply plugging in the XBees' pins to the appropriate holes in the breakout board. Just follow the directions that accompany your kit.
3. **Attach to breadboards**—Plug the breakout boards and XBees into the breadboards. You can see where to place it in Figure 4.6.
4. **Attach the pushbuttons, LEDs, and jumpers**—Attach these items as follows (also shown in Figure 4.6):
   A. GND on the XBee goes to GND on the breadboard. Connect the GND bus of the breadboard to the GND port of the Arduino.
   B. +5V on the XBee goes to 5V on the Arduino.
   C. TX on the XBee goes to RX on the Arduino.
   D. RX on the XBee goes to TX on the Arduino.
   E. Connect a button to pin 8 on the Arduino; the other end connects to the GND bus.

You should end up with two identical units, and if you upload the Arduino code to both of them, they should work identically. Even cooler, the way the networks are set up, you could actually create three or more of these assemblies and they'll all work the way you would expect. Press the button on one, and the LEDs on all the others will light up! It's not super practical, to be sure, but it shows how easily you can set up an XBee network.

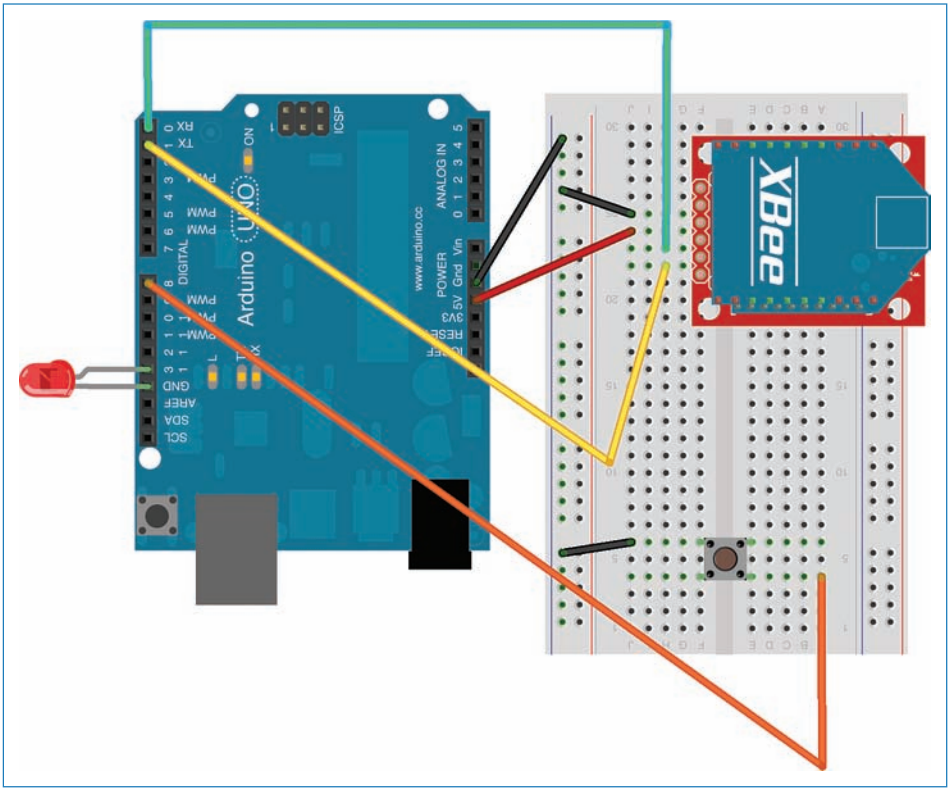


FIGURE 4.6 This diagram shows you how to create these XBee test modules.

Wireless LED Code

Upload the following code to both Arduinos. Remember, both modules are identical, down to the software. If you can't remember how to upload sketches to your Arduino, Chapter 5, "Programming Arduino," explains how.

NOTE

Code Available for Download

You don't have to enter all of this code by hand. Simply go to https://github.com/n1/Arduino-For-Beginners to download the free code.

```
#include <Wire.h>

const int buttonPin = 8;
const int ledPin = 13;
int buttonState = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
 pinMode(buttonPin, INPUT_PULLUP);
}

void process_incoming_command(char cmd)
{
  int speed = 0;
  switch (cmd)
  {
  case '1':
  case 1:
    digitalWrite(ledPin, LOW);
    break;
  case '0':
  case 0:
    digitalWrite(ledPin, HIGH);
    break;

  }
}


void loop() {
  if (Serial.available() >= 2)
  {
    char start = Serial.read();
    if (start != '*')
    {
      return;
    }

    char cmd = Serial.read();
    process_incoming_command(cmd);
  }

  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.write(‘*’);
    Serial.write(1);
  }
  else {
    Serial.write(‘*’);
    Serial.write(0);
  }
  delay(50); //limit how fast we update
}
```

Project: Bluetooth Doorbell

Now you can take what you learned about XBees and apply it to a slightly more robust project: a wireless doorbell. Figure 4.7 shows the doorbell button, and Figure 4.8 shows the buzzer unit that is tucked away on a shelf inside.

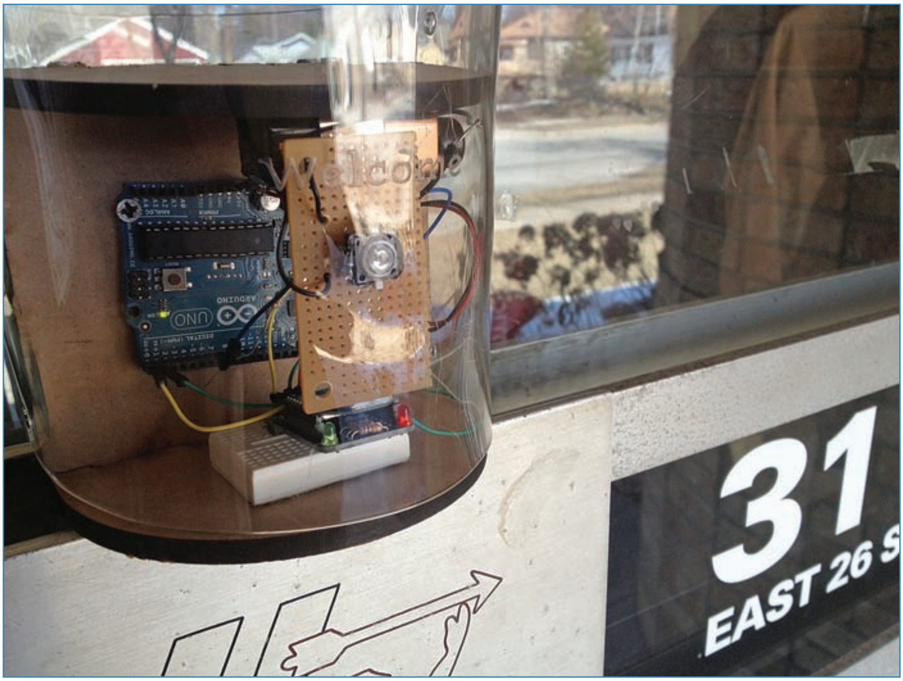


FIGURE 4.7 The doorbell awaits visitors!

FIGURE 4.8 The buzzer unit sits discreetly on a shelf.

Sure, you might say, they make these already! You can buy a wireless doorbell in any hardware store. However, this one you make yourself! Even better, as you get more confident with Arduino, you can modify it to make it uniquely yours. For instance, what if your Arduino triggers a music player instead of a buzzer to let you know that someone has pressed the button?

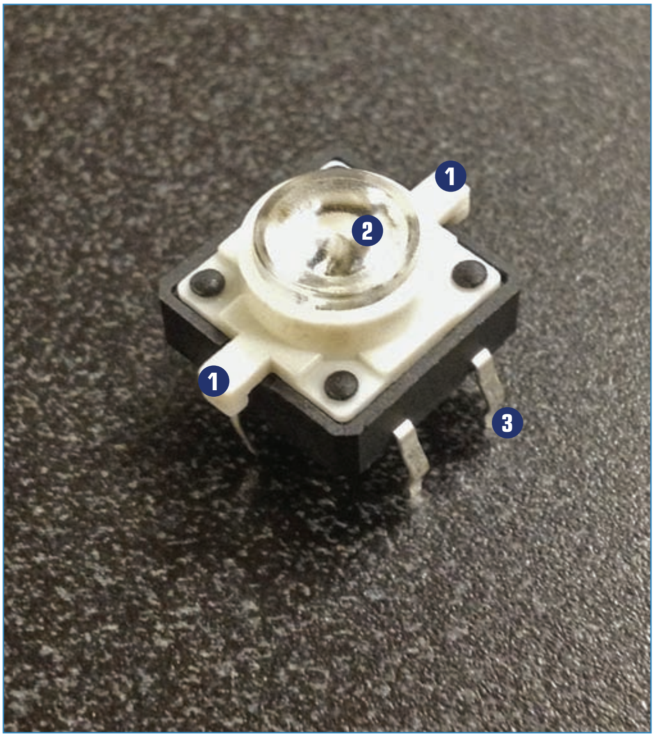
PARTS LIST

Just as in the mini-project earlier in the chapter, you'll be using two Arduinos, linked together. However, in this project, one Arduino waits for a button press, while the other one sets off a buzzer when it detects that the button has been pressed.

- 2 Arduino Unos
- 2 XBee wireless modules (Adafruit P/N 128)
- 2 Adafruit XBee breakout boards (Adafruit P/N 126)
- 2 mini breadboards (these are really small breadboards the sign of a postage stamp, Adafruit P/N 65)
- Button (SparkFun P/N COM-10443)
- A 330-ohm resistor
- Buzzer (Jameco P/N 1956776)
- Jumpers
- 9v battery clip (Jameco P/N 105794)
- 9v connector with barrel plug (Adafruit P/N 80)
- 1/4-inch MDF for enclosure backing and sides
- 5mm acrylic for enclosure front
- 1-inch #4-40 bolts
- Hot glue gun

The Button

The button you use in the button unit, shown in Figure 4.9, is kind of intriguing because it has six connectors: two sets of positive and negative terminals that close when the button is pressed—so you could have two circuits, both of which trip when the button is activated. The last two leads—the white lugs in the photo—are for powering the LED. Be sure to attach a resistor on the power lead so you don't fry your LED inadvertently. I use a 330-ohm resistor in this project.



1 LED Terminals

2 LED Button

3 Switch Connectors (Two Located on Other Side, Too)

FIGURE 4.9 The button you use in the project has six connectors.

Instructions for Wiring Up the Doorbell

The project consists of two Arduinos equipped with XBee modules and breakout boards. One Arduino has a button, and the other has a buzzer to sound out to let you know someone is at your door. Let's get started!

Button Unit

Let's begin with the button unit (see Figure 4.10), which consists of the following components:

A	9V battery
B	XBee module
C	Mini breadboard
D	Arduino Uno
E	Button
F	Perfboard

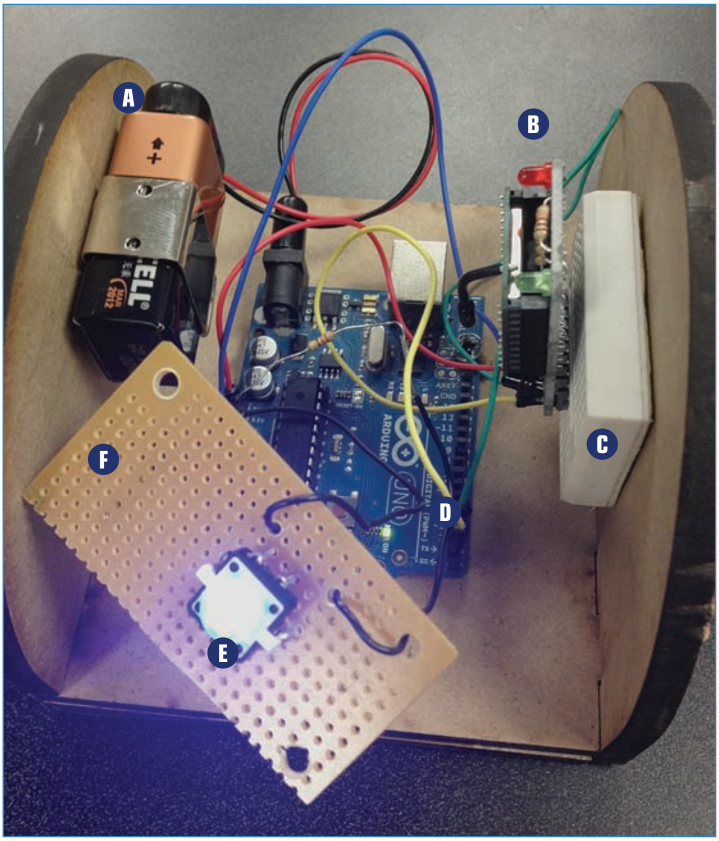


FIGURE 4.10 The button unit before the acrylic is added.

Now, assemble these parts together as shown in Figure 4.11, and you can follow along with these steps:

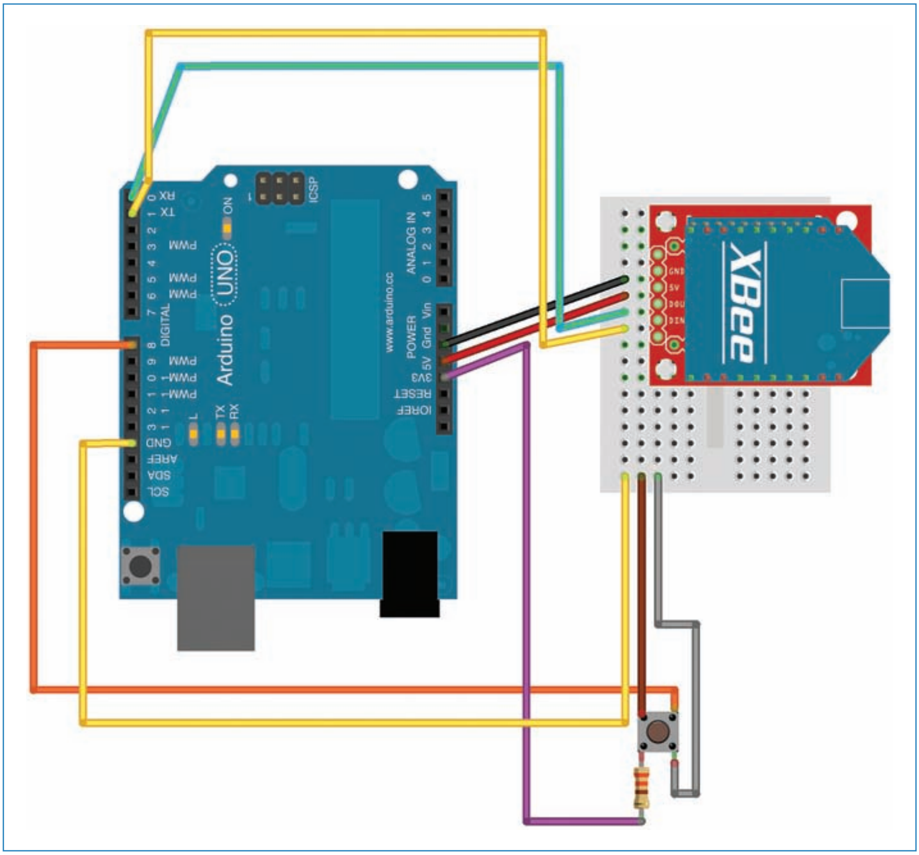


FIGURE 4.11 The button unit consists primarily of a button, an Arduino, and the wireless module.

1. Plug in the XBee and its breakout board to a mini breadboard.
2. Plug the XBee's 5V to the 5V on the Arduino, its TX into RX, its RX into TX, and its GND pin to any free GND on the Arduino.
3. Connect one of the button's leads to pin 8 and the other to GND. (I use the breadboard to accommodate the GND leads coming from the button.)
4. Solder a 330-ohm resistor and a jumper to the button's LED's power terminal, and connect the other end to the 3V3 port of the Arduino. The other terminal of the LED goes to GND.

Buzzer Unit

Next, connect the components that make up the buzzer unit, seen in Figure 4.12. These consist of the following:

- A Arduino Uno
- B Mini breadboard
- C XBee wireless module
- D Buzzer

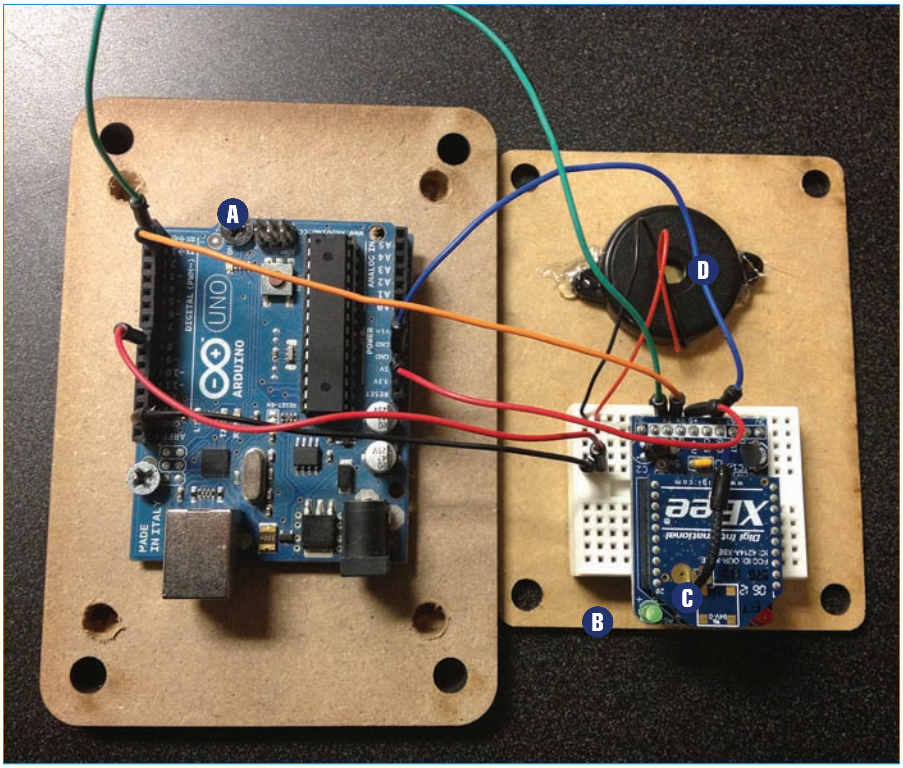


FIGURE 4.12 The buzzer unit waiting to be closed up. The outer holes are for wall mounting.

Next, use Figure 4.13 as a guide for connecting the various parts:

1. Plug in the XBee and its breakout board into a mini breadboard.

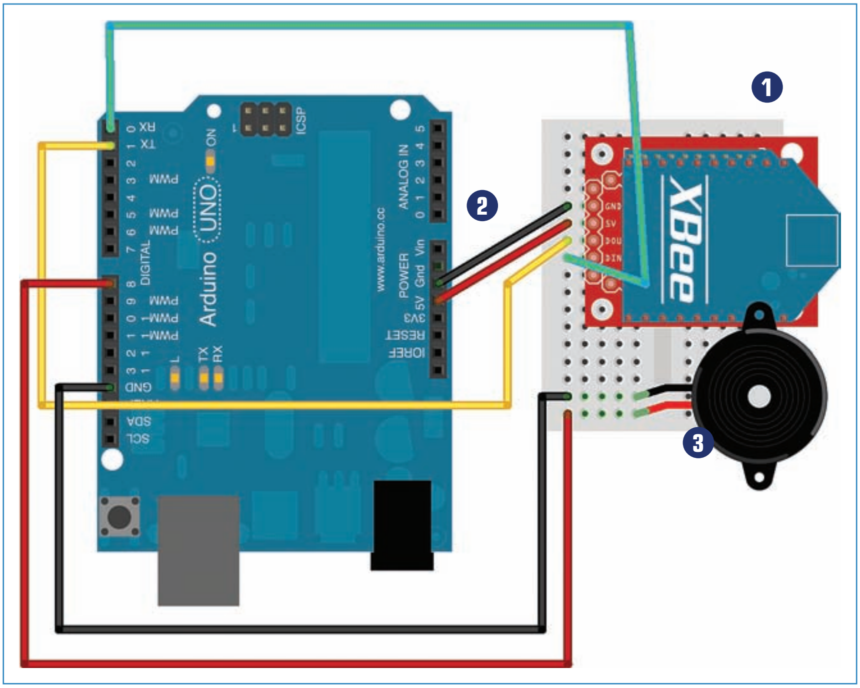


FIGURE 4.13 The buzzer unit consists of an Arduino, XBee, and buzzer.

2. Plug in the XBee's 5V to the 5V on the Arduino, its TX into RX, its RX into TX, and its GND pin to any free GND on the Arduino.
3. Connect the buzzer's leads to the breadboard as well, as shown in Figure 4.13. You can connect them directly to the Arduino if you want—if you go this route, connect the red wire to pin 8 and the black wire to any free GND.
4. To power the buzzer unit, use an Arduino-compatible wall wart or a 9V battery pack.

Building the Doorbell Enclosures

You next need to build the two enclosures for this project. The outside enclosure (see Figure 4.14) is designed to resist the elements—I hesitate to call it "weatherproof"—whereas the inside enclosure is designed to look good.

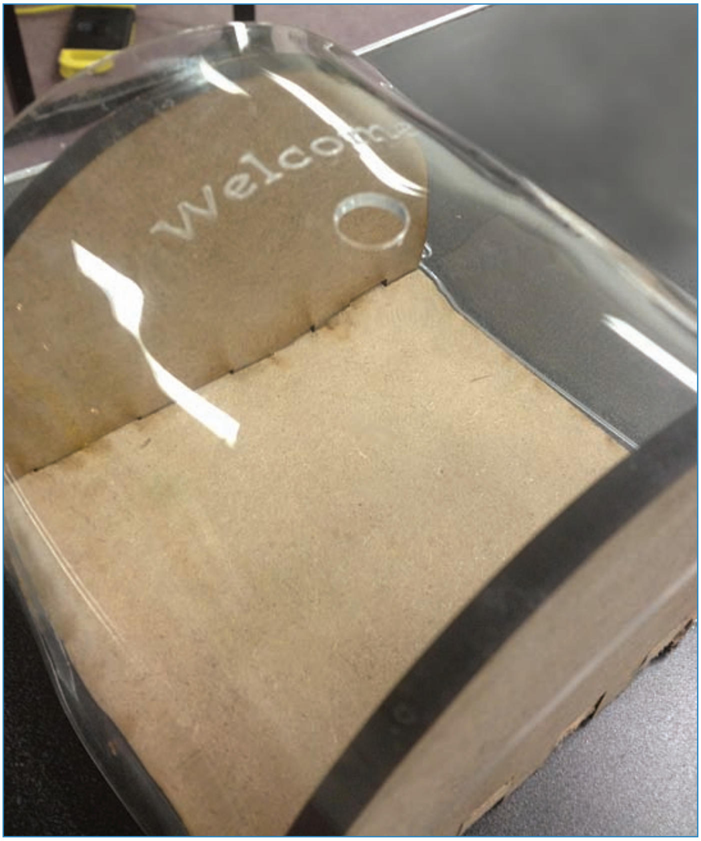


FIGURE 4.14 The outside enclosure is made out of bent acrylic on a wooden back.

Button Unit Enclosure

The button unit is the module that is on the outside of the door—press the button to make the buzzer buzz! To make an enclosure, all you need is a box with a hole for the button, but I'll show you how you can make one of your own. The one I made consists of a sheet of acrylic that I bent by heating it up, and then laying the flexible acrylic over a metal pipe to form a half-circle. I added the acrylic to a wooden back (refer to Figure 4.11) to finish the enclosure. Here are the steps:

1. Laser-cut the top, bottom, and back out of quarter-inch medium-density fiberboard (MDF). If you don't have access to a laser cutter, you can create a box out of pieces of wood, repurpose another container as an enclosure, or buy a commercial project enclosure.
2. Laser-cut the front from 5mm acrylic. (If you want the design files I used to output the wooden backing as well as the acrylic front, you can find them at https://github.com/n1/Arduino-For-Beginners.)

3. Glue the top and bottom wood pieces to the back wood piece. You might want to paint the wood!
4. Attach the completed electronics as shown earlier in Figures 4.10 and 4.12. Use the #4-40 bolts for the Arduinos and hot glue for the buzzer, battery pack, and mini breadboards.

   If you aren't using a laser cutter, you'll need to drill mounting holes in the acrylic. You might want to mock it up using a sheet of paper first.
5. Bend the acrylic front plate as described in the next section, "Bending Acrylic."
6. Attach the acrylic plate to the front so that the button can be pressed through the hole in the plastic.
7. Install the unit outside your door of choice, and eagerly await your first visitor!

Bending Acrylic

For the outside button unit enclosure, you heat-bend acrylic (see Figure 4.15) to form a casing. This task is easy to learn because you don't really need anything unusual or uncommon.

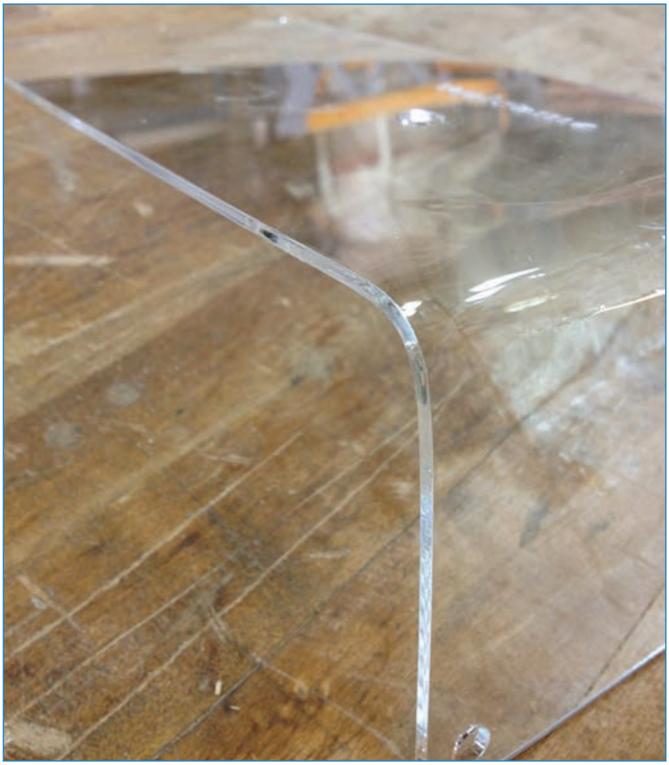


FIGURE 4.15 Bending acrylic is easy and gives a nice effect!

Acrylic (also known as Plexiglas) is also easy to heat and re-form. After it gets to the right temperature—not too hot or cool—the acrylic starts to bow and flex. When it gets a little hotter, it softens. That's when you bend it how you want it, and let it cool into an awesome new shape!

You need three things to get started:

- **The acrylic to be bent**—I suggest 1/8 inch, though you might have luck with the thicker stuff.
- **A form**—This is the surface over which the hot acrylic will cool and harden. You want this close to the actual curve you want the plastic to hold. The easiest form of all is the edge of a table. I used a rounded form—a pipe—to form the acrylic face seen in Figure 4.7. If you go this route, you'll need to find a form that matches the curve of the shape you're looking for.
- **A source of heat**—Heat guns (see Figure 4.16) and propane torches are common tools, though you can purchase commercial acrylic-heating strips (TAP Plastics has one for $80, P/N 169). Finally, you could heat up the plastic in an oven. This last technique is not for the faint of heart and you should definitely monitor the plastic closely so it doesn't bubble or scorch.

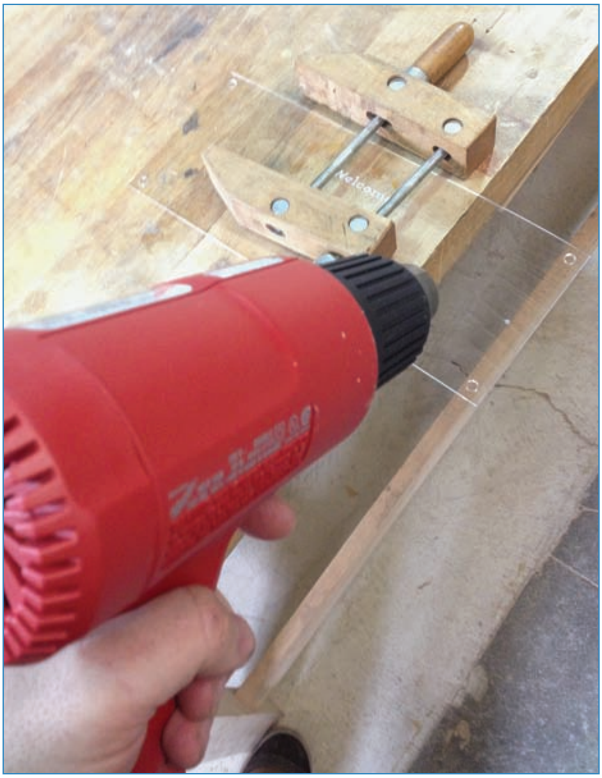


FIGURE 4.16 Using a heat gun to soften acrylic.

Although you could conceivably use any heat-resistant surface to form your acrylic—or even build your own out of pieces of wood—in some respects, using the edge of the table is an easy choice because it bends the plastic perfectly, using gravity and the table's surface to make a fairly perfect 90-degree bend. To bend plastic using the "edge of the table" technique, follow these steps:

1. As shown in Figure 4.17, position the acrylic so the edge of the table is right where you want the plastic to bend. You'll definitely want to weigh it down so it doesn't move.

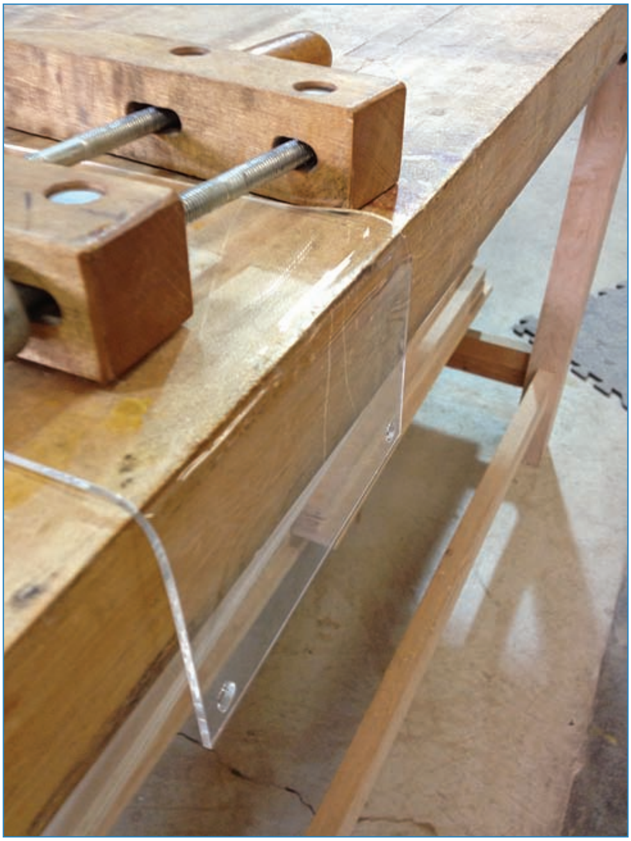


FIGURE 4.17 As the acrylic heats up, it starts to bend.

   When it gets hot enough, gravity starts pulling the soft acrylic down, as shown in Figure 4.17.

2. Position the acrylic how you want it to look—and work quickly because after it cools, it becomes just as brittle as it was before. Don't try to re-bend it without applying more heat!

Buzzer Unit Enclosure

The buzzer unit doesn't use plastic, because who wants plastic in their home? Instead, you can use a simple arrangement of wooden panels separated by bolts. I laser-cut two pieces of wood, one bigger than the other. (I ended up hand-drilling four additional holes, as shown in Figure 4.18, after changing my mind on how to proceed.)

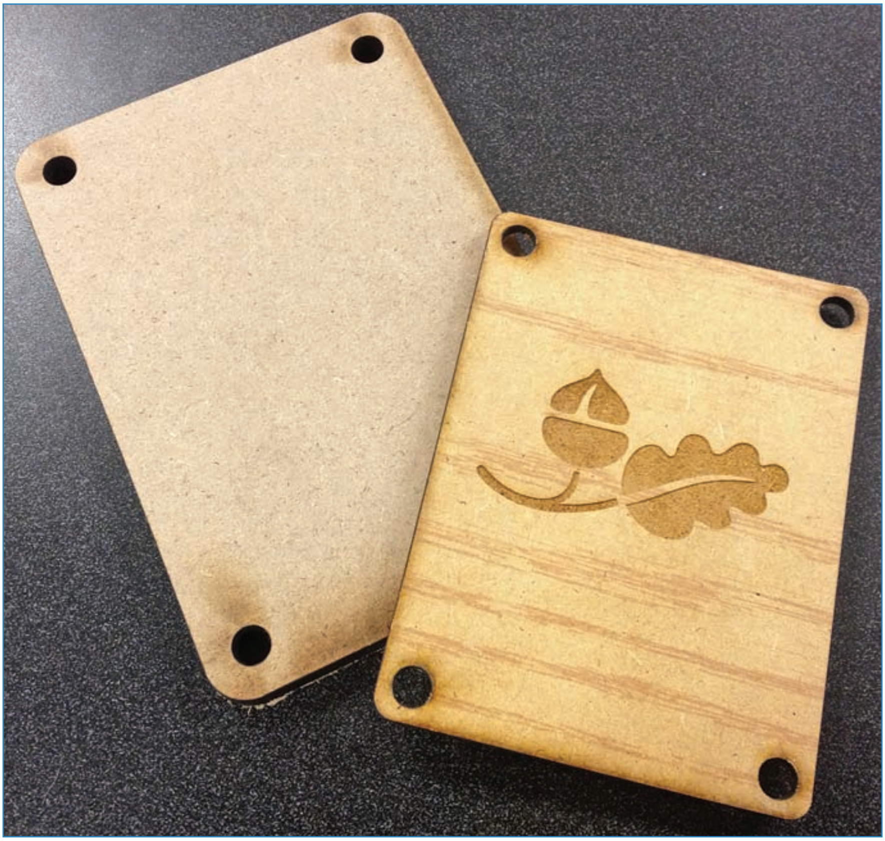


FIGURE 4.18 I used laser-cut wood for the buzzer unit's enclosure.

To connect the two pieces I used brass bolts, #10-24 and 2.5" long, with brass washers and nuts. This enclosure is considerably easier to do than the other enclosure and it looks great!

Wireless Doorbell Code

Upload the following code to your Arduinos. If you're having difficulty figuring out how to upload your sketches, see Chapter 5 to learn how. As before, you can download the code from https://github.com/n1/Arduino-For-Beginners.

Button Unit Code

The Button Unit sketch consists of a loop that waits for the button to be pressed, then transmits a wireless alert.

NOTE

Code Available for Download

You don’t have to enter all of this code by hand. Simply go to https://github.com/n1/Arduino-For-Beginners to download the free code.

```
#include <Wire.h>

const int buttonPin = 8;
int buttonState = 0;

void setup()
{
  Serial.begin(9600);
 pinMode(buttonPin, INPUT_PULLUP);
}

void loop() {
  if (Serial.available() >= 2)
  {
    char start = Serial.read();
    if (start != ‘*’)
    {
      return;
    }

    char cmd = Serial.read();
  }

  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    Serial.write(‘*’);
    Serial.write(1);
  }
  else {
    Serial.write('*');
    Serial.write(0);
  }
  delay(50); //limit how fast we update
}
```

Buzzer Unit Code

The Buzzer Unit code is similarly plain. The loop monitors serial traffic, then sounds the buzzer when it detects the command from the Button Unit.

NOTE

Code Available for Download

You don't have to enter all of this code by hand. Simply go to https://github.com/n1/Arduino-For-Beginners to download the free code.

```
#include <Wire.h>

const int buzzerPin = 13;

void setup()
{
  Serial.begin(9600);
  pinMode(buzzerPin, OUTPUT);
}

void process_incoming_command(char cmd)
{
  int speed = 0;
  switch (cmd)
  {
case 1:
    digitalWrite(buzzerPin, LOW);
    break;
case 0:
    digitalWrite(buzzerPin, HIGH);
    break;

  }
}
```

```
void loop() {
  if (Serial.available() >= 2)
  {
    char start = Serial.read();
    if (start != '*')
    {
      return;
    }

    char cmd = Serial.read();
    process_incoming_command(cmd);
  }

  delay(50); //limit how fast we update
}
```

The Next Chapter

So far we've been talking the hardware angle, but now it's time to switch things up! You get to delve into Arduino code in Chapter 5 and learn a bunch of programming techniques as well as the specific formatting you'll need to successfully write your very own Arduino program.

This page intentionally left blank

Index