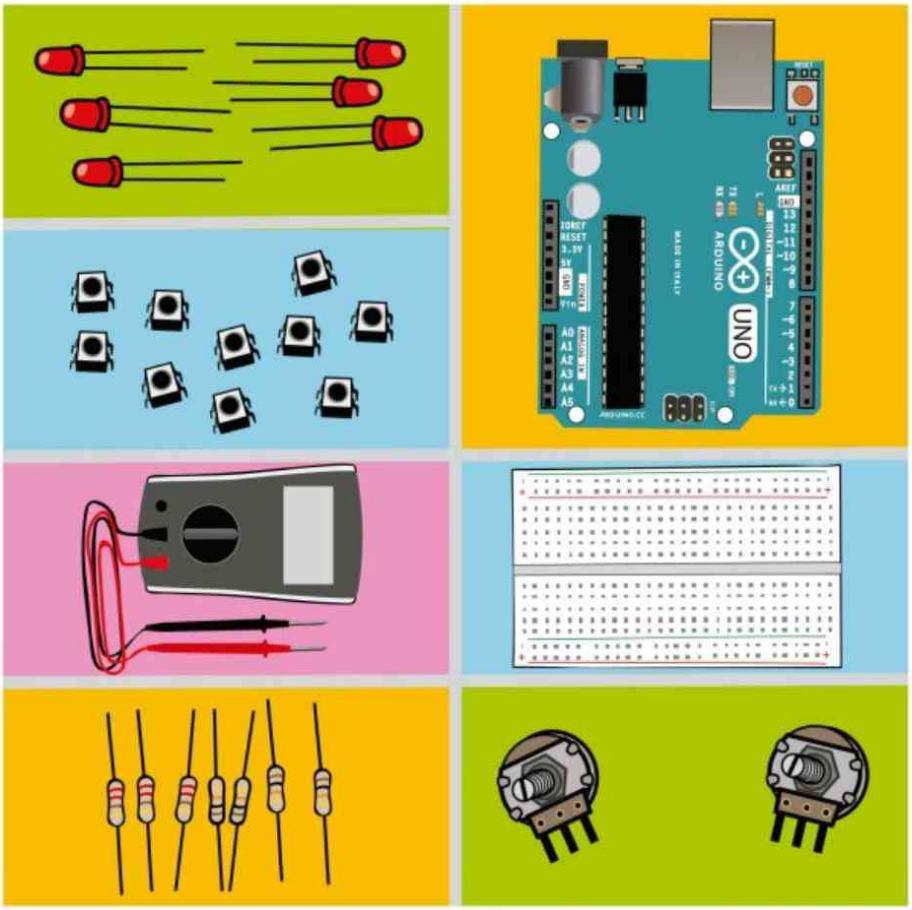
ARDUINO

APRENDE ELECTRÓNICA CON ARDUINO



UNA GUÍA ILUSTRADA PARA PRINCIPIANTES SOBRE LA INFORMÁTICA FÍSICA

JODY CULKIN Y ERIC HAGAN

Marcombo

APRENDE ELECTRÓNICA CON ARDUINO

UNA GUÍA ILUSTRADA PARA PRINCIPIANTES SOBRE LA informática FÍSICA

Jody Culkin Y Eric Hagan

Marcombo

MAKER MEDIA
SAN FRANCISCO, CA

Edición original publicada en inglés por Maker Media, Inc. con el título *Learn Electronics with Arduino*, ISBN 9781680453744, © Jody Culkin y Eric Hagan en 2017.
Título de la edición en español:
Aprende electrónica con Arduino
© 2019 para la edición en español MARCOMBO, S.A. www.marcombo.com
Traducción: Francisco Martínez Carreno
Make:, Maker Shed y Maker Faire son marcas registradas de Maker Media, Inc. El logo de The Maker Media es una marca registrada de Maker Media, Inc. An Illustrated Beginner's Guide to Physical Computing y la imagen comercial relacionada son marcas comerciales de Maker Media, Inc.
Muchas de las designaciones utilizadas por los fabricantes y vendedores para diferenciar sus productos son consideradas como marcas registradas. En los sitios en que las mismas aparecen en esta publicación, siempre que Maker Media estuviera consciente de un reclamo de marca registrada, dichas designaciones aparecen en mayúsculas cerradas o con mayúscula inicial.
A pesar que la editorial y el autor han agotado todo esfuerzo de buena fe para asegurar que la información e instrucciones contenidas en este trabajo sean veraces y precisas, la editorial y el autor declinan toda responsabilidad por los errores u omisiones, incluyendo, sin limitarse a, la responsabilidad por daños resultantes del uso de, o la confianza en este trabajo. El uso de la información e instrucciones aquí contenidas se da a su propio riesgo. Si el uso de muestras de códigos u otras tecnologías contenidas o descritas en este trabajo se encuentra sujeto a licencias de fuente abierta o al derecho de propiedad intelectual de terceros, es responsabilidad suya asegurarse que el uso que usted les dé cumpla con dichas licencias y / o derechos.
Maker Media cohesiona, inspira, informa y entretiene a una creciente comunidad de personas ingeniosas que emprenden proyectos sorprendentes en los

patios, sótanos y garajes de sus casas. Maker Media celebra tu derecho a modificar, piratear y duplicar cualquier tecnología a tu voluntad. La audiencia de Maker Media continúa siendo una cultura y comunidad en pleno crecimiento que cree en mejorarnos a nosotros mismos, al medio ambiente, al sistema educativo, al mundo entero. Esto es mucho más que una audiencia, es un movimiento mundial que Maker Media está liderando. Lo llamamos el Maker Movement.

Para saber más sobre Make: visítanos en makezine.com. Puedes tener más información sobre la empresa en los siguientes sitios web:

•Maker Media: makermedia.com

•Maker Faire: makerfaire.com

•Maker Shed: makershed.com

«Cualquier forma de reproducción, distribución, comunicación pública o transformación de esta obra solo puede ser realizada con la autorización de sus titulares, salvo excepción prevista por la ley. Diríjase a CEDRO (Centro Español de Derechos Reprográficos, www.cedro.org) si necesita fotocopiar o escanear algún fragmento de esta obra».

ISBN: 978-84-267-2659-9

D.L.: B-27530-2018

DEDICATORIA

Dedicado a todos los estudiantes del pasado, presente y futuro.
Su curiosidad los impulsa y nos inspira.

CONTENIDO

AGRADECIMIENTOS

Este libro no hubiera sido posible sin la ayuda de muchas personas, muchas más de las que podemos mencionar aquí. Nos gustaría agradecer a nuestra editora técnica, Anna Pinkas, por la incansable y minuciosa revisión del texto. La versión anterior de este libro también se vio favorecida por la edición técnica de Michael Colombo y Sharon Cichelli. Roger Stewart, el editor y redactor, del que hemos tenido todo su apoyo y ayuda durante el proceso de impresión del libro. El equipo de producción de Happenstance Type-O-Rama; ha sido un placer trabajar con él, especialmente con Liz Welch y Maureen Forys. Nos conocimos en el Programa de Telecomunicaciones Interactivas de la Universidad de Nueva York, y siempre estaremos agradecidos con Tom Igoe por sugerir que trabajáramos juntos en un proyecto. De hecho, nos gustaría expresar nuestro agradecimiento a todo el profesorado y el personal de ITP, especialmente a Dan O'Sullivan y Marianne Petit.

A Eric le gustaría agradecer a su esposa Marie su apoyo incondicional, sin el cual este libro no habría sido posible. También quiere dar las gracias a sus padres, David y Tracey, que siempre han mostrado mucha confianza en su trabajo.

A Jody le gustaría agradecer a su esposo Calvin Reid, que piensa que ella puede hacer cualquier cosa y ha hecho todo lo posible para que eso haya ocurrido. Y le gustaría reconocer la memoria de sus padres, Florence y Hosmer Culkin, que estarían sorprendidos y orgullosos de que haya sido coautora de un libro sobre tecnología.

SOBRE LOS AutORES

Jody Culkin es artista y profesora. Ha expuesto sus esculturas, fotografías y montajes en museos y galerías de todo el país y a nivel internacional. Ilustró How to Use a Breadboard, escrito por Sean Ragan, para Maker Media (2017). Su cómico ¡Arduino! ha sido traducido a 12 idiomas. Ha recibido subvenciones y premios de la National Science Foundation, el New York State Council on the Arts y de muchas otras organizaciones. Actualmente es profesora en el City University of New York's Borough of Manhattan Community College, en el Media Arts and Technology Department. Tiene una licenciatura de la Universidad de Harvard en estudios visuales y un MPS (Máster de Estudios Profesionales) del Programa de Telecomunicaciones Interactivas de la Universidad de Nueva York.

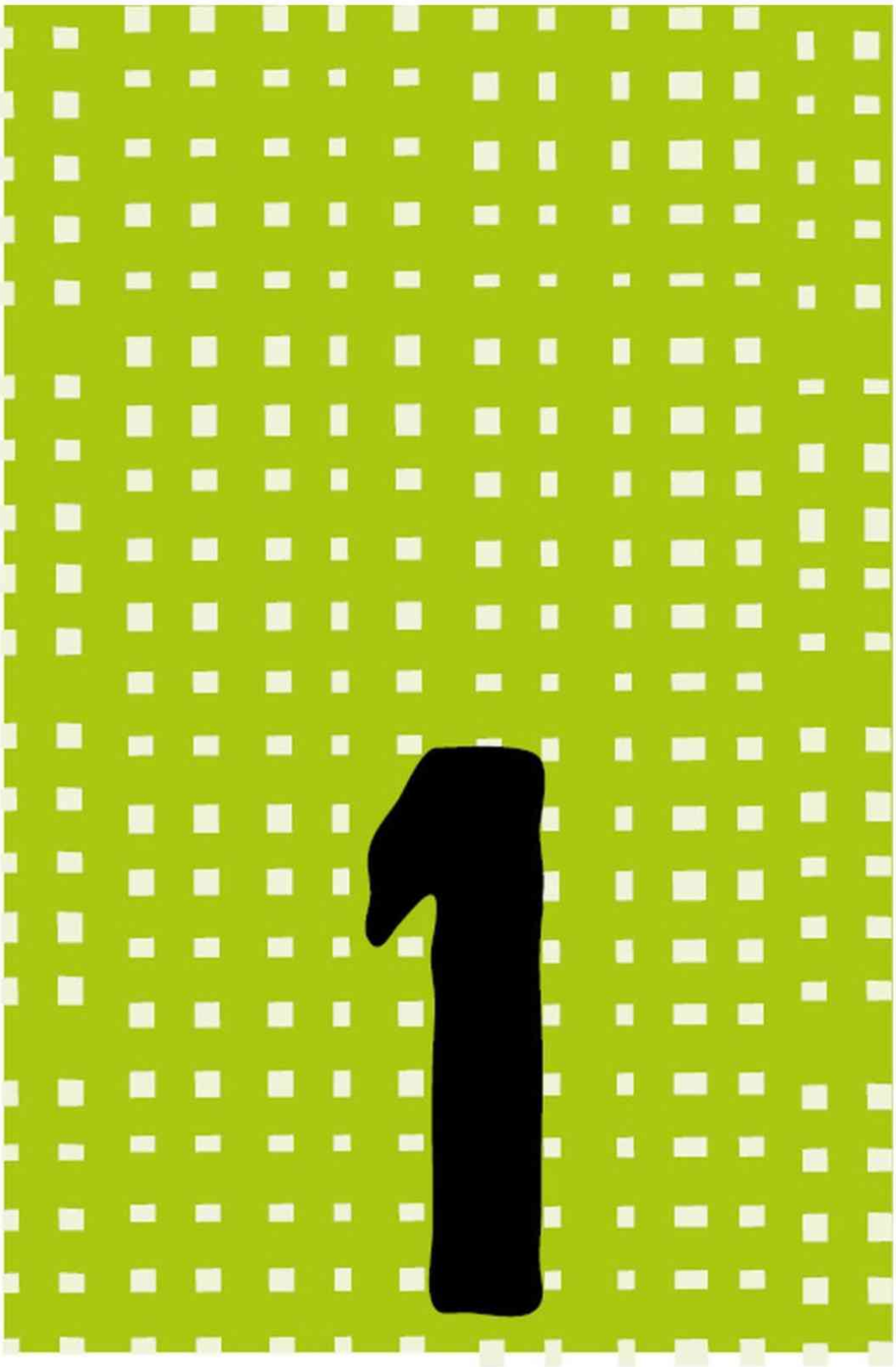
Eric Hagan es un artista interactivo y cinético y profesor residente en Astoria, Nueva York. Ha publicado artículos, incluyendo "Make: magazine y Popular Science". También ha trabajado en varios proyectos de montaje artístico en los alrededores de la ciudad de Nueva York, entre los que se pueden citar las ventanas navideñas anuales en la Quinta Avenida y *A Subtlety* de Kara Walker. Actualmente es profesor asistente en SUNY Old Westbury, en el Departamento de Artes Plásticas. Tiene la licenciatura en filosofía de la Universidad de Duke y un MPS del Programa de Telecomunicaciones Interactivas de la Universidad de Nueva York. A Eric le ilusiona presentar proyectos en la feria anual New York City World Maker Faire.

Prefacio

Concebimos este libro como una introducción a la electrónica y a la plataforma Arduino para principiantes. Lo hemos escrito e ilustrado asumiendo que el lector no tiene conocimientos previos de electrónica ni de programación. A medida que se avanza en su lectura, los conceptos de electrónica y programación se explican detalladamente a través del texto y de imágenes. Una vez que el lector haya completado el libro, podrá usarlo como consulta de electrónica básica de programación de Arduino.

Este libro debería ser el punto de partida para proyectos creativos. Al terminar de leer el libro y completar todos los ejercicios que contiene, los lectores deben estar preparados para comenzar a desarrollar sus propios proyectos. No hemos explorado todo lo que el equipo Arduino puede hacer, pero hemos orientado a los lectores a descubrirlo por sí mismos.

Muchos de los sketches de código usados en este libro están tomados de los ejemplos contenidos en el IDE de Arduino. Los otros sketches están disponibles en: github.com/arduinotogo/LEA.


1

Introducción a Arduino

Es posible que hayas podido ver el equipo Arduino en un distribuidor local, que hayas oído hablar de él a un amigo que lo ha comprado o que hayas visto un magnífico proyecto en Internet que ha despertado tu interés. ¿Qué es Arduino? Sencillamente se trata de un ordenador asequible, sencillo y diseñado a pequeña escala, que se centra en la interacción con el mundo exterior (figura 1.1).

La mayor parte de los ordenadores que conoces se controlan casi exclusivamente a través del teclado, el ratón, la pantalla o el panel táctil. Arduino te permite extraer información del mundo exterior con sensores que miden la temperatura, la intensidad de la luz, los niveles de sonido o incluso las vibraciones del suelo, y convierte estas medidas en movimiento, sonido, luz y muchas otras magnitudes físicas.

FIGURA 1.1 Logo de Arduino.

Fueron profesores los que originalmente desarrollaron Arduino para hacer posible que sus estudiantes de diseño, que no eran ingenieros, crearan objetos y entornos interactivos. Desde el lanzamiento en 2005 del Arduino original, se estima que se han vendido más de 1 millón de unidades. Diseñadores, educadores, ingenieros, aficionados y estudiantes han realizado todo tipo de proyectos capaces de percibir el entorno y responder a él a través

de Arduino.

Existen muchas versiones de Arduino, y cada una de ellas se ha diseñado para realizar una función específica. La figura 1.2 muestra algunas de las placas Arduino.

Hemos escrito este libro pensando en el espíritu del equipo Arduino. Suponemos que no tienes conocimientos de programación ni de electrónica. Aquí aprenderás todo lo que necesitas saber para empezar a trabajar con Arduino. Será de gran ayuda si se te dan bien el montaje y el entretenimiento, y eres una persona que actúa con determinación.

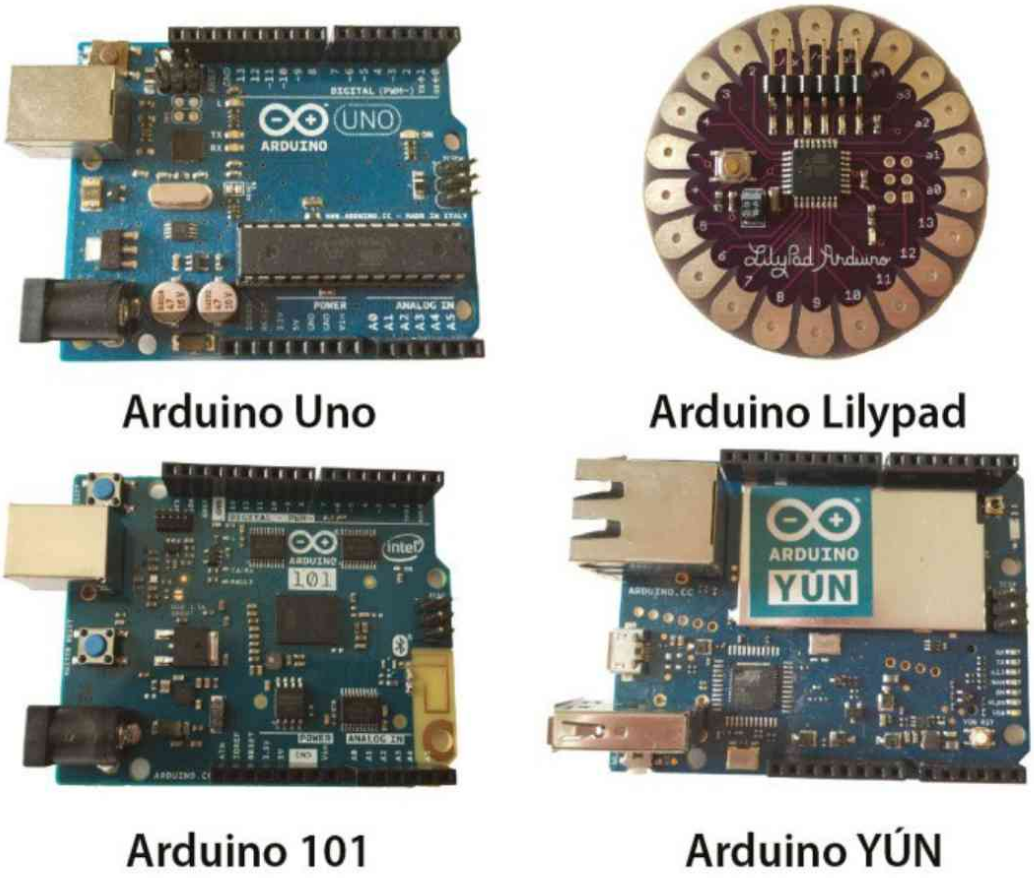


FIGURA 1.2 Existen muchas versiones de Arduino, y cada una se ha diseñado para realizar una función específica.

informática FÍSICA

El Arduino se utiliza para desarrollar proyectos de informática física. ¿Qué quiere decir eso? La informática física consiste en extraer información del mundo que nos rodea, se utilizan para ello las entradas procedentes de sensores e interruptores, y reaccionar a esa información con respuestas de algún tipo. Podría ser tan sencillo como encender un led cuando en una habitación deja de haber luz, o un sistema complejo de luz y sonido que actúe en función de la ubicación de una persona en una habitación. El Arduino puede actuar como el "cerebro" de este tipo de sistemas, gestionando la información entrante y proporcionando una respuesta.

Arduino forma parte del movimiento de hardware libre. Vamos a ver lo que eso significa.

¿QUÉ ES EL HARDWARE LIBRE?

Arduino se define en su sitio web como una plataforma para montar prototipos de electrónica de código abierto. En el movimiento de hardware libre, los tecnólogos comparten su hardware y software para fomentar el desarrollo de nuevos proyectos e ideas. Los diseños originales se comparten en un formato en el que se pueden hacer modificaciones, y para crear los diseños se utilizan, siempre que sea posible, materiales de fácil disponibilidad y herramientas de código abierto.

Al fomentar el intercambio de recursos, el movimiento de hardware libre facilita el desarrollo de nuevos productos y diseños. Los proyectos de código abierto ponen énfasis en la importancia de la documentación y del intercambio, lo que hace que la comunidad de usuarios constituya un importante recurso para los estudiantes.

MONTAJE DE PROTOTIPOS

Arduino es una plataforma para hacer prototipos. ¿Qué es hacer un prototipo? Consiste en la elaboración de un modelo de sistema.

Puede implicar la existencia de muchas fases, desde bocetos iniciales, pasando por planos detallados y una serie de mejoras, hasta la construcción de un modelo totalmente funcional que se pueda emular. O puede tratarse de un breve evento que se monta rápidamente para probar una idea.

¿QUÉ NECESITO Y DÓNDE LO PUEDO CONSEGUIR?

Existen varias versiones del Arduino; desde que apareció en el año 2005 se mantiene en constante evolución. Para el propósito de este libro, nos interesa trabajar con el Arduino Uno. Es posible que el Arduino que has comprado no tenga exactamente la apariencia del Arduino Uno que se muestra en la figura 1.3. Se debe a que hemos simplificado el dibujo para resaltar las secciones que nos interesan. Como el Arduino es de código abierto, también puedes comprar una placa que no proporcione directamente Arduino. Solo hay que saber que en este libro nos concentraremos en el Arduino Uno y las placas que sean compatibles con él.

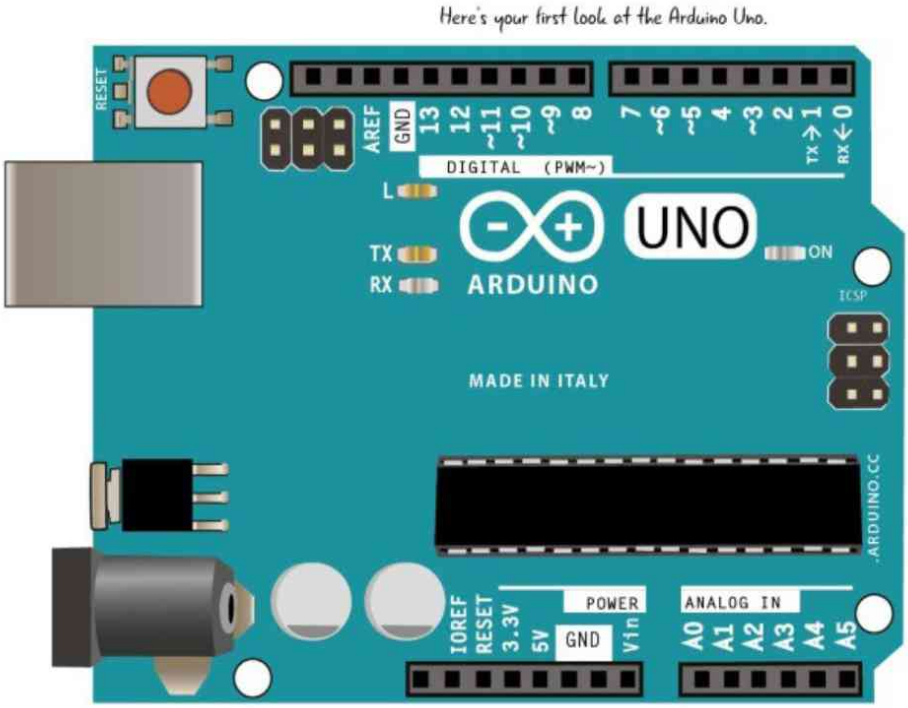


FIGURA 1.3 El Arduino Uno.

PIEZAS Y HERRAMIENTAS

Para poder realizar proyectos con el Arduino vas a necesitar algunas piezas electrónicas adicionales y también herramientas. A continuación se facilita una lista de las piezas que necesitas comprar para poder realizar los proyectos que aparecen en este libro. Te proporcionaremos más detalles sobre las piezas y su función a medida que vayamos desarrollando cada proyecto.

LISTA DE componentes

- ▨Placa de pruebas
- ▨Cable USB tipo A-B
- ▨Batería de 9 voltios
- ▨Fuente de alimentación de 9–12 voltios
- ▨Tapa o soporte de la batería de 9 voltios
- ▨Surtido de ledes, de distintos colores
- ▨Surtido de resistencias
- ▨Potenciómetro de 10 K
- ▨3 interruptores/botones momentáneos
- ▨Fotorresistencia
- ▨Altavoz de 8 ohmios
- ▨2 servomotores
- ▨Cables de conexión

Las siguientes ilustraciones, desde la 1.4 hasta la 1.16, muestran cómo son las piezas, acompañadas por una breve descripción. A las piezas electrónicas se las denomina a menudo componentes, ya que son los componentes del circuito electrónico. Aprenderás más sobre circuitos en el Capítulo 3, "Primer contacto con el circuito".

La placa de pruebas, que aparece en la figura 1.4, se utiliza para montar y comprobar circuitos de forma rápida. El cable USB tipo A-B que aparece en la figura 1.5, conecta el Arduino a un ordenador de manera que puedas

programarlo. También suministra energía. La batería de 9 voltios que se muestra en la figura 1.6 suministra energía cuando el Arduino no está conectado al ordenador.

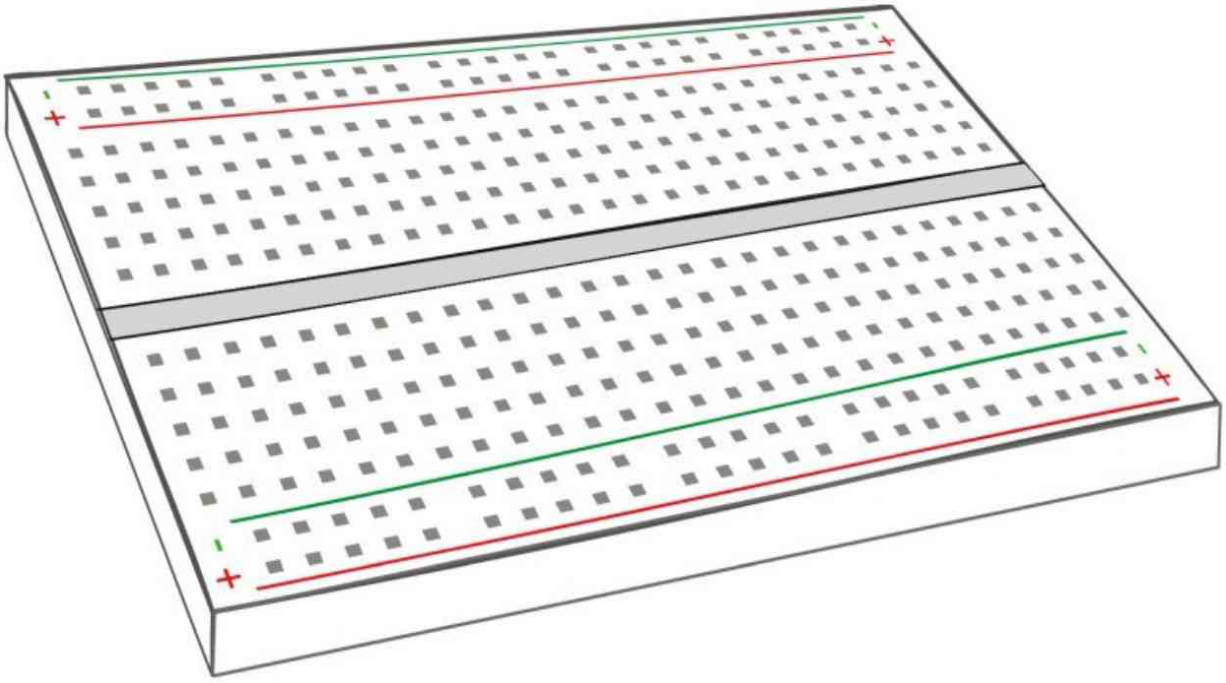


FIGURA 1.4 Placa de pruebas.

Figura 1.5:**Cable USB tipo A-B.**

Figura 1.6:**Batería de 9 voltios.**

Figura 1.7:**Tapa de la batería.**

La tapa de la batería de la figura 1.7 se usará para conectar la batería a la placa de pruebas. El adaptador de corriente que aparece en la figura 1.8 puede alimentar al Arduino cuando este no está conectado al ordenador. Los diodos emisores de luz (ledes), mostrados en la figura 1.9 emiten luz cuando se les aplica un voltaje.

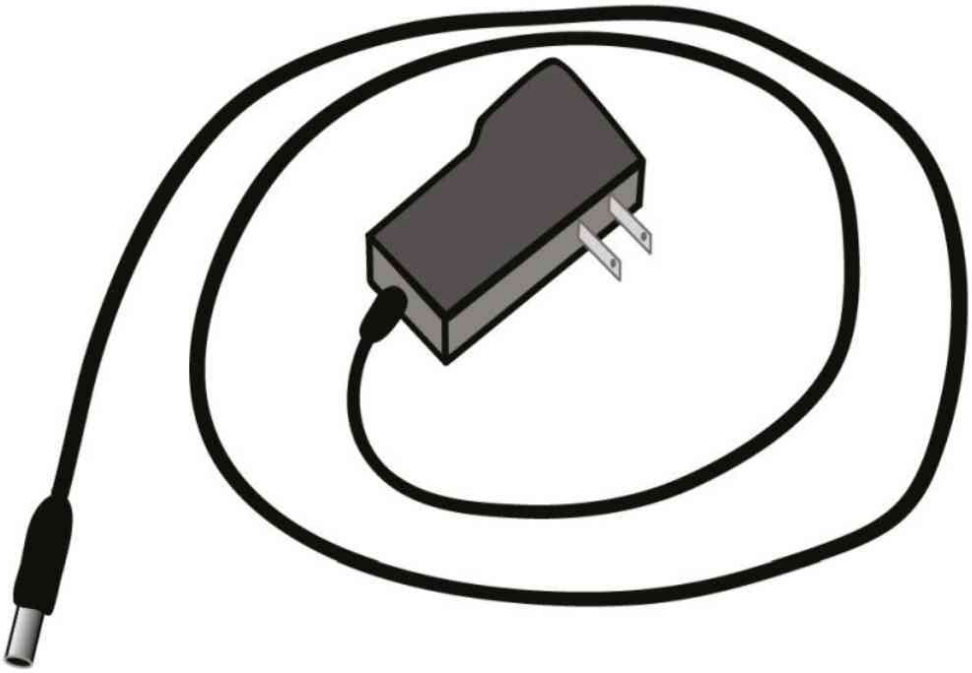


FIGURA 1.8 Adaptador de corriente.

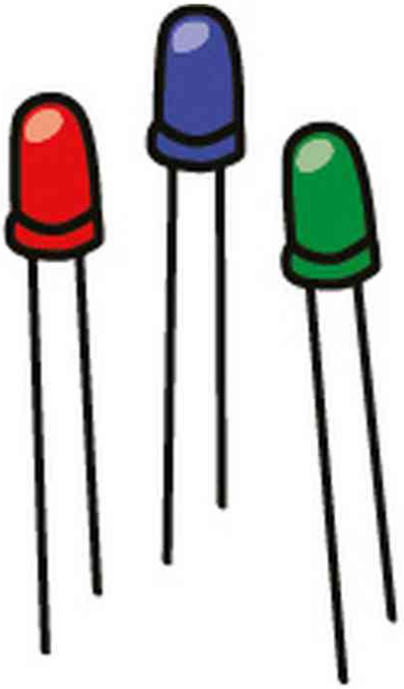


Figura 1.9:**Ledes.**

Las resistencias, como se puede ver en la figura 1.10, limitan el flujo de corriente en un circuito. Usaremos un pulsador momentáneo, mostrado en la figura 1.11, para establecer o interrumpir una conexión en el circuito. La

figura 1.12 muestra un potenciómetro, es decir, una resistencia variable.

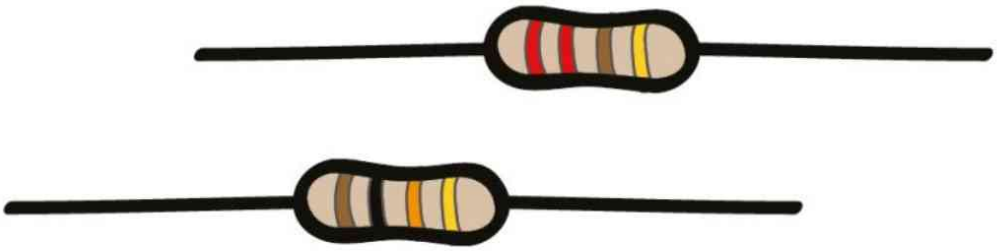


FIGURA 1.10 Resistencias.

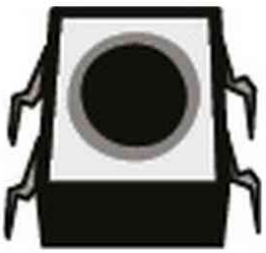


Figura 1.11:
Pulsador momentáneo.

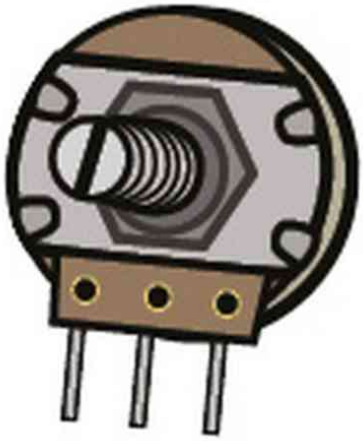


Figura 1.12: **Potenciómetro.**

La fotorresistencia mostrada en la figura 1.13 cambia su resistencia cuando se expone a diferentes niveles de luz. La figura 1.14 muestra un altavoz de 8 ohmios, que reproducirá las señales de audio. El servomotor es un motor

para aficionados que se controla fácilmente, como se puede ver en la figura 1.15. Los cables de conexión, mostrados en la figura 1.16, se utilizan para conectar componentes a la placa de pruebas. Los puedes comprar o los puedes hacer tú mismo con pelacables.

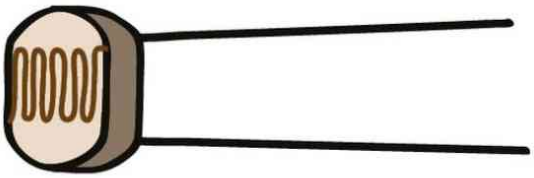


FIGURA 1.13 Fotorresistencia.

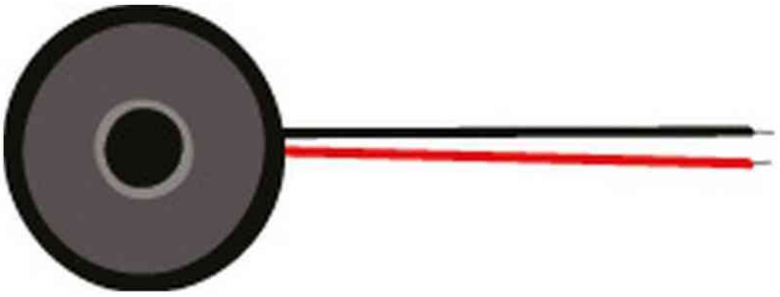


Figura 1.14:**Altavoz de 8 ohmios.**

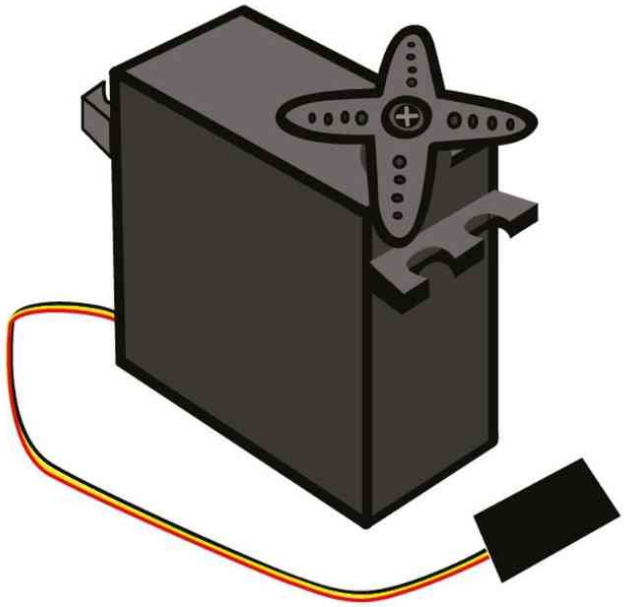


FIGURA 1.15 Servomotor.

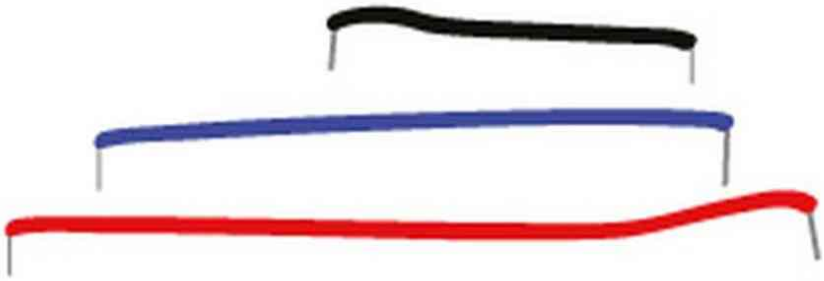


Figura 1.16:**Cables de conexión.**

Una nota sobre los ledes

Los ledes se presentan en una gran variedad de colores, estilos y tamaños. Usaremos ledes en muchos de los proyectos de este libro porque ayudan a exponer de forma visual un buen número de conceptos básicos sobre Arduino y de electrónica.

Es importante recordar que los ledes tienen una polaridad, o dirección en la cual se deben colocar para poder funcionar. Si colocamos los ledes en sentido contrario, no se encenderán. ¿Cómo sabemos la polaridad de un led?

Los ledes tienen dos patas, o cables, con diferentes longitudes, como se

puede ver en la figura 1.17. Al cable más largo se lo conoce como ánodo, y es el terminal que conectaremos a la alimentación. A la pata más corta se la llama cátodo, y se conectará al polo opuesto al de la alimentación. Te mostraremos cómo colocar los cables en un circuito cuando empecemos a montarlo, y siempre te recordaremos la polaridad en circuitos que montaremos después.

> **Note** Si se conecta el led al revés, no lucirá, pero tampoco se producirán daños en el circuito.

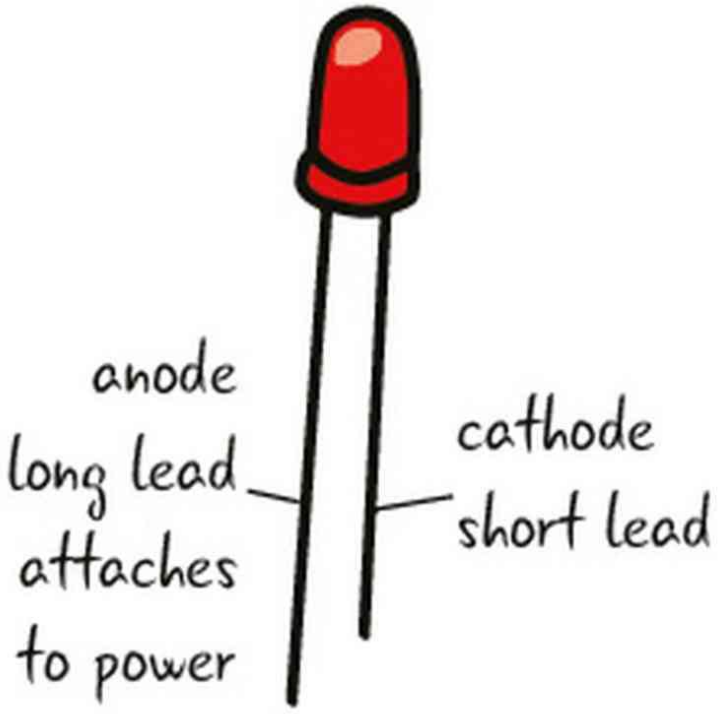


Figura 1.17:**Ánodo (terminal positivo) y cátodo (terminal negativo) de un led.**

¿Qué sucede si tienes un led usado que tiene cortados los cables? En muchos ledes, si observas la cápsula de plástico, un lado del borde de la parte inferior es plano. El cable conectado a ese lado es el cátodo, o lado negativo.

Ahora echemos un vistazo a algunas de las herramientas que necesitarás

para realizar estos proyectos.

Herramientas

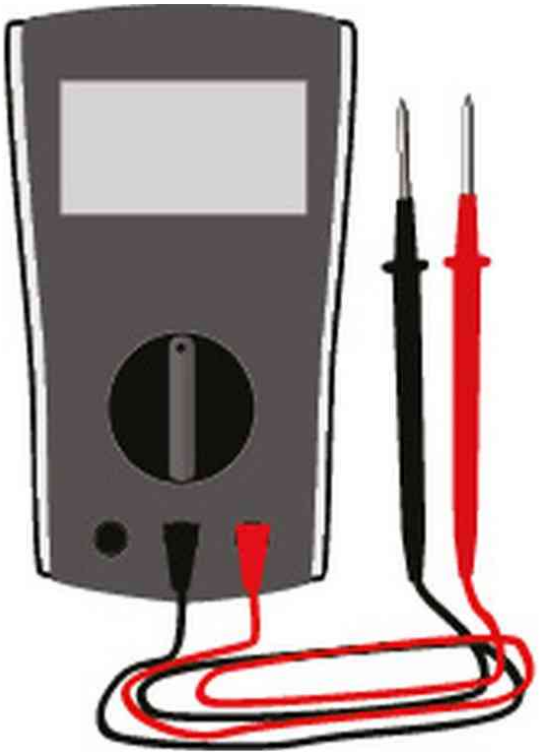


Figura 1.18:**Multímetro.**

El multímetro te dirá todo lo que necesitas saber sobre las propiedades de un circuito eléctrico, propiedades que no necesariamente están a la vista. Te enseñaremos a utilizarlo a partir del capítulo 2. El multímetro representado en la figura 1.18 está disponible en SparkFun (número de parte TOL-12966), pero puedes buscar otro que te guste. Cuando elijas un multímetro, comprueba que es digital, que sus cables son extraíbles y que tiene fusible.

Los alicates de punta de aguja, como se muestra en la figura 1.19, son útiles para extraer los componentes de la placa de pruebas y realizar cambios en un circuito. También son útiles para extraer pequeños componentes.

Los pelacables, en la figura 1.20, se utilizan para extraer el revestimiento aislante de plástico que recubre los cables, que pueden tener distintos

espesores. Te facilitarán mucho la vida cuando utilices rollos de cable, ya que podrás cortar y usar longitudes de cable a medida.

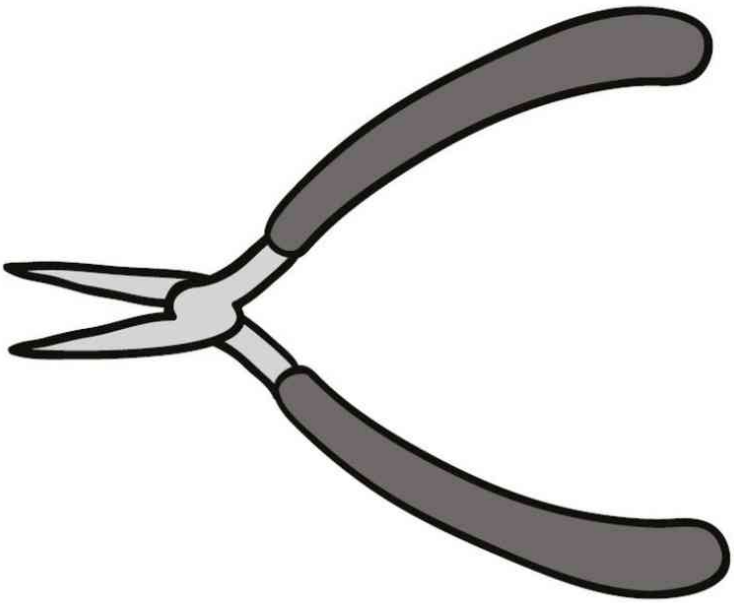


FIGURA 1.19 Alicates de punta de aguja.

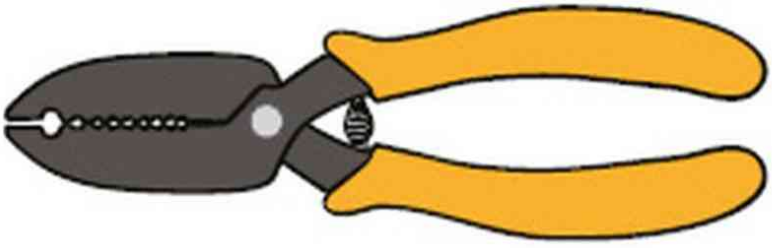


Figura 1.20:**Pelacables.**

Tip Aunque puedes comprar cables de puente precortados, recuerda que puedes hacerlos tú mismo usando los peladores de cable para quitar el recubrimiento plástico en los extremos de un tramo de cable. El cable de conexión calibre 22 funciona bien en las placas de pruebas.

Unas palabras sobre las herramientas: El soldador

Puede que estés familiarizado con el uso del soldador y sepas cómo usarlo en electrónica para conectar componentes. En este libro hemos optado por utilizar una placa de pruebas para hacer las conexiones de todos los circuitos que vamos a tratar. Esto significa que no tienes por qué comprar un soldador o aprender a usar uno para completar los proyectos que se montan en este libro.

¿Preguntas?

P: **¿Qué hace un soldador?**

R: **Un soldador se utiliza para fundir un material conductor ("soldadura") para combinar componentes eléctrónicos de forma permanente. Este proceso se denomina soldadura.**

P: **¿Por qué en este libro no se enseña a soldar?**

R: **Tener habilidad con la soldadura es estupendo y te ayudará a alcanzar el siguiente nivel en electrónica, pero en este libro nos preocupamos fundamentalmente de lo básico. Se pueden montar circuitos que funcionan perfectamente sin tener que acudir a la soldadura.**

P: **La lista de componentes parece tener muchas piezas. Las fotos se ven bien, pero ¿realmente necesito comprar todas las piezas de esa lista?**

R: **¡Verás muchas más fotos! Para responder a tu pregunta: emplearás todas las piezas cuando desarrolles los proyectos del libro. Estas piezas las puedes también reutilizar en tus propios proyectos. Explicaremos qué**

hacen estas piezas a medida que las usemos.

P: Mi amigo/hermano/padre/profesor/perro me dio un modelo más nuevo/antiguo de Arduino. ¿Tengo que usar el Arduino Uno para los proyectos de este libro?

R: Buena pregunta. Los proyectos del libro pueden funcionar con tu Arduino particular, pero tanto la programación como las posibilidades de Arduino han cambiado con el tiempo y difieren según la versión. Todos los ejemplos de este libro se han probado usando el Arduino Uno y la última versión del software Arduino.

P: No conozco ni sé cómo usar ninguna de las herramientas o componentes que hemos visto; ¿hay otro libro más adecuado para mí?

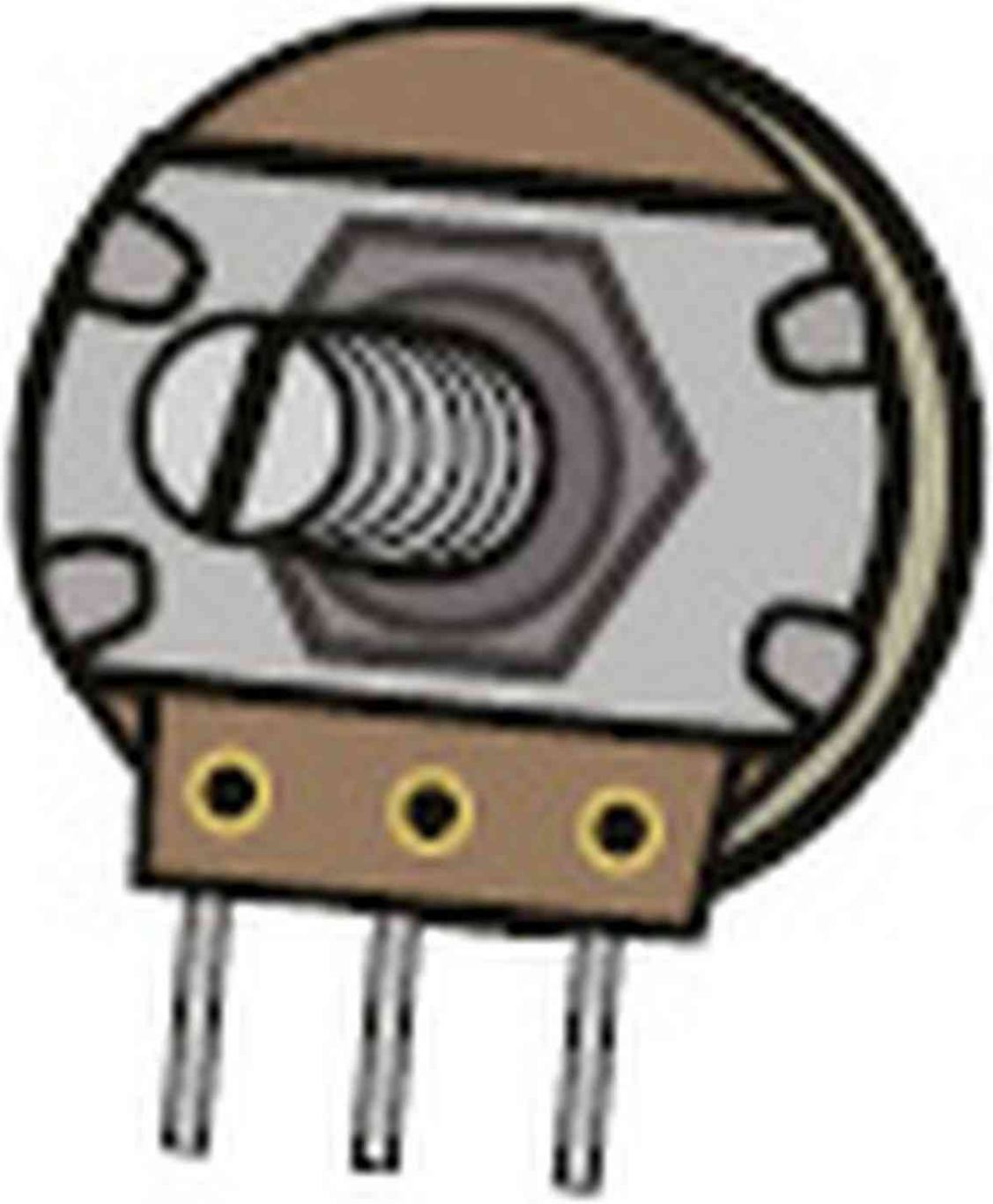
R: ¡No! Este libro está escrito para ti. En los siguientes capítulos trataremos aspectos específicos sobre cómo utilizar todas las piezas y herramientas que hemos enumerado. Siéntate y sigue leyendo.

P: No hay ninguna tienda donde vivo en la que pueda comprar esas piezas. ¿Me podéis recomendar sitios en línea donde pueda encontrarlas?

R: ¡Muy buena pregunta! Ya estás preparado para la siguiente sección.

RECURSOS

Hay proveedores que venden los componentes que se van a necesitar. Aquí están las URL de los sitios web de muchos de ellos. Además, donde vives puedes encontrar tiendas u otros recursos.



Maker Shed (makershed.com)

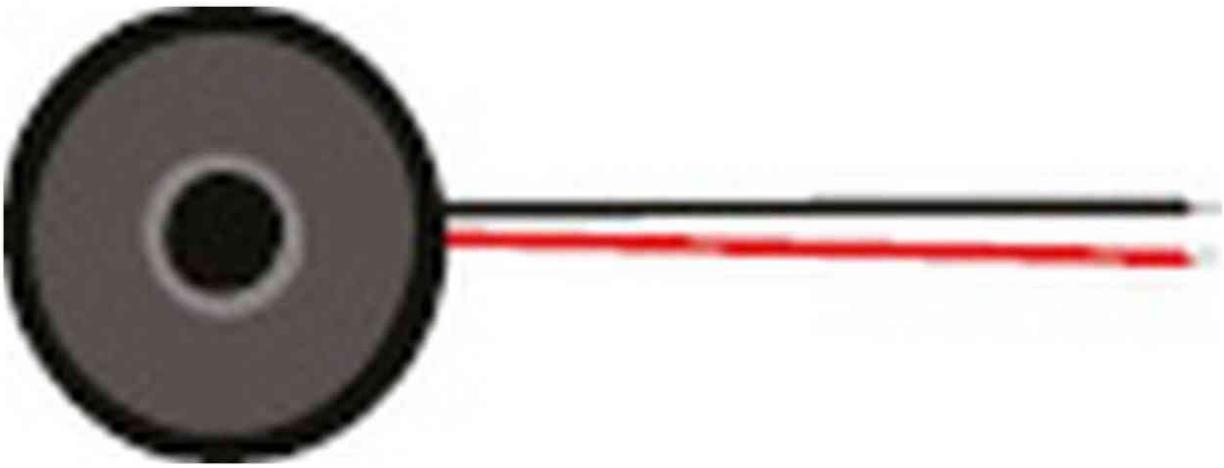
Selección de kits y componentes individuales Arduino. Algunas partes

electrónicas, enfocadas a la comunidad *maker*.

SparkFun Electronics (sparkfun.com)
Amplia gama de sensores y módulos de expansión. Arduinos clásicos y en su versión casera.

Adafruit Industries (adafruit.com)

Arduinos y módulos de expansión, sensores y componentes electrónicos.



Jameco Electronics (jameco.com)

Casi todos los componentes electrónicos, un sinfín de botones e interruptores.

Mouser Electronics (mouser.com)

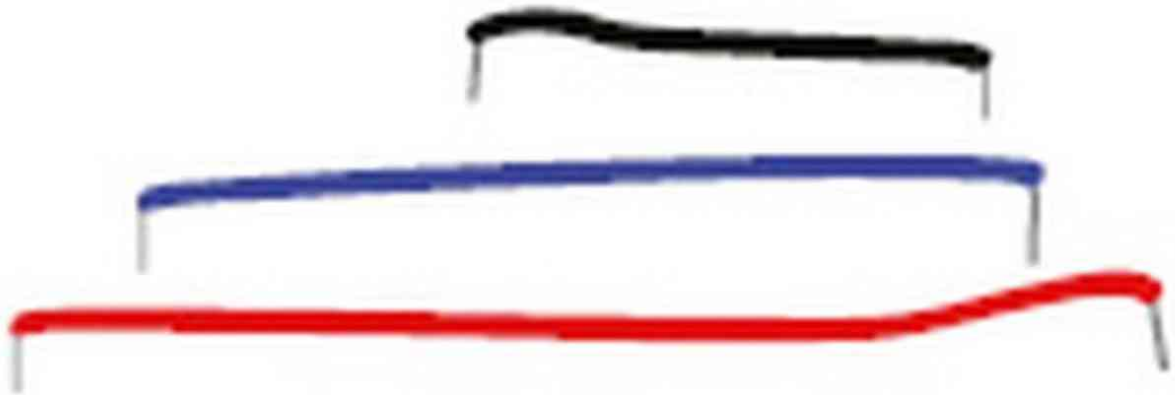
Algunos modelos de Arduino, mucha electrónica, sensores, y otros componentes.

Digi-Key Electronics (digikey.com)

Excelente para pedir componentes, chips, etc.

Micro Center (microcenter.com)

Una gran cantidad de componentes y modelos de Arduino, tienen tiendas tradicionales y un sitio web.



Kits

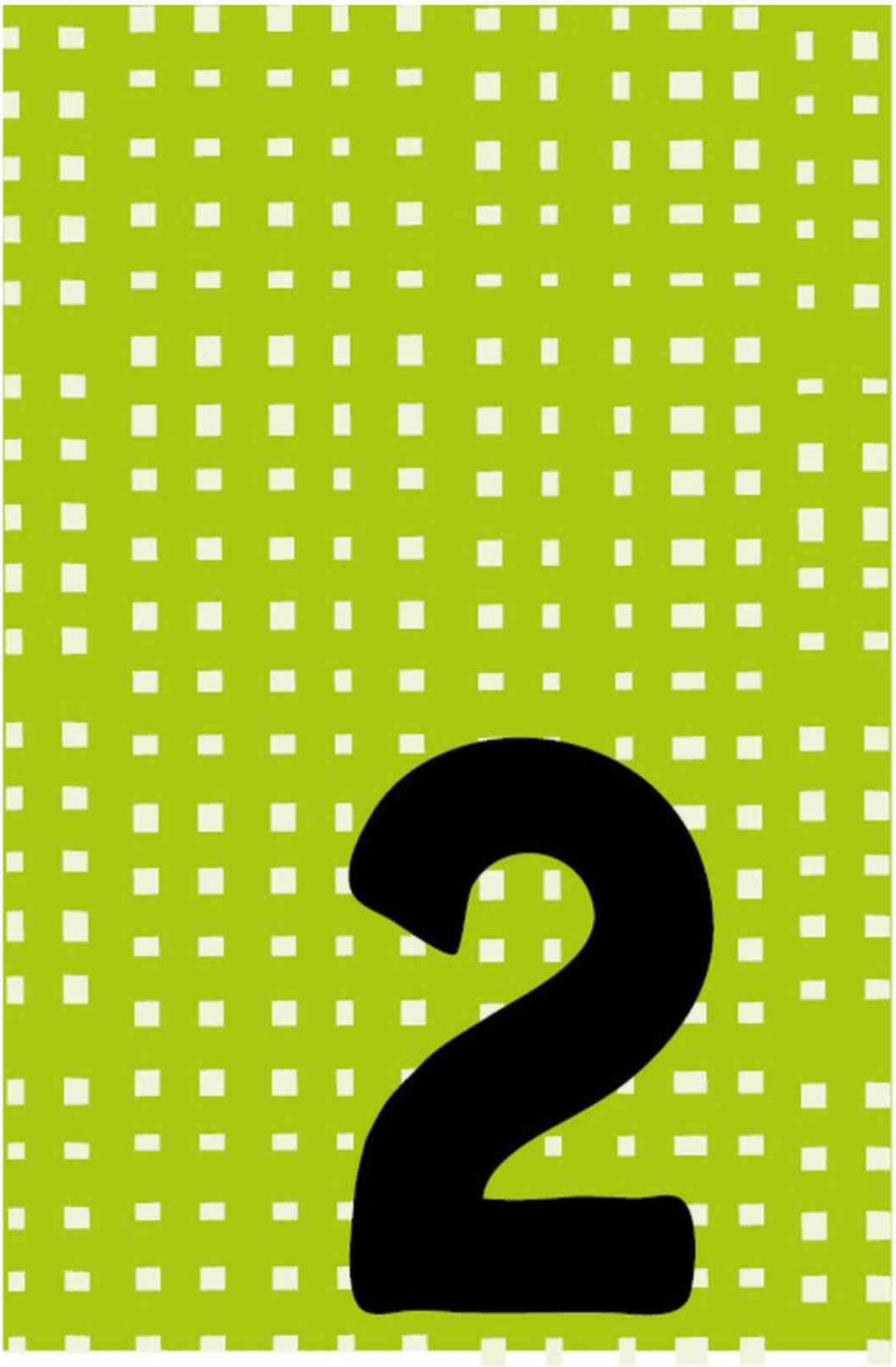
Los kits están disponibles en algunos de los proveedores mencionados antes, que tienen la mayoría de las piezas que necesitarás para llevar a cabo los proyectos. En cada capítulo revisaremos exactamente qué es lo que necesitas para desarrollar los proyectos. Estos son algunos de los kits disponibles, pero hay muchos más:

- Un kit desarrollado por el equipo de Arduino (arduino.cc/en/Main/ArduinoStarterKit). Se puede adquirir en algunos proveedores.
- Este kit está disponible en Maker Shed: makershed.com/products/make-getting-started-with-arduino-kit-special-edition
- Adafruit Industries tiene algunos kits, incluido este: adafruit.com/products/193

RESUMEN

Este capítulo te prepara para utilizar el Arduino. Ahora ya sabes dónde conseguir los elementos necesarios, puedes identificar algunos componentes y herramientas que vas a utilizar, y ya sabes algo sobre las contribuciones del movimiento de código abierto.

En el siguiente capítulo estudiaremos el Arduino Uno con más detalle y te mostraremos cómo conectarlo al ordenador.


2

Tu Arduino

A hora que ya tienes tu Arduino y algunas de las piezas y herramientas, veámoslas más detalladamente. Arduino es la solución ideal para resolver tus necesidades interactivas cotidianas. En este capítulo, estudiaremos las partes que forman el Arduino y cómo conectarlo a un ordenador y a una fuente de alimentación. También veremos cómo desempaquetar nuestras piezas electrónicas, las clasificaremos y aprenderemos más sobre ellas, tanto de los sitios web como de las hojas de datos.

PARTES DEL ARDUINO

Primero echemos un vistazo a las partes etiquetadas de la placa, como se muestra en la figura 2.1.

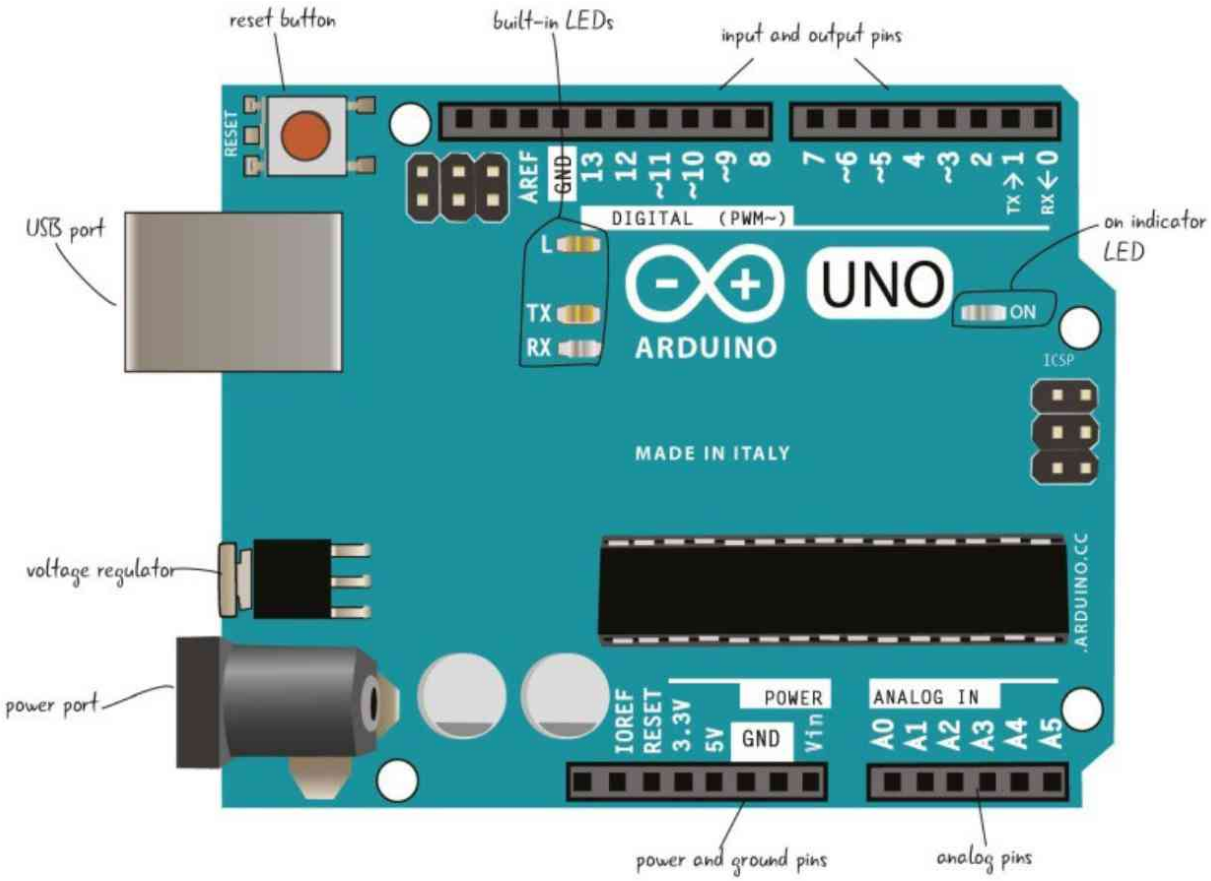


FIGURA 2.1 El Arduino Uno.

Vamos a analizar cada lado de la placa para verla con más detalle, para que puedas saber dónde se encuentra lo importante en el equipo Arduino.

Arduino en detalle

Aprendamos un poco más sobre lo que hay en la placa de Arduino.

Recuerda que hay diferentes tipos de placas, así que la tuya puede parecer ligeramente diferente. Estas ilustraciones son de la revisión 3 del Arduino Uno. Veremos en primer lugar el lado izquierdo de la placa, con el botón de reinicio, puerto USB, regulador de voltaje y puerto de alimentación (como se muestra en la figura 2.2).

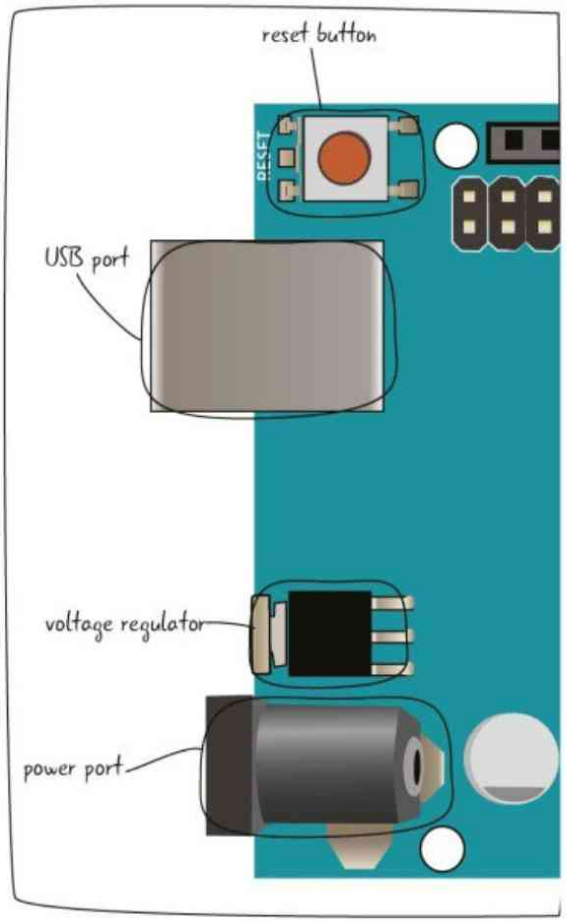


FIGURA 2.2 Lado izquierdo de la placa Arduino Uno.

Botón de reinicio

De la misma forma que apagas el ordenador y lo vuelves a encender de nuevo, algunos de los problemas con el Arduino se pueden resolver pulsando el botón de reinicio. Este botón reinicia el código cargado en ese momento en el equipo Arduino. El botón puede estar ubicado en un lugar diferente al que aparece en la figura 2.2, pero es el único botón que hay.

Puerto USB

El puerto USB requiere un cable estándar USB tipo A-B, que a menudo utilizan impresoras u otros periféricos de ordenador. El puerto USB cumple dos propósitos: en primer lugar, permite la conexión mediante un cable a un ordenador para programar la placa. En segundo lugar, el cable USB

proporciona energía al Arduino si no estás utilizando el puerto de alimentación.

Regulador de voltaje

El regulador de voltaje convierte la energía que llega al puerto de alimentación en el valor estándar de 5 voltios y 1 amperio que utiliza Arduino. ¡Ten cuidado!, porque este componente se calienta mucho.

Puerto de alimentación

El puerto de alimentación tiene un conector tipo barril que recibe la energía de un enchufe en la pared (a través de lo que se llama a menudo un transformador de corriente portátil) o de una batería. Esta alimentación se utiliza en lugar del cable USB. El Arduino admite un amplio margen de voltaje (5 V–0 V C.C.), pero sufrirá daños si se conecta a valores de voltaje superiores a 5 voltios.

Vamos a ver ahora más de cerca el otro lado de la placa (figura 2.3), que incluye los pines de señales digitales, analógicas y de alimentación, así como el chip que usa la placa.

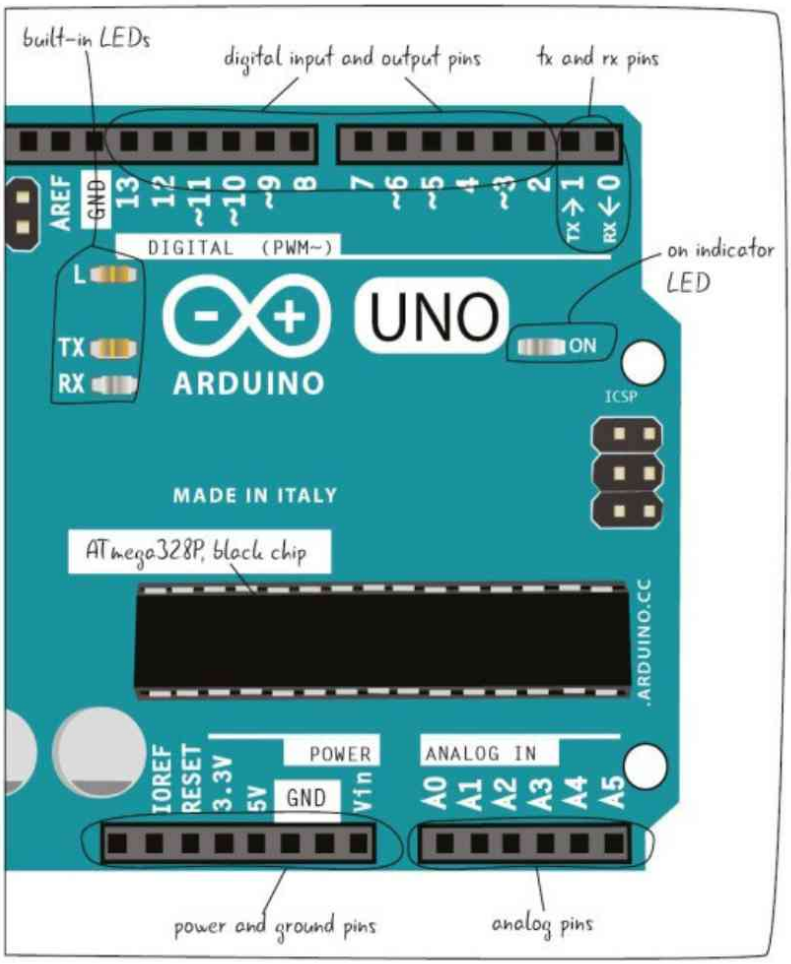


FIGURA 2.3 Lado derecho del Arduino Uno.

Ledes integrados

Los ledes etiquetados con TX y RX indican si el Arduino envía o recibe datos. El led que lleva la etiqueta L está conectado al pin 13.

Led indicador de encendido

Cuando luce, este led indica que Arduino está encendido.

Pines de E/S digitales

A los pines hembra de este lado de la placa se los denomina pines de E/S digitales. Se utilizan para detectar información del mundo exterior (entradas) o para controlar luces, sonidos, o motores (salidas).

Pines de TX/RX

Los pines 0 y 1 son especiales, y están etiquetados con TX y RX. Más adelante

los trataremos con mayor detalle, pero es recomendable dejar estos pines vacíos. Cualquier cambio que realices en tu programa no se cargará si hay algo conectado al pin 0.

Chip ATmega328P

El chip de color negro situado en el centro de la placa es un ATmega328P. Es el "cerebro" del Arduino: interpreta las entradas/salidas y el código cargado en tu Arduino. Los otros componentes de la placa posibilitan la comunicación con este chip cuando desarrollas algún proyecto.

Pines de alimentación y de toma de tierra

En este grupo se localizan los pines de alimentación. Se pueden utilizar para alimentar el circuito de la placa de pruebas desde el Arduino.

Pines de señal analógica

Estos pines recogen las señales de los sensores dentro de un rango de valores (analógicos), en lugar de las señales que indican simplemente si algo está encendido o apagado (valores digitales).

Ahora vamos a conectar el Arduino a tu ordenador. Todavía no lo vamos a programar. Antes veremos cómo conectarlo al ordenador a través del cable USB.

CONEcta tu ARDUINO AL ORDENADOR

Vas a necesitar un cable A-B, el ordenador y el Arduino Uno. Si lo que tienes es el último modelo de MacBook, necesitarás también un adaptador USB-C a USB.

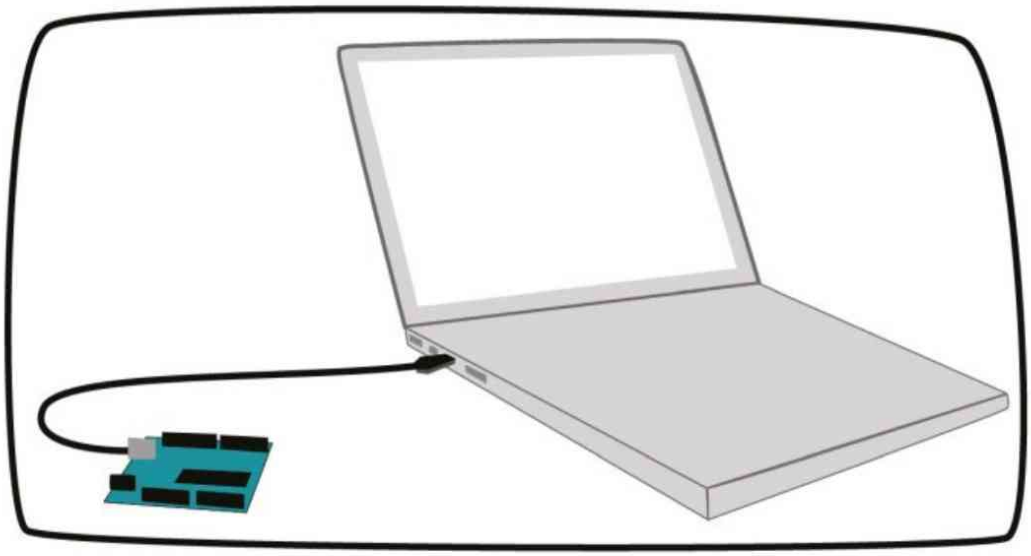


FIGURA 2.4 Conexión del Arduino al ordenador.

Figura 2.5:**Primer plano del puerto USB.**

En primer lugar, conecta el cable USB a uno de los puertos del ordenador, como se muestra en la figura 2.4. Cualquier puerto que esté disponible, como se muestra en la figura 2.5, debería funcionar bien.

Después de conectar el cable USB al ordenador, conecta el otro extremo al puerto USB del Arduino. El puerto USB está etiquetado en la figura 2.6.

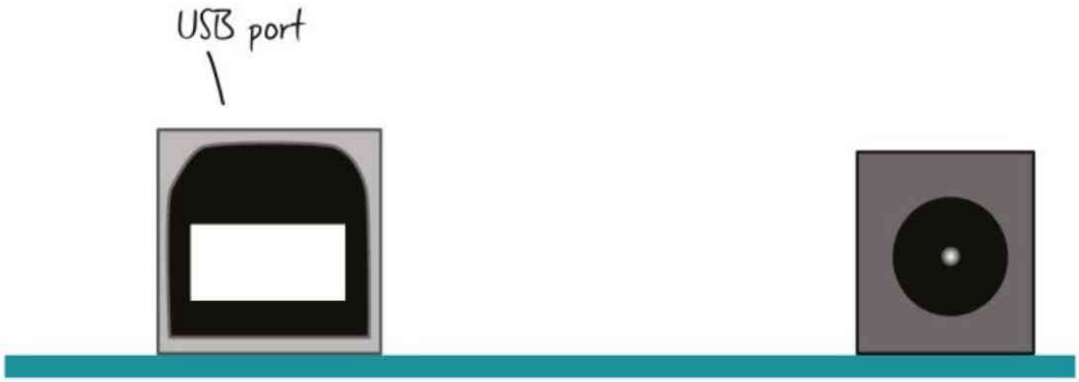


FIGURA 2.6 Puerto USB del Arduino.

En la figura 2.7 se puede apreciar la vista superior del puerto USB de Arduino con el cable USB A-B.

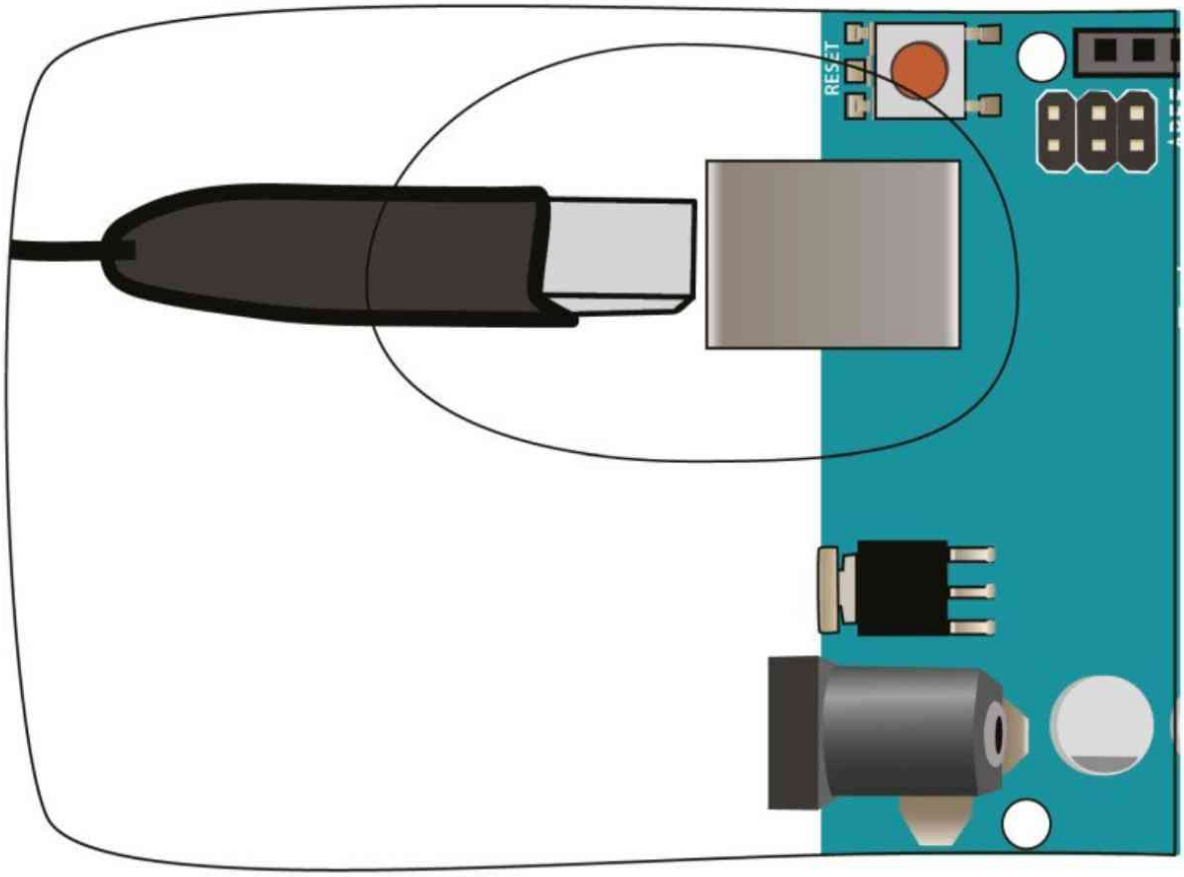


FIGURA 2.7 Vista superior de la conexión del cable USB al equipo Arduino.

Conexión a Arduino

¿Qué ocurre cuando conectas el Arduino? Se encenderá el led de alimentación etiquetado con ON. Y si esta es la primera vez que se conecta, el led situado cerca del pin 13 parpadeará, como se muestra en la figura 2.8.

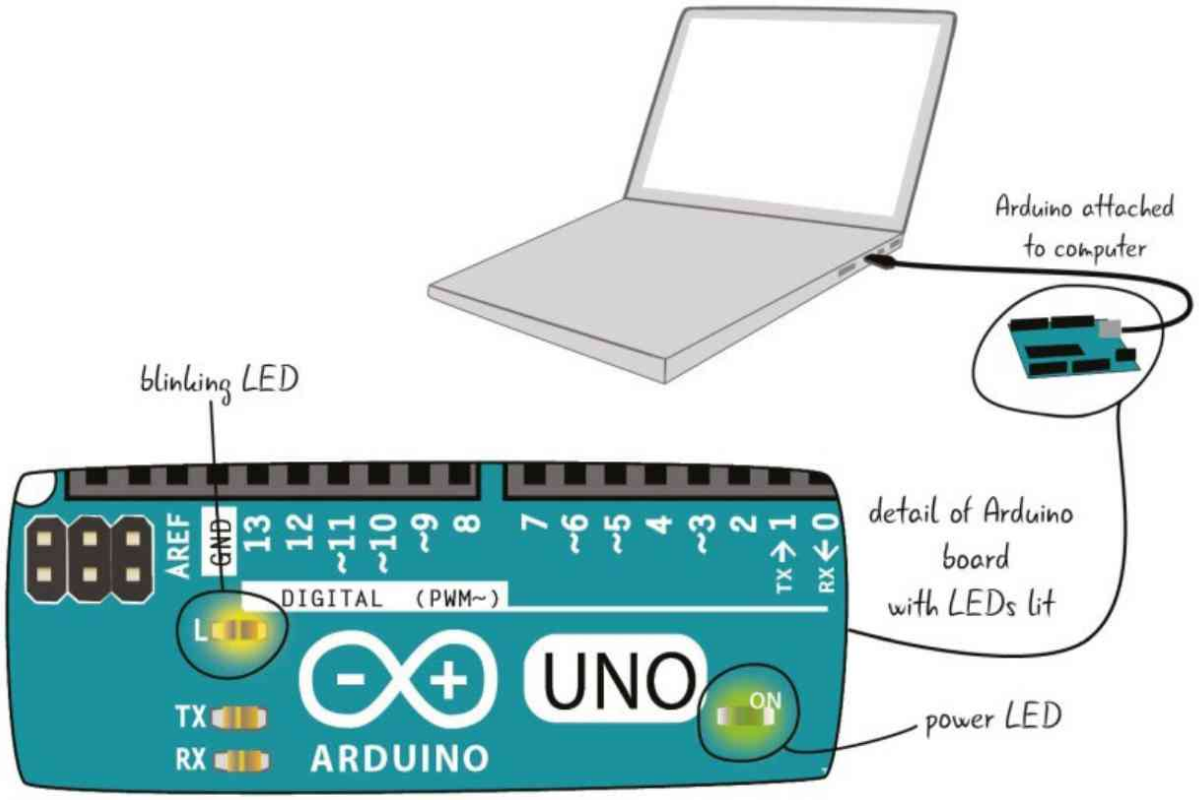


FIGURA 2.8 Los ledes lucen cuando el Arduino recibe corriente del ordenador.

¡Has alimentado el equipo Arduino por primera vez!

Siempre puedes utilizar un cable USB y un puerto del ordenador para alimentar el equipo Arduino. También se puede alimentar de la corriente procedente de una fuente de alimentación conectada a un enchufe de la pared.

> **Note** El Arduino se puede apagar desconectándolo del puerto USB o del puerto de alimentación.

Alimentación de Arduino desde una fuente de alimentación

Necesitas una fuente de alimentación de 9–12V C.C. y un Arduino. El primer paso es desenchufar el cable USB, lo que apagará el Arduino. La figura 2.9 muestra el puerto de alimentación del Arduino.

Warning Siempre que realices cambios, ¡tienes que desconectar el Arduino de cualquier fuente de alimentación!

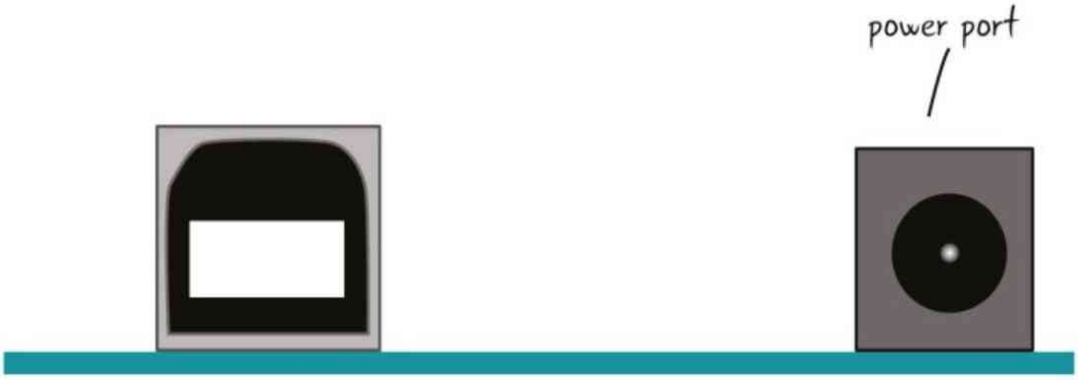


FIGURA 2.9 Puerto de alimentación del Arduino.

Conecta la fuente de alimentación al puerto de alimentación del Arduino (figura 2.10).

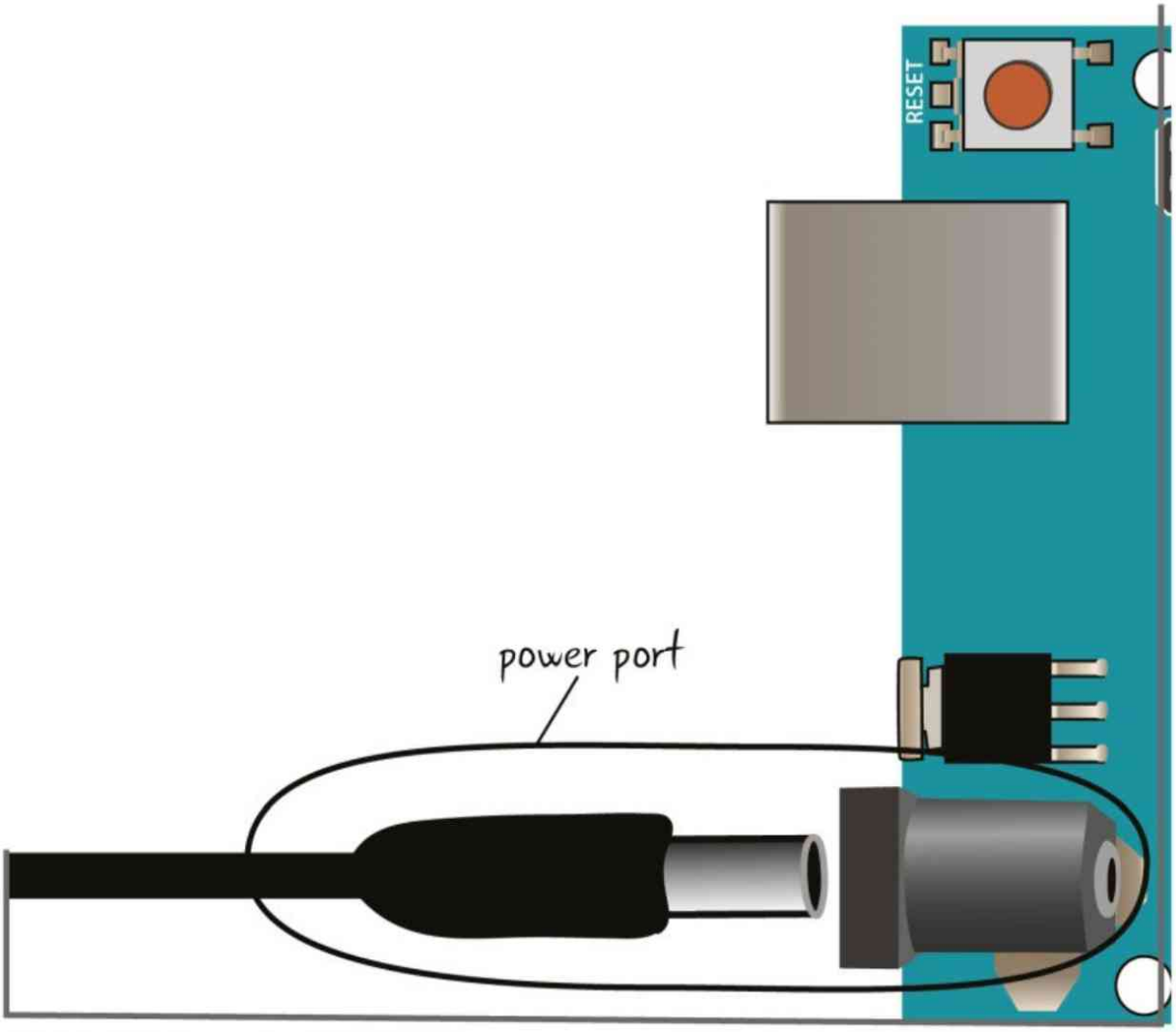


FIGURA 2.10 Vista superior del puerto de alimentación del Arduino.

A continuación conecta la fuente de alimentación a un protector contra sobretensiones, y luego a un enchufe en la pared, como se muestra en la figura 2.11.

¿Qué ocurre ahora? Lo mismo que cuando conectaste el Arduino al ordenador con el cable USB: el led etiquetado con ON indica que el Arduino tiene corriente. Y si el equipo no tiene problemas, el led más próximo al pin 13 empieza a parpadear, como se ve en la figura 2.12.

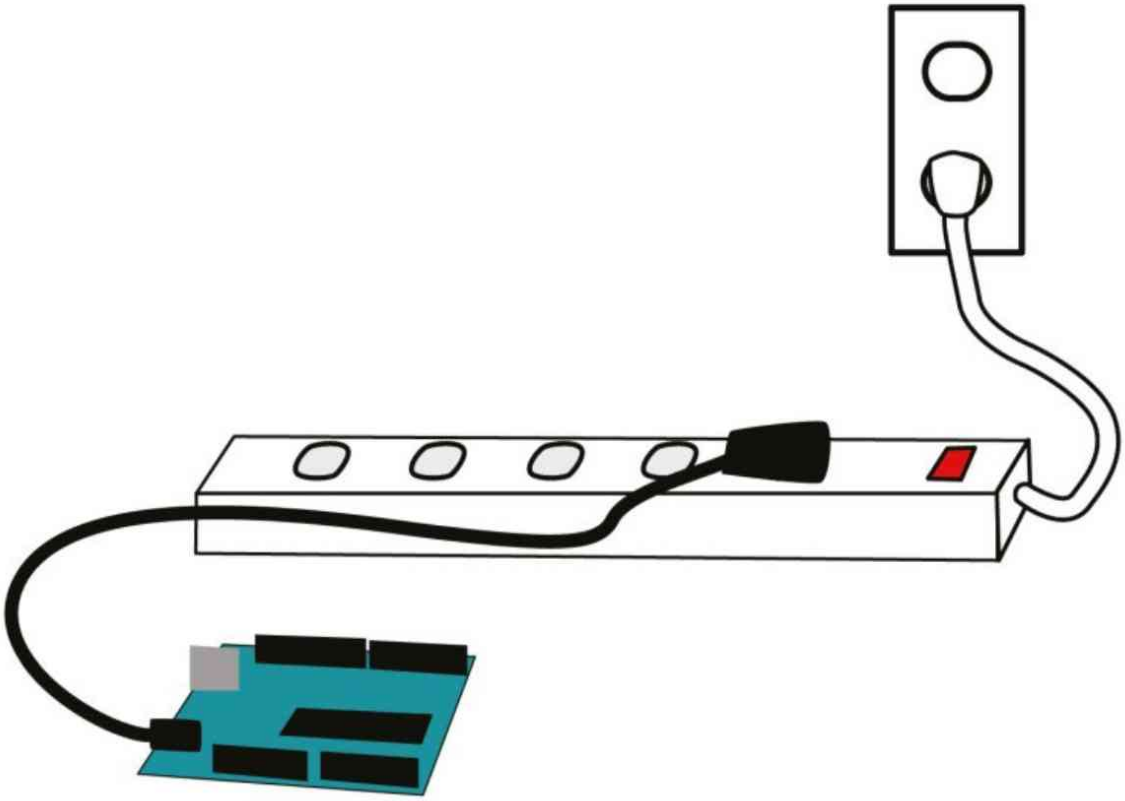


FIGURA 2.11 Conexión de la fuente de alimentación a un protector de sobretensiones.

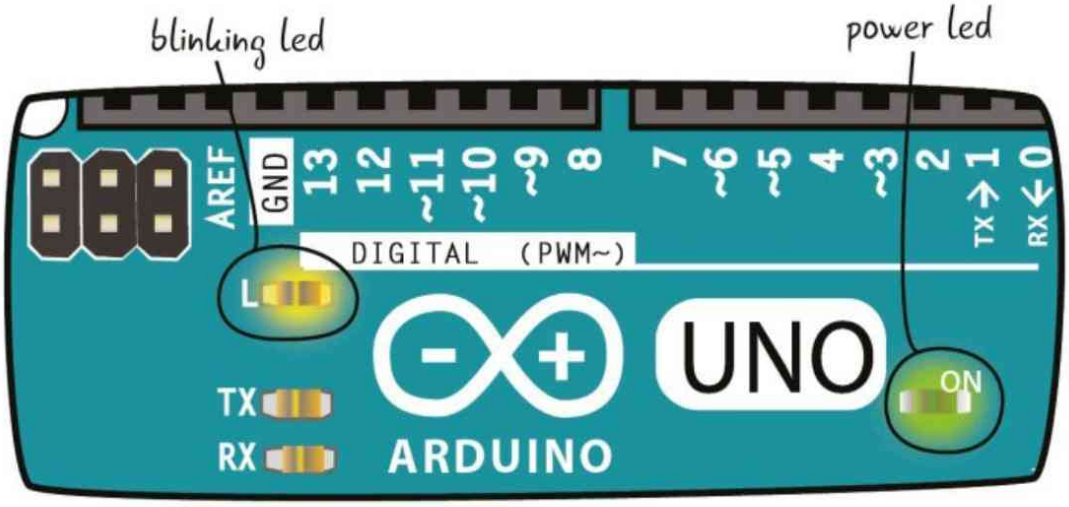


FIGURA 2.12 Ledes de Arduino intermitentes.

Ahora ya conoces las dos formas de alimentar al equipo Arduino. Puedes cambiar la fuente de alimentación a medida que tu proyecto avanza, y no

estás obligado a utilizar un procedimiento u otro.

COMPONENTES Y HERRAMIENTAS

Ahora que ya dispones de los componentes que aparecen en la lista de piezas (figura 2.13), necesitas aprender algo más sobre cada una de ellas. Hay disponibles diferentes tipos de recursos que pueden ayudarte a determinar qué piezas utilizar y dónde ubicarlas.

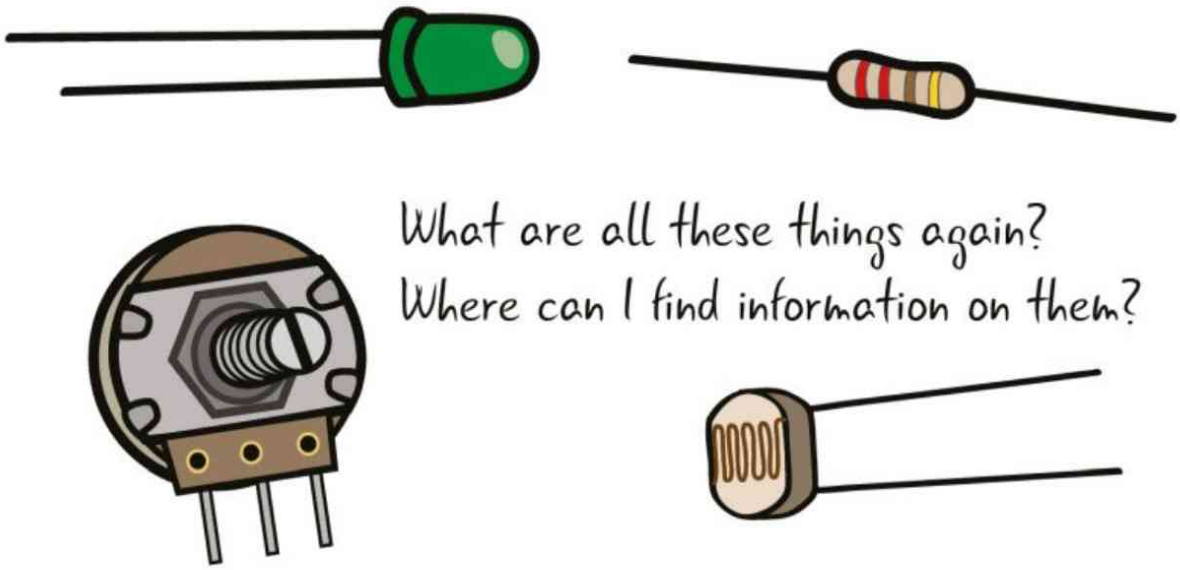


FIGURA 2.13 ¿Dónde puedo encontrar información sobre los componentes?

Clasificación de las piezas

Lo mejor que puedes hacer al desembalar las piezas es clasificarlas por tipos. Está muy bien tener todas las resistencias juntas y separadas de los ledes, o incluso separar los ledes por colores y las resistencias por valores. La mayor parte de las ferreterías o tiendas de artículos de artesanía venden cajas de plástico para facilitar la clasificación de las piezas y saber dónde están cuando las necesites. Recomendamos algo parecido a la caja que se muestra en la figura 2.14.

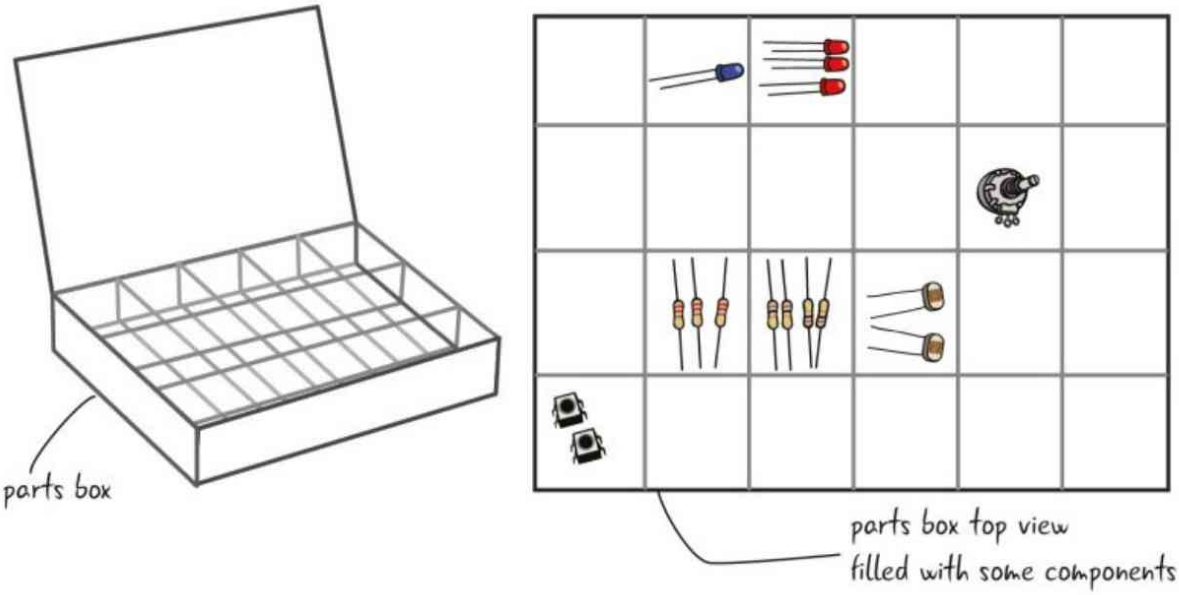


FIGURA 2.14 Clasificar los componentes ayudará a que te resulten familiares.

Números de piezas y guías de tiendas

Ahora que tienes las piezas clasificadas y puedes identificarlas, ¿dónde deberías buscar información sobre ellas? El primer lugar donde debes buscar información sobre los componentes es en los componentes mismos. Resistencias, ledes y la mayoría del resto de componentes son lo suficientemente diferentes como para que aprendas rápidamente a identificarlos. A menudo cada componente tendrá un número de parte en algún sitio, que te puede ayudar a encontrar el sitio web de un proveedor o fabricante. Cuando haces el pedido de un componente o un kit, la tienda te envía también la documentación, o te dirige a una página en su sitio web. Revisa siempre en primer lugar el sitio web de un proveedor de repuestos y te ahorrarás dolores de cabeza.

BUSCAR MÁS INFORMACIÓN: HOJAS DE DATOS

Si no puedes encontrar la información que estás buscando, ya sea en el componente o en su sitio web, lo siguiente que debes buscar es la hoja de

datos del componente. Puedes encontrarla introduciendo el número de pieza, seguido de "hoja de datos", en tu buscador favorito en línea. No buscarás solo el nombre de la pieza, ya que es posible que haya muchas versiones en línea distintas de la pieza con información diferente. Por ejemplo, ¡hay muchos ledes distintos!

Las hojas de datos electrónicas informan del comportamiento, las funciones y las limitaciones de los componentes electrónicos. Tienen una enorme cantidad de información, desde la temperatura de funcionamiento, su comportamiento y los posibles diagramas de cableado, hasta la composición del material y sus aplicaciones industriales.

Por ejemplo, aquí vamos a ver cómo encontrar una hoja de datos en línea para uno de los ledes.

1. Encuentra el número que identifica el led en la factura del proveedor al que le compraste la pieza. Si no puedes encontrar ningún número, utiliza la referencia WP7113SRD para ledes rojos muy brillantes.
2. Abre el navegador y escribe el número de pieza en nuestro buscador favorito, y también "hoja de datos". Si utilizas el número de pieza de nuestro ejemplo, las palabras de búsqueda serán "WP7113SRD hoja de datos".
3. Los resultados de la búsqueda incluirán hojas de datos sobre la pieza, a menudo en formato PDF. Elige un par de enlaces y haz clic en ellos. Echa un vistazo a los resultados y asegúrate de que coincidan aproximadamente con el número de pieza que buscas.

A menudo puede resultar abrumador examinar la hoja de datos para encontrar la información que necesitas, pero las hojas de datos son útiles, especialmente cuando no estás seguro de qué componentes estás usando. Comencemos mirando una hoja de muestra, como la que se puede ver en la figura 2.15.

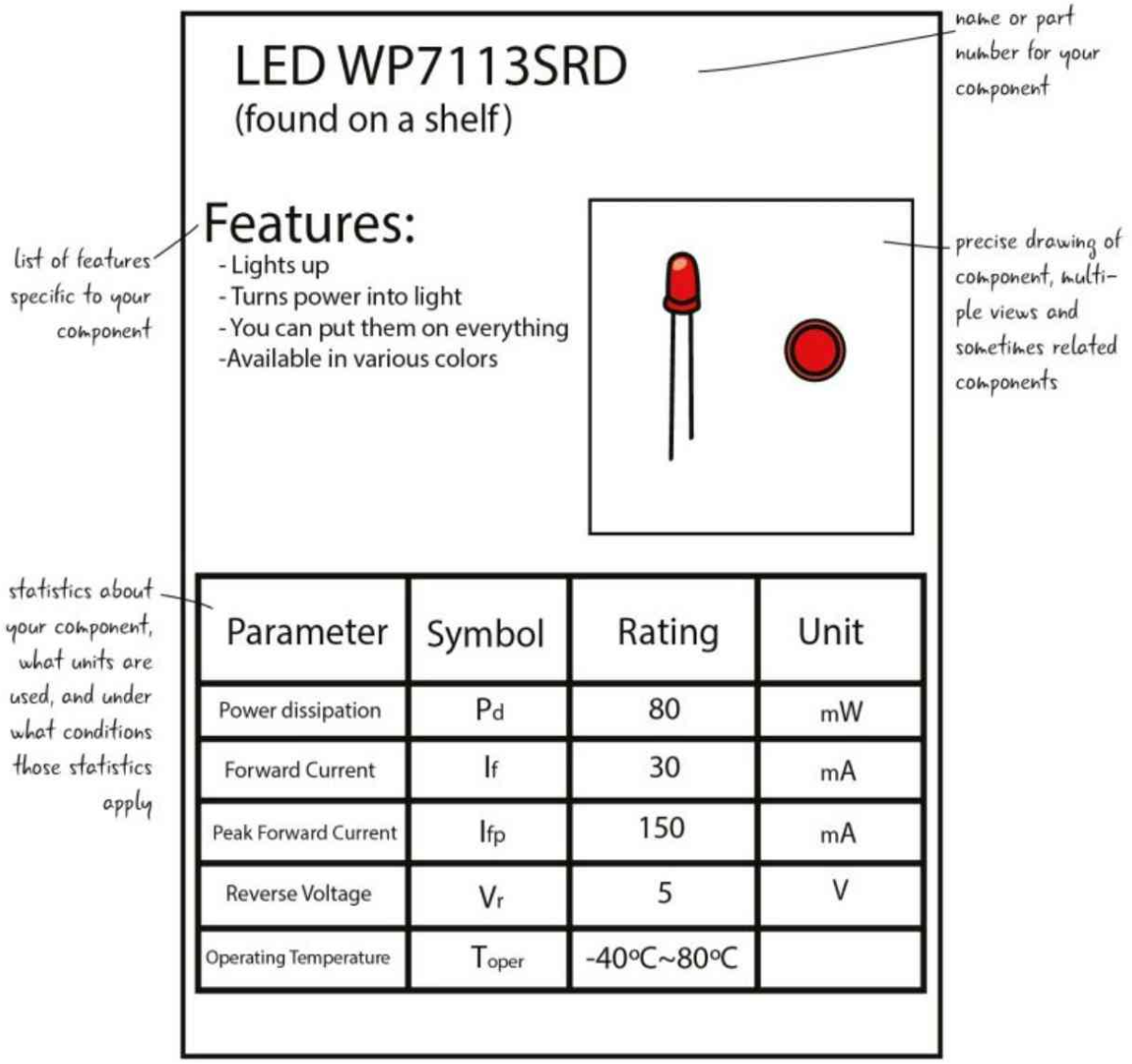
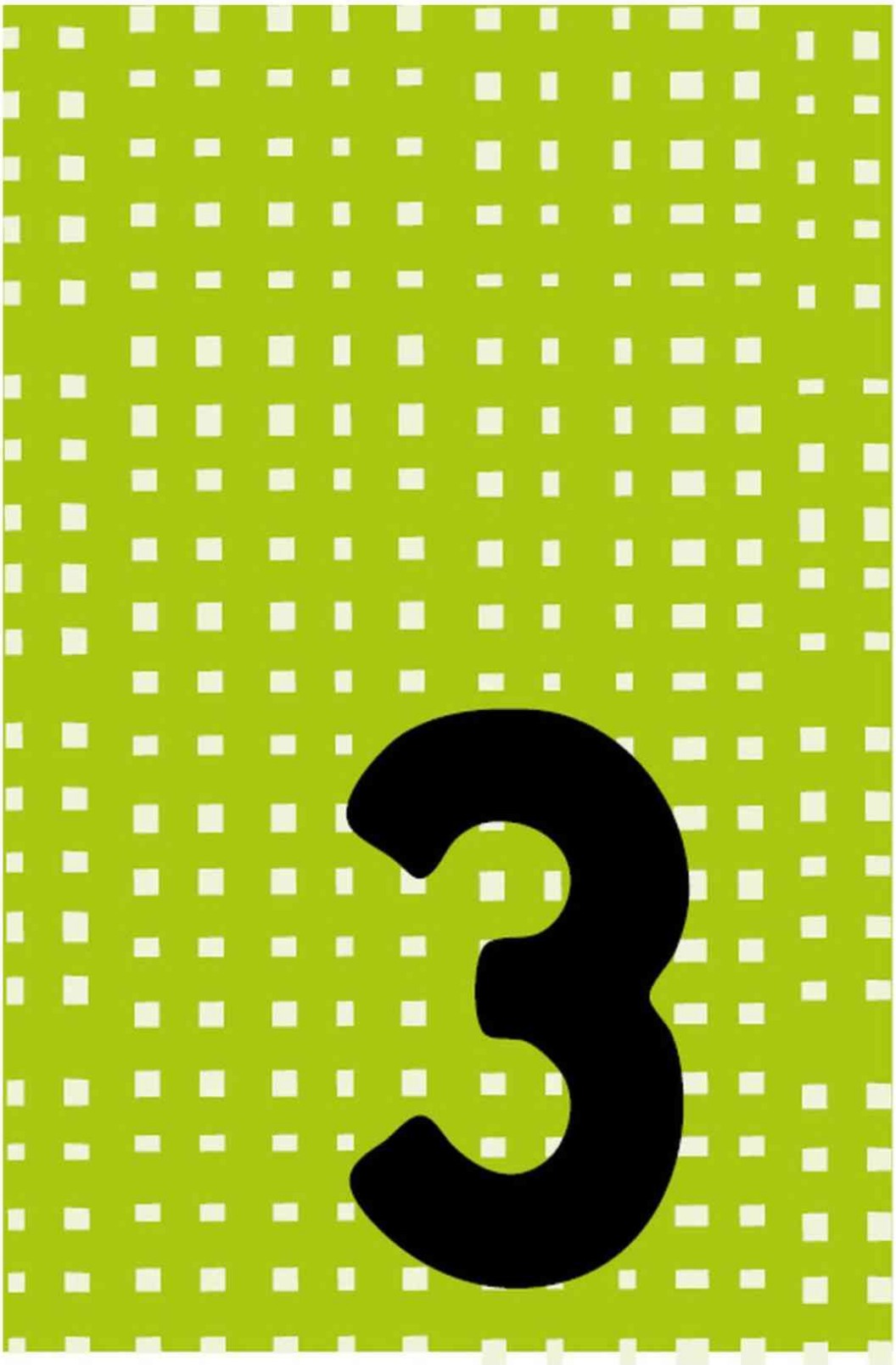


FIGURA 2.15 Hoja de datos de un led encontrada en un estante.

La hoja de datos contiene muchas partes y no toda la información técnica es importante para tu proyecto, pero puede ser de utilidad si te quedas atascado.

RESUMEN

Ahora te sentirás más cómodo con el diseño de tu Arduino. Sabes cómo encenderlo desde el puerto USB y desde el puerto de alimentación. Si alguna vez no estás seguro de los componentes, puedes buscarlos en línea, en el sitio web de la tienda donde los compraste o en su hoja de datos. En el próximo capítulo vamos a echar un vistazo a la utilización de algunos componentes para montar nuestro primer circuito.


3

primer contacto con el CIRCUITO

En el último capítulo has aprendido algo sobre el Arduino y las partes que lo forman. También se presentaron algunos de los componentes y herramientas que emplearás para desarrollar los proyectos de este libro. En este capítulo, aprenderás algunas cosas de la teoría y práctica de la electrónica que necesitarás saber para montar circuitos con el Arduino. Todavía no lo vamos a utilizar, pero lo haremos en breve.

EL CIRCUITO: COMPONENTES ELECTRÓNICOS BÁSICOS

El circuito es el componente básico en todos los proyectos electrónicos que desarrollemos con el Arduino.

Puedes llevar a cabo diferentes tipos de proyectos con un Arduino, el único límite es tu imaginación. Aunque existen muchos tipos diferentes de proyectos, todos los que incluye este libro se montan utilizando circuitos.

En primer lugar, veremos lo que es un circuito; luego pasarás a montar tu primer circuito. También veremos las técnicas para representar los circuitos electrónicos de forma visual y te explicaremos cómo probarlos.

La figura 3.1 ilustra algunos de los proyectos con Arduino. Puedes ver que los circuitos de estos proyectos adoptan diferentes formas. En el robot de cartón no se puede ver el circuito, pero es el circuito el que controla al robot.

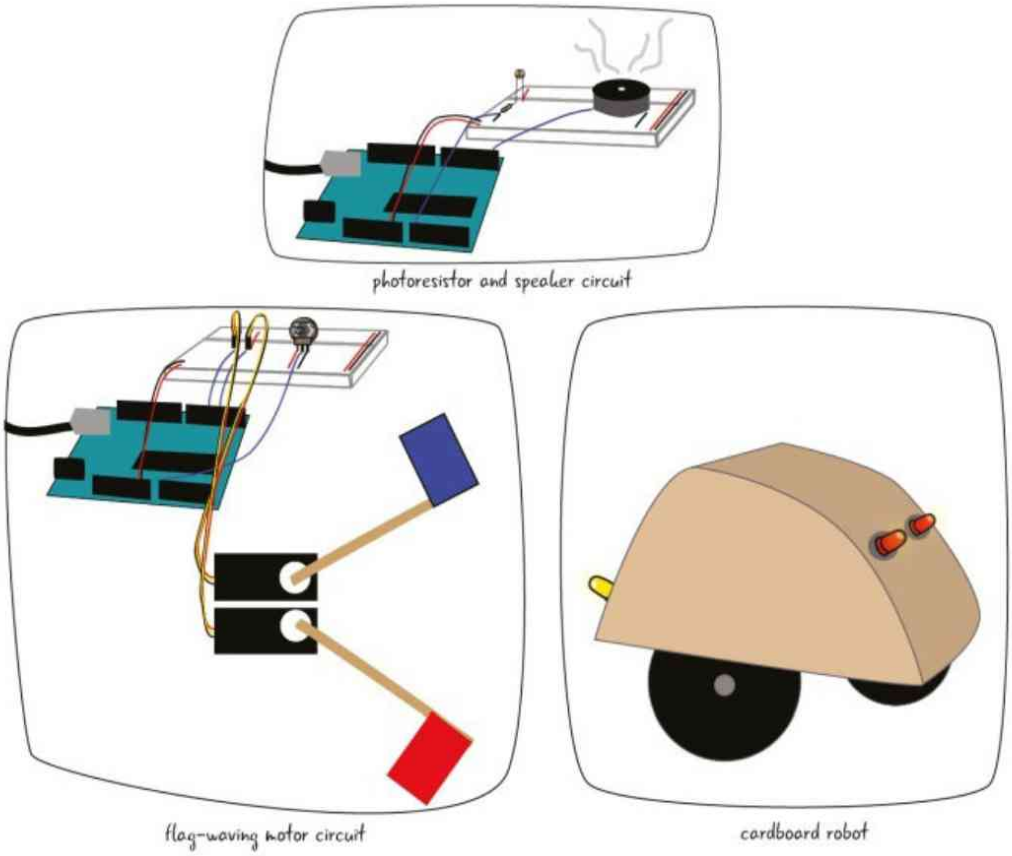


FIGURA 3.1 **Ejemplos de proyectos en los que se utiliza el Arduino como parte del circuito.**

Veamos más de cerca qué es un circuito.

¿Qué es un circuito?

Si alguna vez has estado en una carrera de coches sabrás que a la pista también la llaman circuito. La palabra circuito significa que existe una pista cerrada, como se ve en los ejemplos de la figura 3.2. Los coches arrancan en la línea de salida y terminarán en el mismo lugar.

FIGURA 3.2 Pistas de circuito.

Lo mismo ocurre con los circuitos electrónicos. El trazado de un circuito electrónico recorre una pista cerrada. Un circuito incluye todos los componentes electrónicos que se requieren para la tarea, así como los cables u otro material que permita que la electricidad fluya entre los componentes conectados, como se puede ver en la figura 3.3.

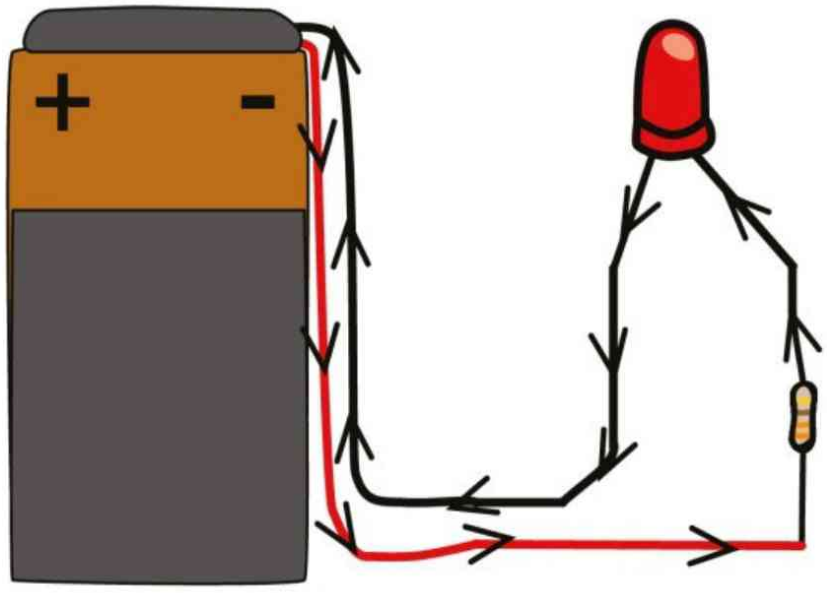


FIGURA 3.3 La corriente del circuito empieza y termina en la fuente de alimentación.

¿Por qué montamos circuitos?

Piensa por un momento en los interruptores de luz de casa como modelo. Para encender o apagar un interruptor debes estar en contacto físico con el interruptor. En nuestros proyectos, el Arduino controlará el comportamiento

de los componentes electrónicos. Nuestros componentes electrónicos estarán dispuestos en un circuito, y el Arduino debe formar parte de ese circuito para controlar su comportamiento.

Los circuitos permiten que el Arduino se conecte a los componentes eléctricos, para encender y apagar una serie de dispositivos (altavoces, ledes, motores, etc.) o extraer información del mundo exterior ("¿Cómo está de caliente?"; "¿Está encendido el interruptor?"; etc.). Hasta que descubramos cómo hacer que el Arduino se conecte al dispositivo, podemos controlarlo con electricidad, y más tarde, con programación.

¿Qué constituye un circuito?

Hay dos partes principales que conforman un circuito: las pistas conductoras y los componentes.

Pistas conductoras

Aunque la mayor parte de la atención en un circuito se centra en los componentes, no se puede tener un circuito sin algún tipo de conexión entre aquellos. Los ordenadores y dispositivos electrónicos llevan instaladas placas de circuito impreso (PCI). Las PCI, que no conducen la electricidad, están compuestas de placas base, un material sobre el que se aplican delgadas pistas de material conductor, como se ve en la figura 3.4. Las pistas conductoras conectan componentes que se han soldado a la PCI. Si observas una PCI, distinguirás las brillantes líneas plateadas que corren entre los componentes, y que los conectan. Estas líneas son como cables pegados a una superficie plana.

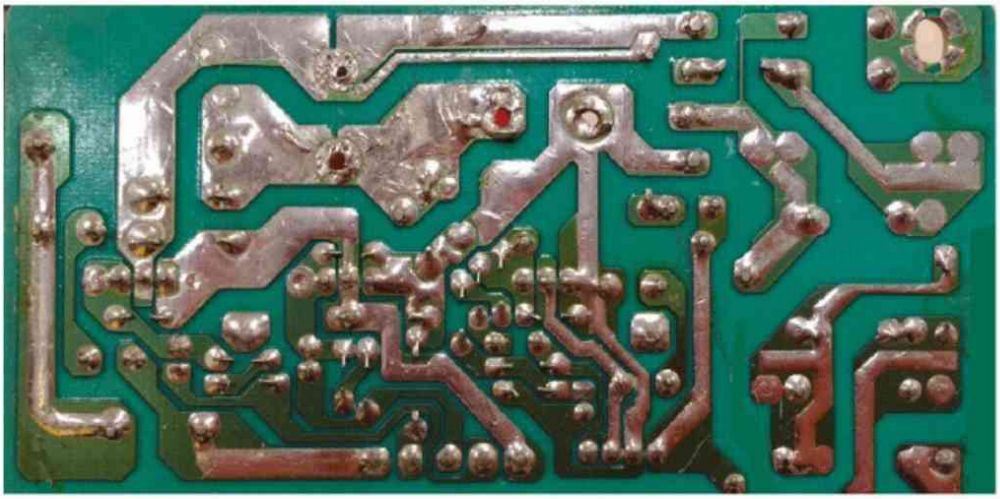


FIGURA 3.4 Detalle de una placa de circuito impreso.

Componentes

Los componentes son el otro requisito para poder montar un circuito completo. En el capítulo 1, "Introducción a Arduino", puedes ver la lista de todos componentes que tienes que comprar. Los componentes definen los puntos que se necesitan conectar en un circuito (figura 3.5).

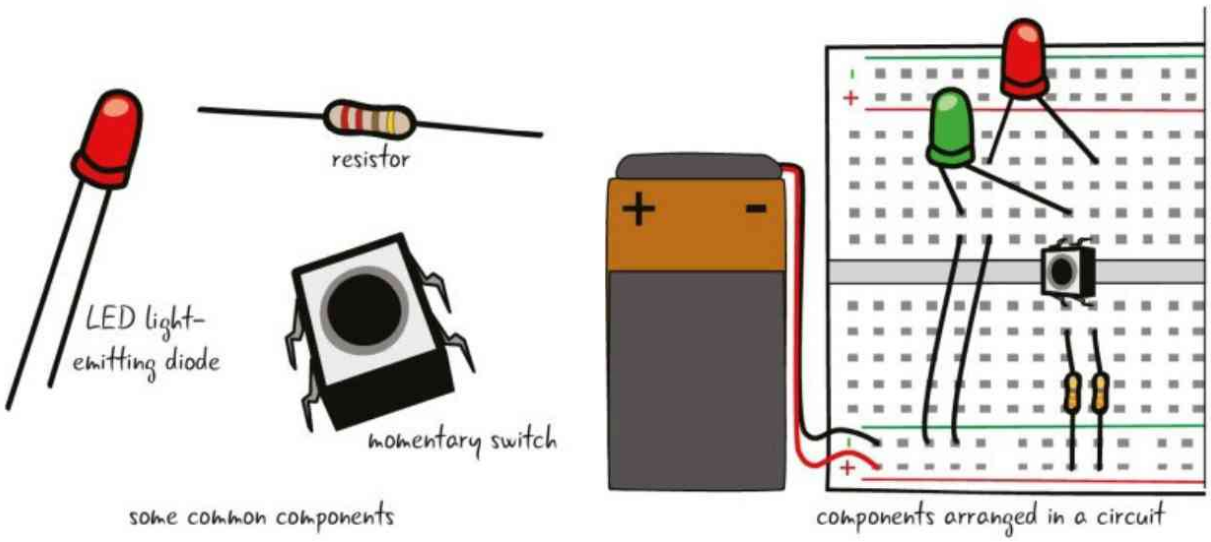


FIGURA 3.5 Los circuitos están formados por componentes.

En la figura 3.6 puedes ver que los conductores de los componentes actúan como líneas conductoras.

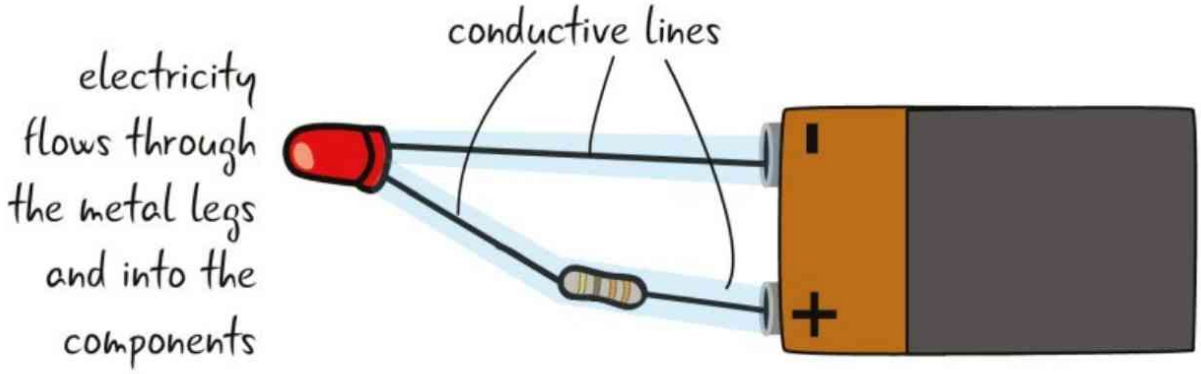


FIGURA 3.6 La electricidad fluye a través de las líneas conductoras.

¿Por dónde empezamos?

El primer circuito que vamos a montar juntos es una linterna cuya bombilla es un led alimentado por una batería. Este circuito es un gran proyecto para principiantes, porque cuando se enciende la luz se confirma visualmente que el circuito funciona. También muestra las técnicas básicas del montaje de circuitos que necesitarás aplicar en la realización de los proyectos de este libro.

La figura 3.7 es un dibujo del circuito ya montado, con los componentes que hemos comentado. En este capítulo y en los siguientes explicaremos con detalle lo que hacen los componentes. Por ahora, hay que saber que este circuito se montará con un led, una resistencia, un puente, una batería de 9 V, y la tapa de la batería, que se coloca en la batería y se conecta a la placa de pruebas, componentes que has visto en el capítulo 1,

Hay muchas maneras diferentes de representar o dibujar circuitos para transmitir la información necesaria. En la figura 3.7 presentamos una aproximación de cómo será el circuito cuando lo hayas montado. Esta no es siempre la forma más conveniente de ver lo que ocurre (algunos circuitos tienen muchos componentes conectados de formas complejas). Los esquemas son una gran manera de hacer un dibujo de un circuito, en el que se han simplificado los componentes y se describe como están conectados. Veamos más de cerca cómo funcionan los esquemas.

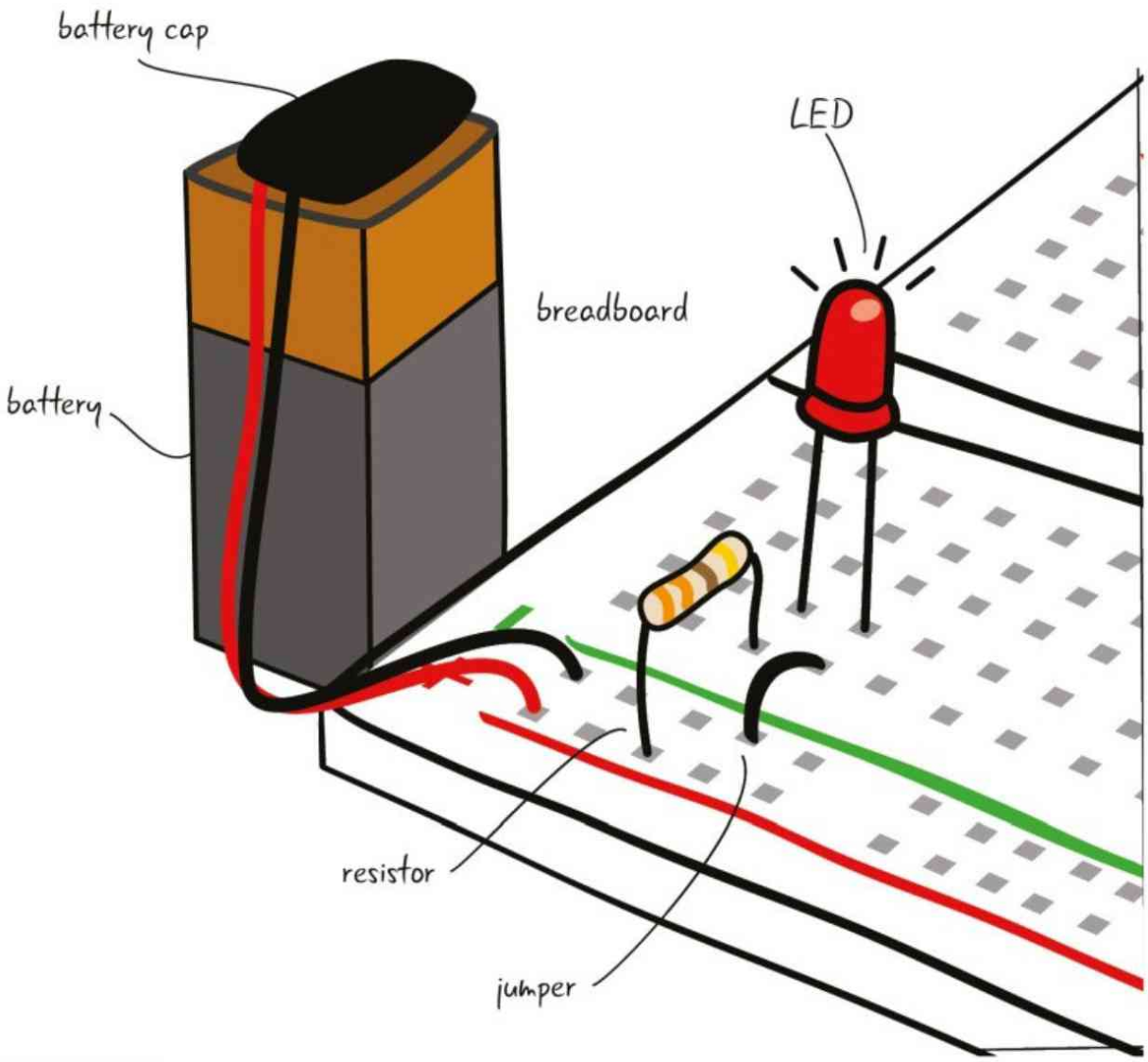


FIGURA 3.7 Circuito que vamos a montar.

EL ESQUEMA

Un esquema es un diagrama de *cómo se relacionan los componentes electrónicos en un circuito*. En un esquema, se pueden ver los componentes que forman parte del circuito y cómo se conectan entre sí. Empezaremos estudiando un esquema sencillo que representa nuestro circuito básico. Entraremos en breve en los detalles sobre lo que significa cada símbolo en el esquema, pero por ahora solo vamos a echarle un vistazo. La figura 3.8 compara un esquema del circuito que estamos a punto de montar con un dibujo del circuito.

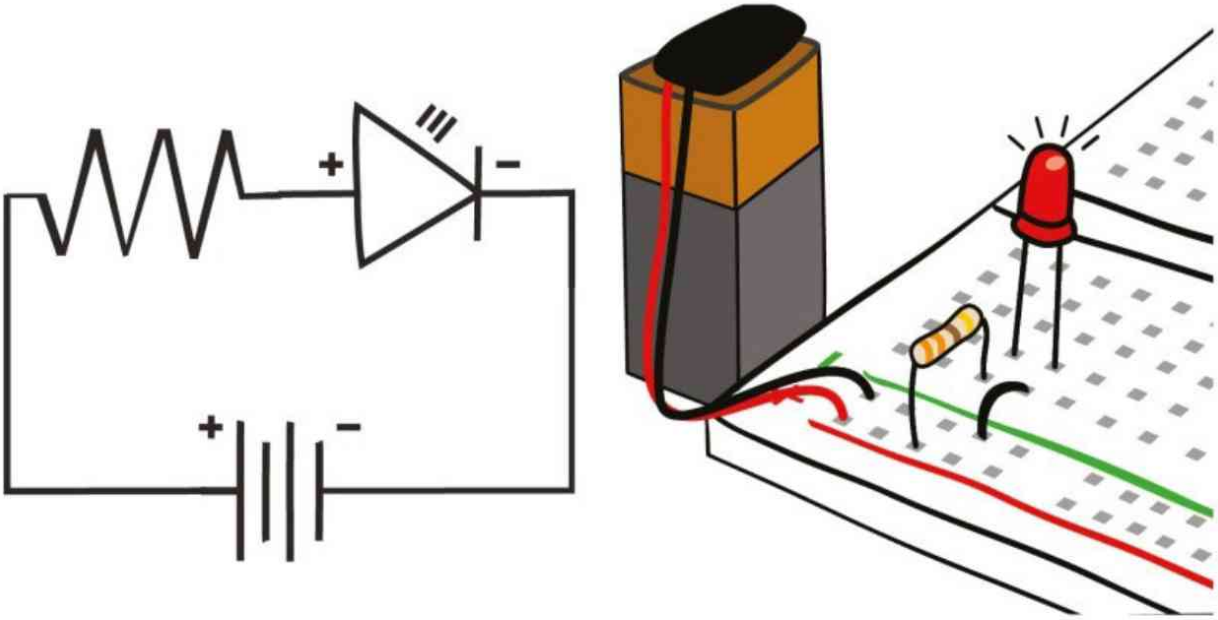


FIGURA 3.8 Esquema de circuito con un dibujo del mismo.

¿Por qué es importante aprender a leer un esquema?

La mayor parte de los proyectos y componentes electrónicos se representan mediante esquemas, no necesariamente con dibujos o fotografías. A medida que avancen tus habilidades en electrónica y quieras montar proyectos que no están en este libro, necesitarás ser capaz de leer y dibujar esquemas para poder investigar sobre los proyectos, así como describirlos y llevarlos a cabo.

Empezaremos con esquemas sencillos; crearemos representaciones

complejas a medida que montemos proyectos más complejos que aparecen en el libro. A medida que observes los esquemas en línea o en otra tipo de documentación, verás que a veces hay variaciones en las formas en que se dibujan o se organizan los símbolos. No tienes que preocuparte si los símbolos en los esquemas no responden a un solo modelo, como se muestra en la figura 3.9.

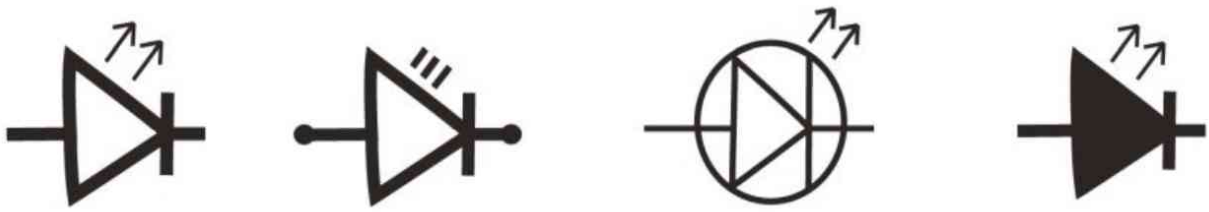


FIGURA 3.9 Símbolos esquemáticos para ledes.

Diagrama del circuito: el esquema

Ya sabes que un esquema es la manera estándar de representar las relaciones eléctricas en un circuito. Todos los componentes electrónicos que se utilizan normalmente tienen un símbolo para representarlos en los esquemas electrónicos con el fin de dejar claro cómo están unidos en el circuito. La figura 3.10 muestra el circuito básico de un led, una resistencia y una batería. El led tiene una orientación, un conductor positivo (ánodo) y un conductor negativo (cátodo), como dijimos en el capítulo 1.

El principal objetivo de los esquemas es diagramar cómo se conectan los componentes en el circuito, por lo que sacrificarán claridad en la configuración física de los mismos para centrarse en cómo se conectan a nivel electrónico.

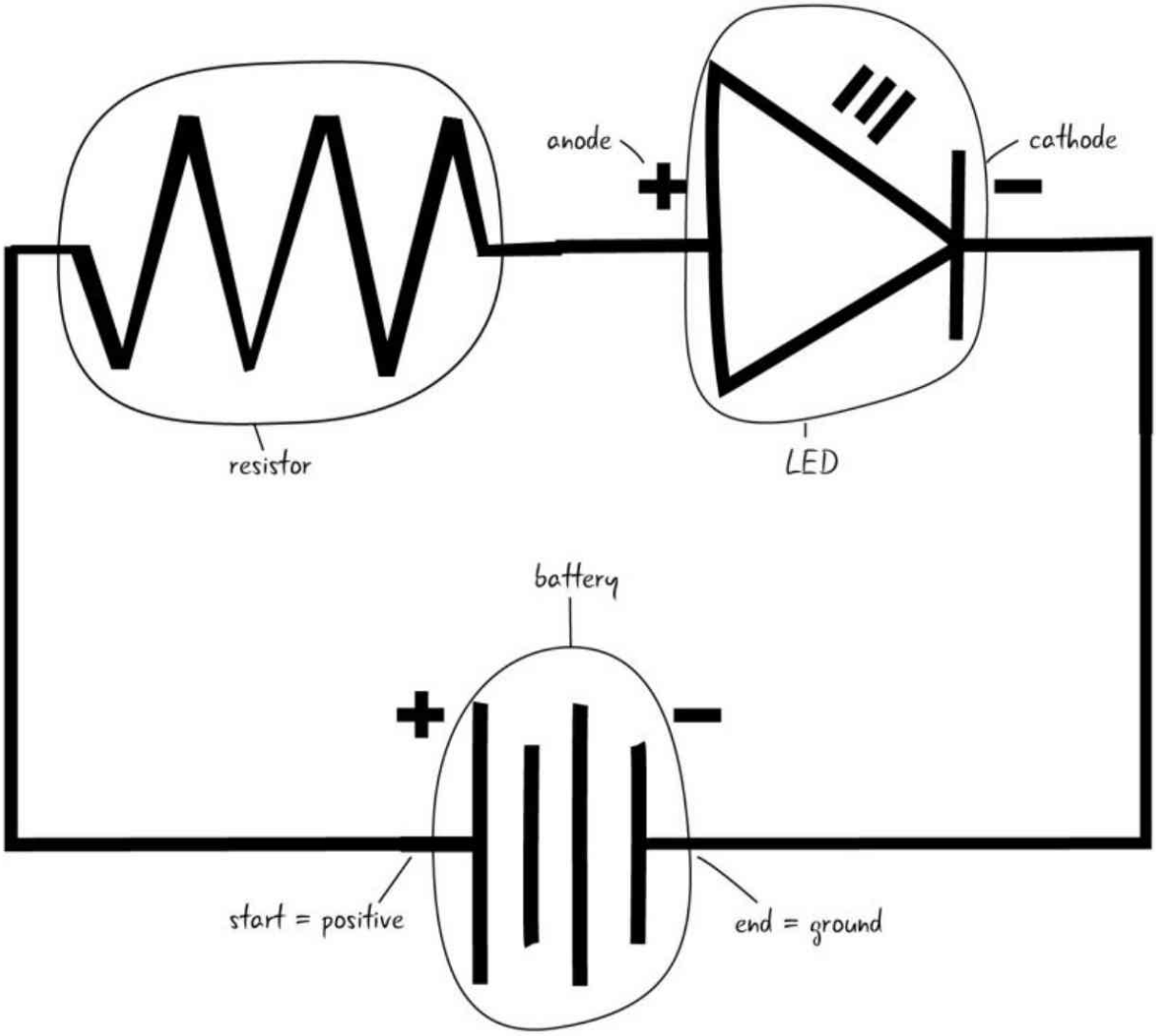
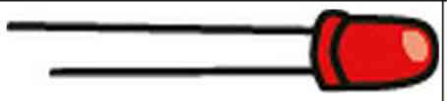


FIGURA 3.10 Esquema del circuito elegido.

La tabla 3.1 muestra los símbolos de los componentes que tiene nuestro primer circuito. La página de Wikipedia sobre símbolos electrónicos es un buen lugar para obtener una visión general de muchos de los símbolos utilizados en los esquemas: en.wikipedia.org/?title=Electronic_symbol.

Tabla 3.1:**Componentes con sus símbolos esquemáticos**

ComponentE	DescripCIÓn	SÍMBOLO ESQUEMÁTICO
	Batería	

		
	led (diodo emisor de luz)	
	Resistencia	

También hay otras maneras de dibujar los símbolos de una fuente de alimentación, como se puede ver en la figura 3.11. Vamos a tratar los conceptos de alimentación y tierra más adelante en este capítulo, pero reconocer los símbolos te ayudará a comprender lo que ocurre en el circuito.

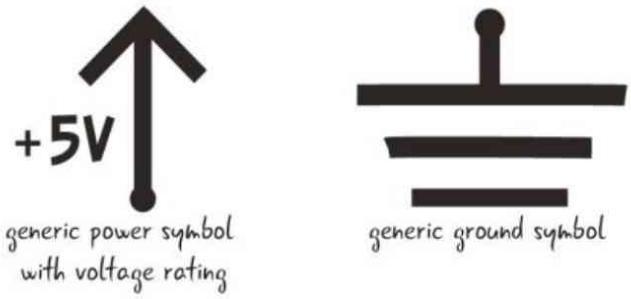


FIGURA 3.11 Símbolos esquemáticos para la alimentación y tierra.

CÓMO DIBUJAR UN ESQUEMA

Has visto un ejemplo de esquema, así como los símbolos que se utilizan en este para nuestro primer circuito. ¿Cómo se conectan los símbolos para dibujar un esquema?

Empezaremos con el símbolo de la figura 3.12, que representa una resistencia. Recuerda que la resistencia no tiene una orientación positivo-negativo, así que no hay que distinguir entre un terminal y otro.

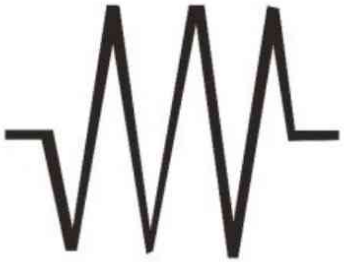


FIGURA 3.12 Símbolo esquemático de una resistencia.

A continuación dibujaremos el símbolo del led y lo conectaremos a la resistencia con una línea continua. ¿Por qué una línea continua? Recuerda que estamos representando la conexión física entre los componentes del circuito, al igual que las pistas plateadas conductoras de la PCI.

El terminal positivo, o ánodo, se conecta a la resistencia, y eso es lo que haremos cuando montemos el circuito, como se ve en la figura 3.13. Cuando conectemos la batería, la energía fluirá a través de la resistencia hasta el

extremo positivo del led.

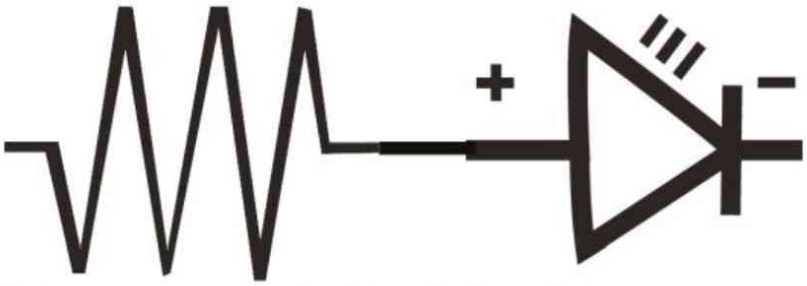


FIGURA 3.13 Resistencia conectada al ánodo de un led.

Ahora añadimos el símbolo de la resistencia y lo conectamos a los símbolos de led y de resistencia, como se muestra en la figura 3.14. El terminal negativo del led, o cátodo, se conecta al terminal negativo de la batería.

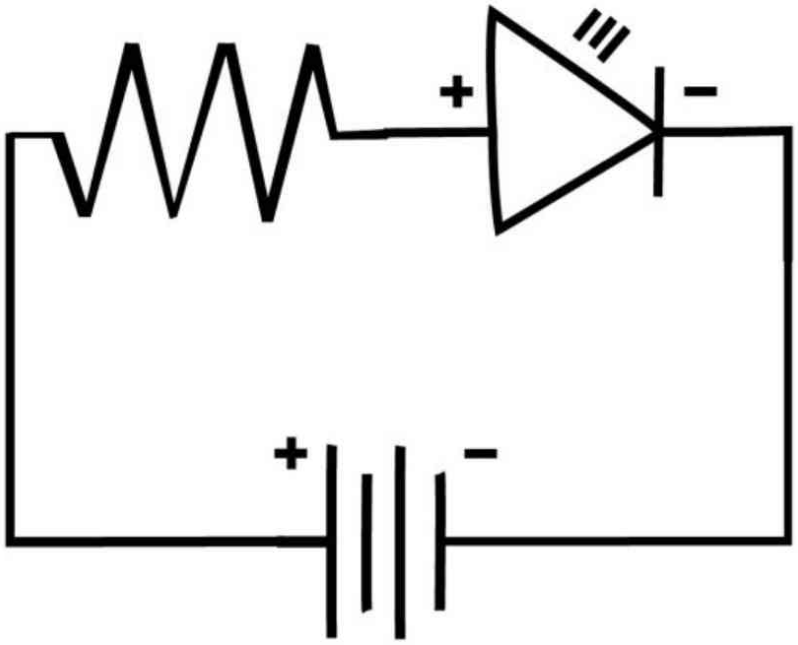


FIGURA 3.14 Esquema del circuito.

Podemos ver en este diagrama esquemático que un extremo de la resistencia está conectado a la alimentación, o signo más de la batería. El otro extremo de la resistencia está unido al extremo positivo del led. El extremo negativo del led está conectado a tierra, o signo menos. Nuestro esquema representa el cierre del trazado completo de nuestro circuito.

LA PLACA DE PRUEBAS

¿Cómo reunimos los componentes para montar un circuito? Si echas un vistazo a la figura 3.15, puedes ver que hay una placa de pruebas debajo de todos los componentes.

¿Por qué usamos una placa de pruebas? La placa de pruebas nos permite conectar todos los componentes. No podemos sujetar todas las piezas con los dedos, y tampoco queremos inicialmente unirlas permanentemente entre sí. Sabemos que un circuito es un trazado cerrado y que los componentes deben estar conectados. La placa de pruebas nos permite conectar nuestros componentes entre sí rápidamente y nos da la flexibilidad para modificar fácilmente nuestros circuitos. Usar una placa de pruebas nos permite crear rápidamente prototipos de nuestros proyectos.

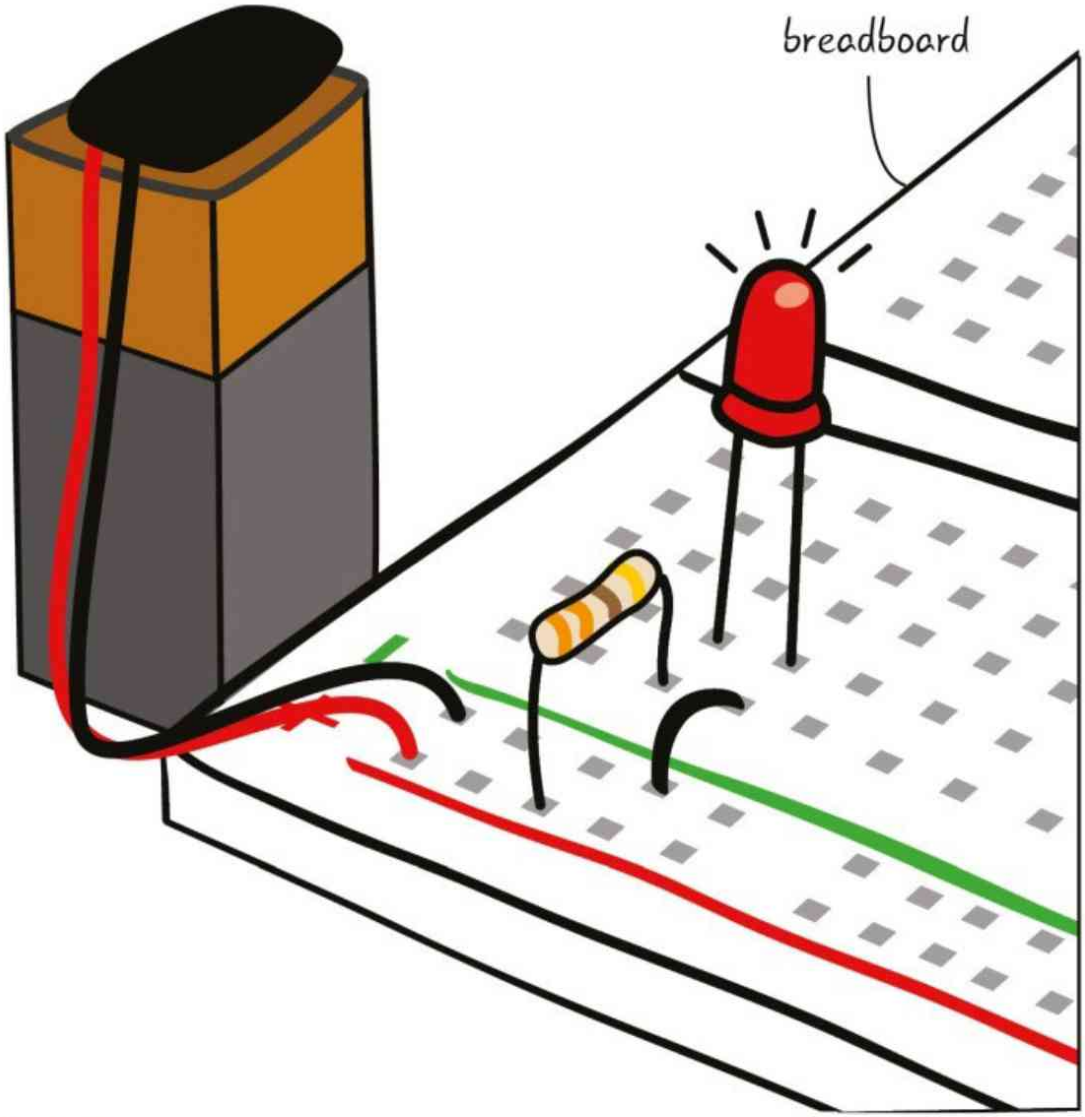


FIGURA 3.15 El circuito que montaremos, con la placa de pruebas.

> Note Utilizar una placa de pruebas nos permite conectar unos componentes con otros rápidamente y hacer modificaciones en el circuito.

FUNDAMENTOS DE LA PLACA DE PRUEBAS

Has visto dibujos de una placa de pruebas y circuitos montados en ella. Sabes también que la utilización de una placa de pruebas te permite montar rápidamente prototipos de circuitos y hacer pruebas con ellos. ¿Cómo está hecha

una placa de pruebas? Veamos la radiografía de la misma.

> **Warning** No quites la parte de abajo, la placa de pruebas se podría romper.

La placa de pruebas está formada por tiras de metal envueltas en plástico con una rejilla de agujeros en la parte superior. Los agujeros llamados puntos de unión se colocan a intervalos regulares y se disponen en filas y columnas.

En la figura 3.16 puedes ver las tiras metálicas dispuestas en filas y columnas de puntos de unión. Los puntos de unión conectados a una de las tiras metálicas están conectados entre sí.

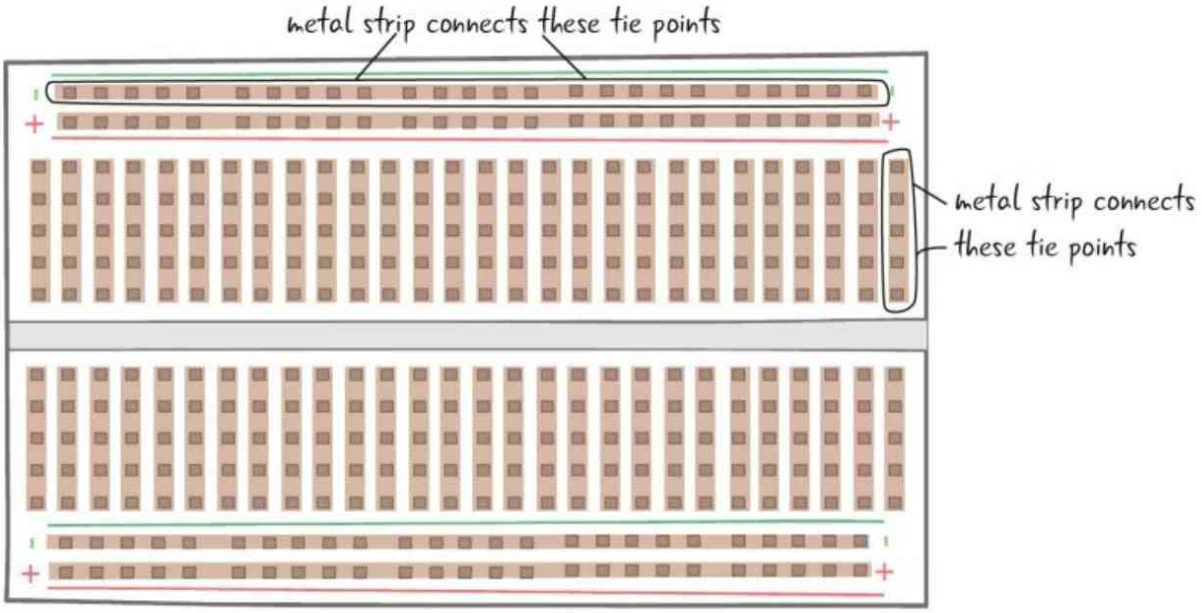


FIGURA 3.16 Vista por "rayos X" de una placa de pruebas.

Las filas y columnas están dispuestas siguiendo unas pautas para facilitar el montaje de circuitos con componentes electrónicos estándar.

Las columnas largas a la izquierda y derecha de la placa, que se muestran en la figura 3.17, se ha convenido que estén unidas a la *alimentación* y a *tierra*, y se las llama *buses de alimentación y tierra*. Hay un signo más (+) y menos (–) en la parte superior de cada columna. Se conectarán a los signos

más y menos de la batería. A menudo hay una línea roja cerca del bus de alimentación, y una verde, azul, o negra próxima al bus de tierra. Algunas placas de pruebas, en concreto las pequeñas, no tienen estos buses de alimentación y de tierra.

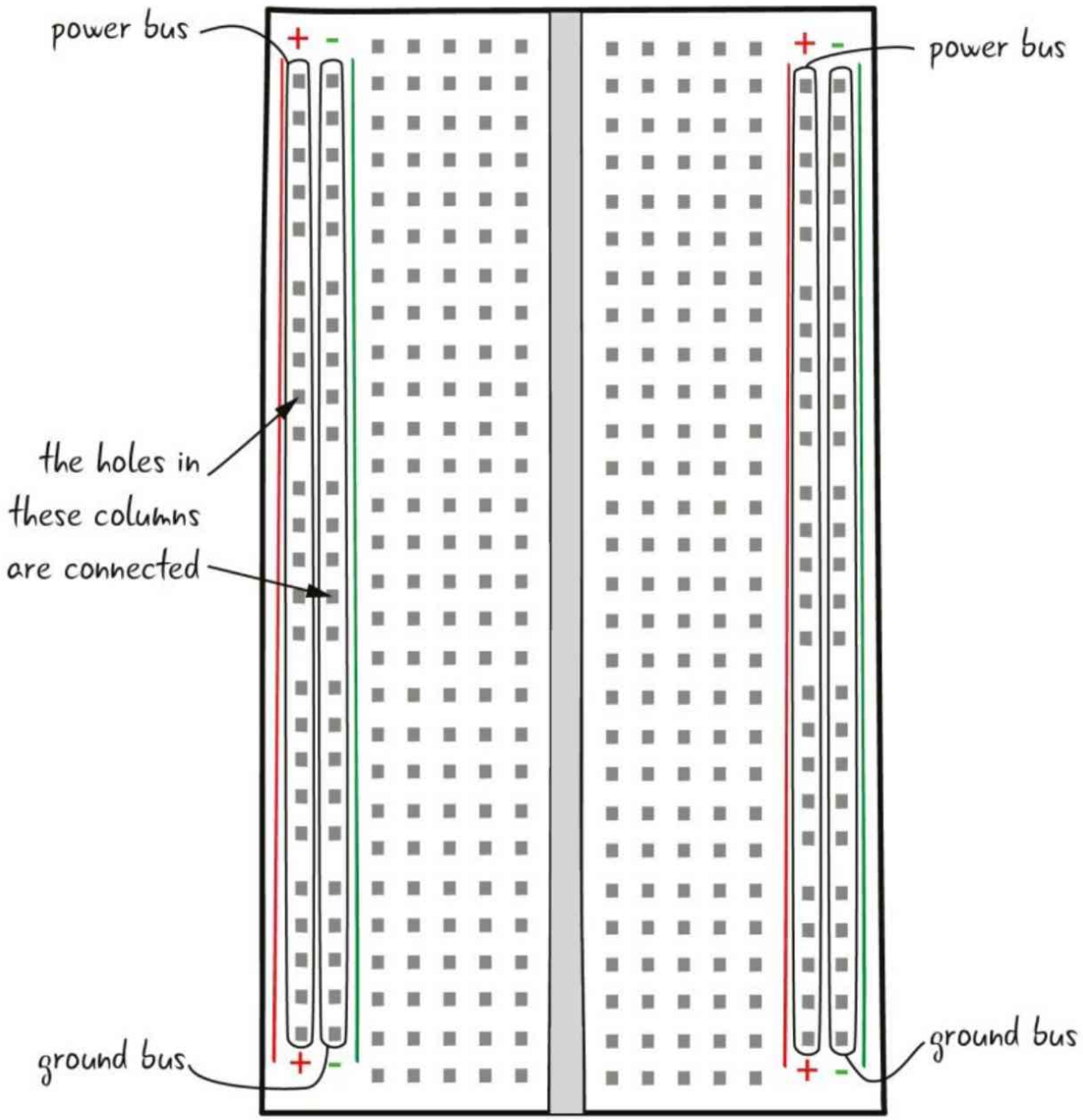


FIGURA 3.17 Buses de alimentación y tierra en una placa de pruebas.

Luego explicaremos más a fondo los términos de alimentación y tierra. Por ahora, solo necesitas saber que conectarás una batería a los buses de un lado de la placa y que los buses de los lados derecho e izquierdo no están

conectados. A la derecha o a la izquierda, no importa a qué lado de la placa de pruebas conectes la alimentación y la tierra, aunque nosotros conectaremos la batería al lado izquierdo de la placa. Es una buena idea ser coherente con la forma en la que configuras la placa de pruebas.

Cómo hacer las conexiones

Generalmente existe un hueco, llamado trinchera, en el centro, con la misma anchura que la de algunos componentes, lo que hace más fácil conectarlos al circuito. Los puntos de unión en cada fila a ambos lados de la trinchera están conectados, lo que te permite hacer conexiones entre componentes cuando los colocas en la placa. En la figura 3.18 no se conectan a través de la trinchera; cada fila de puntos de unión a ambos lados de la trinchera es una fila separada.

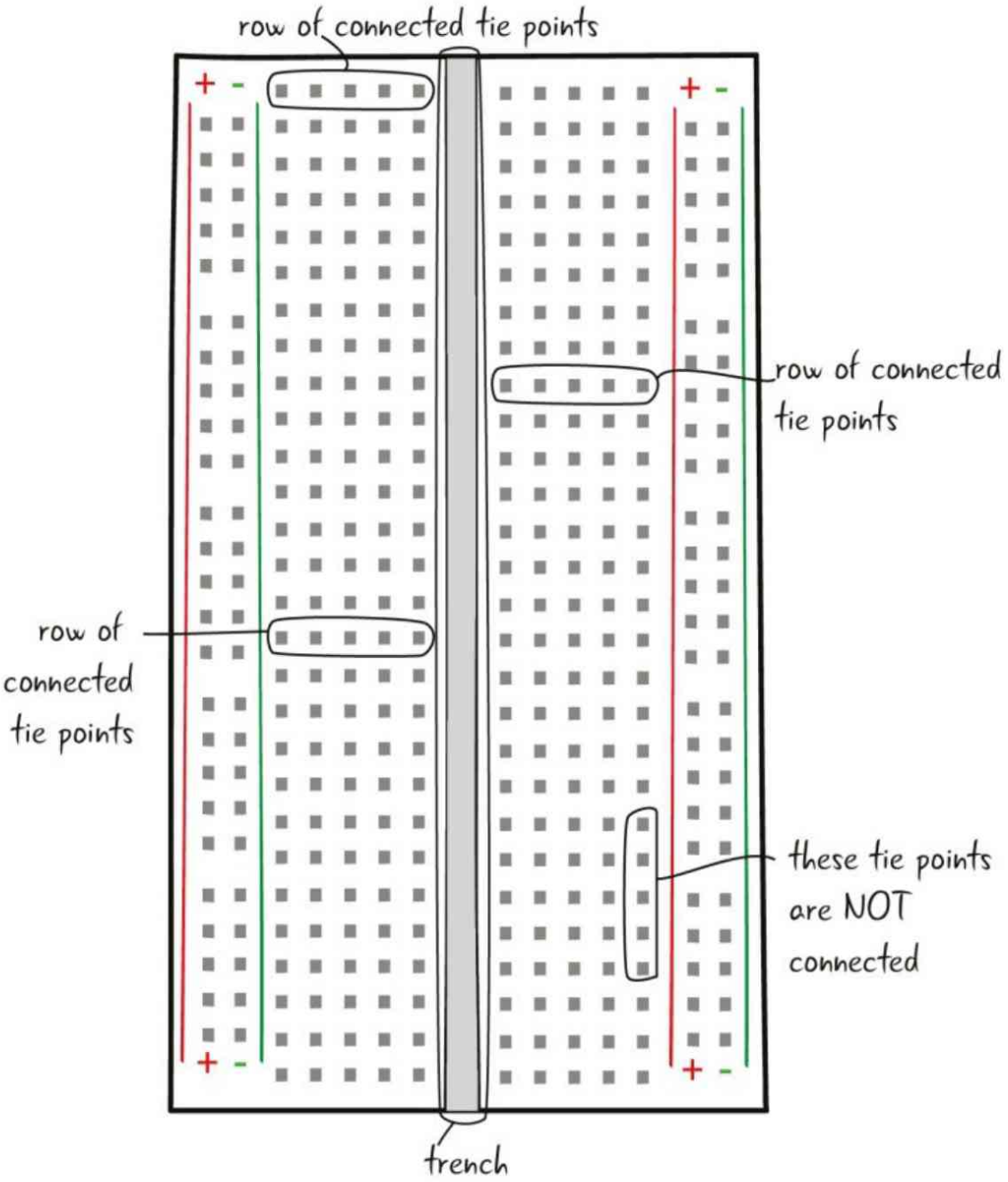


FIGURA 3.18 Fila de puntos de unión en la placa de pruebas.

> **Note** Las filas de la placa de pruebas no se conectan a través de la zanja.

Los componentes se pueden conectar entre sí colocándolos en la misma fila de puntos de unión, como se muestra en la figura 3.19.

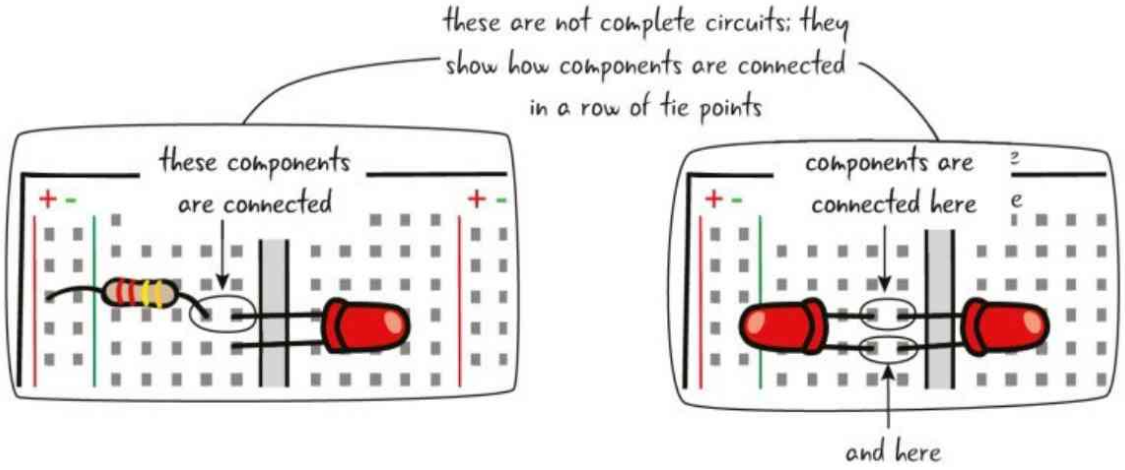


FIGURA 3.19 Componentes conectados en la placa de pruebas.

¿PREGUNTAS?

P: ¿Necesito una nueva placa de pruebas para cada circuito que tenga que montar?

R: Lo bueno de las placas de pruebas es que es muy fácil cambiar partes de un circuito o hacer uno completamente nuevo. Podrías hacer todos los circuitos de este libro reutilizando la placa de pruebas. Si quieres tener montado más de un circuito al mismo tiempo, es útil tener una placa de pruebas adicional.

Montaje del circuito

¡Vamos a montar nuestro primer circuito! Necesitarás las siguientes piezas y herramientas:

- Placa de pruebas
- Batería de 9 V
- Tapa de la batería
- 1 led
- Resistencia de 330 ohmios (bandas de color naranja, naranja, marrón, dorado)
- Cables de puente
- Alicates de punta de aguja

En la figura 3.20 puedes ver todas las piezas para empezar a montar el circuito.

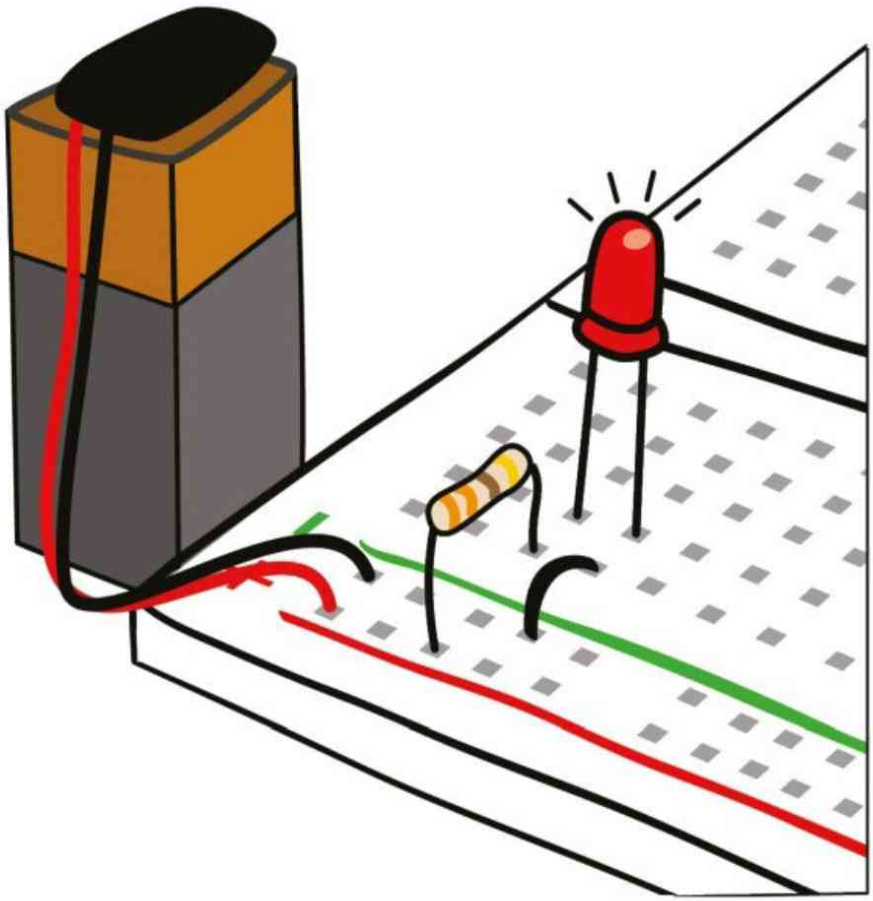


FIGURA 3.20 El circuito.

INSTRUCCIONES PASO A PASO DEL CIRCUITO

Vamos a seguir una guía de los pasos necesarios parar montar el circuito básico que has visto a lo largo del capítulo. Es posible que todavía no entiendas exactamente cómo funcionan en conjunto todos los componentes del circuito. No debes preocuparte por eso, explicaremos con más detalle cómo funciona la electricidad en un circuito y en cada componente a medida que avancemos. Por ahora, solo tienes que seguir cada uno de los pasos.

Lo primero que necesitarás será la placa de pruebas y la resistencia de 330 ohmios. Más adelante aprenderás algo más sobre resistencias, pero ahora mismo solo necesitas una resistencia que tenga cuatro bandas con los colores naranja, naranja, marrón y dorado.

Elige una esquina de la placa de pruebas, nosotros empezamos por la

esquina superior izquierda. (No importa si eliges los buses situados a la derecha o a la izquierda, pero es preferible ser coherente). Primero coloca un extremo de la resistencia de 330 ohmios (con bandas de color naranja, naranja, marrón y dorado) en el bus de alimentación (marcado con el signo +) de la placa de pruebas y el otro extremo en una fila de la placa. Tendrás que doblar los cables un poco para que entren en la placa.

Las resistencias no tienen una dirección hacia adelante o hacia atrás en un circuito, así que no importa cuál sea su orientación. Cada extremo o pata es igual a la otra. La figura 3.21 muestra cómo está conectada la resistencia.

> **Tip** Los componentes deberían colocarse en su sitio a presión. A veces es dificil introducirlos en la placa. Hay que tener paciencia. A algunos usuarios les resulta más fácil utilizar alicates de punta de aguja para insertar componentes en la placa, mientras que otros emplean las manos. Descubre que es más fácil para ti.

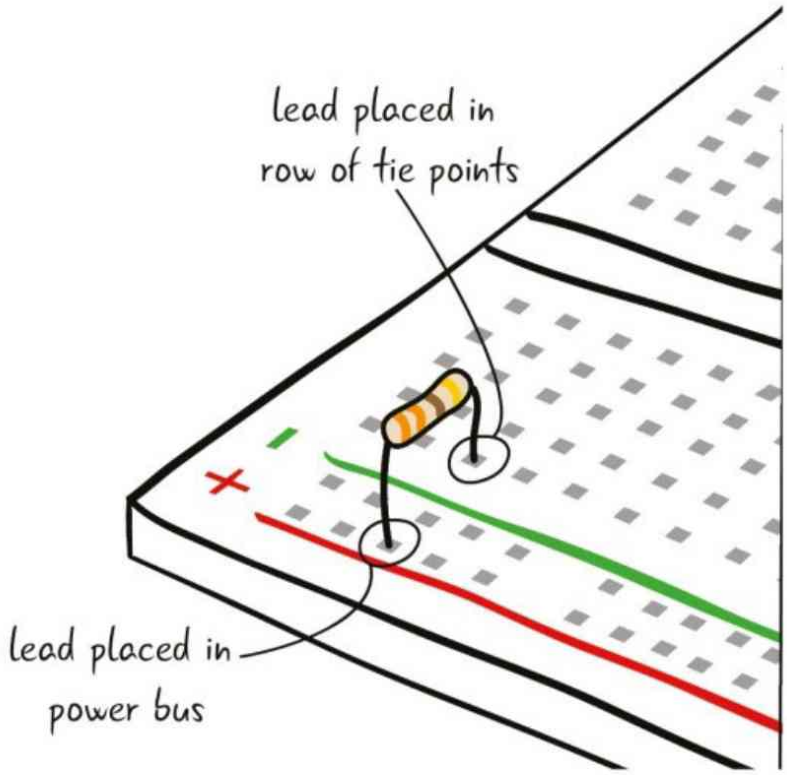


FIGURA 3.21 En primer lugar, inserta la resistencia.

Luego añade un led (figura 3.22). El ánodo (terminal largo) va en la misma fila de puntos de unión que la resistencia. El cátodo (terminal corto) va en la siguiente fila.

La figura 3.23 muestra cómo un terminal de la resistencia está en la misma fila de puntos que el ánodo del led.

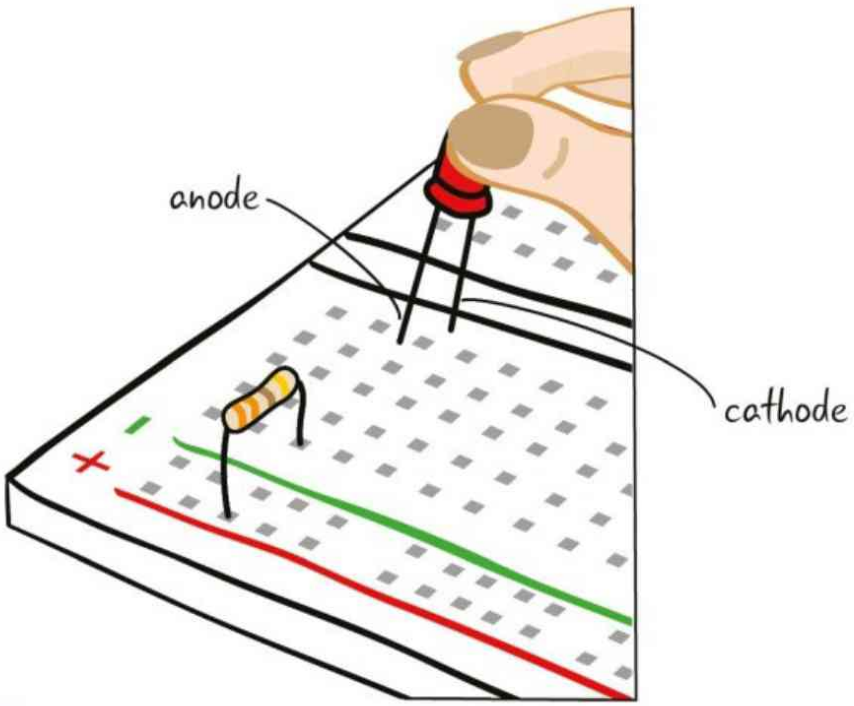


FIGURA 3.22 Inserta el led.

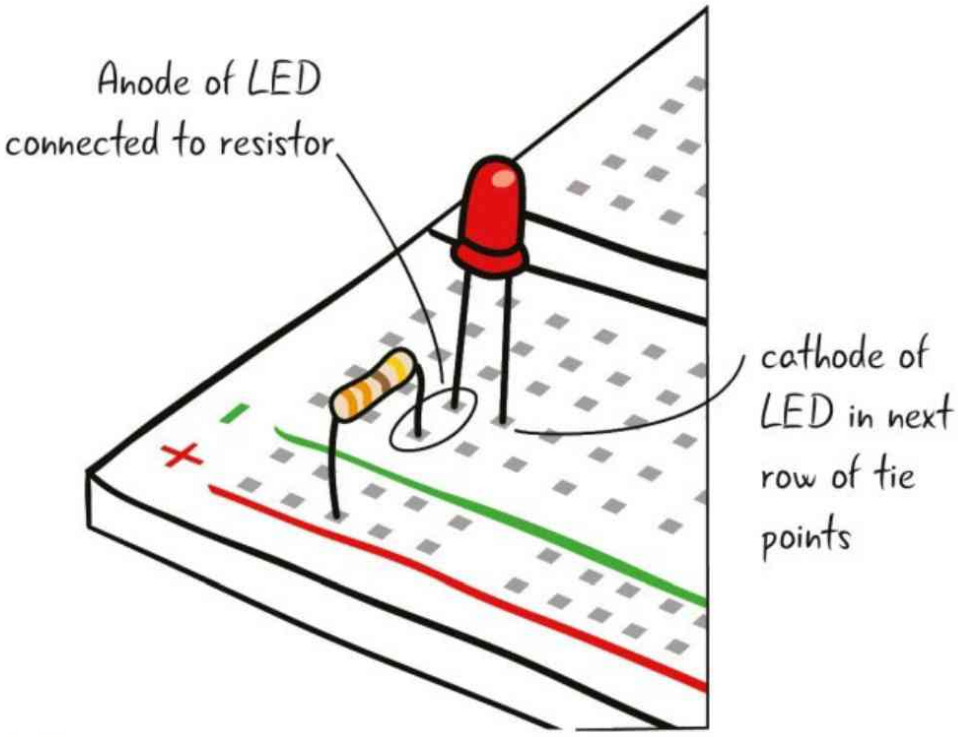


FIGURA 3.23 Led situado correctamente.

A continuación, debes colocar un puente que conecte el bus de tierra (marcado con el signo –) al cátodo del led, como se muestra en la figura 3.24. El

uso de un puente de color negro indicará que está conectado a tierra. El puente está ahí para hacer una conexión entre el cátodo y el bus de tierra.

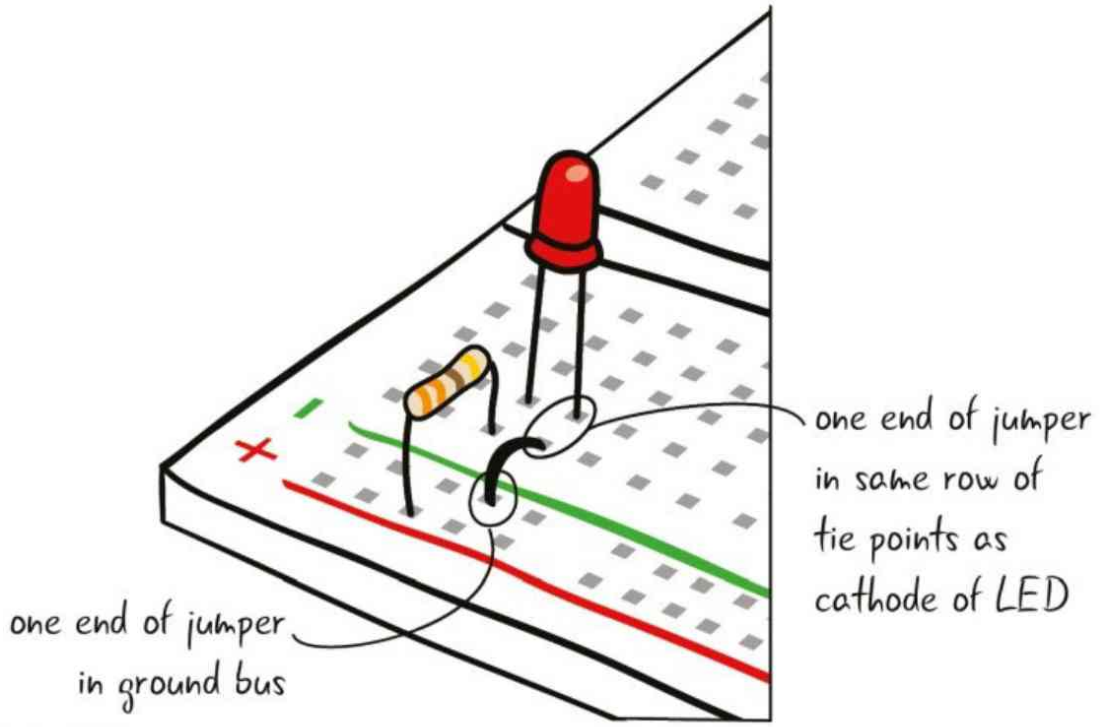


FIGURA 3.24 Instala un puente a tierra.

Coloca los terminales de la tapa de la batería en la placa de pruebas, entre el bus de tierra y el de alimentación (figura 3.25). Tiene extremos metálicos que se insertarán en los buses de alimentación y tierra.

> Tip **Asegúrate de insertarlos bien en la placa. Puede resultar difícil; a veces puede ser más fácil doblar el cable al final de la tapa de la batería.**

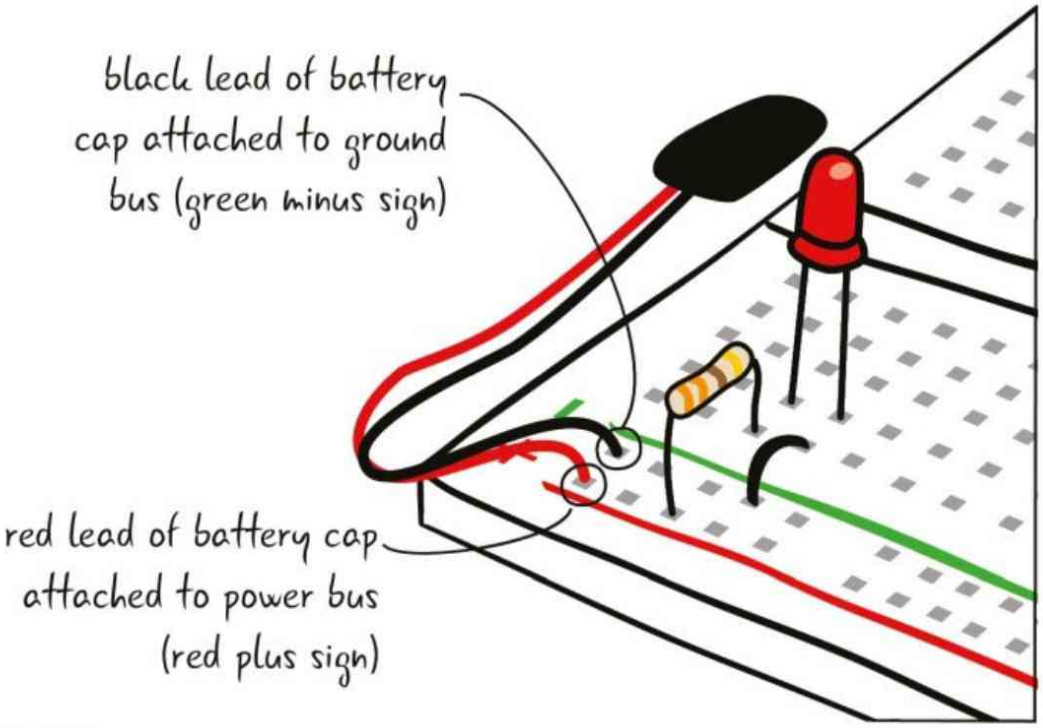


FIGURA 3.25 Inserta la tapa de la batería en la placa de pruebas.

UNA OJEADA A LA BATERÍA

Veamos con más detalle la batería de 9 V y su tapa. La parte superior de la batería tiene dos terminales a los que se conectan a presión los conectores de la tapa de la batería, como se muestra en la figura 3.26. El más pequeño, con el signo más (+), es el terminal de alimentación. El terminal más grande, con el signo menos (–), es el terminal de tierra.

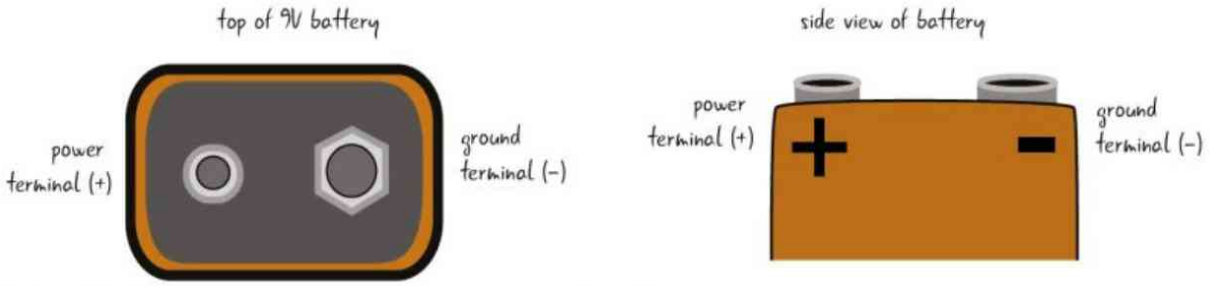


FIGURA 3.26 Detalles de la batería de 9 voltios.

Dale la vuelta a la tapa de la batería y observa los dos conectores a presión. El conector pequeño se conectará al terminal de tierra y el conector grande se conectará al terminal de alimentación, como se muestra en la figura 3.27.

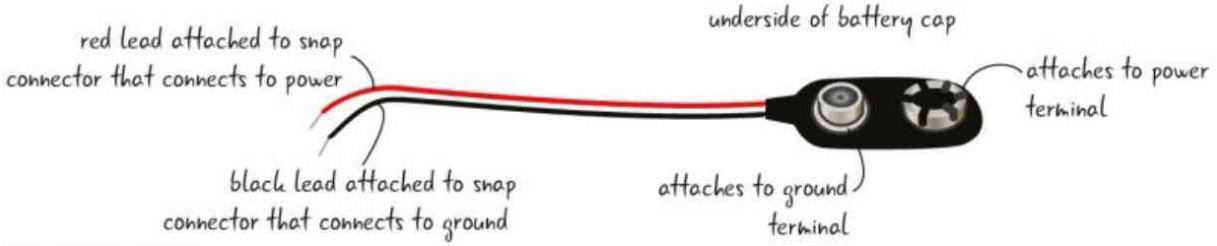


FIGURA 3.27 La tapa de la batería.

Los conectores a presión solo se conectarán adecuadamente si la batería está correctamente orientada, como se muestra en la figura 3.28. Tu tapa o soporte de la batería pueden tener una apariencia distinta, pero seguirá las mismas reglas.

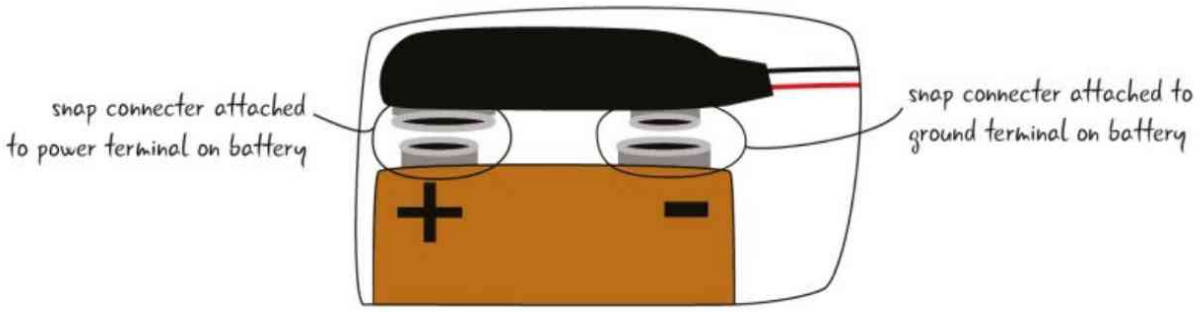


FIGURA 3.28 Fijación de la tapa a la batería.

¡Que se haga la luz!

Ahora conecta la tapa a la batería. El led debería lucir (figura 3.29). ¡Has montado tu primer circuito!

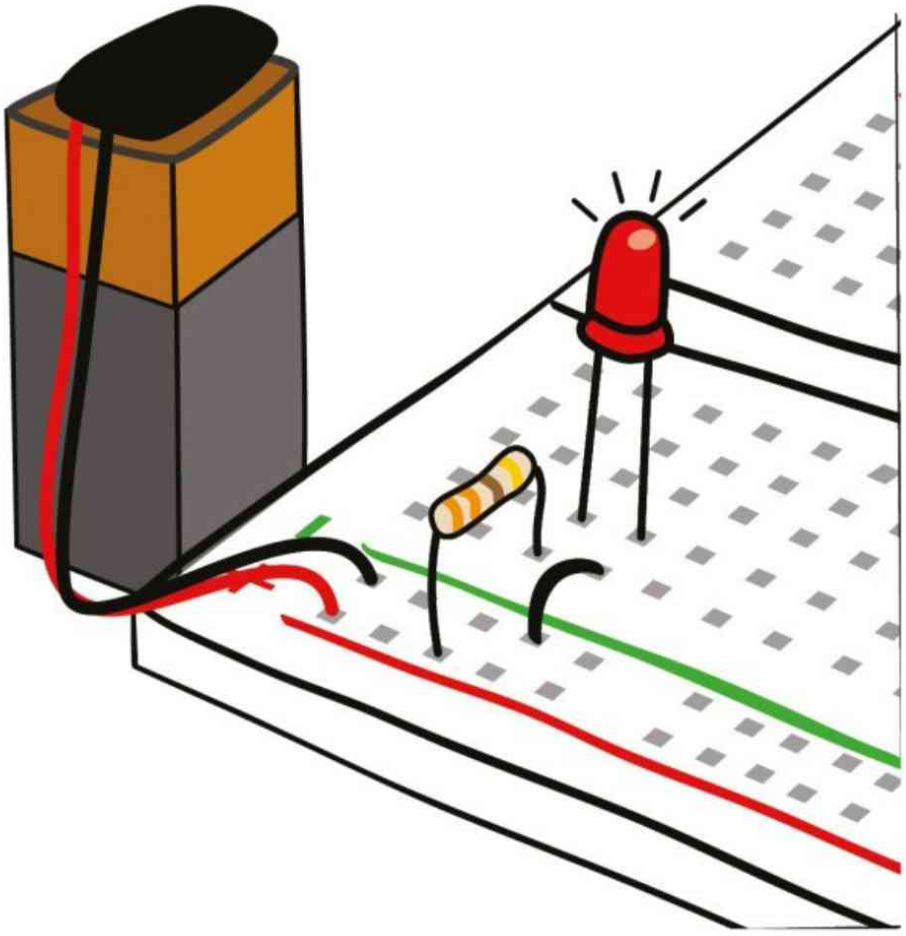


FIGURA 3.29 ¡¡El led se ilumina!!

Este es solo el primer led de los muchos que aparecen en el libro, pero

¡enhorabuena, ya has encendido el primero! Ahora, veamos cómo la batería suministra energía al circuito.

¿PREGUNTAS?

P: ¿Y si no tengo la resistencia que necesito?

R: Al principio es recomendable comprar una amplia gama de resistencias para estar seguro de que tienes las resistencias que necesitarás para los primeros proyectos y capítulos del libro. Aunque hay diferentes formas de combinar resistencias para cambiar su valor, no las tratamos con detalle en este libro. Para empezar es mejor tener una gran variedad de ellas.

ALIMENTACIÓN DEL CIRCUITO: ELECTRICIDAD

El término alimentación tiene un significado específico cuando hablamos de electricidad, que explicaremos más adelante. Por el momento, aquí la alimentación se refiere al hecho de que la electricidad proviene de la batería, pasa a través de la resistencia al led y hace que este se ilumine. Veamos más de cerca cómo se indica esto tanto en la batería como con el color de los cables en el circuito. Antes vimos brevemente los símbolos más y menos de la batería cuando colocamos la tapa; ahora veremos los símbolos con más detalle.

Unas palabras sobre los símbolos de alimentación

Como puedes ver en la figura 3.30, hay un lado + (positivo) y un lado – (negativo) en una batería, los símbolos convencionales usados para indicar qué lado de la batería produce energía (positivo) y qué lado es el lado de tierra (negativo). (Además, has visto los signos más y menos en los buses de la placa de pruebas). También viste que el lado positivo de la batería está unido al cable rojo y el lado negativo, al cable negro de la tapa de la batería.

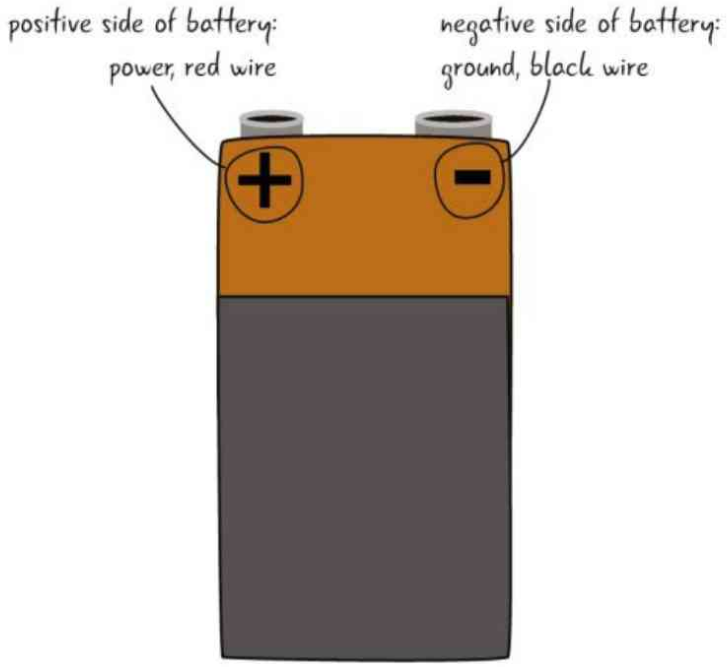


FIGURA 3.30 Lados positivo y negativo de la batería.

Alimentación

El signo +, o positivo, marca el lado de alimentación de la batería. La convención es que la energía fluye desde este lado de la batería, y todos los trazados en el circuito deben llegar al lado de alimentación. Las normas también establecen que todos los cables conectados al lado positivo son de color rojo. De esta manera, cualquier persona que necesite mirar o reparar un circuito puede saber inmediatamente por dónde entra la energía en el circuito.

Tierra

El signo – es el lado negativo de la batería, también conocido como tierra. Al igual que todos los trayectos del circuito deben comenzar por el lado de la alimentación, todos ellos deben terminar en tierra si los sigues a lo largo de toda su longitud. Se puede pensar en la tierra como el lado "cero", el lugar donde se ha agotado toda la energía. Todos los cables que nos llevan a la parte de tierra del circuito deben ser de color negro; esto te facilitará el trabajo en los circuitos y podrás saber de un vistazo qué partes están conectadas a tierra.

Hemos examinado brevemente la alimentación y tierra, y has montado tu

circuito. ¿Pero qué ocurre si el led no se ilumina? ¿Qué pasos debes seguir para detectar el problema y arreglar el circuito?

¿PREGUNTAS?

P: ¿Necesito usar una batería nueva para encender el led? ¿Puedo usar una batería vieja que he encontrado en mi casa o que me han prestado?

R: Sí, puedes, pero es probable que la luz no brille tan bien como cuando se utiliza una batería nueva. Las batería se agotan con el tiempo.

DEPURACIÓN DEL CIRCUITO

¿Algo ha salido mal o el circuito no funciona bien? ¿Qué pasa si el led no se ilumina? ¿Qué podría estar mal? ¡Depuración!

A la revisión del circuito para ver qué es lo que está mal se le llama depuración. La depuración no trata solo de resolver el problema inmediato, sino de crear una lista de verificación de posibles problemas y resolverlos uno por uno. A veces la solución "obvia" es la más difícil de encontrar y, si sigues una lista de verificación, seguro que no te perderás nada.

¿Están la alimentación y tierra conectadAs a la placa de pruebas?

Tienes que asegurarte de haber conectado correctamente los cables de la tapa de la batería a los buses de alimentación y tierra de la placa, como se muestra en la figura 3.31. Recuerda: conecta el conductor rojo al bus con la línea roja al lado, con un signo más (+) en la parte superior, y el conductor negro al bus de tierra con una línea verde, azul o negra (depende de la placa de pruebas) y un signo de menos (–) en la parte superior de la placa.

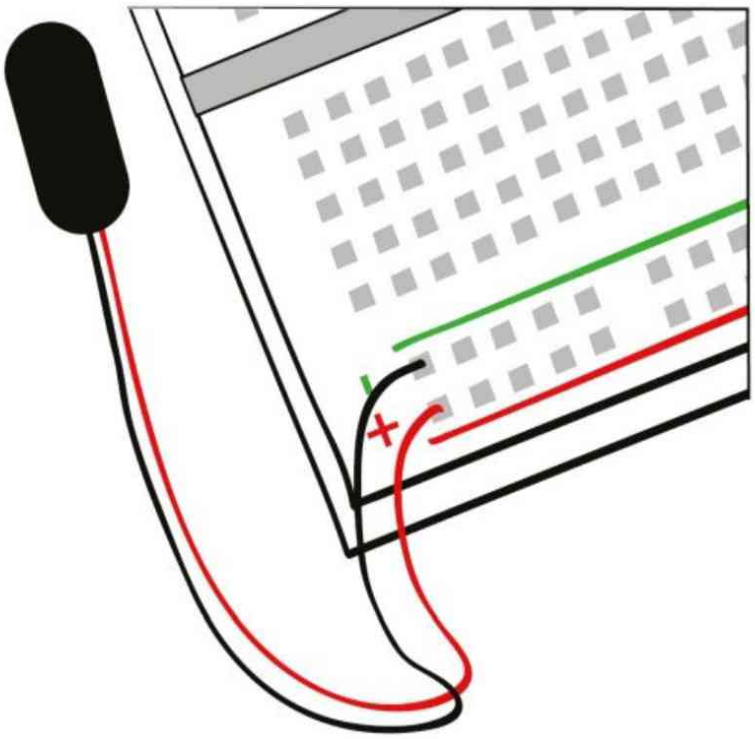


FIGURA 3.31 Cables de la tapa de la batería correctamente conectados a los buses de alimentación y tierra.

¿Está el LED correctamente orientado?

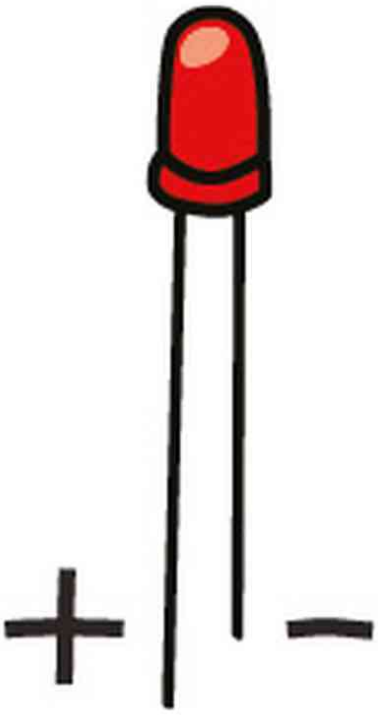


Figura 3.32:

Terminales positivo (ánodo) y negativo (cátodo) del led.

Comprueba, para estar seguro, que has colocado el led correctamente en la placa de pruebas. Recuerda que hay un conductor positivo (ánodo) y un conductor negativo (cátodo) y la corriente fluye solo si el led está correctamente orientado. El conductor positivo es más largo que el negativo, como se muestra en la figura 3.32.

¿He utilizado la resistencia adecuada?

Comprueba a continuación si has utilizado la resistencia adecuada. En capítulos posteriores discutiremos cómo seleccionar una resistencia, pero si has usado una con un valor demasiado alto, el circuito no tendrá suficiente potencia para encender el led. Si utilizas una con un valor demasiado bajo, puedes destruirlo. Para este circuito, la resistencia debería tener las bandas de color naranja, naranja, marrón y dorada (figura 3.33).

FIGURA 3.33 Resistencia de 330 ohmios.

Estos primeros pasos de depuración se basan en la observación cuidadosa y la comprensión de los fundamentos del circuito que hemos visto hasta ahora. Algunos pasos de depuración también dependerán de herramientas que mejoran tu conocimiento sobre lo que sucede en el circuito.

Depuración de trazados del circuito: Continuidad

Tal vez el error más común en el montaje de un circuito utilizando una placa de pruebas es colocar los componentes en los puntos de unión equivocados en la placa de pruebas, de manera que no estén conectados. Como has visto, los circuitos siguen un trazado, y si los componentes no están conectados entre sí correctamente, el trazado se rompe. La *continuidad* es la propiedad

que indica que las cosas están conectadas, como se muestra en la figura 3.34.

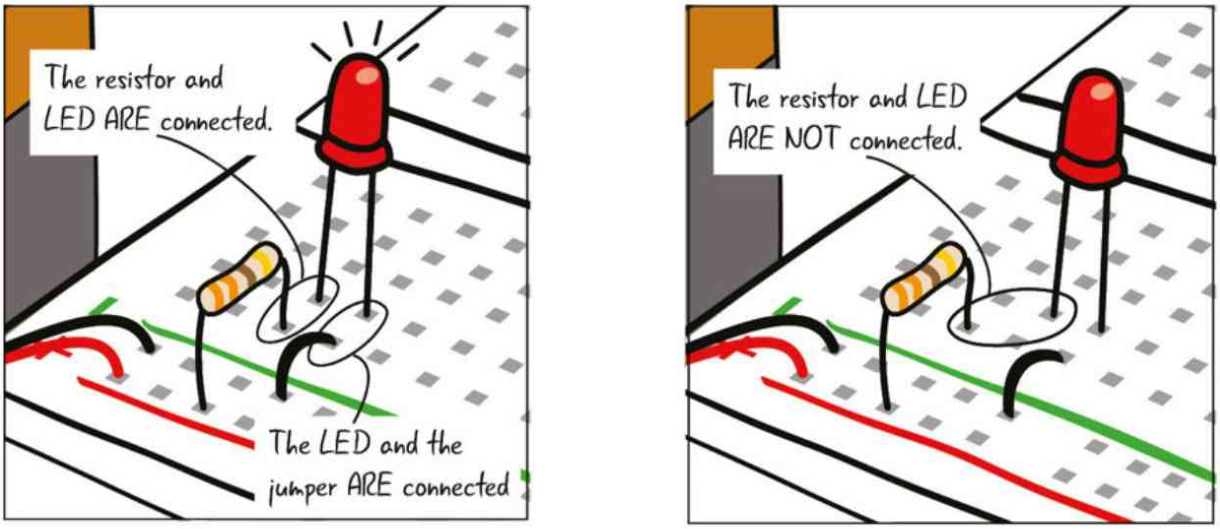


FIGURA 3.34 Componentes bien conectados y componentes que *no* están bien conectados.

Puedes comprobar que los componentes están bien conectados si analizas con detalle la placa. Comprueba cuidadosamente que los cables del led, de la resistencia y del puente están en la fila adecuada de los puntos de la placa de pruebas para que estén conectados correctamente.

Hay otra forma de comprobar la continuidad en un circuito de una placa de pruebas además de inspeccionarlo visualmente: puedes comprobar la continuidad con un multímetro (figura 3.35).

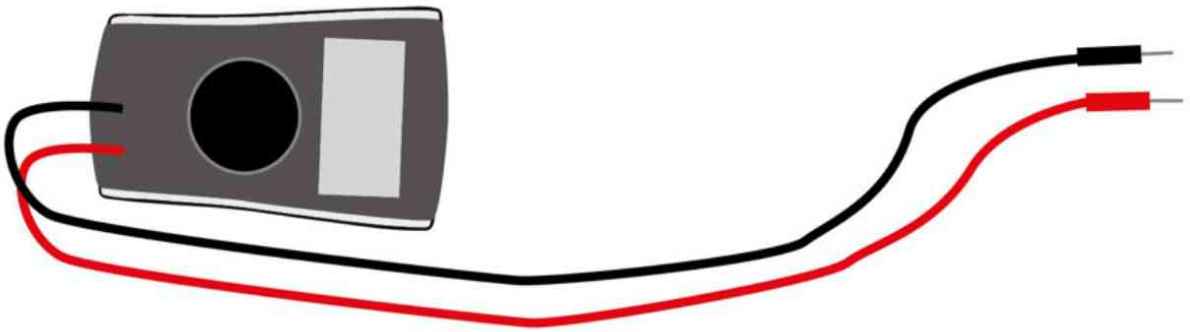


FIGURA 3.35 Multímetro.

¿PREGUNTAS?

P: ¿Tengo que memorizar los pasos de la depuración?

R: Buena pregunta. No se requiere (o no se espera) que tengas que memorizar los pasos a seguir en la depuración. Descubrirás que después de montar los circuitos del libro empezarás a recordar los pasos a seguir en la depuración, ya que los usarás con frecuencia. Haremos referencia a los pasos que hay que seguir durante el resto del libro.

el Multímetro

Otra manera de conseguir información de los circuitos es usando un multímetro. Un multímetro es una herramienta crucial para verificar que nuestros proyectos electrónicos y de Arduino se ejecutan correctamente y que todos los componentes funcionan. El multímetro será una gran herramienta que usarás en todos los proyectos de este libro para asegurarte de que todo funciona como se espera. A veces lo llamaremos multímetro y a veces, medidor. Ahora veremos cómo utilizarlo para comprobar la continuidad.

No usarás el multímetro con el Arduino en este capítulo, pero sí lo harás en los siguientes. ¿Por qué lo vemos ahora? Te ayudará a depurar el primer circuito, y tendrá un valor incalculable más tarde, cuando los proyectos se vuelvan más complejos y aprendas más sobre cómo usarlo. La figura 3.36 muestra diferentes multímetros.

Utilizamos el medidor de SparkFun (el número de parte de SparkFun es TOL-12966), que hemos mencionado en la lista de piezas del capítulo 1. Los dibujos del multímetro en este libro tienen como referencia este modelo. Es posible que tu medidor tenga un aspecto diferente, pero los principios para configurarlo y utilizarlo serán los mismos.

FIGURA 3.36 Los multímetros se presentan en diferentes tamaños y colores.

Visión general del multímetro

La Figura 3.37 muestra las partes de un multímetro: una pantalla, que muestra el valor de la variable eléctrica que estás midiendo, y un dial, que puedes girar para determinar la variable eléctrica que quieres comprobar. Con un extremo de la sonda toca un extremo del componente que estás comprobando, mientras que el otro extremo está conectado al medidor a través de uno de los puertos.

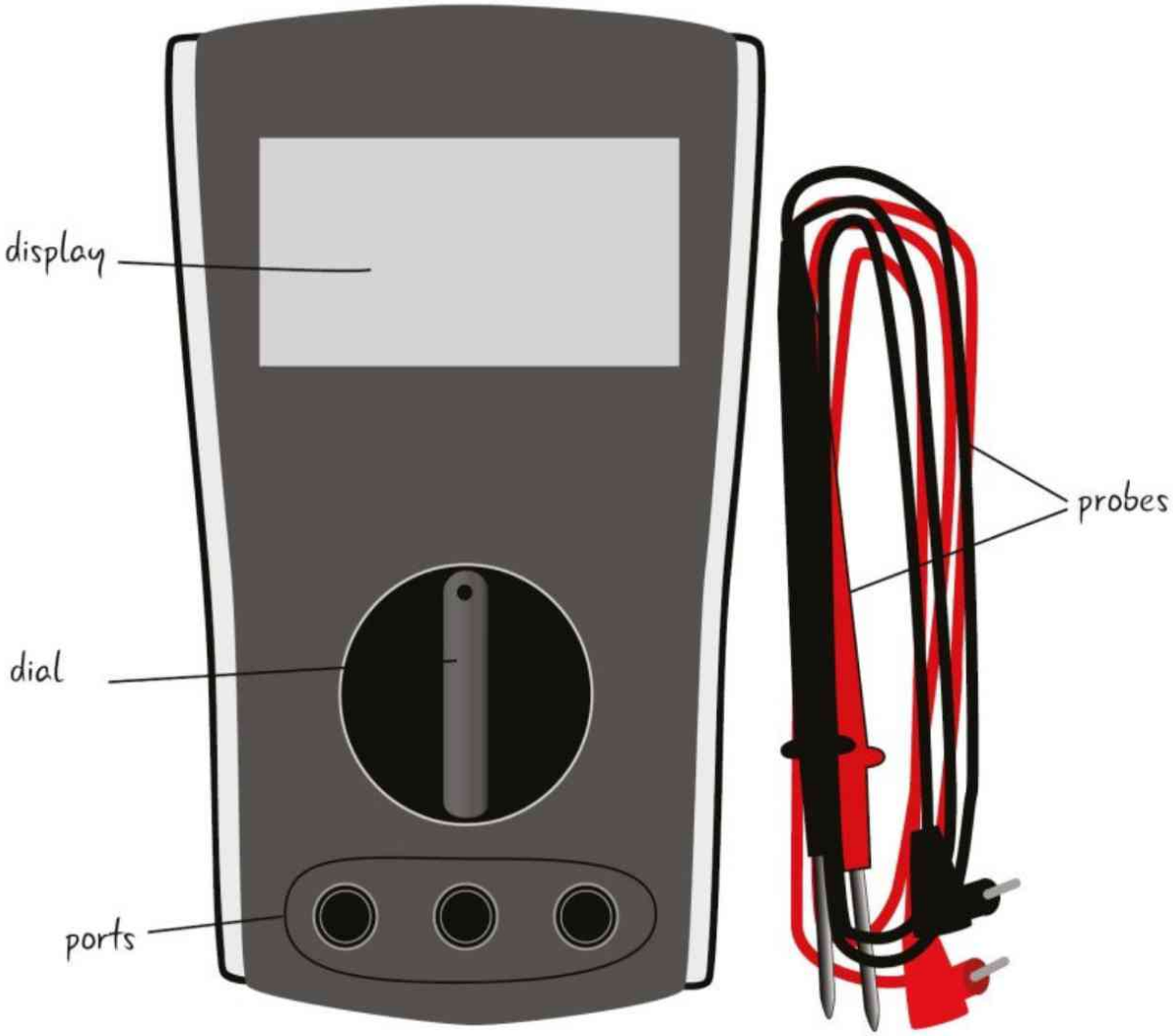


FIGURA 3.37 Partes del multímetro.

Algunos medidores tienen botones de apagado/encendido, mientras que este se enciende con el dial.

> **Warning** Recuerda apagar el medidor cuando termines para no agotar la batería.

La mayoría de los multímetros se alimentan con una batería de 9 V. No vamos a tratar aquí las instrucciones para insertar la batería en el medidor. Si compras este tipo de medidor, viene acompañado por las instrucciones. Si adquieres o te regalan un medidor diferente, las instrucciones para reemplazar la batería serán diferentes.

Partes del multímetro: el dial

La figura 3.38 muestra un detalle de la esfera de un multímetro típico en el que aparecen algunas de las magnitudes eléctricas que puede medir. Explicaremos todos estos símbolos y propiedades a medida que avancemos en el contenido del libro. Ahora solo debes saber que hay diferentes características que podemos medir: voltaje de C.A., voltaje de C.C., resistencia, amperaje de C.C. y continuidad.

FIGURA 3.38 Dial del multímetro con las características eléctricas que puede medir.

En el capítulo 5, "Electricidad y medición", volveremos a ver estas propiedades eléctricas y cómo medirlas con el multímetro.

Partes del multímetro: las sondas

La figura 3.39 muestra las sondas, la parte del multímetro con la que tocas el circuito, componente o lo que sea que estés probando o midiendo. Las puntas metálicas de las sondas se colocan de manera que toquen el circuito o

componente. El otro extremo de cada sonda se conecta a los puertos del multímetro. Cuando desembalas el multímetro, las sondas no vienen conectadas a los puertos.

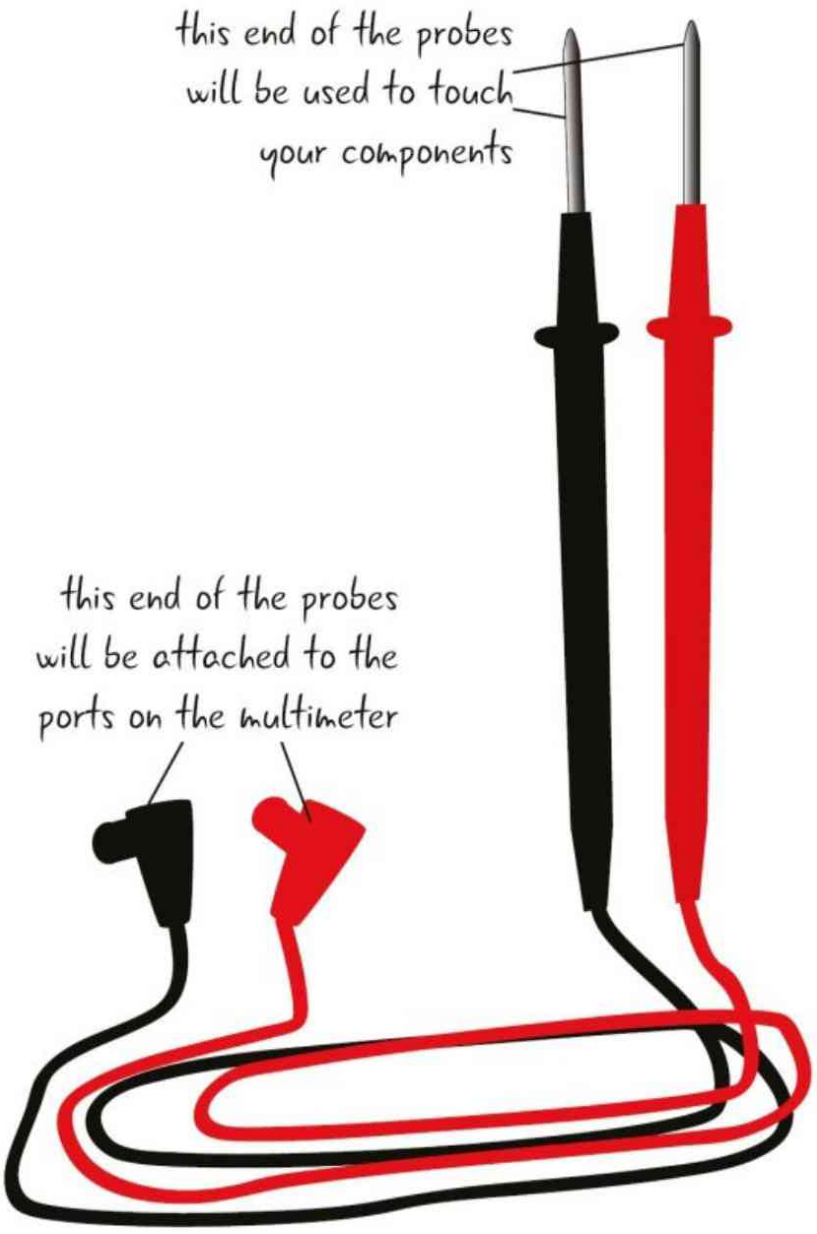


FIGURA 3.39 Sondas del multímetro.

Partes del multímetro: los puertos

Ahora que hemos visto las sondas del multímetro, estudiemos más de cerca los puertos del medidor, que se muestran en la figura 3.40.

Al usar el medidor, es importante que las sondas se coloquen en los

puertos correctos. Para cualquier medición que se haga, la sonda negra se coloca en el puerto COM central. La sonda roja tiene dos puertos diferentes en los que se puede colocar (como se indica en el exterior). En general, es una buena práctica tener conectada la sonda roja en el puerto del extremo derecho.

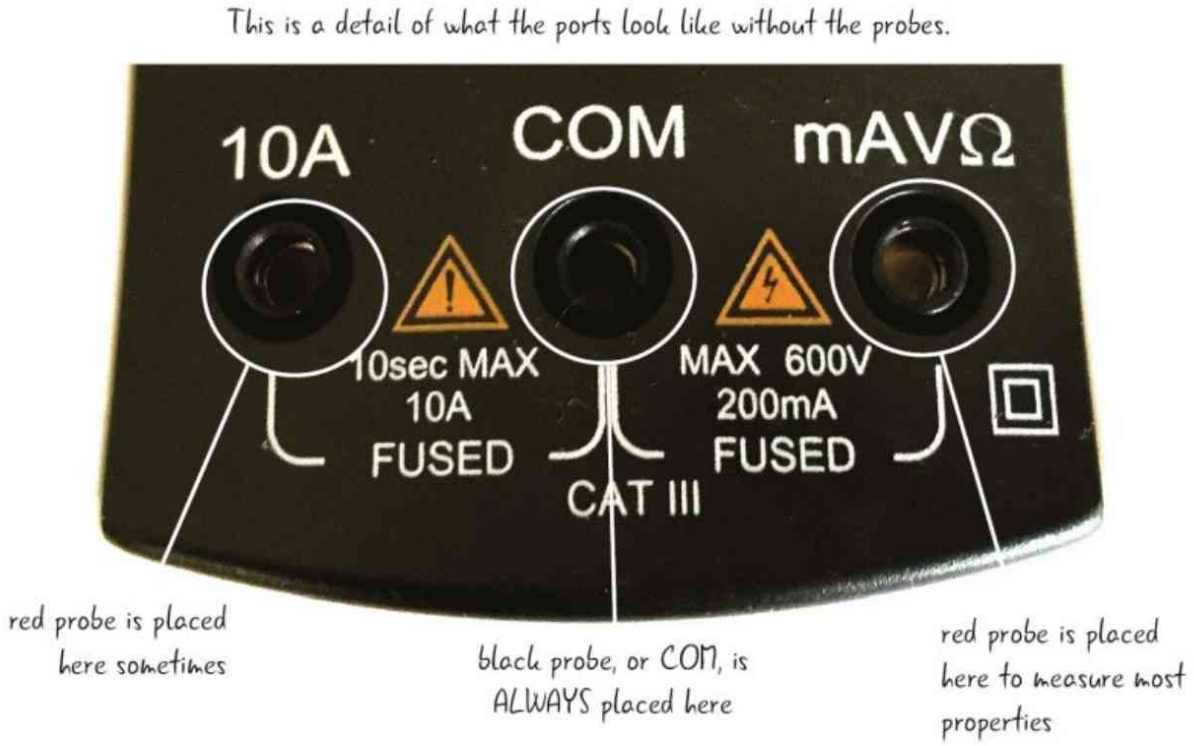


FIGURA 3.40 Puertos del multímetro.

Uso del multímetro

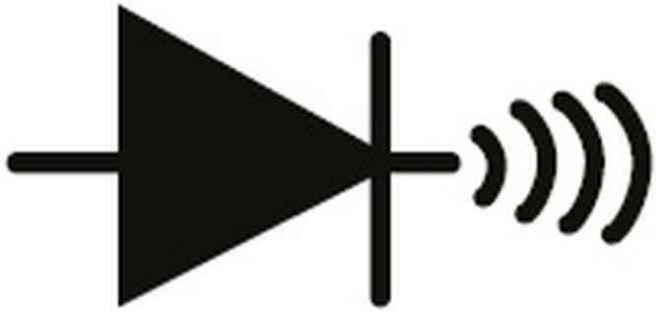


Figura 3.41:**Símbolo de continuidad.**

La continuidad (figura 3.41) es una propiedad eléctrica que se manifiesta cuando existe una conexión entre dos partes. Puedes utilizar el medidor para comprobar esta propiedad. Es un buen método para familiarizarte con las partes del medidor. ¡Y vas a utilizarlo para depurar tu circuito!

Configuración del medidor para probar la continuidad

En primer lugar te mostraremos cómo utilizar el multímetro para comprobar la conexión eléctrica entre las sondas del medidor, comprobando la "continuidad" entre las sondas (figura 3.42).

Luego pasaremos a comprobar la continuidad en tu circuito.

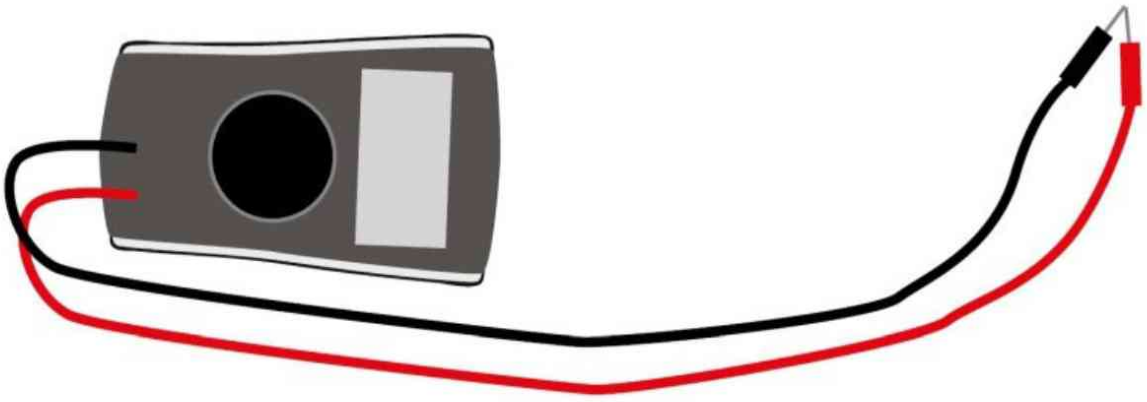


FIGURA 3.42 Multímetro con las sondas en contacto.

Esta prueba es una buena manera de tener la seguridad de que el multímetro está funcionando y de familiarizarte con su uso. Si las sondas se

tocan, forman un circuito eléctrico completo. La misma prueba se puede utilizar más tarde para comprobar si los componentes están conectados correctamente desde el punto de vista eléctrico.

Ajustes del medidor para probar la continuidad

Para comprobar la continuidad, la sonda de color negro se coloca en el puerto COM y la de color rojo se introduce en el puerto mAVΩ, como se ve en la figura 3.43.

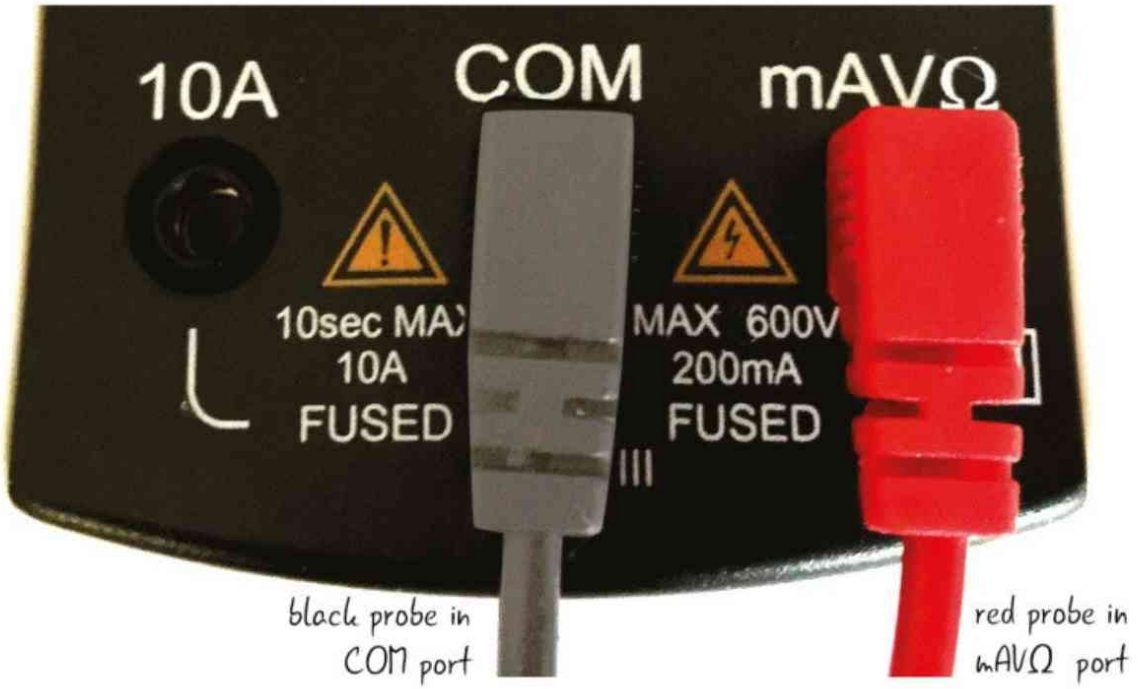


FIGURA 3.43 Configuración de los puertos del medidor para comprobar si hay continuidad.

A continuación, gira el dial de modo que el indicador apunte hacia el símbolo de continuidad (figura 3.44).

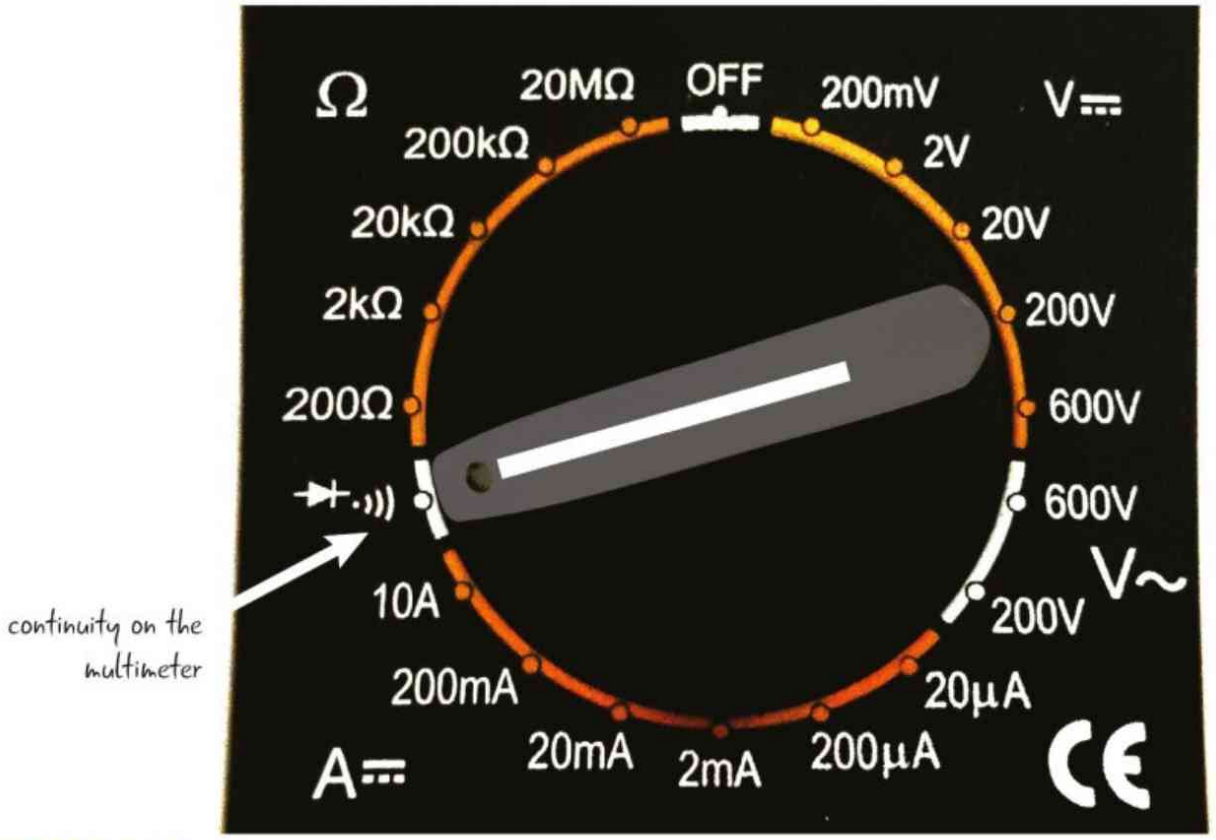


FIGURA 3.44 Gira el selector hasta el símbolo de continuidad.

Comprobación de la continuidad

Cuando las sondas tocan componentes que están conectados, el medidor emitirá un tono si está configurado para probar la continuidad. Cuando las sondas están conectadas a los puertos correspondientes, si los extremos se tocan entre sí, se establece un circuito eléctrico. Estás cerrando un circuito con las sondas, para probar así la continuidad.

Toca ahora las dos sondas entre sí para probarlo, como se muestra en la figura 3.45. Mientras las sondas están en contacto, la pantalla mostrará ".000", aunque puede fluctuar ligeramente. También oirás un tono cuyo sonido variará dependiendo del medidor. Para medir la continuidad, los números que aparecen en pantalla no son tan importantes como lo serán con las medidas de otras propiedades, de las que hablaremos más adelante en el capítulo 5.

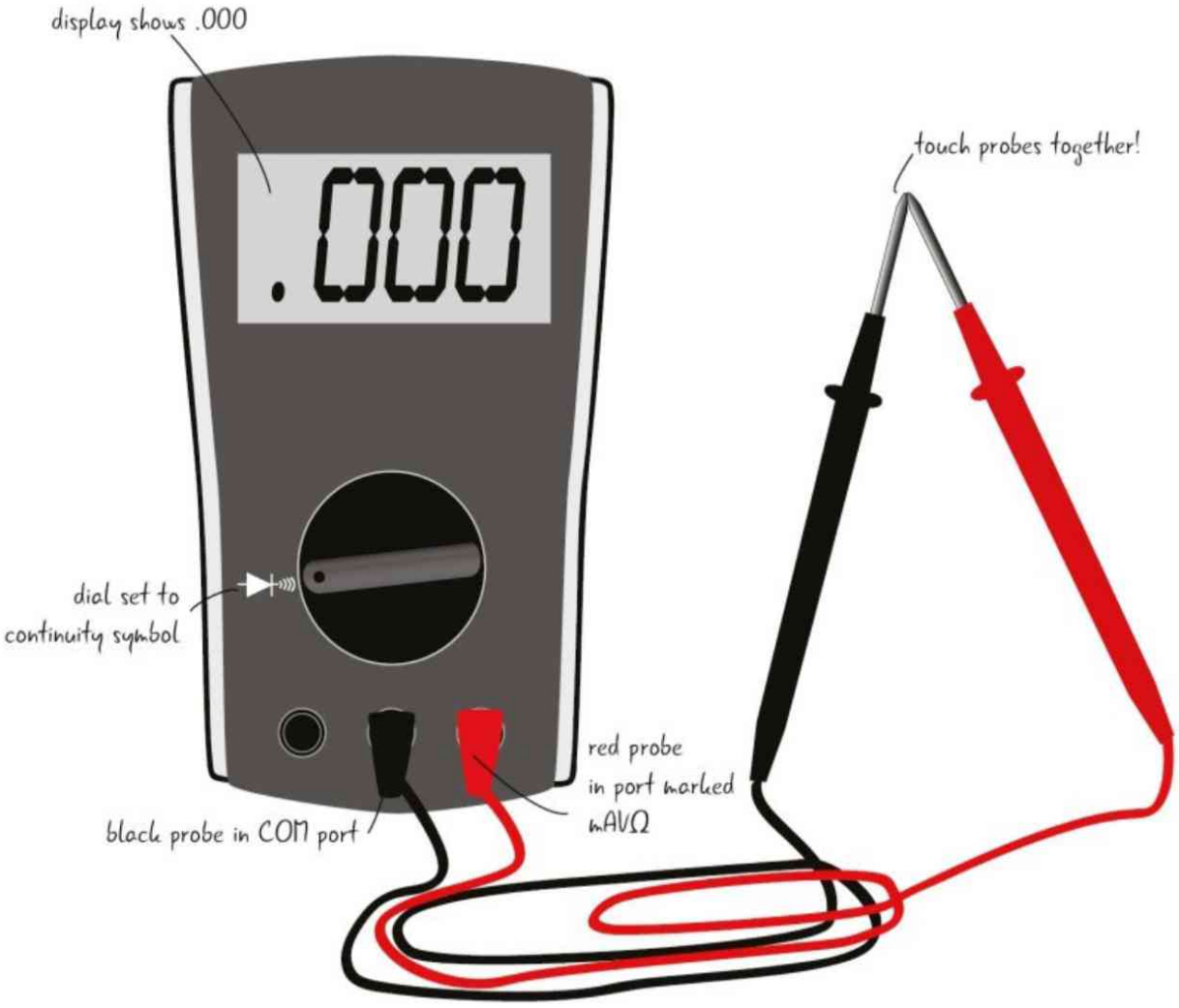


FIGURA 3.45 El multímetro con las sondas en contacto es una prueba de continuidad.

Cuando las sondas están en contacto, como se muestra en la figura 3.46, ¡deberías oír un tono!

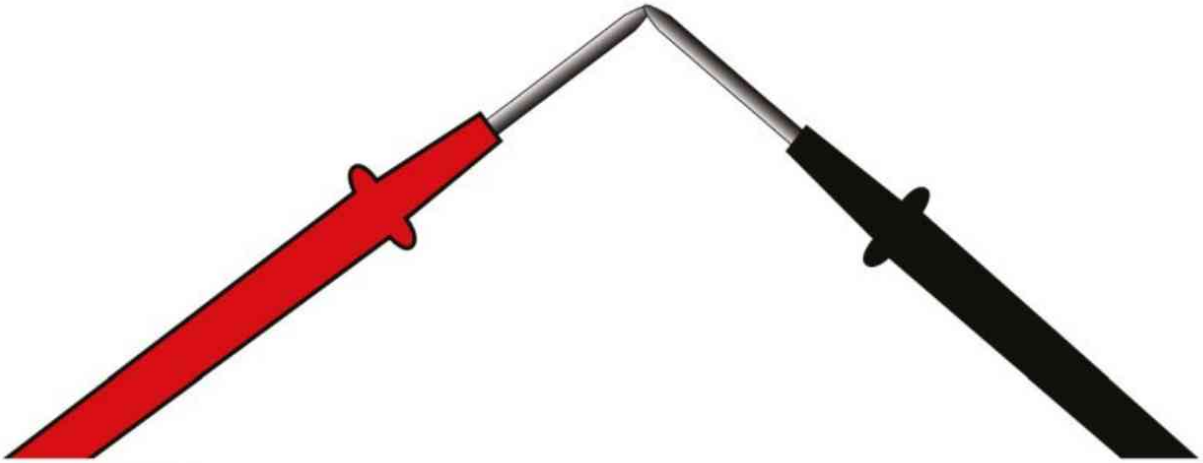


FIGURA 3.46 Las sondas están en contacto y se oye un tono.

La continuidad ayudará a resolver problemas en circuitos más complicados, indicando cuando los componentes no están conectados entre sí o si están conectados en un punto incorrecto. En el capítulo 5 explicaremos más detalladamente cómo la continuidad puede ayudarte a resolver problemas.

SEGUNDA depuraCIÓN DEL CIRCUITO

Volvamos a nuestro circuito básico. Ahora que has desembalado el multímetro y entiendes lo que es la continuidad, apliquemos las sondas del multímetro al circuito y veamos los resultados.

Prueba de continuidad del circuito

Si has completado el último ejercicio, el medidor ya está configurado correctamente para probar la continuidad. Los ajustes del selector y la configuración de las sondas se muestran en la figura 3.47. Debes comprobar que el selector está apuntando al símbolo de continuidad y las sondas están en los puertos adecuados.

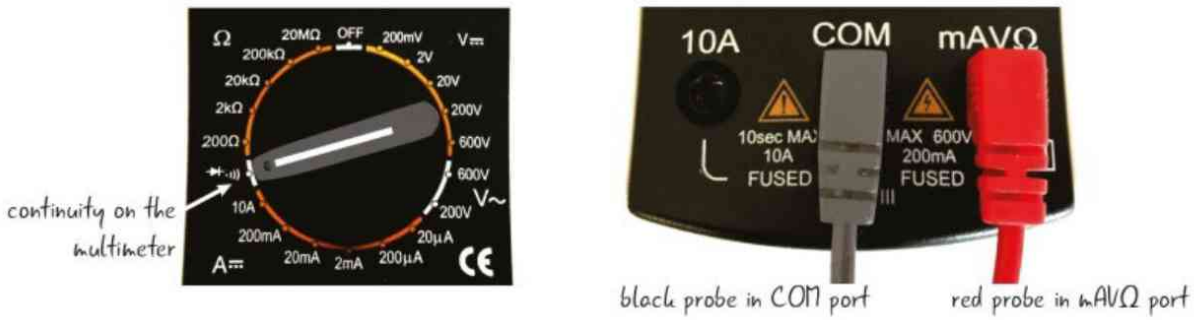


FIGURA 3.47 Ajustes del medidor para comprobar la continuidad.

En primer lugar, retira la batería del circuito. Luego enciende el medidor y toca con una sonda uno de los conductores de la resistencia y con la otra, uno de los conductores del led, como se muestra en la figura 3.48. No importa cuál sea el color de la sonda que toque cualquiera de los conductores.

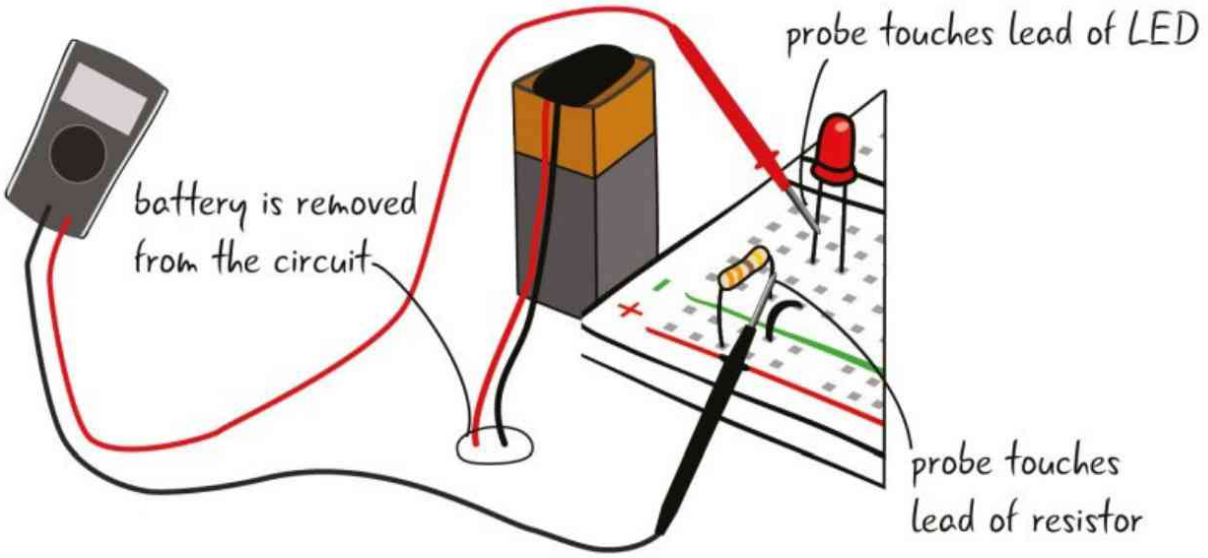


FIGURA 3.48 Comprobación de la continuidad del circuito.

Si los componentes están conectados, oirás un zumbido de nuevo y la lectura en la pantalla será .000 con una posible ligera fluctuación.

¿Y si no sonara ningún zumbido? Comprueba las conexiones en la placa de pruebas entre cada componente para ver si están colocados en los puntos de unión apropiados.

En la figura 3.49, el led no está conectado a ninguno de los otros componentes. La resistencia está conectada a la alimentación y el puente está conectado a tierra, pero ninguno de ellos está conectado al led. Para resolver el problema, coloca los cables en los puntos de anclaje correctos.

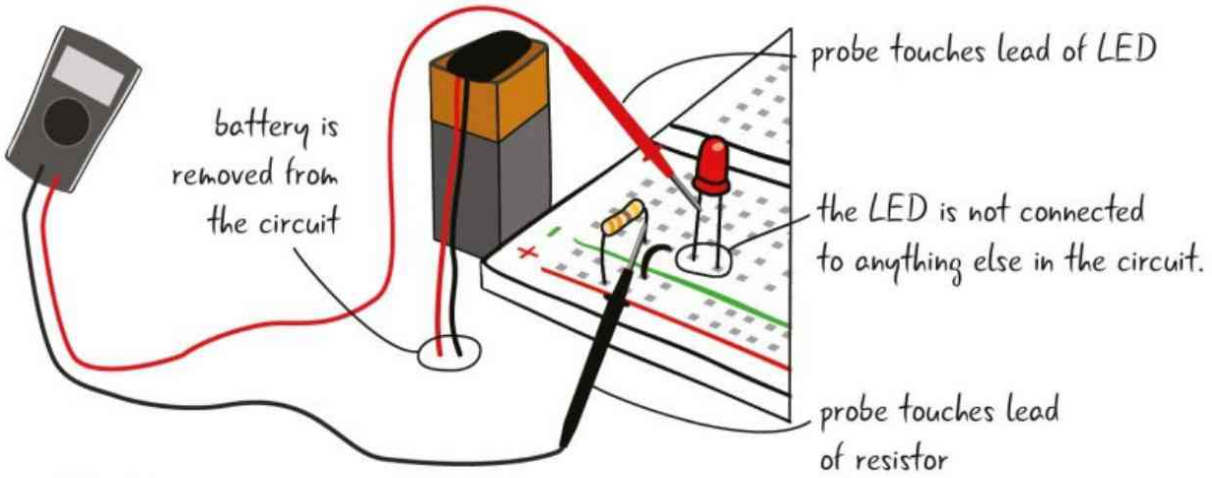


FIGURA 3.49 El multímetro prueba un circuito en el que los componentes no están conectados.

¿PREGUNTAS?

P: ¿Qué hay de todos esos otros símbolos en el multímetro, cuándo usaremos el multímetro para medirlos?

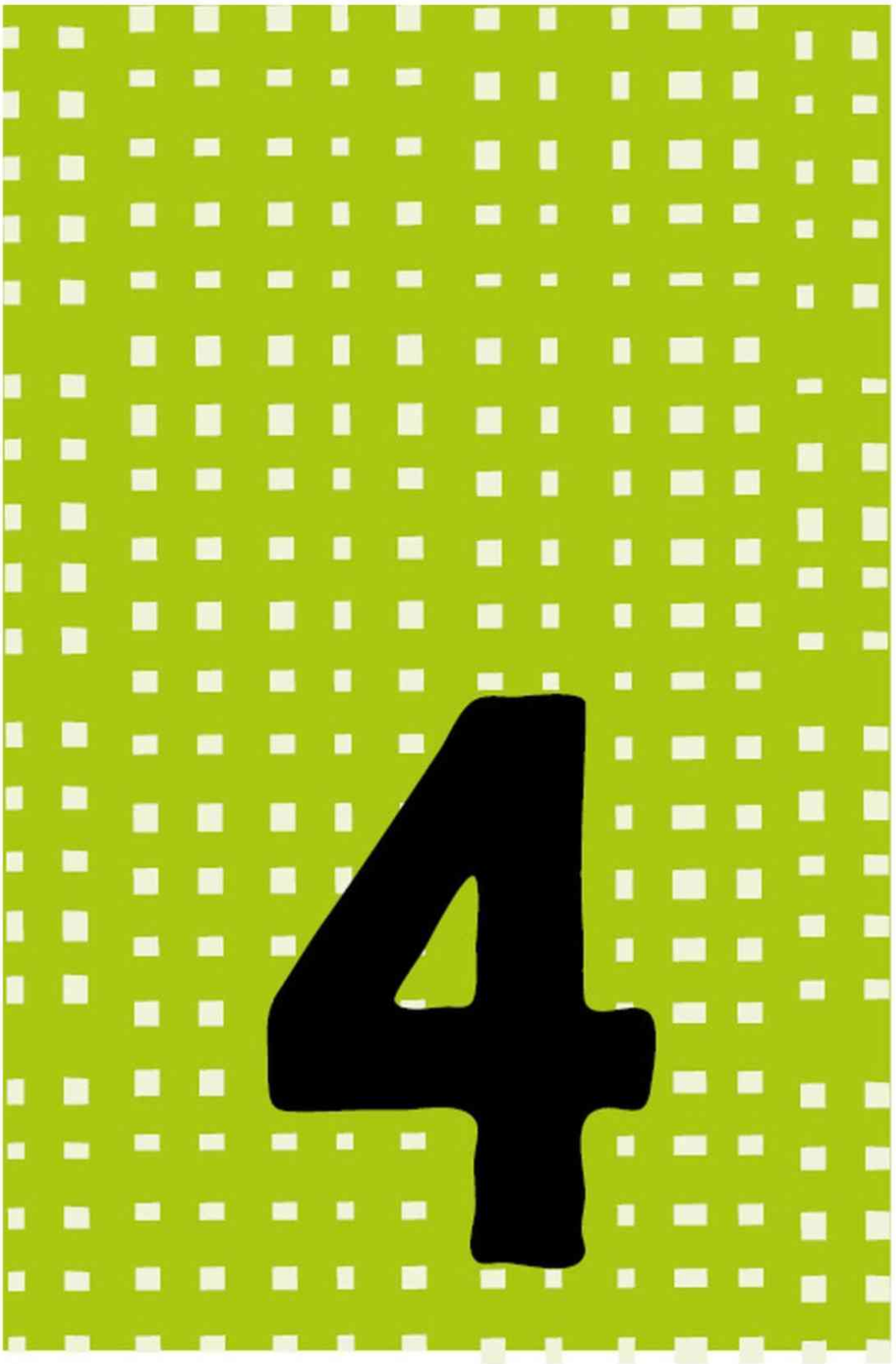
R: Nos extenderemos más sobre el multímetro y cómo medir las diversas propiedades eléctricas (resistencia, voltaje, corriente) en el capítulo 5.

P: ¿Qué pasa si mi medidor tiene una lectura diferente a .000 cuando estoy probando la continuidad?

R: Con el medidor recomendado, lo más importante a tener en cuenta a la hora de comprobar la continuidad es escuchar el tono que emite el medidor, que indica que los componentes están conectados eléctricamente. Los medidores que no disponen de tonos tendrán otra forma de indicar la continuidad en la pantalla.

Resumen

En este capítulo has aprendido a montar y depurar un circuito. Te hemos presentado el multímetro y has aprendido a usarlo para comprobar si todos los componentes están conectados. En el siguiente capítulo, configurarás el Arduino para programarlo y luego lo conectarás a una placa de pruebas, donde lo utilizarás para controlar los componentes del circuito.


4

Programación deL Arduino

En este capítulo, empezarás a ver cómo Arduino controla la electrónica con los programas que vas a escribir. En primer lugar, configurarás el software para programar Arduino en un ordenador; a continuación, conectarás el Arduino a la placa de pruebas. Te mostraremos cómo montar una señal SOS utilizando un led. Aprenderás las reglas básicas sobre cómo escribir código y te familiarizarás con la escritura de código en el entorno Arduino. Para este capítulo necesitarás saber cómo conectar el Arduino a un ordenador y cómo montar un circuito básico en una placa de pruebas.

Arduino, circuitos y código: todo junto

Esta es tu primera oportunidad de combinar el montaje de circuitos con la programación básica. Cuando incorporas a tu circuito el equipo Arduino y la programación, tienes más control sobre el circuito; el led puede parpadear siguiendo diferentes patrones. Aprenderás a programar el Arduino y conectarlo a una placa de pruebas para montar un circuito complejo, en el que la temporización de los componentes del circuito la controla una serie de instrucciones cargadas en el Arduino. Para ilustrarlo, veremos cómo montar una señal SOS luminosa con un led que se enciende y se apaga en función de la temporización controlada por el Arduino.

A partir de este punto, la mayor parte de los proyectos incluirán las tres partes que se muestran en la figura 4.1: el código, el Arduino y un circuito en una placa de pruebas. En este capítulo trataremos la combinación de los tres elementos y cómo interactúan entre sí.

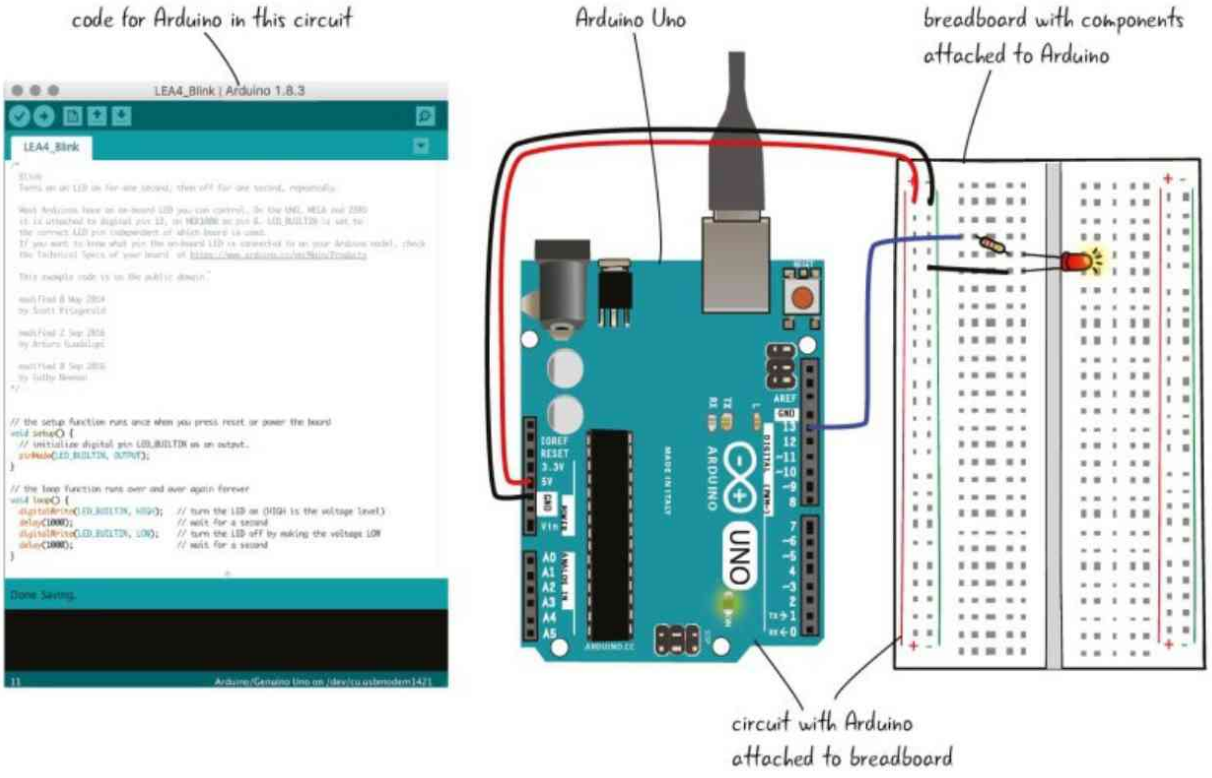


FIGURA 4.1 Código, Arduino y la placa de pruebas.

En el capítulo 2 "Tu Arduino" examinamos el equipo Arduino y algunas de sus características. En el capítulo 3, "Primer contacto con el circuito", aprendiste algo sobre electrónica y circuitos a pequeña escala. En este capítulo te guiaremos en la descarga y uso del IDE de Arduino, que te permitirá cargar el código y cambiar el comportamiento del Arduino.

También te mostraremos los circuitos necesarios a lo largo del libro e incluiremos todos los ejemplos de códigos que necesitarás para ejecutar los proyectos.

Para la codificación, necesitarás el software de Arduino instalado en el ordenador. Descargarás e instalarás el IDE de Arduino. ¿Qué es un IDE? Echemos un vistazo.

¿Qué es un IDE?

Un entorno de desarrollo integrado (IDE, por sus siglas en inglés) es una aplicación de software que permite escribir código y probarlo en el lenguaje de programación que soporta el IDE.

Si tienes experiencia en programación, es posible que hayas usado otro IDE para escribir, probar, depurar y convertir tu código en algo que el ordenador entienda. Si no lo has hecho, el IDE de Arduino es un buen sitio para empezar: es relativamente simple y fácil de entender.

El equipo de Arduino ha diseñado un IDE para usar con sus dispositivos que tiene todas las características que necesitas. Cuenta con un editor de códigos incorporado, que es un programa que se utiliza para escribir los archivos de texto que se crean cuando se programa. Puedes probar el código en el IDE y resolver cualquier problema que surja con la ayuda de un área de mensajes que muestra errores en el código y de una consola que proporciona detalles complementarios sobre la naturaleza de estos errores. Tiene botones para poder comprobar el código, guardarlo, crear una nueva ventana de código, subirlo a Arduino, etc. Esto encaja perfectamente con el diagrama de flujo básico para los proyectos Arduino como se muestra en la figura 4.2.

> **Note** Cargar las instrucciones que escribes en el editor de código es transferirlas a los "cerebros" del Arduino para que lo controle.

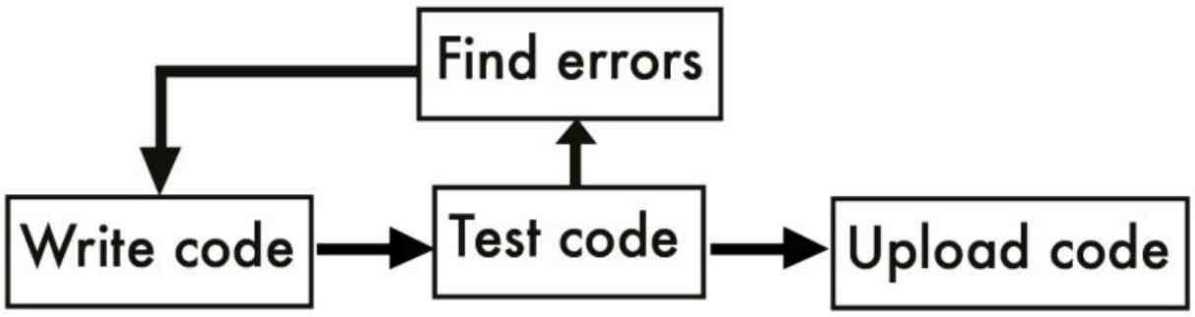


FIGURA 4.2 Flujo de trabajo de Arduino.

El IDE está disponible gratuitamente en el sitio web de Arduino, en arduino.cc/en/Main/Software. Es posible programar Arduino con otro editor de texto o IDE, pero en este libro usaremos el IDE de Arduino.

¿Qué hay en el ide de Arduino?

¿Qué hay en el IDE?

- Una ventana del editor de códigos donde escribimos el código.
- Un área de mensajes que proporciona información sobre el código.
- Una consola de errores que proporciona información detallada y ayuda en la depuración.
- Menús que permiten establecer propiedades para el Arduino Uno y cargar ejemplos de códigos y otras funciones.
- Botones para chequear código, subirlo a Arduino y guardarlo, crear una nueva ventana de código, etc.

¿Qué es el código?

En términos elementales, el *código* se utiliza para dar instrucciones al ordenador. Usamos código para hablar en el lenguaje que el ordenador entiende (en este caso, el lenguaje Arduino) para realizar un conjunto de tareas o para configurar una serie de respuestas programadas. Los ordenadores tienen dificultades para entender lo que se quiere decir, lo que se insinúa o lo que se sugiere. No son capaces de entender los aspectos más sutiles del idioma, así que utilizamos el código para simplificar las instrucciones a un conjunto de comandos a nivel básico.

Hemos visto lo que hay en un IDE, así como una descripción básica del código. Echemos un vistazo al IDE de Arduino.

IDE de Arduino: Primera aproximación

Este es un primer acercamiento al IDE de Arduino. No hay que memorizar

ninguna de las secciones ni tampoco lo que hacen. Más adelante, en este capítulo y en el resto del libro, nos ocuparemos de todas las secciones en detalle. Como se puede ver en la figura 4.3, los menús aparecen en la parte superior de la ventana. También hay botones para funciones de uso frecuente como guardar, un campo donde se puede escribir código y campos de mensajes.

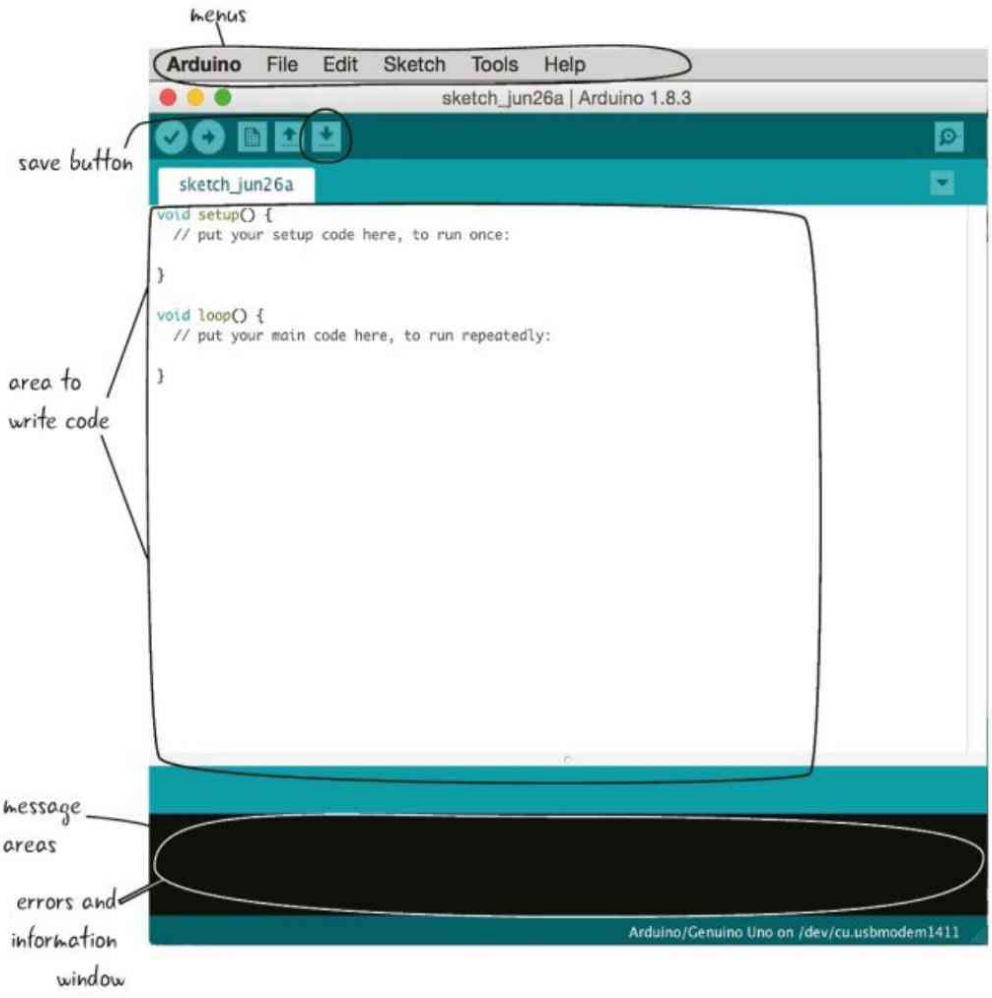


FIGURA 4.3 IDE de Arduino.

Ahora que tienes una idea de lo que hay en un IDE (en el de Arduino), puedes descargarlo e instalarlo en el ordenador.

Descarga del IDE de Arduino: Inicio

El IDE que utilizarás para programar Arduino es gratis y está disponible en el sitio web de Arduino. El procedimiento de instalación es ligeramente diferente para la plataforma Mac que para la plataforma Windows, así que te acompañaremos a través del proceso de descarga e instalación en ambos casos.

> Note **Repetimos la URL para descargar el IDE: arduino.cc/en/Main/Software.**

Si utilizas un Mac

La página de descarga se verá de forma parecida a la figura 4.4. Los sitios web cambian con frecuencia, al igual que el software, por lo que puede parecer diferente cuando lo visites. Haz clic en el enlace para descargar la versión software para Mac. Comprueba que estás descargando la última versión recomendada del IDE de Arduino para Mac.

FIGURA 4.4 Descarga del IDE de Arduino para Mac.

Cuando hagas clic en el enlace, comenzará a descargarse una versión comprimida del IDE. Se cargará en la ubicación de descarga predeterminada del equipo, muy probablemente en la carpeta Descargas. Cuando termine la

descarga, hay que hacer doble clic en el archivo comprimido para descomprimirlo. El archivo descomprimido se llamará Arduino.app y tendrá un aspecto similar al de la figura 4.5.

> **Note** Si no ves .app, no debes preocuparte, significa que el equipo está configurado para no mostrar extensiones de archivo.

FIGURA 4.5 Iconos para la app de Arduino.

Mueve el archivo Arduino.app a la carpeta Aplicaciones del ordenador, como se ve en la figura 4.6.

Ahora ya has descargado e instalado el IDE de Arduino en tu Mac.

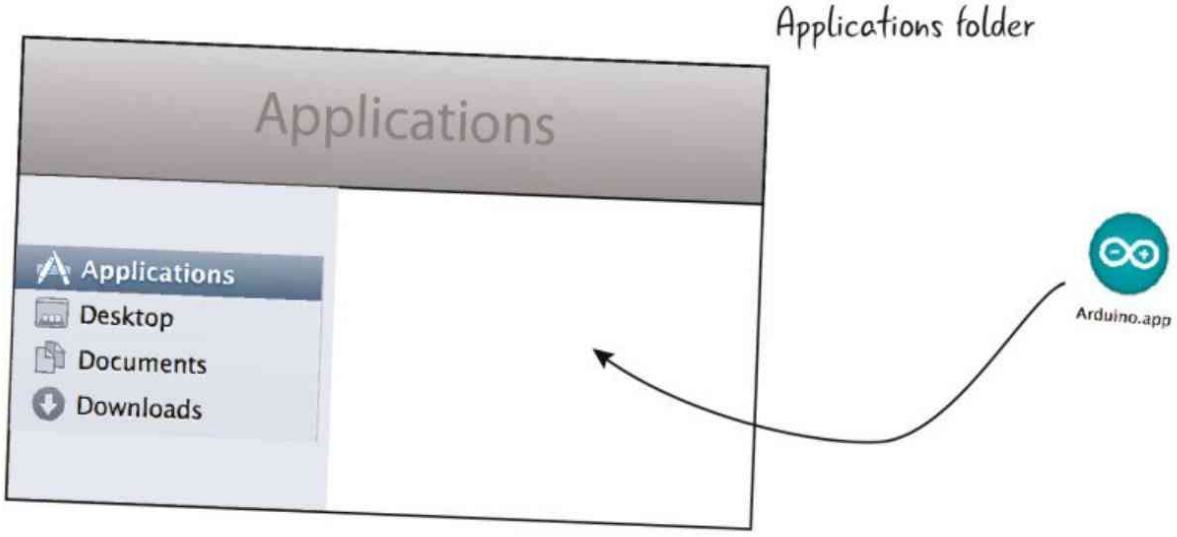


FIGURA 4.6 Arrastra el icono a la carpeta Aplicaciones.

si utilizas un pc con windows

Los pasos a seguir en la descarga y configuración del software para un PC con Windows son muy similares a los que se han seguido para un ordenador Mac, pero hay algunos pasos adicionales menos importantes que necesitas llevar a cabo para asegurar que puedan comunicarse el ordenador y el Arduino.

En primer lugar tienes que descargar el software. La URL es la misma que la de la descarga para Mac. Debes asegurarte de que descargas la última versión recomendada del IDE de Arduino para Windows, como se muestra en la figura 4.7.

> **Note** Nuevamente la URL para descargar el IDE es arduino.cc/en/Main/Software.

FIGURA 4.7 Descarga del IDE de Arduino para Windows.

Te recomendamos utilizar el enlace Windows Installer. Si el ordenador es compartido, por ejemplo, en la escuela o en el trabajo, donde lo utiliza más de un usuario, puede que necesites descargar la versión marcada como "instalación sin administrador".

Cuando haya terminado la descarga, habrá un archivo EXE con el nombre de la versión de Arduino en la ubicación de descarga predeterminada,

generalmente la carpeta Descargas. Haz un doble clic en este archivo para iniciar el proceso de instalación.

El primer cuadro de diálogo te pide que aceptes el Contrato de licencia de Arduino (figura 4.8). Si haces clic en **Acepto** te llevará al siguiente paso de la instalación.

FIGURA 4.8 Acuerdo de licencia de Arduino.

En Arduino Setup Installation Options, tienes que marcar las casillas de "Install USB Driver" y de "Associate .ino Files" (figura 4.9). "Create Start Menu Shortcut" y "Create Desktop Shortcut" son opcionales, pero te ayudarán a navegar en el futuro en el IDE de Arduino.

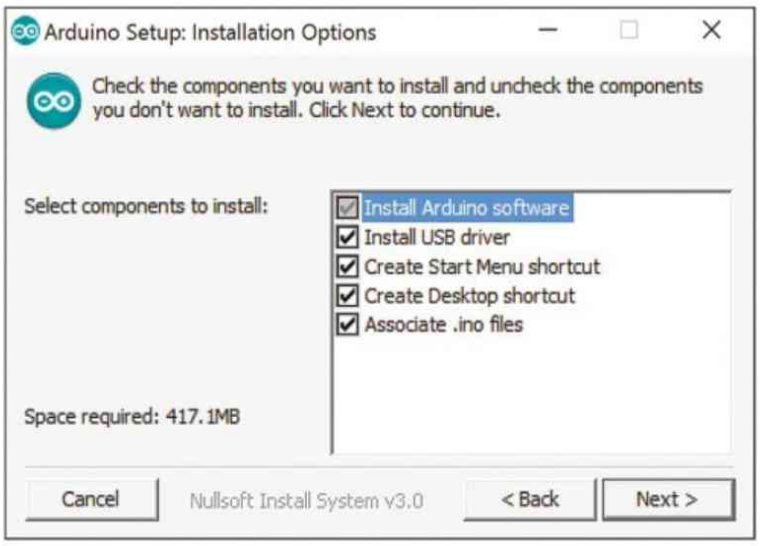


FIGURA 4.9 Opciones de la instalación.

En función de la configuración y de la versión de Windows, puede aparecer un cuadro emergente de seguridad de Windows que pregunta sobre la instalación del controlador USB. Haz clic en **Instalar** cuando aparezca el cuadro de diálogo de seguridad para permitir la instalación completa del IDE (figura 4.10).

FIGURA 4.10 Cuadro de diálogo de seguridad.

¡Eso es todo! Ahora el IDE de Arduino está listo para funcionar en tu PC con Windows.

Conexión del Arduino al ordenador

Has instalado el IDE de Arduino, así que ahora es el momento de conectar el Arduino al ordenador para que puedas programarlo.

Conecta el cable USB al Arduino y el otro extremo del USB, al ordenador, como en la figura 4.11.

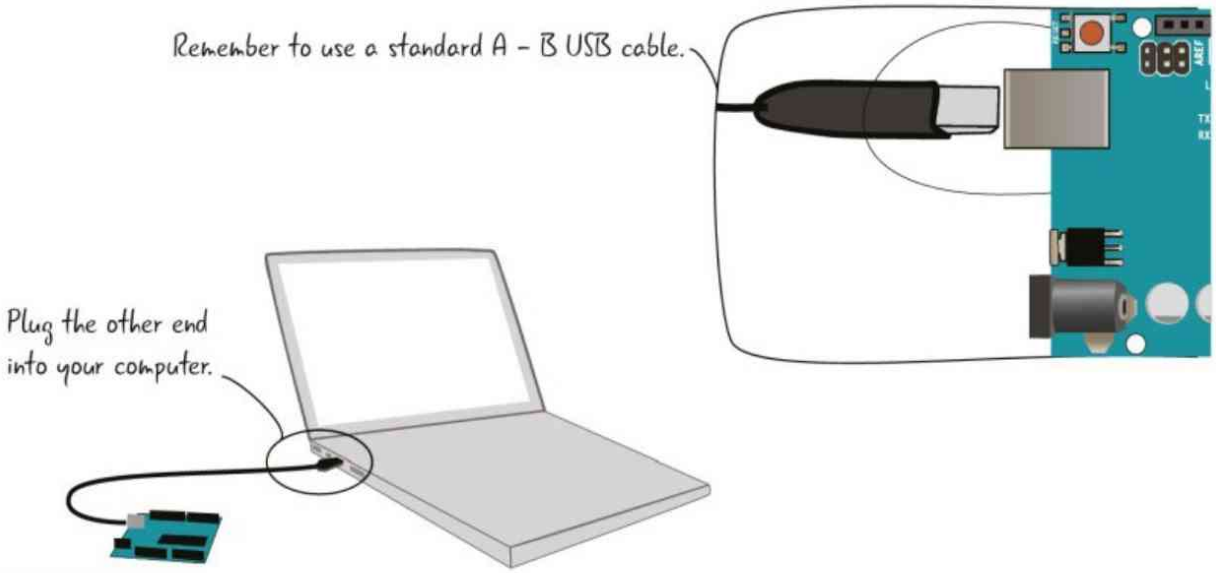


FIGURA 4.11 Conexión del Arduino al ordenador.

El led que corresponde a "ON" debe encenderse, y si el equipo Arduino es nuevo y está preparado, debe parpadear la luz situada cerca del Pin 13, al igual que cuando probaste la conexión del Arduino en el capítulo 2 (figura 4.12).

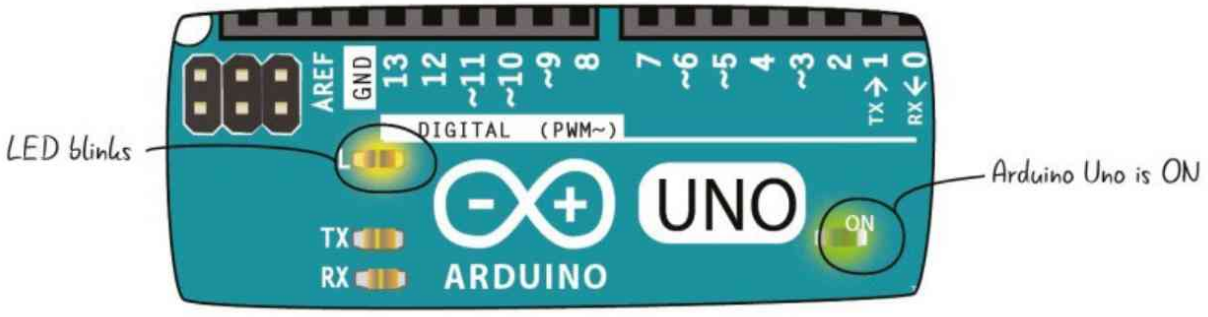


FIGURA 4.12 Ledes indicadores.

El IDE de Arduino: ¿Qué hay en la interfaz?

Echemos un vistazo al IDE de Arduino de la figura 4.13 ahora que ya lo has puesto en marcha.

El IDE te permite comprobar si Arduino está conectado al ordenador, comprobar si el código tiene errores, cargar cualquier código que escribas para controlar el Arduino, y tiene algunas otras opciones útiles para entender cómo se comporta el Arduino. Examinaremos todas las funciones con mucho más detalle antes de que empieces a escribir código para el Arduino.

Al programa que escribimos en el editor de código para Arduino se le llama *sketch* (proyecto). Cuando inicies el software por primera vez, verás el esqueleto del sketch. Explicaremos cómo se usa el código de este programa cuando empieces a programar el Arduino.

> Note *Sketch* es el nombre de un programa que escribes para el Arduino.

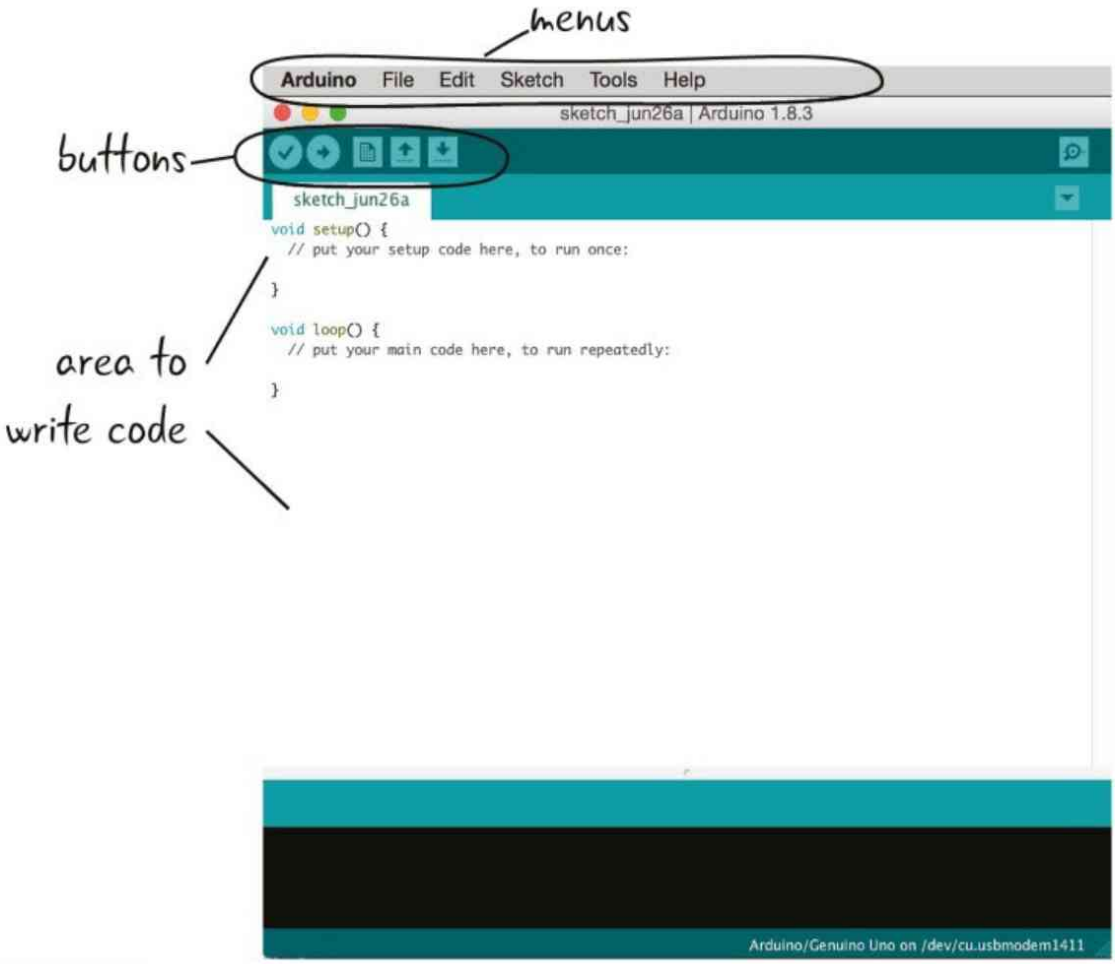


FIGURA 4.13 Fundamentos del IDE de Arduino.

> **Warning** Una peculiaridad del IDE de Arduino es que si cierras todas las ventanas del sketch, el IDE intentará cerrarse. Te pedirá que guardes un sketch si has hecho algún cambio, de lo contrario se cerrará.

Tendrás que realizar algunos ajustes antes de empezar a programar. Vamos a verlos ahora.

Configuración del IDE

Es necesario realizar dos importantes configuraciones en el IDE de Arduino para que el ordenador pueda comunicarse con el Arduino Uno. Necesitas especificar la versión hardware del Arduino, o de la placa, que estás utilizando,

y qué conexión o puerto utilizarás para las comunicaciones entre el Arduino y el ordenador. Estas configuraciones serán las mismas siempre que utilices el mismo Arduino Uno. Las configuraciones serán diferentes si utilizas otro Arduino. En este libro utilizaremos siempre el mismo Arduino para todos los proyectos.

Especificación de la versión hardware del Arduino

Viste en el capítulo 1 que hay muchas versiones diferentes de Arduino. Para programar el tuyo, debes indicar en el software qué versión de la placa Arduino estás usando.

Para ello, ve al menú HERRAMIENTAS y selecciona "Board", como se muestra en la figura 4.14. En el menú desplegable, selecciona "Arduino Uno/Genuino". Una vez hecho esto, debes configurar un puerto a través del cual el Arduino se comunicará con el ordenador.

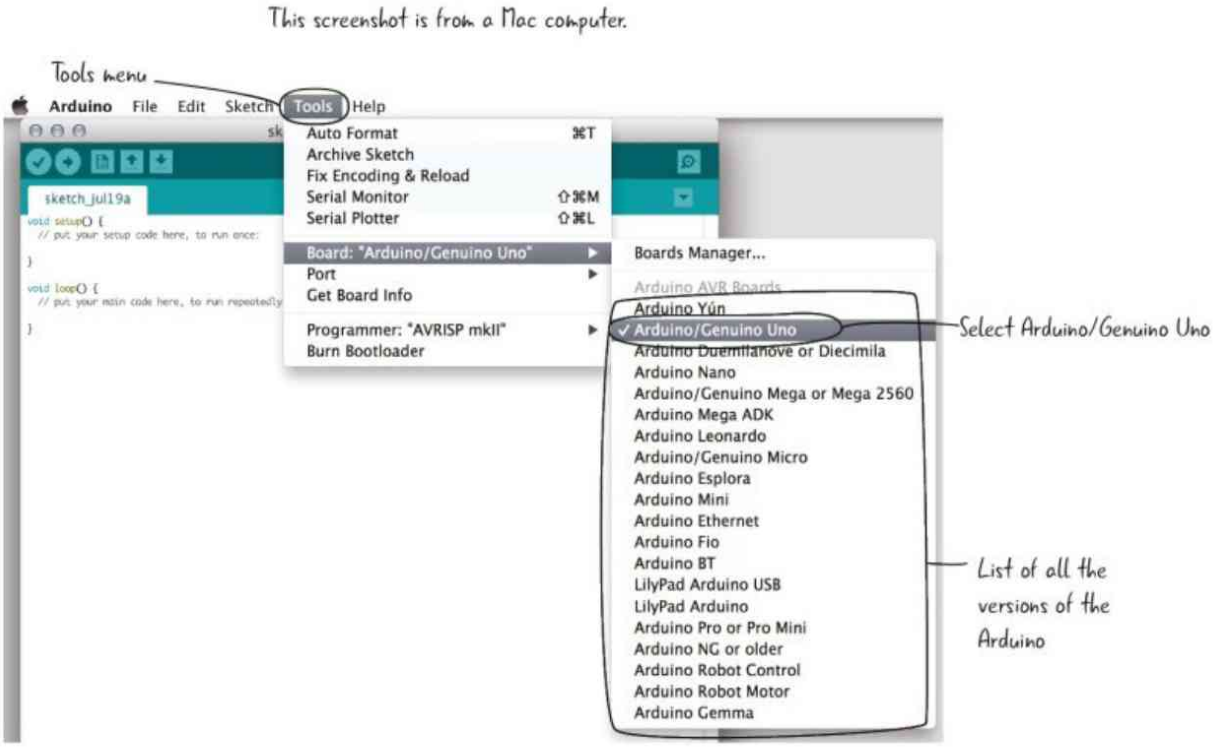


FIGURA 4.14 Selección de la placa del Arduino.

Especificación del puerto que vas a utilizar

Hay un puerto en el Arduino que se comunica con un puerto en el ordenador cuando los dos están conectados por un cable USB. Puedes ver el puerto como el canal a través del cual los dos dispositivos se hablan entre sí. Ahora necesitas configurar el IDE de Arduino para que el puerto correcto del ordenador se comunique con el Arduino.

La selección del puerto correcto se ve ligeramente diferente en un Mac y un ordenador con Windows. Vamos a ver las capturas de pantalla de ambos. Reuerda que estás configurando el ordenador para hablar con Arduino Uno, ya que es la versión de Arduino que estamos usando para los proyectos de este libro. Veamos primero el Mac; si lo que tienes es un PC con Windows, puedes pasar a la siguiente sección.

> Note Un *puerto* es un canal de comunicación que conecta el Arduino con el ordenador.

Selección del puerto en Mac

Para configurar el puerto correcto para que el ordenador se comunique con el Arduino, ve al menú HERRAMIENTAS y selecciona “Port”, como se muestra en la figura 4.15.

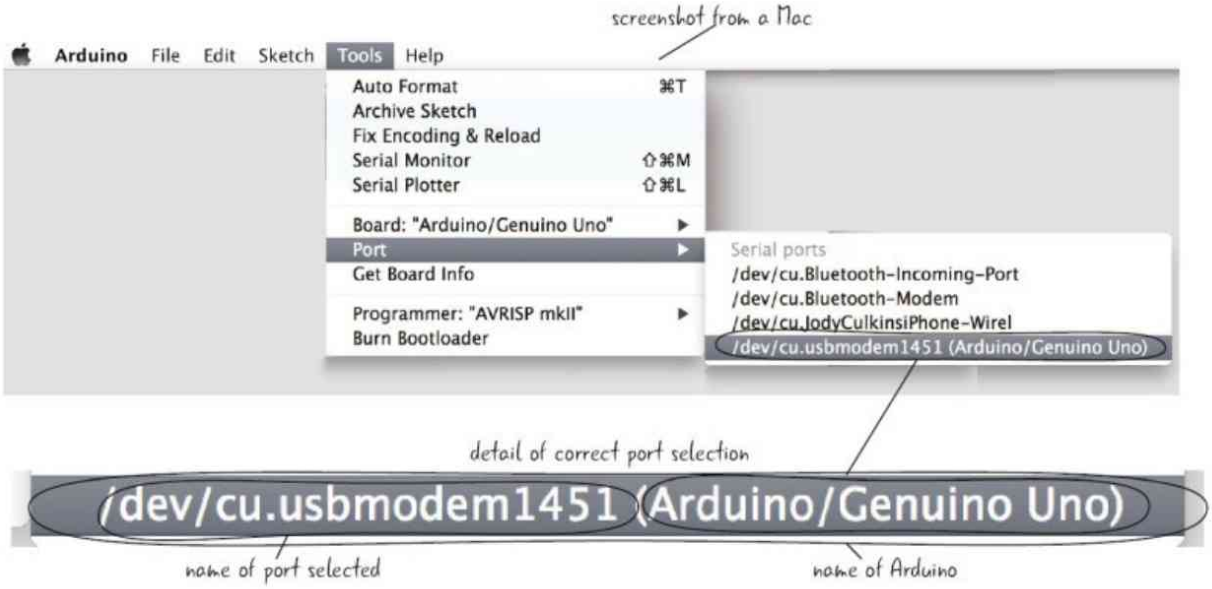


FIGURA 4.15 Selección del puerto correcto.

En el Mac, selecciona el puerto cuya descripción incluye *dev* y *cu* y que está etiquetada como Arduino/Genuino Uno. "Dev" es un prefijo añadido por el Mac, "cu" es la abreviatura de "call-up", y Arduino Uno es la versión del hardware que utilizas. En nuestro anterior ejemplo, el número al final de ese ítem del menú es 1451; en tu pantalla será diferente a este ejemplo, y podría ser diferente cada vez que conectes el Arduino. En algunas versiones de software o del sistema operativo, se puede ver *tty* en lugar de "cu" en la lista de puertos. Debería funcionar también; lo importante es que veas Arduino/Genuino Uno en la descripción del puerto.

No ocurrirá nada irreparable si seleccionas el puerto equivocado, pero el Arduino y tu ordenador no sabrán hablar entre ellos. Si parece que el Arduino y el ordenador no se comunican, vuelve a consultar la lista de puertos y asegúrate de que has seleccionado el correcto.

Selección del puerto en Windows

Veamos la selección de puertos en una máquina con Windows (figura 4.16). En un PC, todos los nombres de los puertos comenzarán con COM. Ve al menú

HERRAMIENTAS

, selecciona “Port”, y luego selecciona el número COM que coincida con la etiqueta Arduino Uno/Genuino etiquetado como “Serial Ports”. Será algo parecido a COM3 (Arduino Uno/Genuino).

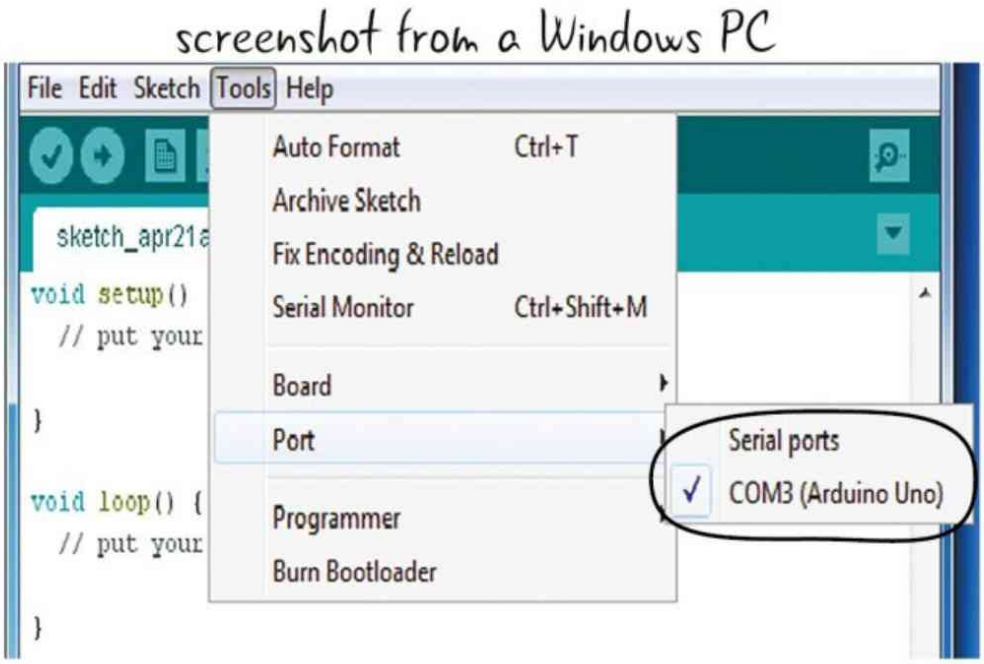


FIGURA 4.16 Selección del puerto.

¿PREGUNTAS?

P: ¿Siempre elegiré el puerto Arduino Uno/Genuino?

R: No necesariamente. Esa es la versión de la placa Arduino que utilizamos en este libro, ya que que todos los proyectos del libro usan Arduino Uno, pero puede que quieras usar otras versiones de Arduino más adelante, cuando montes tus propios proyectos.

P: A veces hay otros puertos listados en el desplegable. ¿Qué son?

R: Son otros puertos que utilizan diferentes medios para que el ordenador se comunique con otros dispositivos. No te preocupes por ellos, no los usaremos.

P: ¿Qué pasa si no tengo el ordenador conectado al Arduino? ¿Veré el puerto para conectarme a mi Arduino entonces?

R: No. Para poder ver el puerto correcto, debes tener tu Arduino y el ordenador conectados con un cable USB.

Ahora que has configurado el puerto y la placa Arduino correcta, vamos a echar un vistazo al IDE de Arduino que vas a utilizar para crear el código.

la ventana de código

Ya nos hemos referido antes a las partes del IDE de Arduino; ahora vamos a echarles un vistazo más de cerca en la figura 4.17.

Como en la mayoría de software, en la parte superior de la interfaz del IDE hay menús que te permiten realizar varias acciones, como crear nuevos archivos, guardarlos y muchas otras opciones. Hay iconos de botones que también te permiten acceder rápidamente a algunas de las acciones que se realizan con mayor frecuencia. Al hacer clic en el botón **Verificar** compruebas que no haya errores en el código. Al hacer clic en **Subir** transfieres tu código desde el ordenador a tu Arduino para que pueda ejecutarse en la placa del

Arduino. Hay una ventana en la que puedes escribir el programa, y las áreas de mensaje te proporcionan información sobre ese programa. Explicaremos más sobre los mensajes cuando trabajemos con el IDE; por ahora, debes saber que te dirá si el código tiene errores y también informará sobre la cantidad de espacio que usa en la memoria del Arduino.

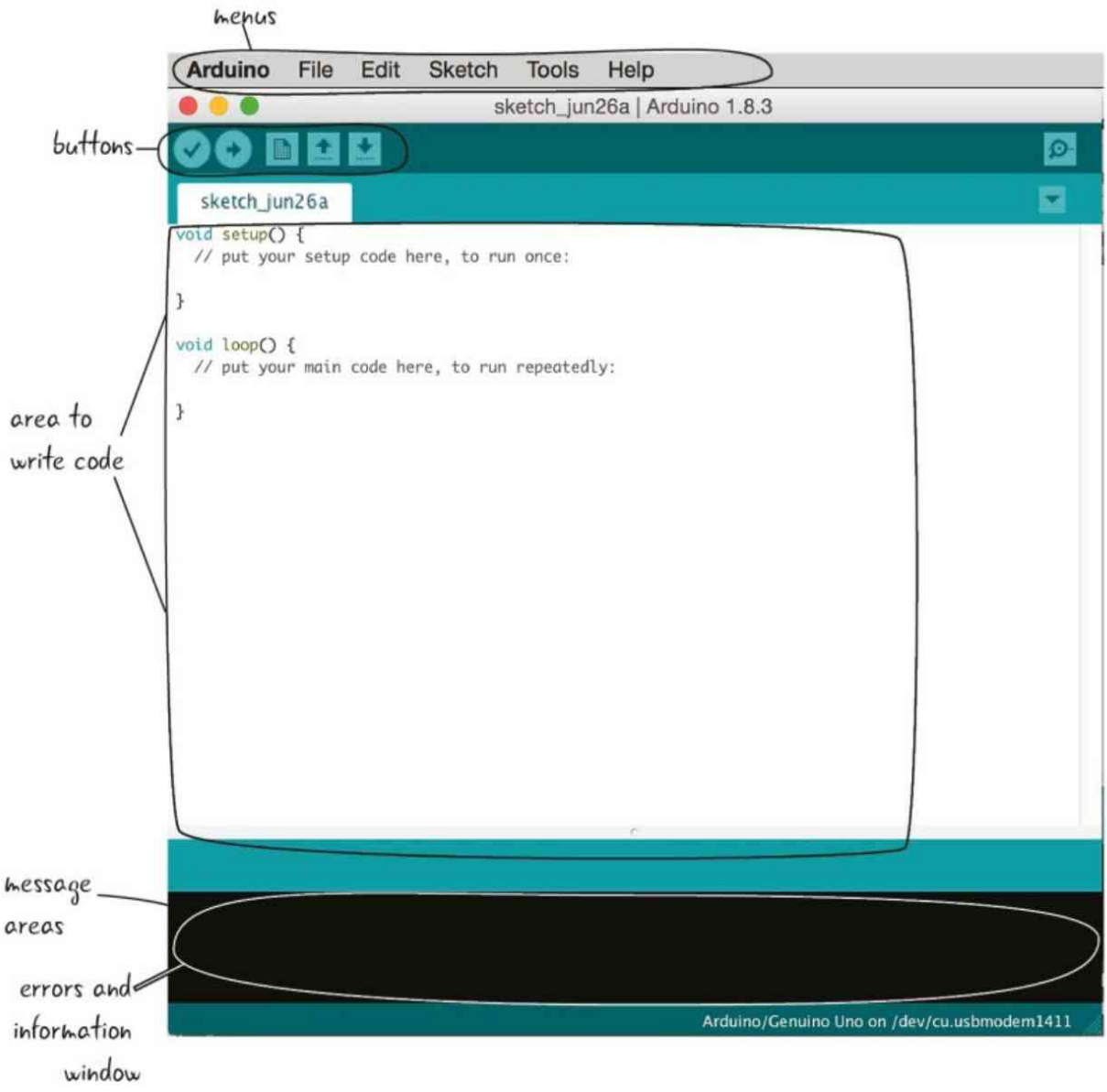


FIGURA 4.17 IDE de Arduino comentado.

Veamos un poco más de cerca los botones situados en la parte superior del editor de código que aparecen en la figura 4.18.

Estos botones te permiten acceder rápidamente a las acciones que realizas con mayor frecuencia en la ventana de código. Estas acciones incluyen comprobar si el código tiene algún error (verificación), enviar el código a la tarjeta del Arduino (carga), crear un nuevo archivo, abrir un archivo y guardarlo.

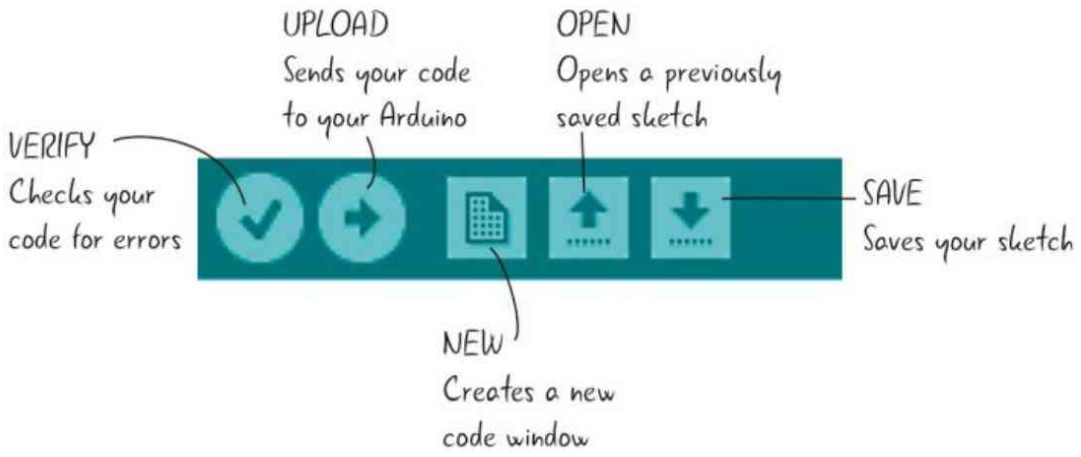


FIGURA 4.18 Botones en el IDE de Arduino.

Usaremos todos estos botones muy pronto, pero antes vamos a aclarar lo que significa realmente escribir un sketch.

Sketch: La unidad básica de programación del Arduino

Puedes pensar en un programa Arduino, o sketch, como un grupo completo de instrucciones para realizar tareas específicas. Un sketch contiene el código, o las instrucciones, para realizar esa tarea o tareas. Es posible tener múltiples sketches separados abiertos a la vez, del mismo modo que un programa de hoja de cálculo puede tener más de una hoja abierta al mismo tiempo. Veamos con más detalle qué forma un sketch.

Cada programa que subes al Arduino se considera un *sketch*. Un sketch puede ser muy simple o extremadamente complejo. Puede encender y apagar un solo led o puede controlar 10 o más motores en función de las entradas al sensor. Aunque cada sketch corresponde a una tarea, esa tarea podría constar de múltiples partes. Por ejemplo, el programa puede detectar valores de parámetros del mundo exterior (como niveles de luz) y usarlos para activar altavoces y ledes. Todo eso iría en un sketch.

El nombre del sketch aparece en una pestaña en la esquina superior izquierda del editor de códigos. La figura 4.19 muestra ejemplos de sketches de Arduino.

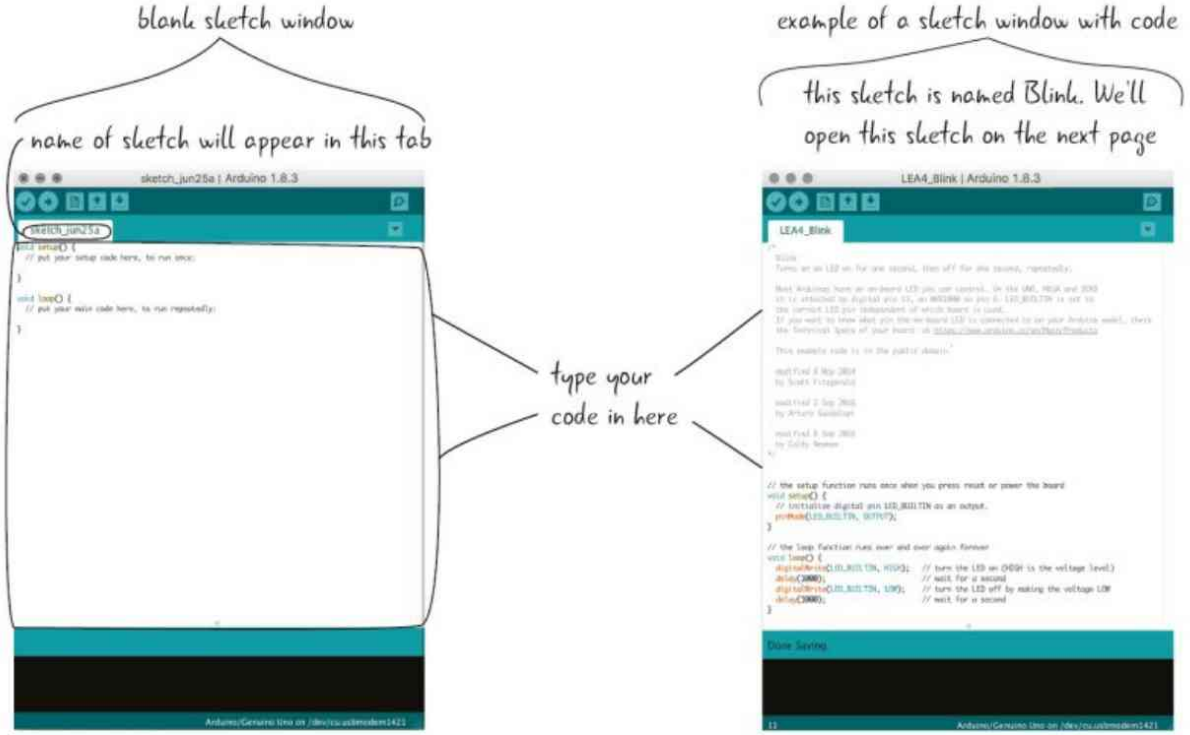


FIGURA 4.19 Una pantalla de sketch en blanco y otra con código.

cómo abrir un ejemplo de sketch

Antes de que empieces a escribir tu propio código, vamos a explorar un ejemplo incluido en el IDE de Arduino. El IDE tiene bastantes ejemplos (códigos de muestra) que ilustran muchas de las cosas integradas en él que puede hacer el Arduino. Puedes cargar un ejemplo en la ventana de código y subirlo al Arduino cuando esté conectado al ordenador.

En primer lugar, abre el sketch de ejemplo llamado Blink. Para ello selecciona Archivo > Ejemplos > 01.Basics > Blink, como se muestra en la figura 4.20.

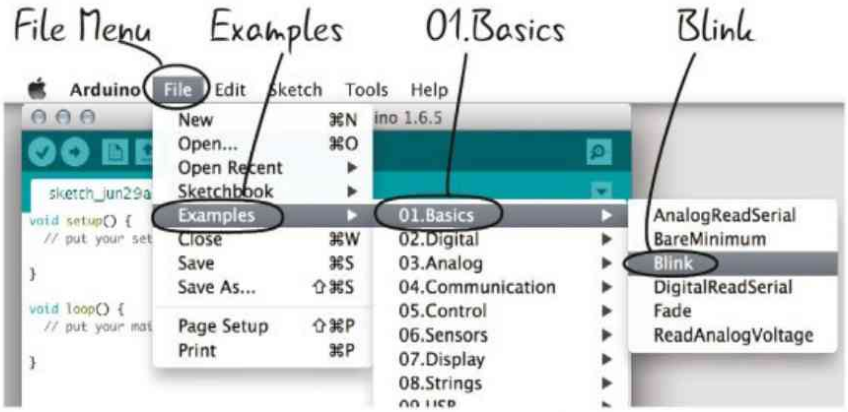


FIGURA 4.20 Cómo abrir el sketch Blink (Parpadeo).

cómo guardar el sketch

De forma predeterminada, los sketches de Arduino se guardarán en la carpeta Arduino, dentro de la carpeta Documentos del ordenador. Es una buena idea seguir guardándolos en este espacio ya que facilita el retorno a los archivos. Arduino también realiza un seguimiento de los archivos anteriores guardados dentro de esta carpeta en el desplegable "Sketchbook dropdown" del menú ARCHIVO.

Aunque estés utilizando el código de un ejemplo, lo mejor es guardarlo ahora con un nombre diferente para que puedas volver más tarde al código de ejemplo original sin modificar. De esta manera, cuando realices cambios y guardes el sketch sabrás que no lo has guardado accidentalmente en el sketch de ejemplo Blink. Guarda el sketch como LEA4_Blink para que puedes encontrar los cambios más tarde.

Guarda los archivos con frecuencia

Tienes que acostumbrarte a guardar los archivos. De la misma forma que no te gustaría perder un documento de trabajo u otro proyecto, guardar pronto y a menudo puede ayudar a evitar frustraciones si por alguna razón el ordenador cierra el IDE de Arduino (por una desconexión de energía, un contratiempo pasajero, etc.). Aunque las probabilidades de que suceda esto son

bajas, cuando ocurra te alegrarás de no tener que repetir todo el trabajo que hiciste, porque guardaste el proyecto y no tienes que preocuparte por él.

> **Tip** Guarda los archivos del sketch mientras trabajas.

Cómo cargar un sketch en Arduino

Ahora que has guardado el sketch de ejemplo con un nombre nuevo, es hora de subirlo al Arduino. Antes de subirlo, compueba si hay errores. Aunque estés usando el código incorporado en el IDE, tienes que habituarte a verificar siempre el código antes de subirlo.

Hay dos botones de los que hablamos anteriormente que debes tener en cuenta cuando estés preparado para cargar el código: **Verificar** y **Subir**. Hemos resaltado ambos botones en la figura 4.21.

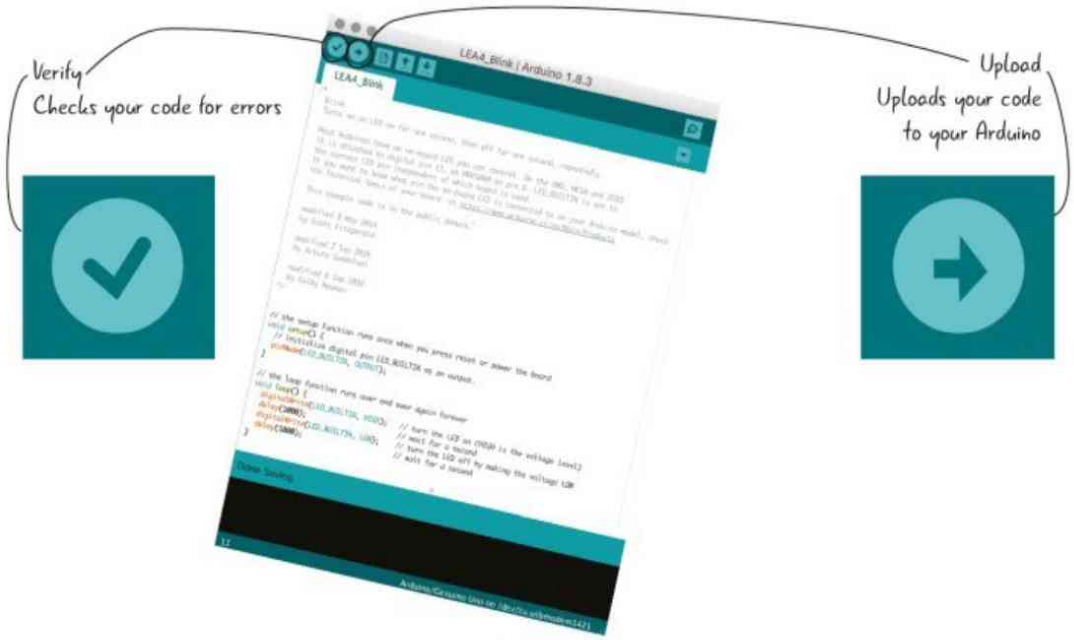


FIGURA 4.21 Botones de verificar y cargar en el IDE de Arduino.

Paso 1: verificar el sketch

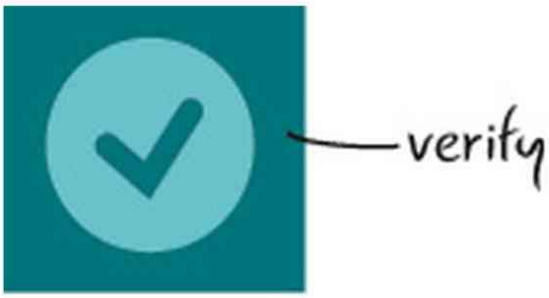


Figura 4.22:**Botón de verificar.**

La verificación asegura que el código está configurado correctamente. Tienes que hacer clic en el botón **Verificar** para asegurarte de que no hay errores (figura 4.22). A menos que hayas hecho cambios en el sketch LEA4_Blink antes de guardarlo, todo funcionará correctamente.

La ventana de mensajes en la parte inferior del IDE de la figura 4.23 presentará el mensaje "Done compiling" y no mostrará ningún error.

FIGURA 4.23 La ventana de mensajes.

Cuando verifiques el código, si hay algún error en el sketch, se mostrará un mensaje que notificará que algo está mal. El IDE de Arduino solo sabe de errores de programación, no de errores que hayas podido cometer al configurar el circuito con el Arduino. (Trataremos esos tipos de errores a medida que progresemos en el libro). Cuando escribimos texto en la ventana del IDE de Arduino, el código se parece a algo que pueden entender las personas, pero el Arduino no sabe cómo interpretarlo. Al hacer clic en **Verificar** para comprobar errores, el ordenador convierte temporalmente el código en un idioma que el Arduino entiende.

Paso 2: cargar el sketch

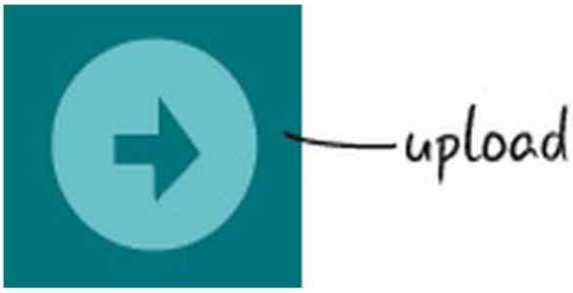


Figura 4.24:**Botón de carga.**

Cuando hacs clic en **Subir** (figura 4.24), el ordenador convierte el código a un lenguaje que el Arduino entiende y entonces inmediatamente comienza a enviar este programa a través del cable USB al Arduino.

Continuación: barra de estado y ventana de mensajes

Una vez que hagas clic en el botón **Subir**, aparecerá en la ventana del IDE de Arduino una barra de estado que te indicará el progreso de la carga y una ventana de mensajes con información como el tamaño del sketch. La barra de progreso y la ventana de mensajes se parecen un poco a la figura 4.25.

Una vez que el archivo se haya enviado al Arduino, la ventana de mensajes dirá "Done uploading".

¡Eso es todo! Ahora el código de la ventana del IDE se ejecuta en el Arduino.

FIGURA 4.25 Barra de progreso de la carga.

Ejecutar el sketch LEA4_Blink

Ahora que has subido el sketch al Arduino, este seguirá funcionando mientras se alimente del ordenador a través del cable USB. El código que has subido al Arduino contiene las instrucciones que le dicen que haga que la luz parpadee. El led próximo al Pin 13 se encenderá y permanecerá encendido durante un segundo, luego se apagará durante un segundo; esto se repite una y otra vez. El proceso se ilustra en la figura 4.26. En breve, estudiaremos el código en detalle, y veremos exactamente cómo funciona.

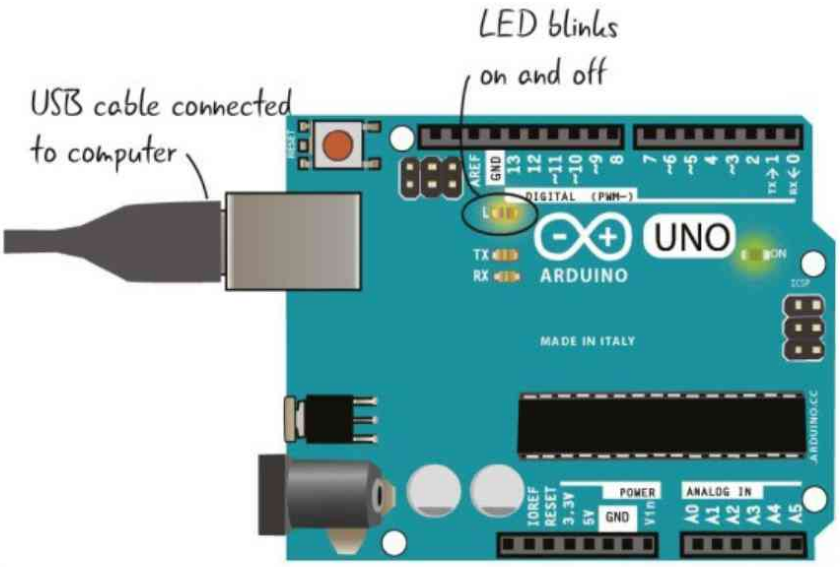


FIGURA 4.26 El led parpadea.

Si el sketch LEA4_Blink no se ejecuta, puedes volver a aplicar el método utilizado para descubrir qué problema está impidiendo que el código funcione. Ya has visto esto antes cuando hemos tratado la electrónica, y se conoce como *depuración*.

> **Note** *Depuración* es el nombre del proceso para resolver problemas con el circuito y con el código en los proyectos Arduino.

Depuración: Qué hacer si el LED no parpadea

Si la carga se ha realizado correctamente y el led está parpadeando, no hay nada que arreglar. ¿Pero y si el led no se iluminara? Del mismo modo que utilizaste la depuración para buscar problemas en el circuito, depurarás el código en los casos que se presenten a lo largo de todo el libro, buscando metódicamente problemas que impidan que este funcione correctamente. También identificarás problemas relacionados con la configuración del hardware del Arduino. Si se presenta algún problema con el sketch LEA4_Blink tienes que asegurarte de que:

- El cable USB está bien conectado al ordenador y al Arduino (figura 4.27).
- Has seleccionado el tipo de tarjeta y el puerto serie adecuados en los menús (figura 4.28).

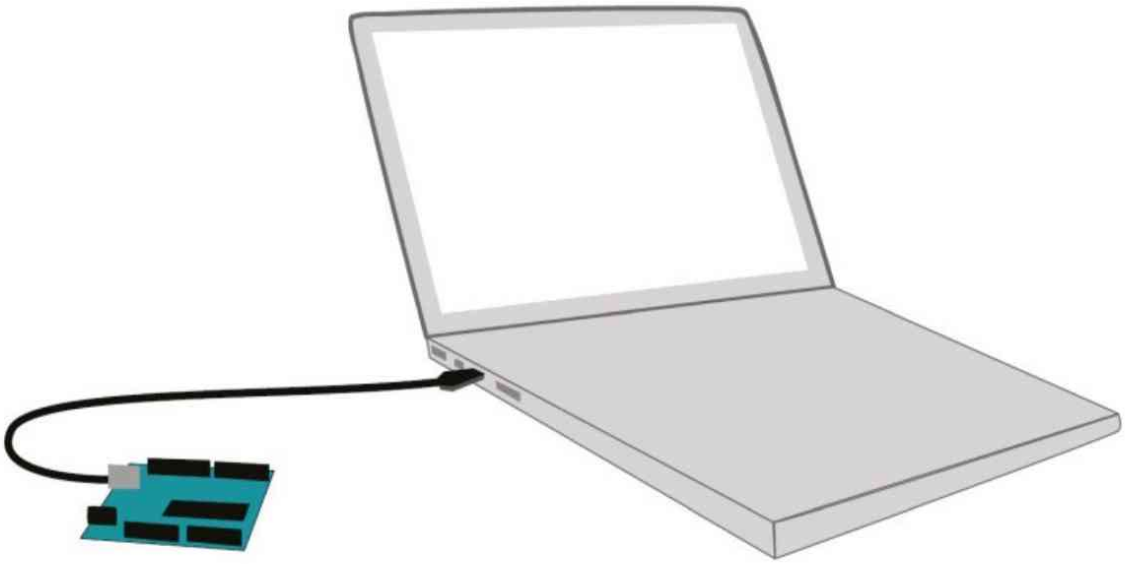


FIGURA 4.27 Asegúrate de que el ordenador está bien conectado al Arduino mediante el cable USB A-B.

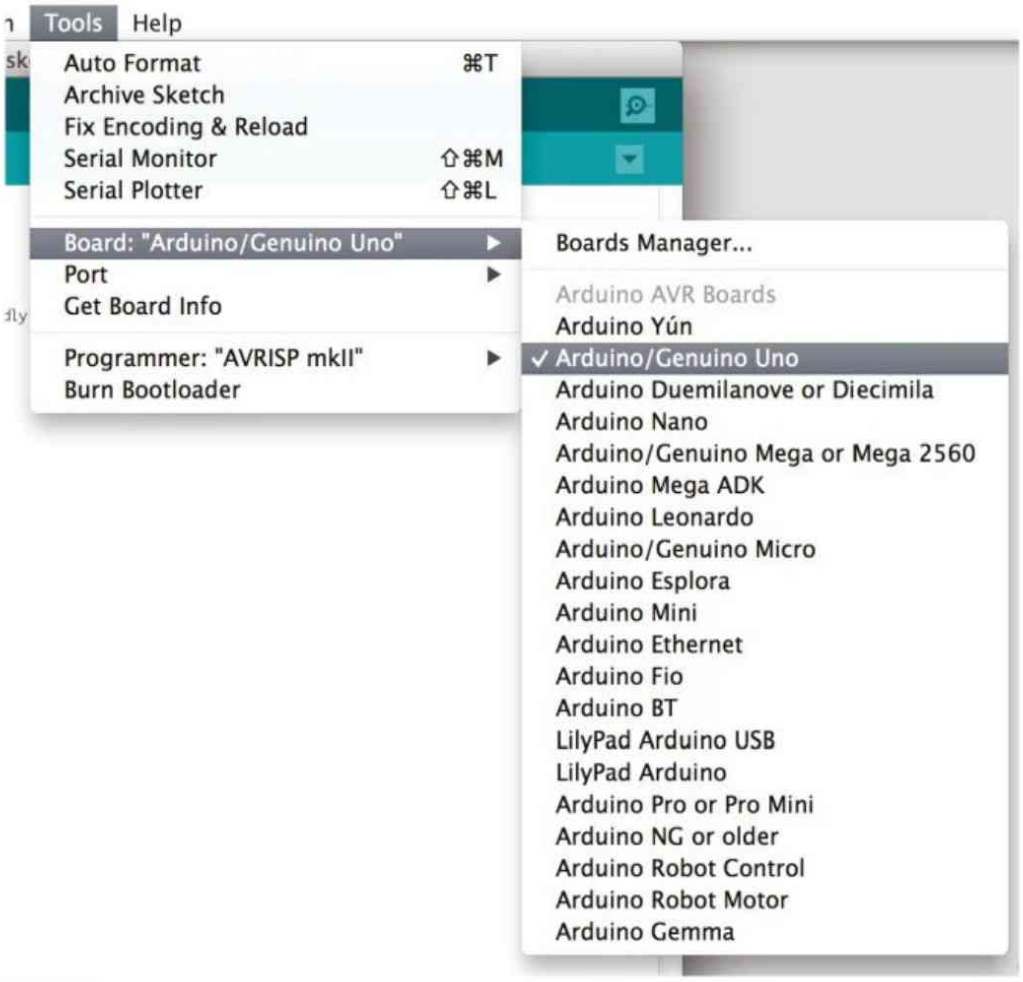


FIGURA 4.28 Asegúrate de que has seleccionado la placa adecuada.

Si tu Arduino parece no estar respondiendo, siempre puedes pulsar el botón de reinicio antes de proceder a la carga, como se muestra en la figura 4.29. El botón de reinicio apagará el Arduino durante un momento antes de volver a encenderlo.

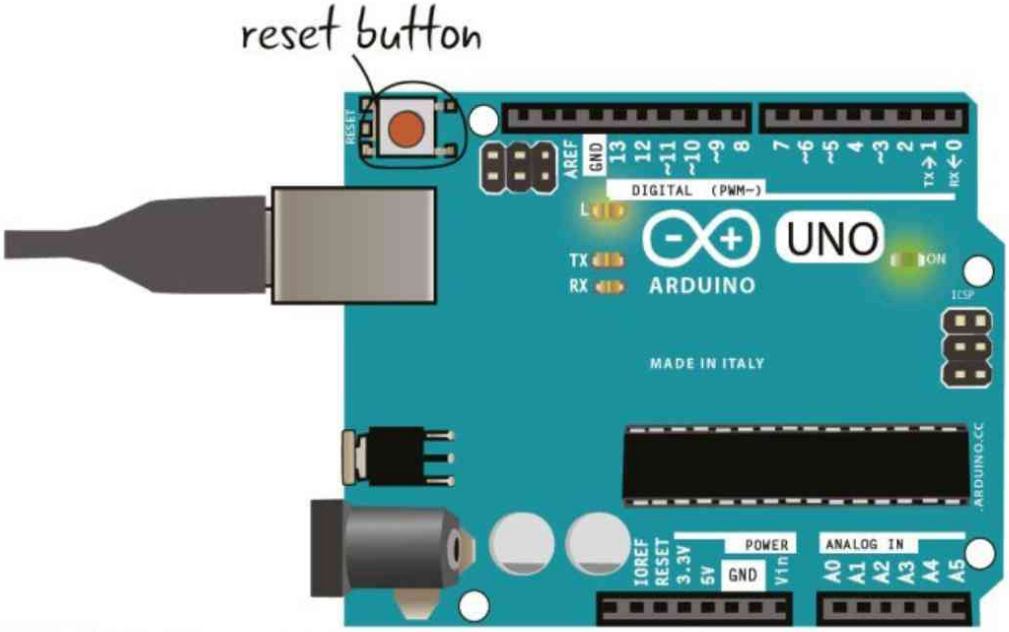


FIGURA 4.29 El botón de reinicio en el Arduino.

También puedes intentar cambiar el puerto USB o reiniciar el ordenador si ninguna de las soluciones anteriores funciona. Veremos todo tipo de trucos de depuración de código a lo largo del libro, pero estos pocos consejos básicos sobre el Arduino pueden ahorrarte muchos dolores de cabeza más tarde.

El sketch LEA4_Blink funcionará siempre y cuando el Arduino esté alimentado, pero ¿cómo funciona realmente?

El sketch LEA4_Blink: VISIÓN DE CONJUNTO

La figura 4.30 muestra una captura de pantalla del sketch LEA4_Blink. Este es un resumen rápido de las partes del sketch; después de que lo veamos, revisaremos cada parte del sketch en detalle.

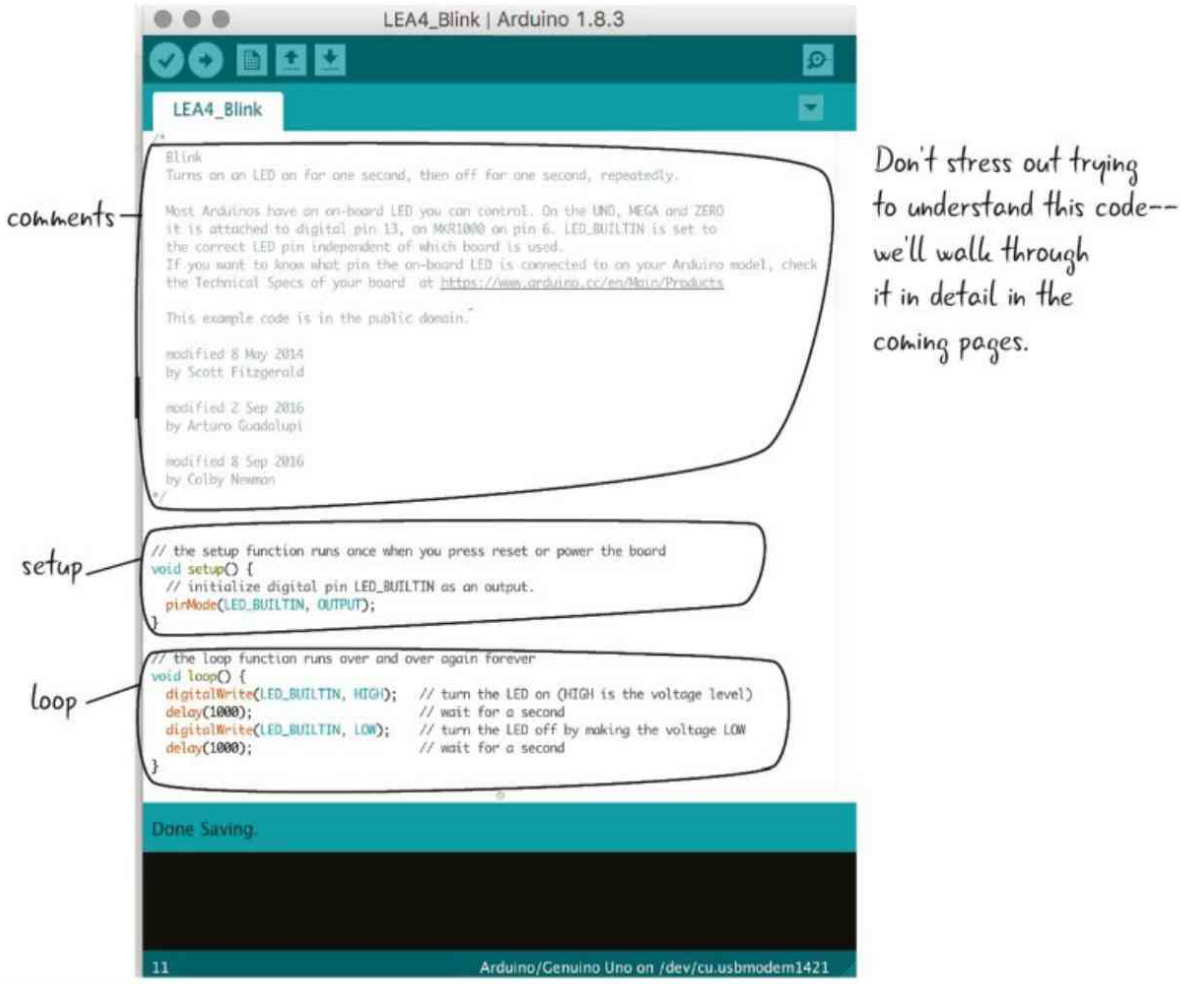


FIGURA 4.30 Primera vista del sketch LEA4_Blink.

Los *comentarios* son notas al programador, texto que no forma parte del programa. En un sketch de Arduino, setup() es donde ubicas las partes del programa que suceden solo una vez, y loop() es donde pones lo que quieres que suceda un determinado número de veces.

En este sketch, el código en setup() y en loop() está escrito en lenguaje de programación Arduino. Si miras el código en el IDE de Arduino, verás que

las partes del código son de diferentes colores; algunas son naranjas, otras azules y otras negras. Estos colores representan algunos de los diferentes roles del código. No es importante memorizar o conocer estos colores; solo están ahí para ayudarte a separar visualmente el propósito de las distintas partes.

> **Note** **En el sketch LEA4_Blink, el código en `setup()` y en `loop()` está especificado en lenguaje de programación Arduino.**

En breve vamos a analizar todas las partes de un sketch detalladamente, pero primero vamos a ver la sección de comentarios en la parte superior del código.

Comentarios: Cómo hacer saber a otros lo que estás pensando

Los comentarios en el código se utilizan para escribir notas a cualquiera que pueda leerlo posteriormente. Este puede ser tu caso cuando vuelvas a un sketch que creaste anteriormente, o el caso de otros con los que compartes el código. Los comentarios no tienen ningún efecto en absoluto sobre cómo el ordenador interpreta el texto, y solo están ahí para recordarte o para facilitar pistas a otros de lo que pretendías hacer, o de cómo el programa funciona como un todo. Usaremos comentarios en los ejemplos de código para ayudar a explicar las secciones. Es un buen hábito que escribas comentarios para poder volver al sketch más tarde y recordar lo que está pasando.

La primera parte del sketch LEA4_Blink incluye un comentario sobre cómo funciona el archivo. Este comentario es largo, con mucha información, pero a veces los comentarios son cortos, solo una o dos palabras. Como puedes ver en este ejemplo, los comentarios a veces tienen información sobre el autor del código y la fecha. En este caso también nos dicen que el código es de

dominio público.

En el lenguaje Arduino, como en muchos otros lenguajes populares, hay un par de maneras de indicar comentarios. Los comentarios de declaraciones múltiples comienzan con /* y terminan con */, lo que permite comentar bloques enteros de código. Los comentarios de una sola declaración empiezan con // y terminan cuando pulsamos Intro para crear una nueva declaración. A veces los comentarios de una sola declaración están al final de una declaración de código Arduino. Cualquier cosa escrita después de la doble barra (//) se ignorará hasta la siguiente declaración.

Como puedes ver en la figura 4.31, la sección superior del sketch LEA4_Blink muestra los comentarios al principio del sketch.

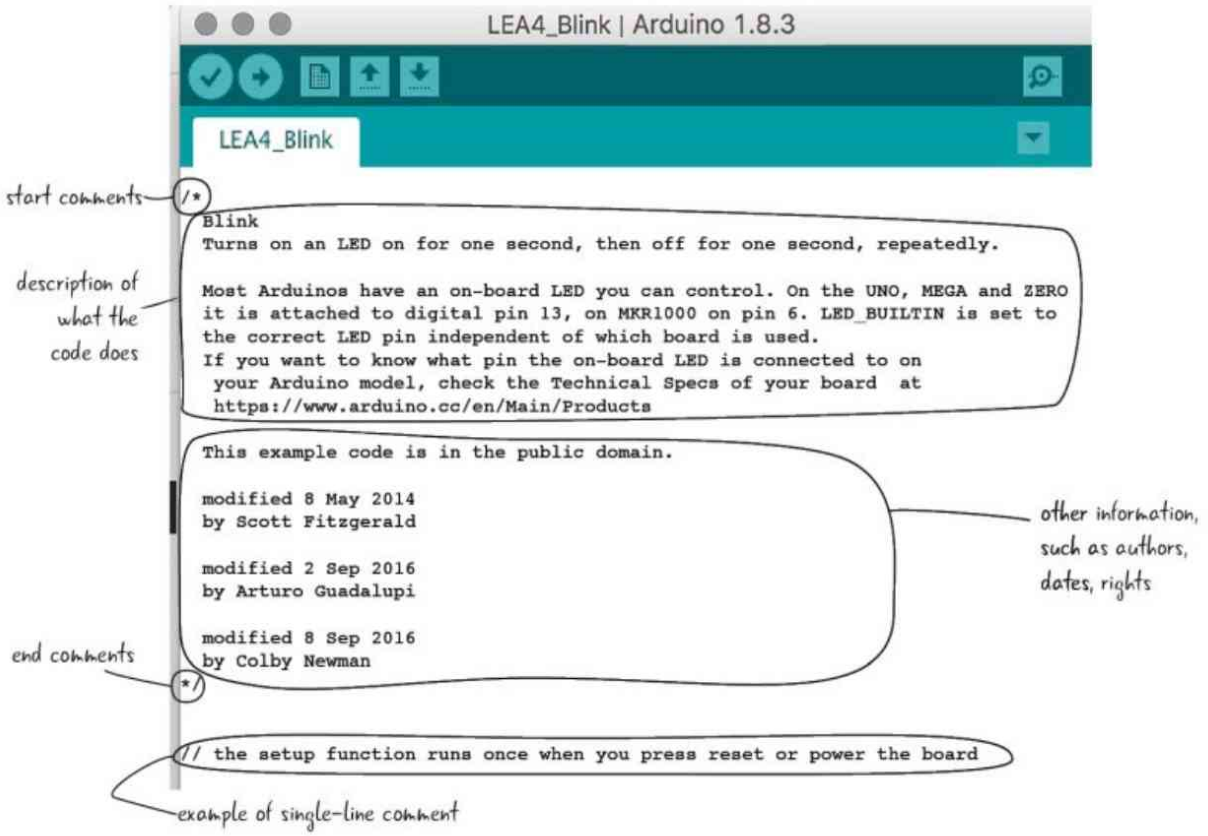


FIGURA 4.31 Comentarios al principio del sketch LEA4_Blink.

Note `/*` y `*/` indican el principio y el final del bloque de un comentario. // indica comentarios de una sola declaración.

setup() y loop(): Las tripas del código

Los comentarios, aunque importantes, no son instrucciones para Arduino. En un sketch de Arduino hay dos secciones básicas: la función setup() y la función loop(). El diagrama de la Figura 4.32 muestra cómo trabajan setup() y loop(): setup() se ejecuta una vez, seguido de loop(), que se ejecuta muchas veces.

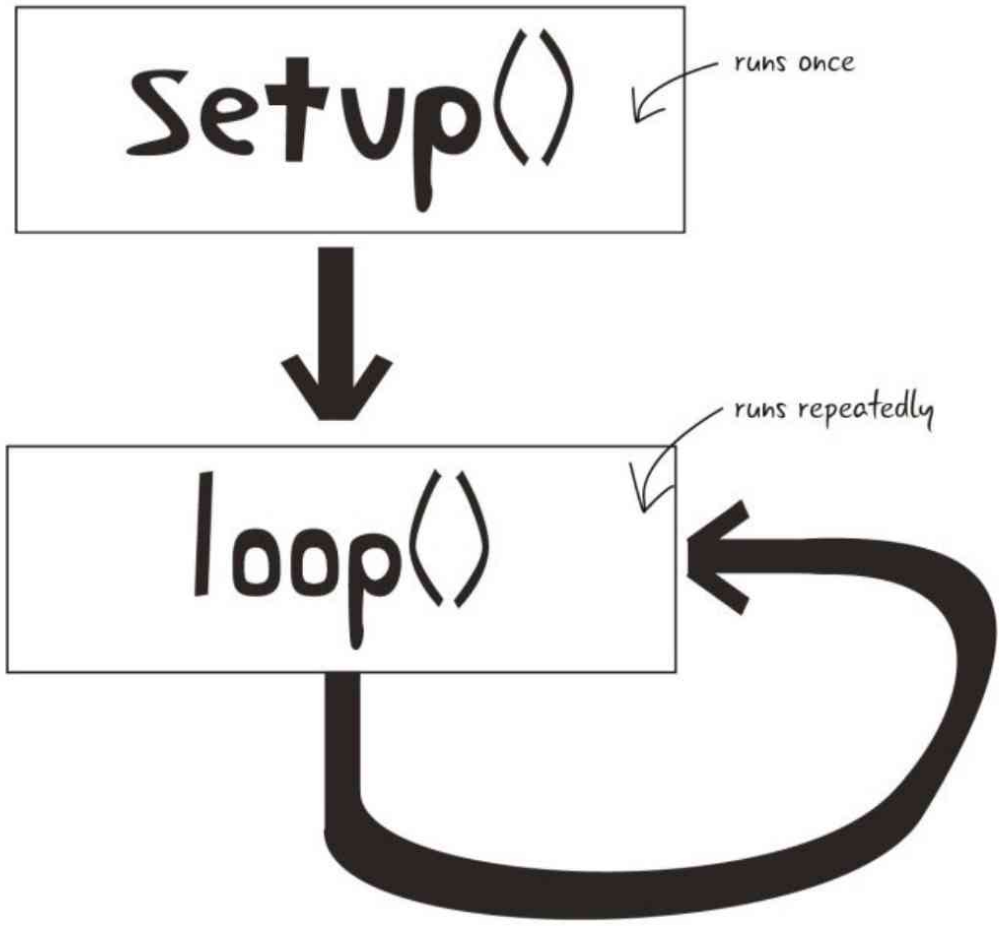


FIGURA 4.32 Diagramas de `setup()` y `loop()`.

setup() es el nombre de una función que se incluye en cada sketch de Arduino. ¿Qué es una función? Podemos decir que es una forma de organizar código o instrucciones para el ordenador.

> **Note** Una *función* es una forma de agrupar expresiones de código o bloques de instrucciones para el ordenador.

En general, cualquier cosa que quieras que suceda solo una vez en el sketch pertenece a la función setup(). setup() se ejecuta exactamente una vez cada vez que reinicias el Arduino.

setup() Y loop() RELACIONADOS

Vamos a echar un vistazo al resto del código Arduino para el sketch LEA4_Blink, pero primero veamos un par de ejemplos de proyectos. Usaremos estos proyectos como ejemplos para entender la diferencia entre las secciones setup() y loop() del código.

EJEMPLOS: ¿CÓMO USAR *SETUP()* Y *LOOP()* EN LOS PROYECTOS?

Ejemplo 1

Más adelante en este capítulo, montarás un led que al apagarse y encenderse continuamente, proporcionará una señal luminosa que sigue la secuencia de la señal de SOS, y controlarás el tiempo de las señales por programación.

En `setup()` configurarás como salida el pin que controla el led, le dirás a Arduino qué pin controlará el led.

En `loop()` escribirás el código que controla la temporización, harás que el led se encienda y se apague continuamente.

Ejemplo 2

Supongamos que deseas crear una caja de música digital que reproduzca diferentes sonidos dependiendo del botón que pulses y además quieres incluir un mando de control de volumen.

Utilizarás `setup()` para asignar diferentes botones a cada uno de los sonidos y determinar qué pin responde al mando de volumen.

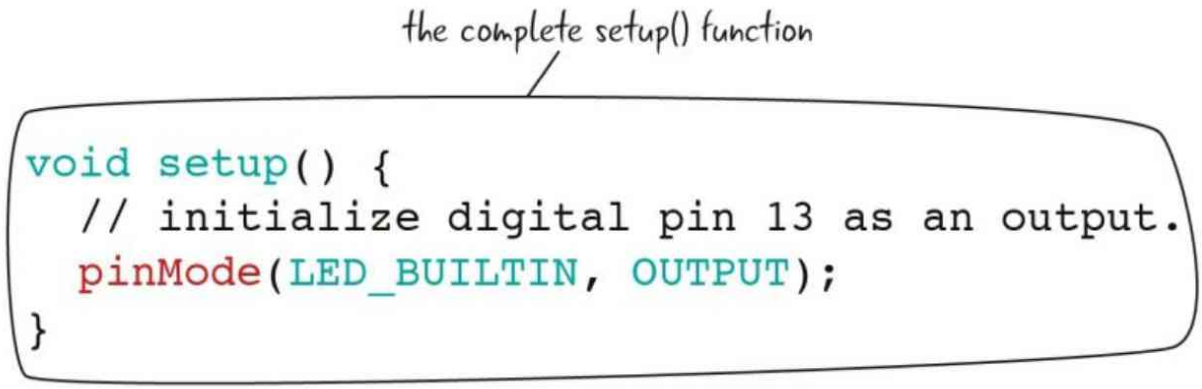
`loop()` se centrará en responder a las pulsaciones de los botones y reproducir cada sonido cuando se activa el botón correspondiente. La función `loop()` supervisará también cambios en el mando con el que cambiarás el volumen.

Has visto ejemplos de cómo funciona setup () y loop (); ahora veamos cómo es la función setup () en el sketch LEA4_Blink que acabas de subir y que se ejecuta en el equipo Arduino.

setup(): configuración de las Condiciones iniciales

Hemos hablado de los comentarios, y has visto que setup() se ejecuta una vez al principio, cuando enciendes o reinicias el Arduino. Veamos ahora en

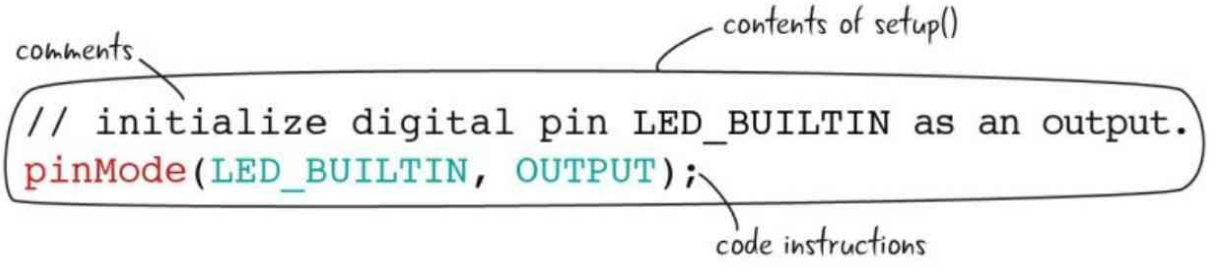
profundidad la función setup() en el sketch de ejemplo LEA4_Blink.



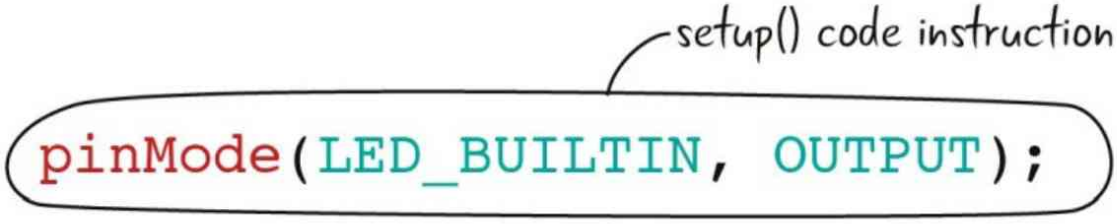
setup() lleva paréntesis; más tarde explicaremos por qué se necesitan y qué hacen. Después de los paréntesis se abre una llave, {, que juega un importante papel. Las llaves contienen un bloque de código y marcan las instrucciones que se llevarán a cabo cuando se ejecute el código. En este caso, siempre que se ejecute setup(), todas las instrucciones se ejecutarán una a una. Cuando hayas terminado con el bloque de código, cierra con una llave, }, para decirle al Arduino que has terminado el bloque de esas instrucciones de código.

Note **Las *llaves* indican el inicio y fin de un bloque de código.**

Echemos un vistazo a las instrucciones de código que se llevan a cabo cuando se ejecuta setup (). Para el código del sketch LEA4_Blink, solo hay una instrucción de código setup () y un comentario de una sola línea.



La primera línea resulta familiar. Comienza con dos barras hacia delante, lo que significa que es un comentario. En este caso, el comentario nos dice que el propósito de la segunda línea es "configurar el pin digital LED_BUILTIN como salida". Todavía no entiendes lo que esto significa, pero ahora sabes que pinMode(LED_BUILTIN, OUTPUT); configura el led integrado como una salida. Echemos un vistazo a la línea de instrucciones de código sin el comentario.



Note Una línea de código está formada por una y solo una instrucción y termina con un punto y coma.

El punto y coma en el código sirve para el mismo propósito que los puntos en español; indica que has llegado al final de la línea. Esto evita que el Arduino malinterprete las instrucciones, porque en cuanto ve el punto y coma, sabe que quieres terminar la línea. Si omites el punto y coma, producirás un error en el IDE y el código no se cargará en el Arduino.

```
pinMode(LED_BUILTIN, OUTPUT);
```
semicolon

Note El punto y coma termina las frases de código, igual que los puntos terminan las oraciones.

A continuación, veamos el final de la línea. pinMode() va seguida de un paréntesis, que contiene el texto LED_BUILTIN, una coma, y la palabra OUTPUT, todo con letras mayúsculas. pinMode() es una función que hace que los pines se comporten de una manera particular.

Note Cuando queremos usar una función o instrucción como `pinMode()`, decimos que se está "llamando" a la función.

```
pinMode(LED_BUILTIN, OUTPUT);
```
set pin mode

Cuando llamas a pinMode(), le dices a Arduino que asigne al pin el número que escribes para que actúe como una entrada o una salida. En este caso, en lugar de ver el número del pin, ves LED_BUILTIN, que está ahí en lugar del número del pin porque Arduino Uno sabe que LED_BUILTIN es

como decir “pin 13”. Así que 13 es el número del pin que configurarás como OUTPUT. Todavía no has instalado ningún componente en el equipo Arduino, pero el Arduino ya tiene un diminuto led incorporado en la placa acoplado al pin 13 que es el origen del término LED_BUILTIN. Estás configurando el modo para el pin 13, diciendo al Arduino que planeas usar el pin 13 como salida.

> **Note** `LED_BUILTIN` **está conectado al pin 13 en el equipo Arduino. Ambos,** `pinMode(13, OUTPUT)` **y** `pinMode(LED_BUILTIN, OUTPUT)` **tienen la misma salida.**

Mediante OUTPUT controlas cuándo el pin tiene que recibir tensión o no. OUTPUT te permite configurar el pin de forma dinámica y cambiar su estado a medida que se ejecuta el sketch.

setup(): SE EJECUTA UNA VEZ

Para resumir, en nuestro sketch LEA4_Blink setup() dile a Arduino que trate al pin digital número 13 como salida. El Arduino recuerda bien las instrucciones que le ordenaste en relación con los pines, así que necesitas decírselo solo una vez. Mientras este sketch siga funcionando, Arduino sabe que el Pin 13 es una salida. Si el Arduino se desenchufa o apaga de alguna manera, lo primero que sucede cuando se reinicia (dentro de la función setup()) es que el pin 13 está configurado como salida. En otras palabras, no necesitas recordar una y otra vez al Arduino cómo se deben comportar cada uno de los

pines. Pondrás todas las funciones pinMode() dentro de setup() para que se ejecuten solo una vez. La figura 4.33 muestra el led parpadeando.

> **Note** `setup()` **se ejecuta una y solo una vez.**

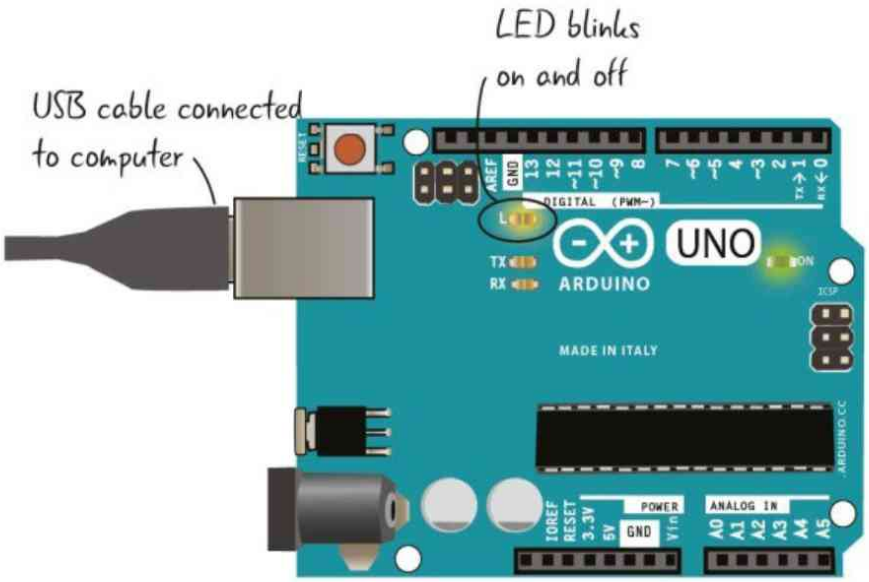


FIGURA 4.33 El led parpadea.

¿PREGUNTAS?

P: ¿Por qué debería preocuparme por poner comentarios en el código?

R: A veces cuando escribes código, no es obvio lo que este hace exactamente, y es muy útil dejarte notas para cuando vuelvas al sketch más tarde. También es útil cuando compartes el código o cuando trabajas en equipo.

P: Va `setup()` siempre en primer lugar?

R: Arduino sabe que siempre ejecuta `setup()` en primer lugar, una vez, y luego continúa ejecutando la sección del código `loop()`. Para tener un sketch que funcione correctamente, el código debe incluir un bloque `setup()`.

P: ¿Pueden explicarnos de nuevo qué significa que establezcamos un pin utilizando `pinMode()`?

R: Mediante el uso de `pinMode()`, estamos indicando al equipo Arduino que pensamos utilizar un pin concreto, en este caso el número 13 (etiquetado también como `LED_BUILTIN`), en el sketch. Esta información es necesaria para que el Arduino sepa qué pines tiene que controlar en cada sketch.

P: ¿Siempre tengo que configurar los pines como salidas?

R: No, solo debes configurar los pines como salidas cuando desees utilizarlos para encender y apagar cosas. Los pines también pueden ser declarados como entradas. Veremos las entradas en el próximo capítulo.

P: ¿Está siempre `pinMode()` conectado a `LED_BUILTIN`?

R: No, hay muchos otros pines en el Arduino que puedes usar para el sketch. Ahora utilizas `LED_BUILTIN` (pin 13) porque es el único pin que tiene un led convenientemente conectado a él.

P: ¿Importa cuáles son los pines de declaro?

R: Solo debes declarar los pines que piensas usar en el sketch. En el sketch LEA4_Blink, solo declaras el `LED_BUILTIN`, pin 13, porque sabes que vas a usar ese pin para encender y apagar el led.

P: Configurar el modo del pin como salida no es lo único que haré en `setup()`, ¿verdad?

R: Cierto. Hay muchas otras instrucciones del Arduino que necesitarás ejecutar solo una vez, y las pondrás en `setup()`. Las explicaremos más tarde.

análisis de loop(): lo QUÉ se repite UNA Y OTRA VEZ

Ahora que has estudiado la función setup() del sketch LEA4_Blink, echemos un vistazo a la función loop().

loop() function from the LEA4_Blink sketch

```
void loop() {
digitalWrite(LED_BUILTIN, HIGH);//turn the LED on (HIGH is the voltage level)
delay(1000);                    //wait for a second
digitalWrite(LED_BUILTIN, LOW);//turn the LED off by making the voltage LOW
delay(1000);                    //wait for a second
}
```

La función loop() contiene el código que deseas que se repita una y otra vez. Mientras el Arduino siga funcionando, este código seguirá repitiéndose, después de que el código en setup() se haya ejecutado una vez.

> **Note** `loop()` **continuará ejecutándose mientras el Arduino esté encendido.**

Cuando viste cómo LEA4_Blink se ejecutaba en el Arduino, el led parpadeaba, encendiéndose y apagándose cada segundo. El código contenido en el bucle produce este comportamiento. Echemos un vistazo de cerca, línea por línea, a lo que está en loop() en el sketch.

```
digitalWrite(LED_BUILTIN, HIGH);// turn the LED on (HIGH is the voltage level)
```

first statement in this loop()

> **Note** **En un sketch de Arduino, "write" (escribir) significa establecer un valor.** `digitalWrite()` **establecerá un valor a** `HIGH`**, o a** `LOW`**.**

La primera instrucción de código dentro de loop() es similar a la

declaración pinMode(LED_BUILTIN, OUTPUT); que viste en setup(). De nuevo estarás tratando con LED_BUILTIN, que es una etiqueta para el pin 13, ya que declaraste en el setup() que el sketch utiliza este pin. La función digitalWrite() en este contexto se usa para indicar si el pin está encendido o apagado. Cuando escribes, o configuras el valor del pin a HIGH, estás haciendo que el pin se active.

Note `digitalWrite(pin #, HIGH)` **hará que el** `pin #` **se active.**

Cuando el Arduino llegue a esta línea en loop(), encenderá el led conectado al pin 13. Veamos la segunda línea.

EXAMEN DE loop(): digitalWrite() Y delay()

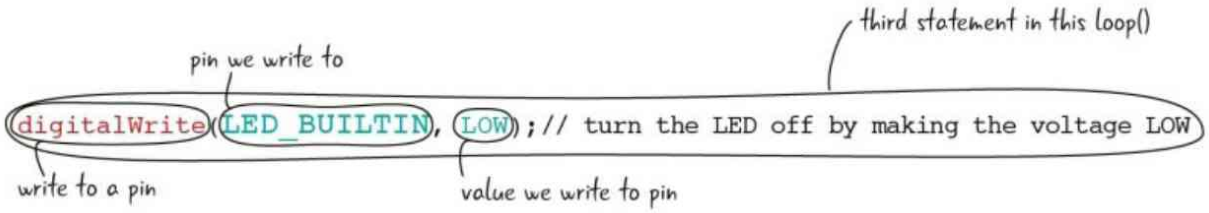
Después de activar el pin, quieres programar un breve delay() en el programa. Este delay() detendrá el programa, impidiendo que el Arduino lea la declaración que sigue durante un corto espacio de tiempo. ¿Cuánto tiempo delay() pausa el programa? Depende de ti. delay() necesita que incluyas en el paréntesis el número de milisegundos de espera (1 000 milisegundos son un segundo). En este caso, has declarado que el Arduino esperará durante 1 000 milisegundos, o un segundo, antes de pasar a la siguiente sentencia del programa.

```
delay(1000);                    // wait for a second
```

second line in this loop()

Note La función `delay()` detiene el Arduino para que este no haga nada durante un corto periodo de tiempo.

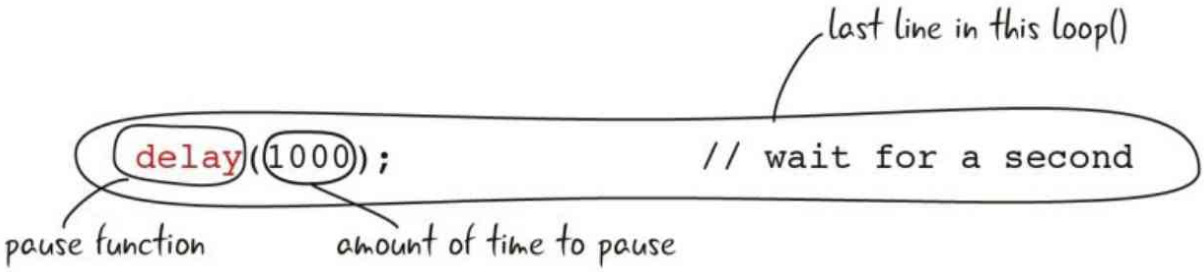
Veamos ahora la tercera línea de la función loop().



Esta tercera línea de instrucción de código dentro de la secciónloop() es casi idéntica a la primera línea, digitalWrite(LED_BUILTIN, HIGH);, excepto que HIGH se ha sustituido por LOW. Todavía estás centrado en el pin 13, que es el único pin que usas en este sketch. Como aprendiste con la primera línea, digitalWrite() determina si el pin está activado o desactivado. Si utilizas Arduino para escribir LOW desactiva el pin, en otras palabras, apaga el led.

Note `digitalWrite(pin #, LOW)` **apagará el** `pin #`.

Finalmente, veamos la cuarta y última línea de loop(). Introducirás otra pausa en el programa, esta vez por valor de 1 000 milisegundos, o un segundo. Esto hace que el led permanezca apagado durante un segundo, ya que el Arduino se detiene durante este tiempo. El Arduino hace una pausa de un segundo y luego regresa a la primera línea de loop() del código, repitiendo el ciclo que acabamos de describir.



¿PREGUNTAS?

P: Puedo cambiar el tiempo que dura el retardo, ¿verdad?

R: Absolutamente. Verás cómo hacer las pausas más largas y más cortas, además de cómo hacer otras modificaciones en el código de `loop()`, más tarde en este capítulo.

loop(): EXAMEN DE LA FUNCIÓN CompletA loop()

Aquí está de nuevo el código completo de loop() otra vez, incluyendo los comentarios:

loop() function from the LEAS_Blink sketch

```
void loop() {
digitalWrite(LED_BUILTIN, HIGH);//turn the LED on (HIGH is the voltage level)
delay(1000);                    //wait for a second
digitalWrite(LED_BUILTIN, LOW);//turn the LED off by making the voltage LOW
delay(1000);                    //wait for a second
}
```

Otra vez, como se muestra en la figura 4.34, loop() se ejecuta continuamente, y setup() se ejecuta una vez. Nuestro código de loop() hará parpadear la luz, que se encenderá y apagará hasta que el Arduino se desconecte.

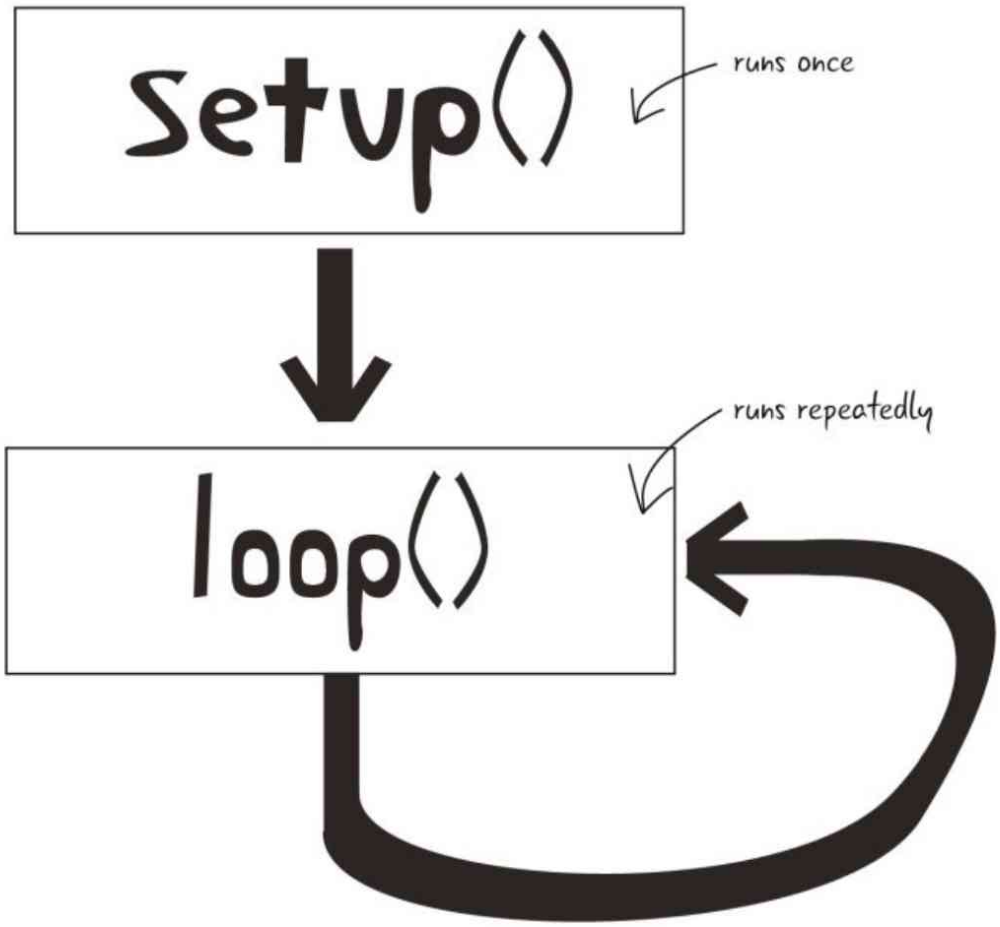


FIGURA 4.34 `setup()` y `loop()`.

> **Note** La configuración de los modos de pines se realiza siempre en `setup()`. Cualquier cosa que desees ejecutar más de una vez debe incluirse en la función `loop()`.

Aunque los sketches que escribirás a lo largo del libro serán cada vez más complicados e incluirán más líneas de código, los fundamentos expuestos en el sketch LEA4_Blink continuarán siendo la base para elaborar un buen código. Los pines solo se necesitan declarar como entradas o salidas en setup(), y cualquier código que necesites que aparezca más de una vez debe incluirse en loop(). Recordar estos dos principios te ayudará enormemente a medida que entres en proyectos más complejos.

Has visto lo básico de un sketch de Arduino, y también lo que hacen las funciones setup() y loop(). Después de contestar algunas preguntas pasaremos a revisar los esquemas y veremos el esquema del Arduino. A continuación explicaremos cómo conectar el Arduino a la placa de pruebas para que puedas ejecutar LEA4_Blink y encender un led en la placa de pruebas.

¿PREGUNTAS?

P: Cualquier cosa que ponga en `loop()`, ¿seguirá repitiéndose una y otra vez?

R: Correcto. Como indica su nombre, `loop()` sigue en bucle recorriendo las mismas líneas de código, una y otra vez.

P: ¿Qué hace la función `delay()`?

R: La función `delay()` especifica la cantidad de tiempo que el Arduino está en pausa, o en espera inactiva. Durante este tiempo todo se mantiene igual, así que si la luz está encendida permanece encendida. Con `delay()` en este sketch, se puede ver claramente que la luz se enciende durante un segundo y luego se apaga durante otro segundo.

P: ¿Estará siempre `digitalWrite()` en `loop()`?

R: No, solo lo estará cuando quieras configurar un pin y lo que sea que esté unido a ese pin, `HIGH` o `LOW`. En este caso, utilizas `digitalWrite()` para encender o apagar el led.

P: ¿Qué es exactamente una función?

R: Por ahora, piensa en una función como una forma de organizar instrucciones para el Arduino. Explicaremos más sobre ellas mientras que desarrollas más sketches.

P: Punto y coma, llaves... parece que hay mucha puntuación en el sketch. ¿Cómo la recordaré?

R: Puede ser confuso al principio. Tienes que estudiar los ejemplos y ver cómo se emplea la puntuación. Las llaves marcan un bloque de código, y el punto y coma marca el final de una línea.

P: ¿El lenguaje de programación Arduino tiene una guía de referencia en línea?

R: Sí. arduino.cc/en/Reference/HomePage es un gran sitio para obtener más información sobre este lenguaje. Puedes utilizarlo para averiguar más sobre el código que empleas en este libro e investigar sobre tus proyectos después de terminar el libro.

ESQUema DEl Arduino

Ahora que ya has ejecutado el código y encendido el led en la placa del Arduino, vas a conectar el Arduino a una placa de pruebas, montar un circuito y ejecutar el sketch LEA4_Blink de nuevo. Quieres aprender a controlar componentes externos con el Arduino, no solo a encender un led en el mismo Arduino, por lo que debes utilizar una placa de pruebas para instalar los componentes.

Para ejecutar LEA4_Blink en el Arduino conectado a una placa de pruebas, no necesitarás hacer cambios en el código. Cuando en LEA4_Blink se configura LED_BUILTIN a HIGH, se enciende el led situado cerca del pin 13 en el Arduino, y lo mismo le ocurrirá a cualquier cosa que se conecte al pin 13 (un led) configurado a HIGH (o activado) a través de una placa de pruebas.

Antes de empezar a montar el circuito, echemos un vistazo al esquema. Hacerlo así te ayuda a visualizar las conexiones electrónicas en el circuito.

Los esquemas a partir de ahora incluirán un símbolo que representa al Arduino. La figura 4.35 muestra el esquema del Arduino, con todos los pines digitales, analógicos, y alimentación y tierra etiquetados con sus números o funciones, colocados junto a un dibujo del Arduino Uno para poder compararlos. Ahora no es necesario memorizar los números de los pines ni su funcionalidad, luego explicaremos con más detalle los pines y sus usos.

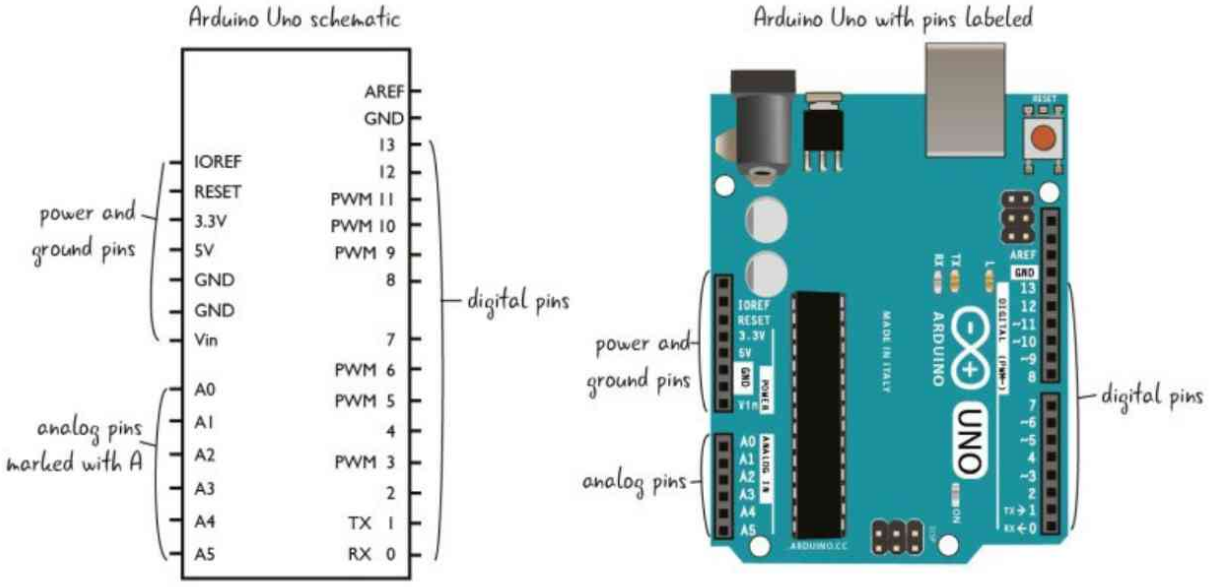


FIGURA 4.35 Esquema y placa de Arduino.

El esquema del Arduino parece mucho más complicado que los otros esquemas que hemos visto anteriormente. Su complejidad refleja el número de conexiones posibles del hardware del equipo. En lugar de tratar de introducir este detallado esquema de Arduino en el esquema de cada circuito que montes, utilizarás una versión simplificada: un rectángulo que representa el Arduino, con etiquetas solamente para los componentes que vas a utilizar en ese circuito. Veamos un esquema completo del circuito que vas a montar con el Arduino.

ESQUema DEL CircuitO

Para mayor claridad, cuando incluyas el Arduino en los esquemas solo vas a etiquetar los pines relacionados con el circuito que estás montando. Por ejemplo, la figura 4.36 muestra el esquema del circuito que vas a montar. Solamente se muestran el pin de 13, el de 5 voltios y el de tierra, así como el led y la resistencia.

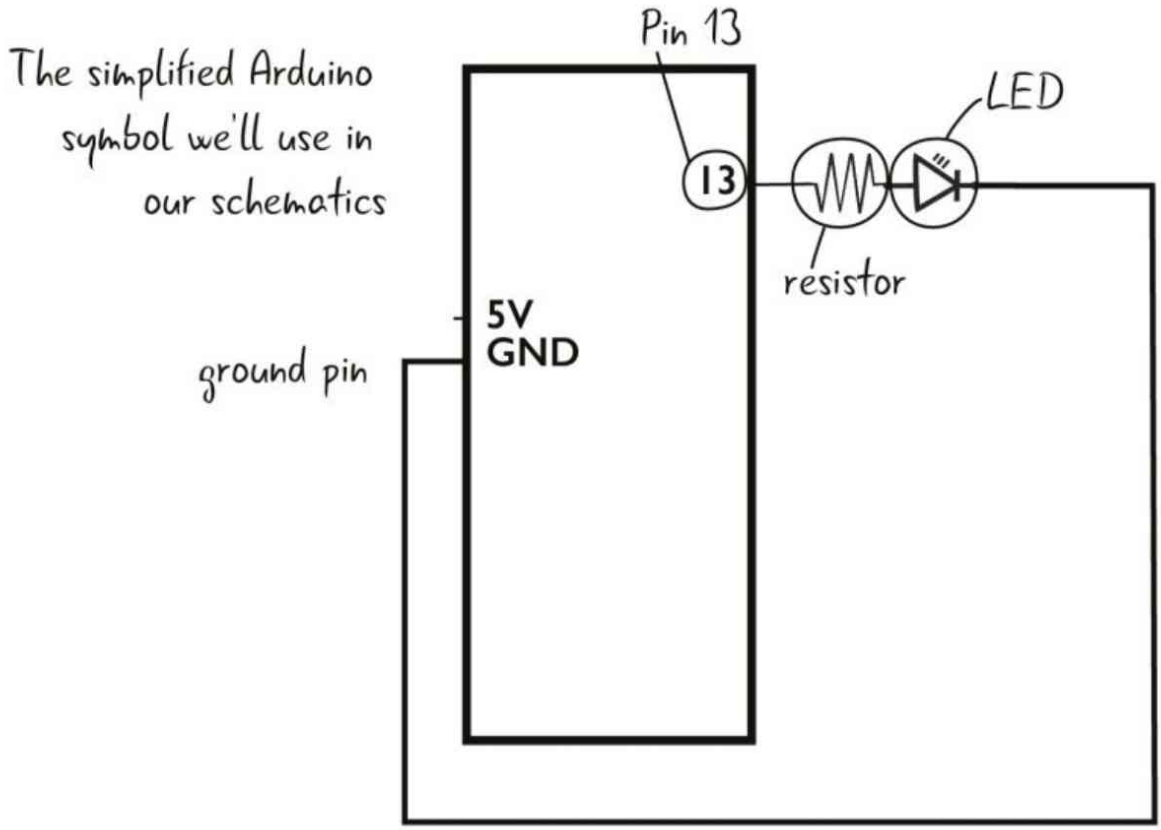


FIGURA 4.36 Esquema del circuito de LEA4_Blink.

Ahora que te has familiarizado con el esquema, veamos cómo vas a montar el circuito. Partirás del circuito que montaste en el capítulo 3 (figura 4.37).

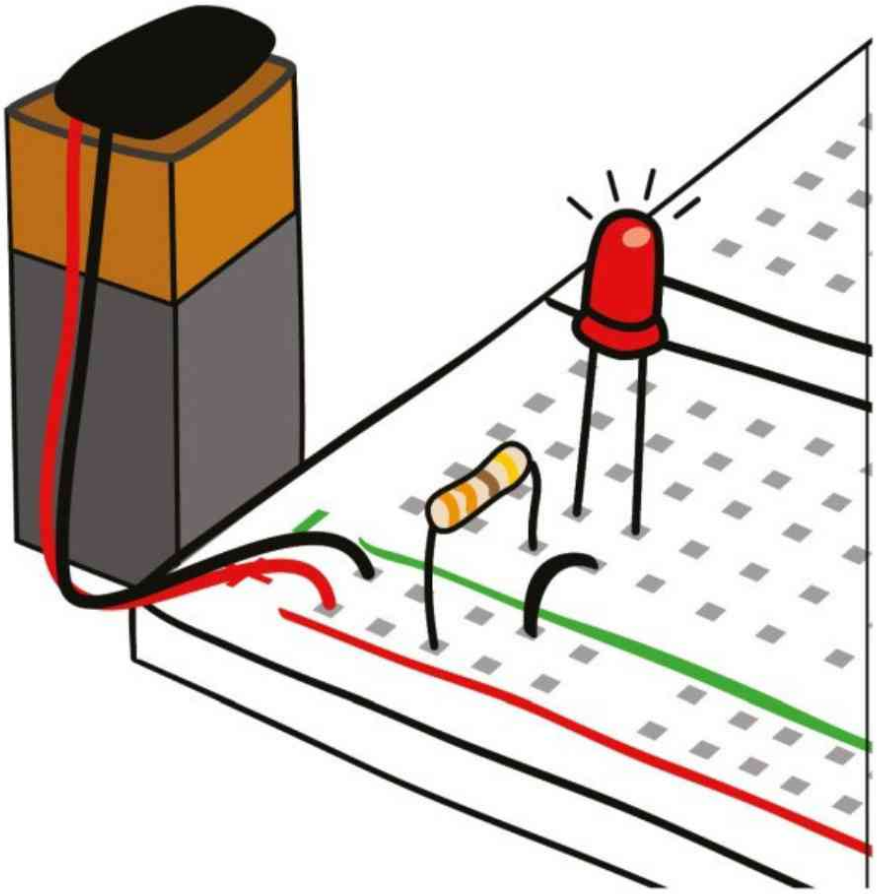


FIGURA 4.37 Circuito del capítulo 3.

montaje del Circuito básico

Ahora que has echado un vistazo al esquema, vas a montar el circuito. Vas a ejecutar el sketch LEA4_Blink y a encender un led en la placa de pruebas. Vas a conectar al Arduino una placa de pruebas con una resistencia y un led; el programa que se ejecuta en el Arduino no cambiará. El Arduino será la fuente de alimentación para el circuito cuando este se conecte a un ordenador con un cable USB.

> **Warning** Recuerda, siempre que hagas cambios en un circuito, el Arduino *no* debería estar conectado al ordenador.

Necesitarás los siguientes componentes:

- LED (rojo)
- resistencia de 220 ohmios (rojo, rojo, marrón, dorado). Es diferente a la que empleaste en el capítulo anterior.
- Cables para hacer puentes
- Placa de pruebas
- Arduino Uno
- Cable USB tipo A-B
- Ordenador con el IDE de Arduino

La figura 4.38 compara un dibujo de este proyecto con un esquema del circuito completo. Como puedes ver, el circuito utiliza una resistencia y un led como el que montaste en el capítulo 3.

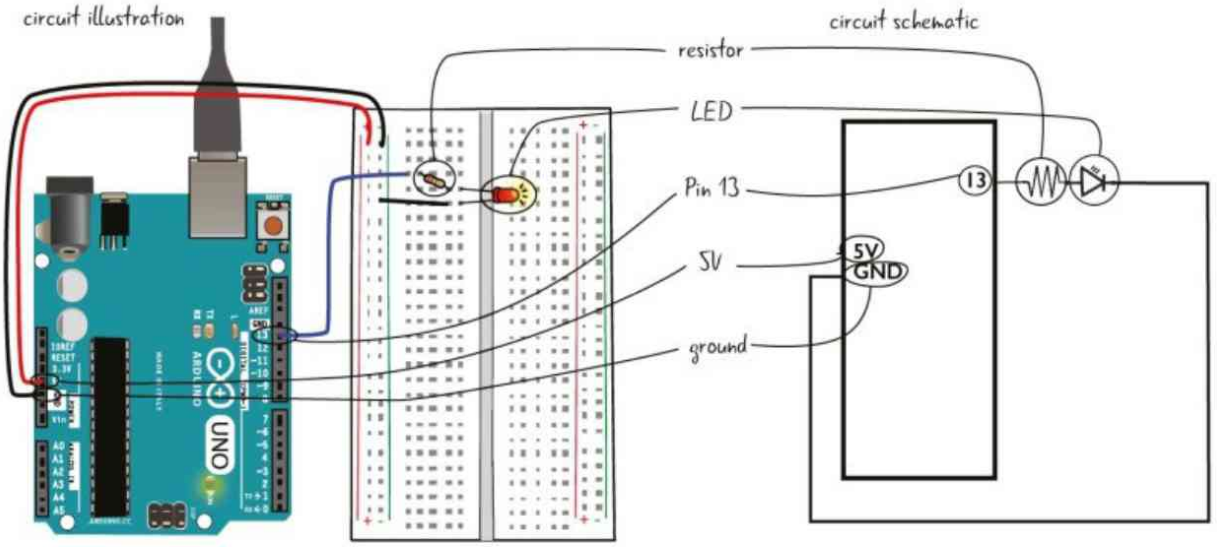


FIGURA 4.38 Placa de pruebas de Arduino con etiquetas y esquema comentado.

ConEXIÓN DE Arduino A La PLACA dE PRUEBAS: PRIMEROS PASOS

Quieres montar circuitos con el Arduino, no solo encender un led en la placa Arduino, así que lo vas a conectar a una placa de pruebas. ¿Cómo se hace esto?

Ya hicimos referencia al uso de los pines de alimentación y de tierra en el Arduino en el Capítulo 2. Estos dos pines te permiten usar la electricidad del Arduino para alimentar los componentes del circuito, sustituyendo la batería de 9 voltios que utilizaste antes.

Para usar los pines, empieza instalando un puente desde el pin marcado con 5 V a uno de los buses de alimentación de la placa de prueba. A continuación, haz un puente entre uno de los pines marcados con GND (que significa "tierra") a uno de los buses de tierra de la placa de pruebas, como se muestra en la figura 4.39.

Warning Tienes que asegurarte de que el ordenador *no* está conectado al Arduino cuando estás montando el circuito.

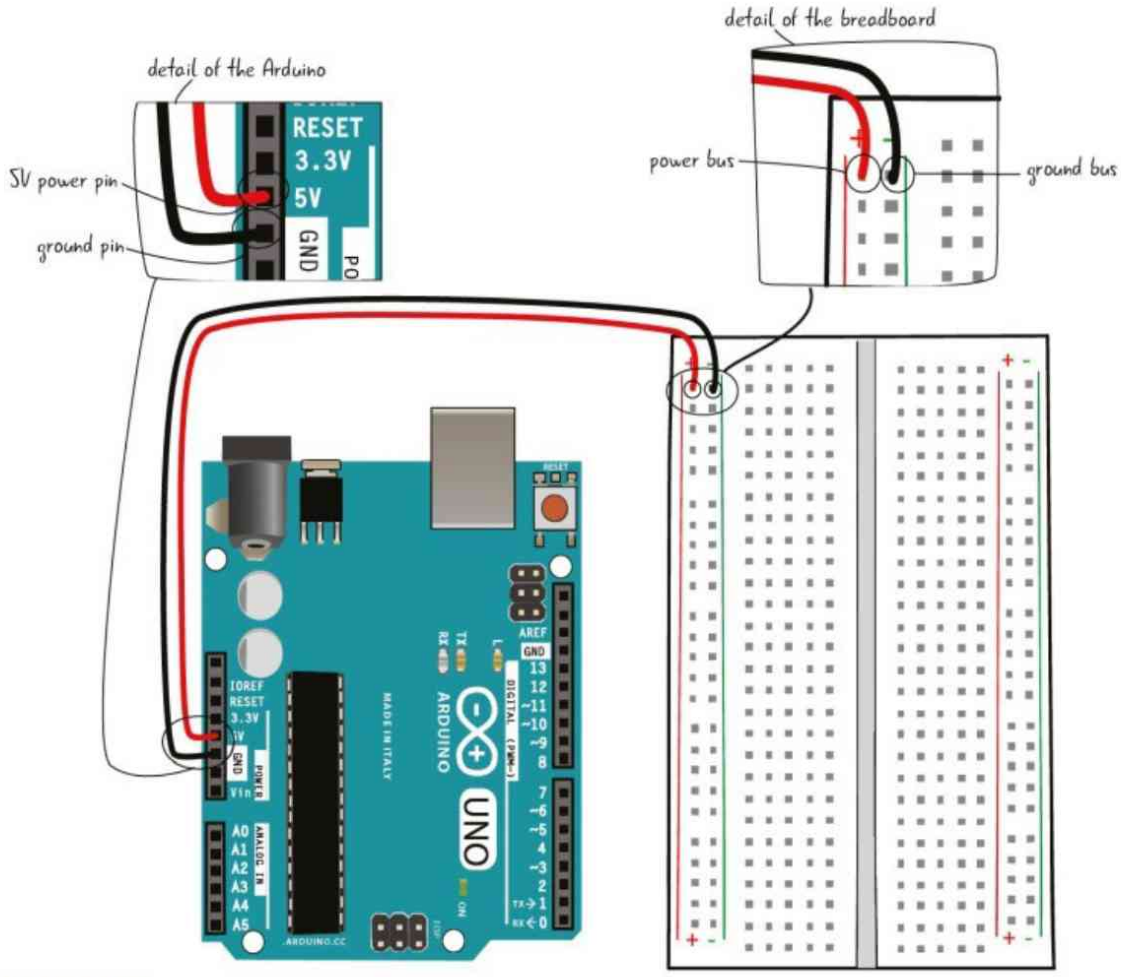


FIGURA 4.39 Conexiones de alimentación y tierra a la placa de pruebas.

Es el procedimiento estándar para conectar la alimentación (5 V) y la tierra (GND) a la placa de pruebas cuando la conectas al Arduino. Incluso aunque no necesites la fuente de energía en ese momento, puede ser útil disponer de ella posteriormente, a medida que instalas más componentes en el circuito. En este circuito, en lugar de usar el pin de 5 V como alimentación, emplearás el pin 13 para suministrar energía al led.

MONTAJE DEL CircuitO PASO A PASO: Conexión del Pin y

la Resistencia

Ahora que el Arduino y la placa de prueba están interconectados, conecta el pin 13 de la placa del Arduino a una línea de puntos de conexión en la placa de pruebas con un puente, como se ve en la figura 4.40.

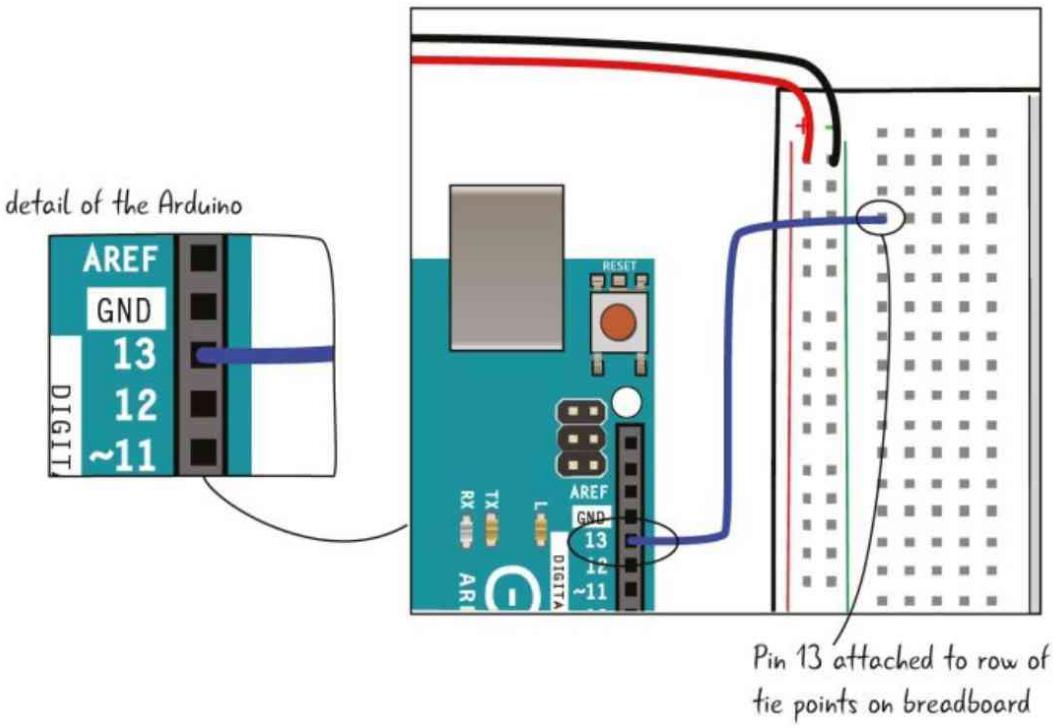


FIGURA 4.40 Instalación de un puente desde el pin hasta la placa de pruebas.

A continuación coloca el extremo de una resistencia de 220 ohmios (con las bandas de colores rojo, rojo, marrón y dorado) en la misma fila de puntos de conexión en la que está el puente del pin 13. Coloca el otro extremo en otra fila de puntos de contacto (figura 4.41).

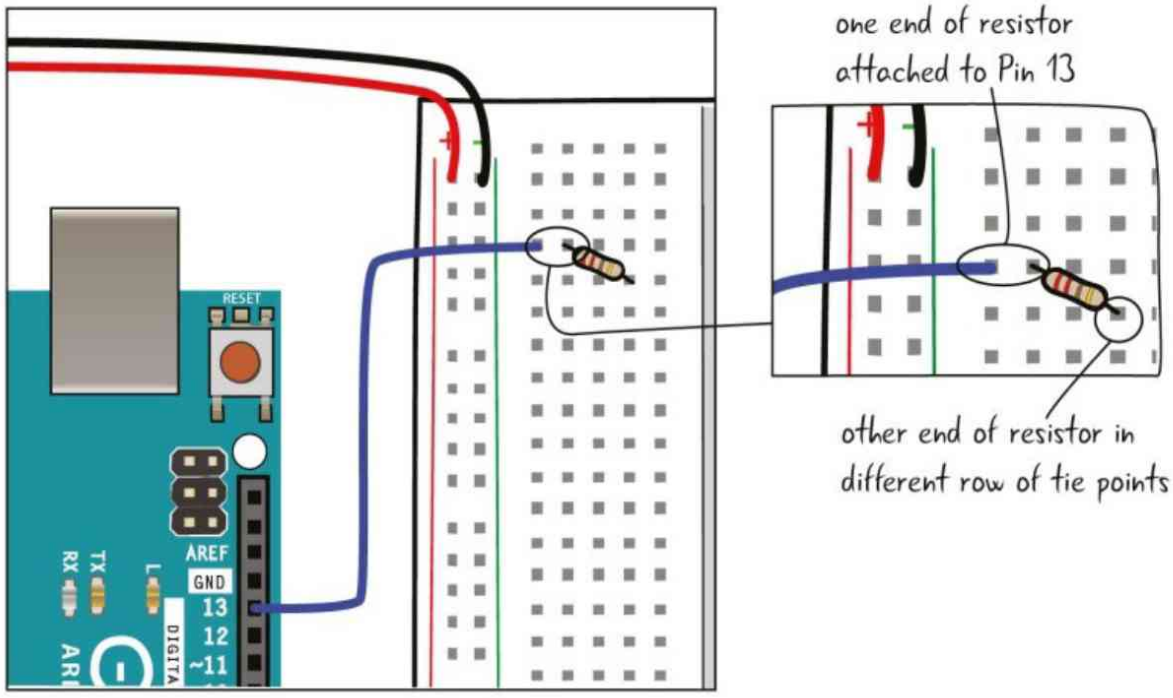


FIGURA 4.41 Inserción de la resistencia en el circuito.

MONTAJE DEL CircuitO PASO A PASO: ConEXIÓN DEL LED

Coloca el ánodo (terminal largo, extremo positivo) del led en la misma fila de puntos de contacto que el otro extremo de la resistencia. Coloca el cátodo (terminal corto, extremo negativo) en otra fila (figura 4.42).

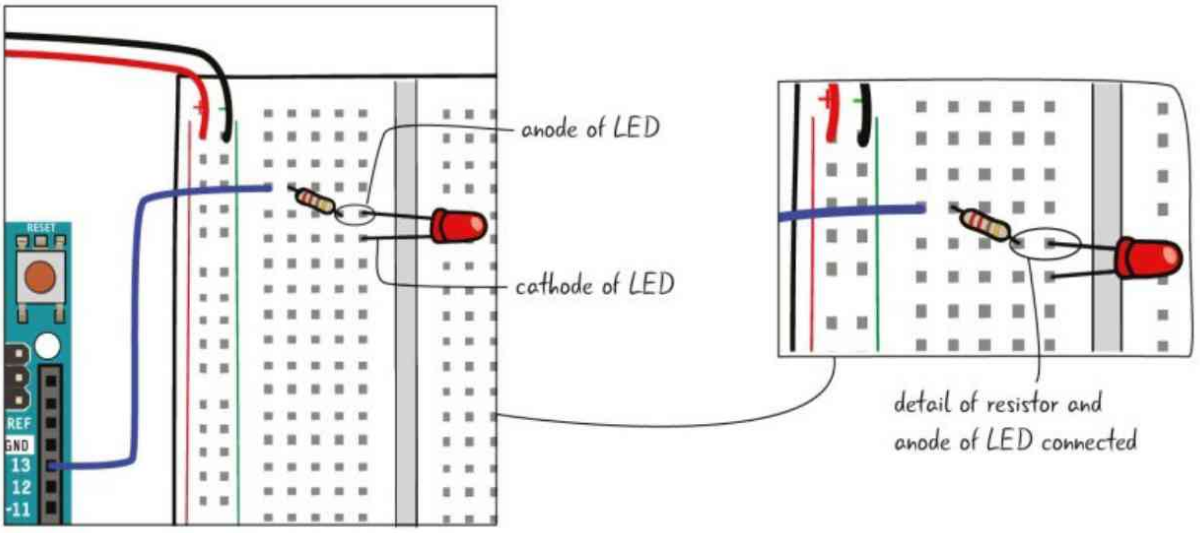


FIGURA 4.42 Inserción del led en el circuito.

A continuación, añade un puente que conecte el cátodo (terminal corto, polo negativo) del led al bus de tierra (figura 4.43).

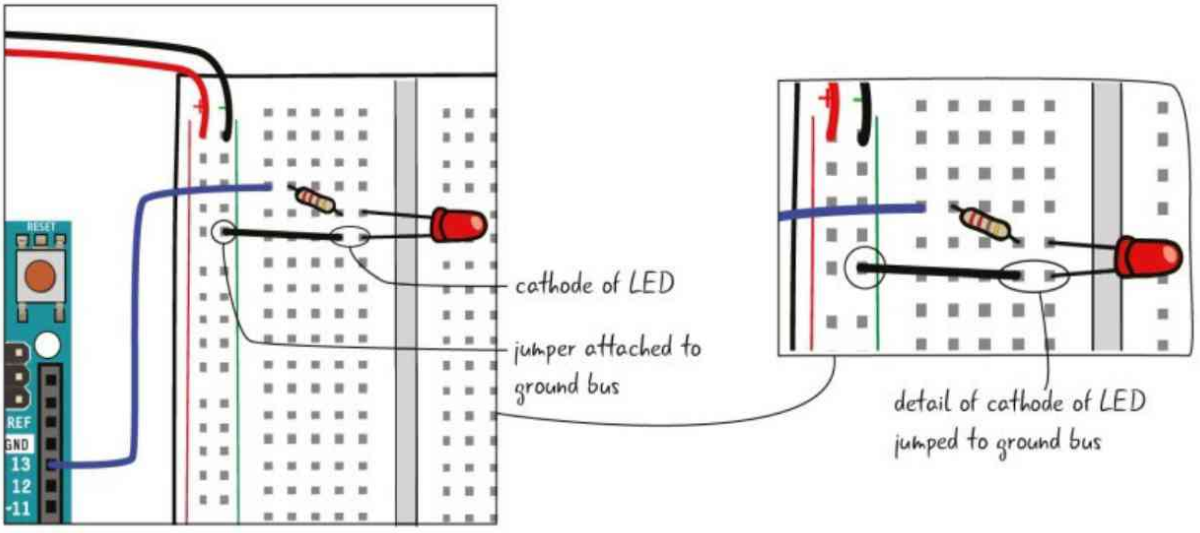


FIGURA 4.43 Inserción de un puente del led a tierra.

MONTAJE DEL CIRCUITO PASO A PASO: CONEXIÓN AL ORDENADOR

Finalmente, conecta el cable USB que está enchufado al ordenador para proporcionar energía al circuito (figura 4.44).

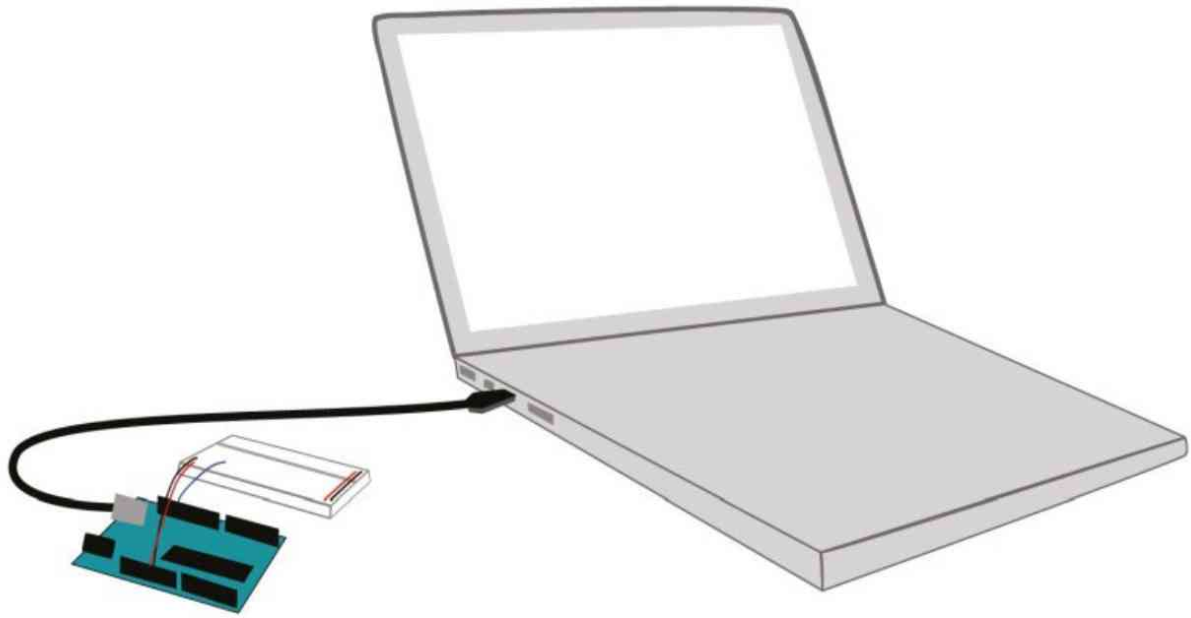


FIGURA 4.44 Conexión del Arduino al ordenador.

El led instalado en la placa de pruebas debería empezar a parpadear (figura 4.45). Este circuito es como el circuito básico que montaste en el capítulo anterior, pero ahora el led parpadea controlado por el equipo Arduino, que ejecuta el sketch LEA4_Blink. Tienes un mayor control sobre el led si usas el Arduino; has añadido la temporización.

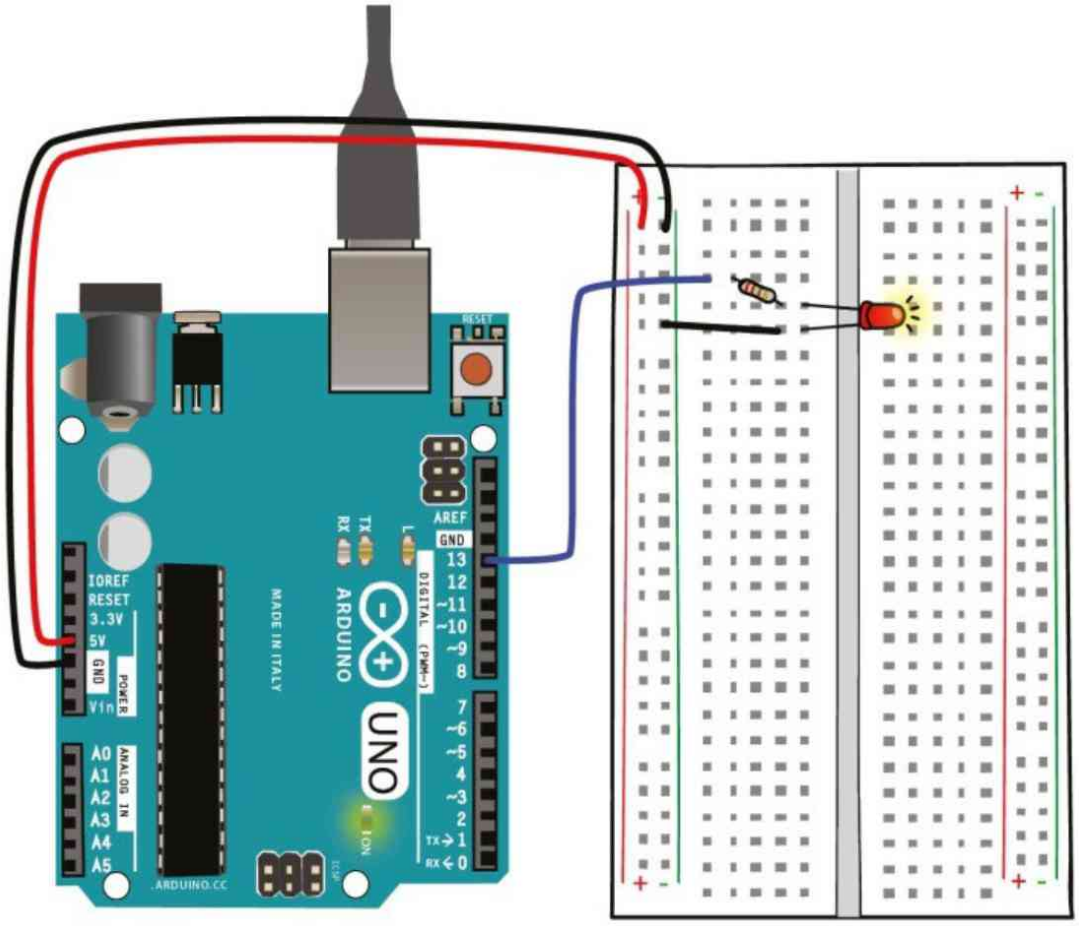


FIGURA 4.45 Led intermitente.

¿PREGUNTAS?

P: El led no se ilumina, ¿qué pasa?

R: ¿Recuerdas la sección del Capítulo 3 sobre como depurar el circuito? Comprueba la continuidad observando cuidadosamente la placa o con un multímetro. Asegúrate de que el led tenga la orientación correcta. También compueba que los puentes hagan buen contacto con la placa de pruebas y con el Arduino.

P: No he cambiado el código en el sketch LEA4_Blink; ¿por qué funciona?

R: El código en el sketch LEA4_Blink controla el `LED_BUILTIN` en el Arduino Uno. El diminuto led que lleva incorporado está conectado al pin 13 de la placa de Arduino, pero el código en el sketch controlará también cualquier componente conectado al pin 13.

P: No me ha quedado claro. ¿Por qué conectamos 5 voltios a la placa de pruebas cuando no parece que la vayamos a utilizar?

R: Es una convención conectar la alimentación y tierra a los buses de alimentación y tierra en una placa de pruebas cuando la montamos. A medida que montes circuitos más complejos, utilizarás el bus de alimentación. Este circuito obtiene la energía del pin del Arduino.

LUZ DE SEÑALIZACIÓN SOS: una temporización más compleja

Aunque el circuito anterior ha sido muy similar al del proyecto del capítulo 3, has logrado algo conectándolo al Arduino: descubrir las posibilidades del código. Anteriormente en este capítulo has visto que puedes hacer que la luz parpadee con solo unas simples líneas de código, y la oportunidad de complicarlo crece a partir de este punto de partida.

Ahora vas a trabajar en el código, lo ajustarás para crear una señal luminosa SOS, una luz que utiliza el código morse para transmitir un mensaje SOS mediante un patrón de luz intermitente. Es un patrón de tres destellos breves de luz, seguidos de tres destellos largos, y por último tres destellos breves con una pausa larga al final, antes de repetir el patrón nuevamente (figura 4.46).

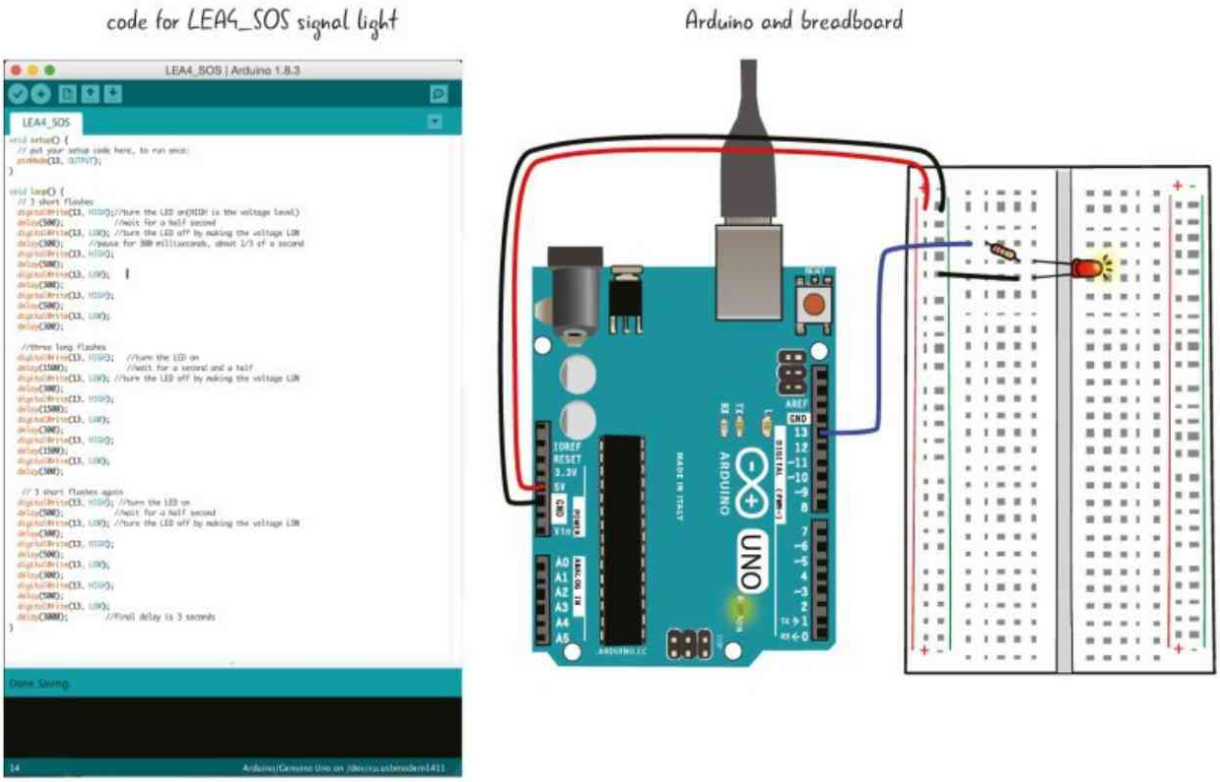


FIGURA 4.46 Sketch y circuito para LEA4_SOS.

Al examinar la figura 4.46 puedes ver que el hardware (el Arduino y la

placa de pruebas con los componentes) es el mismo. Todos los cambios para conseguir que el led parpadee reproduciendo el patrón SOS se hacen en el sketch que escribiste en el IDE de Arduino. No es necesario que desconectes el Arduino del ordenador cuando modificas el código, solo cuando cambias los componentes en el circuito.

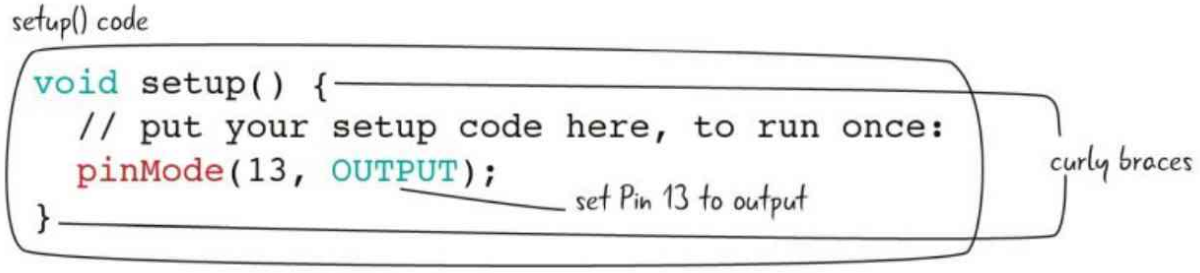
> **Note** El Arduino puede estar conectado al ordenador solo si estás modificando el código, pero no cuando cambias el hardware.

GUARDAR Y CAMBIAR EL NOMBRE

Selecciona **Guardar como** y cámbiale el nombre al sketch por **LEA4_SOS**. Algunas partes de este nuevo sketch tendrán el mismo código que has usado, pero tendrás que añadir bastante código nuevo. El código dentro de setup() cambiará poco y el código loop() será mucho más largo. Examina el código de LEA4_Blink, y luego revisa el código en loop().

Examen y revisión del código: ¿qué vas a cambiar?

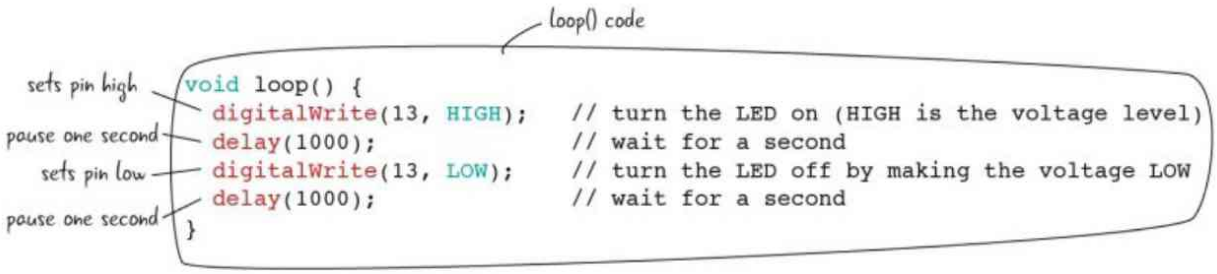
Primero echa un vistazo al código de setup(). Después de un comentario que te dice lo que hace la siguiente línea, hay una línea que configura LED_BUILTIN, asociado con el pin 13 como salida. En lugar de dejarlo como LED_BUILTIN, vas a cambiar el código de setup() para incluir la línea pinMode(13, OUTPUT)



> **Note** Cambias `LED_BUILTIN` a 13 en el `pinMode()` en el código de `setup()`.

A diferencia del código de setup(), revisarás el código de loop() y lo ampliarás bastante. Examina el código del sketch LEA4_Blink antes de hacer cambios.

La primera línea en loop() configura LED_BUILTIN a HIGH, y enciende el led. delay() entonces detiene el Arduino, en este ejemplo durante 1.000 milisegundos, o un segundo. A continuación pon LED_BUILTIN a LOW, y apaga el led. delay() detiene de nuevo durante 1.000 milisegundos. Como el código en loop() se repite una y otra vez, el led parpadea.



Vamos a ver cómo revisas loop().

Ajuste de *loop()* en el sketch SOS

El código para la señal SOS será de tres destellos breves del led seguidos por tres destellos largos, después otros tres destellos breves, con una pausa final antes de que se repita de nuevo el código. Escribirás el código para los tres destellos cortos. Primero lo veremos en su conjunto, y luego veremos cada línea. También haremos referencia al pin 13 por su número, sustituyendo todo lo referente a LED_BUILTIN del código LEA4_Blink.

TRES DESTELLOS BREVES

Después de un comentario que explica lo que hace el código, el pin 13 está

configurado a HIGH, seguido de un retardo, a continuación está configurado a LOW, seguido de un retardo. Esto se repite tres veces.

```
// 3 short flashes
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(500);               // wait for a half second
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(300);               //pause for 300 milliseconds, about 1/3 of a second
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
delay(300);
```

Echemos un vistazo más de cerca. La primera línea de código en loop() permanecerá igual que en el sketch LEA4_Blink. Como has visto, esta línea pone el pin 13 a HIGH.

set Pin 13 to high

```
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
```

Vas a introducir un cambio en la siguiente línea de código. Recuerda que la función delay() produce una pausa, medida en milisegundos. En el sketch original hacías una pausa de 1 000 milisegundos, o un segundo. Ahora quieres hacer una pausa más corta, de 500 milisegundos, o medio segundo. Cambia los comentarios para explicar lo que hace tu código.

pause for 1/2 second

```
delay(500);               // wait for a half second
```

La siguiente línea configura el pin a LOW, es decir, apaga el led. Puedes dejar esta línea tal como está, ya que no hay necesidad de cambiarla en el

sketch LEA4_Blink.

set Pin 13 to low

```
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
```

Sin embargo, cambiarás el número de milisegundos en delay(). En el sketch LEA4_Blink, el retardo era de 1 000 milisegundos, o un segundo. Ahora harás una pausa de 300 milisegundos, aproximadamente un tercio de segundo. Adaptarás también los comentarios.

pause for 300 milliseconds

```
delay(300);              //pause for 300 milliseconds, about 1/3 of a second
```

Este es el ciclo completo:

```
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(500);               // wait for a half second
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(300);               //pause for 300 milliseconds, about 1/3 of a second
```

Quieres que el led se encienda y se apague tres veces. Añade primero un comentario indicando lo que hace esta parte del código, luego copia dos ciclos más de activación y desactivación. Aquí está el código de nuevo:

```
// 3 short flashes
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(500);               // wait for a half second
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(300);               //pause for 300 milliseconds, about 1/3 of a second
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(500);
digitalWrite(13, LOW);
delay(300);
```

Veamos ahora el código para los destellos más largos.

ADICIÓN DE TRES DESTELLOS LARGOS

La sección de los tres destellos largos es similar a la sección de los destellos cortos. Después de poner el pin a HIGH, la función delay() hace una pausa de 1 500 milisegundos, o un segundo y medio, manteniendo el led encendido durante ese tiempo. Veamos primero el código de destellos largos. Un comentario explica lo que hace el código inmediatamente después.

```
// 3 long flashes
digitalWrite(13, HIGH);   // turn the LED on
delay(1500);              // wait for a second and a half
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(300);
digitalWrite(13, HIGH);
delay(1500);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(1500);
digitalWrite(13, LOW);
delay(300);
```

De nuevo tienes un ciclo repetitivo: de configuración del pin a HIGH, pausa, configuración del pin a LOW, pausa, que se repite tres veces. En primer lugar, pon el pin a HIGH.

```
set Pin 13 to high
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
```

A continuación, haz una pausa con la función delay(), en este caso de 1 500 milisegundos, o un segundo y medio. Los comentarios se han cambiado también para indicar la cantidad de tiempo ajustada.

```
pause for 1 1/2 seconds
delay(1500);              //wait for a second and a half
```

Al igual que en el código para los destellos cortos, debes configurar el pin a LOW, y luego definir la pausa con delay().

```
set Pin 13 to low
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
```

A continuación, utilizarás el mismo número de milisegundos que delay() entre los destellos cortos: 300 milisegundos.

```
pause for 300 milliseconds
delay(300);               //pause for 300 milliseconds, about 1/3 of a second
```

Una vez más, estás creando un ciclo que se va a repetir. Después de los últimos ciclos de destellos cortos, harás que la pausa sea más larga para poder diferenciar cada señal de SOS. Veamos eso, y luego veremos todo el código junto en loop().

Esta última línea de código en loop() detiene el Arduino durante 3 000 milisegundos, o 3 segundos. A esto le sigue una línea en la que se pone el pin 13 a LOW. Quieres que haya una pausa más larga entre cada señal SOS para asegurar que los observadores puedan distinguir los ciclos.

final lines in loop()

```
digitalWrite(13, LOW);
delay(3000);                //final delay is 3 seconds
```

¿PREGUNTAS?

P: Si no he cambiado el código en `LED_BUILTIN` a 13. ¿Por qué funciona todavía?

R: `LED_BUILTIN` es lo mismo que el pin 13, así que aunque cambies de uno a otro, el sketch seguirá funcionando. Es mejor escoger solo uno para no confundirte con lo que está sucediendo en el sketch.

TODO EL CÓDIGO DE SOS loop()

Veamos ahora todo el código en loop(). Es largo, así que lo dividimos en secciones.

loop() declaration and start curly brace

```
void loop() {
// 3 short flashes
 digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
 delay(500);               // wait for a half second
 digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
 delay(300);               //pause for 300 milliseconds, about 1/3 of a second
 digitalWrite(13, HIGH);
 delay(500);
 digitalWrite(13, LOW);
 delay(300);
 digitalWrite(13, HIGH);
 delay(500);
 digitalWrite(13, LOW);
 delay(300);
```

3 short flash code

```
// 3 long flashes
digitalWrite(13, HIGH);   // turn the LED on
delay(1500);              // wait for a second and a half
digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
delay(300);
digitalWrite(13, HIGH);
delay(1500);
digitalWrite(13, LOW);
delay(300);
digitalWrite(13, HIGH);
delay(1500);
digitalWrite(13, LOW);
delay(300);
```

3 long flash code

```
// 3 short flashes again
 digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
 delay(500);               // wait for a half second
 digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
 delay(300);               //pause for 300 milliseconds, about 1/3 of a second
 digitalWrite(13, HIGH);
 delay(500);
 digitalWrite(13, LOW);
 delay(300);
 digitalWrite(13, HIGH);
 delay(500);
 digitalWrite(13, LOW);
 delay(3000);          //final delay is 3 seconds
}
```

3 short flash code

final delay is 3000 milliseconds

curly brace closes the loop code

Una vez que has escrito el código para la señal luminosa de SOS y lo has guardado, tienes que hacer clic en el botón **Verificar** para comprobar si hay errores (figura 4.47).

FIGURA 4.47 Verificación satisfactoria.

Si está bien, tienes que asegurarte de que el ordenador está conectado al Arduino, y que tienes seleccionados la tarjeta y el puerto correctos. A continuación, haz clic en el botón **Subir** para subir el código en el Arduino

(figura 4.48).

FIGURA 4.48 Carga satisfactoria.

¿Cómo se ve el led ahora en la tarjeta?

¡PARPARDEO DE LA SEÑAL LUMINOSA SOS!

El led debería estar parpadeando ahora con una señal con el patrón SOS: tres ráfagas cortas, seguidas de tres destellos largos, tres ráfagas cortas de nuevo, una pausa de 3 segundos, luego se repite todo el patrón una y otra vez. Hay otras formas más eficientes de escribir código, pero por ahora lo que queremos es que hagas ajustes y entiendas lo que hace el código al ver los resultados en el circuito (figura 4.49).

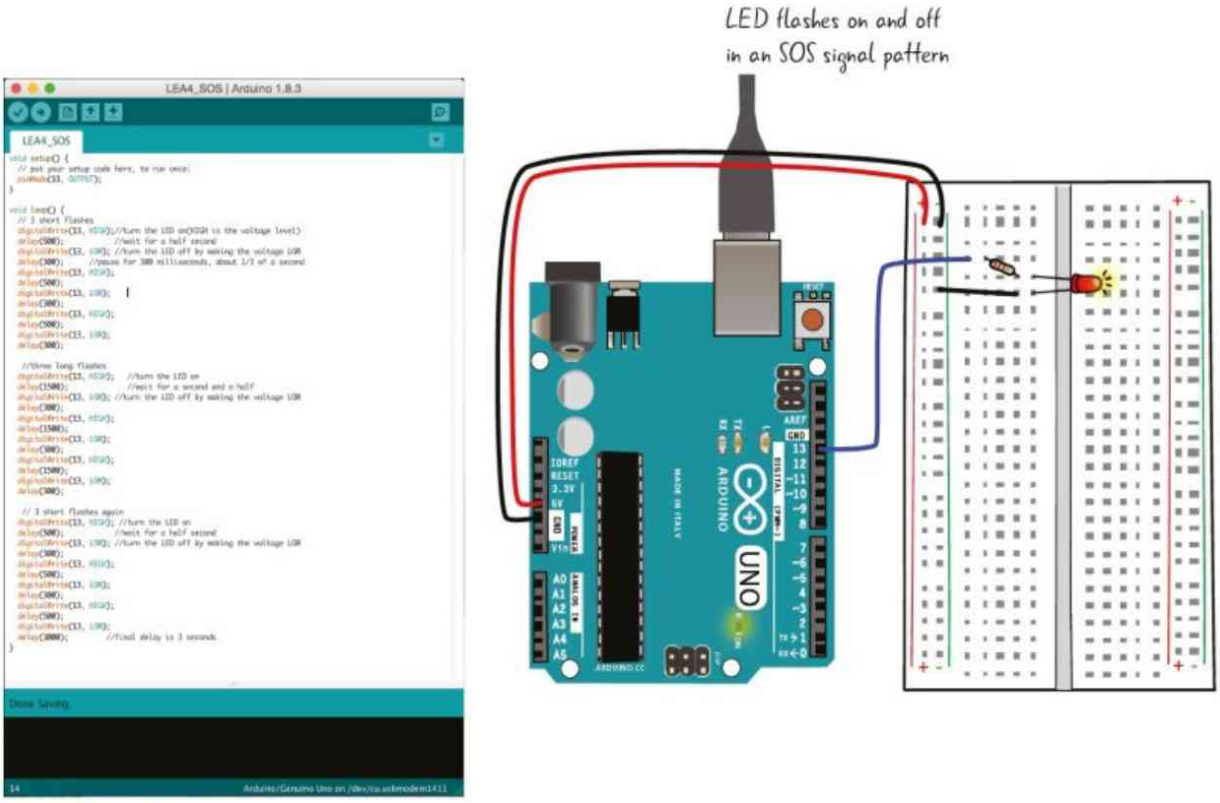


FIGURA 4.49 El led parpadea siguiendo el patrón SOS.

RESUMEN

Has configurado el IDE de Arduino, has aprendido a verificar y subir el código, has visto cómo conectar una placa de pruebas al Arduino, y explorado como escribir un sketch en el lenguaje de programación Arduino. Puedes descargar el código de LEA4_SOS en https://github.com/arduinotogo/LEA/blob/master/LEA4_SOS.ino.

En el siguiente capítulo seguirás aprendiendo cómo escribir código en el lenguaje de programación Arduino y cómo acoplar diferentes tipos de componentes a un circuito.


5

ElectriciDAD Y Medición

¿Q ué son el voltaje, la corriente y la resistencia? ¿Cómo están relacionados? ¿Y por qué deberían importarte?

En este capítulo estudiarás el voltaje, la corriente y la resistencia y cómo interactúan entre ellos. Esto te ayudará a comprender cómo funcionan los circuitos y cómo proceder a su ajuste. Aprenderás también a utilizar el multímetro para medir sus valores. Estos conocimientos te ayudarán a depurar los circuitos y te permitirán diseñar y montar tus propios proyectos desde el principio.

QUÉ ES LA ElectriciDAD

La electricidad es un flujo de electrones a través de un material, como se puede ver en la figura 5.1. En los proyectos de este libro, los electrones fluyen a lo largo de pistas cuidadosamente dispuestas y organizadas en los circuitos.

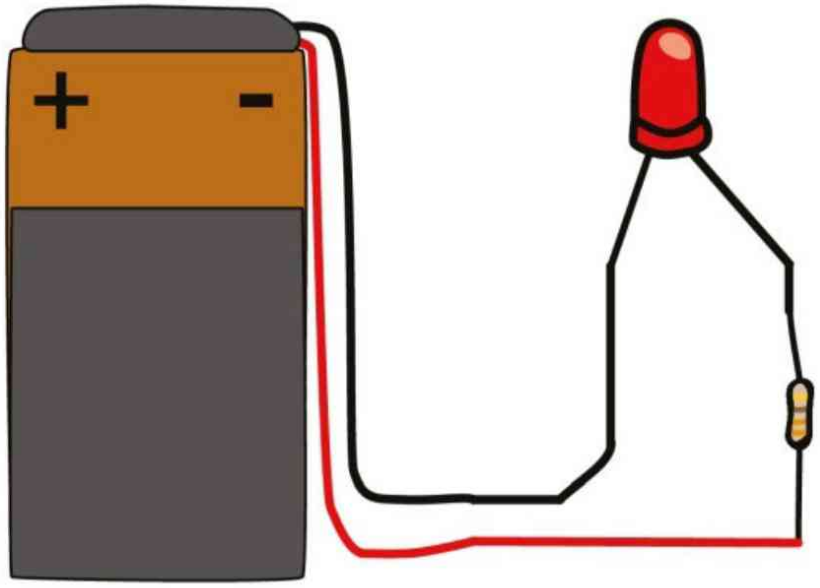


FIGURA 5.1 La electridad circula por los circuitos..

La electricidad tiene tres características importantes: voltaje, corriente, y resistencia. En este capítulo, verás cómo estas propiedades interactúan entre ellas, en una relación fundamental llamada ley de Ohm. Aprenderás también cómo situar los componentes en diferentes disposiciones que afectan a las características eléctricas de un circuito.

¿Por qué nos detenemos en estudiar las propiedades de la electricidad en lugar de dedicarnos a montar circuitos con el Arduino y otros componentes? Si no entiendes cómo actúan estas características en un circuito, será muy difícil poder avanzar en el montaje de circuitos una vez hayas completado los proyectos de este libro. Además, sin un conocimiento de estas propiedades, la resolución de los problemas que aparecen en los proyectos es casi imposible. En este capítulo aprenderás más técnicas para depurar tus proyectos.

MeDIDAS DE LAS PROPIEDADES EléctricaS CON EL MultÍMETRO

¿Recuerdas el multímetro (figura 5.2) del capítulo 3, "Primer contacto con el circuito"? Aprendiste a manejarlo para comprobar la continuidad (si los componentes estaban conectados entre sí). El multímetro te ayuda a resolver los problemas en un circuito. Al probar la continuidad, por ejemplo, puedes verificar que el circuito es un bucle completo. En este capítulo, aprenderás a utilizar el multìmetro para medir el voltaje, la corriente y la resistencia. ¿Por qué necesitas hacer esto? Medir el voltaje te ayudará a analizar los problemas que se presentan en los circuitos; por ejemplo: ¿hay voltaje en el circuito?, ¿qué caída de tensión se produce en cada uno de los componentes?

> **Note** Comprender cómo el voltaje, la corriente y la resistencia interactúan en un circuito te ayuda a resolver los problemas que aparecen en los proyectos, así como a montar nuevos circuitos.

FIGURA 5.2 El multímetro.

Si vas a utilizar un multímetro para comprobar las propiedades eléctricas en un circuito, necesitarás en primer lugar montar el circuito. Empezaremos con un circuito básico que tiene un led, una resistencia, una placa de pruebas y un Arduino. Esta vez no vas a elaborar un sketch, solo utilizarás el Arduino como fuente de energía. Comprobarás el voltaje que suministra el Arduino, y

después medirás el voltaje que tiene cada uno de los componentes. A continuación añadirás un segundo led a la placa de pruebas y verás cómo cambian las características eléctricas de los componentes en función de si los montas en serie o en paralelo. En breve explicaremos exactamente lo que queremos decir con todo esto.

MONTAJE DEL CIRCUITO PASO A PASO

Para montar el circuito básico descrito en este capítulo, necesitarás los siguientes elementos:

- 1 led rojo
- 1 resistencia de 220 ohmios (rojo, rojo, marrón, dorado)
- Cables de puente
- Placa de pruebas
- Arduino Uno
- Cable USB tipo A-B
- Ordenador

Este circuito es muy similar al que montaste en el capítulo 4, "Programación del Arduino". La única diferencia es que no estás alimentando el led desde un pin del equipo Arduino sino desde el bus de alimentación de 5 voltios de la placa de pruebas.

¡MONTAJE!

Como dijimos anteriormente, hay una gran diferencia entre este circuito, que se muestra en la figura 5.3, y el que montaste en el capítulo 4: no lo conectarás al pin digital del equipo Arduino. En su lugar, vas a conseguir la alimentación del bus de alimentación de la placa de pruebas. Recuerda que el bus de alimentación está conectado mediante un puente de color rojo al pin etiquetado 5 V (5 voltios) del Arduino.

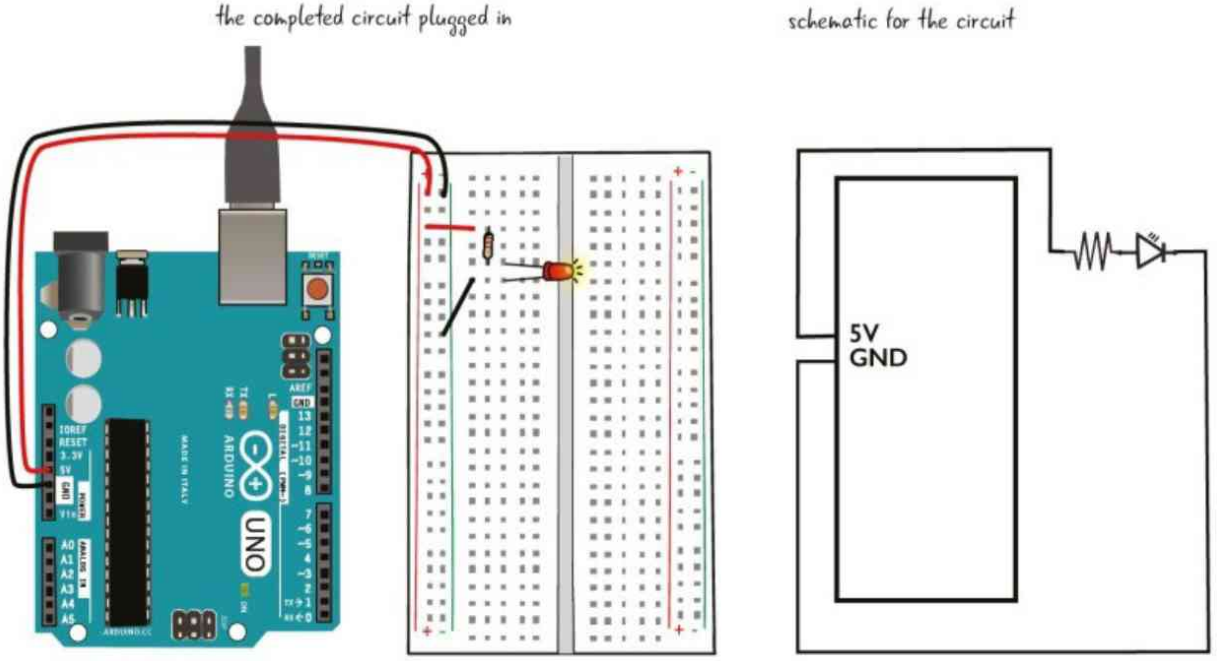


FIGURA 5.3 Circuito con el esquema.

Antes de empezar a montar el circuito, echemos un vistazo a los pines de alimentación y de tierra, mostrados en la figura 5.4. Hay un pin etiquetado con 5 V, y también otro con 3,3 V, así como pines etiquetados con GND para indicar la toma de tierra.

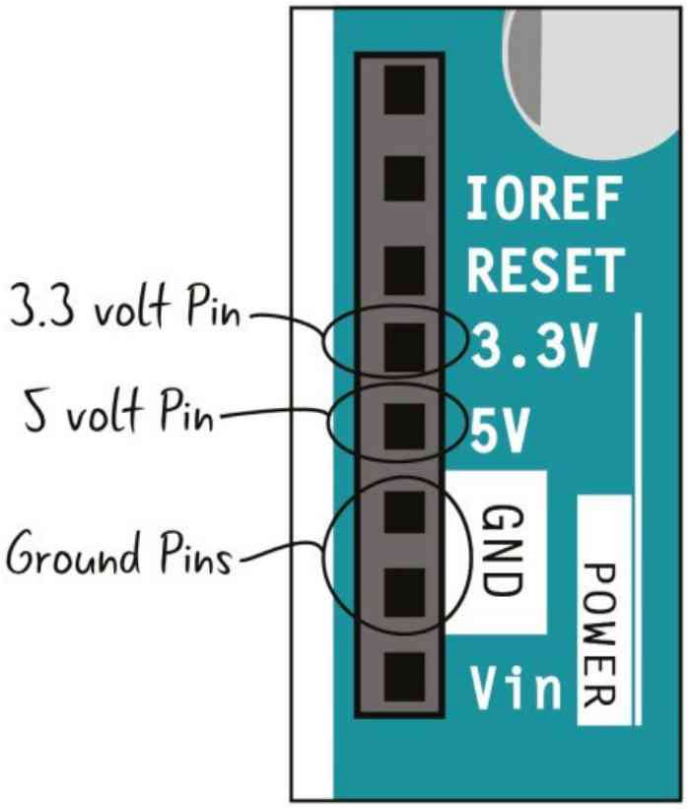


FIGURA 5.4 Pines de alimentación y tierra del Arduino.

Aunque hemos estado montando los circuitos utilizando 5 V, también puedes utilizar 3,3 V para montar otros circuitos cuyos componentes requieren menos voltaje. El puerto de 3,3 V funciona igual que el de 5 V excepto que su salida presenta un menor voltaje.

A continuación, los pasos que hay que seguir para montar un circuito, y que se ilustran en la figura 5.5:

1. Conecta un extremo de un puente al pin de 5 V del Arduino y el otro extremo, al bus de alimentación de la placa de pruebas (la columna etiquetada con un signo + de color rojo).
2. Toma otro puente y conecta un extremo al pin GND del Arduino y el otro extremo, al bus de tierra de la placa de pruebas (la columna etiquetada con un signo – de color verde).
3. Conecta con un puente el bus de alimentación y una fila de puntos de contacto.

4. Conecta un terminal de la resistencia de 220 ohmios a la misma fila de puntos de contacto. El otro terminal se conecta a otra fila de puntos de contacto.
5. Conecta el ánodo (terminar largo) del led al otro terminal de la resistencia.
6. Instala un puente entre el cátodo (terminal corto) del led y el bus de tierra.

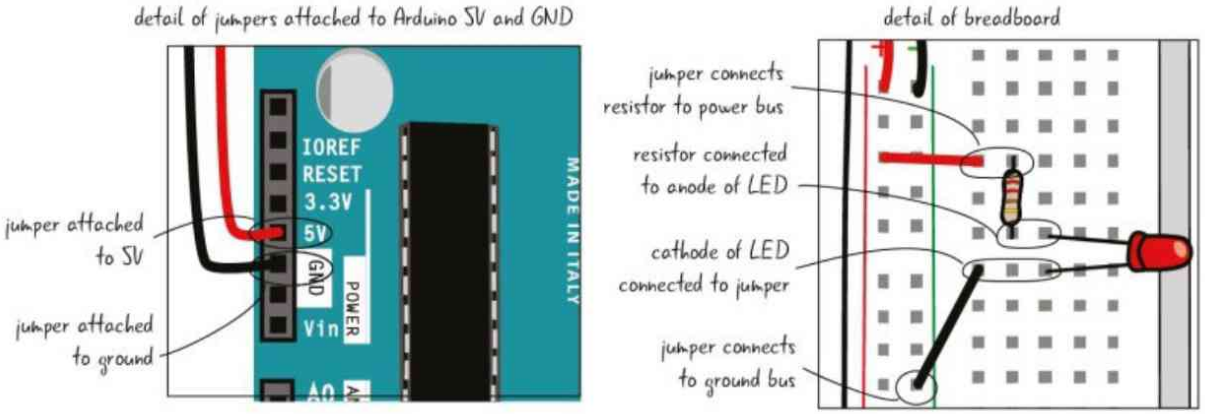


FIGURA 5.5 Circuito con los detalles de los pines de alimentación y tierra en el Arduino y sus componentes sobre la placa de pruebas.

Cuando tengas montado el circuito, utiliza el cable USB para conectar el Arduino al ordenador. No vamos a crear ningún sketch; solo vas a usar el ordenador como fuente de alimentación para el Arduino.

DePURACIÓN DEL CircuitO

Si el led se ilumina cuando conectas el cable USB al ordenador, puedes pasar a la página siguiente. Si no es así, vamos a solucionar el problema, o a "depurar" el circuito.

En los capítulos anteriores hicimos una referencia a la depuración. Como recordatorio, la depuración se define como el proceso por el cual compruebas metódicamente el proyecto para eliminar cualquier inconveniente que pudiera estar causando problemas.

Comprobueba que la alimentación y la tierra están conectados a los buses de la placa de pruebas y a los puertos adecuados del Arduino. La conexión mostrada en la figura 5.6 no está bien hecha.

Comprueba que el led está correctamente orientado (el ánodo conectado a la resistencia que está conectada a la alimentación, el cátodo conectado al puente que está conectado a tierra). La figura 5.7 muestra cómo debería ser.

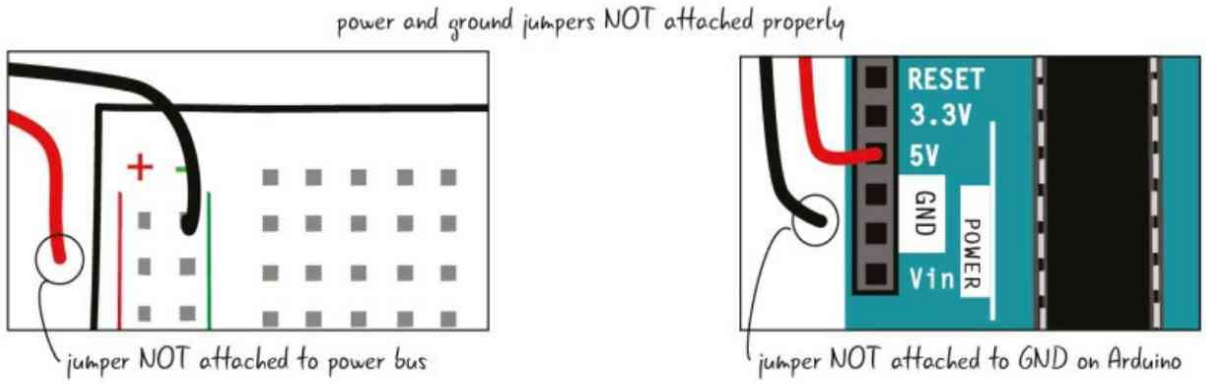


FIGURA 5.6 Alimentación y tierra en el Arduino y placa de pruebas mal conectadas.

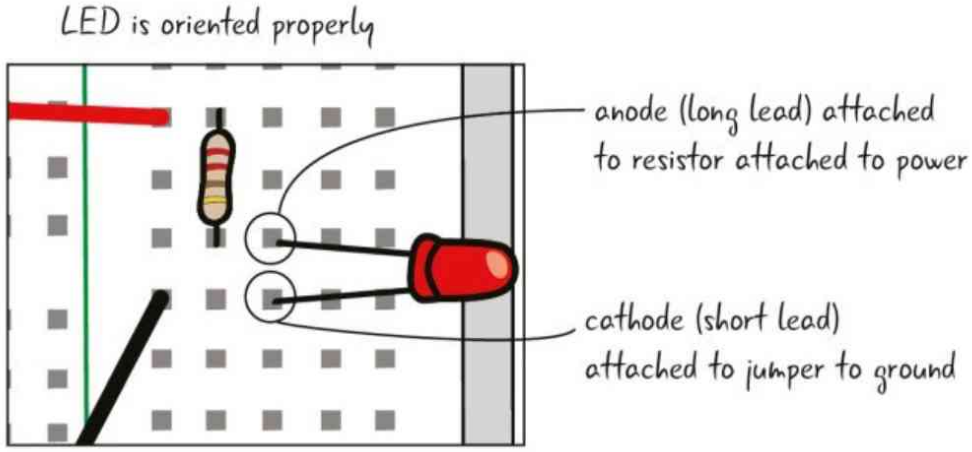


FIGURA 5.7 Comprobación de la orientación del led.

Comprobueba la continuidad; los terminales de los componentes que se supone que están conectados deben estar en la misma fila de puntos de conexión. La figura 5.8 muestra un circuito en el que los componentes no están conectados.

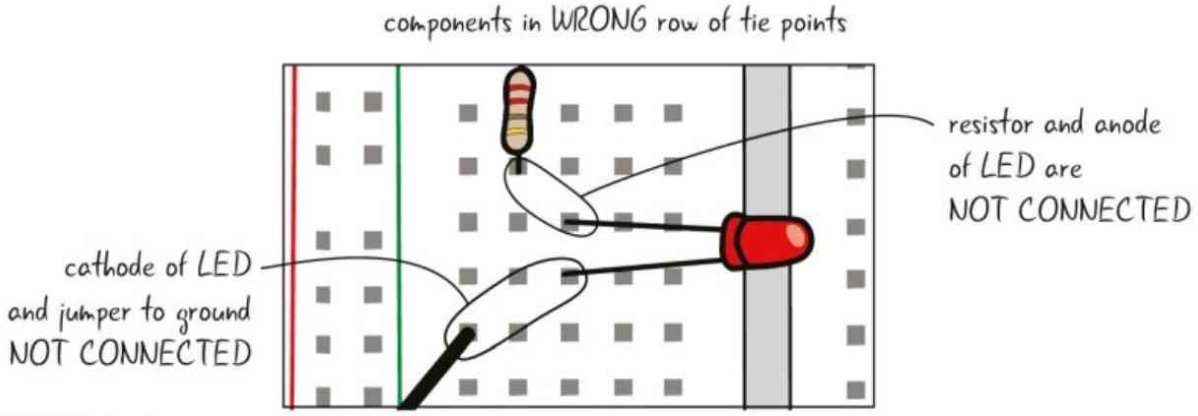


FIGURA 5.8 Componentes que *no* están conectados entre ellos.

Ahora que tienes el circuito operativo, vamos a ver cómo fluye la electricidad a través del circuito.

¿PREGUNTAS?

P: ¿Por qué ponen tanto énfasis a la depuración?

R: En cualquier proyecto electrónico, pueden salir mal muchas cosas. Es inteligente mantener un enfoque integral de la comprobación de errores.

la ElectriciDAD: UNA VISIÓN GENERAL

La electricidad es un flujo de electrones a través de un material. La electricidad requiere un trazado cerrado para fluir de principio a fin. Los circuitos crean un trazado cerrado con líneas conductoras y componentes. La electricidad sigue el recorrido del circuito. La figura 5.9 muestra el recorrido de la electricidad en el circuito que acabas de montar.

> **Note** Hemos simplificado este análisis de la electricidad. Queríamos reducir la complejidad de las explicaciones para adaptarlas a los proyectos a pequeña escala que estamos montando. Este no es un manual adecuado para tener una idea clara de la electricidad o de una teoría electrónica compleja.

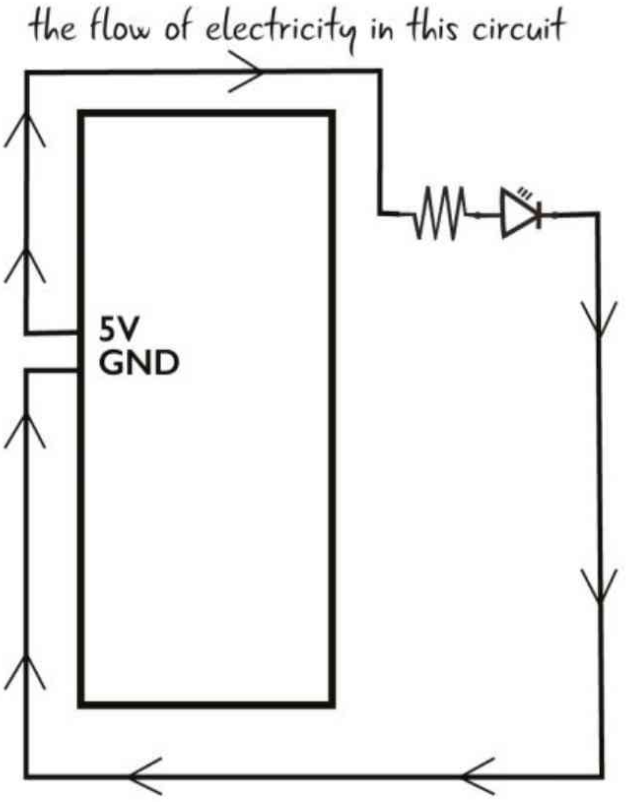


FIGURA 5.9 Esquema en el que se indica el sentido del flujo eléctrico.

¿CÓMO SE COMPORTA LA ELECTRICIDAD?

Los materiales se pueden clasificar en dos tipos. El primer tipo son los conductores, que permiten que la electricidad fluya a través de ellos. Los cables están hechos de metal, ya que el metal es un buen material conductor. El segundo tipo de materiales son los aislantes, que se oponen al paso de la electricidad. El caucho es un ejemplo de aislante.

> **Note** Los conductores dejan fluir la electricidad, mientras que los aislantes dificultan su paso.

CORRIENTEs CONTiNUA Y ALTERNA

Hay dos tipos diferentes de fluido eléctrico: corriente alterna (C.A.) y corriente continua (C.C.). La electricidad que sale de un enchufe de la pared es C.A., mientras que el Arduino, así como muchos proyectos y componentes electrónicos sencillos, utiliza C.C. En el caso de la corriente alterna, representada en la figura 5.10, el flujo de la electricidad cambia su sentido. En la corriente continua el flujo se produce en un solo sentido.

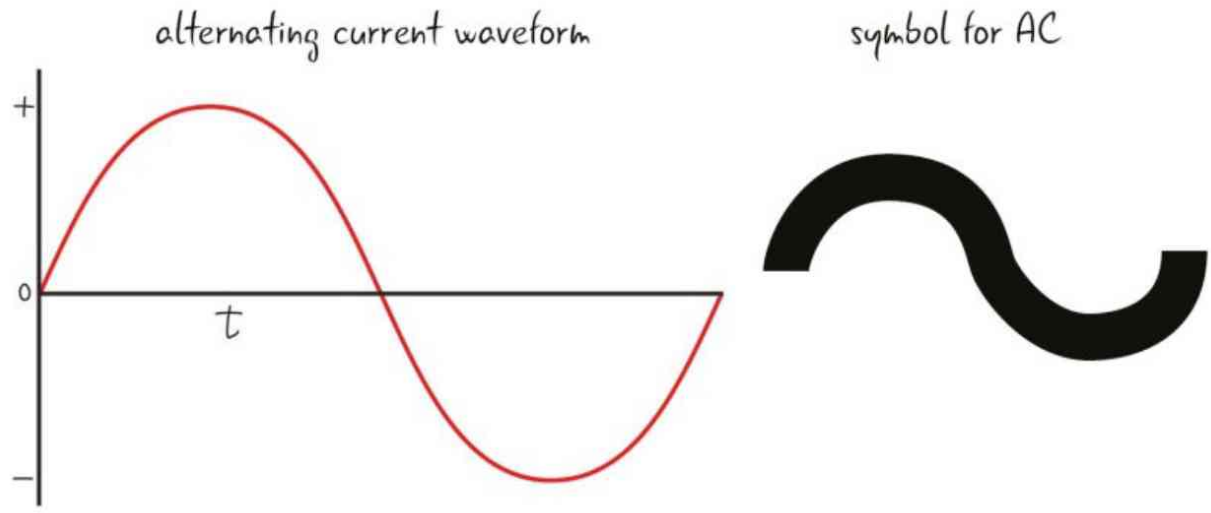


FIGURA 5.10 Corriente alterna.

Una de las principales ventajas de la C.A. es que se puede distribuir a grandes distancias, algo que es mucho más complicado con la C.C. La C.A. es también capaz de aumentar la cantidad de voltaje suministrado de forma mucho más eficiente que la C.C. Los proyectos electrónicos a pequeña escala, como los que montamos con el Arduino, no necesitan transportar la electricidad a grandes distancias, ni requieren generalmente usar alto voltaje. Por estas razones nos limitaremos a hacer una descripción de la corriente continua, que explicaremos con mayor detalle en las siguientes páginas. La corriente continua, que utilizan la mayor parte de los proyectos a pequeña escala, se representa en la figura 5.11.

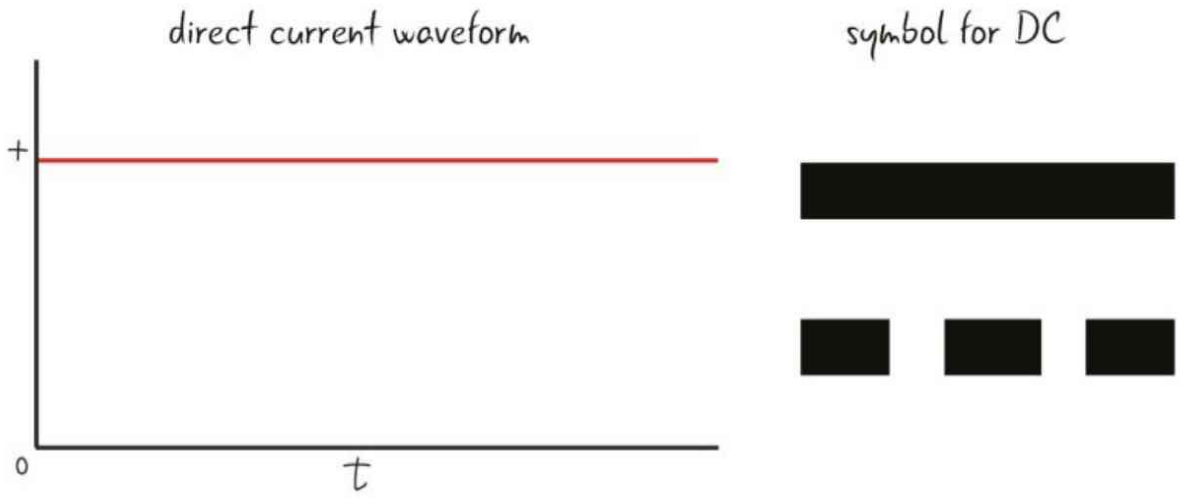


FIGURA 5.11 Corriente continua.

Warning La electricidad es peligrosa. ¡No intentes hacer nada relacionado con la corriente alterna!

¿PREGUNTAS?

P: ¿Es una batería de corriente continua?

R: Sí, una batería utiliza corriente continua.

P: La fuente de alimentación que utilizamos con el Arduino ¿es C.A. o C.C.?

R: La fuente de alimentación del Arduino lo que realmente hace es convertir la C.A. en C.C. Utiliza un transformador, pero no lo vamos a ver en este libro.

PARA COMPRENDER LA ELECTRICIDAD: LA ANALOGÍA DEL TANQUE DE AGUA

Veamos las tres características más importantes de la electricidad: voltaje, corriente y resistencia. Analizamos cómo funciona la electricidad en C.C.; el funcionamiento de la C.A. es algo diferente, y no lo vamos a tratar aquí. Este capítulo proporciona suficiente información para que puedas montar tus proyectos de Arduino; sin embargo, si tu interés por la electrónica y por la ingeniería eléctrica es más profundo, la descripción simplificada que presentamos aquí no será suficiente. Sugerimos los siguientes títulos: Getting Started in Electronics, de Forrest M. Mims, III (Master Publishing, Inc., 2003); Make: Electronics: Learning Through Discovery, 2nd Edition, de Charles Platt (Maker Media, 2009); y Practical Electronics for Inventors, 4th Edition, de Paul Scherz y Simon Monk (McGraw-Hill Education, 2016).

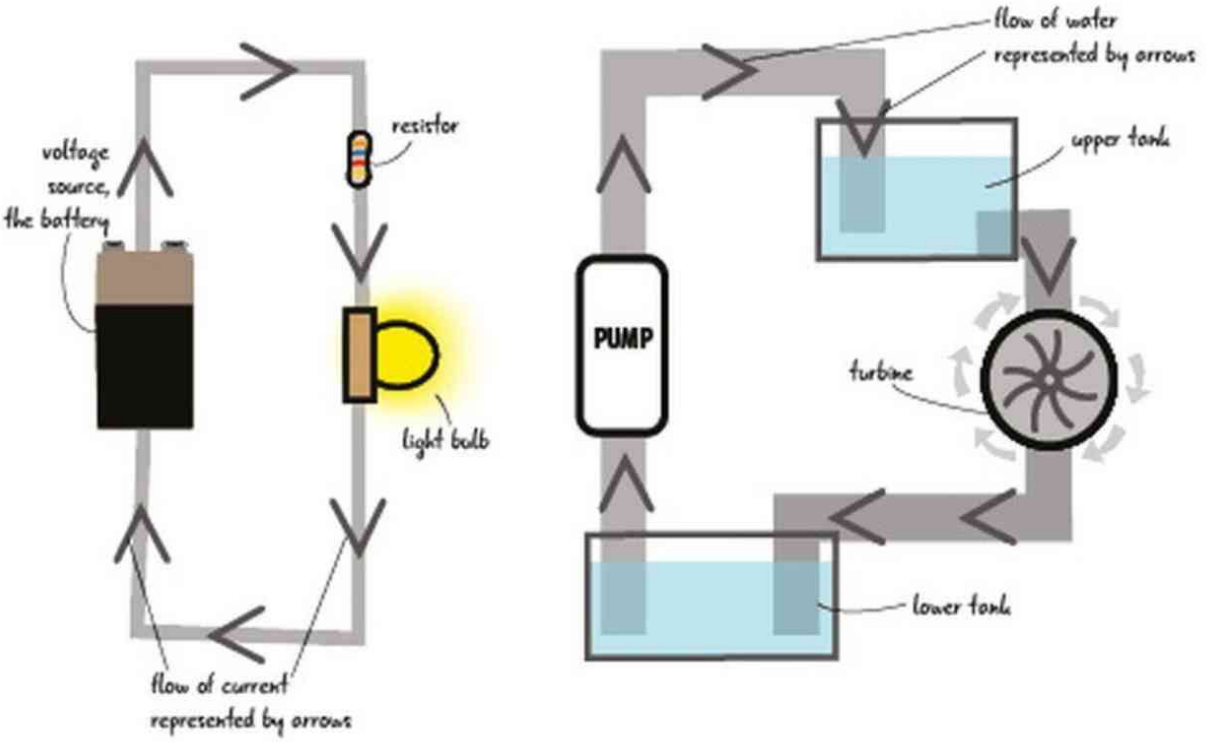


Figura 5.12:**Analogía entre los circuitos hidráulico y eléctrico.**

Para poder comprender cómo el voltaje, la corriente y la resistencia

interactúan entre ellos, usaremos una analogía que se emplea frecuentemente y que relaciona las propiedades eléctricas con un sistema de agua, como se muestra en la figura 5.12. A la izquierda, vemos un circuito eléctrico que tiene una fuente de voltaje, una bombilla y una resistencia. El flujo de la electricidad está representado por las flechas. A la derecha tenemos un sistema hidráulico, con una bomba, dos tanques de agua, una turbina y tubos que conectan todas las piezas. El flujo de agua se representa también por las flechas.

¿PREGUNTAS?

P: ¿Por qué empleamos la analogía del agua para comprender las características de la electricidad?

R: Los conceptos eléctricos son necesariamente abstractos y difíciles de visualizar. Aunque la analogía del agua es una simplificación de cómo funcionan las características eléctricas, nos ayuda a conceptualizar la interacción entre ellas.

¿Cómo utilizamos esta analogía para comprender las propiedades de la electricidad? Veamos primero el voltaje. En nuestro sistema eléctrico, tenemos una fuente de alimentación: una batería. ¿A qué se asemeja esto en nuestro sistema hidráulico?

EL VOLTAJE: EL POTENCIAL

En nuestro sistema hidráulico, el agua tiene un potencial para caer desde el tanque superior, mover la turbina y caer en el tanque inferior. Cuando el agua llega al tanque inferior no tiene más potencial para caer (porque ya está en el punto más bajo del sistema). Si aumentamos la cantidad de agua en el tanque superior, aumentamos la presión, o el potencial, para que caiga el agua. Si hay un aumento de la presión del agua, la turbina se moverá más rápidamente y producirá más trabajo. Si se disminuye la cantidad de agua en el tanque, hay menos presión o potencial para caer, así la turbina se moverá más despacio y realizará menos trabajo (figura 5.13).

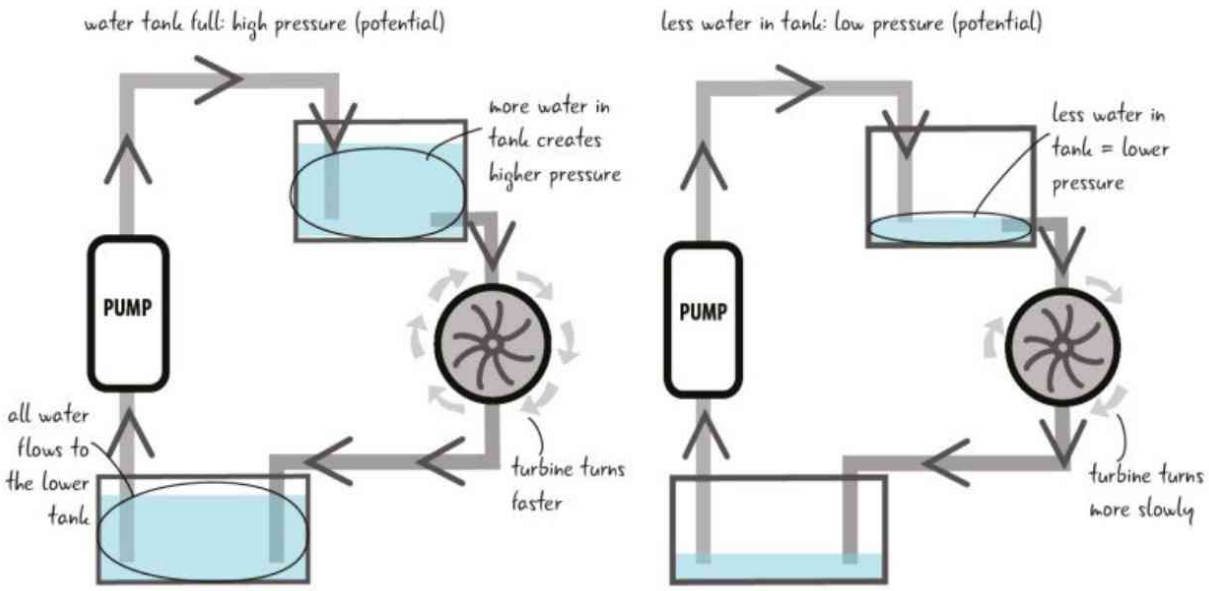


FIGURA 5.13 Voltaje en la analogía con el agua.

¿Cómo se relaciona esto con el voltaje? En el sistema eléctrico, los electrones tienen el potencial, es decir, tienen la presión para moverse desde un área de mayor carga a otra de menor carga. De la misma forma que una mayor cantidad de agua hace que la turbina se mueva más rápido, en un circuito, un mayor voltaje de la fuente de alimentación hace que la luz brille más (mayor potencial eléctrico), mientras que una fuente de alimentación con un voltaje menor (menos potencial eléctrico) hace que el brillo sea menor, como se ve en la figura 5.14. A este potencial se lo conoce también como fuerza electromotriz.

Note **La *fuerza electromotriz* es el potencial que hace fluir la electricidad.**

De manera similar a la forma en la que el agua fluye hacia abajo (desde el punto más alto hacia el punto más bajo), la electricidad fluye desde el punto

de voltaje más alto hacia el punto de voltaje más bajo. Medir la tensión implica medir la diferencia entre la presión en dos puntos cualesquiera del sistema. La medición de la diferencia entre dos puntos es siempre un valor relativo. La figura 5.15 muestra el esquema y el modelo eléctricos, con el flujo de electricidad marcado desde la alimentación a tierra.

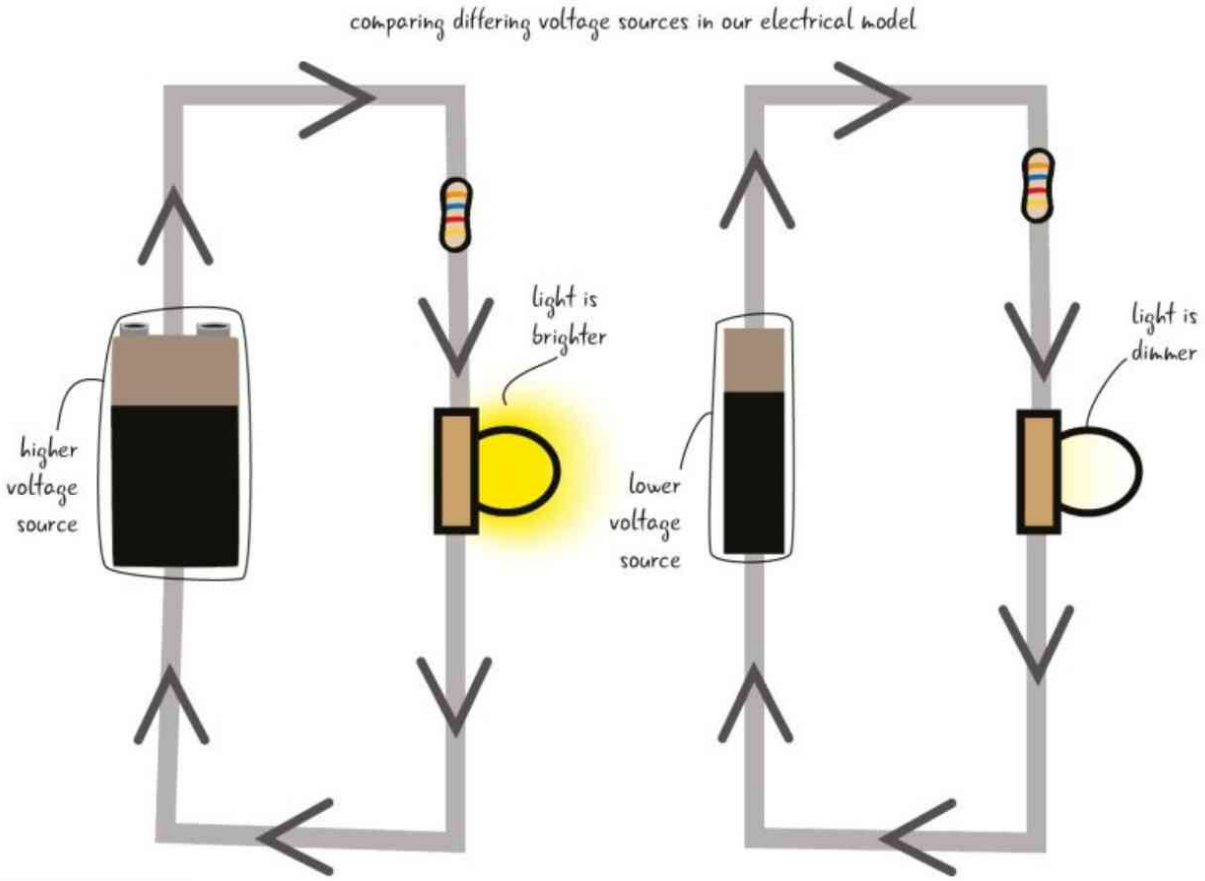


FIGURA 5.14 Modelo eléctrico con diferentes tensiones de alimentación.

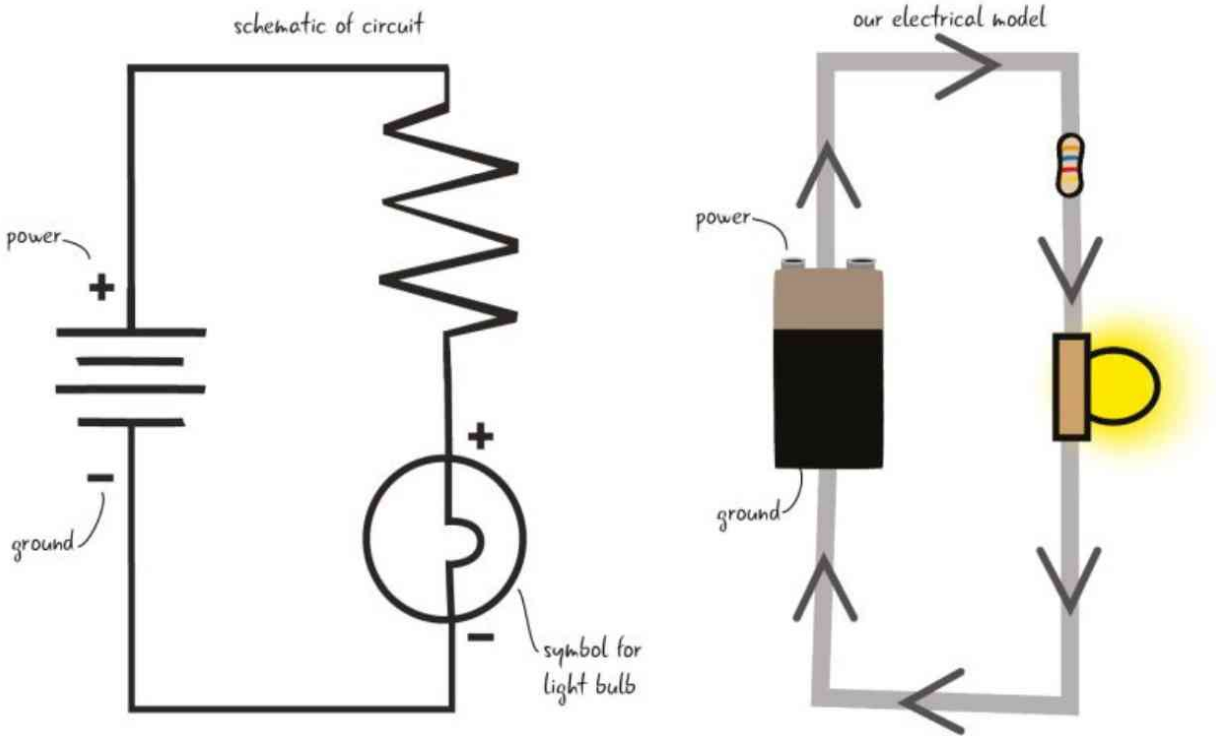


FIGURA 5.15 Modelo eléctrico con el esquema.

> **Note** El *voltaje* es la diferencia de potencial eléctrico entre dos puntos cualesquiera del circuito.

A medida que la electricidad viaja a través del circuito y sus componentes, desde un punto de mayor voltaje a otro de menor voltaje, el potencial eléctrico se distribuye y se consume hasta el punto en el que no hay más energía potencial. Este punto cero, medido a cero voltios, también se conoce como tierra eléctrica. Es análogo al tanque inferior del circuito hidráulico, el punto más bajo del sistema, donde el agua no puede seguir cayendo, ya no tiene potencial para caer. Esta es la misma tierra de la que hemos hablado en los circuitos y en el Arduino.

Note A los cero voltios (el punto donde no hay potencial eléctrico) se lo conoce como tierra.

¿PREGUNTAS?

P: Entonces, ¿la tierra que han dicho que es cero voltios es la misma tierra de la que hemos estado hablando en los circuitos a patir del capítulo 3?

R: Sí, tierra es un punto de referencia en un circuito con un potencial eléctrico igual a cero.

¿Cuál es el valor del voltaje para Arduino?

Puede que estés familiarizado con el voltaje, ya que los electrodomésticos, los componentes eléctricos y electrónicos a menudo indican un voltaje nominal. La mayoría de los dispositivos electrónicos a pequeña escala, como los cargadores de teléfono, utilizan entre 3 y 12 voltios de C.C.

Nuestro Arduino funciona con 5 voltios. ¿Recuerdas cuando conectaste la placa de pruebas al pin etiquetado con 5 V en el Arduino? Cuando el Arduino se conecta al ordenador, recibe 5 voltios de este. En los circuitos, los componentes (el led, por ejemplo) consumen parte del voltaje. Las resistencias que has utilizado en los circuitos te permiten cambiar (reducir) el valor del voltaje, sobre el que aprenderás más adelante en este capítulo.

Ahora que te has familiarizado con el voltaje, veamos cómo lo mides con un multímetro.

Comprobación del voltaje

¿Por qué medimos el voltaje? Es crítico saber que la placa de pruebas y sus componentes reciben voltaje; este es siempre uno de los primeros pasos que hay que llevar a cabo en la depuración de los proyectos electrónicos.

Continuamos utilizando el multímetro SparkFun (SparkFun part number TOL-12966). Volviendo al Capítulo 3, viste que el multímetro tiene puntas de prueba que se deben conectar a los puertos adecuados para medir las distintas propiedades eléctricas. Comprueba las puntas de prueba para tener la seguridad de que están conectadas a los puertos adecuados, y a continuación selecciona el dial del multímetro para medir el voltaje.

Medición del voltaje

Asegúrate de que la punta de prueba de color negro está conectada al puerto COM (común) y la de color rojo está en el puerto etiquetado como mAVΩ situado a la derecha del multímetro, como se puede ver en la figura 5.16. A continuación sitúa el dial en la sección del dial que mide el voltaje C.C. Cuando mides el voltaje, necesitas poner el dial del medidor en un valor por encima del valor que supones que tendrá el voltaje. Por ejemplo, si sabes que el Arduino utiliza 5 voltios, pon el dial en 20 V.

Tip **Para medir el voltaje, sitúa el dial en un valor mayor al que piensas que pueda ser la lectura.**

FIGURA 5.16 Preparación del multímetro para medir el voltaje en C.C.

Dijimos que del Arduino salen 5 voltios, comprueba con el multímetro si es cierto.

Utiliza un nuevo puente y coloca uno de sus extremos en el bus de alimentación de la placa de pruebas. El otro extremo no debería estar conectado a nada. A continuación conecta otro puente al bus de tierra, con el otro extremo suelto también. Los extremos sueltos de los puentes no pueden tocarse entre ellos, ya que se produciría un cortocircuito, en el que el camino eléctrico encuentra un atajo a tierra, y es posible que tu Arduino resultara dañado. Para evitar la posibilidad de que se produzca un cortocircuito, mantén un espacio entre los dos terminales.

> Warning **Un cortocircuito permite a la electricidad pasar a lo largo de un trayecto imprevisto y puede dañar el circuito.**

A continuación, toca el extremo metálico del puente conectado al bus de alimentación con la punta de prueba de color rojo y el extremo metálico del puente conectado al bus de tierra, con la punta de prueba de color negro, como se muestra en la figura 5.17.

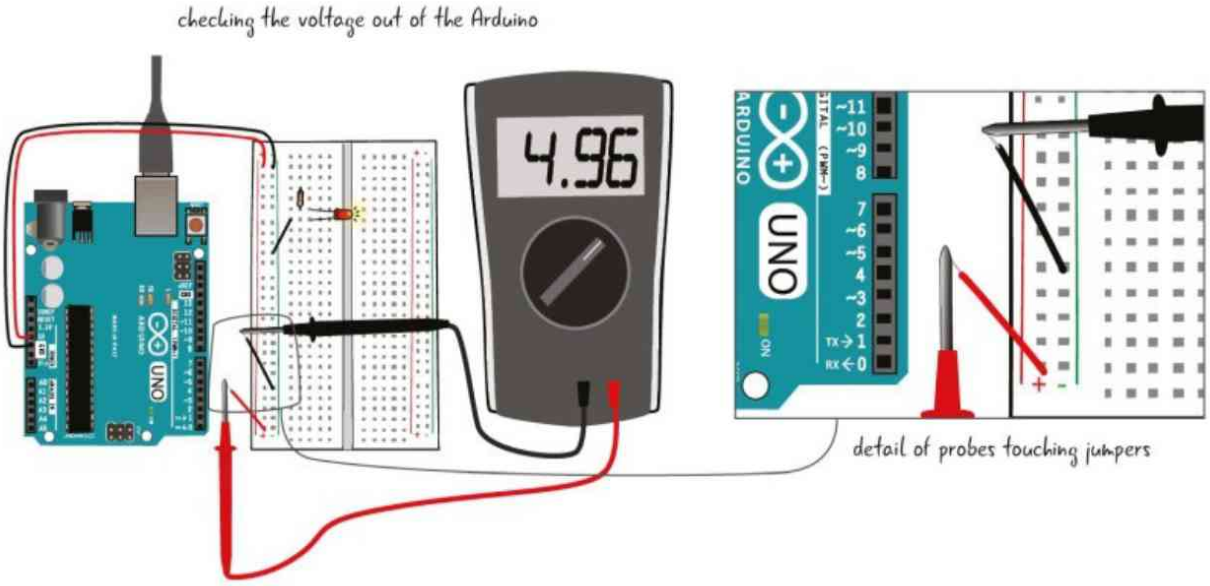


FIGURA 5.17 Medida del voltaje en la placa de pruebas del Arduino.

En la pantalla del multímetro deberías ver el número 5, aunque el número que veas puede ser un poco más bajo. El medidor lee 4.96. Este es el valor del voltaje que sale del Arduino y que entra en la placa de pruebas. Esta pequeña diferencia tiene que ver con la resistencia de la placa de pruebas, los componentes y/o el circuito interno del equipo Arduino.

¿Qué ocurre si la pantalla presenta un valor negativo? Significa que probablemente las puntas de prueba están cambiadas; la punta de prueba de color rojo está tocando el puente conectado al bus de tierra y la de color negro, al bus de alimentación. Intenta intercambiar las puntas de prueba con los puentes opuestos y obtendrás un valor positivo.

Si aparece el número 1, significa que el selector del multímetro está situado en un valor incorrecto. Sencillamente, aumenta el valor del voltaje del dial al siguiente nivel girándolo en el sentido de las agujas del reloj. Deberías ver su valor exacto.

Tip Si el multímetro muestra valores extraños, comprueba si el voltaje del dial es mayor que el valor esperado. Si tienes un voltaje negativo, hay que cambiar la conexión de las sondas de prueba.

Después de haber medido el voltaje, quita los puentes (teniendo la precaución de no ocasionar un cortocircuito si se tocan accidentalmente). Vamos a medir la tensión en los componentes del circuito.

Warning Debes apagar el multímetro si no lo estás utilizando, ya que puede agotarse la batería.

medida DEL VOLTAJE EN LOS COMponentEs

Ahora medirás el voltaje a través de la resistencia y del led. Al hacer esto verás cuánto voltaje está "consumiendo" cada componente.

Situa el selector del medidor en la posición de 20 V C.C. El Arduino debe permanecer conectado al ordenador para recibir alimentación. Toca con la sonda roja el extremo de la resistencia conectada al puente del bus de alimentación y la sonda negra tocando el otro extremo, como se ve en la figura 5.18. ¿Qué ves en la pantalla del multímetro?

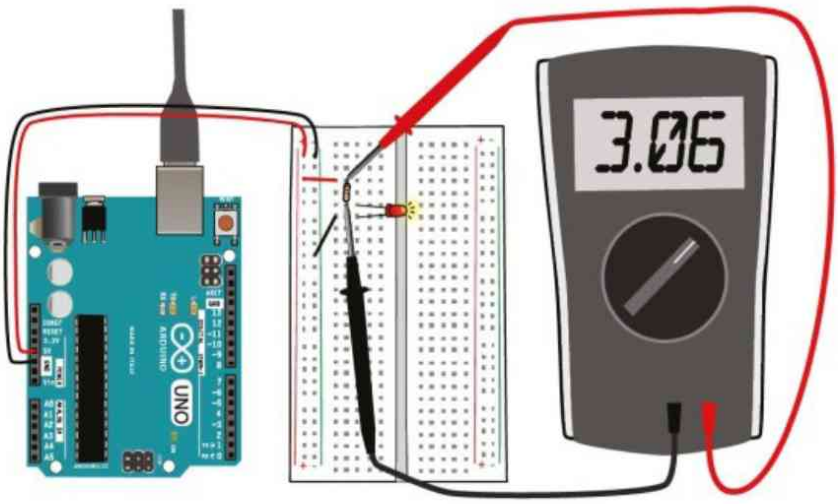


FIGURA 5.18 Medida del voltaje a través de la resistencia.

Ahora toca el ánodo del led con la sonda roja y el cátodo con la sonda negra para ver cuánto voltaje consume el led. En la figura 5.19 se muestra un detalle de las sondas del multímetro conectadas a los extremos de la resistencia y del led.

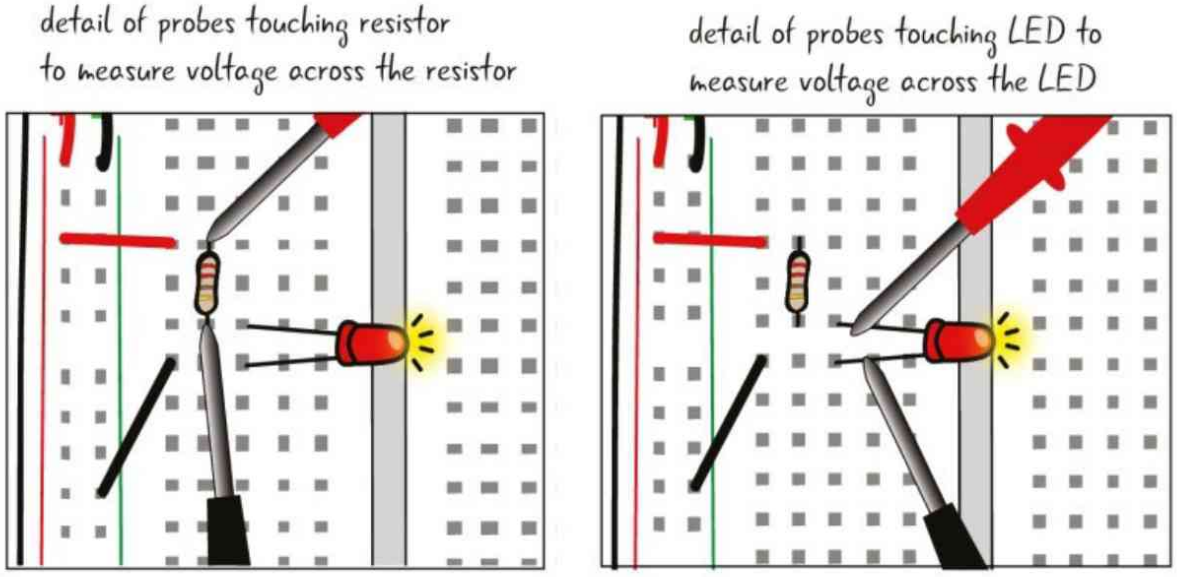


FIGURA 5.19 Detalles de la medida del voltaje en el circuito.

La pantalla del medidor debería indicar algo así como 3,06 para la resistencia y 1,86 para un led de color rojo. Estos números también variarán, en parte dependiendo del color del led que utilicemos. No hay que preocuparse si la suma no es exactamente de 5 voltios. El número que aparece en el medidor es

la cantidad de voltaje que necesita el led. A la acción de medir el voltaje a través de un componente como este, se le llama medir la caída de voltaje. La caída de voltaje es la cantidad de voltaje que consume un componente.

CAÍDA DE VOLTAJE

Si la caída de voltaje es la cantidad de voltaje que consume un componente, ¿qué significa esto para nuestros circuitos? Cada uno de los componentes consumirá parte del voltaje proporcionado por la fuente de alimentación, hasta que se consuma todo el voltaje, como muestra la figura 5.20. Si solo tenemos un componente (por ejemplo, si colocamos el led sin la resistencia), entonces todo el voltaje caería a través del componente y quemaría el led. ¿Cómo determinamos la cantidad de voltaje que consume un componente sin dañarlo? ¿Recuerdas las hojas de datos que discutimos en el capítulo 2, "Tu Arduino"? En ellas está la información que necesitamos. Ya hemos visto cómo medir esta caída de voltaje, pero más adelante en este capítulo también estudiaremos cómo calcular el valor con anticipación.

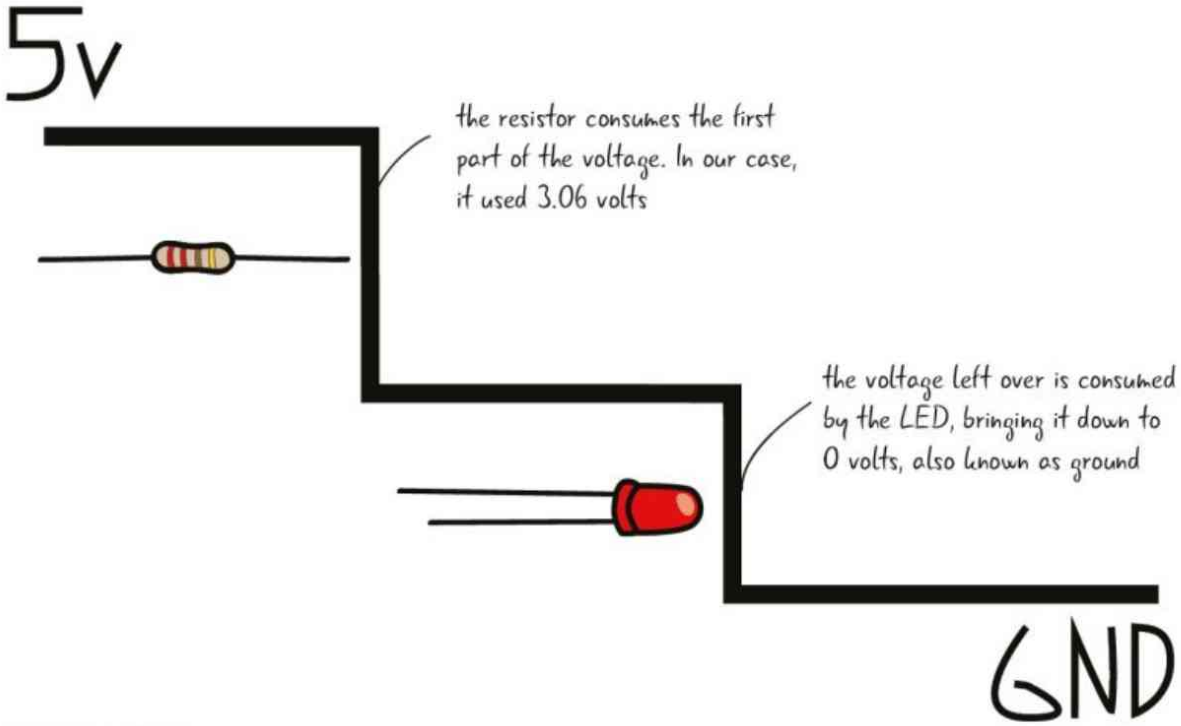


FIGURA 5.20 Visualización de la caída de voltaje.

Note La *caída de voltaje* es la cantidad de voltaje que consume un componente. Los componentes del circuito consumirán toda la tensión suministrada.

¿PREGUNTAS?

P: ¿Qué medimos cuando comprobamos el voltaje a través de un componente?

R: El valor en el multímetro para el voltaje es la diferencia entre el voltaje en un extremo del componente y el que hay en el otro lado. Esto nos permite ver cuánta tensión utiliza el componente.

P: ¿Así que la caída de voltaje en un componente se refiere a cuánta tensión está consumiendo ese componente?

R: Sí. Como vimos cuando medimos los componentes, cada uno de ellos consume parte del voltaje total del circuito.

P: ¿Qué ocurre si los componentes no consumen todo el voltaje?

R: Los componentes siempre consumirán todo el voltaje suministrado, y si se aplica una tensión más alta, los valores que consumen serán más altos.

Ahora que sabes lo que es el voltaje y cómo medirlo en un circuito, volvamos a la analogía del agua y veamos la corriente.

LA CORRIENTE: EL FLUJO

En nuestro modelo hidráulico, reflejado en la figura 5.21, podemos medir la cantidad de agua que fluye a lo largo de las tuberías. Si medimos la sección transversal de uno de los tubos en cualquier punto, podemos calcular la cantidad de agua que pasa a través de ella en un tiempo dado, por ejemplo, podríamos medir un litro por segundo. Todo el mundo se refiere generalmente a esto como corriente de agua, cuanta mayor cantidad de agua fluye en un tiempo dado, más fuerte es la corriente.

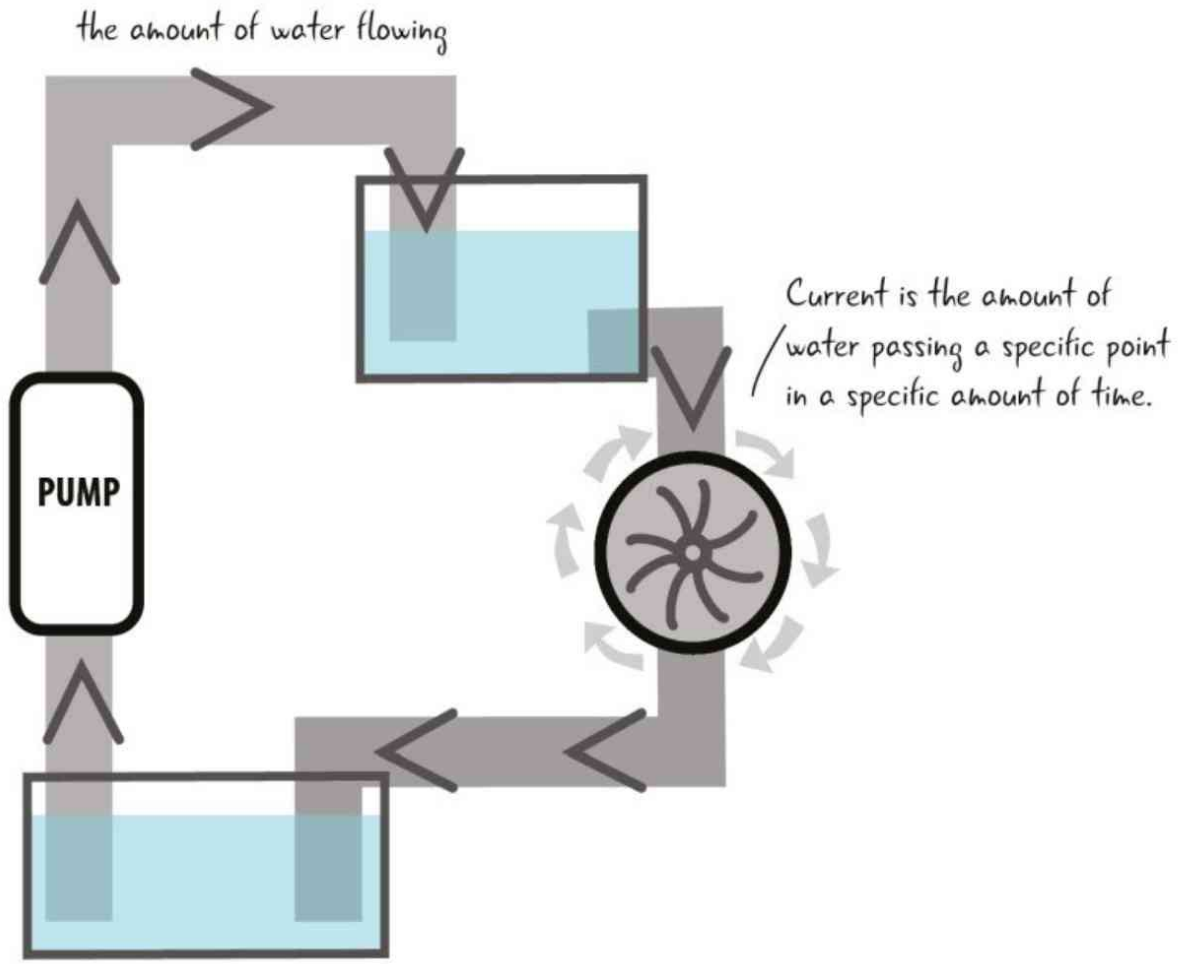


FIGURA 5.21 La corriente en la analogía hidráulica.

La palabra corriente significa más o menos lo mismo en un circuito. La corriente es la cantidad de carga eléctrica que atraviesa el circuito por segundo. La corriente se mide en amperios (también se conocen como amperes), y es por esto por lo que a la corriente también se le llama amperaje. La corriente requiere un trayecto completo y cerrado para poder fluir. Si el circuito no es un trayecto cerrado completo (digamos que hay un cable roto en el circuito), entonces no hay corriente. La corriente en el modelo eléctrico se muestra en la figura 5.22.

Note La corriente se mide en amperios, o amperes, que es la cantidad de electricidad que fluye por segundo.

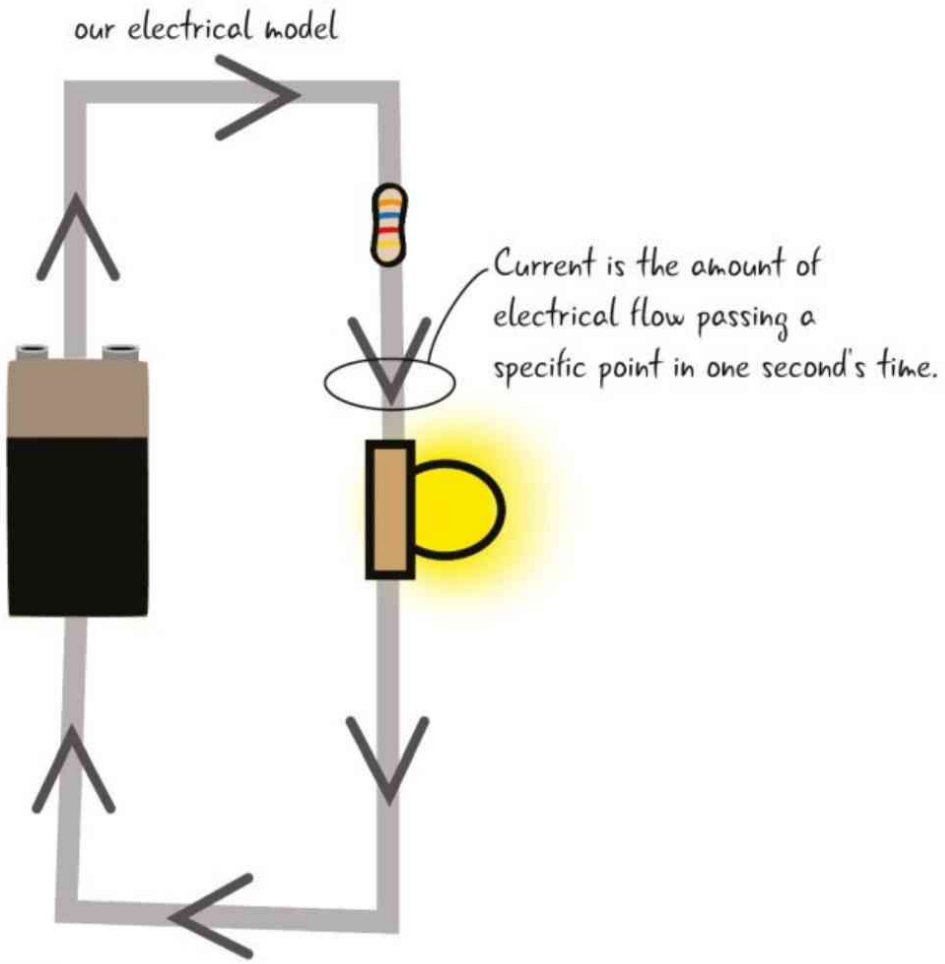


FIGURA 5.22 La corriente en el modelo eléctrico.

Corriente en el circuito

La cantidad de corriente en el circuito está determinado por dos cosas:

La resistencia de los componentes en el circuito

Los componentes que requieren más corriente normalmente tendrán menos resistencia. Más adelante, en este capítulo, ampliaremos las explicaciones sobre la resistencia.

Intensidad nominal de la fuente de alimentación

Con esta denominación nos referimos a la cantidad máxima de corriente que la fuente de alimentación puede suministrar. Puedes comprobar el valor nominal de la corriente (así como el voltaje) observando el valor nominal situado generalmente en la parte inferior de la fuente de alimentación, junto con

otra información. La figura 5.23 muestra la potencia de salida en la parte inferior de la fuente de alimentación. Recomendamos comprar una fuente de alimentación con una corriente de 500 miliamperios a 1 amperio y de 9 a 12 voltios de voltaje.

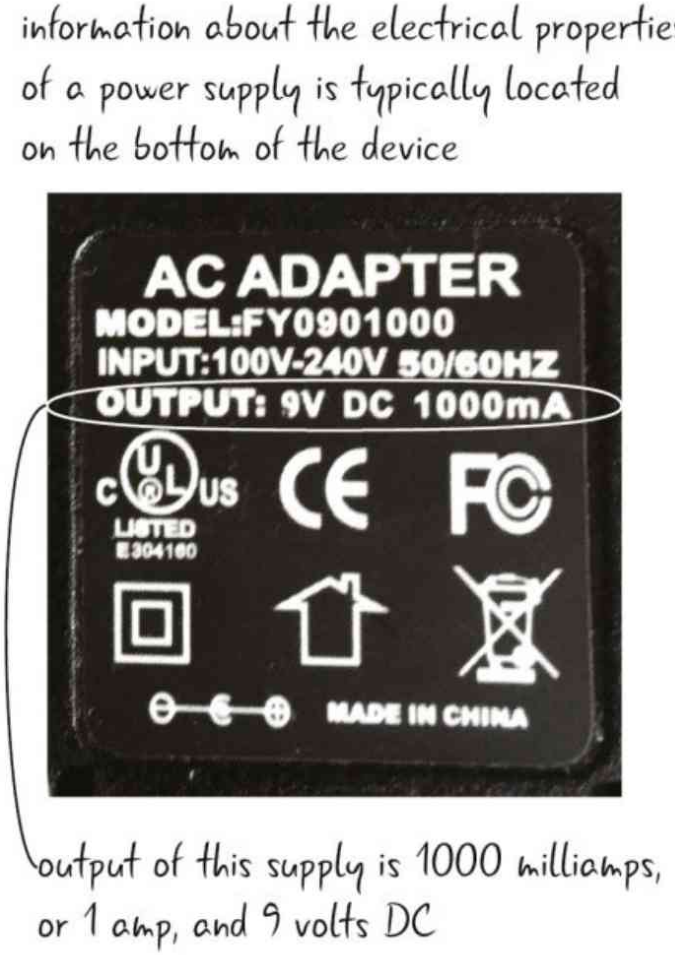


FIGURA 5.23: La potencia de salida figura en la parte inferior de la fuente de alimentación.

Todos los componentes tienen una corriente nominal, indicada en la caja o en su hoja de datos, que muestra la corriente que pueden soportar. Un componente no puede forzar a una fuente de alimentación a suministrar más corriente de la que el diseño de la misma le permite.

¿CUÁL ES EL LÍMITE DE CORRIENTE PARA Arduino?

La tarjeta de Arduino tiene un límite de entrada de corriente de un amperio.

Recomendamos comprar una fuente de alimentación de 500 miliamperios (1/2 amperio) a 1 000 miliamperios (1 amperio). El cable USB que conecta el Arduino con el ordenador proporciona 500 miliamperios (1/2 amperio), suficiente para hacer funcionar la placa de Arduino y proporcionar energía a los pines. Una fuente de alimentación con una potencia superior a un amperio podría dañar el equipo Arduino.

El Arduino solo puede proporcionar 40 miliamperios a cada uno de los pines de E/S. Hay otros componentes electrónicos que pueden ayudar al Arduino a llevar a cabo aplicaciones con valores más altos de corriente, pero 40 miliamperios son suficientes para alimentar los componentes que vamos a utilizar en este libro.

> **Note** La entrada de corriente en el Arduino tiene el valor máximo de un amperio. Los pines de E/S de tu Arduino solo proporcionarán un máximo de 40 miliamperios.

Veamos ahora el máximo valor de corriente que puede medir un multímetro.

MeDICIÓN DE LA COrrIENTE

Medir la corriente es más difícil que medir el voltaje, y se hace con mucha menos frecuencia, como parte del proceso de depuración, que medir el voltaje. Entonces, ¿por qué lo estudiamos? Es un ejercicio útil para aprender cómo fluye la corriente a través del circuito, y para entender la diferencia entre voltaje y corriente. El multímetro es la principal herramienta que se utiliza en el proceso de depuración, por eso queremos que sepas cómo utilizarlo para que puedas comprobar las magnitudes eléctricas.

Para medir la corriente tienes que extraer uno de los cables de un componente del circuito, como se muestra en la figura 5.24, para insertar el

medidor y hacerlo parte del trazado del circuito. En el circuito extraerás el ánodo del led. Como siempre, cuando realices ajustes en el circuito, tienes que estar seguro de que no está conectado a la corriente.

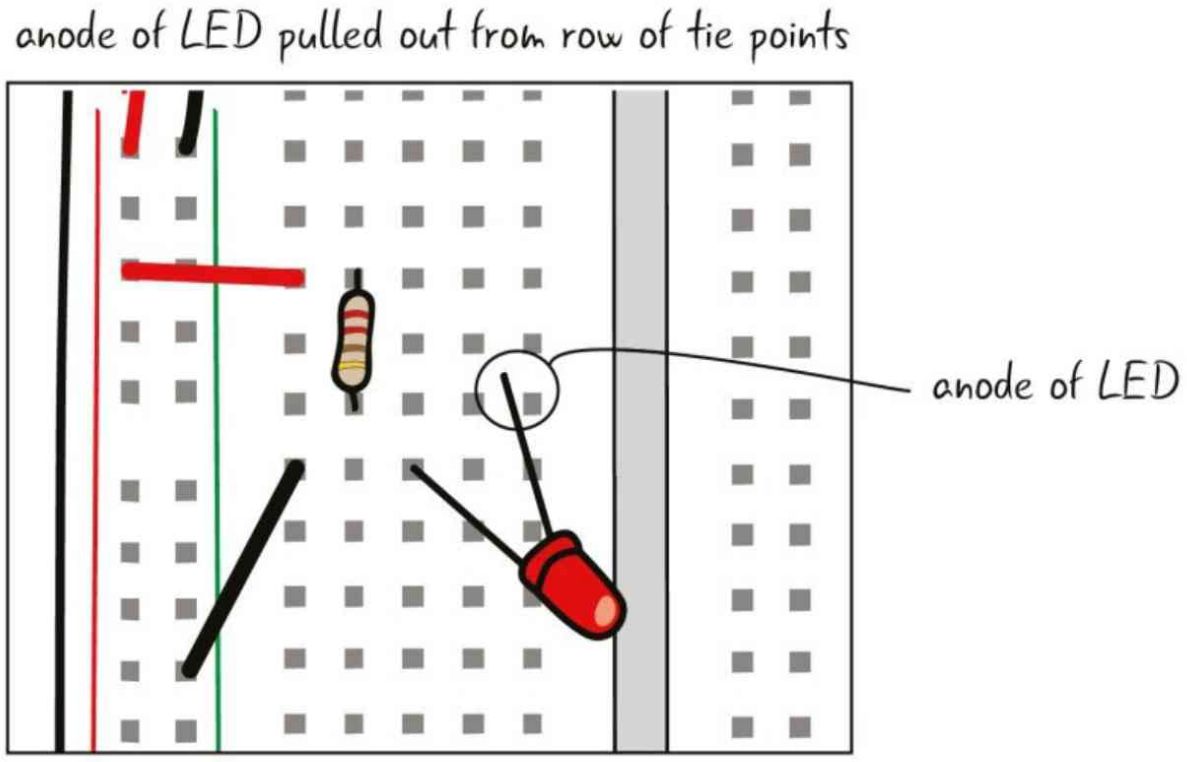


FIGURA 5.24 Saca el ánodo del led para preparar la placa y medir la corriente.

AJUSTE DEL MultÍMETRO

Necesitas mover el selector del multímetro para medir 200 miliamperios de amperaje de C.C. Al igual que cuando se mide el voltaje, para medir la corriente elegirás un valor superior al valor que esperas medir, 200 miliamperios es el máximo valor de corriente que puede medir el multímetro sin tener que mover las sondas (hablaremos de eso más adelante). Como no utilizas componentes que consumen mucha corriente, como puede ser el caso de los motores, puedes estar seguro de que 200 miliamperios es un valor superior al valor de la corriente a medir. Por lo tanto, mantén las sondas del multímetro conectadas a los mismos puertos, como se muestra en la

figura 5.25.

Ahora que tienes configurado el multímetro correctamente y el circuito dispuesto de manera que el ánodo del led se ha extraído de la fila de puntos de contacto, puedes volver a conectar el Arduino al cable USB. A continuación, con la sonda de color rojo del medidor toca el conductor de la resistencia que estaba en la misma fila de

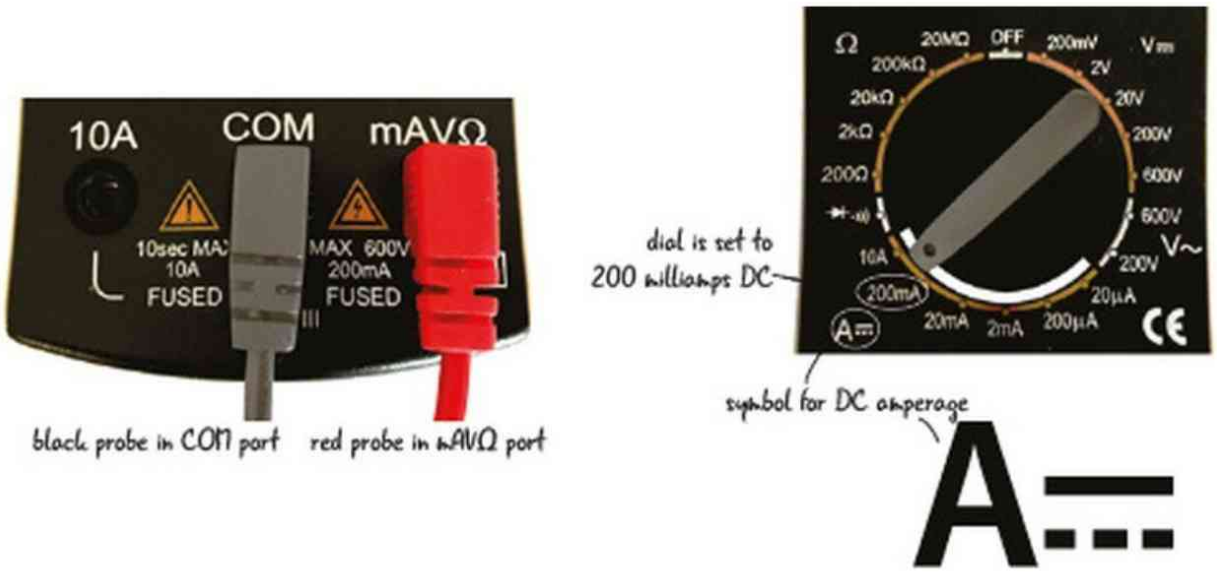


FigurA 5.25:**Preparación del multímetro para medir pequeños valores de corriente.**

puntos de contacto que el ánodo del led (extrae antes el ánodo para realizar este ejercicio), y con la sonda negra toca el ánodo del led, como se muestra en las figuras 5.26 y 5.27. El led debería iluminarse, porque el multímetro es ahora parte del trazado cerrado de tu circuito. Puesto que el multímetro se inserta en el circuito, mide la corriente (también conocida como amperaje). En el medidor se lee 14 miliamperios. Podrías leer un valor ligeramente distinto dependiendo en parte del color que sea el led.

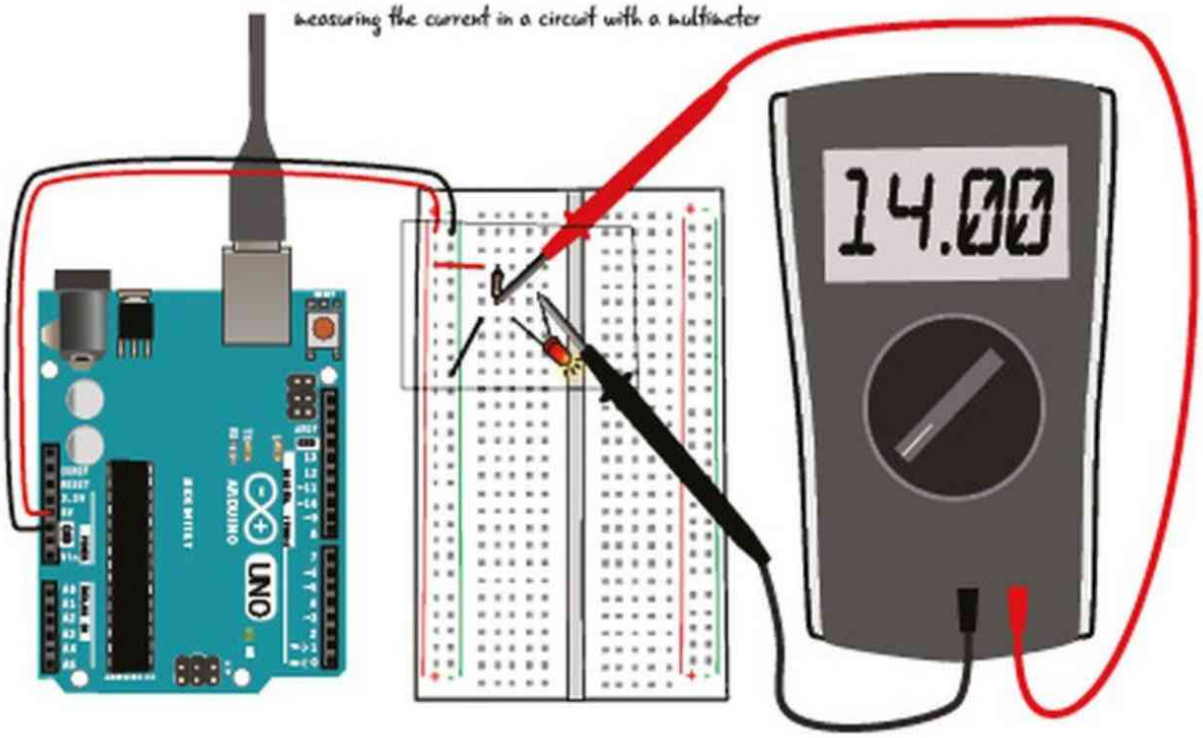


Figura 5.26:Las sondas tocan un extremo de la resistencia; el ánodo del led se extrae del punto de unión.

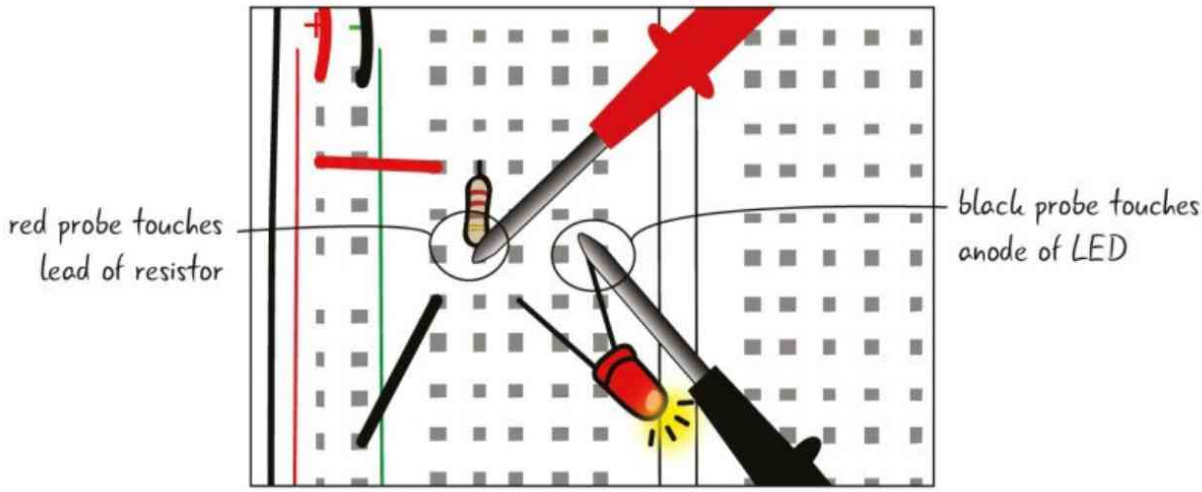


FIGURA 5.27 Detalle de la posición de las sondas para medir la corriente.

Warning **¡Hay que tener cuidado cuando se miden altos niveles de corriente!**

Mientras estés midiendo cantidades relativamente pequeñas de corriente, como son los 14 miliamperios que medimos en el circuito, ¿está bien tener la sonda roja conectada al puerto mAVΩ del multímetro (medidas de miliamperios, voltaje y resistencia). Sin embargo, si trabajas con corrientes más intensas (por encima de 200 miliamperios), necesitas hacer dos cosas para evitar que se achicharre el multímetro:

- Ajustar el selector del multímetro a 10 A.
- Mover la sonda de color rojo del puerto mAVΩ al puerto de 10 A.

Si olvidas estas dos cosas, el exceso de corriente puede dañar el multímetro. Recomendamos no medir valores de corriente superiores a 200 miliamperios.

Es una buena idea mantener la sonda roja en el puerto mAVΩ, que es el puerto adecuado para medir la mayoría de las características eléctricas.

¿PREGUNTAS?

P: ¿Podemos controlar la cantidad de corriente que fluye a través de los circuitos?

R: Sí, la cantidad de corriente la controlan los componentes conectados en el circuito. Controlar la cantidad de corriente es una cualidad importante para poder utilizar con seguridad componentes de mayor consumo de energía, como son los motores.

P: ¿Están relacionados la corriente y el voltaje?

R: Sí, lo están. Veremos una fórmula más adelante, en este capítulo, que explica su relación.

P: ¿Por qué hay un puerto separado en el medidor para corriente alta?

R: El medidor utiliza diferentes circuitos eléctricos internos para medir el voltaje y altos niveles de corriente para evitar que se dañe. Los niveles bajos de corriente (inferiores a 200 miliamperios) no dañarán el circuito de medición de voltaje, pero cualquier valor por encima de este puede causar problemas. Al cambiar el puerto, cambiamos el circuito que se utilizará dentro del medidor para hacer la medición.

P: ¿Por qué la medición de corriente requiere que extraigamos las patas de los componentes del circuito?

R: Para medir la corriente, el multímetro debe convertirse en un componente del circuito. Toda la corriente del circuito fluye a través del medidor para calcular la cantidad total. Explicaremos más adelante, en este capítulo, la relación entre el medidor y los componentes.

LA RESISTENCIA: LA reduCCIÓN DEL FLUJO

Veamos cómo se puede explicar la resistencia con la analogía hidráulica de la figura 5.28. Si las tuberías son más anchas en el sistema hidráulico, puede fluir más agua a través de ellas. Si son más estrechas, fluye menos agua. Se podría decir que la cantidad de resistencia, o la disminución de flujo, es mayor en tuberías más estrechas. Donde hay más resistencia, en el sistema con las tuberías más estrechas, la turbina gira más lentamente y produce menos trabajo.

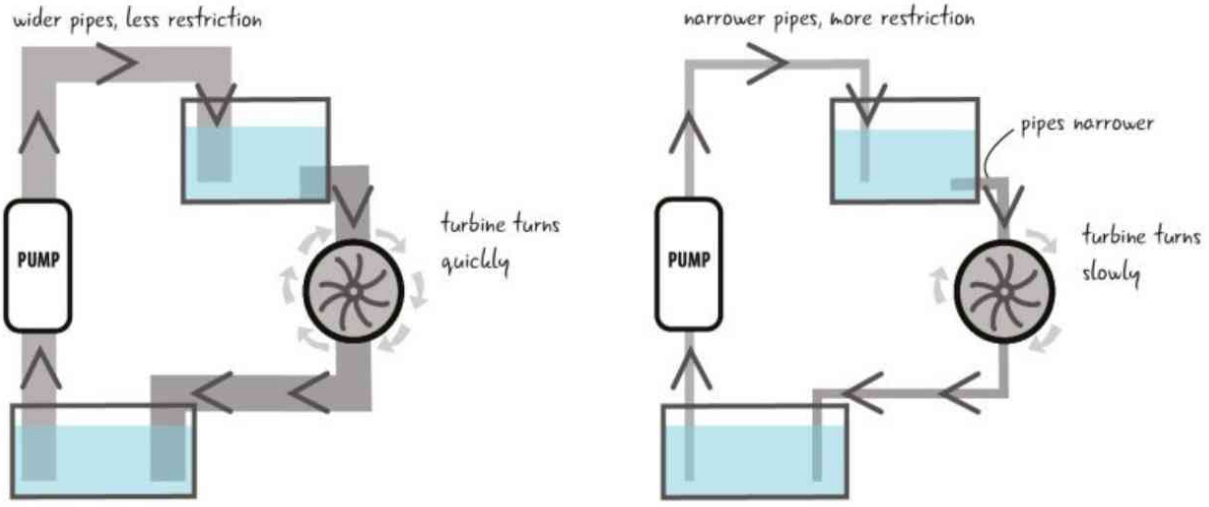


FIGURA 5.28 La resistencia en el modelo hidráulico.

> **Note** En circuitos, la *resistencia* se refiere a en qué medida un material restringe el flujo de la electricidad.

En un circuito, las resistencias son equivalentes a tubos estrechos porque restringen el flujo de electrones. En los diagramas del sistema eléctrico de la figura 5.29, la imagen de la izquierda tiene solo una resistencia y entonces la luz brilla intensamente. En la imagen de la derecha hay tres resistencias, lo que hace que la resistencia aumente y esto hace que la luz brille menos.

La resistencia se mide en ohmios, y se representa por el símbolo omega que se muestra a la izquierda. En este capítulo, veremos más adelante cómo los ohmios se relacionan con las otras características eléctricas, pero por el momento, solo sabemos que una resistencia tiene un valor que indica cuánto se opone al paso de la electricidad.

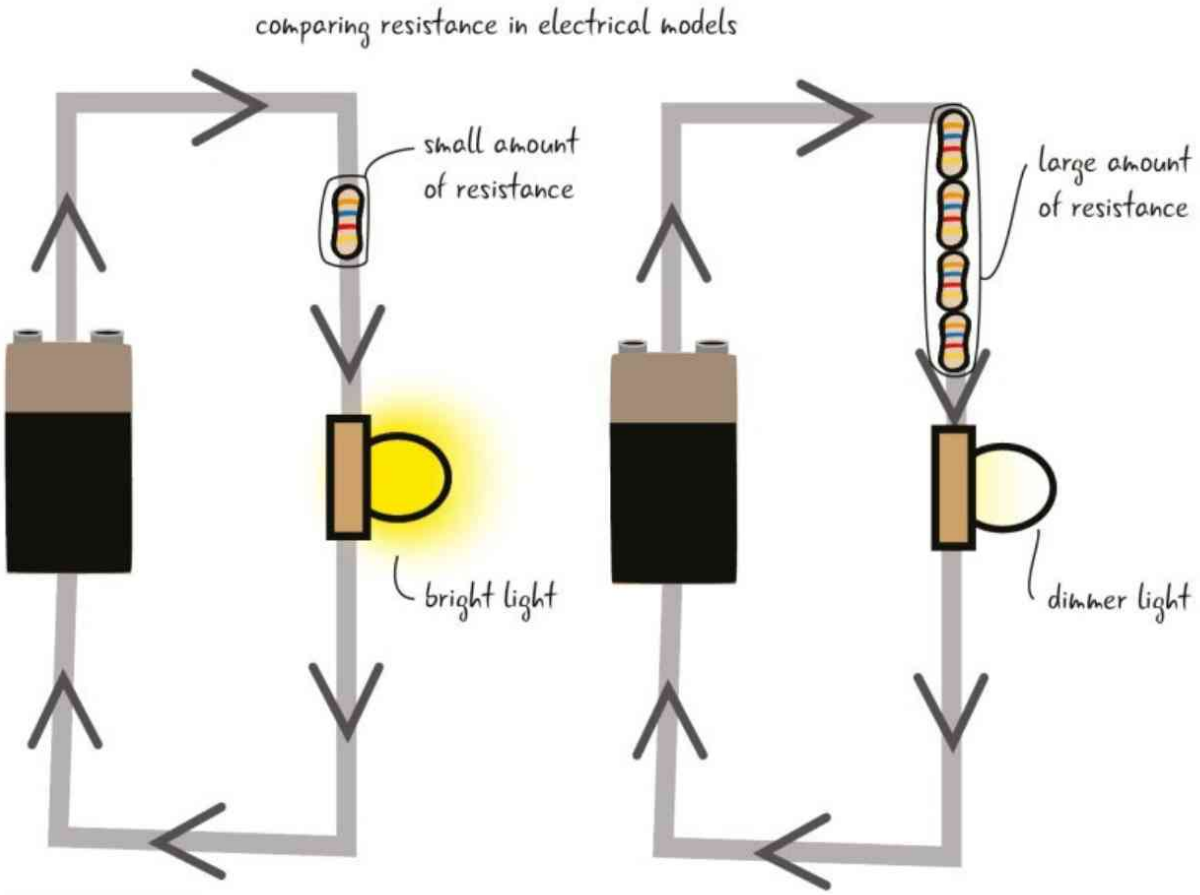


FIGURA 5.29 La resistencia en el modo eléctrico.

LAS ResistENCIAS DE CERCA

Como has visto a lo largo del capítulo, el voltaje y la corriente de los circuitos varían dependiendo de los componentes que lo forman. Los componentes electrónicos pueden ser muy sensibles a los picos de electricidad. Además, si tienes una fuente de voltaje que puede suministrar una potencia mayor de la que un componente puede recibir, podría dañarlo. ¿Cómo puedes proteger los componentes electrónicos del circuito? La respuesta es mediante las resistencias. La figura 5.30 muestra un grupo de resistencias de 220 ohmios. Ya has utilizado resistencias para proteger los ledes de la energía de 5 V que suministra el Arduino.

FIGURA 5.30 Conjunto de resistencias.

Si la resistencia es una propiedad de todos los componentes electrónicos, ¿por qué necesitamos un componente de resistencia especial? Las resistencias son geniales porque se presentan en un amplio rango de valores diferentes y pueden ayudar a controlar el flujo de la electricidad en el circuito. Ya has visto circuitos en los que hemos utilizado resistencias de 220 ohmios, pero los circuitos que vamos a ver en el libro van a necesitar resistencias con diferentes valores. ¿Cómo puedes saber el valor de cualquier resistencia? Existen un par de procedimientos. Veamos su valor utilizando un multímetro.

Medida de la Resistencia con el multímetro

Mide el valor de una resistencia fuera del circuito. Se trata de algo diferente a

lo que has visto al medir el voltaje o la corriente dentro del circuito. Ahora vas a medir una resistencia de 220 ohmios.

En el multímetro, la sonda negra debe estar conectada al puerto COM, y la sonda roja debe estar en el puerto etiquetado con mAVΩ.

Mueve el selector a la sección en la que se mide la resistencia. Para este ejemplo, sitúa el selector en 2 KΩ. La disposición correcta se muestra en la figura 5.31.

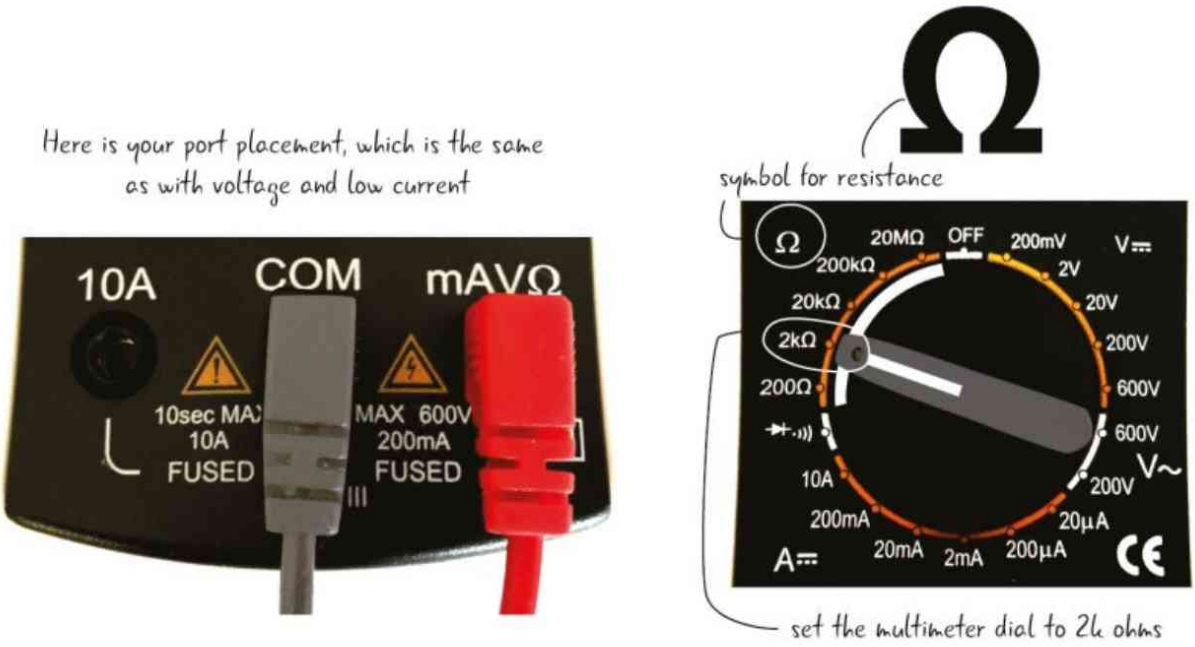


FIGURA 5.31 Multímetro preparado para medir la resistencia.

Has aprendido cómo ajustar el rango cuando has medido el voltaje. También es necesario ajustar el rango cuando mides la resistencia. Sabes que la resistencia es de 220 ohmios, Así que debes situar el selector en un valor mayor. La posición de 200 ohmios será demasiado baja. Sitúa el selector en 2 kΩ; estás buscando un valor entre 200 ohmios y 2 000 (2 K) ohmios. Ahora que has ajustado el dial y sabes que las sondas están conectadas en los puertos adecuados, estás preparado para medir la resistencia.

Toca con las sondas del multímetro cada uno de los extremos metálicos de la resistencia, como se muestra en la figura 5.32. Cuando se mide la

resistencia, no importa en qué lado de la resistencia está cada sonda. Puede que tengas que sujetar los conductores de la resistencia para conseguir un punto de contacto seguro con las sondas, o puedes situar la resistencia plana sobre una mesa. ¿Qué valor presenta el multímetro? El multímetro debería presentar un valor próximo a .221, ya que es una medida en kiloohmios. Hay que recordar que como el medidor está ajustado para medir 2 k ohmios, o 2 000 ohmios, .221 k ohmios es realmente igual a 221 ohmios.

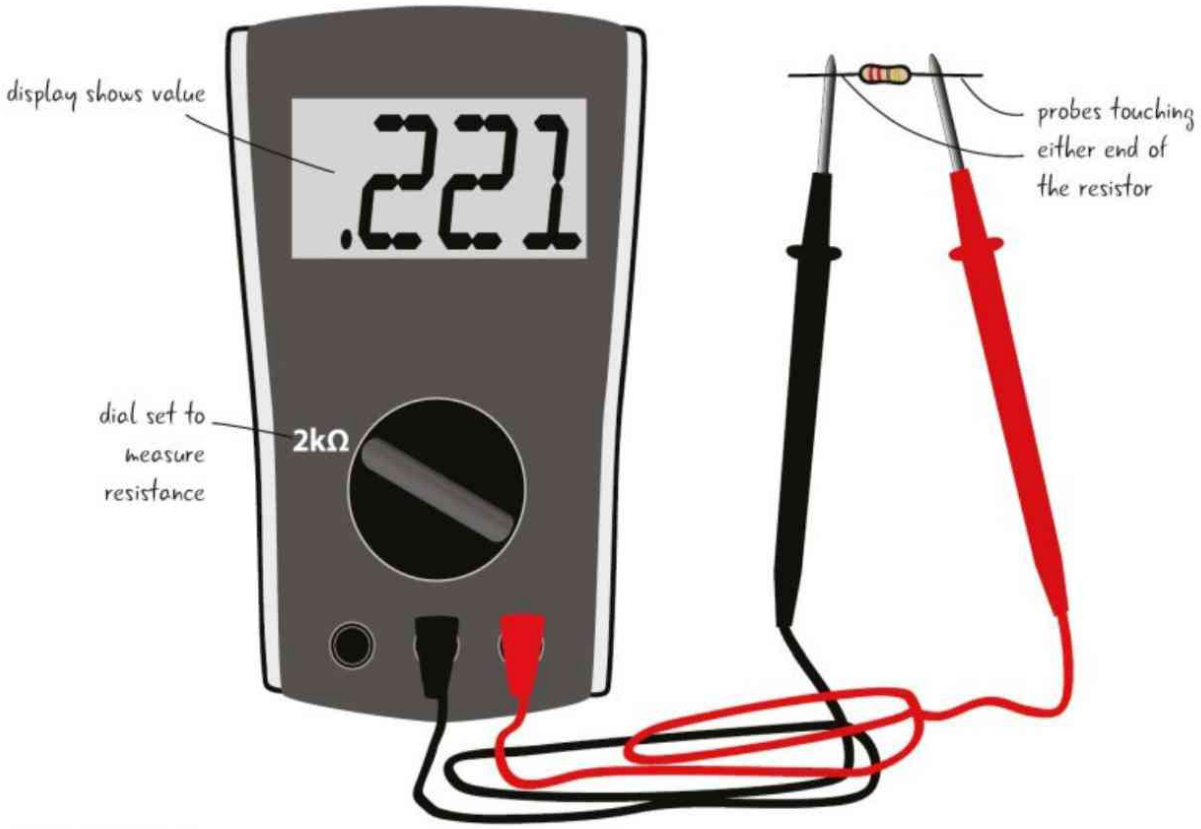


FIGURA 5.32 Medida de la resistencia con el multímetro.

El valor de la resistencia aparecerá en la pantalla del medidor. La figura 5.33 muestra a qué se parece la lectura de nuestro medidor.

FIGURA 5.33 La pantalla del multímetro muestra el valor de la resistencia.

¿Por qué el valor es ligeramente diferente de los 220 ohmios del valor de la resistencia? Es porque el valor de la resistencia tiene una tolerancia, que indica el rango de precisión de la resistencia. Las resistencias que utilizarás en los proyectos con Arduino pueden tener valores reales que son más o menos un 10 % diferentes del valor indicado. En términos generales, trabajas con componentes que no son lo suficientemente sensibles como para que estas diferencias sean importantes, así que no te tienes que preocupar por esas desviaciones.

Las resistencias contienen un conjunto de bandas de color que ayudan a identificar su valor y su precisión. En el apéndice se explica cómo leer estas bandas de color.

¿PREGUNTAS?

P: ¿Qué tienen que ver el voltaje, la corriente y la resistencia con el Arduino?

R: Cuando trabajas con el Arduino, montas circuitos que usan electricidad. Si entiendes cómo funcionan el voltaje, la corriente y la resistencia, te ayudará a depurar tus circuitos y, finalmente, a montar proyectos más complejos.

Voltaje, Corriente Y Resistencia: Repaso

Veamos por última vez el diagrama de la analogía hidráulica (figura 5.34), después repasaremos rápidamente las propiedades que acabas de aprender, qué unidad de medida se utiliza con cada una y el símbolo que se emplea para representarla.

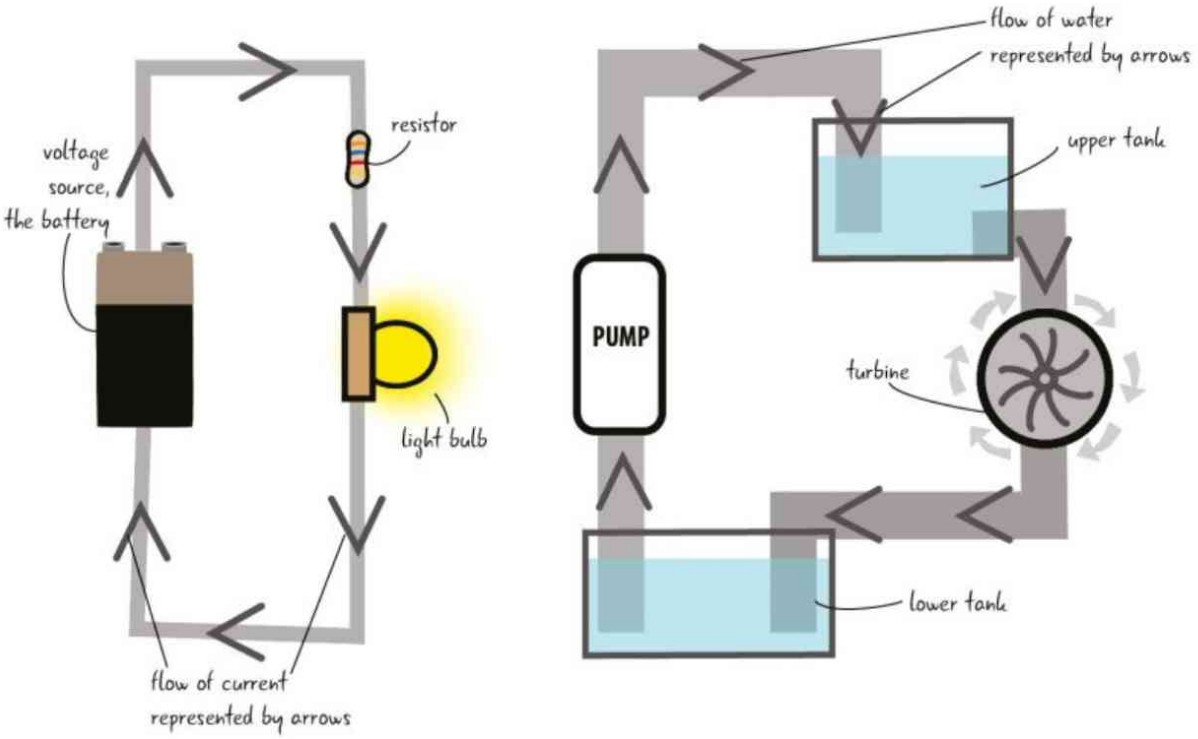


FIGURA 5.34 Analogía hidráulica para la electricidad.

La tabla 5.1 repasa las propiedades eléctricas junto con sus símbolos y las unidades en las que se miden.

Tabla 5.1:**Características eléctricas**

NOMBRE	DescripCIÓn	UniDAD	SÍMbolO
Voltaje	Fuerza electromotriz, o flujo de potencial	Voltio	V
Corriente	Cantidad de flujo eléctrico	Amperio, o ampere	A

Resistencia	Restricción de flujo eléctrico	Ohmio	Ω

¿CÓMO AFECTA LA electriciDAD A LOS COMPONENTES?

En un circuito, la corriente, el voltaje y la resistencia están relacionados. Si un sistema tiene corriente, entonces hay necesariamente voltaje y resistencia. Examinemos qué sucede cuando reduces solo el valor de una de estas características.

Voltaje

Recordemos que el voltaje representa el potencial para que la electricidad se mueva en un circuito. El voltaje siempre se establecerá entre la carga más alta y la carga más baja hasta que alcanza el estado de equilibrio cero, también conocido como tierra. Si colocamos el mismo led y la misma resistencia en el circuito, y lo alimentamos usando solamente 3,3 voltios en lugar de los 5 voltios a los que estamos acostumbrados, el led brillará menos. Si continuamos reduciendo el voltaje, el led continuará reduciendo su brillo, hasta que

finalmente se apaga.

Corriente

La corriente es la característica relacionada con el movimiento de electrones en el circuito. La corriente es lo que estimula los componentes. ¿Qué sucede si no tenemos suficiente corriente en el circuito? Sin la suficiente corriente no hay suficientes electrones para activar los componentes. Cuando las pilas de una linterna están agotadas, tienen muy poca corriente para encender la bombilla. Si reducimos la corriente del circuito añadiendo resistencias, el led se apagará repentinamente una vez que desaparece la corriente mínima necesaria para encenderlo.

Resistencia

La resistencia es una medida de cómo un material reduce el flujo de la electricidad. Naturalmente, todos los materiales tienen una cierta resistencia, pero si esta es demasiado alta, el flujo eléctrico se detendrá por completo. Sin embargo, si hay muy poca resistencia los componentes se ven sometidos a una excesiva cantidad de corriente y se pueden quemar. A menudo utilizamos resistencias para restringir el flujo con el fin de preservar los otros componentes de los circuitos. Si añadimos más resistencias o cambiamos su valor en el circuito básico del led, aumentaremos el valor de la resistencia y reduciremos la cantidad de electricidad que llega al led, quizá incluso limitando la capacidad del led para emitir luz.

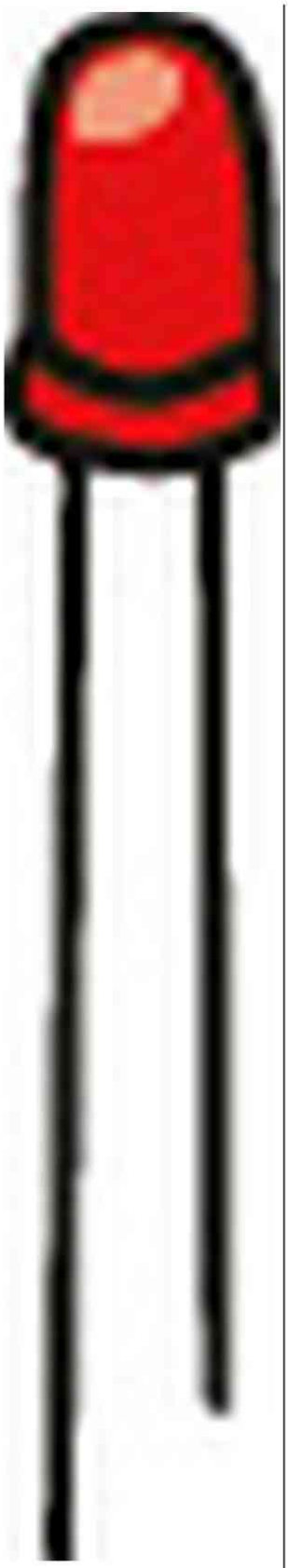
¿CÓMO AFECTA A LOS ComponentEs UN CAMBIO DE

LAS PROPIEDADES ELÉCTRICAS?

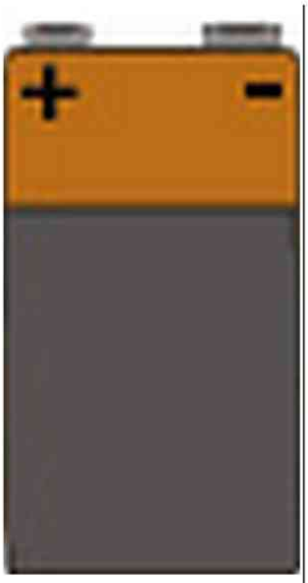
Echemos un vistazo rápido a cómo se ven afectados los componentes por los cambios en las características eléctricas en la tabla 5.2.

TablA 5.2:**Efectos en los componentes de los cambios en las características eléctricas**

COMPONENTE	ImageN	VoltaJe	COrrIEntE	ResistEncIA
Led		El led se atenuará a medida que el voltaje disminuye, o se iluminará más a medida que aumenta el voltaje. Si hay demasiado voltaje, el led se quemará.	Los ledes solo necesitan una pequeña cantidad de corriente para funcionar. Sin embargo, reducir demasiado la cantidad de corriente, apagará el led.	La resistencia de los ledes es muy pequeña.

				
Resistencia		El voltaje se	Las resistencias	La cantidad de

		convierte en calor cuando se aplica a una resistencia. Mayor voltaje implica más calor y menos voltaje implica menos calor.	reducen la cantidad de corriente que se consume en un circuito.	resistencia depende del valor nominal de la resistencia. Consulta el apéndice para aprender a identificar los valores de resistencia por los colores.
Battery		Las baterías	La corriente proviende	Dado que la batería no es

		imponen los niveles de voltaje, tanto el nivel alto como el de cero voltios, también conocido como tierra.	de la batería. El flujo de corriente cambiará dependiendo de los compnentes que estén conectados a la batería y de la corriente que necesiten.	un conductor perfecto, hay una pequeña cantidad de resistencia dentro de la batería, pero cuando está conectada en un circuito se considera prácticamente cero.

Ahora veamos cómo el voltaje, la corriente y la resistencia interactúan entre sí en una regla llamada ley de Ohm.

¿CÓMO INTERACTÚAN EL VoltaJe, LA COrrIEntE Y LA RESISTENCIA? LA LEY DE OHM

El voltaje, la corriente y la resistencia están relacionados a través de una fórmula conocida como ley de Ohm. La ley de Ohm, mostrada en la figura 5.35, indica que en un circuito dado, el voltaje (en voltios) es igual a la corriente (en amperios) multiplicado por la resistencia (en ohmios).

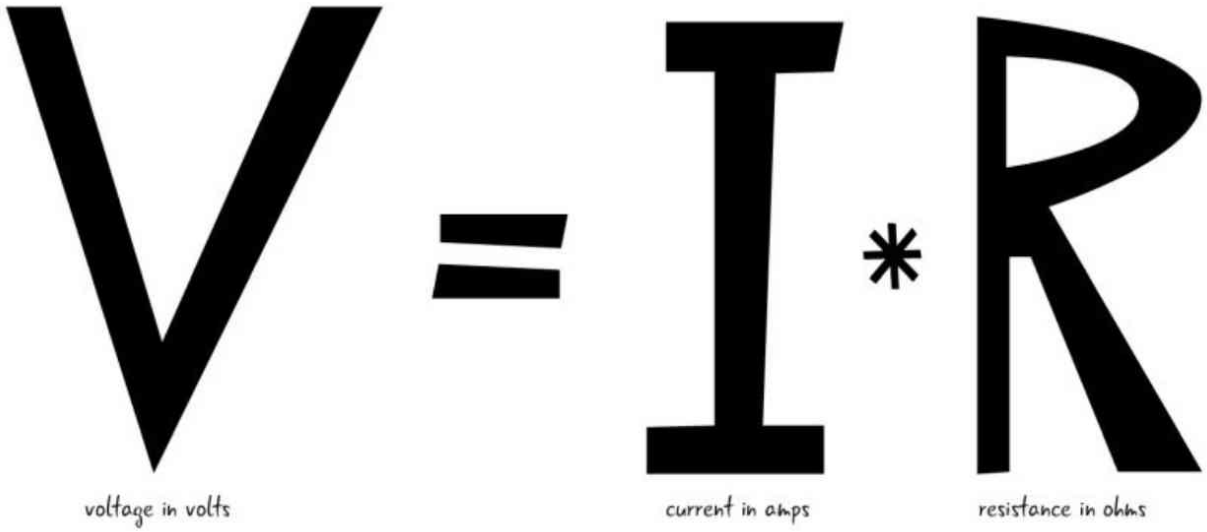


FIGURA 5.35 La ley de Ohm.

Esta ecuación nos muestra que independientemente del valor de la presión (voltaje), si la resistencia es alta, la corriente disminuye. Esto es cierto para todo el cableado eléctrico.

Un beneficio de la ley de Ohm es que si conocemos dos de las características eléctricas, siempre podemos calcular el valor de la tercera. Podemos ver estas relaciones en la figura 5.36.

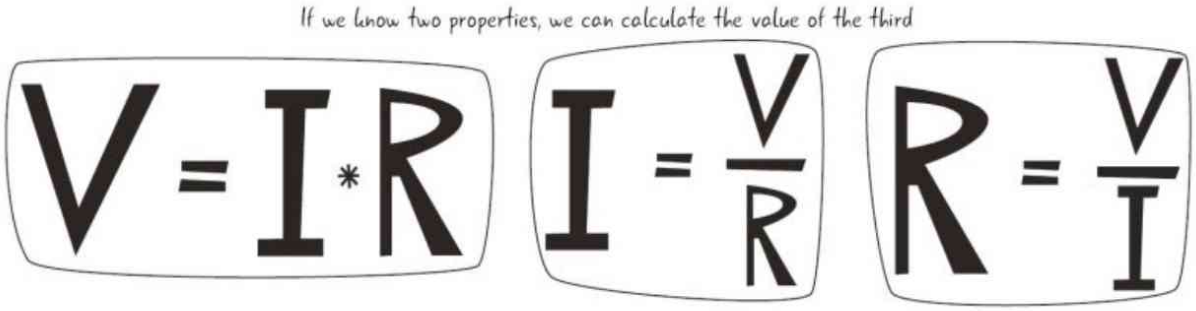


FIGURA 5.36 Combinaciones de la ley de Ohm.

Como ya has aprendido a medir los valores de voltaje y corriente en los circuitos, puedes calcular el voltaje, la corriente o la resistencia , siempre y cuando conozcas el valor de dos de las tres características.

LA LEY DE OHM EN LOS CIRCUITOS

Ahora que conoces la ley de Ohm, ¿cómo te puede ayudar a montar los circuitos? Puedes utilizar la ley de Ohm para determinar el valor de las resistencias que necesitas. También puedes utilizar la ley de Ohm como medida de seguridad para confirmar que los valores de voltaje y corriente que atraviesan los componentes están por debajo de los límites de dichos componentes.

Por ejemplo, si tienes una resistencia en el circuito cuyo valor es de 220 ohmios, y hay 20 mA (que es lo mismo que decir 0,020 amperios) que circulan por el circuito, entonces puedes utilizar la ley de Ohm para calcular cuánto voltaje consumirá la resistencia. La figura 5.37 muestra los cálculos.

$$V = IR$$

$$V = (0.020\ amps) * 220\ ohms$$

$$V = 4.4\ Volts$$

FIGURA 5.37 Aplicación de la ley de Ohm.

APLICACIÓN DE LA LEY DE Ohm

¿De qué otra forma puedes utilizar la ley de Ohm? Supongamos que quieres montar dos circuitos que contengan cada uno de ellos un led y una resistencia. Alimenta el circuito a través del pin de 3,3 voltios del Arduino, y el otro con el pin de 5 voltios (recuerda que el Arduino puede proporcionar los dos voltajes). Los ledes que vas a utilizar en los circuitos necesitan 2,2 voltios para iluminarse completamente, y utilizan 25 miliamperios, o 0,025 amperios. Debido a la diferencia de voltaje entre los dos circuitos, necesitarás diferentes resistencias en cada circuito para proteger los ledes. ¿Qué valor de resistencia necesitarás para cada circuito?

Como ya sabes que el led de cada circuito necesita 2,2 voltios, puedes calcular la diferencia entre los voltajes suministrados (3,3 voltios y 5 voltios) y 2,2 voltios para calcular el voltaje que caerá en cada resistencia (figura 5.38).

FIGURA 5.38 Cálculo del voltaje.

Ahora puedes utilizar la ley de Ohm para calcular el valor de la resistencia que necesitas para que el voltaje y la corriente indicados de 0,025 amperios pasen a través de la resistencia que protege el led (figura 5.39).

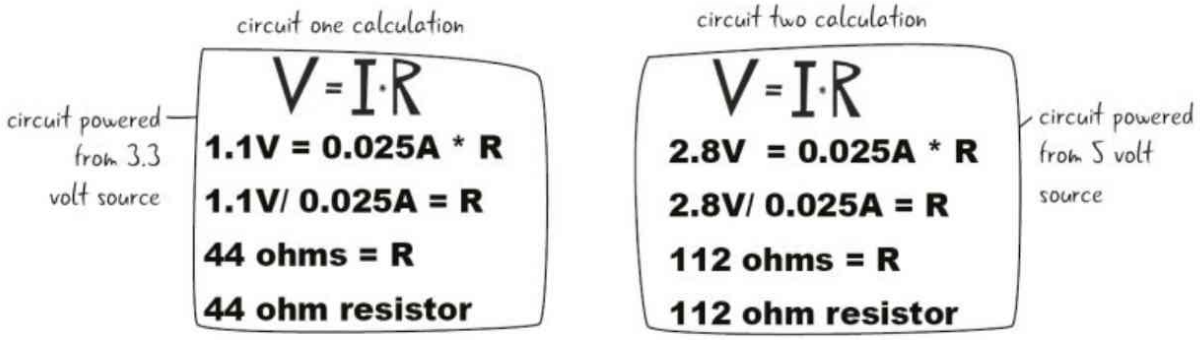


FIGURA 5.39 Cálculos con la ley de Ohm.

El circuito de 5 voltios necesita una resistencia de valor superior a la que

necesita el circuito de 3,3 voltios. Puedes ver a través de la ley de Ohm que debes cambiar el valor de la resistencia en función del voltaje que utilices. La ley de Ohm es útil para asegurarte de que la corriente que circula por los componentes sea la adecuada.

DISPOSICIÓN DE LOS ComponentEs EN LOS CIRCUITOS

¿Cómo sabes cómo ordenar los componentes en un circuito? Sabes que el circuito debe describir un trazado completo. Algunos componentes paracen estar dispuestos uno al lado del otro con puntos eléctricos comunes, mientras que otros están conectados con un terminal a continuación del otro. ¿Qué son estas disposiciones y qué efecto tienen en las características eléctricas del circuito?

ComponentEs En ParalelO Y EN Serie

Veamos el orden de disposición de los componentes en un circuito. Veremos la disposición en paralelo en primer lugar.

Orden de los componentes en un Circuito: disposición en Paralelo

Los componentes en paralelo se colocan uno al lado del otro y comparten puntos de contacto eléctricos, como se muestra en la figura 5.40. La electricidad fluye a lo largo de cada trayecto a través de los componentes dispuestos en paralelo.

Orden de los Componentes en un Circuito: disposición en Serie

En contraste con la disposición anterior, los componentes en serie se suceden uno tras otro, como se puede ver en la figura 5.41. Los circuitos que has montado hasta ahora se han dispuesto en serie. Toda la electricidad que pasa a través de la resistencia, entra en el led.

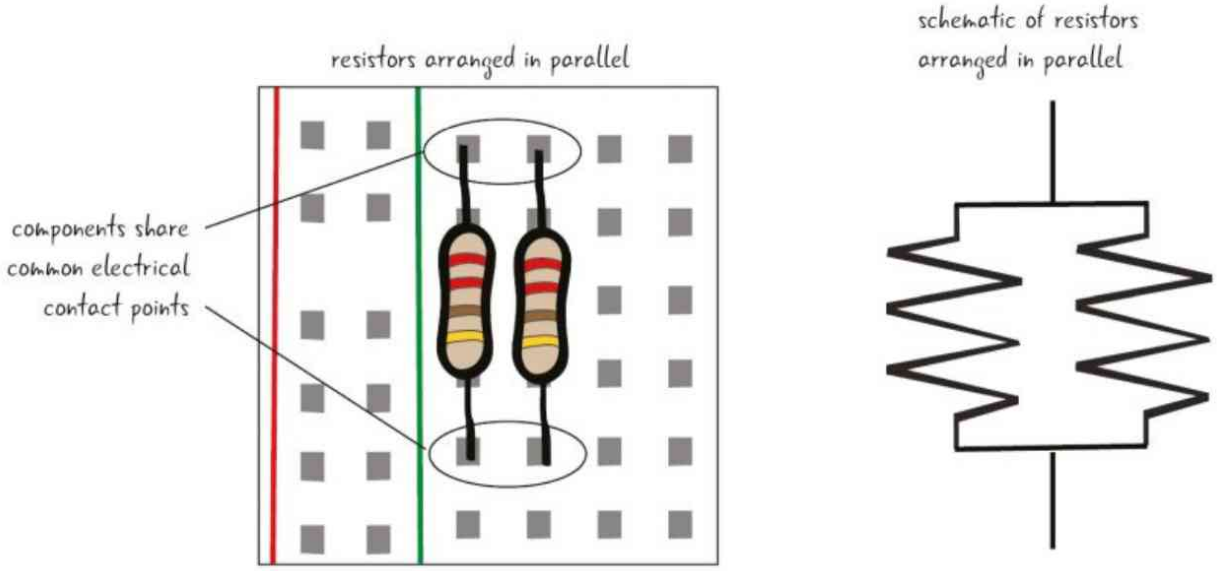


FIGURA 5.40 Resistencias colocadas en paralelo.

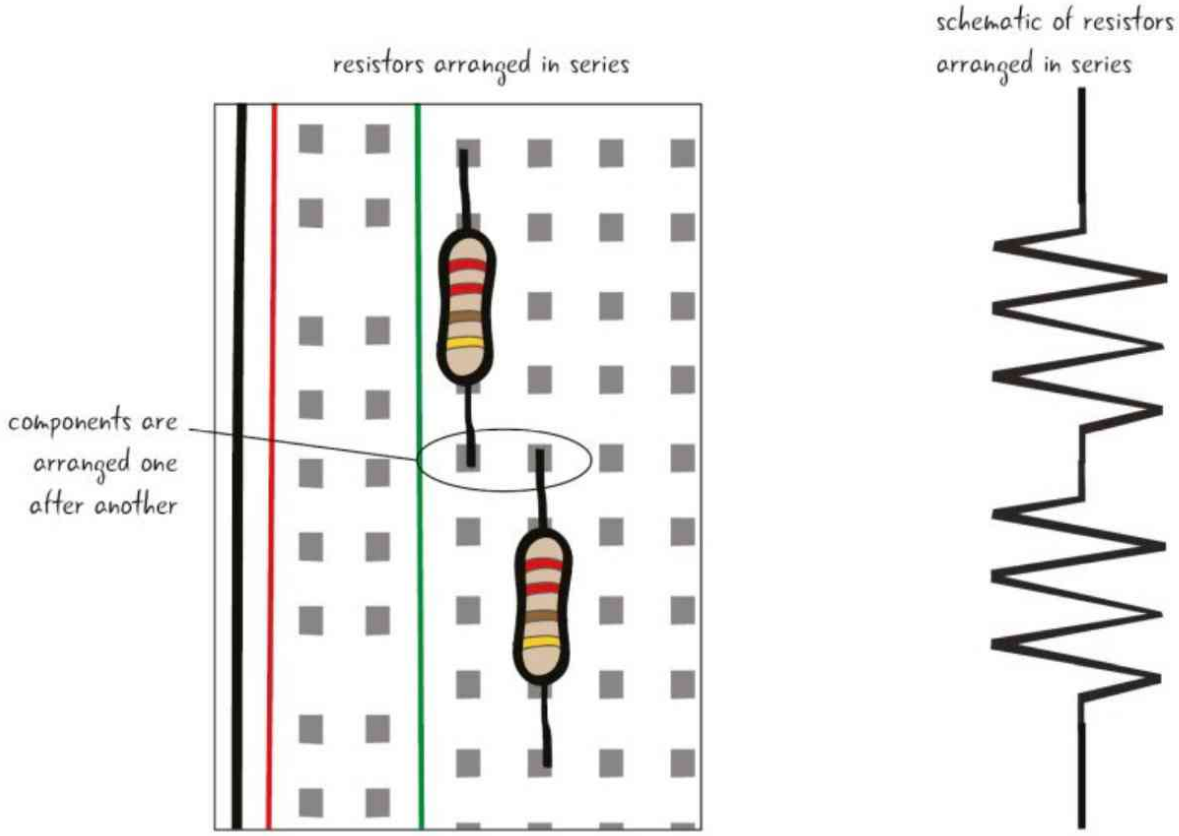


FIGURA 5.41 Resistencias colocadas en serie.

Para ver exactamente lo que entendemos por componentes en serie y paralelo, te mostraremos cómo añadir un led más al circuito básico, primero en paralelo y después en serie. Luego medirás la caída de voltaje en cada uno de los ledes.

Circuito con dos LEDes en Paralelo

Añadirás este led en paralelo con el primer led, como se muestra en la figura 5.42. La disposición de componentes en paralelo significa que los componentes están conectados a puntos eléctricos comunes. Podemos decir que los componentes están uno al lado del otro. Veamos el esquema de un circuito con los ledes dispuestos en paralelo. Puedes ver que la resistencia está conectada a 5 voltios y también a ambos ledes.

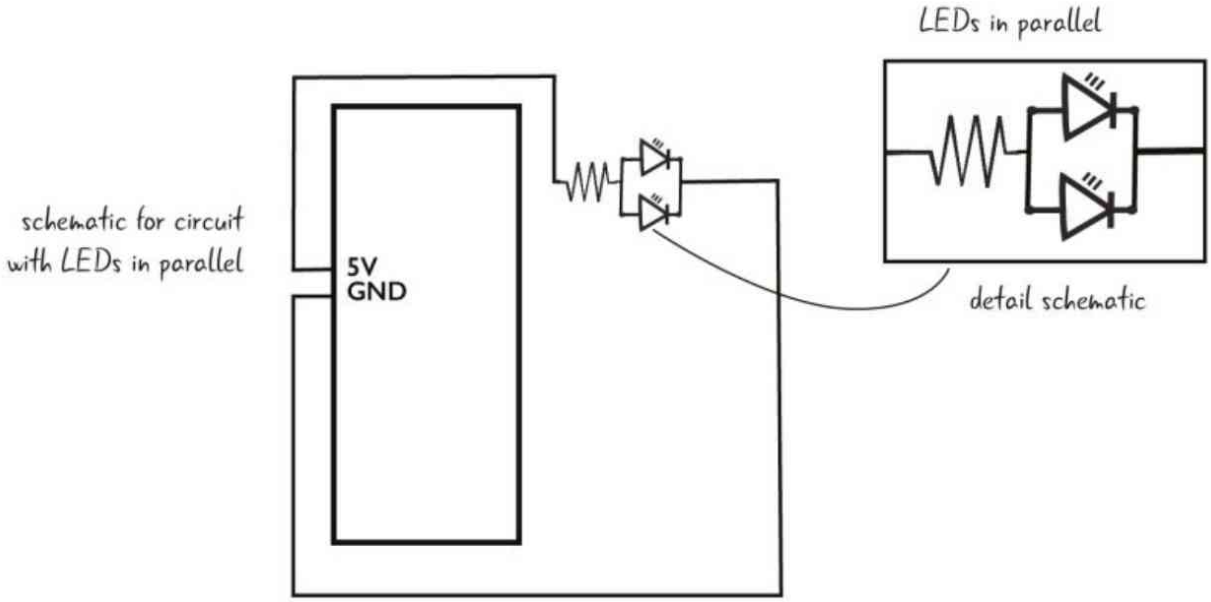


FIGURA 5.42 Esquema de un circuito con ledes en paralelo.

Incorporación de un segundo led al circuito

Para montar este circuito, añade un segundo led a la placa de pruebas de manera que los ánodos de ambos ledes estén en una fila de puntos de conexión y los cátodos en otra fila diferente. Los dos ánodos están ahora conectados a un mismo terminal de la resistencia, y los dos cátodos están conectados por un puente a tierra, como se muestra en la figura 5.43. Recuerda que hay que desconectar el ordenador antes de hacer cualquier cambio en el circuito.

Has añadido el segundo led, así que estás casi preparado para comprobar el voltaje de este circuito.

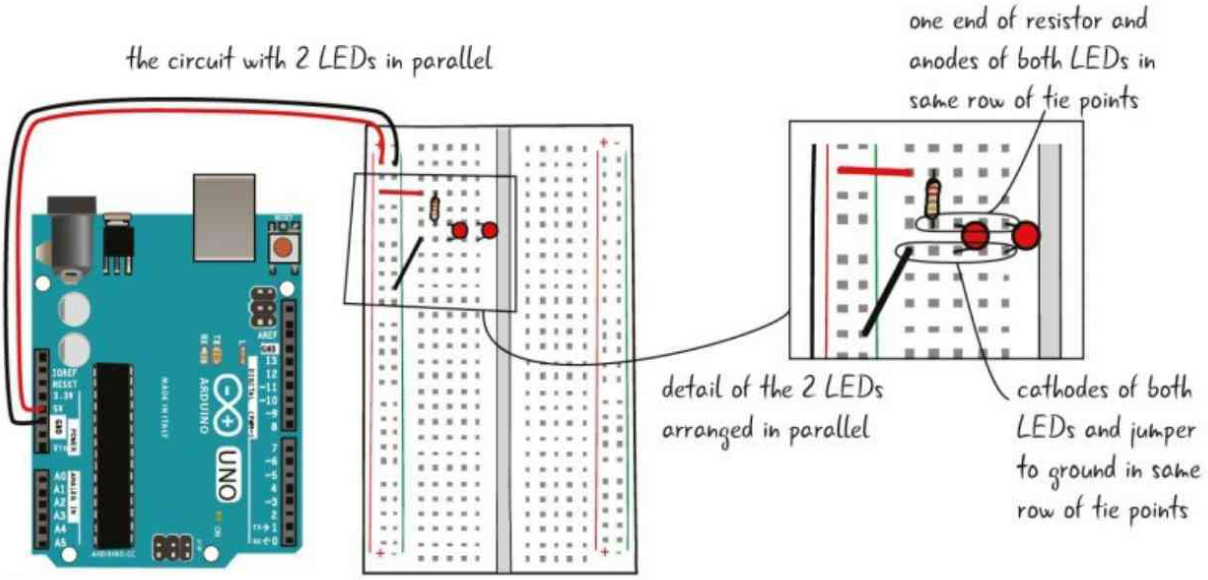


FIGURA 5.43 Incorporación del segundo led en paralelo.

MeDICIÓN DEL VoltaJe DE LEDes En ParalelO

Cuando tengas el led colocado correctamente en la placa de pruebas, vuelve a conectar el Arduino al ordenador. A continuación ajusta el selector del multímetro para medir 20 voltios. Coloca la sonda roja en el ánodo de uno de los ledes, y la sonda negra en el cátodo del mismo led, como se indica en las figuras 5.44 y 5.45.

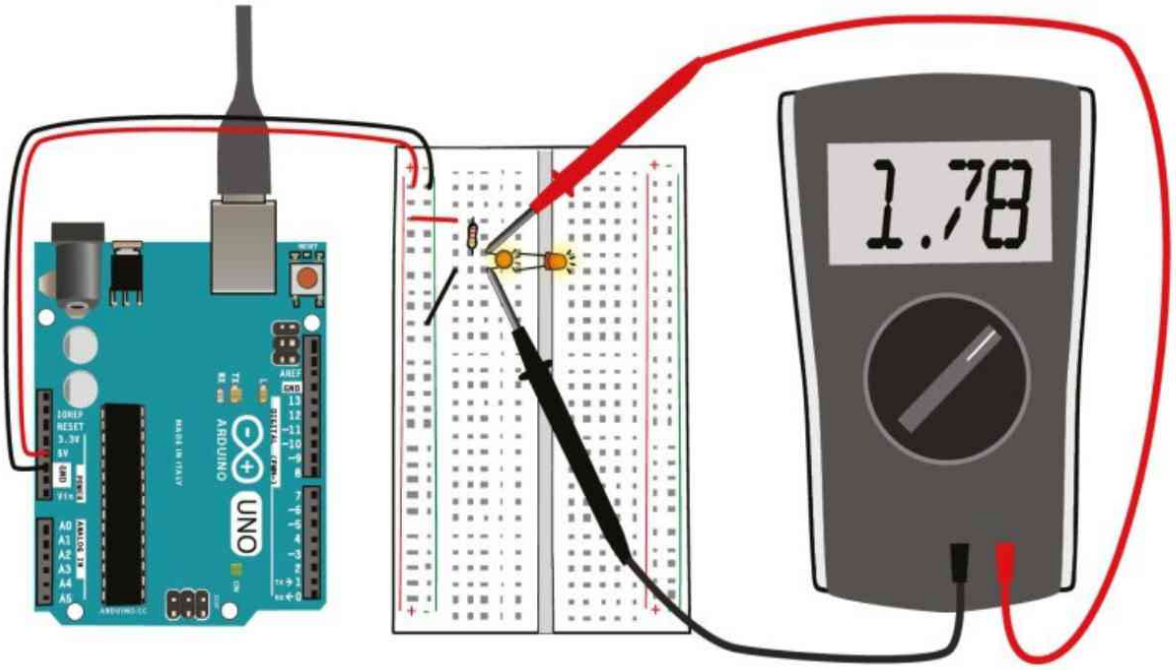


FIGURA 5.44 Medición del voltaje de un componente en paralelo.

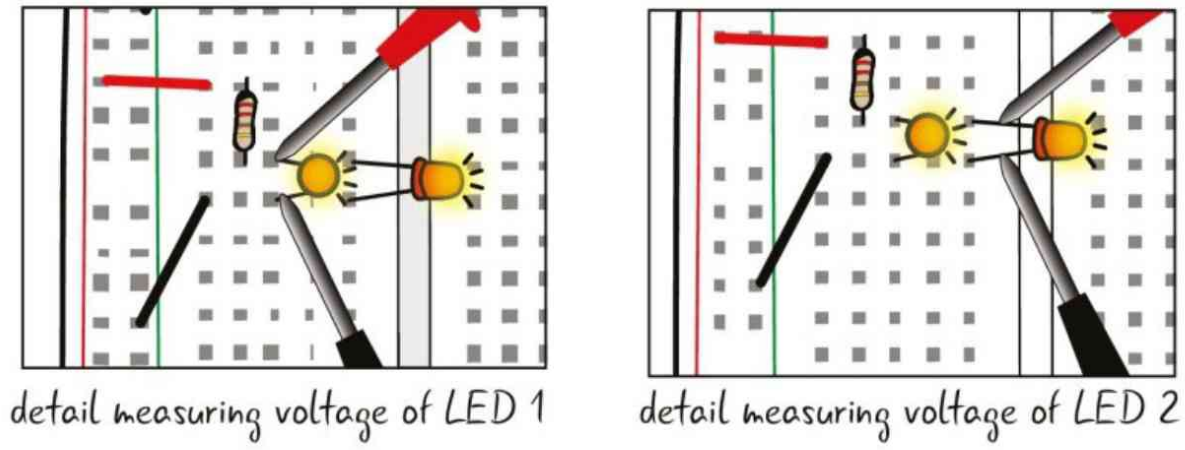


FIGURA 5.45 Medición del voltaje de los ledes 1 y 2.

La pantalla del medidor debe presentar una lectura de aproximadamente 1,78 voltios para ledes de color rojo (si no es exactamente igual es porque los ledes que estás usando puede que sean distintos a los que utilizaste antes para montar el circuito). Después de haber comprobado el voltaje de uno de los ledes, comprueba el otro, como se muestra en la figura 5.45. Si utilizas ledes idénticos, el valor de la caída de voltaje debe ser exactamente el mismo para ambos ledes. No necesitas medir el voltaje a través de la resistencia

porque será el mismo valor que tenía en el circuito básico.

> **Note** En paralelo, ambos ledes tienen el mismo voltaje.

el Multímetro en Paralelo

Ya has visto que los ledes comparten los mismos puntos de contacto, al igual que el multímetro. Cuando utilizas el voltímetro para medir el voltaje, el multímetro está en paralelo con el componente cuyo voltaje estás midiendo (figura 5.46).

> **Note** Para medir el voltaje en un circuito coloca el multímetro en paralelo con el componente que estás midiendo.

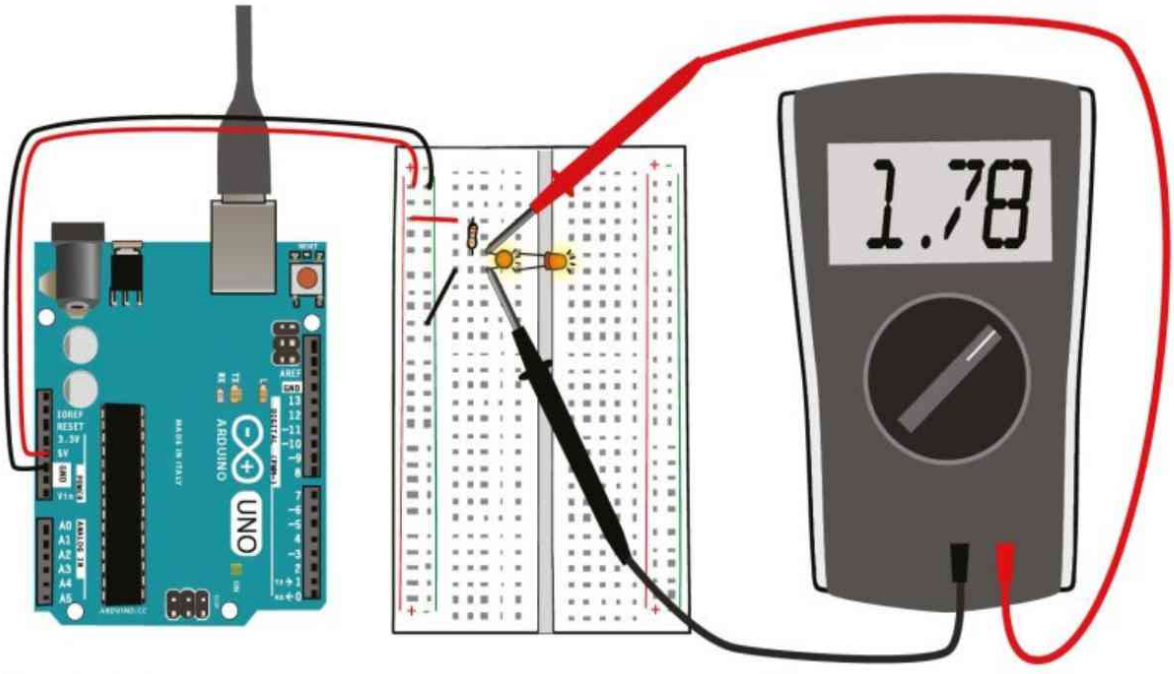


FIGURA 5.46 El multímetro está en paralelo con los ledes.

Componentes en Paralelo: ¿qué Efecto tienen sobre el Voltaje?

Ya sabes que los componentes en paralelo comparten los mismos puntos de contacto eléctricos. La corriente eléctrica utilizará todos los caminos posibles desde el principio de un circuito hasta el final. Como has podido ver con la medición del voltaje, todos los componentes en paralelo tendrán el el mismo voltaje entre sus extremos (figura 5.47).

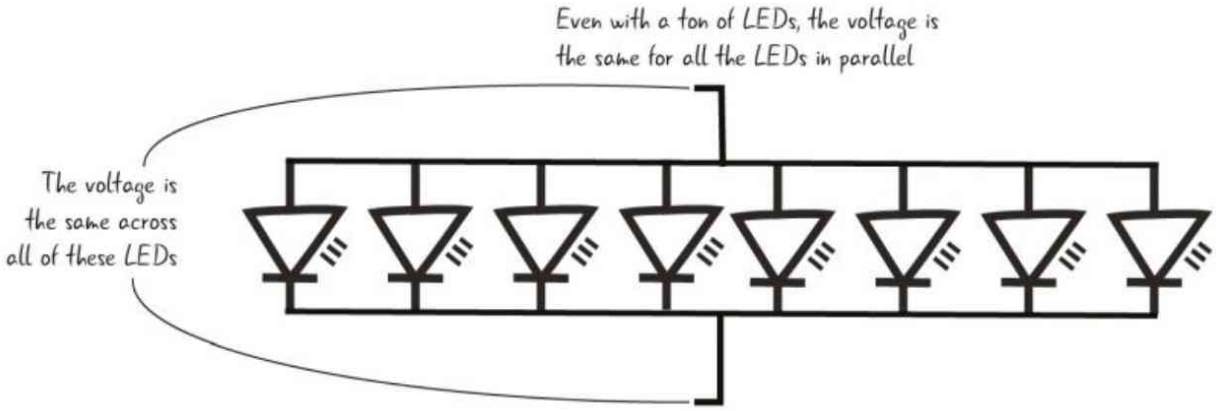


FIGURA 5.47: Muchos ledes en paralelo.

Si quieres que varios ledes tengan el mismo brillo, puedes colocarlos en paralelo; sabes que todos recibirán el mismo voltaje, sin que este cambie por el número de ledes conectados. Sin embargo, solo puedes hacer lucir unos cuantos ledes en paralelo utilizando el Arduino, ya que estás limitado por la cantidad de corriente que este puede suministrar.

> **Note** Todos lo componentes en paralelo estarán sometidos al mismo voltaje.

MONTAJE DE UN CIRCUITO CON DOS LEDeS en serie

Ahora vamos a configurar el circuito, colocando los ledes en la llamada

disposición serie. Los componentes que están en serie se colocan uno detrás de otro. Es fácil observar esto en la figura 5.48, en la que se puede ver un led detrás de otro; de hecho, la resistencia también está en serie en esta disposición. La mayoría de los circuitos serán una combinación de componentes dispuestos en serie y en paralelo.

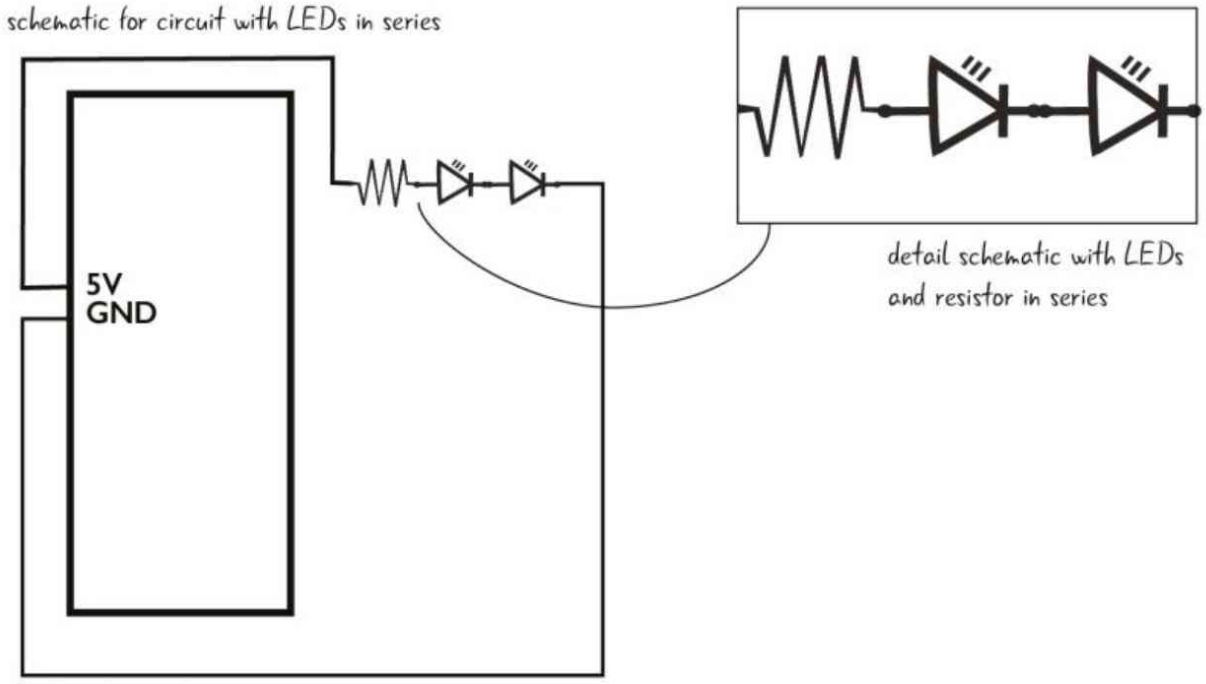


FIGURA 5.48 Esquema del circuito con los ledes en serie.

Empieza por desenchufar el equipo Arduino del ordenador. Para colocar el segundo led en serie con el primero, coloca el ánodo (terminal largo) del segundo led en la misma fila de puntos de contacto que el cátodo (terminal corto) del primer led. Coloca el cátodo del segundo led en una fila separada de puntos de contacto. Tendrás que mover el puente que va a tierra para que quede en la misma fila de puntos de contacto que el cátodo del segundo led, como se indica en la figura 5.49. (Recuerda: el circuito tiene que seguir un trazado completo para que funcione.)

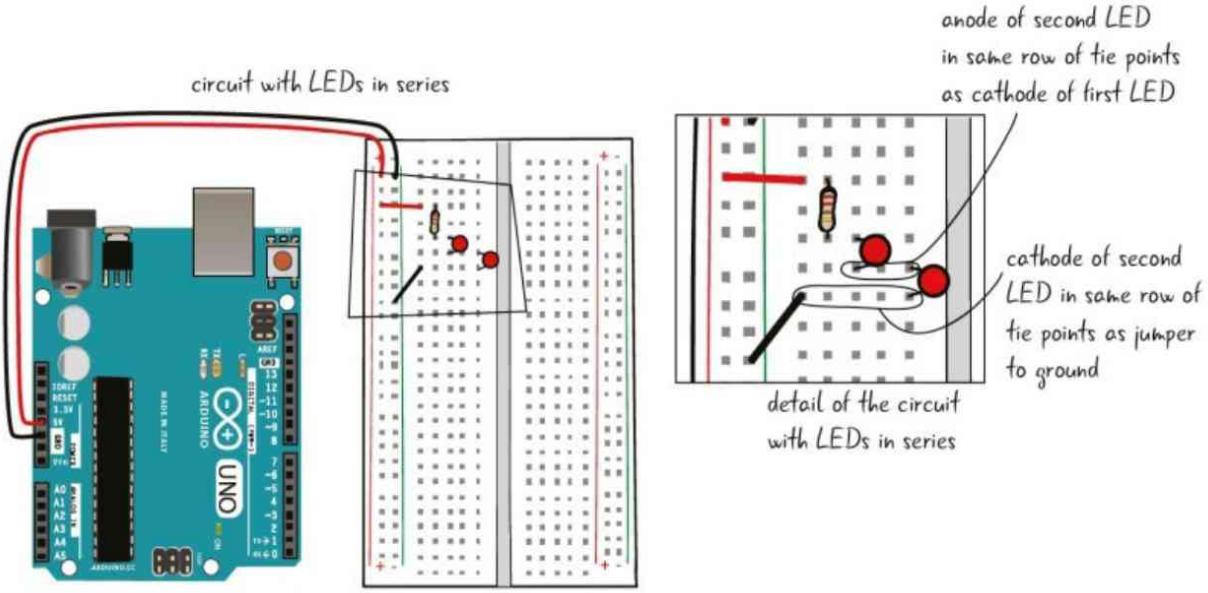


FIGURA 5.49 Instalación del segundo led en serie.

Una vez montado el circuito, estás preparado para medirlo.

Medida del Voltaje de Componentes en Serie

La medición del voltaje de componentes que están en serie es prácticamente igual que la medición de componentes que están en paralelo. Enchufa el equipo Arduino al ordenador y ambos ledes deberían encenderse. Con el selector de nuevo en 20 V, coloca la sonda de color rojo en el ánodo de uno de los ledes y la sonda de color negro en el cátodo del mismo led, como se indica en la figura 5.50. El voltaje debería ser de 1,77 voltios. A continuación mide el voltaje del otro led; el resultado debe ser similar al que obtuviste para el primer led. Finalmente, mide el voltaje de la resistencia (para nosotros el valor era de 1,38 voltios). La figura 5.51 muestra un detalle de la medición de los componentes del circuito.

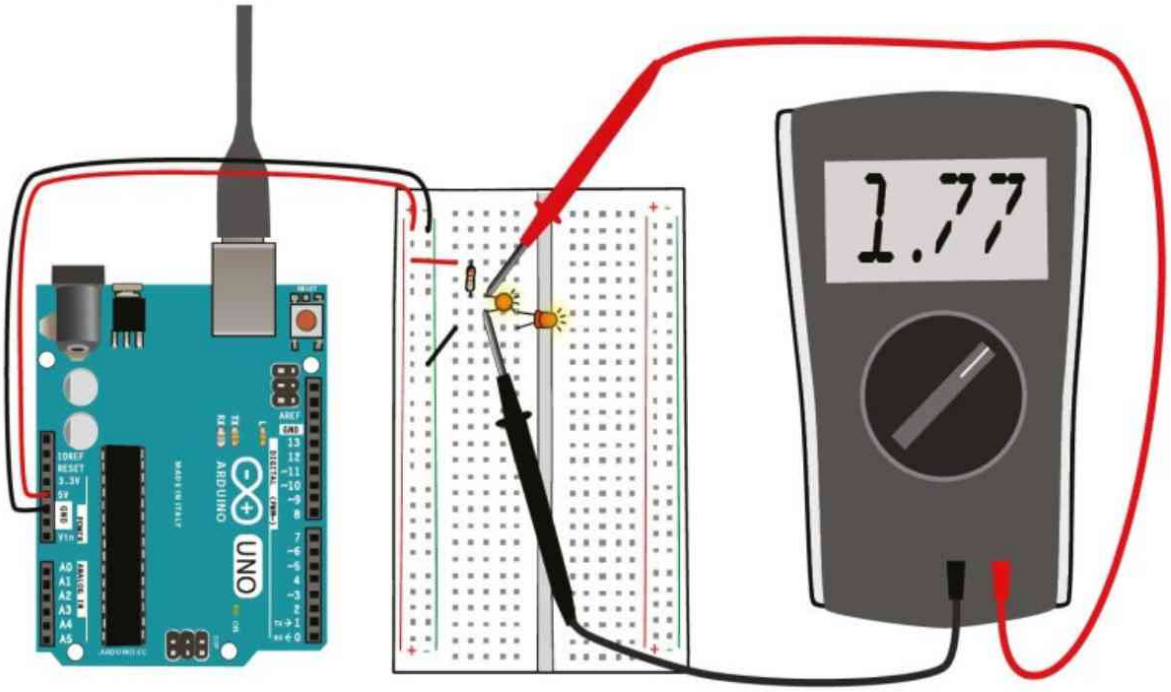


FIGURA 5.50 Medición del voltaje de componentes en serie.

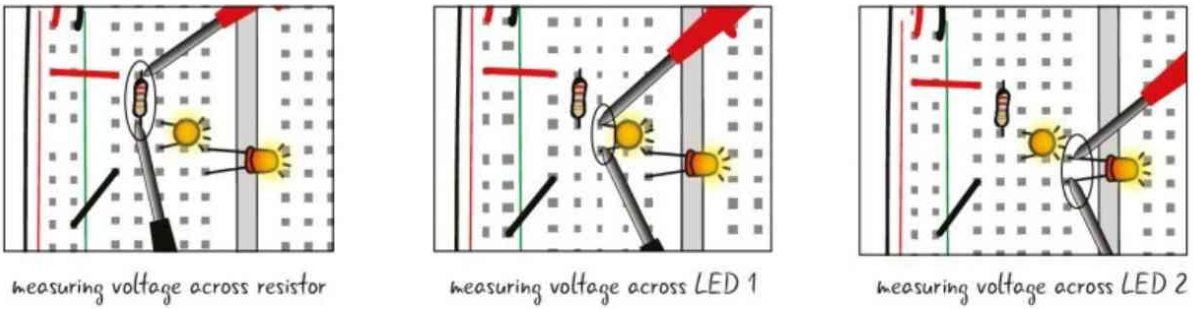


FIGURA 5.51 Detalles de la medición del circuito.

¿PREGUNTAS?

P: ¿Por qué son tan parecidos los valores del voltaje tanto en el circuito en serie como en paralelo?

R: En este caso, no apreciamos una diferencia de voltaje tan grande en serie y en paralelo como podríamos encontrar en un circuito diferente. Creemos que era importante mostrar las diferencias en la forma en la que estos componentes están dispuestos y cómo fluye la electricidad en estos circuitos, y ayudarte a familiarizarte con el uso del medidor.

Componentes en Serie: ¿qué efectos tienen sobre el Voltaje?

En el circuito, la electricidad debe pasar por la resistencia antes de que llegue al primer led. Como viste en las mediciones realizadas con el voltímetro, el voltaje va cayendo a medida que pasa por cada componente. Aunque el voltaje a través de cada led es más o menos el mismo que el del led del circuito básico, el voltaje a través de la resistencia cae también. La resistencia consume menos voltaje en este ejemplo de la disposición en serie debido a que los dos leds del circuito consumen voltaje. Debes tener en cuenta que el valor de la resistencia no cambia, pero dado que cada led ahora requiere su propio voltaje, la resistencia consume una porción más pequeña del voltaje total. La figura 5.52 representa la caída de voltaje en el circuito. Los valores de voltaje se ajustan cada uno según la ley de Ohm y se pueden medir con un multímetro.

A menudo será necesario que pongas resistencias en serie con otros componentes, como los ledes, para reducir el valor del voltaje que entra en el componente.

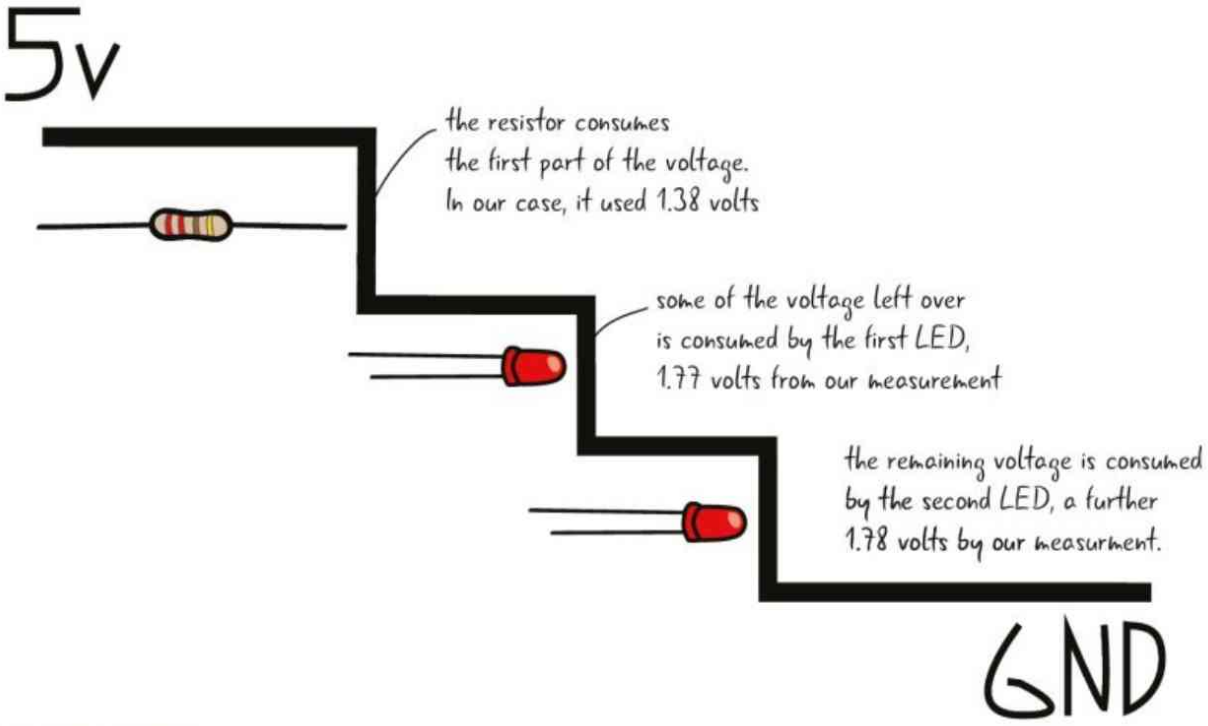


FIGURA 5.52 Visualización de la caída de voltaje en el circuito con ledes en serie.

Es poco probable que coloques varios ledes en serie, ya que cada led adicional hace que todas las luces se atenúen. Las viejas cadenas de luces navideñas, como las de la figura 5.53, son un ejemplo del mundo real de luces diseñadas para conectase en serie. Estar conectadas en serie es la razón por la que si una de las bombillas se funde toda la cadena de luces se apaga. Las tiras de luces más recientes se han diseñado para evitar este problema.

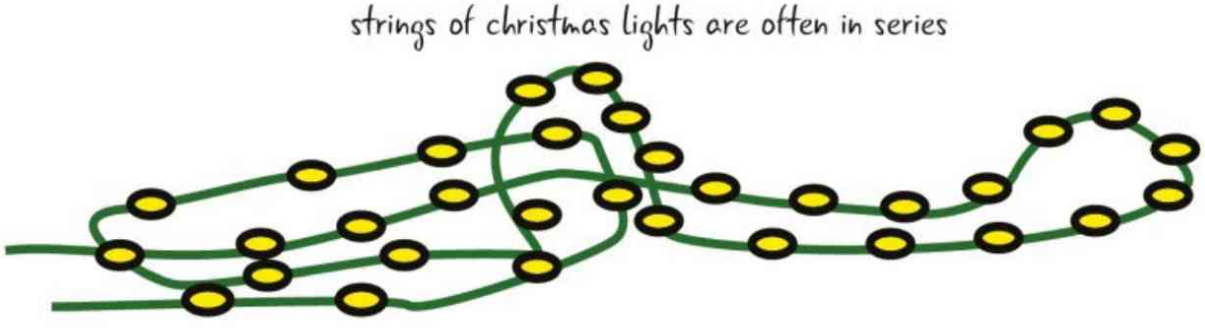


FIGURA 5.53 Luces de Navidad, dispuestas en serie.

el Multímetro en Serie

¿Recuerdas cómo sacaste el ánodo del led cuando estabas midiendo la corriente en el circuito básico? Insertaste el multímetro directamente en el circuito, tocando un extremo de la resistencia y el ánodo del led para completar el circuito. En esa disposición el multímetro estaba en serie con la resistencia y el led. El multímetro tiene que estar en serie para medir la corriente, porque así no altera el valor de la corriente (figura 5.54).

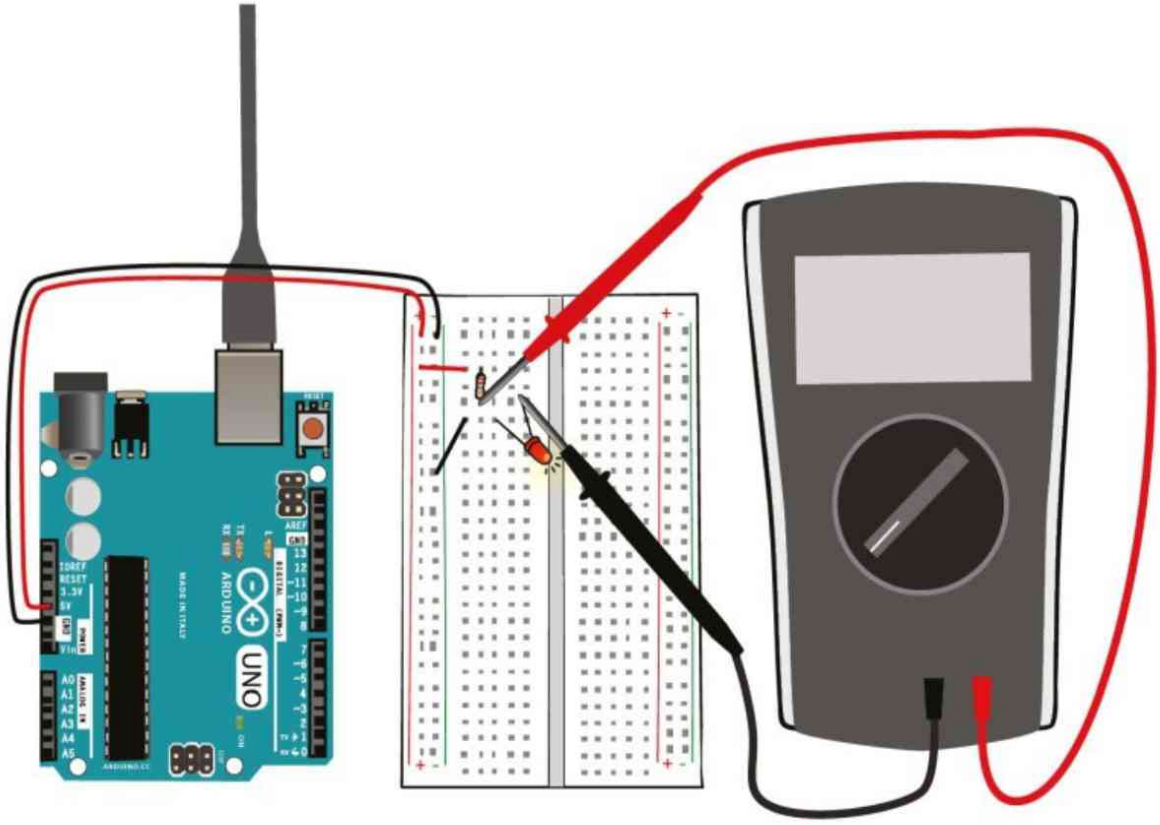


FIGURA 5.54 El multímetro está en serie con los otros componentes cuando se mide la corriente.

La tabla 5.3 muestra los efectos de las características eléctricas sobre los componentes tanto en serie como en paralelo.

Tabla 5.3:**Efectos de las características eléctricas sobre los componentes en serie y en paralelo.**

Efecto sobre las características eléctricas	ComponentEs En serie	ComponentEs En paralelO

Efecto sobre el voltaje	Cada componente consume parte del voltaje	Todos los componentes tienen el mismo voltaje
Efecto sobre la corriente	La misma corriente pasa por todos los componentes	La corriente se divide en función del valor de la resistencia de cada componente
Efecto sobre la resistencia	La resistencia total es igual a la suma de todas las resistencias	La resistencia total se reduce cuando los componentes están en paralelo

resumen

Has aprendido sobre el voltaje, la corriente y la resistencia, y cómo interactúan a través de la ley de Ohm, también sabes cómo medir estas propiedades con el multímetro. También has aprendido a configurar los componentes en serie y en paralelo. En el siguiente capítulo volveremos a los proyectos de Arduino y realizarás prácticas adicionales de programación.


6

interruptores, LEDEs Y OTROS COMPONENTES

En este capítulo, vas a aprender cómo hacer que los proyectos sean interactivos, empezando por añadir un botón para encender y apagar un led. Luego, conectarás un altavoz al Arduino y controlarás con el sketch tanto el sonido como la luz. Finalmente, añadirás dos botones más para montar un instrumento con teclado, con el que podrás interpretar sencillas melodías. En la realización de estos proyectos, aprenderás más sobre la programación del equipo Arduino. Para completar los proyectos de este capítulo, necesitarás tener instalado el IDE de Arduino, saber cómo conectar una placa de pruebas a tu Arduino y estar familiarizado con escribir un sketch y subirlo al Arduino.

¡Interactividad!

En el capítulo 4, "Programación del Arduino", viste como conectar el Arduino a una placa de pruebas para montar un circuito que iluminaba un led siguiendo el patrón SOS. El led se encendía y apagaba una y otra vez, siguiendo siempre el mismo patrón. ¿No sería genial si pudieras montar un circuito que respondiera a la acción del usuario?

Precisamente vas a hacer eso en este capítulo, diseñarás un mini teclado con tres botones, un altavoz y un led. El altavoz reproducirá un tono diferente, en función del botón que pulses, para que puedas interpretar una melodía. Y el led se encenderá cada vez que presiones cualquiera de los botones. Empezaremos con un circuito con un led y un botón, y avanzaremos a partir de este circuito básico. La figura 6.1 es una vista previa de cómo será el circuito terminado.

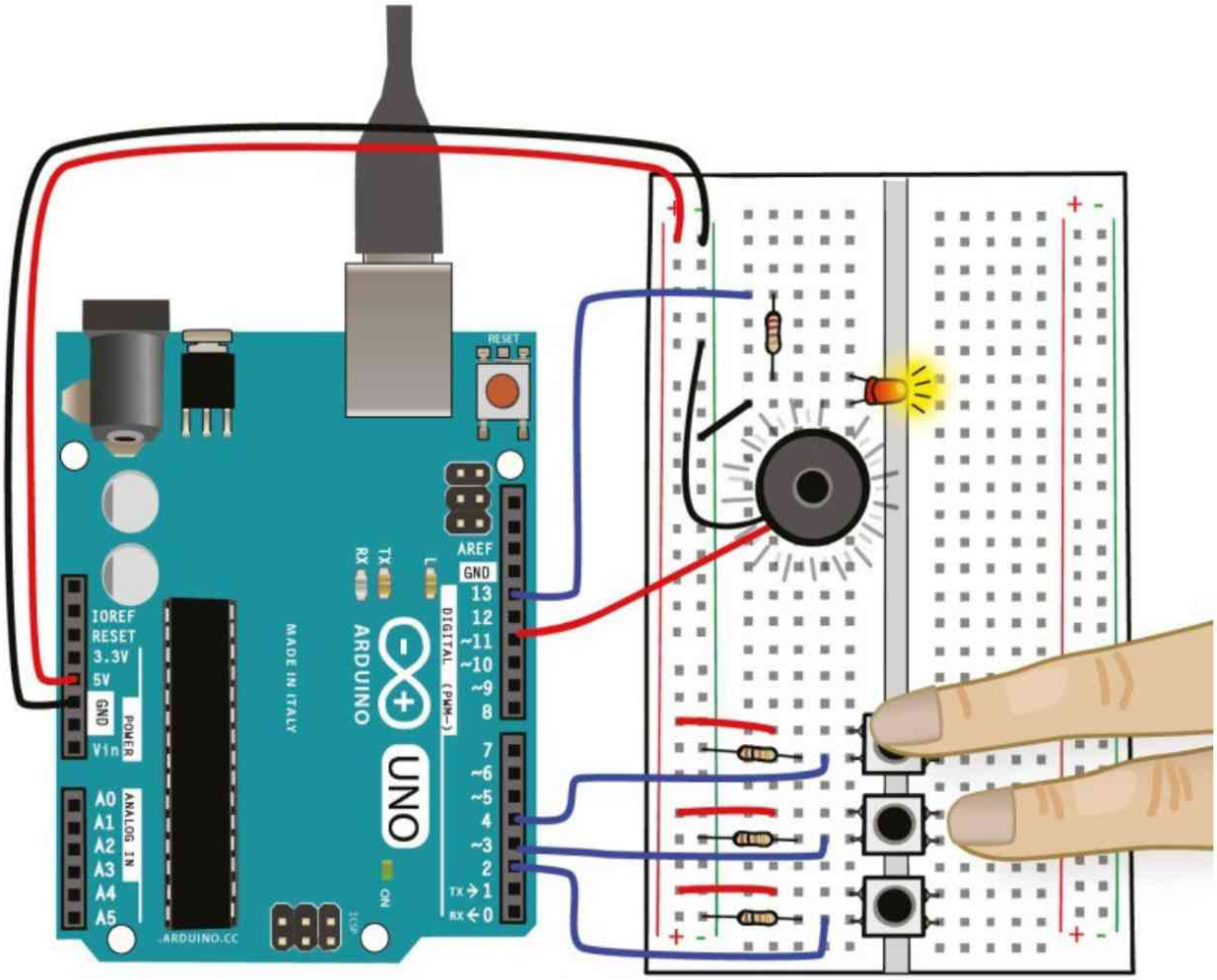
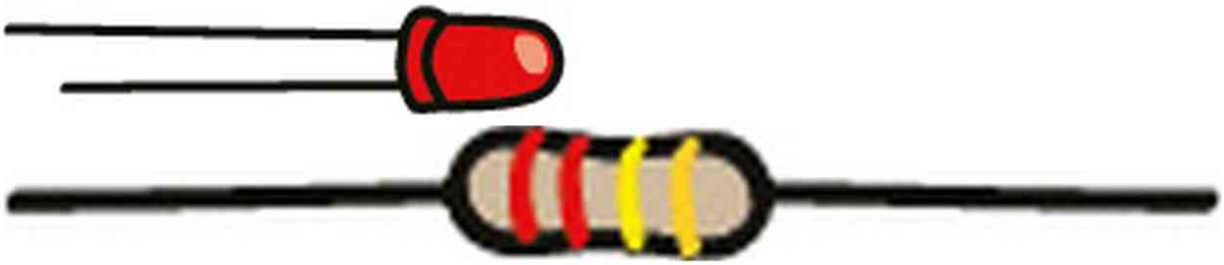


FIGURA 6.1 Teclado con botones para reproducir tres tonos.



Al igual que en capítulo 4, el proyecto utilizará un Arduino, una placa de pruebas para el circuito y el código escrito en el IDE de Arduino. Iremos

montando el circuito y escribiendo el código en el sketch, paso a paso. En este capítulo, aprenderás un poco más sobre la lectura de esquemas.

Este proyecto utiliza entradas y salidas digitales. En el último capítulo has utilizado una salida digital (un led). Antes de empezar a montar el circuito, echemos un vistazo más de cerca a lo que entendemos por entrada y salida digital.

RESUMEN DE LAS ENTRADAS Y SALIDAS DIGITALES

Piensa en tu ordenador: ¿cómo puedes obtener información de él? Puedes utilizar un ratón y, probablemente, un teclado. Los teclados y los ratones son ejemplos de *entradas* (figura 6.2). Puedes añadir muchos tipos diferentes de entradas al equipo Arduino. En este capítulo añadirás botones al Arduino que servirán como entradas.

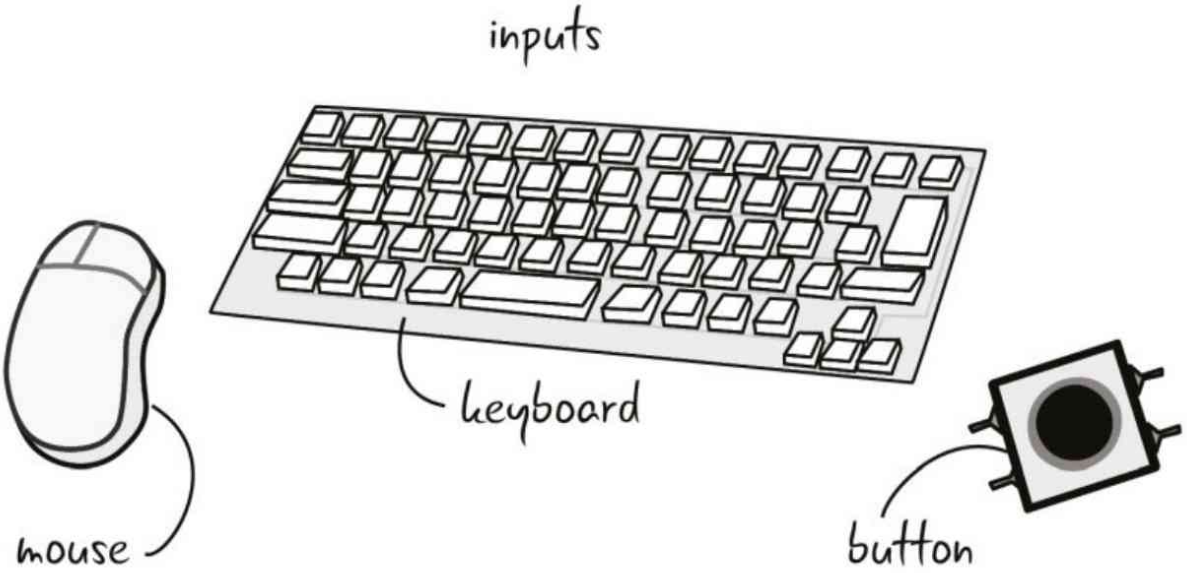


FIGURA 6.2 Entradas más frecuentes.

¿Qué entendemos por salidas? Piensa de nuevo en tu ordenador. Es posible que tengas altavoces conectados, o un monitor, o una impresora. Todos estos son ejemplos de dispositivos de *salida* (Figura 6.3). Un Arduino puede tener muchos tipos diferentes de salidas conectadas a él. De hecho, ya has utilizado un dispositivo de salida con el Arduino: el led que conectaste en el último capítulo.

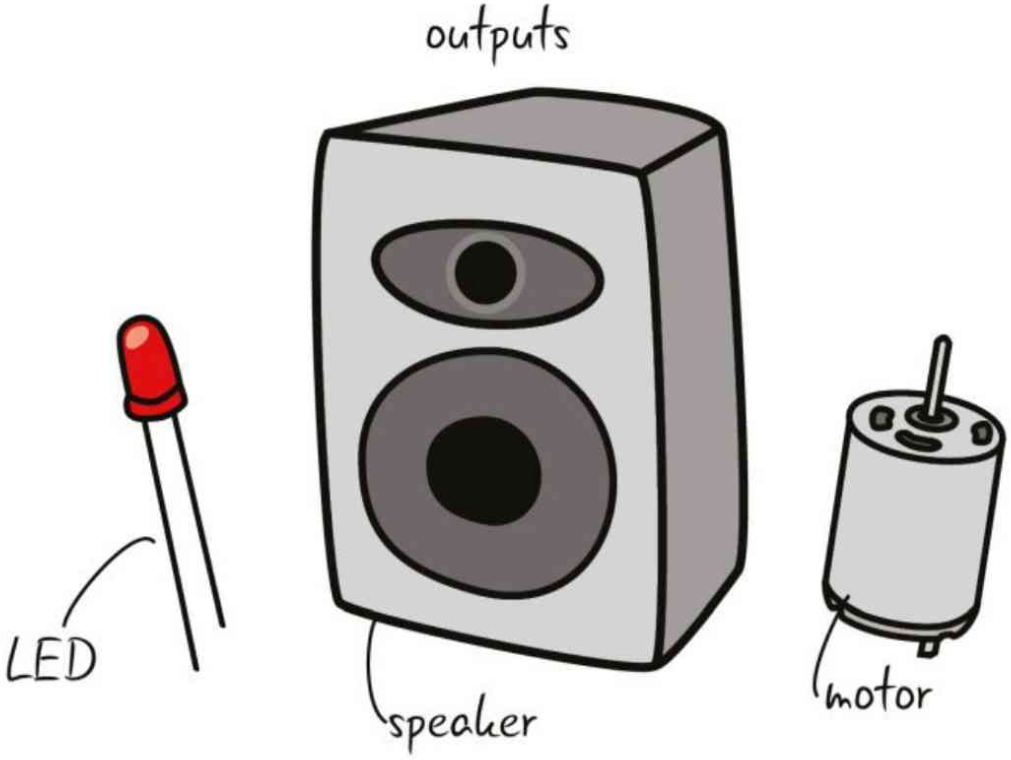


FIGURA 6.3 Salidas más frecuentes.

Por ahora, puedes pensar en las entradas y salidas digitales como componentes que solo tienen dos estados posibles: encendidos y apagados. Las entradas envían mensajes al ordenador. Las salidas reciben mensajes del ordenador. Explicaremos esto con más detalle más adelante, en este capítulo.

Antes de que empieces a montar el circuito, veamos el esquema correspondiente a un botón o interruptor. Esto te ayudará a comprender cómo funciona la entrada digital.

interruptores

Hay un millón de maneras diferentes de activar dispositivos electrónicos o de encender algo. Los interruptores y dispositivos de encendido/apagado activan televisores, equipos de música y luces, ¡incluso los electrodomésticos de la cocina! ¿Cómo funciona un interruptor?

Todos los interruptores funcionan según el mismo principio básico: o "cierran el circuito" y abren algo, o "abren el circuito" y apagan algo. Cuando

el interruptor está cerrado, la electricidad puede fluir a través de él; no puede fluir cuando el interruptor está abierto. La figura 6.4 ilustra cómo funciona.

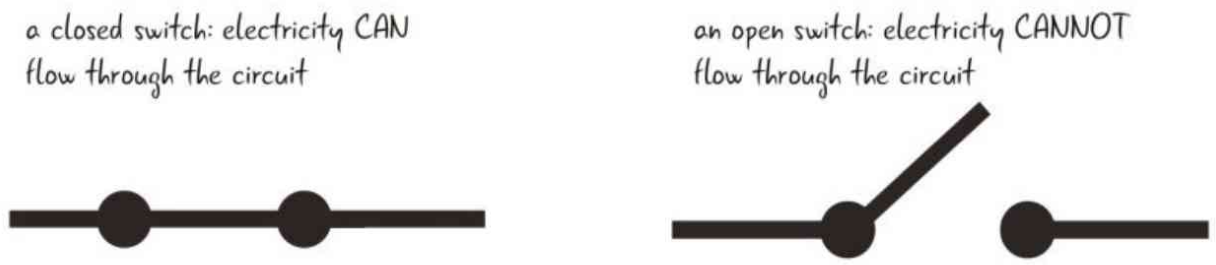


FIGURA 6.4 Diagrama de un interruptor.

Como en todas las entradas digitales, como viste antes, los interruptores solo tienen dos posibles estados: encendidos y apagados. En el IDE de Arduino, encendido y apagado son equivalentes a HIGH y LOW (respectivamente). Recuerda cómo, en el circuito SOS del capítulo 4, la luz se encendía cuando lo ponías en HIGH y se apagaba cuando lo ponías en LOW. Cada tecla del teclado es en realidad un interruptor puesto en la posición de apagado, hasta que se presiona y se cierra.

Los botones son una especie de interruptores. Para nuestro circuito utilizaremos un botón pulsador momentáneo, que cierra y completa el circuito cuando lo presionamos. Tan pronto como lo soltamos, el interruptor se abre otra vez y el circuito ya no está cerrado.

entrada Digital: añAde un Botón

Reúne todas las piezas para empezar a montar un circuito con un botón.

Necesitarás lo siguiente:

- ▨1 led
- ▨1 resistencia de 220 ohmios (rojo, rojo, marrón, dorado)
- ▨1 resistencia de 10 kΩ (marrón, negro, naranja, dorado)
- ▨1 interruptor pulsador momentáneo
- ▨Cables de puente
- ▨Placa de pruebas
- ▨Arduino Uno
- ▨Cable USB tipo A-B
- ▨Ordenador con el IDE de Arduino

La figura 6.5 muestra el esquema y un adelanto de cómo será el circuito una vez montado. Como el esquema es un poco diferente a los que hemos visto anteriormente (incluye algunos símbolos nuevos), lo examinaremos con más detalle.

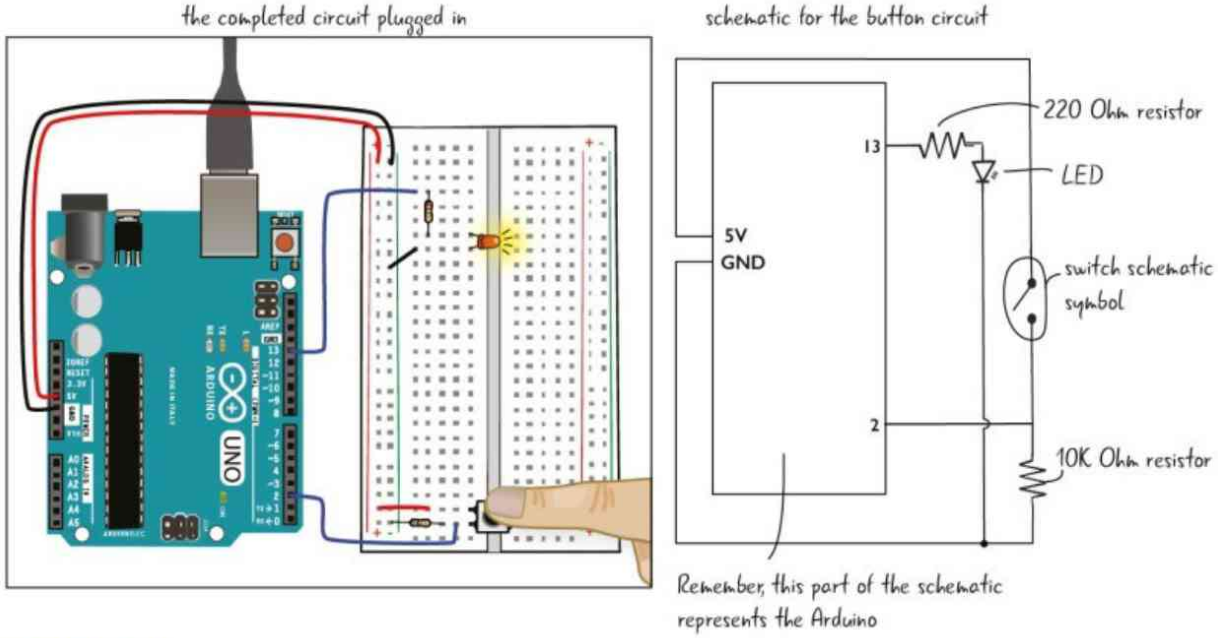


FIGURA 6.5 Circuito con un botón led.

comprensión de esquemas más complejos

El esquema de este circuito sigue un par de convenciones que no has visto antes.

En la parte inferior de este diagrama esquemático hay un pequeño círculo que indica que el cátodo del led (recuerda: el cátodo es el terminal corto, es decir, el terminal negativo) en el pin 13 está conectado a la misma tierra de la resistencia, que está conectada a un extremo del interruptor. En los esquemas se utiliza a menudo un círculo relleno para indicar los puntos de conexión.

A medida que los esquemas se van haciendo más complejos, a veces hay que trazar líneas que *no* están conectadas entre sí. Para indicar que esas líneas no están conectadas, dibujaremos un pequeño bucle como el que se muestra a la derecha de la figura 6.6.

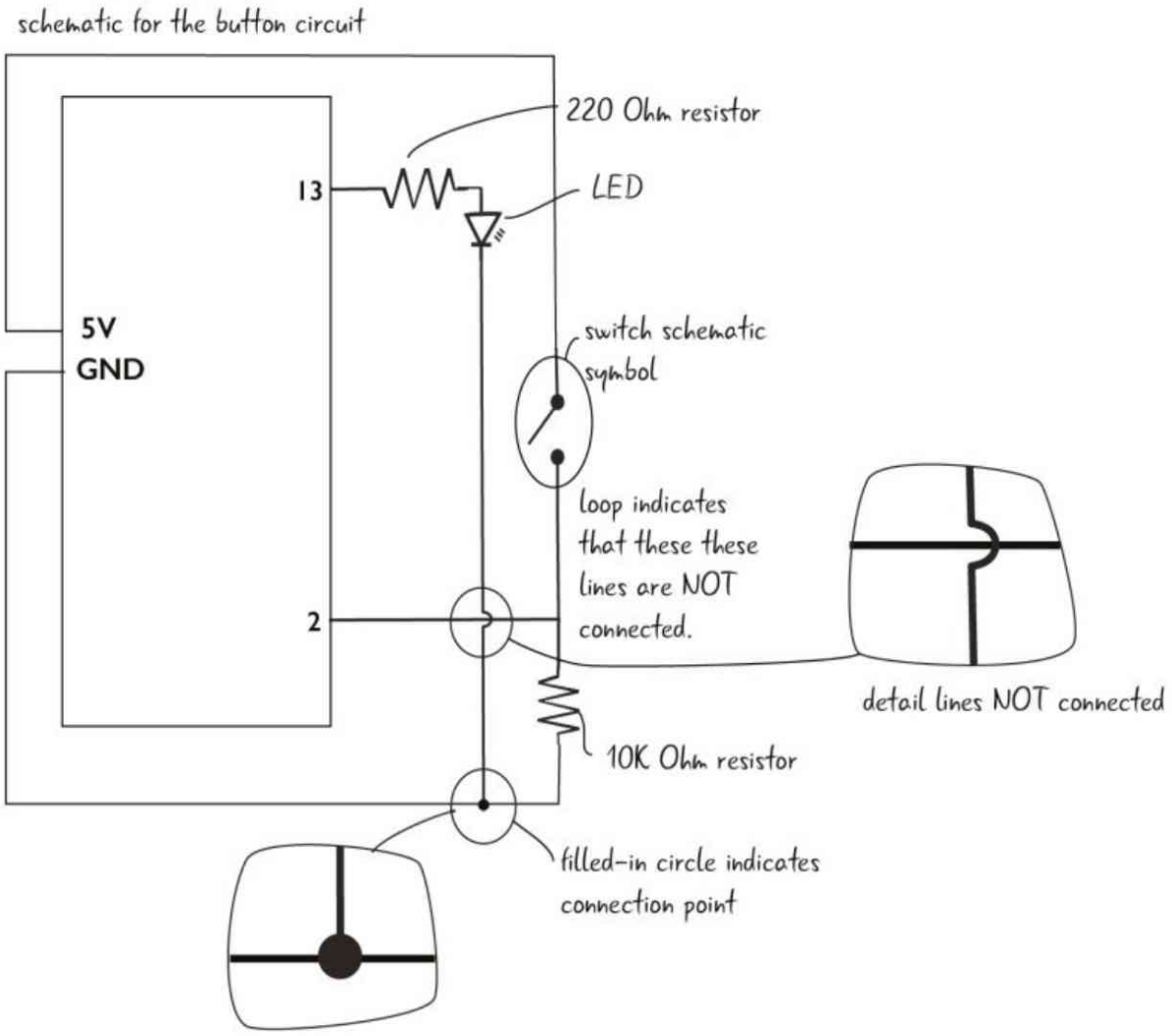


FIGURA 6.6 Diagrama esquemático.

MONTAJE DEL CIRCUITO CON UN BOTÓN

Antes de añadir el botón al circuito, tienes que volver a montar el circuito que utilizaste con el sketch LEA4_Blink en el capítulo 4, que se reproduce en la figura 6.7. Aquí se presenta una breve descripción de cómo hacerlo. Vamos a continuar y revisaremos cada paso a medida que avanzas:

1. Conecta los pines de alimentación y tierra del Arduino a los buses de alimentación y tierra de la placa de pruebas.
2. Conecta un puente desde el pin 13 del Arduino a la fila de los puntos de conexión de la placa de pruebas.

3. Conecta una resistencia de 220 ohmios a la misma fila de los puntos de conexión del pin 13.
4. Conecta el ánodo (terminal largo) del led al otro extremo de la resistencia y un puente entre el cátodo (terminal corto) del led al bus de tierra.

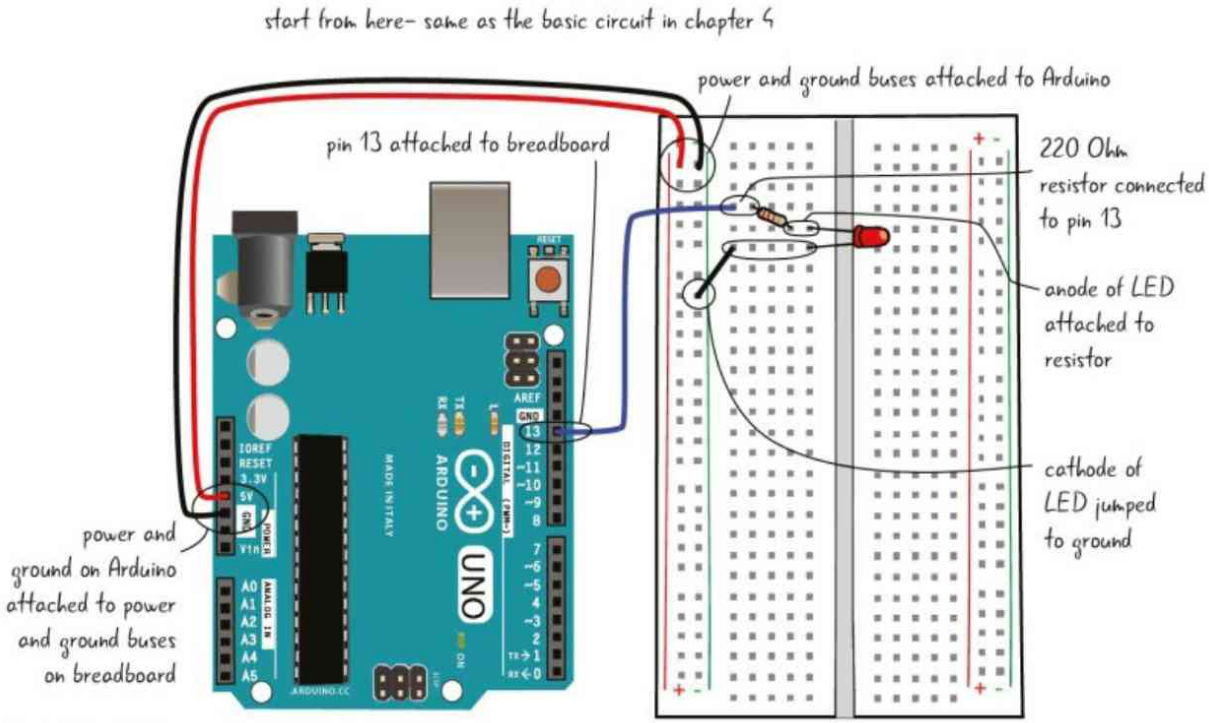


FIGURA 6.7 Revisión del circuito básico del capítulo 4.

Ahora que tienes el circuito básico montado, añadirás un botón. Ya has visto el esquema de un botón, pero antes de colocar el botón en la placa de pruebas, veamos cómo se hace el botón.

instalación del Botón

El botón que vas a añadir es de tipo pulsador (también conocido como interruptor momentáneo). Mientras pulsas el botón, el led se encenderá; tan pronto como levantes el dedo, el led se apagará (figura 6.8). Este tipo de botón recibe su nombre por el simple hecho de poder cambiar su estado momentáneamente.

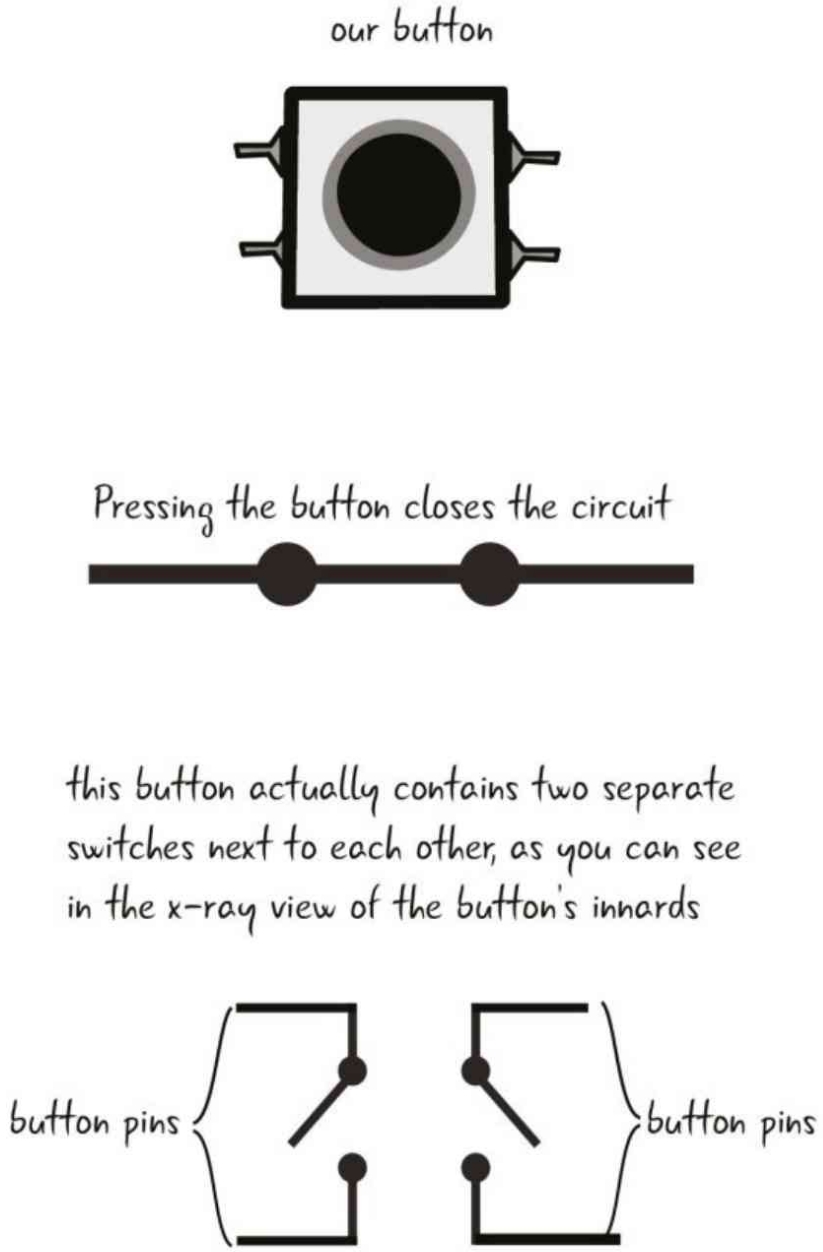


FIGURA 6.8 Diagramas del botón.

Cuando se presiona el botón, el circuito está cerrado y la electricidad puede pasar a través de él, como muestra el diagrama que viste anteriormente en este capítulo.

Este botón contiene en realidad dos interruptores separados (por eso tiene cuatro pines que salen de él). Ambos se cierran al presionar el botón hacia abajo. Tu circuito solo utilizará un lado del botón.

¿Recuerdas la separación que hay en medio de la placa de pruebas? (se

comentó en el capítulo 3 "Primer contacto con el circuito"). Colocarás los pines del botón a ambos lados de la separación, con los dos pines insertados en la fila de puntos de contacto en cada lado. El botón encajará a ambos lados de la separación siguiendo una orientación. Colocar el botón a ambos lados de la separación te asegura que está correctamente orientado con cada pin conectado a una fila de puntos de contacto separada (figura 6.9). Mientras el botón se ajuste a ambos lados de la separación, no importará la dirección del mismo.

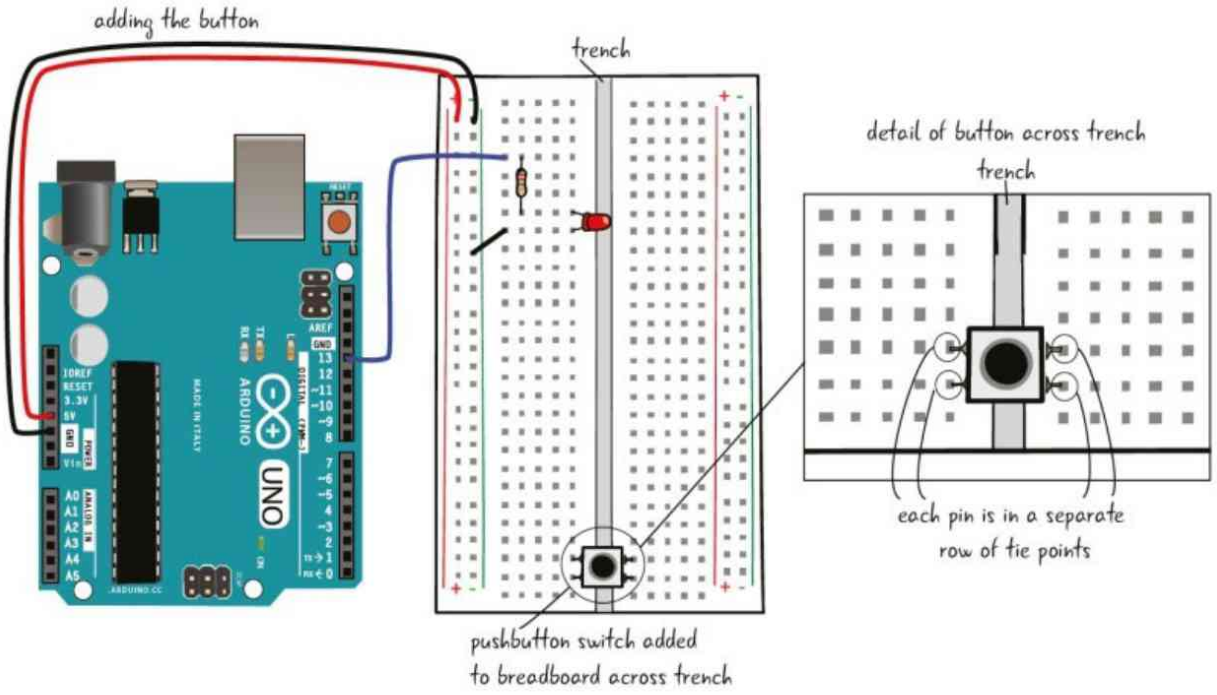


FIGURA 6.9 Instalación del botón en la placa de pruebas.

Tip Coloca el botón cerca de un extremo de la placa para facilitar la instalación del resto de componentes.

Conexión del Botón a la alimentación, a una resistencia y a tierra

Sigamos con la instalación del botón. Añade un puente de color rojo que conecte el bus de alimentación (el que tiene el signo "+") en la placa de pruebas, a la parte superior izquierda del botón (figura 6.10).

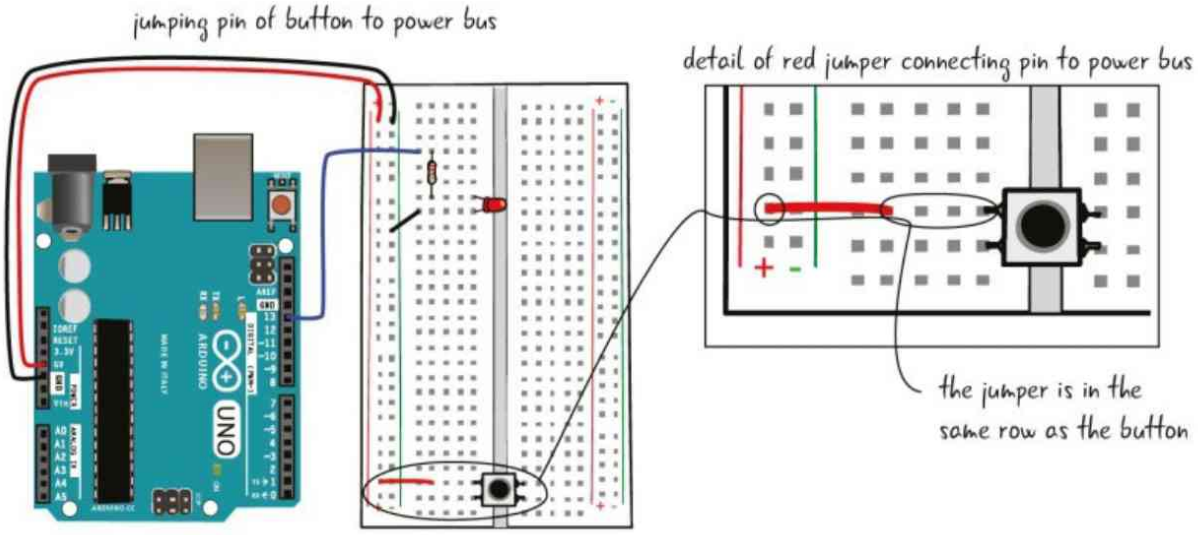


FIGURA 6.10 Instalación del primer cable de puente.

A continuación conecta una resistencia de 10 kΩ (colores marrón, negro, naranja y dorado). Conecta uno de los terminales de la resistencia al botón, y el otro terminal al bus de tierra (el que tiene el signo"–"), como se muestra en la figura 6.11.

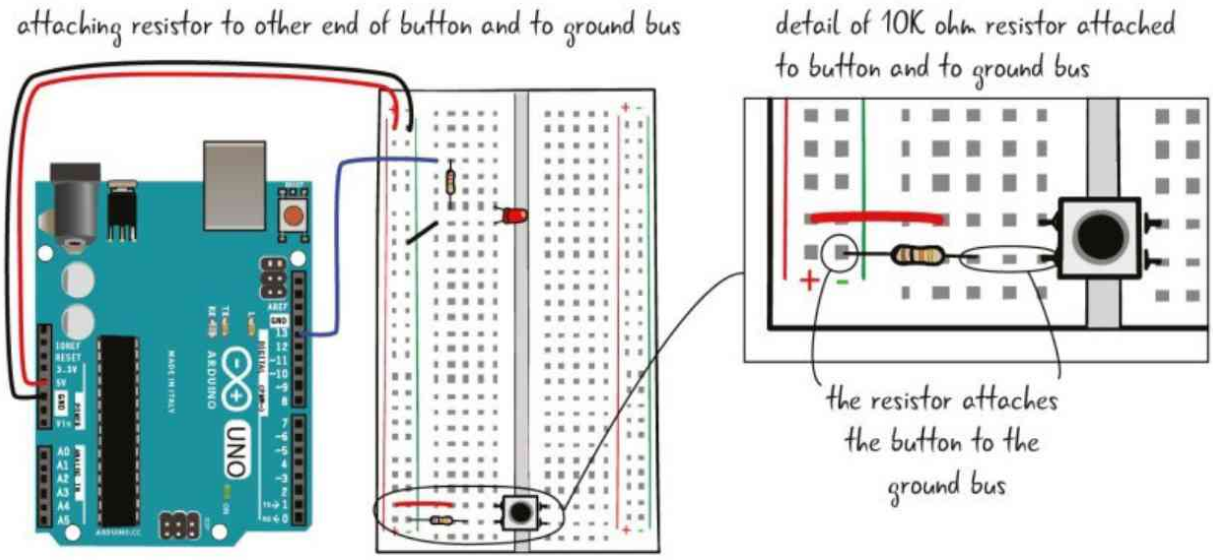


FIGURA 6.11 Instalación de la resistencia conectada con el botón.

CONEXIÓN DEL BOTÓN A UN PIN DE ARDUINO Y CARGA DEL Sketch

Por último, instalarás un puente al pin 2 del Arduino. Este puente se conectará a la resistencia de 10 kΩ que está conectada al bus de tierra. Como puedes ver en la figura 6.12, la resistencia, el puente al pin y uno de los pines del botón deberían estar todos en la misma fila de puntos de contacto. El cable puente también debe estar entre la resistencia y el botón.

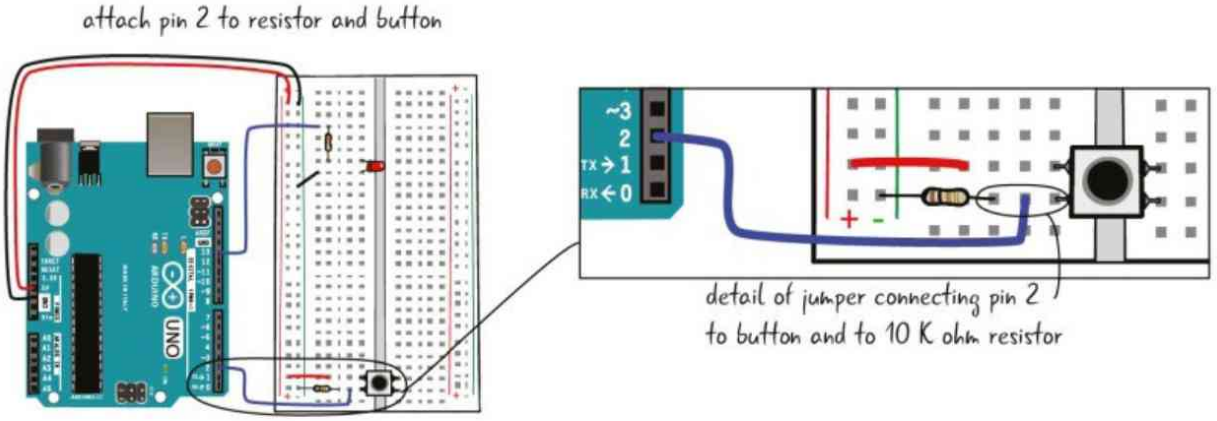


FIGURA 6.12 Conexión del puente al pin digital.

El botón ya está conectado. Ahora que el Arduino y el botón están conectados a la placa de pruebas, conecta el Arduino al ordenador, así puedes cargar el sketch que controlará el comportamiento del botón y del led.

ABRIR, GUARDAR, VerifICAR Y CARGAR

Conecta el ordenador al equipo Arduino con el cable USB para poder cargar el sketch Button. Es uno de los ejemplos de sketches que vienen con el IDE de Arduino.

1. Inicia el IDE de Arduino, y a continuación abre el sketch Button; selecciona File > Examples > 0.2 Digital > Button.
2. Guarda el sketch del botón como **LEA6_Button**.
3. Haz clic en el botón **Verificar** primero para asegurarte de que el código está bien.

4. Haz clic en el botón **Subir** para subir el código al Arduino. Todo esto se muestra en la figura 6.13.

FIGURA 6.13 Procedimiento para cargar el código en el Arduino.

encender y apagar el led

Ahora, cuando presiones el botón, el led se encenderá, como se puede ver en la figura 6.14.

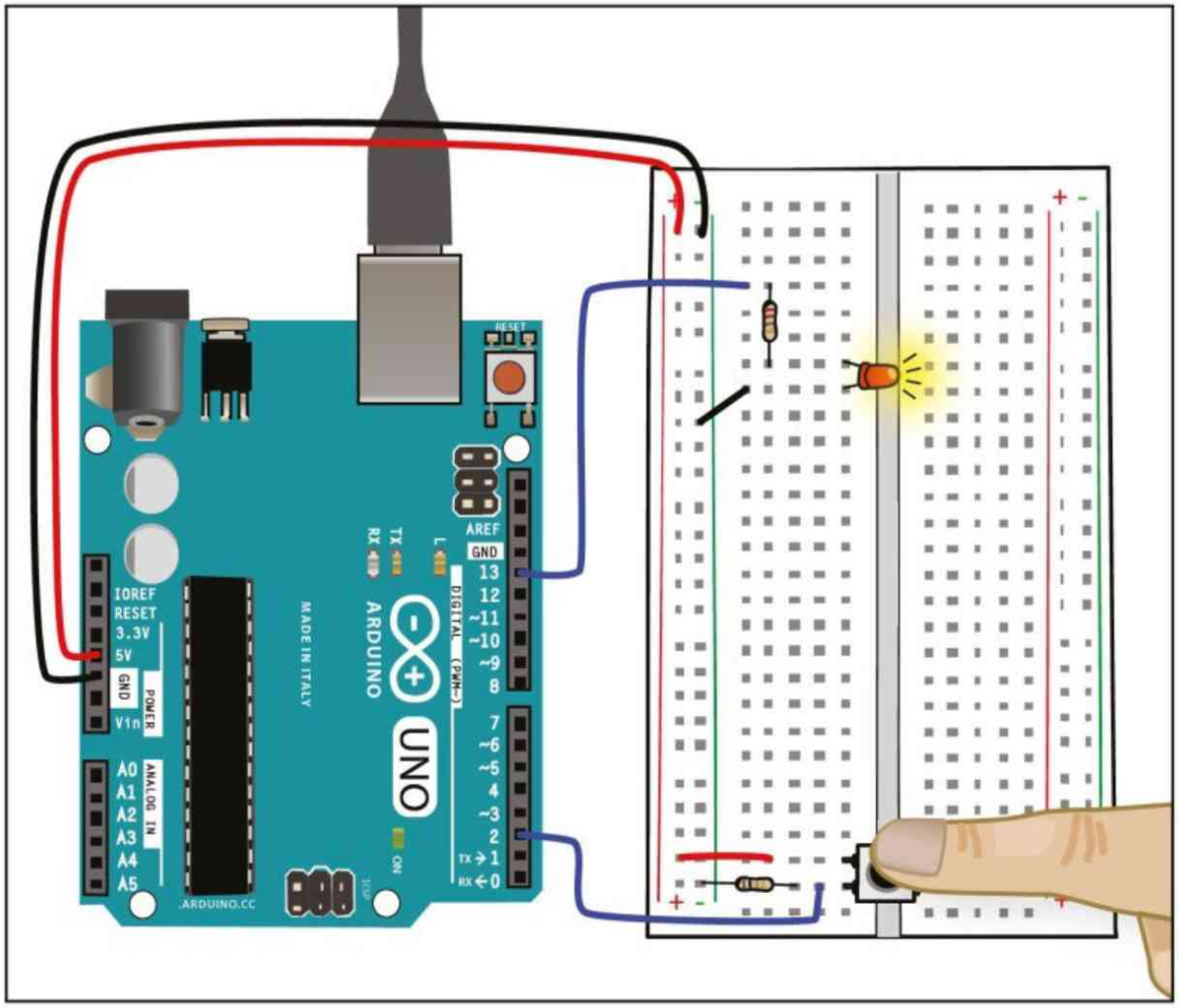


FIGURA 6.14 Presiona el botón y el led se enciende.

¿PREGUNTAS?

P: ¿Puedo utilizar otros tipos de interruptores o botones que encuentre?

R: ¡Sí! Todos los interruptores y botones funcionan según el principio de cerrar el circuito (haciendo un trazado completo) o abrirlo (rompiendo el trazado).

P: ¿Puedo usar un solo botón para disparar más de una salida? Por ejemplo, ¿podría usar un solo botón para activar toda una cadena de luces?

R: Aunque se puede programar un botón para activar muchas cosas diferentes al mismo tiempo, la mayoría de los aparatos electrónicos tienen un botón para cada función porque esa configuración facilita al usuario entender exactamente lo que se está activando en cada momento. Si el mismo botón dispara muchas funciones, el usuario se puede confundir en cuanto al modo en el que funciona la interacción.

Has montado el circuito y examinado el esquema. Ahora vamos a estudiar el código de LEA6_Button en detalle.

análisis del Sketch: Variables

Aquí está el código del sketch para LEA6_Button. Lo veremos en detalle en las siguientes páginas. Hemos eliminado los comentarios iniciales del código por razones de brevedad.

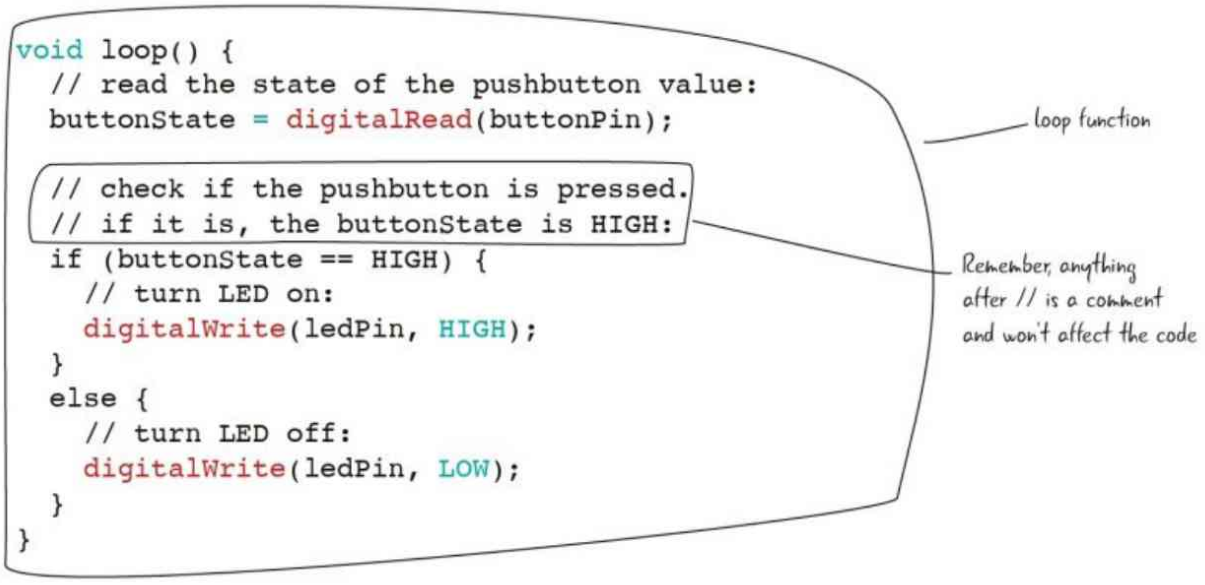
initialization section: some values declared here

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status
```

setup function

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```



Inicialización del Código y de las variables

LEA6_Button es diferente del sketch de LEA4_Blink, en el que hay código que se ejecuta antes de la función setup(). A este código inicial se lo llama *código de inicialización* (código en la parte superior de un sketch en el que declaras los valores a los que quieres tener acceso durante el sketch). Echemos un vistazo a las tres líneas de código de inicialización de este sketch:

```
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

int buttonState = 0;        // variable for reading the pushbutton status
```

Las tres líneas de código se parecen; todas tienen algunas palabras a la izquierda y números a la derecha, con un signo de igual en el centro. ¿Pero qué hacen? Para facilitar la comprensión de esta sección del código, introduciremos un nuevo concepto de programación: las *variables*.

¿qué es una variable?

En términos sencillos, una *variable* es un lugar en el que se almacena un valor específico y se le proporciona un nombre de referencia. Piensa que una variable es como un contenedor que contiene un valor. Si has estudiado álgebra, las variables te resultarán familiares. ¿Recuerdas las ecuaciones en las que se declaraban cosas como
"si x = 1"?

Las variables pueden contener diferentes tipos de valores. Por ahora, veremos las variables que contienen números enteros.

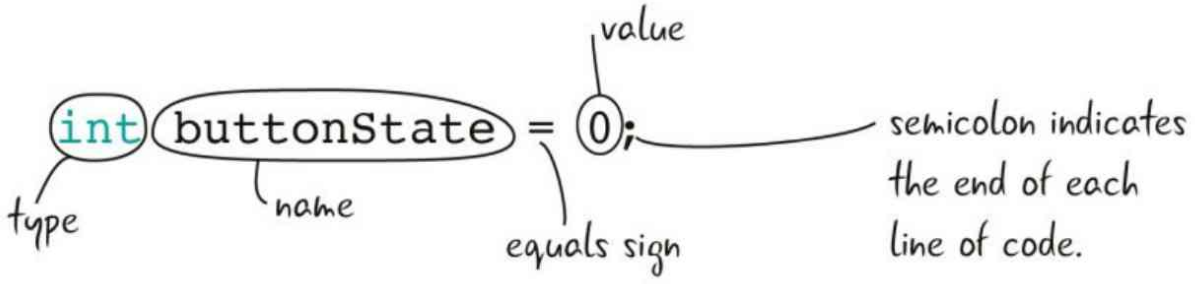
En la siguiente línea de código, *declaramos* y *asignamos* una variable. Declarar una variable significa darle un nombre y asignar una variable consiste en asignarle un valor. Puedes declarar variables sin darles un valor, pero no puedes dar un valor a algo que aún no se ha declarado.

Declaración de Variables

A una línea de código que define una variable se la denomina *declaración de variable*. A menudo, el código Arduino también contendrá una *asignación de variable*. Las declaraciones y asignaciones de variables no están estructuradas de la misma manera en todos los lenguajes de programación; solo estamos viendo cómo se declaran las variables en el lenguaje de programación Arduino.

En el lenguaje Arduino, todas las declaraciones y asignaciones de variables tienen al menos cuatro elementos diferentes: el *tipo* de datos que contiene la variable, el *nombre* de la variable, un *signo de igual*, y el *valor* real que deseas

establecer para la variable. Las variables pueden tener más de cuatro elementos; de momento, vas a aprender sobre un quinto componente.



En este ejemplo, asignas el valor 0 a una variable con el nombre buttonState que es del tipo int, que significa entero. Veamos todas las partes de esa declaración en detalle. Empezaremos con el nombre y luego veremos el valor.

Nombre de variable

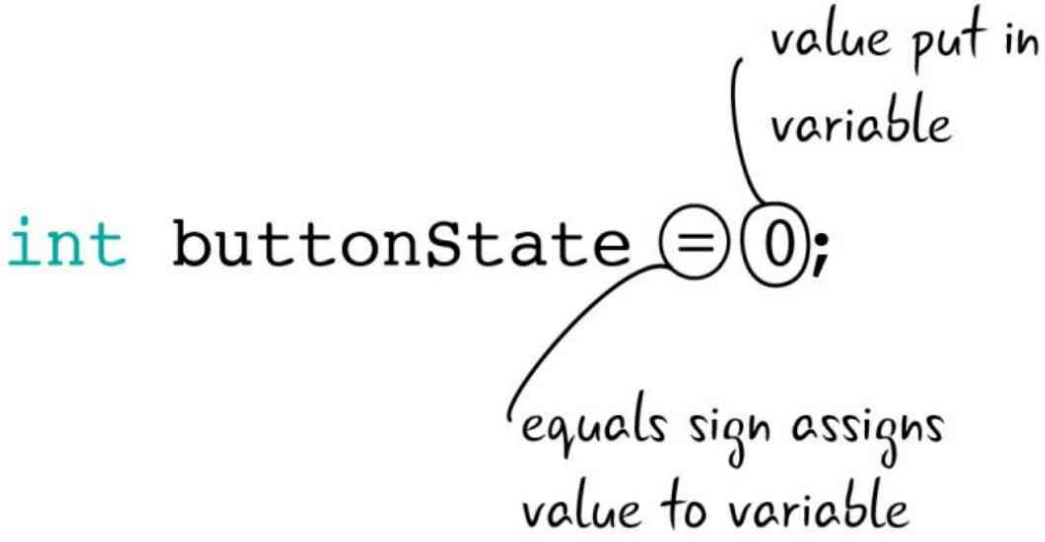
La parte del nombre de la declaración determina cómo te vas a referir a la variable en el sketch. Hay unas cuantas reglas para seleccionar nombres: un nombre de variable no puede empezar con un número, no puede contener espacios y no puede ser una palabra que el lenguaje Arduino ya utiliza para otro propósito (por ejemplo, no podemos darle a una variable el nombre "delay", ya que ese término está reservado). Para cada sketch, debe haber solo una variable con un nombre particular. Se considera una buena práctica nombrar a las variables como algo que indique su propósito.

```
int buttonState = 0;
```

Note Los nombres de las variables no pueden comenzar con un número, incluir espacios o símbolos, ni ser una palabra que utiliza el lenguaje Arduino.

Valor de la variable

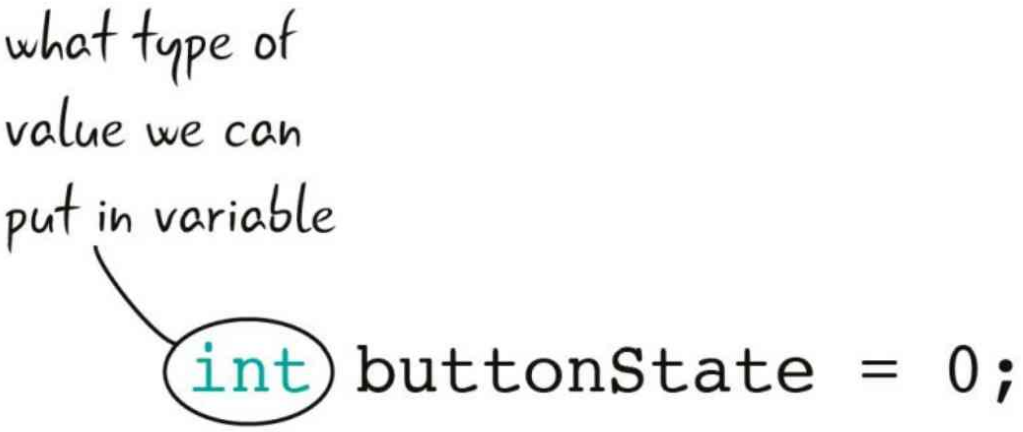
El elemento valor de la declaración es lo que se almacena en la variable. En este ejemplo tenemos un valor entero de 0, que corresponde al estado o valor del voltaje de nuestro pin. Los valores se establecen usando un = (signo de igual), que significa que en cualquier lugar donde veas el nombre de esta variable (buttonState), quiere decir que "Usas el valor de 0" o "Estás asignando el valor de 0 a esta variable".



Tipos de variables

type establece qué *tipo* de información se puede almacenar dentro de una variable. En la declaración que estamos viendo, int representa un *entero*, lo que significa que solo puedes almacenar valores que sean números enteros. No todos los lenguajes tienen tipos de variables, pero en el lenguaje Arduino,

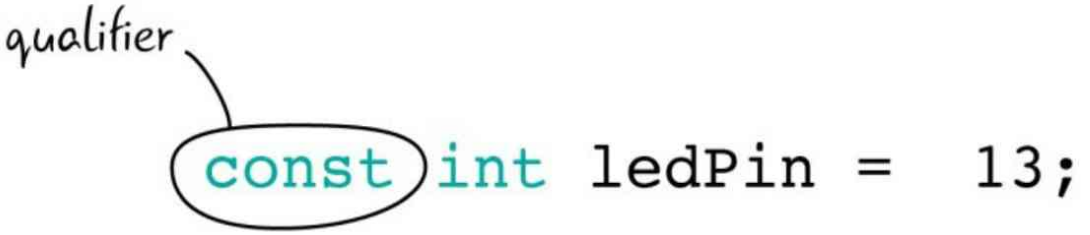
debes declarar el tipo de cada variable en el sketch. Otros tipos son la coma flotante, cadena, carácter y booleano; puedes aprender más sobre los tipos de variables en arduino.cc/en/Reference/VariableDeclaration.



Ahora que conoces las cuatro partes que se requieren en una declaración de variable, vamos a examinar una parte opcional que se usa en un par de declaraciones de variables en nuestro sketch: el calificador.

Calificadores de la variable

Algunas variables tienen un calificador, que determina si puedes cambiar el valor de una variable después de crearla. El calificador const configura la variable para que tenga un valor permanente cuando ejecutas el sketch. En este contexto, const significa *constante*. Aquí se muestra un ejemplo de nuestro sketch:



Puede parecer un poco extraño pensar en una variable constante, pero

recuerda que establecer una variable solo significa *llevar un registro de un valor con un nombre*. Cuando conectas los cables en los pines de tu Arduino, los números de pin no van a cambiar. Puesto que el valor de la variable no va a cambiar, añades const a la declaración, lo que deja claro que se trata de una variable constante.

> **Note** El calificador de variables es opcional; la mayoría de las declaraciones de variables solo tienen tres elementos: tipo, nombre y valor.

Como puedes ver en el siguiente código de inicialización, el sketch contiene dos variables constantes y una variable que puede cambiar.

```
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

int buttonState = 0;        // variable for reading the pushbutton status
```

¿PREGUNTAS?

P: ¿Qué ocurre si le doy a una variable algo que no está permitido?

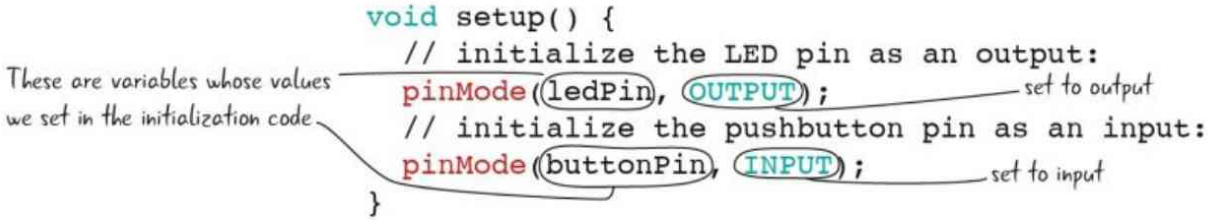
R: La consola de errores en el IDE de Arduino presentará la advertencia "unexpected unqualified id" (identificación no calificada e inesperada), que se mostrará en texto de color naranja. La solución más fácil es cambiar el nombre de variable por otro diferente.

P: ¿Necesito usar `const` para todas mis variables?

R: No, depende de para qué sirve la variable. Por ejemplo, en el sketch LEA6_Button, las variables de los pines nunca cambiarán en el sketch, pero la variable de `buttonState` sí. Quieres poder cambiar el valor de la variable `buttonState`, así que dejas `const` fuera de la declaración.

setup() de LEA6_Button

Ahora que hemos visto el código de inicialización del sketch LEA6_Button, pasemos a la función setup():



La función setup() para el sketch LEA6_Button solo tiene dos líneas de código. Como en el caso del código del capítulo 4, estás configurando un pin para que sea una salida usando la función pinMode(). Esta vez, sin embargo, estás usando la variable ledPin para sustituir al número 13. También utilizas pinMode() para establecer un pin diferente con el botón de buttonPin como entrada. Estas son dos de las tres variables que has creado en el código de

inicialización del sketch. Nombrar las variables te proporciona un modo de referirte a los números que necesitas en el sketch de una manera significativa, y hace que el código sea más fácil de leer.

Veamos más de cerca lo que entendemos por entrada digital.

repaso de entrada digital

Digamos que estás llegando a casa de un amigo cuando este te llama y te pide que mires a través de la ventana y veas si hay luz. Tu amigo pregunta: "¿Está encendida la luz?" Tu trabajo es decirle a tu amigo: "Sí, la luz está encendida" o "No, la luz está apagada". Eso es exactamente lo que hace una entrada digital: informar si la luz está encendida o apagada (figura 6.15).

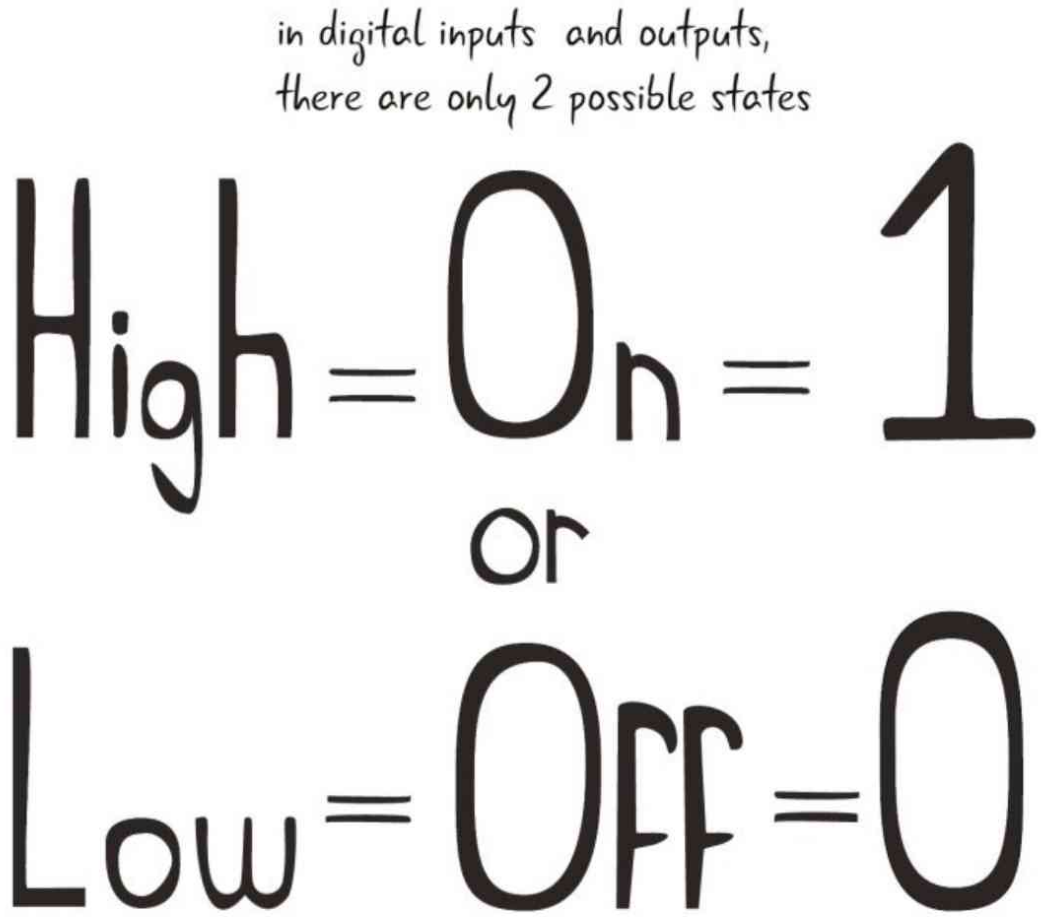


FIGURA 6.15 Estados de una entrada digital.

En las entradas digitales solo hay dos estados posibles: HIGH y LOW, que se pueden considerar como activado (HIGH) o desactivado (LOW). Las entradas digitales miden si algo está encendido (en un estado HIGH) o apagado (en un estado LOW). HIGH/encendido es también igual a 1 y LOW/apagado es igual a 0. Podemos usar los pines digitales del Arduino para comprobar los botones e interruptores y ver si se han disparado o no.

¿por qué tres maneras diferentes de decir lo mismo?

Si HIGH, 1 y encendido son todos equivalentes (como lo son LOW, 0 y apagado), ¿por qué hay varias maneras de decir lo mismo? Esto puede dar lugar a confusión.

Cada valor se refiere a un aspecto diferente de nuestro proyecto con Arduino:

- *Encendido* y *apagado* se refieren a lo que sucede en el mundo. Por ejemplo, ¿luce el led o no? No utilizamos los términos *encendido* y *apagado* en el código para programar Arduino, sino solo en nuestras discusiones generales.
- *1* y *0* son valores enteros de la variable que representan "encendido" y "apagado", respectivamente. Usamos 1 o 0 cuando estamos inicializando variables en nuestro código. Vimos un ejemplo de esto en el código de inicialización del sketch, que incluye la línea int buttonState = 0;. Esta línea indica al Arduino que el botón está inicialmente apagado.
- HIGH y LOW se refieren al estado eléctrico del pin: ¿proporciona el pin 5 voltios o actúa con 0 voltios (tierra)? En el lenguaje de programación Arduino HIGH y LOW se utilizan para ajustar o leer el estado de un pin (mediante las funciones digitalWrite() y digitalRead()).
- Los ceros y unos son parte del lenguaje binario que hablan los ordenadores. HIGH y LOW significan 1 y 0 para los ordenadores, incluyendo nuestro Arduino. HIGH y LOW hacen que el código sea un poco más fácil de leer para las personas, y se utilizan cuando usamos las funciones digitalWrite() y digitalRead(). Usarás 0 y 1 cuando vayas a crear nuevas variables.

anÁlisis del Sketch: declaraciones condicionales

Ahora que ya entiendes lo que hacen las entradas digitales, echemos un vistazo al código loop() del sketch LEA6_Button.

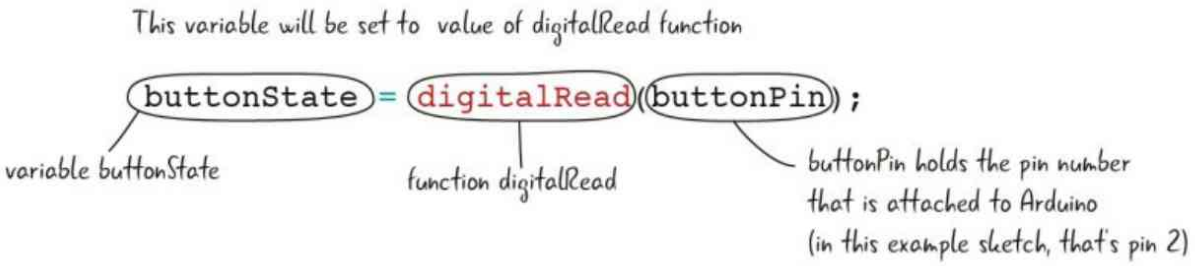
this code involves some new concepts, explained shortly

```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

Exploración de loop()

En la primera línea de la sección loop() del sketch LEA6_Button, el Arduino utiliza una función llamada digitalRead() para comprobar si un pin está activado o no. En este caso, estás comprobando el pin representado por la variable buttonPin, por lo que estás evaluando el estado del pin 2. Los resultados de la función digitalRead() serán 1 (HIGH) o 0 (LOW). A continuación configuras la variable denominada buttonState a este valor.



El siguiente elemento del código de loop() te proporciona otro nuevo concepto de programación: el uso de *declaraciones condicionales*.

¿Qué es una declaración Condicional?

Las declaraciones condicionales son una forma efectiva de cambiar lo que sucede en el código, en función de las condiciones que especifiques, como es el caso de si un botón está encendido o apagado. Has experimentado con el uso de declaraciones condicionales en el lenguaje cotidiano, como se muestra en la figura 6.16.

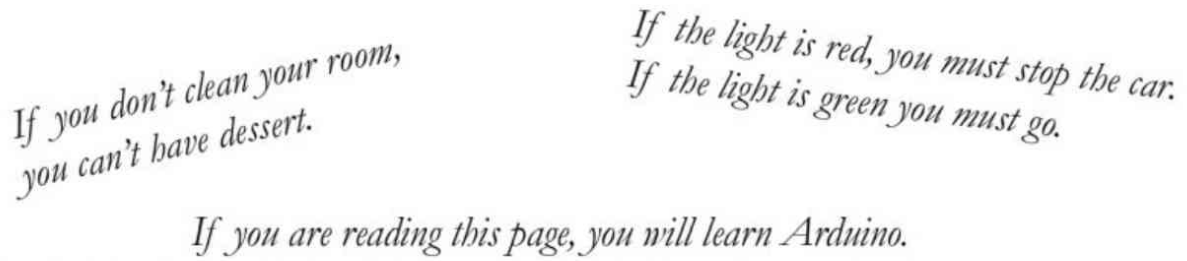
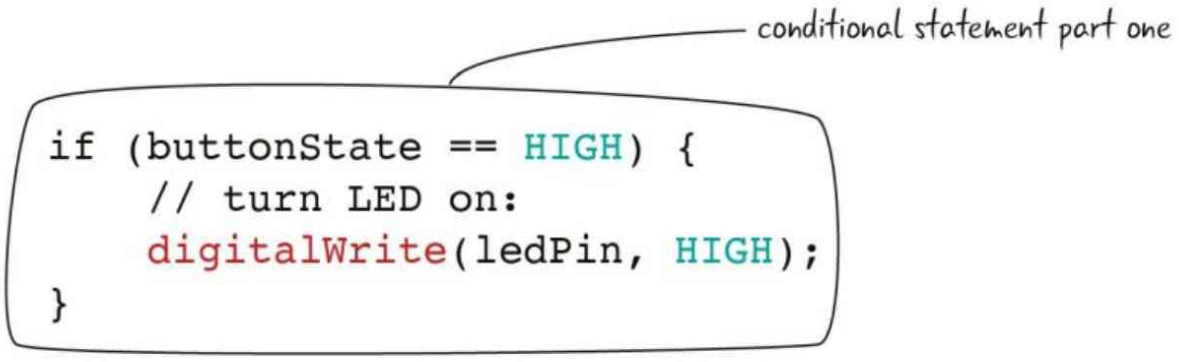


FIGURA 6.16 Declaraciones condicionales en español.

Las declaraciones condicionales en programación funcionan de la misma manera. Tienen tres partes básicas: el if, la expresión que estás evaluando, y lo que quieres que suceda si nuestra declaración es verdadera. Echemos un vistazo a las declaraciones condicionales en nuestro código loop().

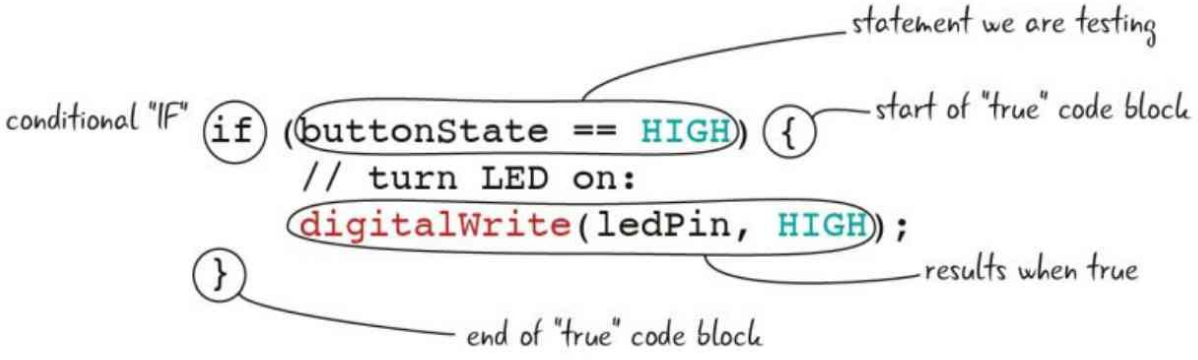
La primera parte de la declaración condicional dentro del código loop() para LEA6_Button se muestra aquí. En algunos sketches, esta podría ser la declaración condicional; la nuestra tiene una segunda parte, que pronto veremos.



Esta parte de la declaración condicional está formada por if, la expresión que estás evaluando, y lo que sucede si la expresión es verdadera. Todo lo que sucederá, si la declaración es verdadera, está contenido entre llaves. Tú, como programador, puedes decirle al programa qué hacer si ocurren ciertas situaciones.

DECLARACIONES CondiCionalES En loop()

Las declaraciones condicionales comienzan con if. El if le dice al ordenador que evalúe la siguiente expresión.

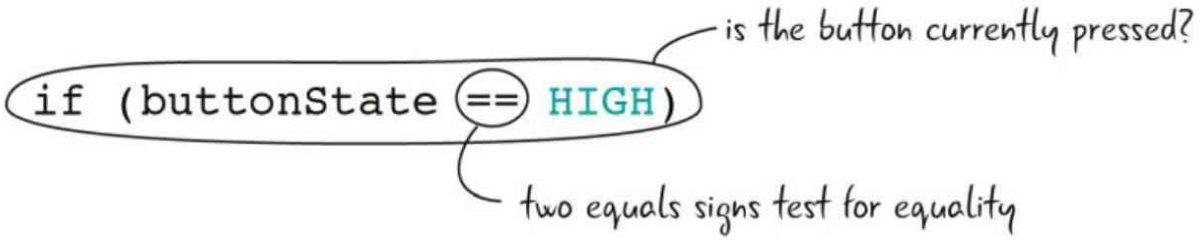


La siguiente parte de la declaración condicional es la *condición a evaluar*. Esta es la parte del código que el Arduino tiene que evaluar para saber la verdad. "True" en un contexto de programación significa que la condición es lógicamente válida. Por ejemplo, la frase "Uno es igual a uno" no nos dice nada interesante, pero es cierta. La afirmación absurda "Dos más dos es igual

a cinco" es falsa. Verás varios tipos de declaraciones condicionales a lo largo del libro que evalúan lo que ocurre con tu Arduino y con el resto del circuito.

En el caso de este sketch, el código trata de evaluar si el botón está pulsado en ese momento. Recuerda que pulsado significa "encendido", que es lo mismo que HIGH en el lenguaje de programación Arduino. Para comprobar si un valor es igual a otro, utilizas dos signos =, es decir ==.

> **Note** Las declaraciones condicionales empiezan con un `if` (si).



La última parte es el bloque de código "true", los comandos que se ejecutan si la condición es verdadera. No existe un límite para el número de acciones que puedes incluir dentro del bloque de código "true", siempre y cuando estén todas entre llaves. En este caso, el bloque de código encenderá el led asociado a ledPin, también conocido como pin 13.

```
if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
}
```

Note Las declaraciones condicionales verifican si algo es lógicamente cierto.

Tip Debes pensar detenidamente en lo que quieres que haga la declaración condicional. Podrías decirlo en voz alta para escucharlo tú mismo.

declaración Condicional: else

¿Qué sucede si *no* se pulsa el botón? Para esta declaración condicional existe la cláusula else, que maneja cualquier evento que sucede cuando la declaración *no es verdadera*. else es útil para tratar casos en los que la declaración if es falsa, pero no se requiere para cada declaración condicional. Algunas declaraciones condicionales tienen else, y otras no. Si la declaración condicional no tuviera else y la condición que estás evaluando fuera falsa, no pasaría nada.

Para el código del botón, la declaración else también se puede desglosar en una simple declaración: "Si no se pulsa el botón, entonces se apaga la luz".

```
else {
  // turn LED off:
  digitalWrite(ledPin, LOW);
}
```

the second part of our conditional statement means "if the button is not pressed, turn off the LED"

Note `else` **no es obligatorio en todas las declaraciones condicionales.**

La tabla 6.1 muestra un breve resumen de la declaración condicional que acabamos de ver.

Tabla 6.1:**Declaración condicional en LEA6_Button**

¿qué ocurre en el circuito?	Condición a evaluar	valor de verdad	Resultado
Botón pulsado	if (buttonState == HIGH)	true	Enciende el led
Button sin pulsar	if (buttonState == HIGH)	false	Apaga el led

¿PREGUNTAS?

P: ¿Qué ocurre si quiero obtener más de dos resultados posibles?

R: Entonces podrás usar `else if`, o tal vez varios `else if`. Puedes encontrar más información en: arduino.cc/en/Reference/Else.

P: ¿Puedo incluir un condicional dentro de otro condicional?

R: Sí, es posible tener condicionales dentro de otros condicionales. Si bien no vas a ver ejemplos en este libro, se llaman condicionales anidados, y nos permiten evaluar lógicas complejas.

Ahora que ya has conectado el botón y has encendido y apagado el led, estás preparado para hacer tu circuito más interesante. Vas a incorporar un altavoz, y después añadirás el código de manera que el altavoz emita un tono cuando presiones el botón. Primero veremos cómo añadir el altavoz a la placa de pruebas.

AÑADe UN ALTAVOZ Y modifica EL CÓDIGO

En este circuito, el botón, el led, las resistencias y los puentes permanecerán donde están. Solo vas a añadir un altavoz, todo lo demás permanece igual (figura 6.17).

Componente a añadir:

▨1 altavoz de 8 ohmios

Como siempre, antes de conectar el altavoz, asegúrate de que el ordenador no está conectado al Arduino. Conecta un extremo del altavoz al pin 11 del Arduino y el otro extremo al bus de tierra (figura 6.18). No importa en qué extremos los conectes; como en el caso de una resistencia, el altavoz no tiene polaridad. Los colores de los cables del altavoz pueden variar, pero el altavoz no tiene polaridad.

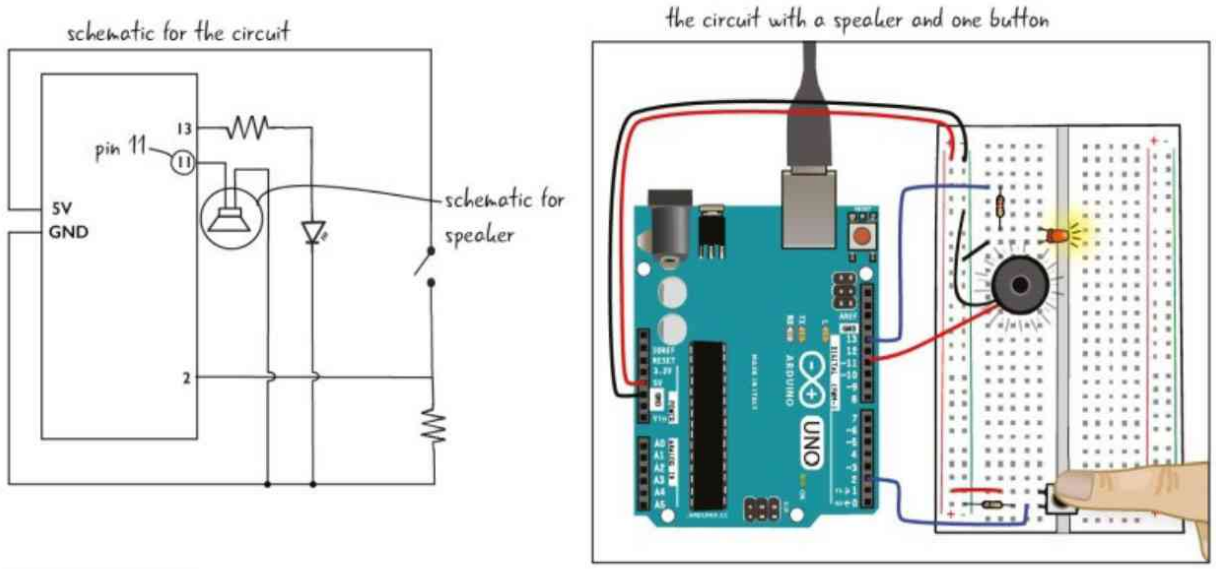


FIGURA 6.17 Altavoz instalado en el circuito.

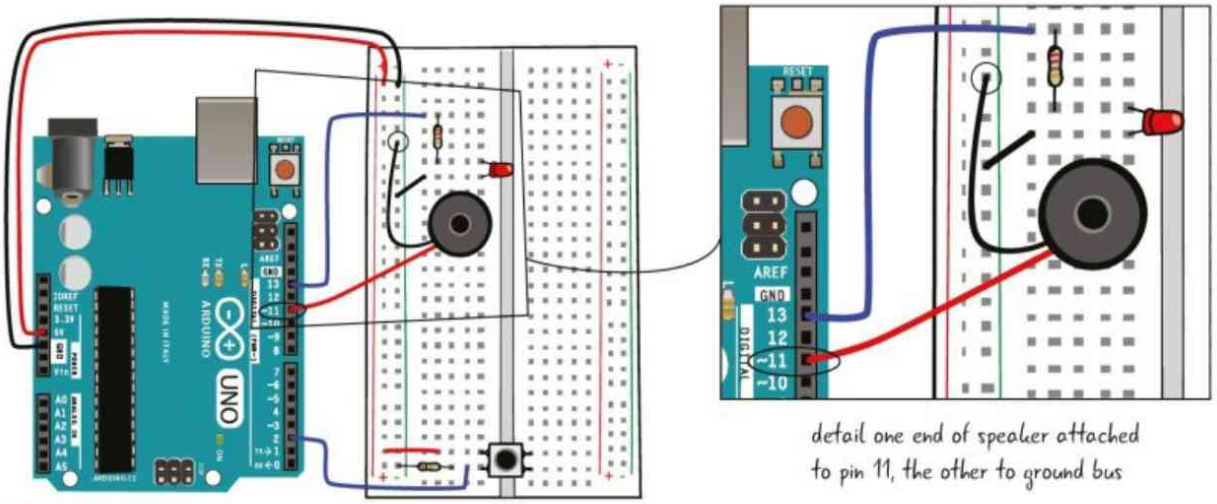


FIGURA 6.18 Montaje del altavoz.

Esto es todo lo que hay que hacer para añadir un altavoz. Ahora estás preparado para modificar el código.

adición del código para el altavoz

Ahora que has conectado el altavoz, adaptarás el código. Primero guarda el sketch como si fuera un sketch nuevo, dale el nombre **LEA6_1_tonebutton**.

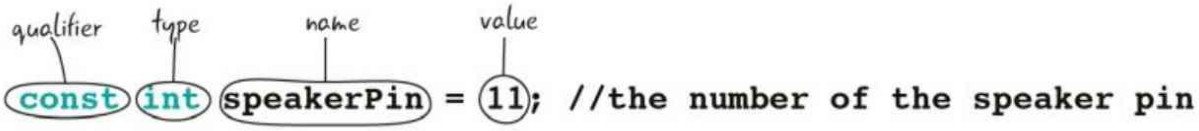
Vas a añadir una línea de código a la sección de inicialización del sketch y una variable para el pin del altavoz.

```
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;     //the number of the LED pin
const int speakerPin = 11; //the number of the speaker pin

// these variables will change:
int buttonState = 0;
```

add this variable to hold value of the speaker pin

Analicemos la nueva línea de código más de cerca. Puedes ver que es como las otras declaraciones de variables: hay un calificador, un tipo, un nombre y un valor. Recuerda que utilizas el calificador const, que significa constante, cuando tienes una variable cuyo valor no cambia.



Tip Es una buena idea añadir comentarios a medida que escribes el nuevo código para recordar lo que estás añadiendo en el sketch.

Ajuste del setup()

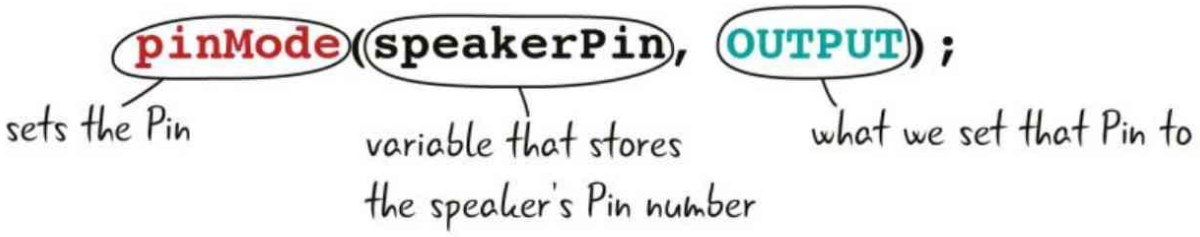
¿Qué crees que tendrás que añadir a la sección setup() del sketch? Recuerda que setup() es donde indicas si los diferentes componentes del circuito son entradas o salidas.

¿Qué es el altavoz? Una salida. Por lo tanto, añadirás una línea de código que declarará que el pin al que está conectado el altavoz es una salida. Debido a que creas una variable que contiene este valor, la usarás cuando declares el pin como salida.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
  pinMode(speakerPin, OUTPUT);
}
```

use the pinMode function to declare speakerPin an output

He aquí una vista más de cerca de esta nueva línea de código de setup():



¡Vamos a ver loop()!!

Ajuste de loop()

Como has visto, el código de loop() indica si se presiona o no el botón, y luego usa un condicional para ordenar al Arduino hacer algo utilizando esa información. Ahora harás uso de las funciones de Arduino tone() y noTone() dentro del condicional. tone() generará una nota o tono; noTone() impedirá que se reproduzca el tono. Veamos primero todo el código loop(), y luego exploraremos más de cerca las funciones tone() y noTone().

```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
    tone(speakerPin, 330);
  }
  else {
    //turn speaker off
    noTone(speakerPin);
    digitalWrite(ledPin, LOW);// turn LED off:
  }
}
```

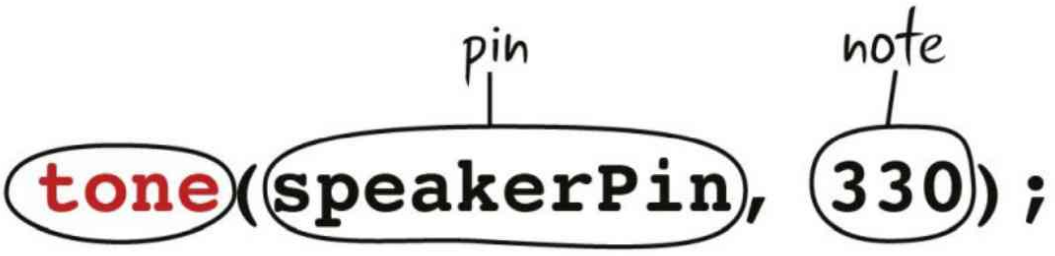
loop function

tone function

noTone function

Como dijimos, tone() generará una nota o tono que reproducirá el altavoz

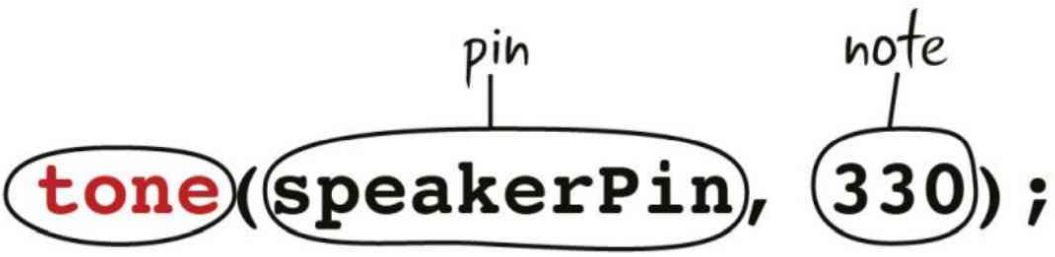
que acabas de conectar. Cuando utilizas la función tone(), necesitas decirle al Arduino en qué pin generar el tono y qué nota sonará. Tiene sentido que quieras generar una nota precisamente en el pin que tiene el altavoz conectado.



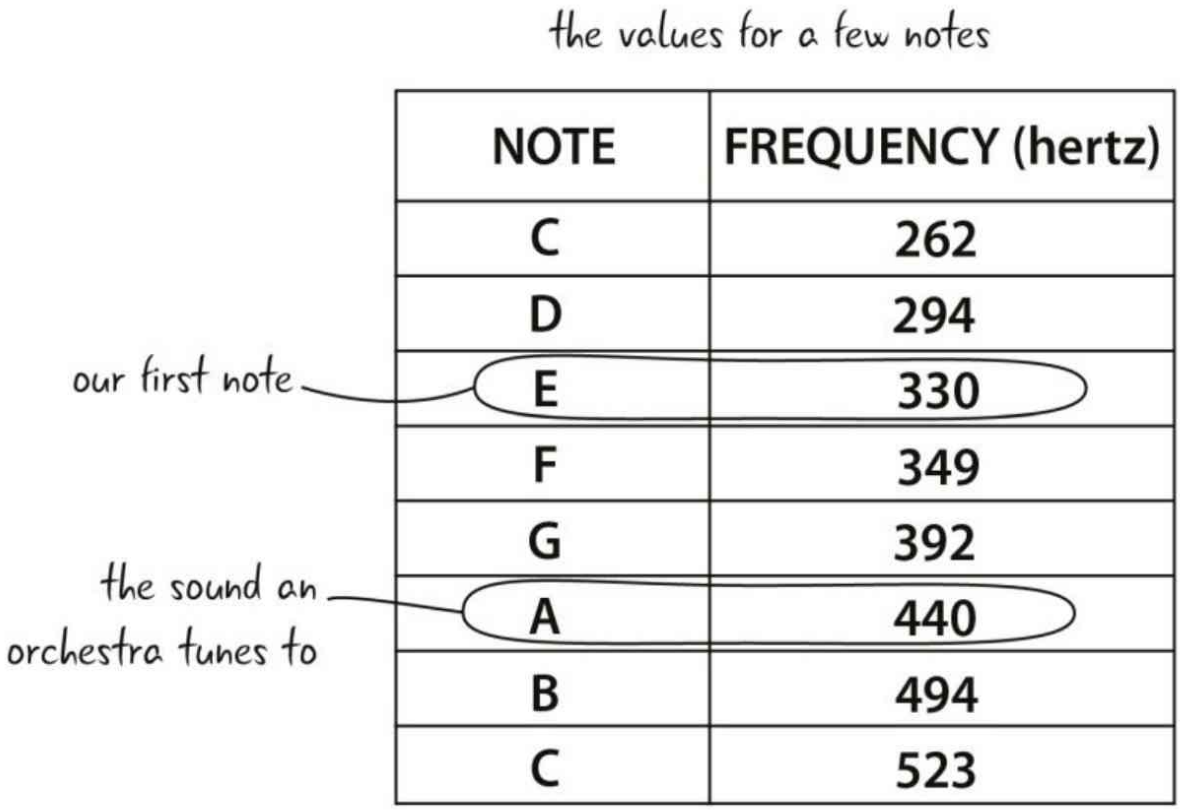
Veamos más a fondo las funciones tone() y noTone().

tone() y noTone() de cerca

¿Qué significa 330? Sabes que es la nota que sonará en el altavoz, pero ¿cómo descubres ese número? Arduino genera ondas sonoras cuya frecuencia se mide en hercios; 330 es el valor en hercios de la nota que quieres que interprete el circuito.



En el mundo de la música, esta nota se conoce como E. A partir de ahora, cuando mencionemos la función tone(), diremos que genera una nota. La figura 6.19 muestra algunos de los posibles valores de las notas.

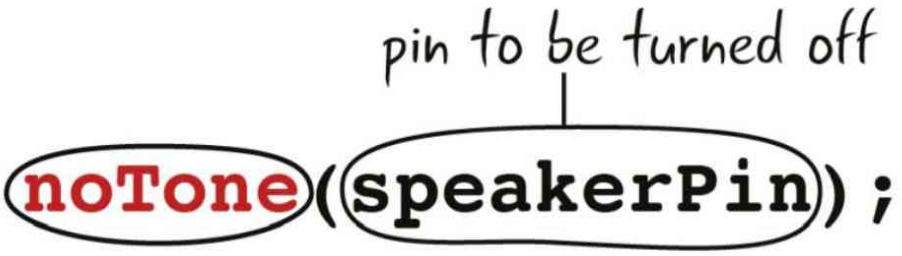


NOTE	FREQUENCY (hertz)
C	262
D	294
E	330
F	349
G	392
A	440
B	494
C	523

FIGURA 6.19 Tabla de notas.

> **Tip** Se puede ver una tabla de notas más completa en el sitio web de Arduino arduino.cc/en/Tutorial/ToneMelody.

Ahora veamos noTone(). Esta función detiene la reproducción del sonido en el pin especificado. En este caso, es speakerPin, que almacena el valor del pin que tiene el altavoz conectado. Si omites noTone(), entonces la nota se reproducirá continuamente una vez que se presiones el botón por primera vez.

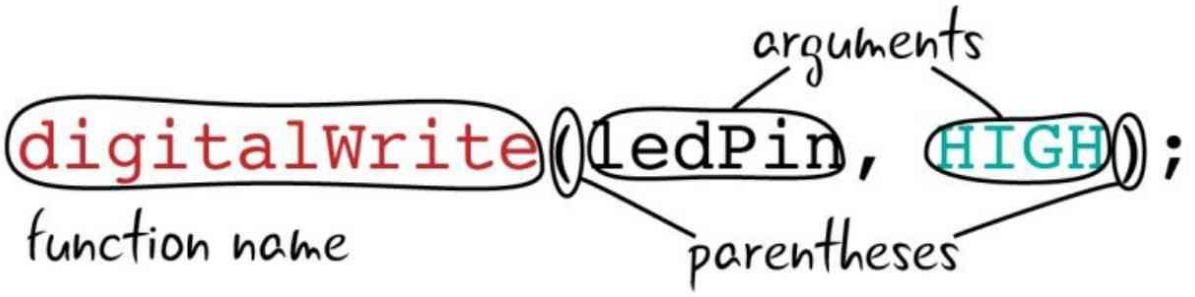


Añade las funciones tone() y noTone() al código, guarda el sketch, y cárgalo. Ahora, cuando presiones el botón, oirás una nota procedente del altavoz y verás que el led se enciende.

Antes dijimos que explicaríamos lo que hay entre paréntesis en las funciones. Por ejemplo, en la función tone(), ¿qué significa speakerPin y 330? Estos valores se llaman *argumentos*. Vamos a echarles un vistazo ahora.

Argumentos

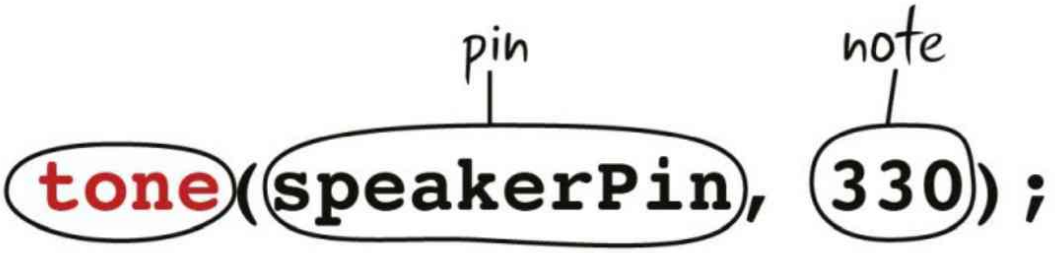
Hasta ahora has utilizado varias funciones de Arduino en este libro, desde pinMode() a digitalWrite(). Puede que hayas notado que la mayoría de las funciones requieren que se coloque algo entre paréntesis, a menudo una combinación de números y palabras. Los valores colocados dentro del paréntesis de la función se llaman *argumentos*.



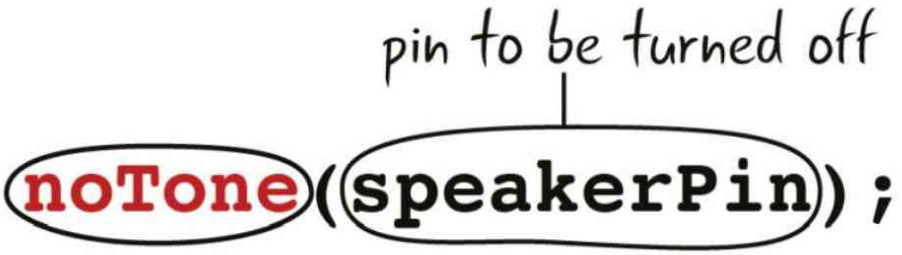
Los argumentos le proporcionan a la función de Arduino información importante, como qué pines se utilizan como entradas. Las distintas funciones tienen a menudo un número diferente de argumentos. digitalWrite()

tiene dos argumentos (el número de pin y el valor) mientras que la función de retardo de Arduino delay() solo tiene un único argumento (cuántos milisegundos se detiene el programa). Algunas funciones no necesitarán argumentos, mientras que otras requerirán varios. Veamos las funciones tone() y noTone() y cómo funcionan los argumentos con ellas.

En la función tone(), se introducen dos argumentos: el pin al que está conectado el altavoz (en este caso, la variable speakerPin) y el valor de la nota. Hay que tener en cuenta que los valores del argumento están separados por una coma.



La función noTone() tiene un argumento: el pin al que está conectado el altavoz (de nuevo, la variable speakerPin).



Algunas funciones no tienen argumentos, mientras que otras tienen muchos. En algunas funciones no se requieren todos los argumentos. Aprenderás más sobre funciones y argumentos en capítulos posteriores.

A continuación añadirás un segundo botón a tu teclado para que puedas

tocar más de una nota.

Añade dos Botones más y modifica el código

Vas a agregar otro botón al circuito para poder tocar una melodía de dos notas en el instrumento con un mini teclado (figura 6.20). Necesitarás otro botón, otra resistencia de 10 KΩ y más cables de puente. Recuerda desconectar Arduino del ordenador antes de añadir el botón al circuito.

Piezas que necesitas:

- 1 interruptor pulsador momentáneo
- 1 resistencia de 10 KΩ (marrón, negro, naranja, dorado)
- Cables de puente

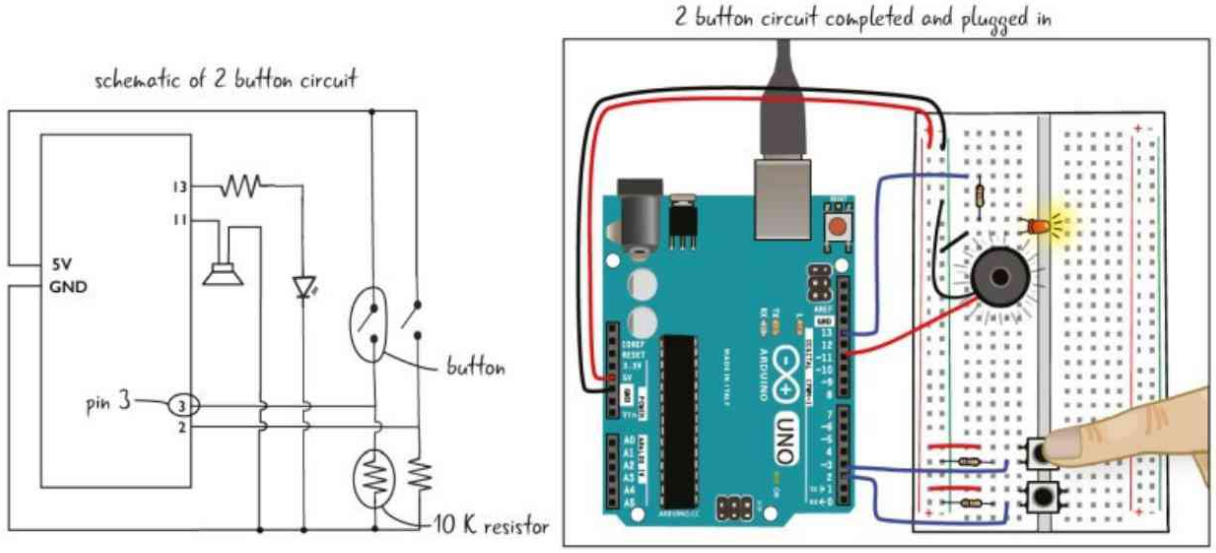


FIGURA 6.20 Circuito con dos botones.

La configuración de este botón será muy similar a la del primer botón que instalaste en el circuito, con la excepción de que el nuevo botón se conectará a un pin diferente en el Arduino (figura 6.21).

Coloca el botón nuevo a ambos lados de la línea divisoria de la placa. Utiliza un cable de puente para conectar el pin superior izquierdo del botón al bus de alimentación. Conecta el conductor de la resistencia de 10 KΩ a tierra y el otro conductor al pin inferior izquierdo del botón. Finalmente, conecta el pin 3 al pin inferior izquierdo del botón y un extremo de la resistencia de 10

KΩ con un cable puente.

Ahora que has añadido el segundo botón, es hora de reprogramar el código del sketch.

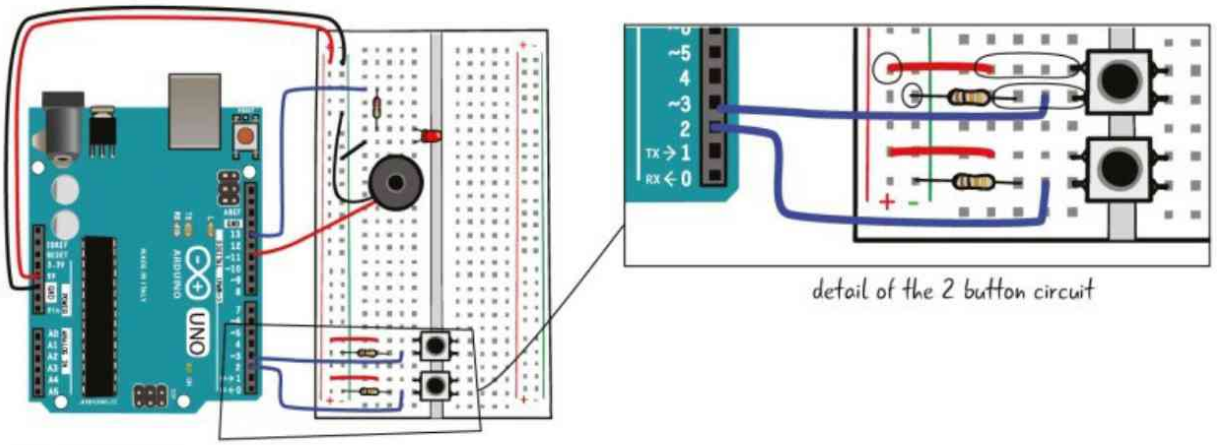


FIGURA 6.21 Instalación del segundo botón.

Edición de LEA6_2_tonebuttons

Primero, guarda el sketch como **LEA6_2_tonebuttons**. Editarás el código, añadirás las líneas marcadas en negrita en el siguiente par de páginas.

Ajustes del código de inicialización

Aquí está el código de inicialización actualizado con dos nuevas variables. Una tiene el valor del número del pin que has conectado al segundo botón (3); la otra tendrá el estado de ese botón, e inicialmente está puesto a 0.

```
const int buttonPin = 2;    // the number of the pushbutton pin
const int buttonPin2 = 3;   // a second pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status
int buttonState2 = 0;       // variable holds second pushbutton state
```

variable set to number of pin attached to the second button

this variable will hold the state of the second button, as a 0 or 1 to indicate whether it is being pressed

Ajustes del código de setup()

El siguiente gráfico muestra de nuevo el código de setup(), editado para tener en cuenta el segundo botón. Vuelve a utilizar la función pinMode(), esta vez para configurar buttonPin2 (que pones a 3 en el código de inicialización) como entrada.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pins as inputs:
  pinMode(buttonPin, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(speakerPin, OUTPUT);
}
```

A continuación echemos un vistazo al código loop() y veamos lo que necesitas añadir.

Ajuste del código loop(): else if

Puedes ver que estás leyendo el valor de buttonPin2 (bien 1 para encendido o 0 para apagado) con la función digitalRead() y lo almacenas en la variable buttonState2.

También tienes que agregar una nueva sección dentro de la declaración if, lo que se conoce como else if. Cuando la declaración if se evalúa, comprobará la primera condición después de if para ver si esa condición es verdadera o falsa. Como viste antes, si la primera condición es verdadera (en otras palabras, si el botón 1 está pulsado en ese momento), el altavoz reproducirá el tono de 330 hercios. Pero si la primera condición es falsa (en otras palabras, si el botón 1 *no* está pulsado en ese momento), entonces el Arduino pasará al código else if para determinar si la declaración que sigue a else if es

verdadera o falsa. Si es verdadera (en otras palabras, si en ese momento tienes pulsado el botón 2), entonces el Arduino seguirá las instrucciones que aparecen dentro de las llaves.

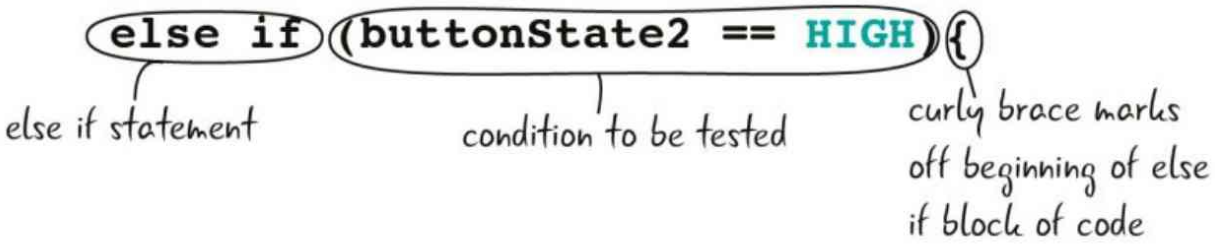
Veamos cada una de las líneas de else if.

```
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  buttonState2 = digitalRead(buttonPin2);
  // check if the pushbutton is pressed.
  if (buttonState == HIGH) {
    digitalWrite(ledPin, HIGH);
    tone(speakerPin, 330);
  }
  // check if the second button is pressed
  else if (buttonState2 == HIGH) {
    digitalWrite(ledPin, HIGH);
    tone(speakerPin, 294);
  }
  else {
    noTone(speakerPin); //turn speaker off
    digitalWrite(ledPin, LOW); // turn LED off:
  }
}
```

reading state of buttonPin2 and storing it in buttonState2

else if tests for second condition

En primer lugar, else if nos dice que el Arduino va probar otra condición. Prueba si buttonState2 está HIGH, es decir, ¿se está pulsando el botón del pin3?



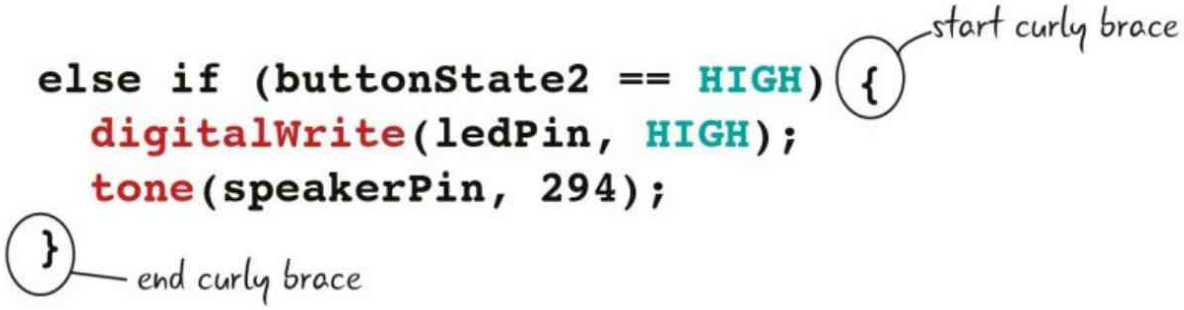
Ya has visto antes el código de la siguiente línea del bloque else if; pone el pin conectado al led en HIGH para que este se encienda.

```
digitalWrite(ledPin, HIGH);
```

Y aquí está la última línea en el bloque else if. Utiliza la función tone() para reproducir la nota en el altavoz. Esta vez, la nota es de 294 hercios, ligeramente más baja que la nota que toca el primer botón.

```
tone(speakerPin, 294);
```

El siguiente gráfico muestra de nuevo el bloque completo del código de else if. En este bloque los paréntesis contienen el código que describe la condición que se está probando, y las llaves contienen lo que quieres hacer si la condición es verdadera.



Para probar el código, conecta el ordenador al Arduino, guarda el código, verifícalo y súbelo al Arduino.

Ahora puedes tocar dos notas diferentes en el teclado que tiene dos botones.

¿PREGUNTAS?

P: ¿Qué ocurrirá si pulso ambos botones a la vez?

R: Por la forma en la que hemos escrito la declaración condicional, si pulsas los dos botones solo sonará la primera nota. Esto te viene bien porque la función Arduino `tone()` no puede reproducir más de un tono a la vez.

P: ¿Qué sucederá si cambio el número de la segunda nota en la función `tone()` a otro número que no sea 294?

R: La tabla de notas que viste antes ofrece una selección de las posibles notas que puedes reproducir. Hemos omitido todas las notas agudas/graves (esto no es una lección de teoría musical), pero si eliges aleatoriamente un número para la segunda nota, lo más probable es que suene un poco fuera de tono (como una guitarra desafinada).

Adición de un Tercer Botón

Ahora vas a añadir un tercer y último botón al circuito (figura 6.22). Asegúrate de desconectar el Arduino del ordenador antes de añadir el botón!

Piezas a añadir:

- 1 interruptor de botón momentáneo
- 1 resistencia de 10 KΩ (marrón, negro, naranja, dorado)
- Cables de puente

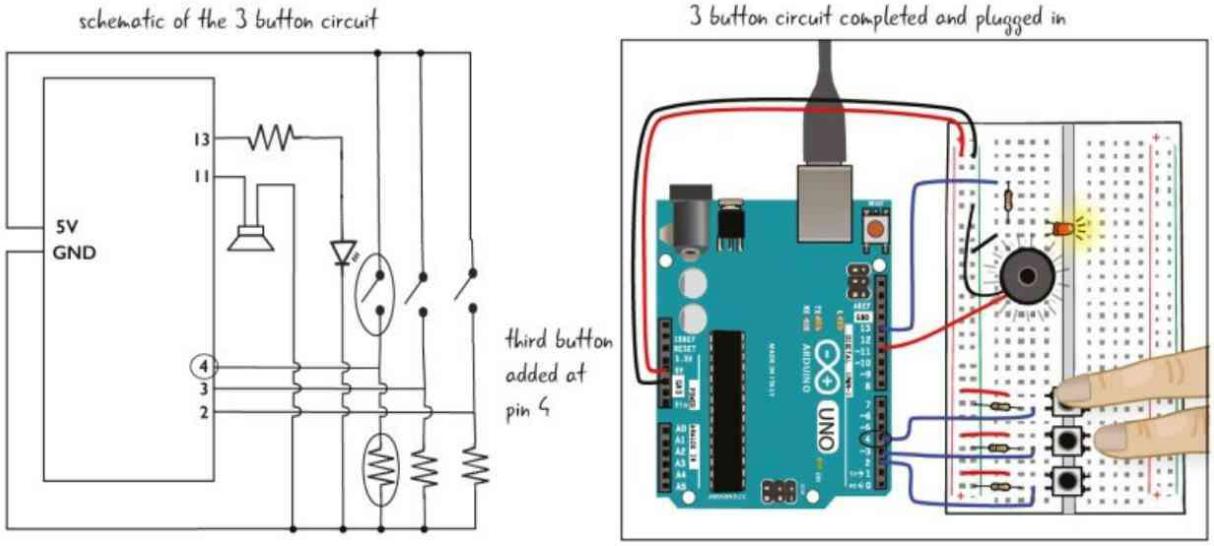


FIGURA 6.22 Circuito con tres botones.

Coloca el botón a ambos lados de la línea divisoria de la placa, por encima de los otros dos. Utiliza un puente para conectar el pin superior izquierdo del botón al bus de alimentación. Conecta uno de los terminales de la resistencia de 10 KΩ a tierra y el otro extremo al pin inferior izquierdo del botón. Finalmente, conecta el pin 4 al pin inferior izquierdo del botón y a un extremo de la resistencia de 10 KΩ (figura 6.23).

Ahora que ya has añadido el tercer botón al circuito, ajusta el sketch.

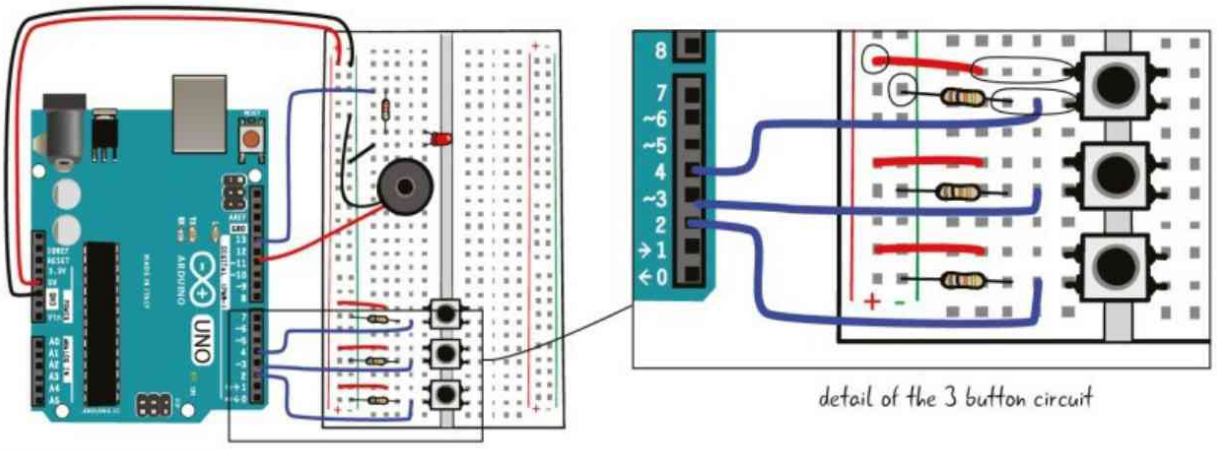


FIGURA 6.23 Adición de un tercer botón.

Edición de LEA6_3_tonebuttons Sketch

Guarda el sketch como **LEA6_3_tonebuttons**; ahora veremos el código que debes ajustar. Será muy similar a los ajustes que hiciste en el sketch anterior, cuando añadiste el segundo botón. Vamos a avanzar y a editar tu sketch para que coincida con el siguiente código.

Edición del código de inicialización

Vas a conectar el tercer botón al pin 4. También vas a añadir una variable llamada buttonState3, que mantiene el valor que indica si se está pulsando o no el pin (1 o 0, HIGH o LOW, encendido o apagado).

```
const int buttonPin = 2;    // the number of the first pushbutton pin
const int buttonPin2 = 3;   // a second pushbutton pin
const int buttonPin3 = 4;   // third pushbutton pin attached to pin 4
const int ledPin =  13;     // the number of the LED pin
const int speakerPin = 11; // the number of the speaker pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status
int buttonState2 = 0;       // variable holds second pushbutton state
int buttonState3 = 0;       // third pushbutton state
```

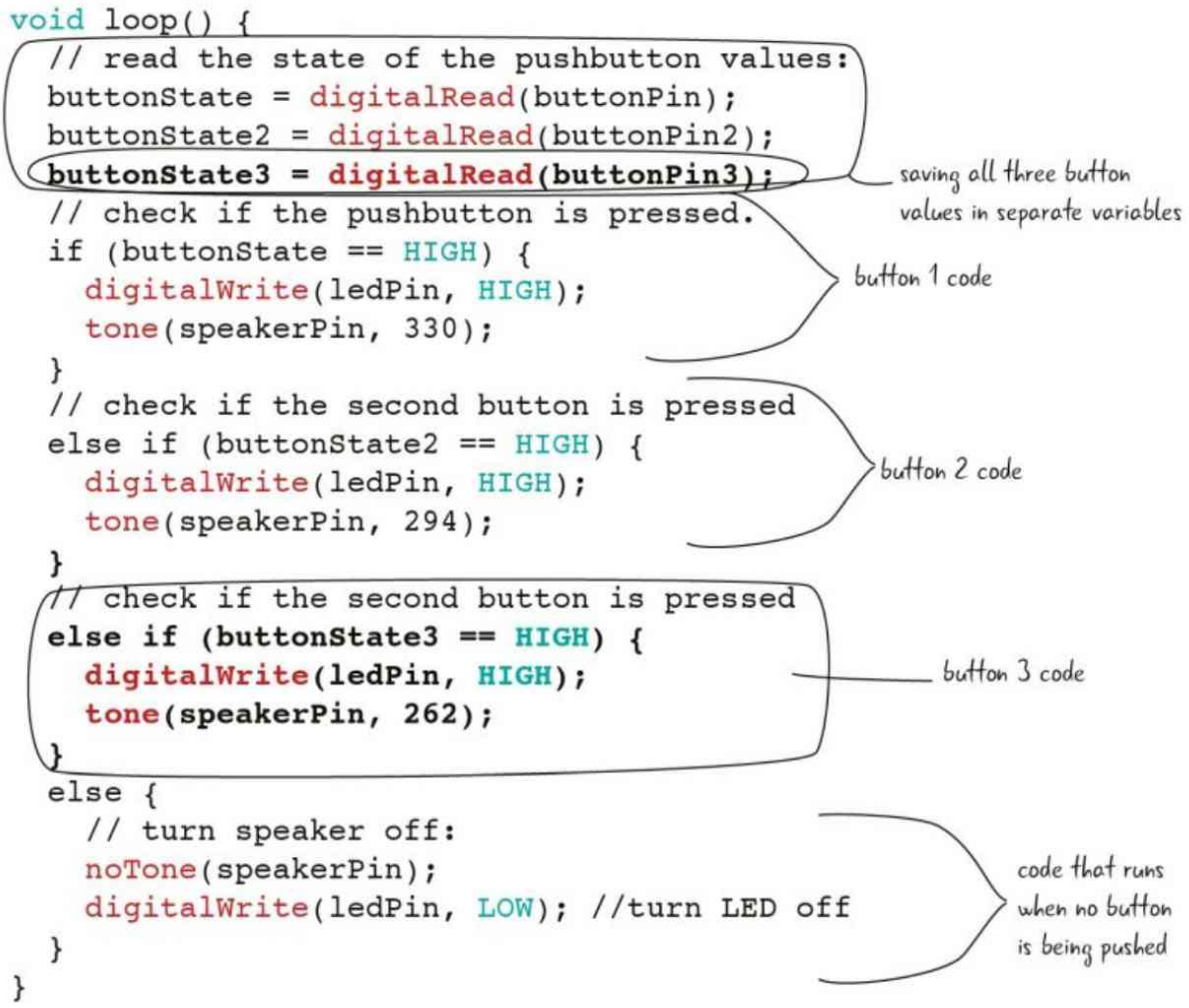
Edición del código de *setup()*

En nuestro código setup(), fijamos la variable buttonPin3 (que mantiene el valor 4 para indicar el pin 4) como ENTRADA. Los tres botones se han configurado como ENTRADAS.

```
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pins as inputs:
  pinMode(buttonPin, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);
  pinMode(speakerPin, OUTPUT);
}
```

La función *loop()* del "instrumento" de tres botones

A continuación se muestra la función loop() actualizada. Esta función lee el estado de buttonPin3 y lo almacena en buttonState3. También tiene una instrucción else if adicional, que comprueba si se está pulsando el tercer botón y si es así, reproduce una nota (un poco más baja que la nota del botón 2) y enciende el led.



El código ahora responde cuando pulses cada uno de los tres botones.

toca el instrumento de teclado miniatura

Has escrito el código y has ajustado el circuito. El instrumento con teclado miniatura de tres botones debería funcionar. Para hacer un repaso, conecta el ordenador a tu Arduino, guarda el código,verifícalo, súbelo al Arduino y pulsa los botones. Recuerda que hay que pulsar un botón a la vez, ya que el altavoz puede tocar solamente una nota a la vez.

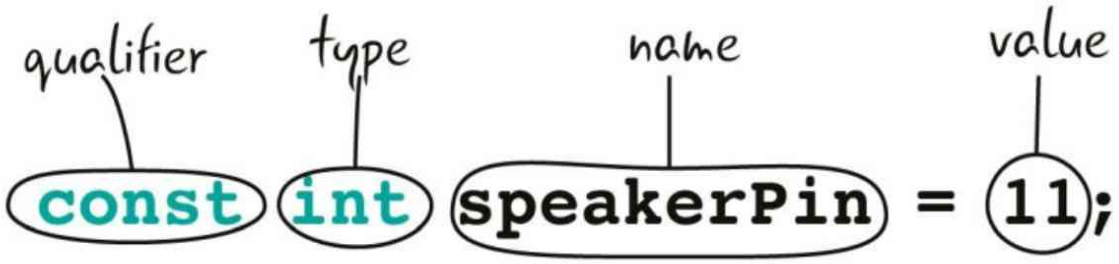
Antes de pasar a ver cómo funcionan algunos de los componentes en este circuito, repasemos brevemente lo que has aprendido sobre escribir código cuando has desarrollado este proyecto.

Revisión de conceptos electrónicos y de Código

En este capítulo has aprendido algunos conceptos de programación nuevos y muy importantes. Estos conceptos son críticos para escribir código en todos los lenguajes de programación, aunque los detalles pueden ser un poco diferentes dependiendo del lenguaje. Veamos una vez más las variables y las declaraciones condicionales.

Variables

En el código, una variable es un contenedor de código que puede contener diferentes valores.



declaraciones Condicionales

Una declaración condicional evalúa y ejecuta instrucciones si la condición es verdadera. Si la declaración condicional contiene un bloque else if o else opcional, entonces la declaración puede evaluar múltiples condiciones, y a veces le dice al código que haga algo si las condiciones no son ciertas.

```
if (buttonState == HIGH) {
  digitalWrite(ledPin, HIGH);
}
// check if the second button is pressed
else if (buttonState2 == HIGH) {
  digitalWrite(ledPin, HIGH);
}
```

Veamos rápidamente cómo funcionan en el circuito los componentes electrónicos que has utilizado en este capítulo.

¿cómo funciona el Botón?

Por defecto, cuando no se ejerce presión sobre el botón, este está abierto, lo que significa que la electricidad no pasa a través de él. Para que la electricidad pase por el botón, tienes que hacer presión sobre él, se establece así una conexión entre los pines (figura 6.24). Si lees el valor del pin conectado al botón, verás que es HIGH.

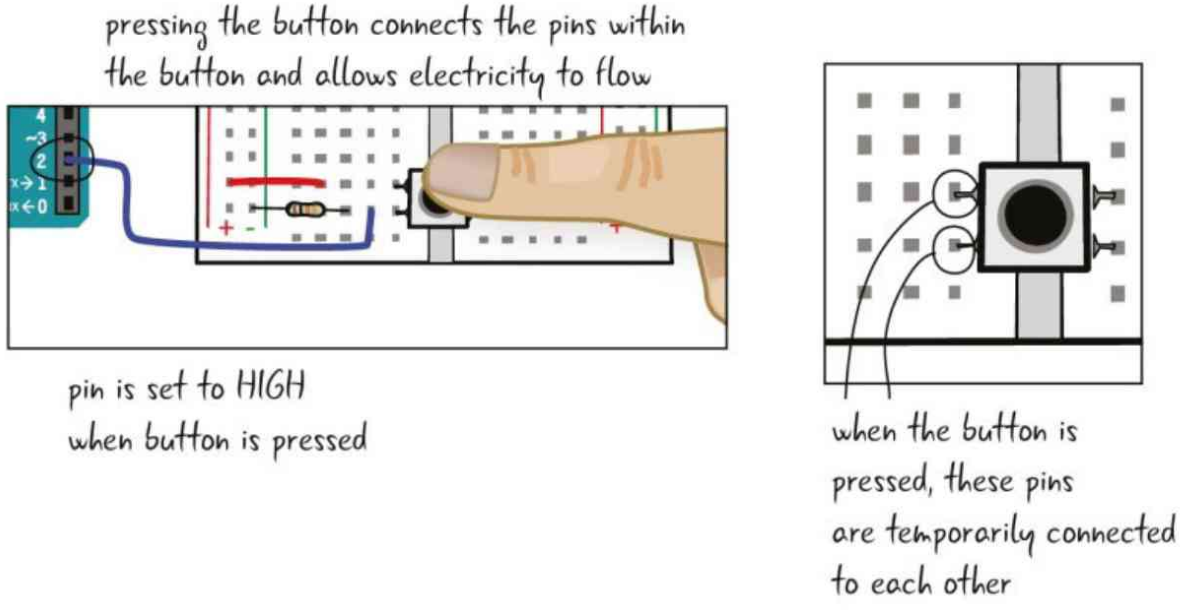


FIGURA 6.24 Pulsación del botón.

No hay nada conectado a los pines al otro lado de la línea divisoria de la placa, pero cuando se pulsa el botón, los pines en ese lado de la línea están también conectados (figura 6.25).

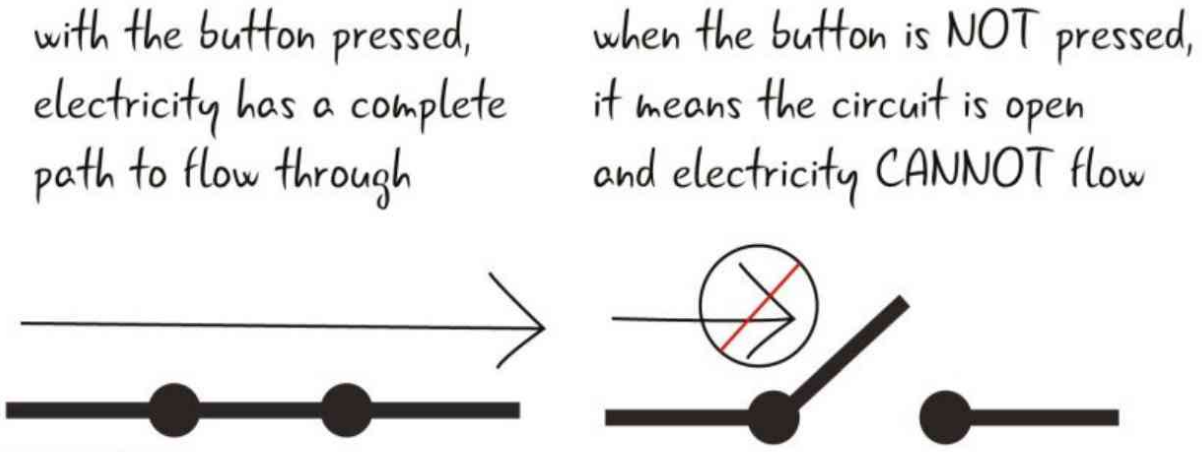


FIGURA 6.25 Funcionamiento del interruptor.

¿cómo produce el altavoz Diferentes Notas?

La función tone() integrada en el Arduino sabe cómo cambiar la energía proporcionada por el pin digital para crear notas diferentes en el altavoz. Sin llegar a ser demasiado técnicos, el valor de la nota que incluyes en la función tone() le indica al Arduino cómo cambiar rápidamente el voltaje para crear diferentes notas (figura 6.26).

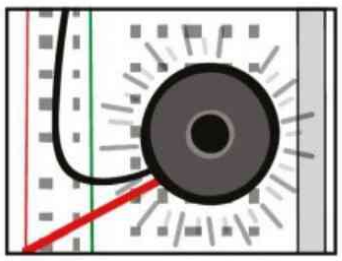
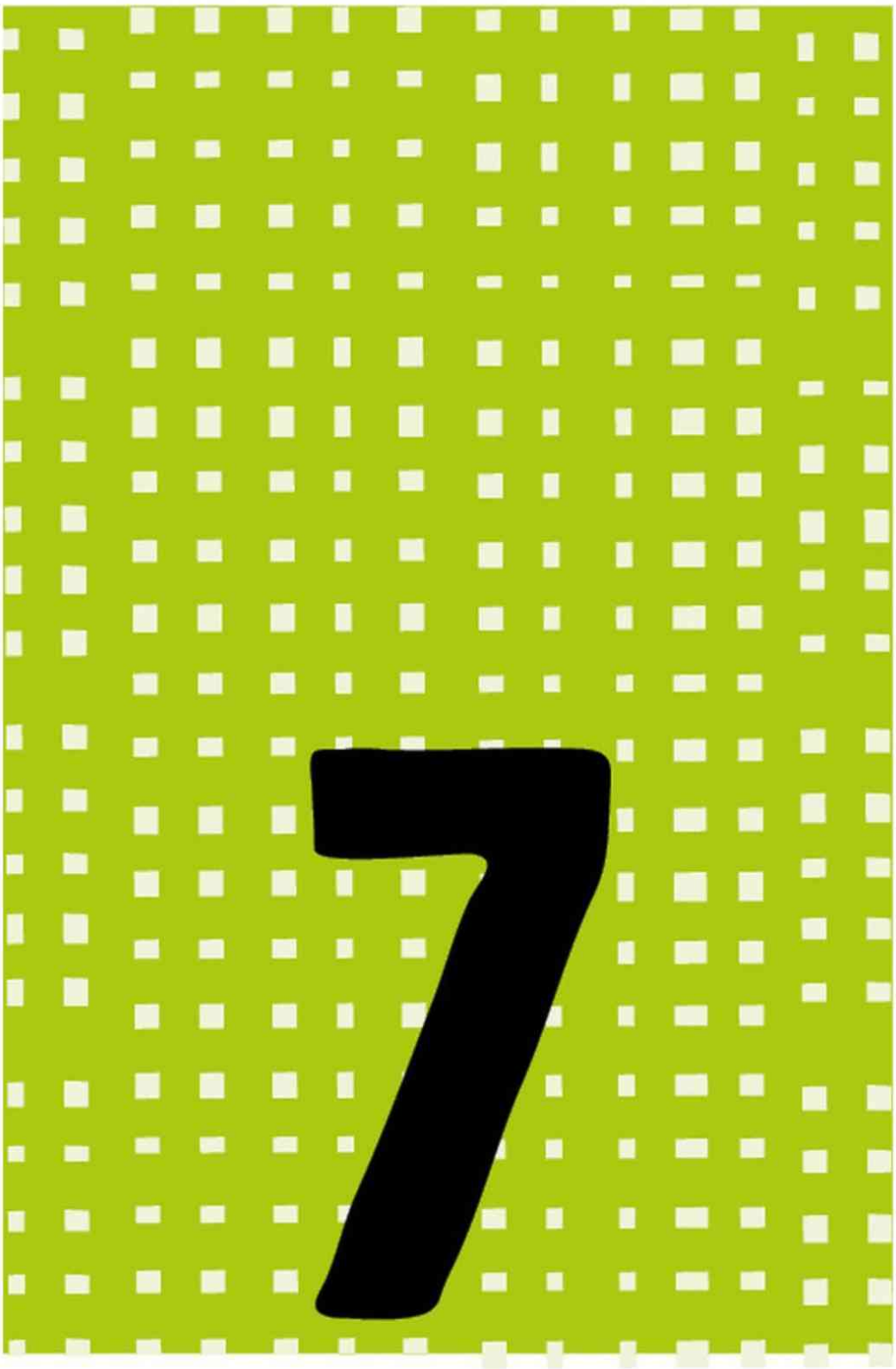


FIGURA 6.26 Un cambio en el voltaje del altavoz produce diferentes notas.

resumen

En este capítulo has aprendido algo más sobre programación. Has aprendido lo que es una variable y cómo se utiliza, y cómo se utilizan las declaraciones condicionales para controlar el flujo del programa. También has aprendido algo más sobre las salidas digitales y cómo añadir una entrada digital al circuito para que el proyecto sea interactivo. Puedes descargar el código desde: github.com/arduinotogo/LEA/blob/master/LEA6_3_toneButtons.ino.

En el siguiente capítulo te mostraremos cómo conectar sensores analógicos u otras entradas al circuito, y utilizarás la información que obtengas de ellos para hacer más cosas que encender y apagar los componentes de salida.


7

valores Analógicos

En el capítulo anterior, aprendiste a instalar botones en un circuito para reproducir notas musicales utilizando un altavoz y a encender y apagar un led. En este capítulo te explicamos cómo conectar sensores a un circuito y utilizar la información que se obtiene de ellos para crear las experiencias más variadas. Aprenderás también cómo utilizar el IDE de Arduino para ver la información que llega de los sensores.

¡Hay más cosas en la vida que ceros y unos!

Has aprendido a añadir botones al circuito de manera que puedes llevar a cabo proyectos interactivos utilizando entradas y salidas digitales con los sketches del Arduino. Con una entrada digital, tienes solamente dos posibles valores: encendido o apagado (conocidos normalmente como HIGH o LOW, 1 o 0). Sin embargo, a veces es posible que desees utilizar valores que no son tan simples como encender o apagar algo. En este capítulo verás cómo leer valores de sensores y resistencias variables, y luego usar estos valores en los sketches del Arduino para producir diferentes efectos.

Aprenderás estos conceptos montando un circuito con un potenciómetro, que es un componente con un acoplamiento que se puede girar para proporcionar un rango de valores de voltaje que va más allá de los ceros y los unos. Usarás primero un potenciómetro para ajustar el brillo de un led y luego para hacer sonar distintas notas en un altavoz.

¿Por qué vamos a ver cómo utilizar los sensores e información analógicos? ¿Y qué entendemos por *analógico*?

Has visto que la información digital tiene solo dos posibilidades: activado y desactivado. La información analógica, por otra parte, puede contener un rango de valores posibles. Las personas percibimos el mundo como un flujo de información analógica a través de la vista, el oído y otros sentidos. Al usar la información analógica con el Arduino, puedes responder a los datos introducidos por el usuario de una manera más elaborada. Puedes controlar la luminosidad de un led, configurándolo para que aumente su brillo, se atenúe o muestre cualquier rango de valores intermedios.

> Note **La información analógica es continua y puede contener un rango de valores posibles.**

Una vez que entiendas cómo funcionan los valores analógicos, los

utilizarás para crear un instrumento musical casero llamado *theremin*. Un theremin es un instrumento musical en el que la afinación del sonido está controlada por la distancia entre las manos del músico y el instrumento, como se muestra en la figura 7.1. (Es cierto: realmente no tocas un theremin para interpretar música.) Es posible que hayas escuchado los tonos perturbadores de un theremin en una banda sonora de una película o en un programa de televisión. Nuestra versión utilizará un altavoz y una célula fotoeléctrica; a medida que separes o acerques la mano a la célula fotoeléctrica, el altavoz reproducirá diferentes notas.

En todos los proyectos de este capítulo, un sketch leerá la información de una entrada analógica y luego usará esa información para controlar una salida, como puede ser el brillo de un led o los tonos que emite un altavoz.

En este capítulo utilizarás información analógica, así como en algunos de los proyectos de los próximos capítulos. ¡Vamos a empezar!

FIGURA 7.1 Tocando un theremin.

el circuito con Potenciómetro

La figura 7.2 muestra el esquema y un dibujo del primer circuito del

capítulo. El circuito utiliza un potenciómetro para cambiar el brillo del led. El brillo aumenta a medida que giramos el potenciómetro, hasta que llega a su máximo brillo, mientras que si lo giras en sentido contrario, el led va reduciendo su brillo hasta que se apaga.

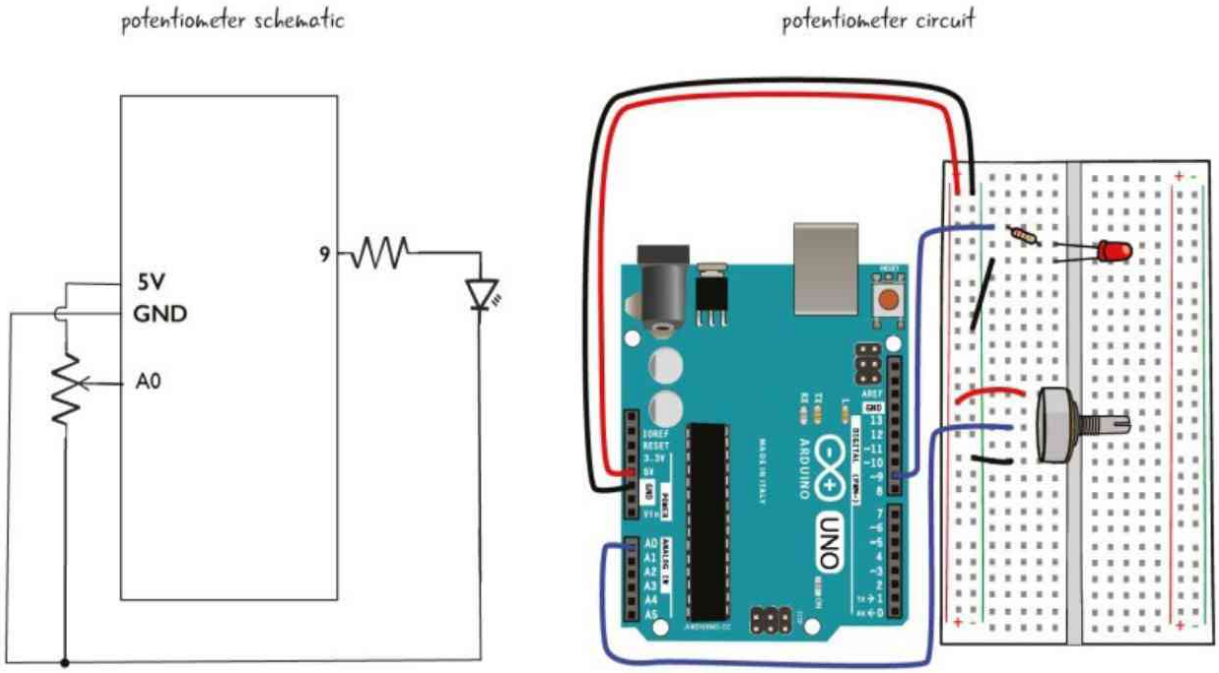


FIGURA 7.2 Este es el primer circuito que montarás en este capítulo.

Antes de empezar a montar el circuito, vamos a ver los pines de entradas analógicas de Arduino.

pines de entradas analógicas de arduino

¿Recuerdas en el capítulo 2, "Tu Arduino," cuando desembalaste el equipo? Señalamos que tenía pines de entradas analógicas, que son pines que pueden leer sensores que tienen un rango de valores posibles. Veamos más detalladamente estos pines en la figura 7.3.

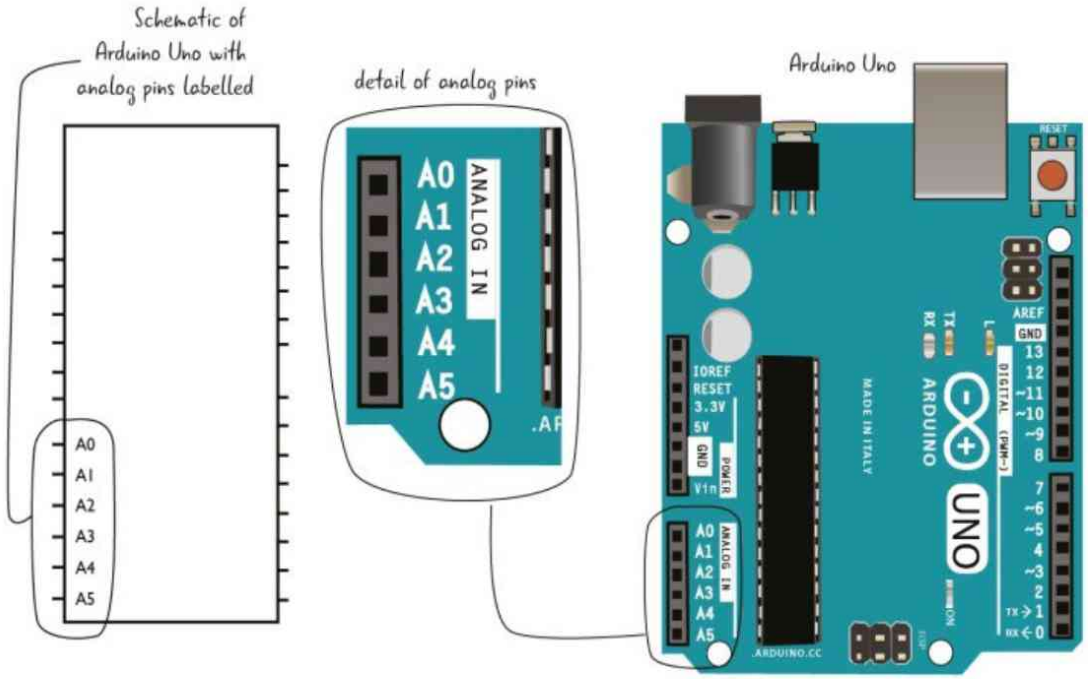


FIGURA 7.3 Pines analógicos en el Arduino Uno.

Los pines de entradas analógicas están situados frente a los pines de entradas/sal-idas digitales del Arduino, debajo de los pines de alimentación y tierra. Hay seis pines etiquetados desde A0 a A5, en los que la "A" indica que son entradas "analógicas".

Cuando uno de estos pines está conectado a una entrada analógica, puede proporcionar un rango de valores de 0 a 1.023. Este rango de números está relacionado con la forma en que el Arduino maneja la memoria. Una explicación detallada va más allá del alcance de este libro; lo que es importante que sepas es que es un rango mucho más grande que el de solo ceros y unos, que te permite crear experiencias muy variadas, en lugar de simplemente encender y apagar algo.

¿Qué es una entrada analógica? Cualquier componente, a menudo algún tipo de sensor, que pueda proporcionarte un rango de valores, no solo encendido y apagado. La primera entrada analógica con la que vas a trabajar es un potenciómetro, que conectarás al pin A0.

El esquema de la figura 7.3 muestra la ubicación de todos los pines en tu

Arduino.

conocer el potenciómetro

Un potenciómetro es un tipo de resistencia variable, lo que quiere decir que su resistencia puede variar. Un potenciómetro, a veces llamado *pot* (en inglés) es un botón o disco que se puede girar para aumentar o disminuir la resistencia en función de hasta dónde y en qué dirección se gira (figura 7.4). Los potenciómetros se presentan en muchos tamaños y formas. En tu circuito utilizarás un potenciómetro de 10 KΩ.

Un potenciómetro tiene tres pines, uno de ellos se conecta a la alimentación, otro se conecta a tierra y otro se conecta a un pin del Arduino. En las siguientes páginas te explicaremos cómo conectar el potenciómetro a una placa de pruebas.

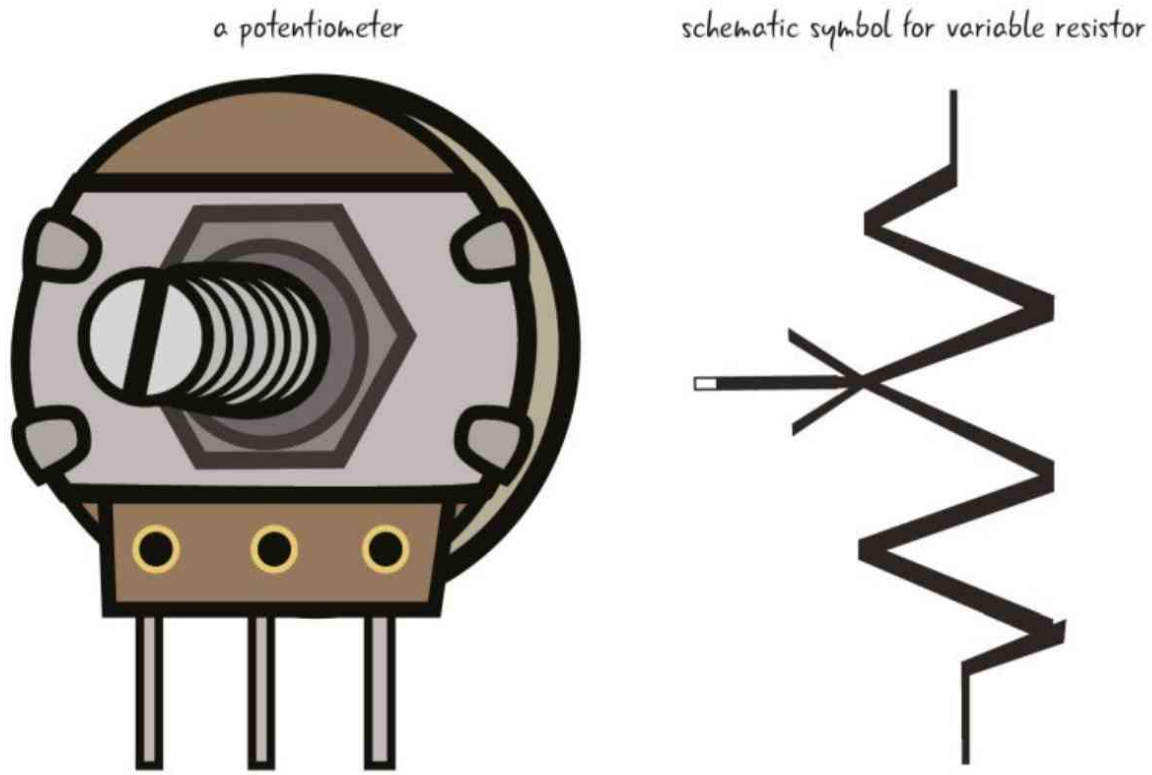


FIGURA 7.4 Dibujo y esquema del potenciómetro.

Note Una resistencia variable proporciona distintos valores de resistencia.

¿PREGUNTAS?

P: ¿Es un potenciómetro igual que el mando de un televisor antiguo?

R: No exactamente. El sintonizador de un televisor antiguo tiene puntos de ajuste, en los que se detiene cuando lo giras para "sintonizar" un canal. Un potenciómetro tiene generalmente puntos de parada en ambos extremos, en los que presenta resistencia máxima y mínima. Puedes hacerlo girar suavemente entre los dos extremos.

el circuito del potenciómetro, paso a paso

El primer circuito que montarás llevará un potenciómetro que controlará el brillo de un led (figura 7.5).

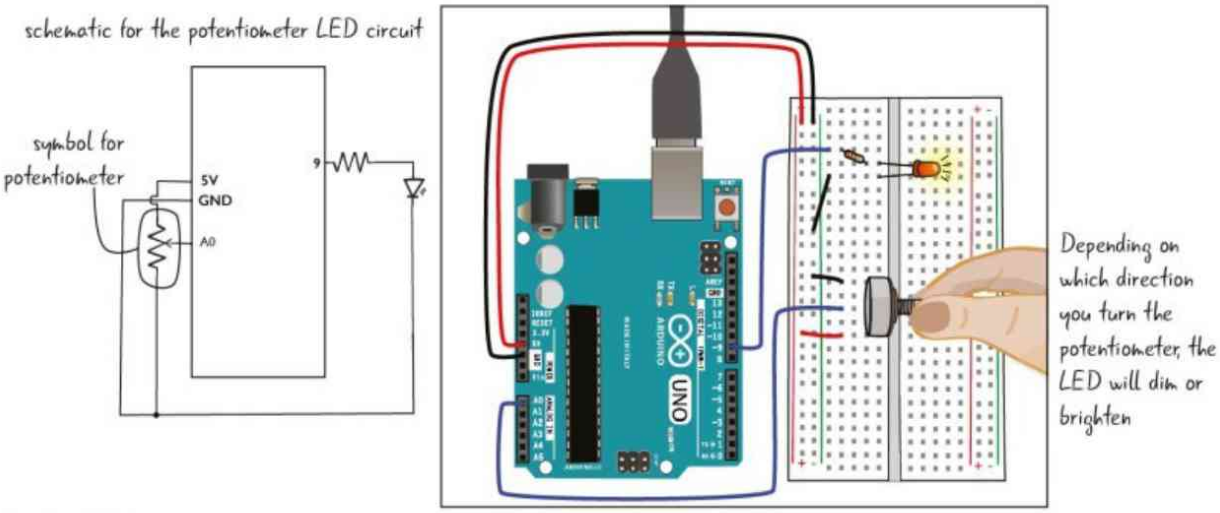


FIGURA 7.5 Circuito completo con potenciómetro y led.

Necesitarás los siguientes componentes:

- 1 led
- 1 resistencia de 220 ohm (rojo, rojo, marrón, dorado)
- 1 potenciómetro de 10 KΩ
- Cables de puente
- Placa de pruebas
- Arduino Uno
- Cable USB tipo A-B
- Ordenador con el IDE de Arduino

Empezarás con un circuito básico en el que el ánodo del led se conecta a través de una resistencia de 220 ohmios al Arduino y el cátodo se conecta a tierra. Hay una diferencia clave entre este circuito y el circuito que has utilizado en los capítulos anteriores: empleas el pin 9 en lugar del pin 13 en la placa de Arduino, como se muestra en la figura 7.6. Pronto explicaremos el motivo.

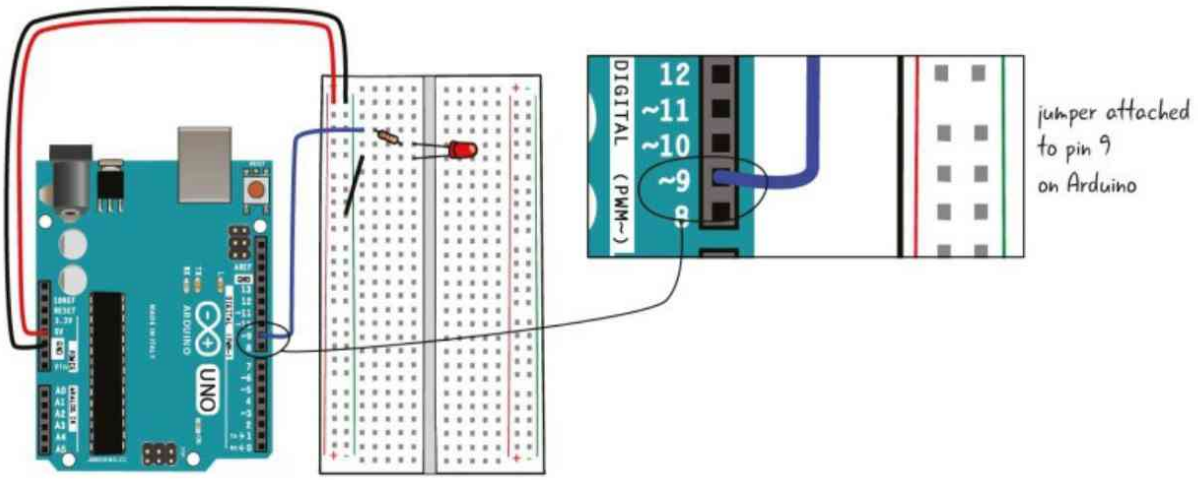


FIGURA 7.6 Led conectado al pin 9.

A continuación colocarás el potenciómetro en la placa de pruebas.

instalación del potenciómetro

Como has visto, un potenciómetro tiene tres pines. En el circuito, conectarás el pin central a un pin del Arduino, uno de los pines exteriores, al bus de alimentación y el otro pin exterior, al bus de tierra. No importa qué pin externo conectes a la alimentación y cuál de ellos a tierra.

Coloca el potenciómetro paralelo a la trinchera, como se muestra en la figura 7.7. Cada uno de los pines del potenciómetro se encuentra en una fila separada de puntos de anclaje, dejando un punto de anclaje libre entre cada uno de los pines. Orienta el vástago del potenciómetro hacia fuera del Arduino, con el eje sobre la zanja, como se puede ver en la figura 7.7. Esto te facilitará el acceso al potenciómetro para girarlo e integrarlo en el resto del circuito.

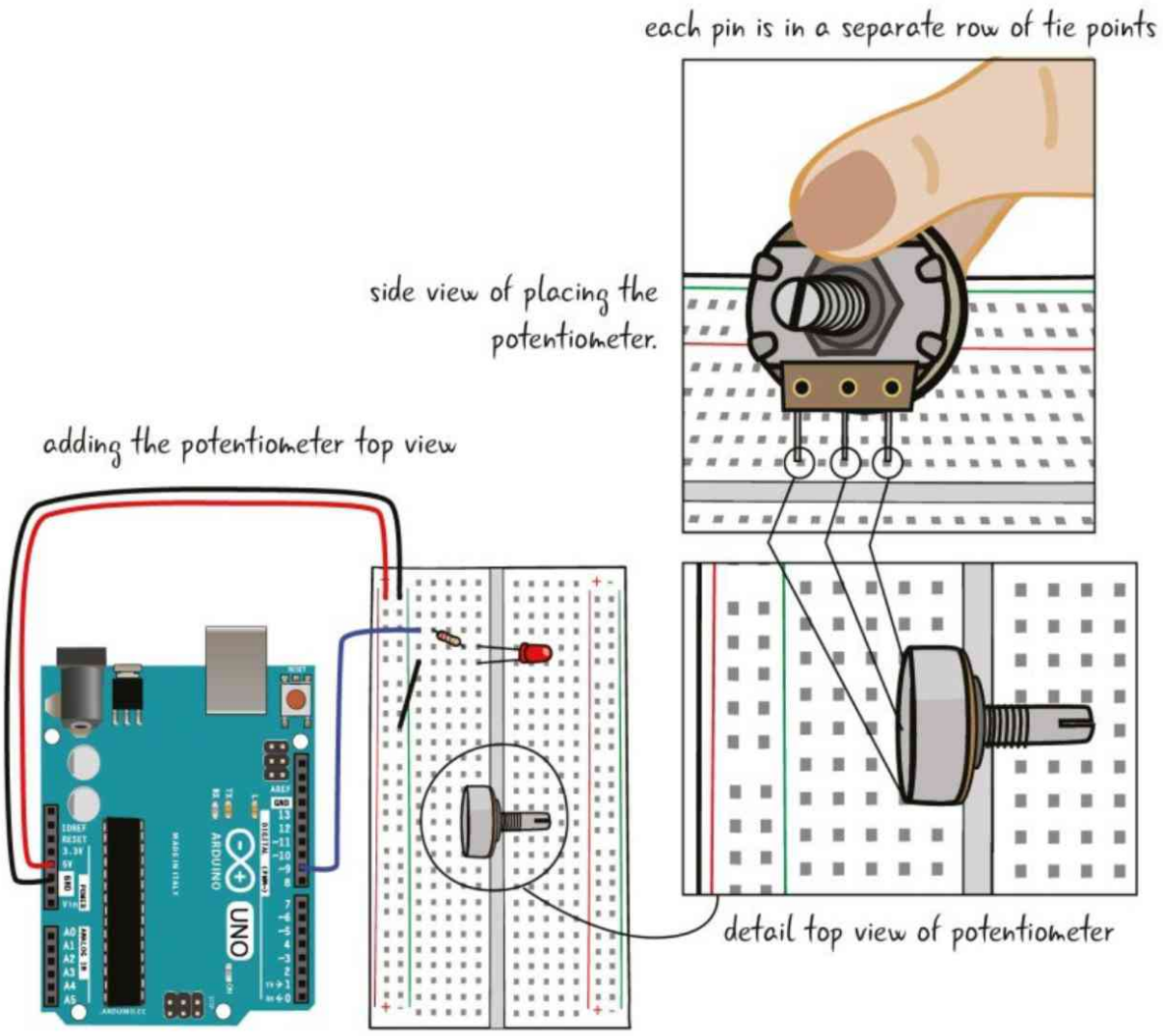


FIGURA 7.7 Montaje del potenciómetro.

A continuación conectarás el potenciómetro a los buses de alimentación y tierra.

Conecta el pin situado en la parte superior del potenciómetro al bus de tierra con un puente. A continuación, conecta el pin situado en el otro extremo del potenciómetro al bus de alimentación (figura 7.8). Debes asegurarte de que los puentes y los pines del potenciómetro estén en la misma fila de puntos de conexión.

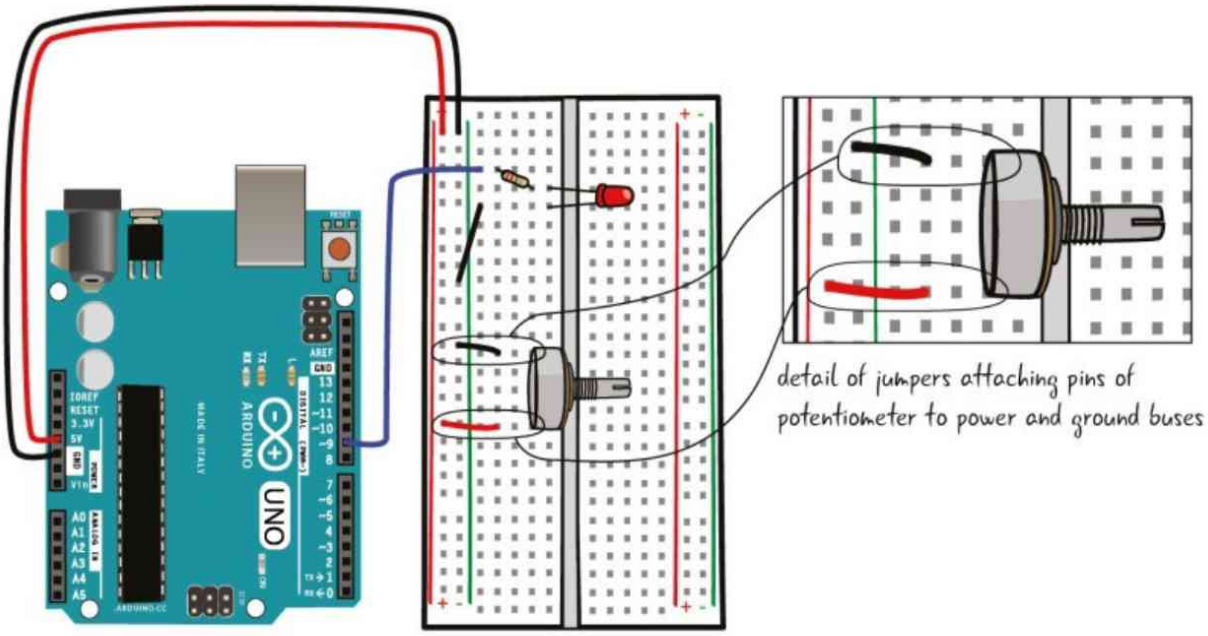


FIGURA 7.8 Adición de puentes al potenciómetro.

Finalmente, conecta el terminal central del potenciómetro a la entrada analógica pin A0 con un puente (figura 7.9).

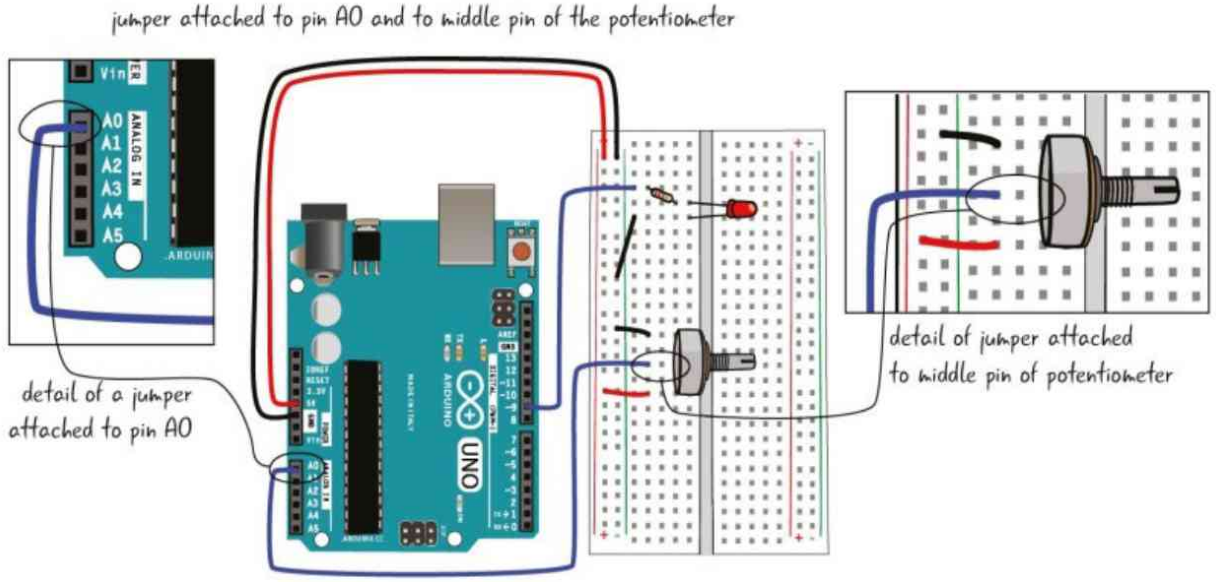


FIGURA 7.9 Conexión del potenciómetro al pin analógico.

atenuación de la luMINOSIDAD

En este punto, conecta el Arduino al ordenador con el cable USB. Vas a cargar un sketch de ejemplo del IDE de Arduino. Para cargar el sketch,

selecciona "Archivo > Ejemplos > 03.Analog" y selecciona "AnalogInOutSerial". Guarda este sketch como **LEA7_AnalogInOutSerial**.

Una vez guardado, haz clic en Verificar, y después en Subir.

Cuando gires el potenciómetro, el led debería lucir más o menos, en función de la dirección en la que lo gires (figura 7.10).

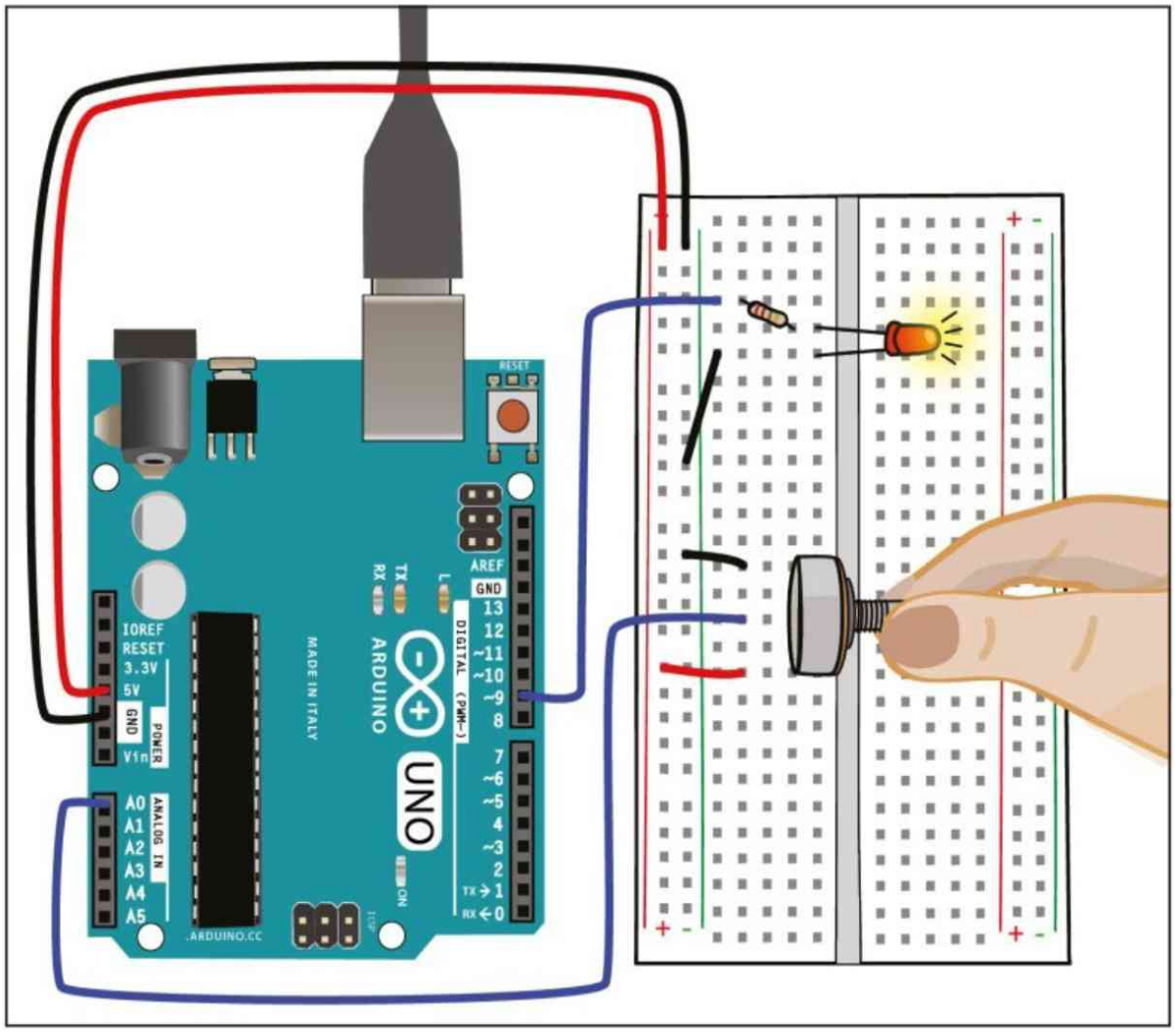


FIGURA 7.10 El led brilla más o menos cuando giras el potenciómetro.

PIENSA SOBRE ELLO...

Probablemente utilizas potenciómetros todos los días para controlar el volumen de un equipo estéreo o un regulador de intensidad de luz. ¿Se te ocurren otros dispositivos que podrías tratar de controlar con un potenciómetro?

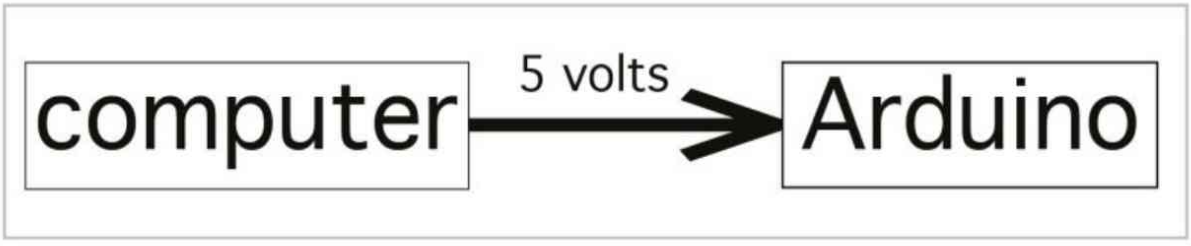
A continuación verás cómo el sketch permite al circuito interpretar el valor de la resistencia del potenciómetro y cambiar el brillo del led en función de esos valores.

¿cuál es el papel del sketch en el Circuito?

Has visto el circuito funcionando, así que entiendes lo que hace, pero ¿cómo traduce el Arduino el valor de la resistencia del potenciómetro a un valor de luminosidad del led? Para averiguarlo, veamos cómo la electricidad y la información fluyen a través del circuito.

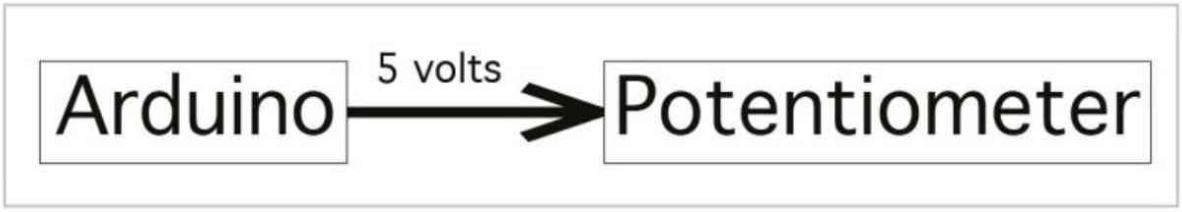
Paso 1: alimentación del Arduino

Al Arduino le llegan cinco voltios de alimentación del ordenador a través del cable USB.



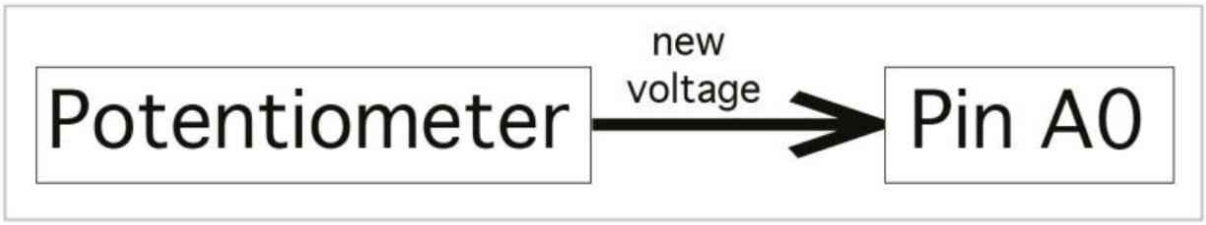
Paso 2: alimentación del potenciómetro

Se envían cinco voltios a un terminal del potenciómetro desde el pin de 5 voltios del Arduino (a través del bus de alimentación).



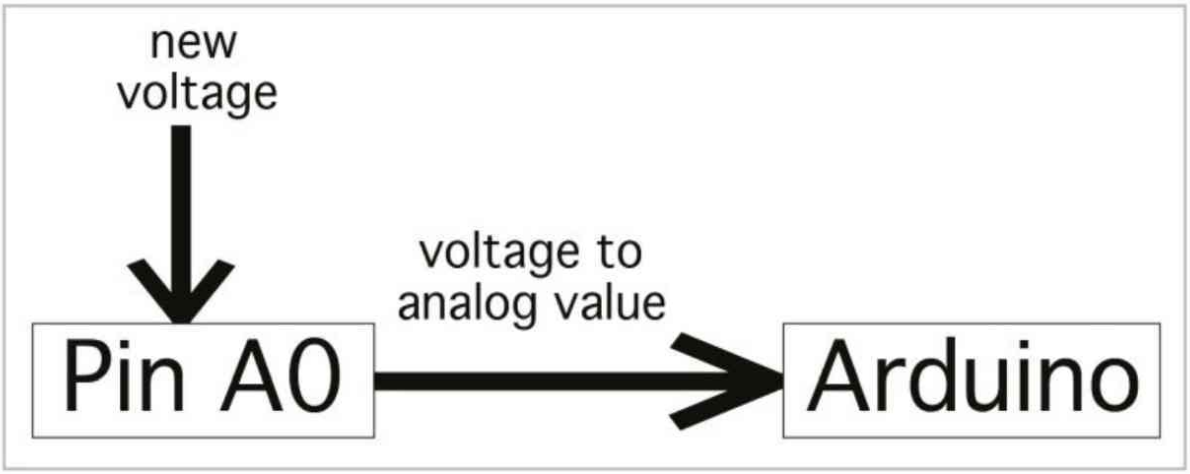
Paso 3: el potenciómetro cambia el voltaje

El potenciómetro presenta una mayor resistencia, baja el voltaje, y lo envía al Arduino, al pin A0.



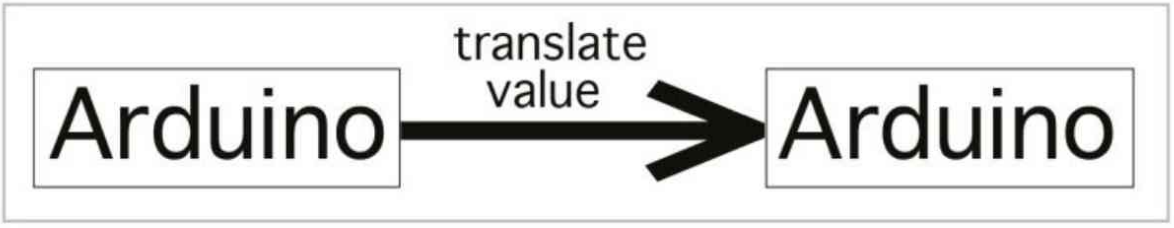
Paso 4: Arduino lee el voltaje

En el pin A0, el Arduino lee el voltaje que entra del potenciómetro y traduce su valor en un número en la escala analógica 0-1023 que mencionamos anteriormente. A veces tarda en hacer esta lectura. Más adelante, en este capítulo, hablaremos de cómo el Arduino determina este valor.



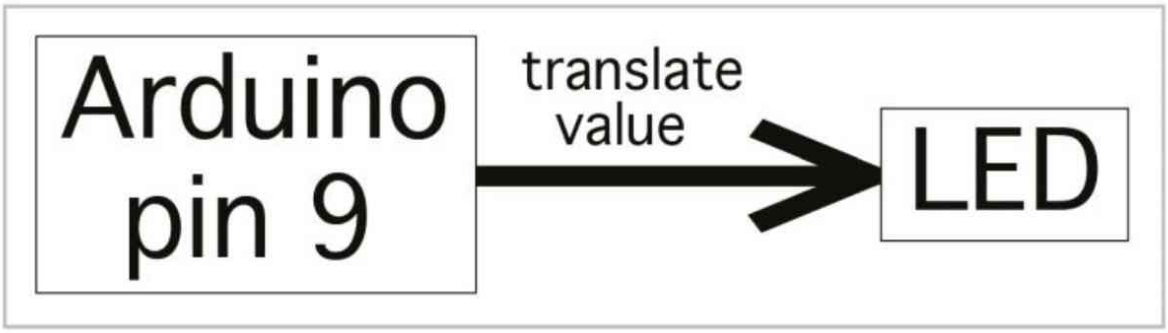
Paso 5: Arduino convierte los valores

El Arduino cambia el valor analógico del potenciómetro a un valor analógico traducido utilizando la función map(), que explicaremos más adelante en este capítulo. Este paso es crucial, ya que el led no entiende los valores entre 0 y 1023, pero acepta valores entre 0 y 255.

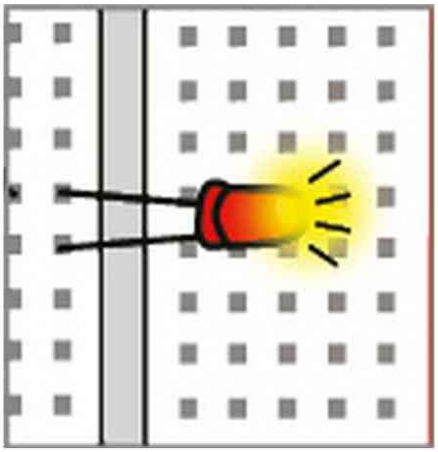


Paso 6: Arduino envía el valor al led

El Arduino envía este valor analógico traducido al led a través del pin 9 usando PWM. Más adelante en este capítulo explicaremos qué es PWM y cómo funciona.



Paso 7: el led se enciende



El led se ilumina; la mayor o menor luminosidad la determinará el valor analógico que reciba.

Como puedes ver, el sketch realiza el importante paso de traducir la información del potenciómetro en un valor que se utiliza para controlar el brillo del led.

Ahora que has estudiado una visión general de lo que está pasando en el circuito y cómo este interactúa con el sketch, es hora de sumergirse en los detalles del sketch.

el sketch LEA7_AnalogInOutSerial

Este sketch lee el valor del voltaje en el pin A0, lo traduce a un valor que el led puede entender y luego lo envía al pin 9. Como en otros sketches que has visto, hay una sección de inicialización, una función setup() y una función loop(). Hemos omitido de nuevo los comentarios al principio del sketch.

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

initialization

setup()

loop()

inicialización de LEA7_AnalogInOutSerial

La sección de inicialización declara y establece un valor inicial para algunas variables que necesitarás en el sketch. Como viste en el capítulo anterior, cuando declaras una variable, le das un nombre, que indica el tipo de información que contendrá, dándole un valor, y en algunos casos incorpora un

calificador que indica si se trata de una constante.

```
const int analogInPin = A0; //Analog input pin attached to potentiometer
const int analogOutPin = 9; //Analog output pin attached to LED

int sensorValue = 0;        //value read from the pot
int outputValue = 0;        //value output to the PWM (analog out)
```

Como puedes ver en estas líneas de código, nuestro sketch contiene cuatro variables. Aquí están los detalles de lo que hace cada una:

analogInPin

Contiene el número de pin del que se extrae la lectura del potenciómetro. Está asociada al pin A0 (figura 7.11).

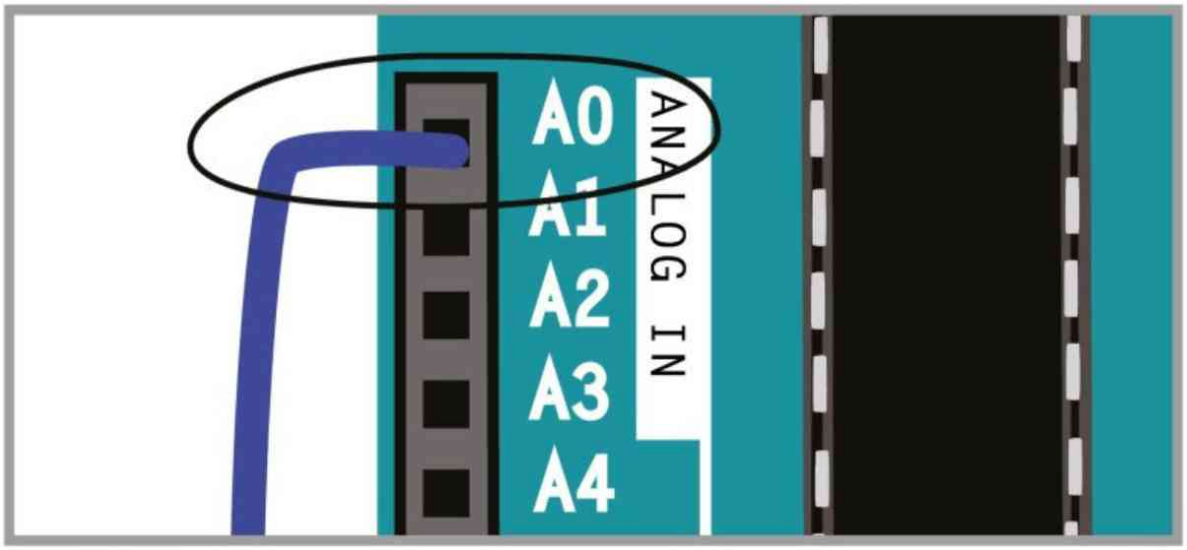


FIGURA 7.11 analogInPin asociada al pin A0.

analogOutPin

Contiene el número de pin que está conectado al led. Está asociada al pin 9 (figura 7.12).

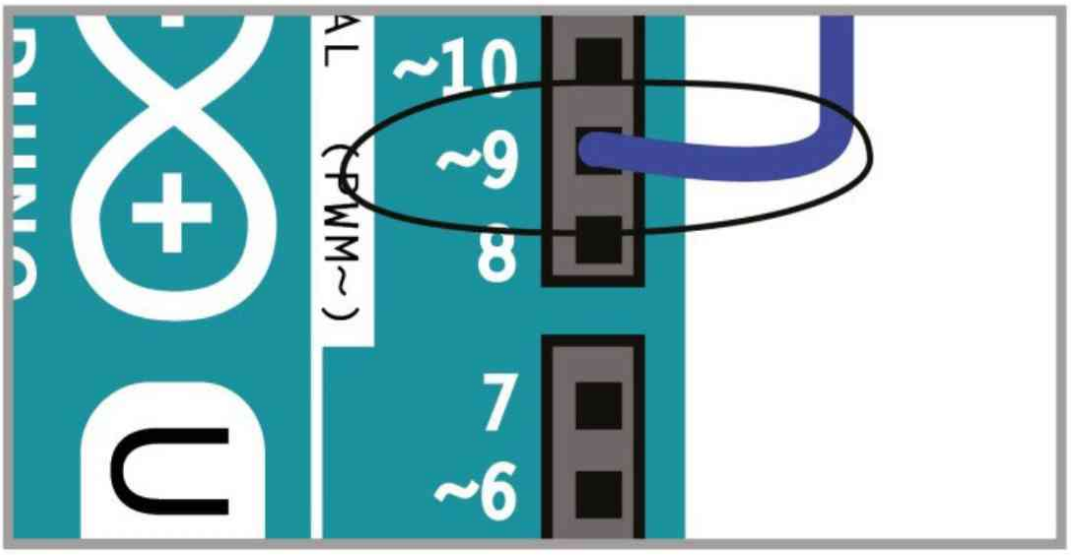


FIGURA 7.12 `analogOutPin` asociada al pin 9.

sensorValue

Esta variable está inicialmente a 0; contendrá el valor del voltaje procedente del potenciómetro (figura 7.13).

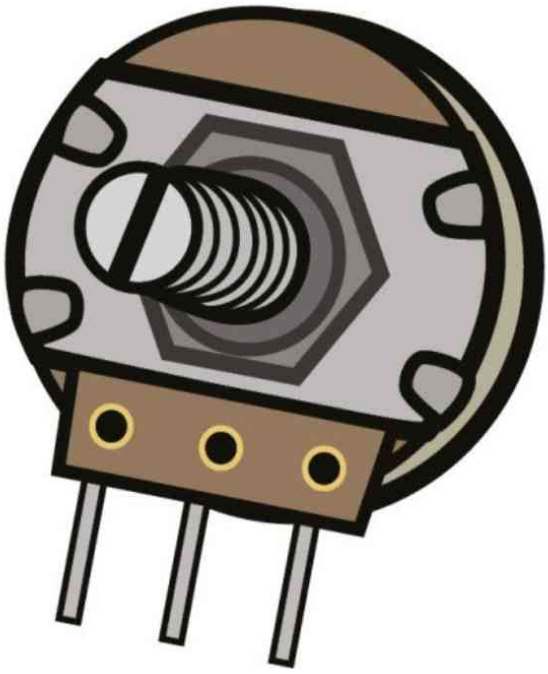


FIGURA 7.13 `sensorValue` contendrá el cambio del nivel de voltaje en el Pin A0 procedente del potenciómetro.

outputValue

Inicialmente está a 0; contendrá el valor que el Arduino enviará al led, que determina su brillo (figura 7.14).

FIGURA 7.14 `outputValue` **contendrá el valor que el Arduino enviará al pin 9 para controlar el brillo del led.**

La sección de inicialización crea estas variables para que puedas utilizarlas más tarde en la sección loop(). El siguiente paso: la sección setup().

setup() de LEA7_AnalogInOutSerial

La sección setup() para el sketch tiene una sola línea; se trata de una nueva función de Arduino de la que no hemos hablado: Serial.begin(). Esta función utiliza un objeto serie.

El objeto serie es un conjunto de funciones y variables que permiten al Arduino comunicarse con otros dispositivos. En este sketch, lo utilizarás para comunicarte con el ordenador. begin() es una función del objeto serie.

> Note **Una *función* es una forma de organizar código o bloques de instrucciones para el ordenador.**

Hablamos de funciones en el Capítulo 3 cuando nos referimos a setup() y a loop(). Así es cómo la función begin() aparece en nuestro código de setup():

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
```

Esta línea de código le dice al Arduino que abra una línea de comunicación con el ordenador (se comunicarán a través del cable USB que los conecta). También establece una velocidad de comunicación para que el Arduino y el ordenador se comuniquen: 9 600 baudios por segundo (bps). La velocidad exacta de baudios no es importante en este punto, siempre y cuando el Arduino y el ordenador tengan una velocidad de comunicación compartida.

Veremos la comunicación serie más de cerca unas páginas más adelante; por ahora, pasemos a la sección loop() del código.

Note `begin()` **es una función del objeto serie que configura la comunicación entre dispositivos.**

código loop() de LEA7_AnalogInOutSerial

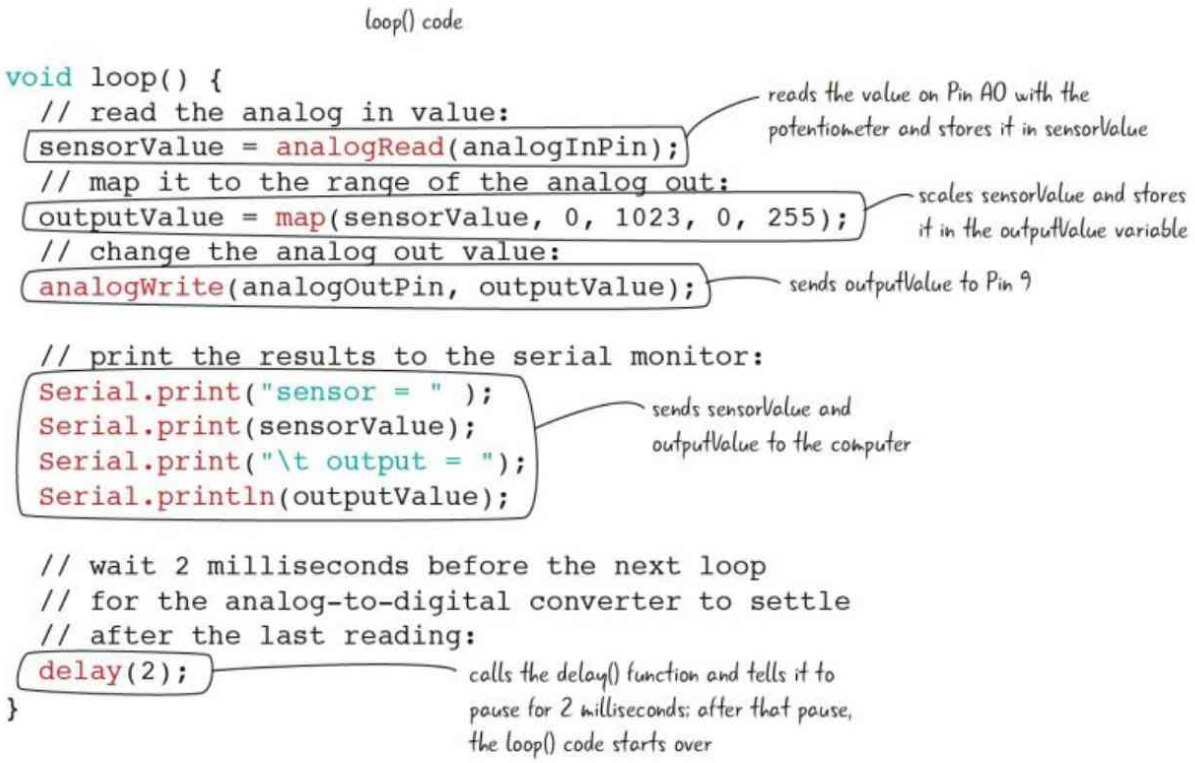
Este es un resumen de lo que hace el código loop():

1. Toma una lectura analógica del pin al que está conectado el potenciómetro y la almacena en una variable.
2. Traduce ese valor en algo que el led puede entender (un valor dentro de la escala 0–255).
3. Escribe el valor adaptado al led (pin 9).
4. Envía los valores (sensorValue y outputValue) al ordenador, para que pueda ver cómo cambian con el tiempo.
5. Espera un instante (2 milisegundos) antes de la siguiente lectura.

Estos pasos siguen el orden indicado y de forma repetitiva indefinidamente

hasta que desconectes el equipo Arduino.

Veamos el código otra vez:



entrada Analógica: Valores del Potenciómetro

La primera línea de código hace que el Arduino compruebe el valor del voltaje que procede el potenciómetro, en el pin A0. Este valor se almacena en sensorValue.

```
read value from the potentiometer attached to pin A0
// read the analog in value:
sensorValue = analogRead(analogInPin);
```

¿Cómo afecta la resistencia variable del potenciómetro a los valores que salen del pin A0?

Si utilizaras un multímetro para leer el voltaje de salida del pin A0 cuando el potenciómetro está completamente hacia un lado, dando la máxima resistencia, leerías una tensión de 0 voltios. Si giraras el potenciómetro completamente hacia el otro lado, sin presentar resistencia, leerías un voltaje de 5 voltios. ¿Recuerdas la ley de Ohm? Aquí la vemos en acción, al producirse una variación de la resistencia que afecta al valor del voltaje.

Los valores que se obtienen utilizando los pines analógicos son una *medida escalada del voltaje*. El Arduino convierte los valores de voltajes entre 0 V y 5 V en un número entre 0 y 1.023. A este proceso se le llama *conversión de analógico a digital*.

La figura 7.15 ilustra una regla que demuestra la lectura de la conversión de valores analógicos a digitales. En la parte superior se ve el voltaje, que va de 0 a 5 voltios; en la parte inferior ves el rango, que se puede obtener del pin de una entrada analógica de Arduino, 0–1.023.

Note Los pines de entradas analógicas leen los niveles de 0V a 5V y los convierten a un rango de valores de 0 a 1.023.

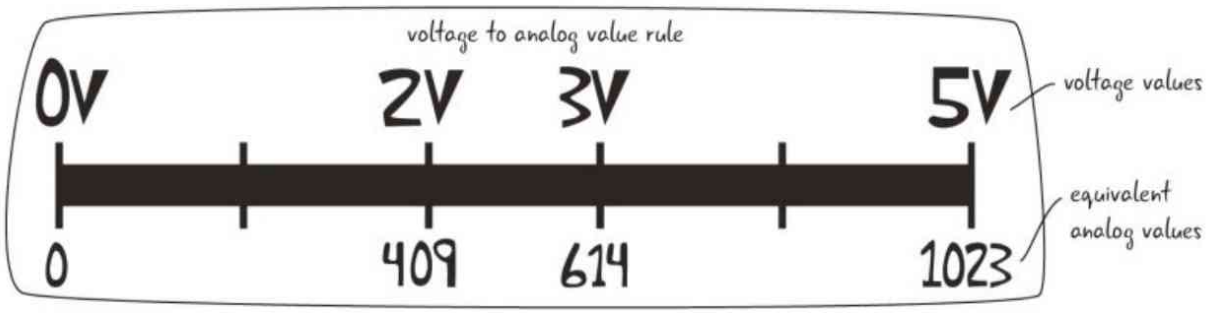


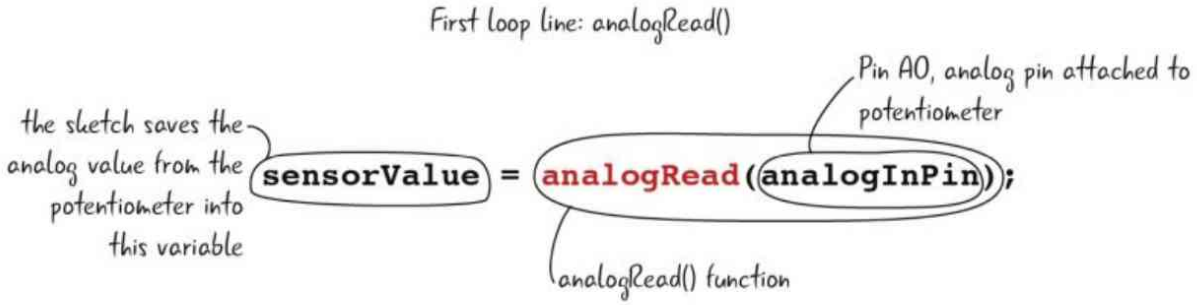
FIGURA 7.15 Conversión del voltaje a una lectura analógica.

En la regla de la figura 7.15, puedes ver que a 0 voltios obtienes una lectura de 0, y a 5 voltios obtienes una lectura de 1.023. ¿Qué sucede en los valores entre 0 y 5 voltios? El valor de la entrada analógica es un número entre 0 y 1.023, por lo que a 2 voltios obtienes un valor de 409. A 3 voltios obtienes un valor de 614. Estos valores los calcula automáticamente el sketch. Más adelante, en este capítulo, verás cómo funciona esto con nuestros proyectos.

¿Por qué tienes que convertir el valor del voltaje? Verás más adelante en este capítulo, y en el capítulo 8, "Servomotores", cómo usamos el valor (entre 0 y 1.023) con algunas otras funciones.

la entrada Analógica en el Código: analogRead()

El sketch toma primero una lectura analógica del pin al que está conectado el potenciómetro, el pin A0, y almacena el valor de la lectura en una variable llamada sensorValue.



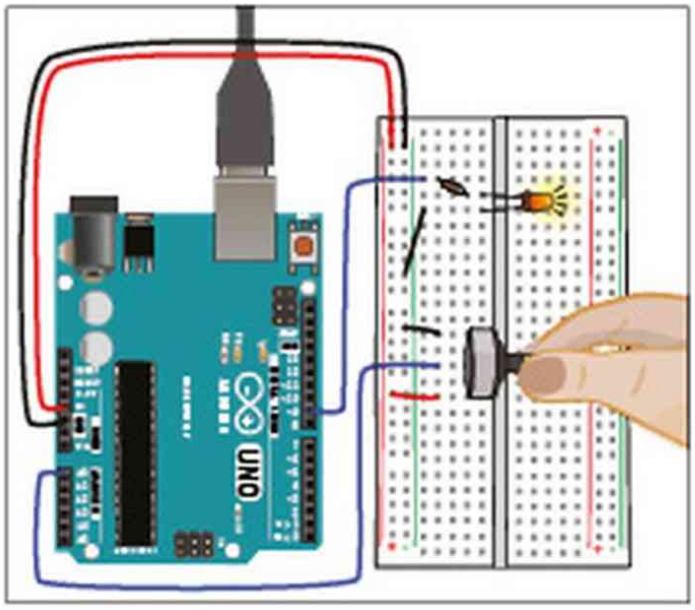


Figura 7.16:**Al girar el potenciómetro el led aumenta o disminuye su brillo.**

Recuerda que analógico significa que podemos tener valores distintos de 0 o 1. analogRead() lee un valor de un pin de entrada analógica que puede variar entre 0 y 1.023, como ya sabes. Si giras el potenciómetro completamente hacia un lado, no hay resistencia, por lo que el valor máximo es 1.023. Cuando lo giras hasta el otro lado, la resistencia es máxima, por lo que el valor se reduce a 0. El proceso de lectura del valor que aparece en el pin necesita algún tiempo.

Volvamos a la regla de valores analógicos y echémosle un vistazo de nuevo (figura 7.17).

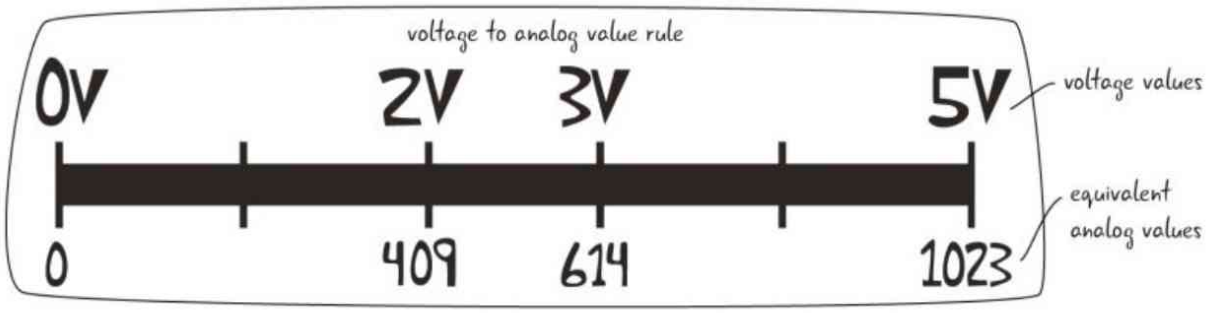


FIGURA 7.17 Conversión a voltaje analógico.

Si el potenciómetro envía 2 voltios al pin A0, entonces el valor leído en ese pin es 409. Si en lugar de ello, giras el potenciómetro para que haya 3 voltios en el pin, entonces el nuevo valor que leerás será 614.

Ahora que comprendes mejor analogRead() veamos la siguiente línea de código en el sketch.

Ajuste de Valores: map()

El siguiente paso es ajustar el valor del sensorValue y almacenarlo en una nueva variable. En la siguiente línea de la función loop(), verás la segunda variable outputValue, a la que se le asigna el valor de una función map(). La función map() escala automáticamente un valor de sensor y lo convierte en otro rango de valores.

loop() code line two: map() function

```
// map it to the range of the analog out:
outputValue = map(sensorValue, 0, 1023, 0, 255);
```

¿Por qué mapeas el valor? ¿Por qué no puedes usar los números que se obtienen al leer el pin A0? Cuando observamos el sketch anterior, viste que usamos analogWrite() para escribir valores en el pin 9. Esta función toma un rango de valores de 0 a 255. Como los valores que proceden del pin de entrada analógica pueden variar de 0 a 1 023, tienes que convertir ese rango,

0-1 023, a 0-255 para pasar el valor al pin 9 (figura 7.18).

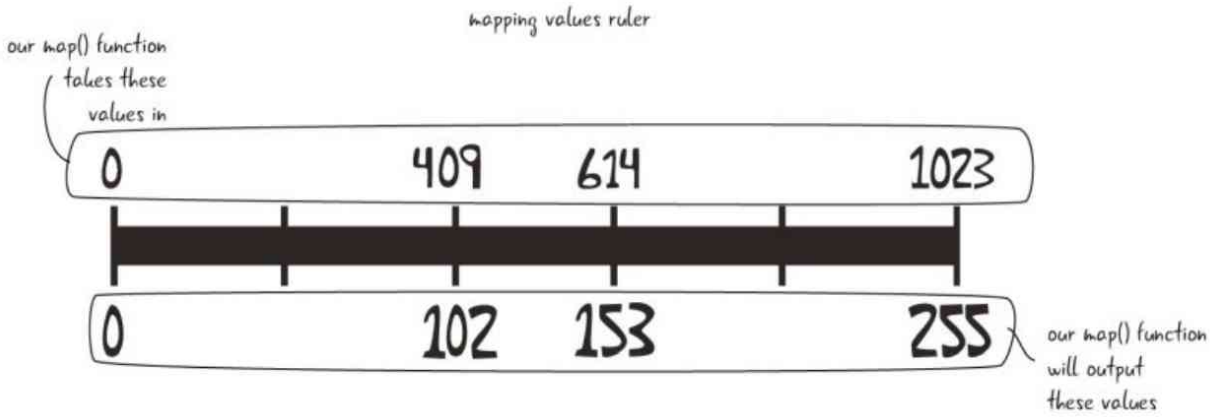
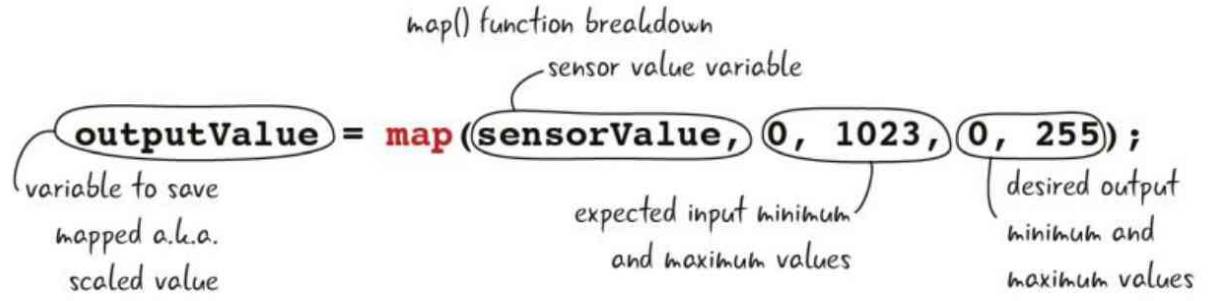


FIGURA 7.18 Conversión del rango de 0–1.023 a 0–255

La función map() te pregunta qué variable quieres escalar, cuáles son los valores máximo y mínimo asignados para la variable del sensor y cuáles deberían ser sus valores máximo y mínimo.



El valor guardado en outputValue es un número reducido. Si el sensor lee 1.023, map() ajustará la variable outputValue a 255. Al dar la función map() a estos rangos, casi todos los valores guardados en outputValue serán menores que la lectura original. La única excepción es si lees 0 en el sensorValue, que se guarda en la outputValue como 0.

> **Note** La función `map()` se utiliza para escalar valores de un rango a otro.

escribir UN VALOR en UN PIN: analogWrite()

Ahora que has asignado los valores, estás listo para enviar un valor al led y encenderlo. El siguiente paso que realiza el sketch es escribir el valor ajustado en el pin del led.

Usarás la función analogWrite() para enviar valores analógicos a algunos pines del Arduino. Hablaremos de estos pines especiales muy pronto, pero primero echemos un vistazo al código analogWrite() del sketch.

loop() code line three: analogWrite()

```
// change the analog out value:
 analogWrite(analogOutPin, outputValue);
```

analogWrite() es similar a la función digitalWrite() que ya hemos discutido. analogWrite() necesita saber dos cosas: el pin en el que quieres escribir y el valor que deseas escribir en ese pin. En este sketch, envías analogOutPin, al que has configurado como pin 9, un valor procedente de la variable outputValue, que ajustaste con la función map(). Al enviar un valor analógico entre 0 y 255, el Arduino envía una tensión entre 0 V y 5 V al led. Veamos cómo el rango de 0 a 255 asigna el nivel de voltaje en el pin (figura 7.19).

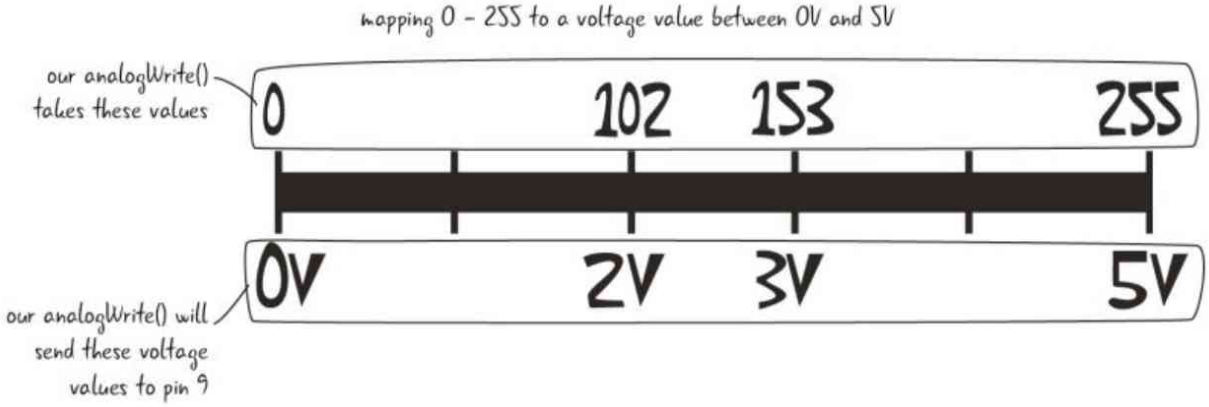


FIGURA 7.19 Asignación de 0–255 a un valor de voltaje.

> **Note** `analogWrite()` **toma un valor entre 0 y 255 y entrega al pin un valor entre 0 y 5 voltios.**

El Arduino es capaz de enviar un valor analógico mediante un proceso llamado PWM. Analizaremos cómo funciona un poco más adelante.

¿En qué se diferencian las funciones analógicas y digitales?

Antes de profundizar en analogWrite() y en cómo funciona con PWM, analizaremos cómo analogRead() y analogWrite() se comparan con las funciones digitalRead() y digitalWrite() que hemos utilizado en los capítulos anteriores. Hemos estado haciendo la comparación todo el tiempo, pero la tabla 7.1 muestra la comparación de estas cuatro funciones.

Tabla 7.1: **Comparación de funciones analógicas y digitales**

Nombre de la función	qué hace	Argumentos que requiere	Rango de valores
digitalRead()	Lee el valor en un pin de entrada digital	El número del pin al que se le asigna la lectura	Lee del pin 1 o 0
digitalWrite()	Escribe el valor	El número del pin	Escribe en el

	en un pin de salida digital	en el que se escribe un valor y el valor que se escribe	pin 100
analogRead()	Lee el valor en un pin de entrada analógica	El número del pin al que se le asigna la lectura	Lee del pin un número entero entre 0 y 1.023
analogWrite()	Escribe el valor en un pin de salida con PWM	El número del pin en el que se está escribiendo un valor y el valor que se está escribiendo	Escribe en el pin un número entero entre 0 y 255, que se traduce en un valor de voltaje entre 0 y 5 voltios

A TENER EN CUENTA

¿Puedes pensar en un circuito que tal vez quieras montar utilizando información analógica? ¿En qué se diferenciará de un circuito que utiliza información digital?

VALORES ANALÓGICOS de SALIDA: PWM

Como has visto en capítulos anteriores, el Arduino es capaz de proporcionar valores de voltaje diferentes: bien 5 V o 3,3 V. Todos los pines de E/S en el Arduino se fijan a 5 V de salida cuando se utilizan para controlar los componentes del circuito. Si el Arduino proporciona solamente 5 V de los pines de salida, ¿cómo puedes crear valores analógicos? El Arduino tiene la capacidad incorporada de utilizar una técnica llamada *modulación por ancho de pulso*, o PWM.

Entonces, ¿cómo funciona PWM? Imagina que enciendes las luces en tu habitación y luego las apagas. La habitación se ilumina durante un instante y luego vuelve a estar a oscuras. Si continuas accionando el interruptor hacia arriba y hacia abajo a una velocidad lenta, la habitación se ilumina, luego estás a oscuras, y así una y otra vez (figura 7.20).

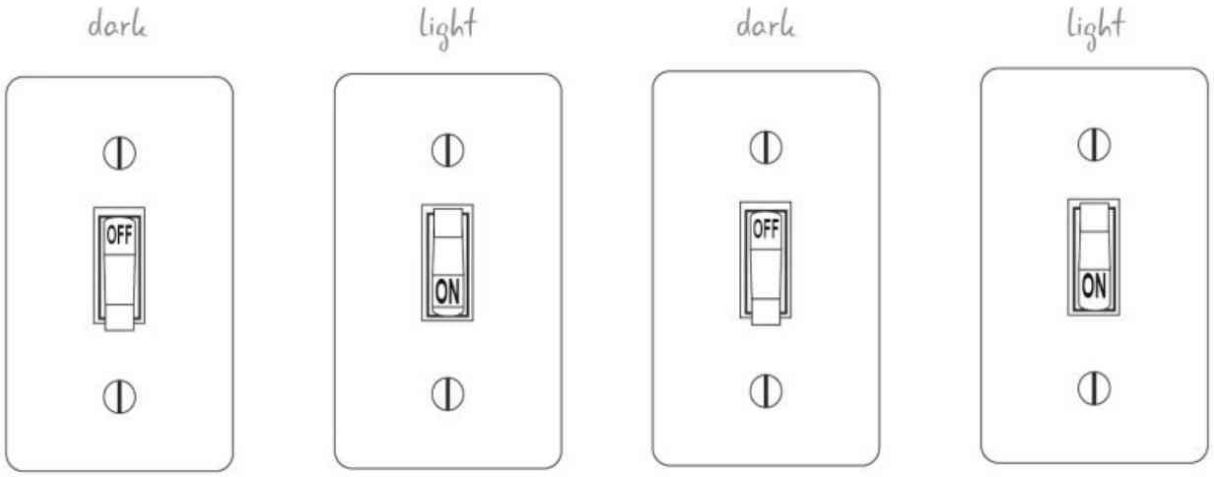


FIGURA 7.20 Apagado y encendido del interruptor.

Pero algo extraño sucede cuando mueves el interruptor de la luz cada vez más rápido. En vez de pasar de estar iluminada a estar a oscuras, la luz en la habitación tendrá una intensidad media. De hecho, la habitación también se verá más iluminada si dejas la luz encendida durante un poco más de tiempo del que la dejas apagada. Así que la habitación tendrá una intensidad de luz o iluminación media que depende del porcentaje de tiempo en el que la luz

está encendida en relación con el porcentaje de tiempo en el que está apagada. Es decir, el nivel de iluminación que es el *brillo medio*, depende del porcentaje de tiempo que la luz está encendida en relación con el porcentaje de tiempo que está apagada.

PWM utiliza una técnica similar a la de encender el interruptor de luz para crear un nivel de iluminación más brillante o más tenue. Cuando se utiliza PWM en el Arduino, el nivel de voltaje del pin PWM se activa y desactiva a diferentes velocidades a intervalos regulares. Unas veces es de 0 voltios y otras veces, de 5 voltios.

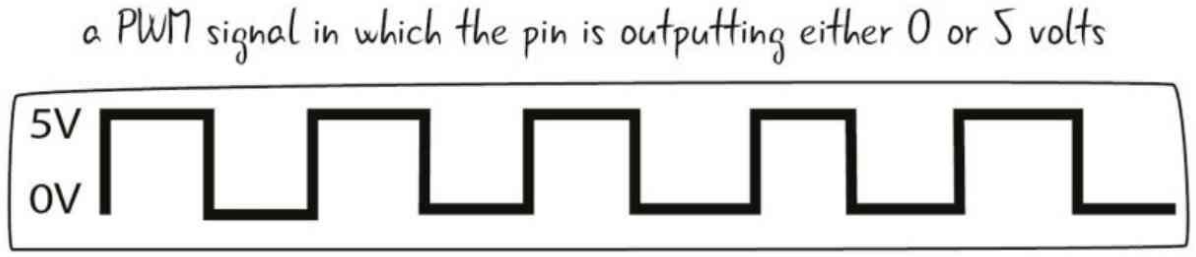


FIGURA 7.21 Señal PWM.

> **Note** PWM crea un valor promedio al activar y desactivar el pin muy rápido.

Al variar la cantidad de tiempo en la que el pin se activa y se desactiva, el Arduino crea un valor de voltaje promedio entre 0 y 5 voltios.

¿dónde están los pines pWM?

Entonces, ¿qué pines puedes usar con PWM? Se pueden utilizar con PWM algunos de los pines digitales situados a la derecha de la placa de Arduino: los pines 3, 5, 6, 9, 10 y 11. Como puedes observar, cada uno de los pines está etiquetado con el símbolo ~ (figura 7.22).

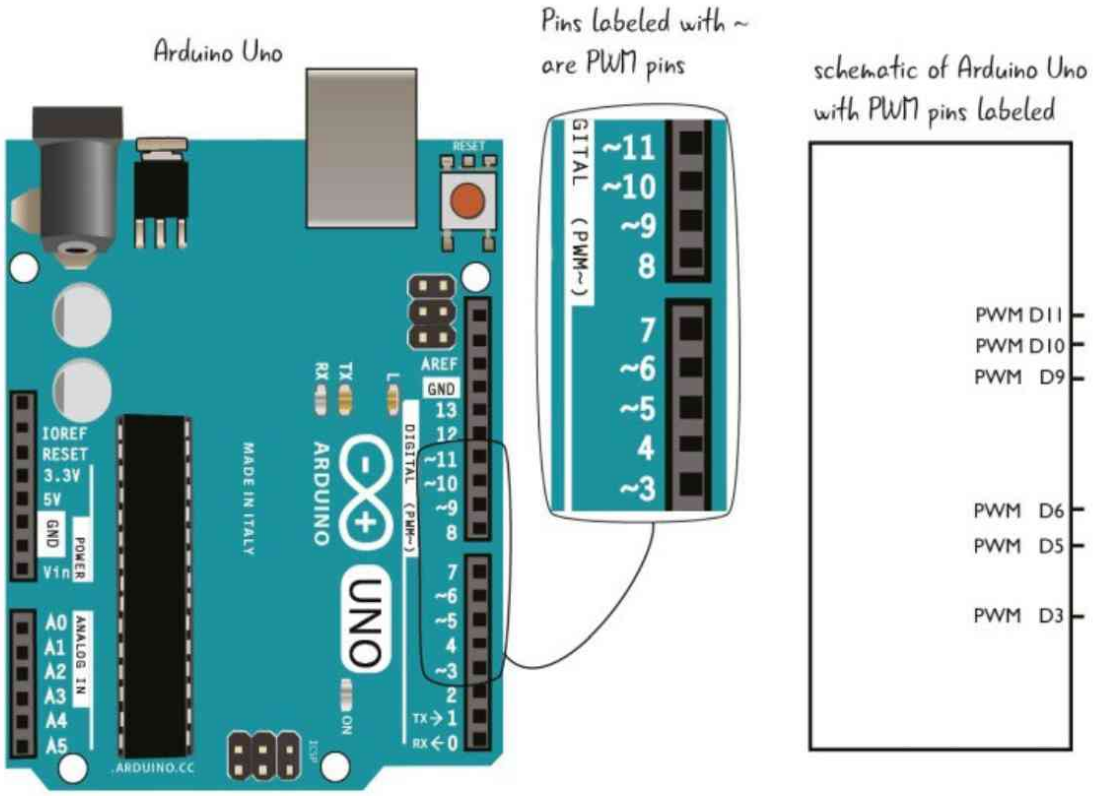


FIGURA 7.22 Pines PWM etiquetados en el Arduino.

¿PREGUNTAS?

P: ¿Son PWM y `analogWrite()` la misma cosa?

R: No, PWM y `analogWrite()` están relacionados, pero no son la misma cosa. `analogWrite()` es una función de Arduino que le dice al Arduino que utilice un pin para crear valores analógicos. Esta función utiliza la técnica PWM para crear valores analógicos.

P: PWM activa y desactiva los pines, ¿con eso se consigue, de alguna manera, un valor diferente?

R: Correcto. Dado que el Arduino activa y desactiva el pin de forma muy rápida, el valor efectivo del voltaje que sale del pin está relacionado con el tiempo medio que el pin está en `HIGH`. Has de tener en cuenta que el Arduino no produce un valor de voltaje distinto del digital, sino que utiliza este truco de valores promedios para crear un valor analógico.

P: ¿Se pueden usar los pines PWM como pines digitales?

R: Sí. En función del circuito y del sketch, puedes utilizar los pines 3, 5, 6, 9, 10 y 11 bien como pines digitales, bien como pines PWM de salida.

Comunicación serie

Ya has visto cómo funciona el sketch con el Arduino en lo que se refiere a recoger información del potenciómetro, cambiarla y enviarla al pin que controla el led. A continuación veremos cómo y por qué escribes los valores en el ordenador mediante la comunicación serie. El penúltimo paso que realiza el sketch es escribir dos valores en tu ordenador (el valor de un pin de entrada analógica y el valor que envias al pin del led) para que puedas ver cómo cambian con el tiempo.

¿por qué necesitas ver la entrada y salida de información del Arduino en el ordenador?

A veces es útil tener información sobre cómo funciona el sketch. Este conocimiento puede ser útil si estás intentando depurar el circuito. Por ejemplo, puedes ver los valores que tiene en sus pines de entrada y salida. En breve te mostraremos cómo hacerlo.

¿qué signIfica el término serie?

Serie, en este contexto, es un tipo de protocolo de comunicaciones. Se refiere a la forma en la que dos dispositivos pueden comunicarse enviando información a través de un par de cables. Al cambiar el valor del voltaje en un cable de HIGH a LOW, el Arduino puede transmitir información a través del cable USB al ordenador (figura 7.23). ¿Recuerdas los pines de salida digital marcados TX y RX en el Arduino (pines digitales 1 y 0)? Estos pines se utilizan para comunicarse con el ordenador. TX envía; RX recibe.

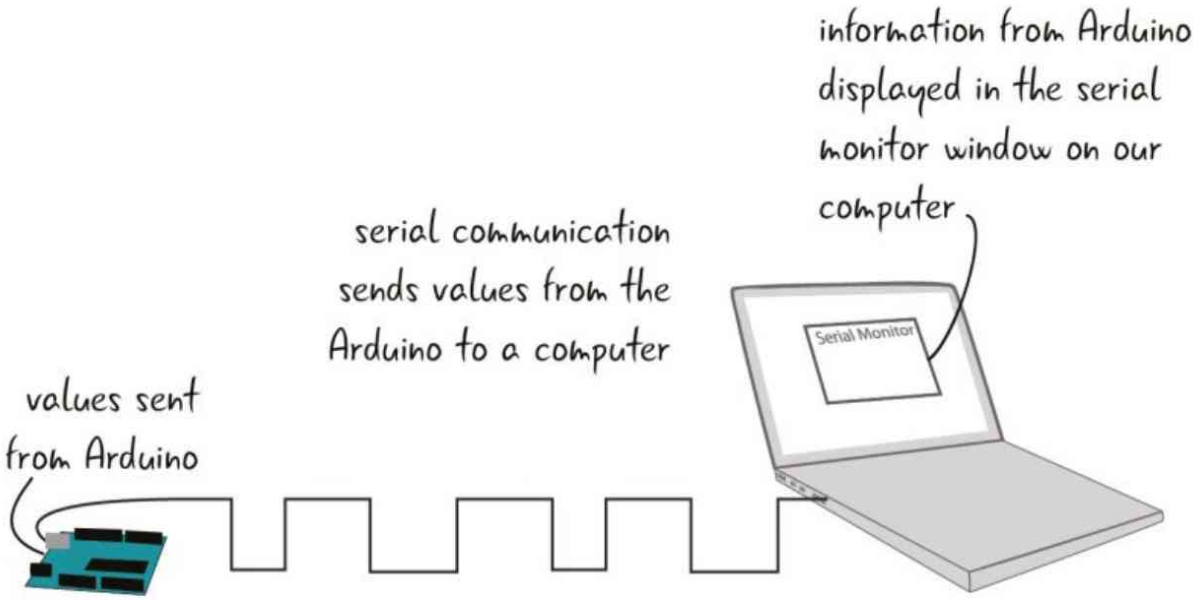


FIGURA 7.23 La comunicación serie permite al Arduino hablar con el ordenador.

La comunicación serie es un método fácil y eficiente para que el Arduino se comunique con el ordenador. El IDE de Arduino contiene una ventana llamada *monitor serie*, que muestra la información que recibe del Arduino, como son los valores que detectan sus sensores o qué función se está ejecutando en ese momento.

Echaremos un vistazo al monitor serie antes de volver al código.

Uso del monitor serie

El monitor serie es una característica del IDE de Arduino que te muestra la información enviada desde el Arduino. Es útil para depurar y saber qué valores produce un sensor o una resistencia variable. Para abrir el monitor serie, haz clic en el botón en la parte superior del IDE de Arduino (figura 7.24).

FIGURA 7.24 Botón del monitor serie.

Cuando abres el monitor serie, ves una ventana que muestra las respuestas del Arduino y un menú desplegable que controla la velocidad de la comunicación, en baudios, entre el ordenador y el Arduino. Como hemos mencionado antes, la tasa en baudios es la velocidad de la comunicación que el ordenador y el Arduino utilizan para comunicarse entre sí. De forma predeterminada, la velocidad en baudios de nuestro monitor serie se ajustará a 9 600, que coincide con el valor establecido en la función Serial.begin() en el código setup(), por lo que no deberías tener que realizar ningún ajuste.

setup()

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
```

Note **El Arduino y el ordenador deben utilizar la misma velocidad de comunicación: el valor configurado en** `Serial.begin()`.

La figura 7.25 muestra el aspecto de la ventana del monitor serie cuando se ejecuta este sketch.

FIGURA 7.25 Ejecución del sketch en el monitor serie.

Ahora que sabes dónde encontrar el monitor serie, vamos a explorar el uso del objeto serial en el código loop().

análisis del código Serie

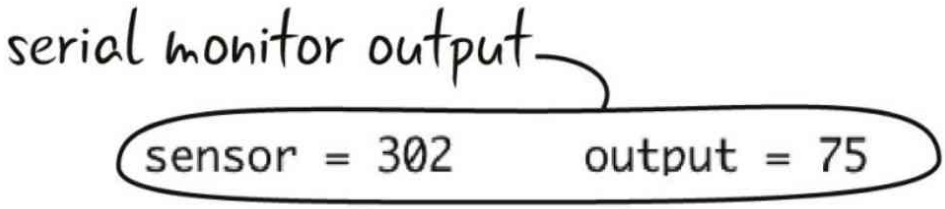
Para enviar información al ordenador, el objeto serial tiene dos funciones: Serial.print() y Serial.println(). El sketch utiliza ambas funciones para dar formato a la información en la pantalla del ordenador.

code from loop() for printing to our computer

```
Serial.print("sensor = ");
Serial.print(sensorValue);
Serial.print("\t output = ");
Serial.println(outputValue);
```

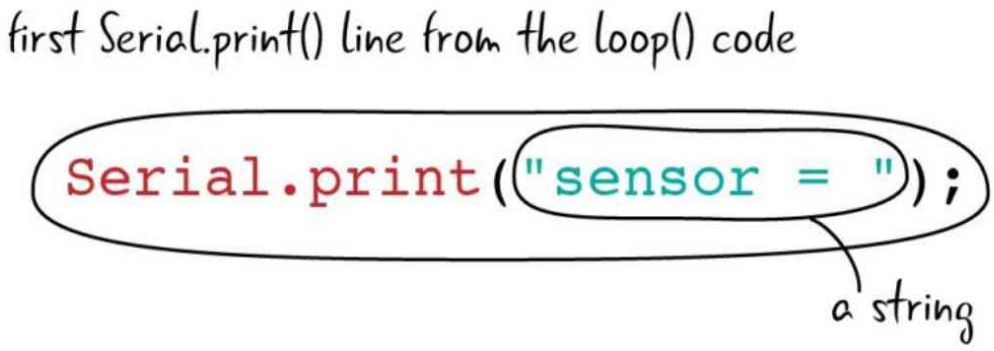
Estas cuatro líneas de código juntas presentan la única línea en el monitor serie que incluye sensor =, el valor de nuestro sensor, una pestaña, la salida de texto output =, y el valor asignado. Aquí hay un ejemplo de una línea de

salida que este código mostrará en el monitor serie:



Envío de palabras al monitor serie: cadenas

Si ves la primera línea Serial.print(), observas que hay palabras acompañadas de comillas. Para enviar palabras al monitor serie, utilizas lo que se llama una cadena o *string*.



Una cadena es una representación de texto en un lenguaje de programación. Todas las letras, números u otros caracteres alfanuméricos (incluyendo espacios y signos de puntuación) en tu código están representados por cadenas.

¿Por qué necesitas cadenas? Los ordenadores normalmente solo funcionan con valores de números. A veces necesitas usar texto en el código, para pasar información textual o para dar contexto a otros datos. Explicaremos cómo funciona esto mientras estudiamos más de cerca el código en las siguientes dos páginas.

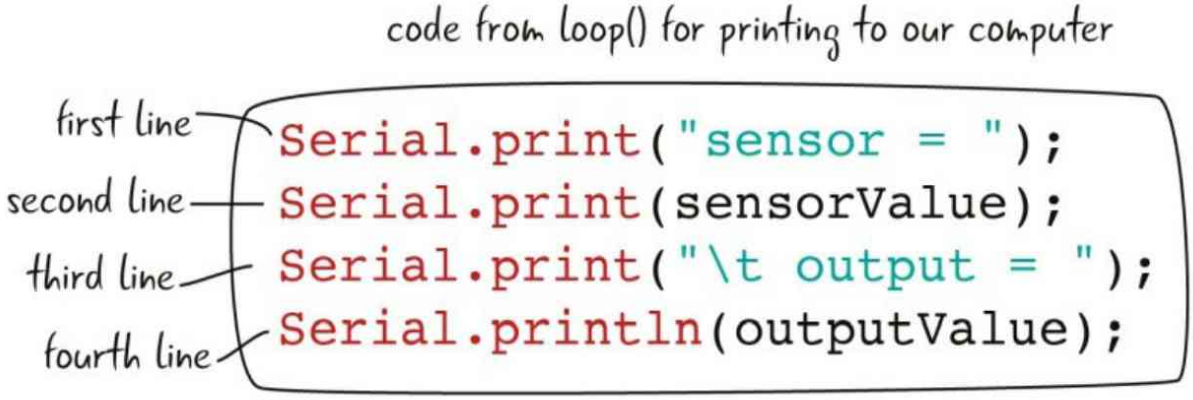
¿Cómo se utilizan las cadenas en el código? Colocas *comillas* antes y después de una cadena para identificarla. Las comillas encierran el grupo completo de caracteres, incluyendo todas las letras, espacios y puntuación.

> **Note** En el código, el texto está representado por cadenas. Todos los caracteres alfanuméricos, incluidos espacios y signos de puntuación, están representados por cadenas.

presentación en el monitor serie

Ahora veremos más de cerca cómo cada línea de código aparece en el monitor serie. Sabes que cualquier carácter entre comillas es una cadena y será representado como texto. También verás las variables referenciadas en el código. Veamos cómo colaboran con Serial.print().

> **Note** Todo lo que esté dentro de las comillas, incluidos espacios y puntuación, aparecerá en el monitor serie.



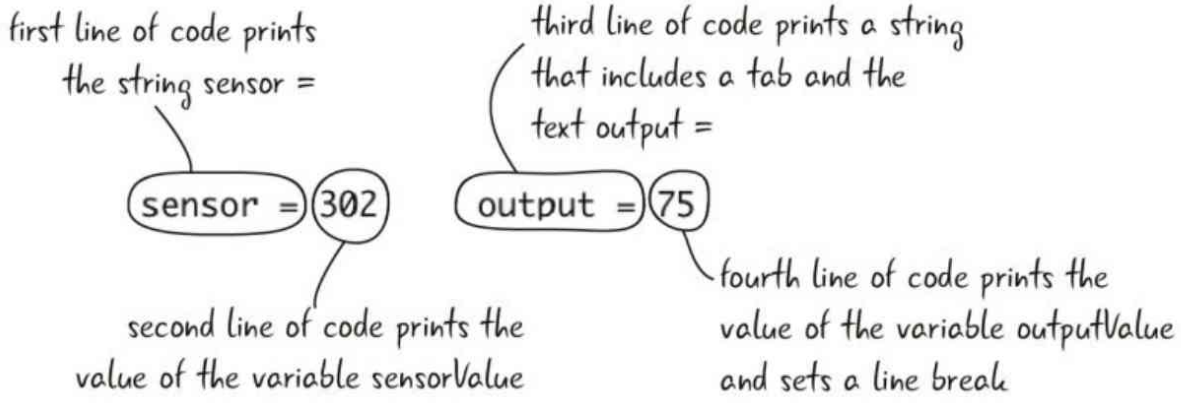
La primera línea del código presenta la cadena "sensor = " (incluyendo los espacios antes y después del signo igual).

La segunda línea de Serial.print() presentará el valor de la variable sensorValue, que es un número. Sin comillas, el Arduino presentará el valor numérico almacenado en la variable, en lugar del nombre de la variable.

La tercera línea del código serie loop() también utiliza comillas, así que sabes que se va a presentar una cadena. Sin embargo, también hay un nuevo símbolo: ¿qué significa "\t"? La \t le dice al monitor serie del Arduino que incluya una pestaña (un conjunto de espacios), en la presentación.

La cuarta línea presenta el valor de la variable outputValue. Pero en lugar de Serial.print(), utilizas Serial.println(), que producirá un salto de línea.

Recuerda que los valores de sensorValue y outputValue que ves en el monitor serie, cambiarán a medida que gires el potenciómetro.



La última línea del código serie utiliza Serial.println() en lugar de Serial.print(). Vimos que Serial.println() añade automáticamente un salto de línea; ¿cómo afecta esto a la forma en que el código aparece en el monitor serie?

final Serial.println() line from loop() code

```
Serial.println(outputValue);
```

El salto de línea significa que la siguiente vez que presentes algo en el monitor serie (incluyendo la siguiente vez mediante el código loop()), la información aparecerá en una nueva línea.

La única diferencia entre Serial.print() y Serial.println() es que el salto de línea puede facilitar la lectura de la información en el monitor serie.

> **Note** `Serial.println()` **incluye un salto de línea, que permite leer más fácilmente la información serie.**

Y como este código está en loop(), las líneas aparecerán una y otra vez. Los valores de las variables cambiarán si giras el potenciómetro.

This is how all four lines of code will appear in the serial monitor.

```
sensor = 302      output = 75
sensor = 303      output = 75
sensor = 306      output = 76
```

¿Qué significan esos valores y cómo se relacionan con las escalas de números que vimos anteriormente? sensor es el valor derivado de la lectura del voltaje (0–5) en el pin A0 y su ajuste a un rango de 0 a 1.023 con la funciónanalogRead(). output es el valor de sensor asignado a un rango de 0 a 255 por la función map() que debe utilizar el pin 9.

código de línea Final loop(): delay()

El paso final es esperar durante un breve margen de tiempo (2 milisegundos) antes de realizar la siguiente lectura. Esto se logra con una sola línea de código que incluye la función delay(),que has visto en los capítulos anteriores.

El retardo detiene el programa durante un momento y permite que haya tiempo suficiente para tomar otra lectura del sensor. Hay un límite del número de lecturas del sensor que se pueden tomar cada segundo, por lo que el retardo ayuda a espaciar las lecturas el tiempo suficiente para obtener lecturas válidas.

> **Note** Un breve retardo al final del sketch permite que el código funcione sin problemas y las lecturas del sensor sean precisas.

resumen de LEA7_AnalogInOutSerial

Como has visto, el código loop() lee un valor analógico de un pin de entrada analógica, escala que se reduce a un número más pequeño, escribe el valor analógico en un pin PWM y presenta los resultados de todos esos pasos en el monitor serie para que puedas hacer una lectura de cómo cambia el valor.

El valor de la salida analógica, que puede ser cualquier número entre 0 V (apagado) y 5 V (máxima iluminación), cambia el brillo del led. En un punto intermedio, como por ejemplo 3,5 V, el led brillará menos que con 5 V. ¿Qué

más se puede modular de esta manera? A continuación, conectarás el altavoz y cambiarás las notas que se producen de una forma más dinámica que en el proyecto del Capítulo 5.

¿PREGUNTAS?

P: ¿Existen otras funciones que utilicen `Serial` además de `Serial.begin()`, `Serial.print()`, y `Serial.println()`?

R: Sí, hay bastantes, incluyendo `Serial.write()` y `Serial.read()`, que también se utilizan para comunicarte con el ordenador.

P: ¿Por qué nos molestamos en usar el carácter especial `\t` para crear una pestaña? ¿Hay otros caracteres especiales que deba conocer?

R: El uso de `\t` hace que la salida en el monitor serie sea mucho más fácil de leer, y esa es la única razón por la que lo utilizamos. Hay muchos caracteres especiales; uno de los que a veces puede ser útil es `\n`, que crea una nueva línea. Este carácter formatea el texto de una manera similar a la forma en la que lo hace `Serial.println()`, añade un salto de línea.

P: He oído hablar de la cadenas antes; son una forma de describir el texto en otros lenguajes de programación, ¿no?

R: Sí, a los caracteres alfanuméricos, incluidos espacios y puntuación, se los llama cadenas en muchos lenguajes de programación.

PIENSA SOBRE...

¿Qué otra información podrías querer enviar desde el Arduino al ordenador que te ayudara a depurar tus proyectos?

Antes hemos enviado lecturas analógicas de sensores, pero también puedes

presentar cadenas (para comprobar si algo sucede, como la pulsación de un botón) o lecturas digitales.

Has visto cómo el potenciómetro proporciona un rango de valores cuando se conecta a una entrada analógica, y sabes cómo asignar esos valores en tu sketch para obtener otros valores que puedas usar con un pin de salida PWM. Ahora vas a añadir un altavoz al circuito para controlar los tonos con valores analógicos.

montaje del altavoz

Vas a mantener todos los componentes que tiene el circuito y vas a montar un altavoz (figura 7.26). Tanto el led como el altavoz utilizarán los valores que se obtienen al girar el potenciómetro para controlar su funcionamiento.

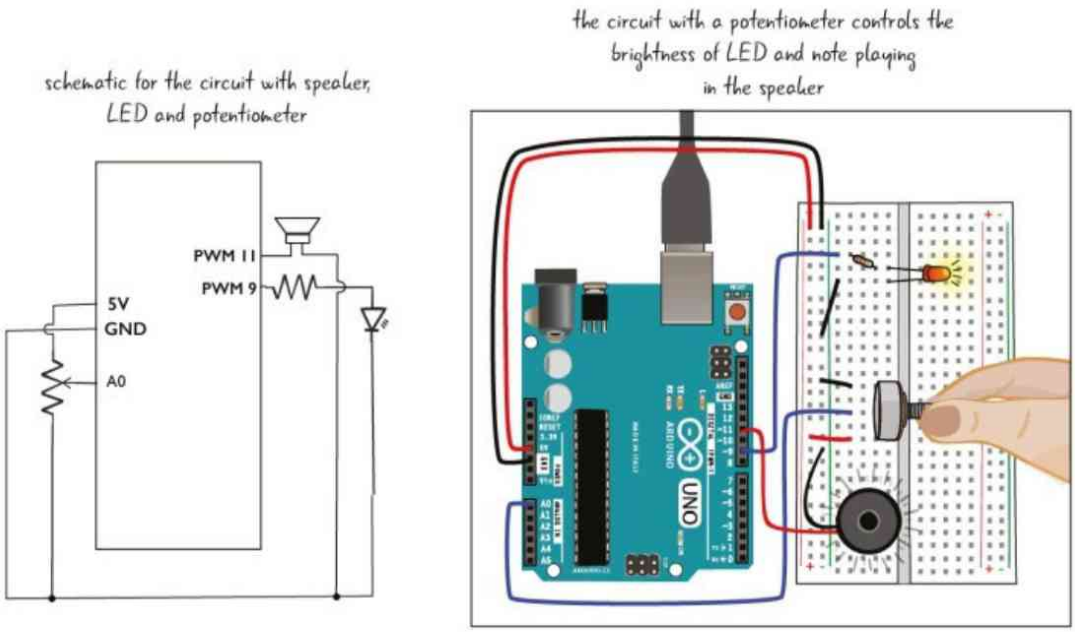


FIGURA 7.26 Incorporación de un altavoz al circuito.

Componente a añadir

1 altavoz de 8 ohmios

Conecta un extremo del altavoz al pin 11 y el otro, a tierra. Recuerda que el altavoz no tiene orientación (figura 7.27).

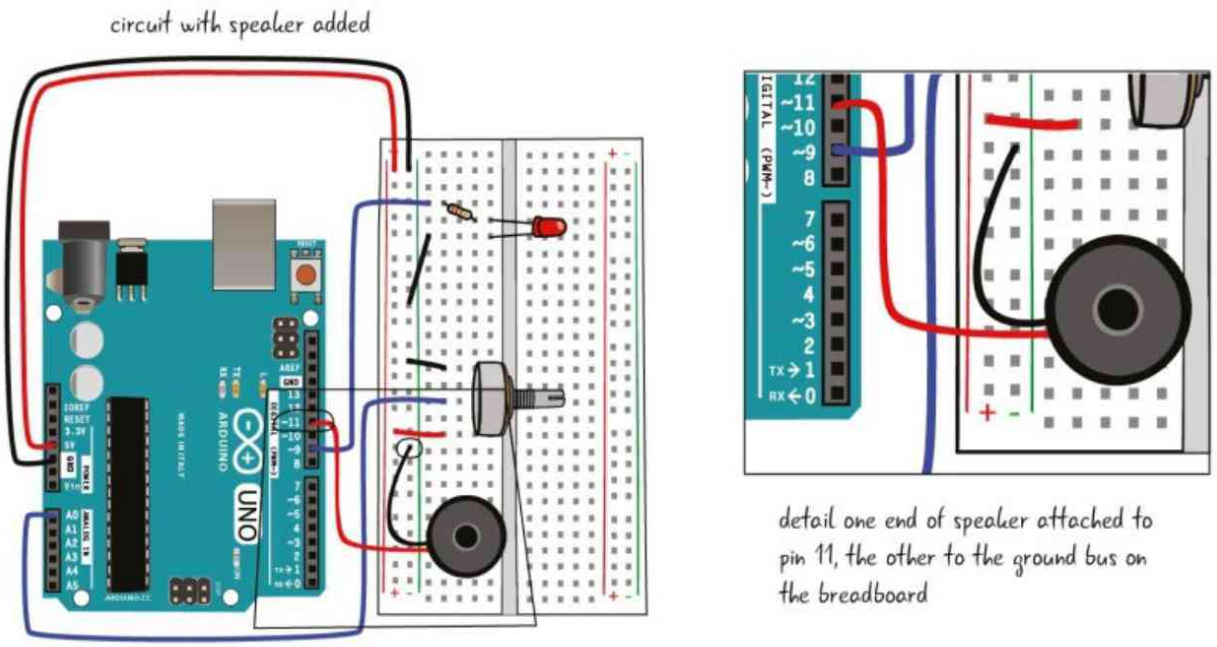


FIGURA 7.27 Adición del altavoz al circuito con el potenciómetro.

Una vez que hayas añadido el altavoz al circuito, conecta el ordenador al Arduino y abre el sketch LEA7_AnalogInOutSerial. Lo adaptarás.

Actualización del código

Guarda el sketch como **LEA7_VariableResistorTone**. Hay que añadir dos líneas de código para utilizar el altavoz: en la sección de inicialización, una variable para mantener el valor del pin del altavoz, y en la sección loop(), una llamada a la función tone(). También comentarás cada línea para explicar lo que hace.

```
// Analog input pin that the potentiometer is attached to
const int analogInPin = A0;
// Analog output pin that the LED is attached to
const int analogOutPin = 9;
// Analog output pin that the speaker is attached to
const int speakerOutPin = 11;

int sensorValue = 0;      // value read from the pot
int outputValue = 0;      // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}
void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1024, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);
  //call to the tone function
  tone(speakerOutPin, sensorValue);
  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  delay(2);
}
```

variable to hold the pin attached to the speaker

calls the tone() function

LEA7_VariableResistorTone

Un vez que hayas añadido estas líneas de código (inicializado la variable para sostener el pin del altavoz, añadido una llamada a la función tone() y comentado cada línea), conecta el ordenador al Arduino. Verifica y después sube el sketch.

Observa de nuevo que estás utilizando el potenciómetro para ajustar el tono del audio que sale del altavoz (figura 7.28). A medida que lo giras, el tono cambia: aumenta a medida que el led brilla con más intensidad y disminuye a medida que el led reduce su iluminación.

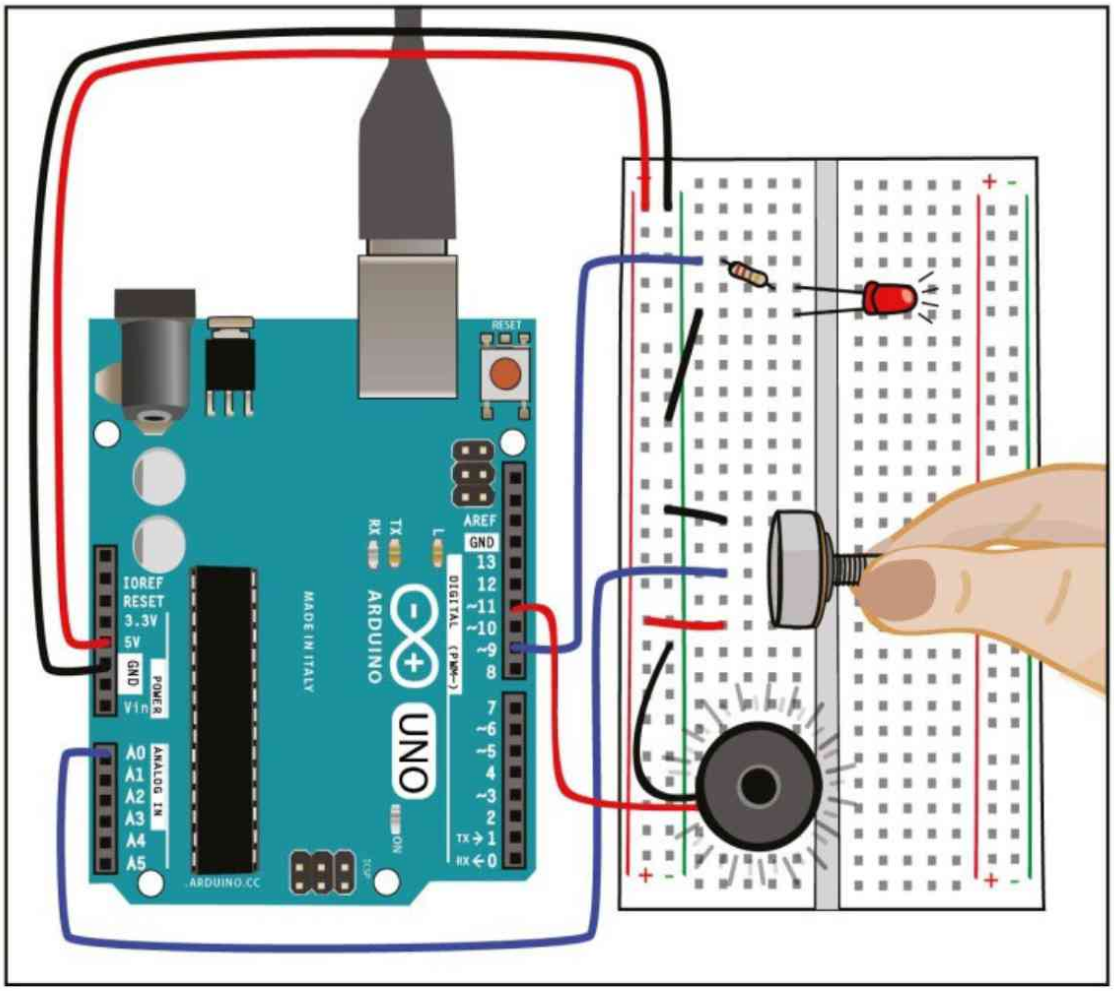


FIGURA 7.28 Al girar el potenciómetro, cambia el tono.

Antes de pasar a sustituir el potenciómetro por una fotorresistencia para montar el theremin, veamos más de cerca la llamada a la función tone().

```
tone(speakerOutPin, sensorValue);
```

Si recuerdas, cuando en el Capítulo 6 ("Interruptores, ledes y más") utilizaste la función tone(), esta toma dos argumentos: el pin al que está conectado el altavoz (en este caso, la variable speakerOutPin [asociada al pin 11]) y la frecuencia del tono que se va a reproducir, asociado en este caso a la

variable sensorValue, que es el valor derivado de la lectura del potenciómetro en el pin A0. No es necesario asignar este valor a la escala más pequeña, ya que el rango de frecuencias que acepta la función tone() es mucho más amplio que 0–255.

Ahora que ya has montado el circuito con el potenciómetro y el altavoz, cambiarás el potenciómetro por una fotorresistencia para montar el theremin.

MONTAJE DE LA FOTORRESISTENCIA

Coloca la fotorresistencia en la placa de pruebas de forma que un extremo esté en la misma fila de puntos de unión que el puente del pin analógico A0 (figura 7.29). El otro extremo debe estar en la fila de puntos, por debajo. La fotorresistencia no tiene orientación, así que no tienes que preocuparte por la forma en la que la colocas en la placa de pruebas.

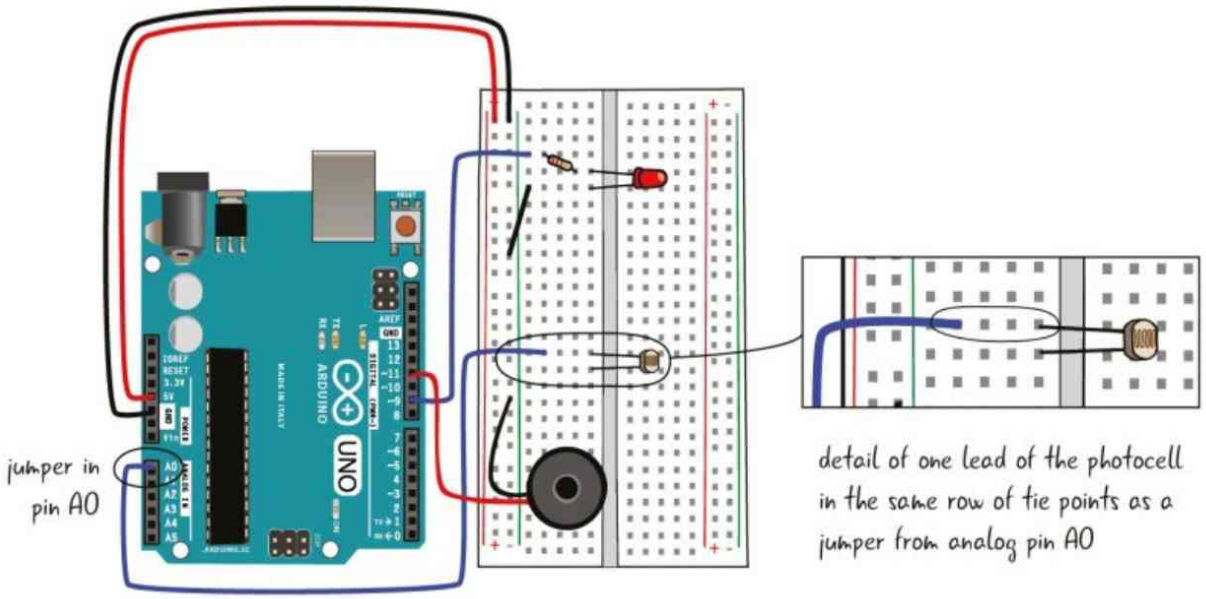


FIGURA 7.29 Adición de la fotorresistencia al circuito.

> **Note** Las fotorresistencias no tienen orientación y da igual el orden en el que las coloques en el circuito.

A continuación, coloca un puente para conectar el otro extremo de la fotorresistencia al bus de alimentación (figura 7.30).

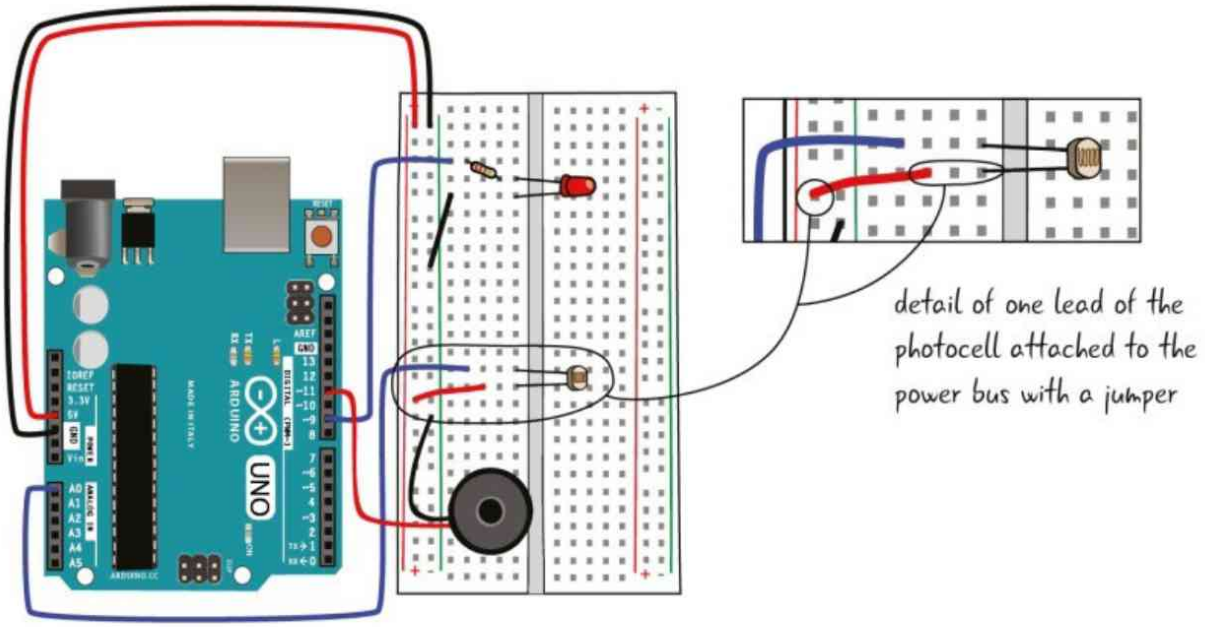


FIGURA 7.30 Colocación del cable del puente a la alimentación.

Ahora añade la resistencia de 10 KΩ a la misma fila de puntos de unión que el puente al pin A0 y a un extremo de la fotorresistencia (figura 7.31). El otro extremo de la resistencia de 10 KΩ se conecta mediante un puente al bus de tierra.

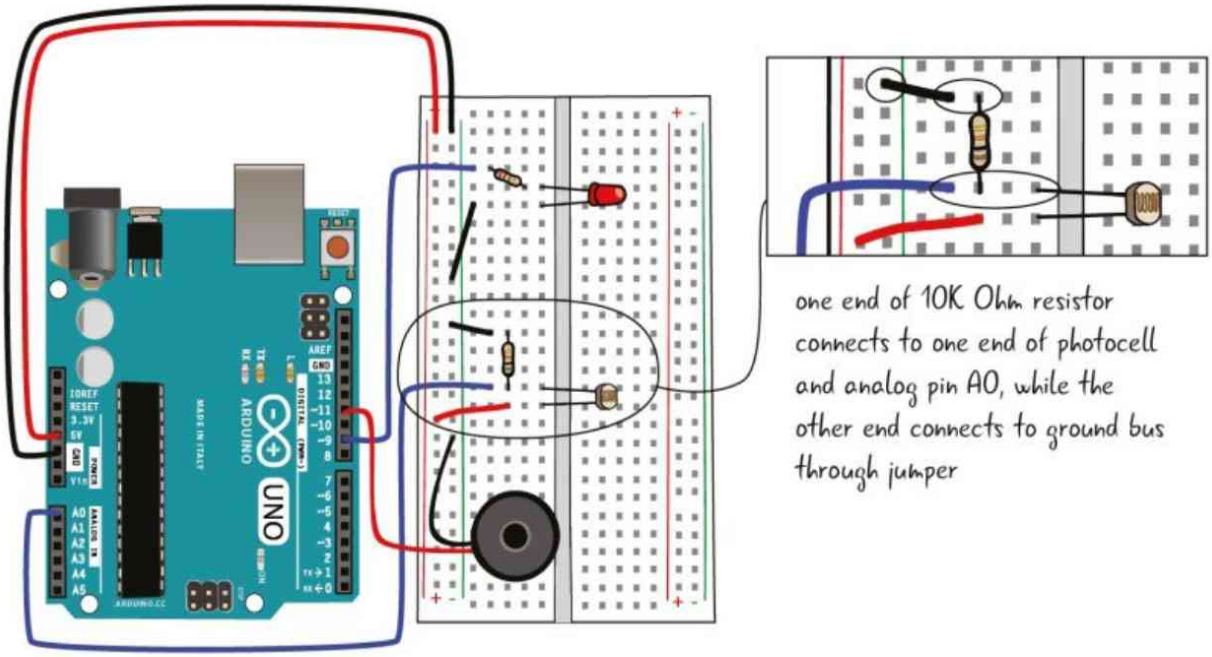


FIGURA 7.31 Incorporación de la resistencia de 10 KΩ al circuito.

Has completado el circuito. Conecta el ordenador al Arduino mediante el

cable USB y vamos a ver qué sucede (figura 7.32).

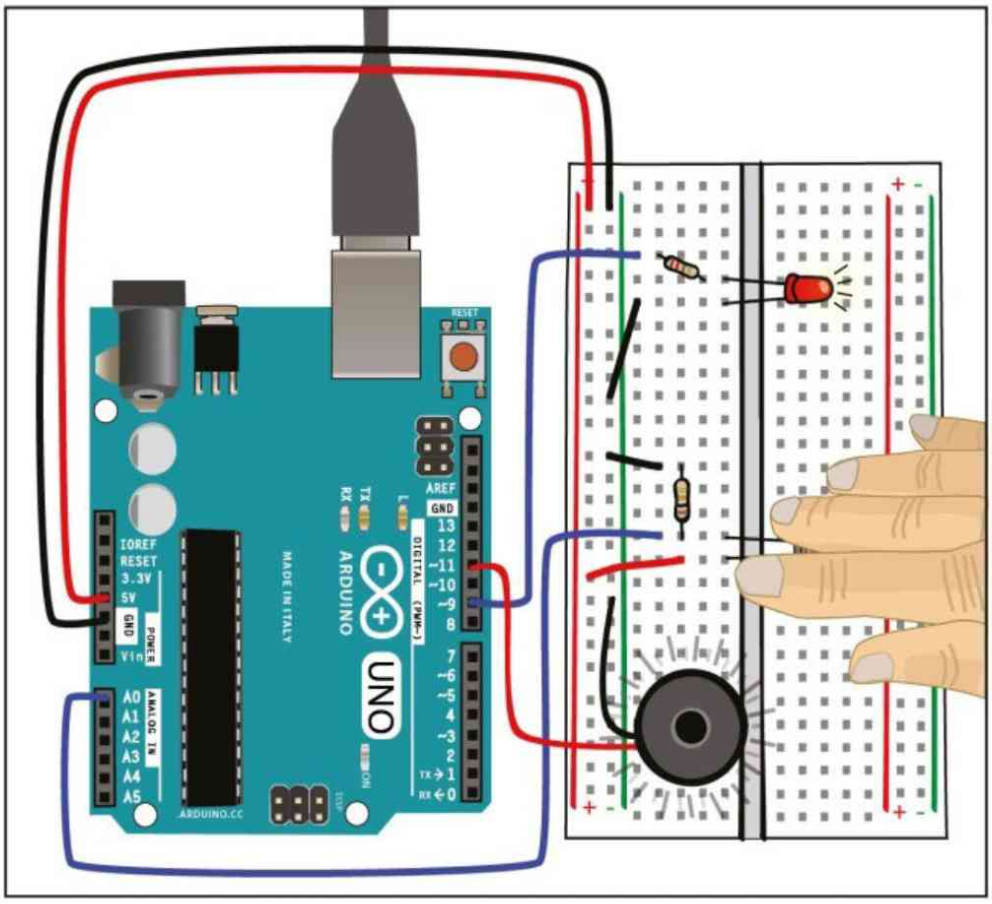


FIGURA 7.32 Prueba de funcionamiento del circuito completo.

DIVISOR DE VOLTAJE

La disposición de la fotorresistencia con la resistencia en serie es un ejemplo muy corriente de circuito al que se lo conoce como *divisor de voltaje*. Los divisores de voltaje son muy útiles cuando se utilizan varios sensores, como la fotorresistencia, pero no serán necesarios para todos los circuitos. Para comprender cómo funciona el divisor de voltaje, puedes pensar en él como si cambiara un determinado voltaje por otro voltaje de menor valor.

¿Por qué no necesitas otra resistencia cuando utilizas el potenciómetro en el circuito? Este lleva una resistencia con una escobilla, que divide la resistencia en dos partes (ver la figura 7.33). Al mover la escobilla se ajusta la relación de resistencia de las dos mitades.

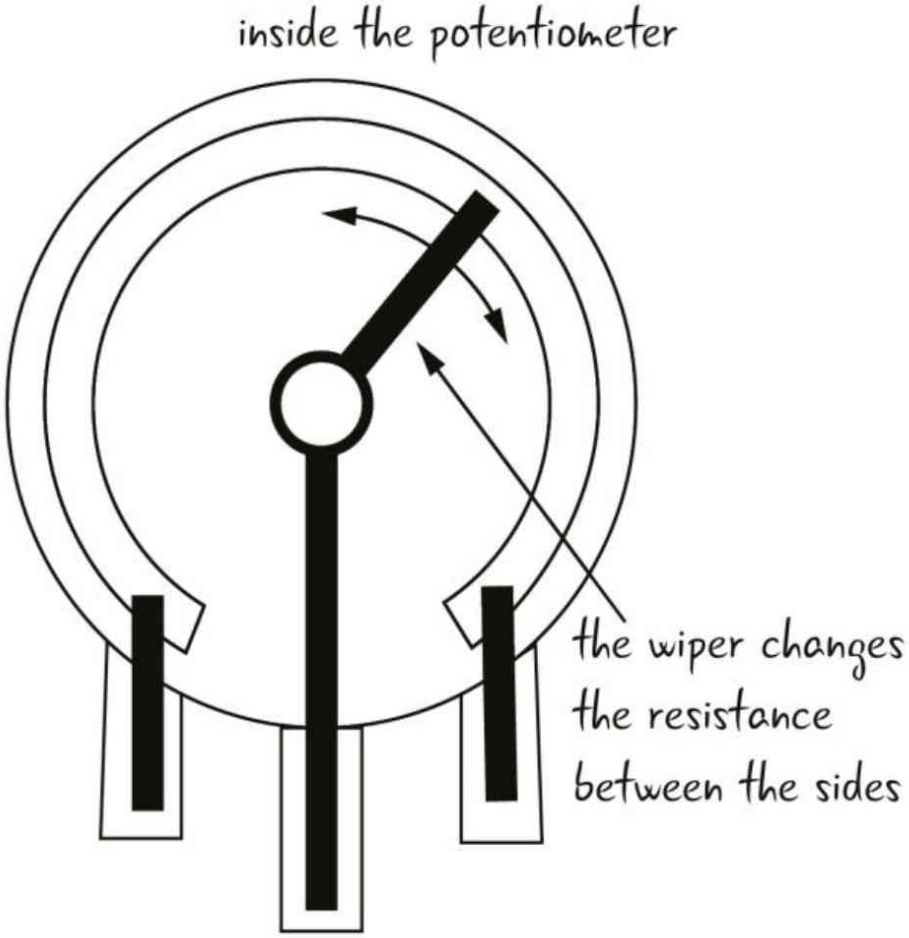


FIGURA 7.33 Potenciómetro visto a través de rayos X.

cómo tocar el Theremin

Puedes tocar el theremin luminoso acercando y alejando las manos a la fotocélula. Al variar la cantidad de luz que incide sobre la fotocélula, cambia su resistencia. Si mueves las manos acercándolas y alejándolas, podrás oír los inquietantes sonidos que cambian de tono (figura 7.34).

Puedes iluminar con una linterna la fotocélula (figura 7.35); el tono debería aumentar rápidamente al alcanzar el nivel de luz más alto.

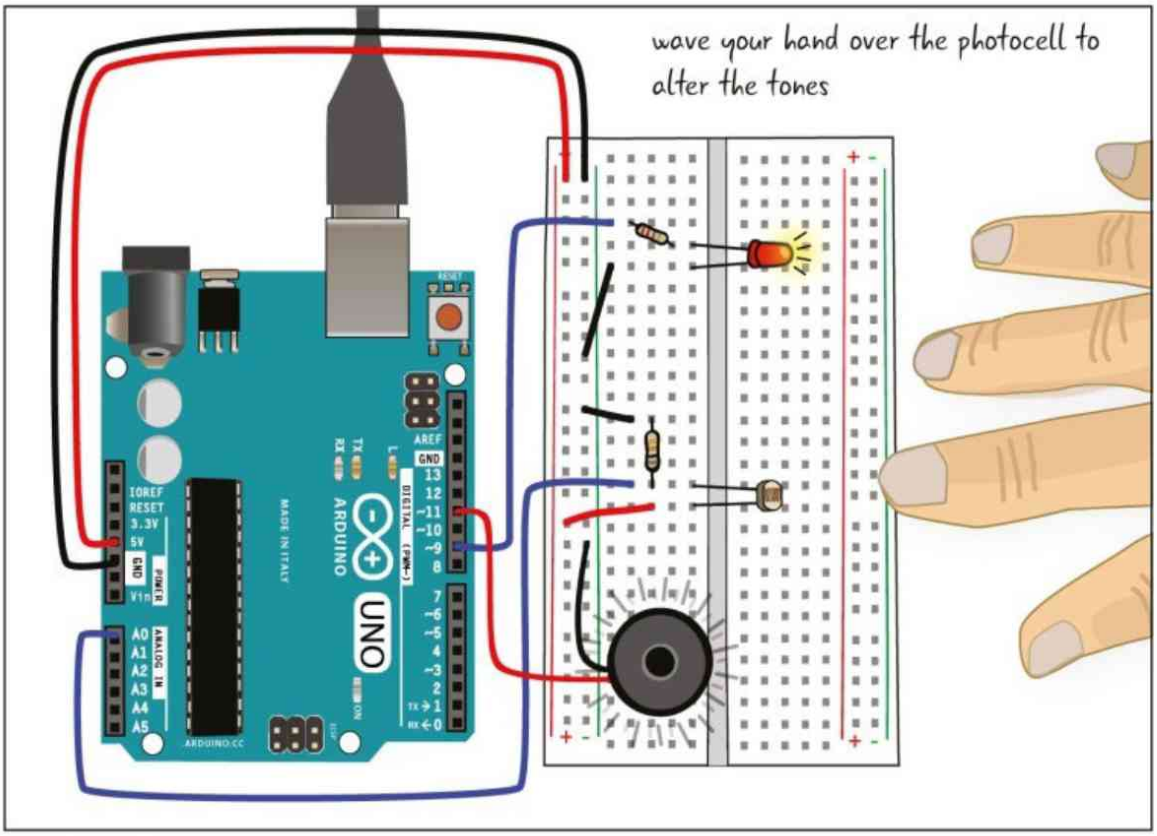


FIGURA 7.34 El tono cambia a medida que la fotocélula se expone a diferentes intensidades de luz.

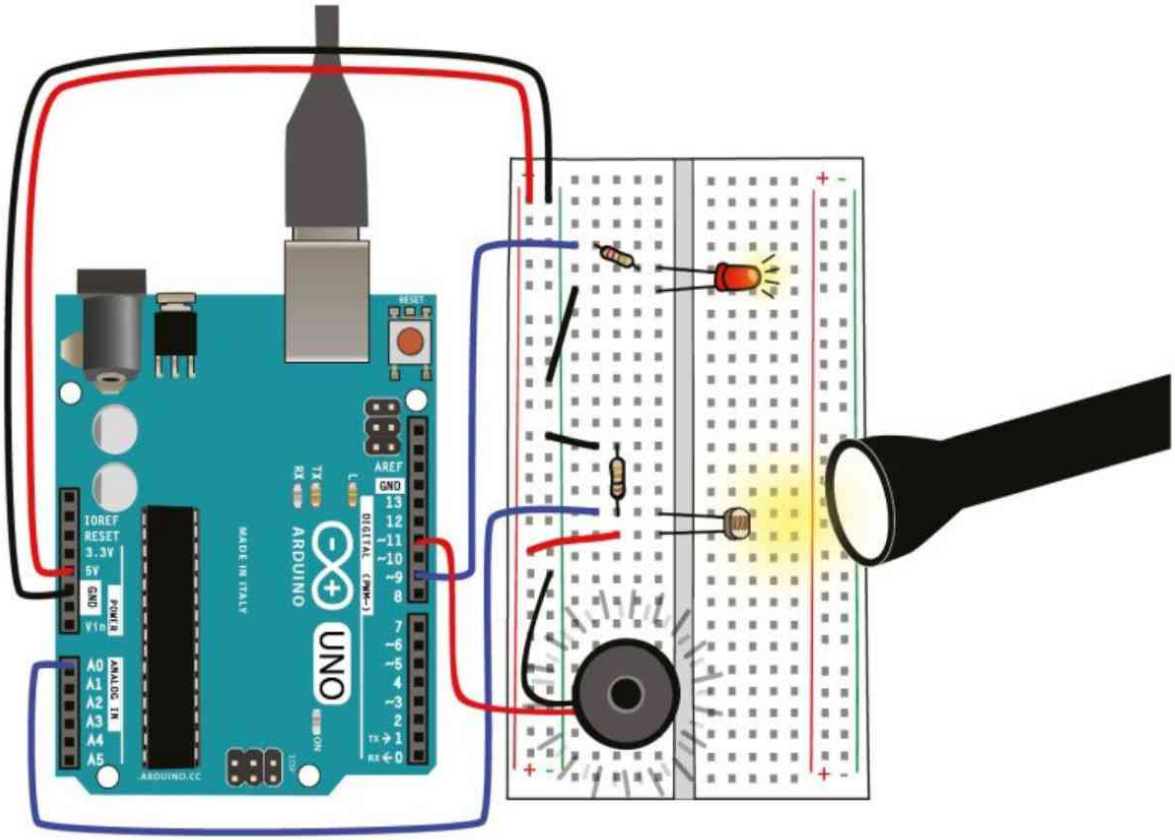


FIGURA 7.35 Iluminación de la fotocélula con una linterna.

¿Por qué no ha cambiado el código?

No era necesario cambiar el código al sustituir el potenciómetro por la fotocélula. ¿Cómo puede ser esto? Como se ha descrito anteriormente, la fotocélula funciona siguiendo el mismo principio básico que el potenciómetro. Ambos tipos de resistencias variables cambian el valor de su resistencia en el circuito, lo que, como se sabe por la ley de Ohm, altera el valor del voltaje (así como la corriente) en el Arduino. El código que has escrito para el theremin que funciona accionado por luz, funcionará con un potenciómetro, una fotorresistencia o cualesquiera otras resistencias variables que quieras usar.

LECTURA DE LA SALIDA SeriE

La ventana serie muestra los valores detectados por la fotocélula, pero ¿qué significan esos números? Una mayor incidencia de luz sobre la fotocélula

crea una menor resistencia y, consecuentemente, el valor del sensor será mayor (figura 7.36).

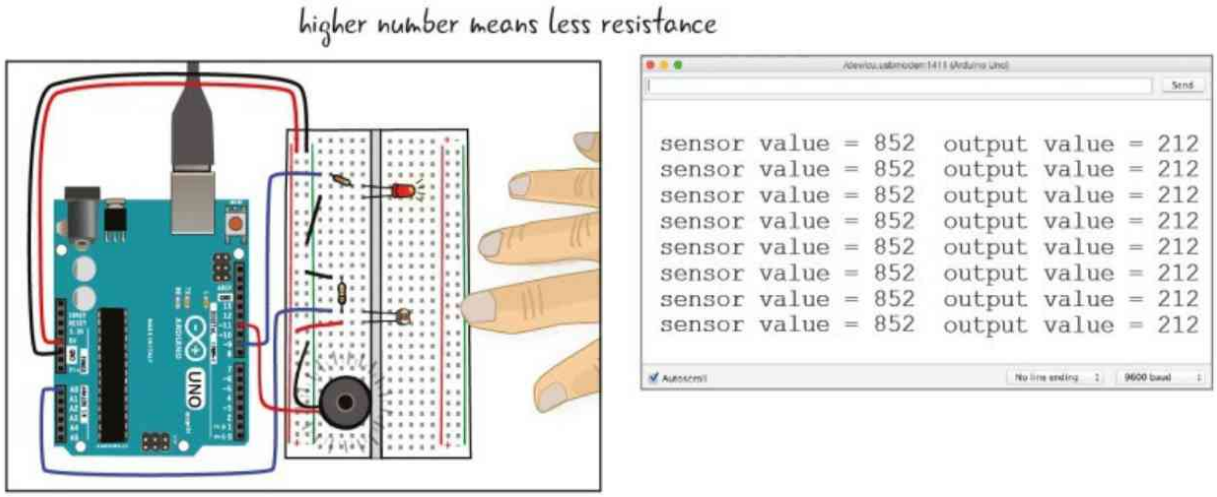


FIGURA 7.36 Mayor cantidad de luz supone un menor valor de la resistencia.

Si la fotocélula detecta menos luz, el valor de la resistencia del sensor es mayor, y el número que aparece en el monitor serie será menor (figura 7.37).

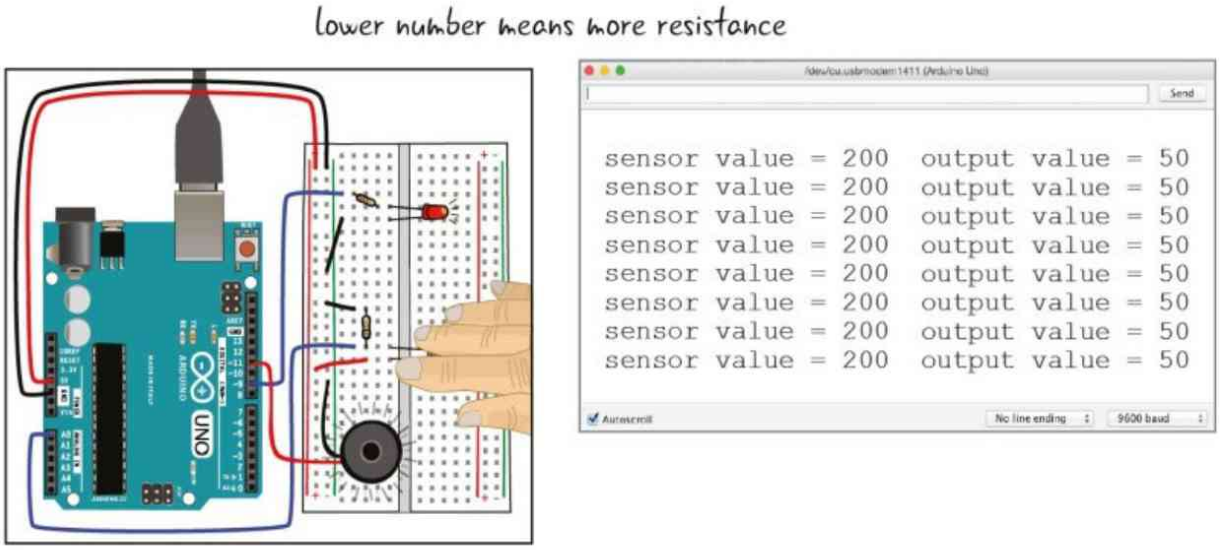
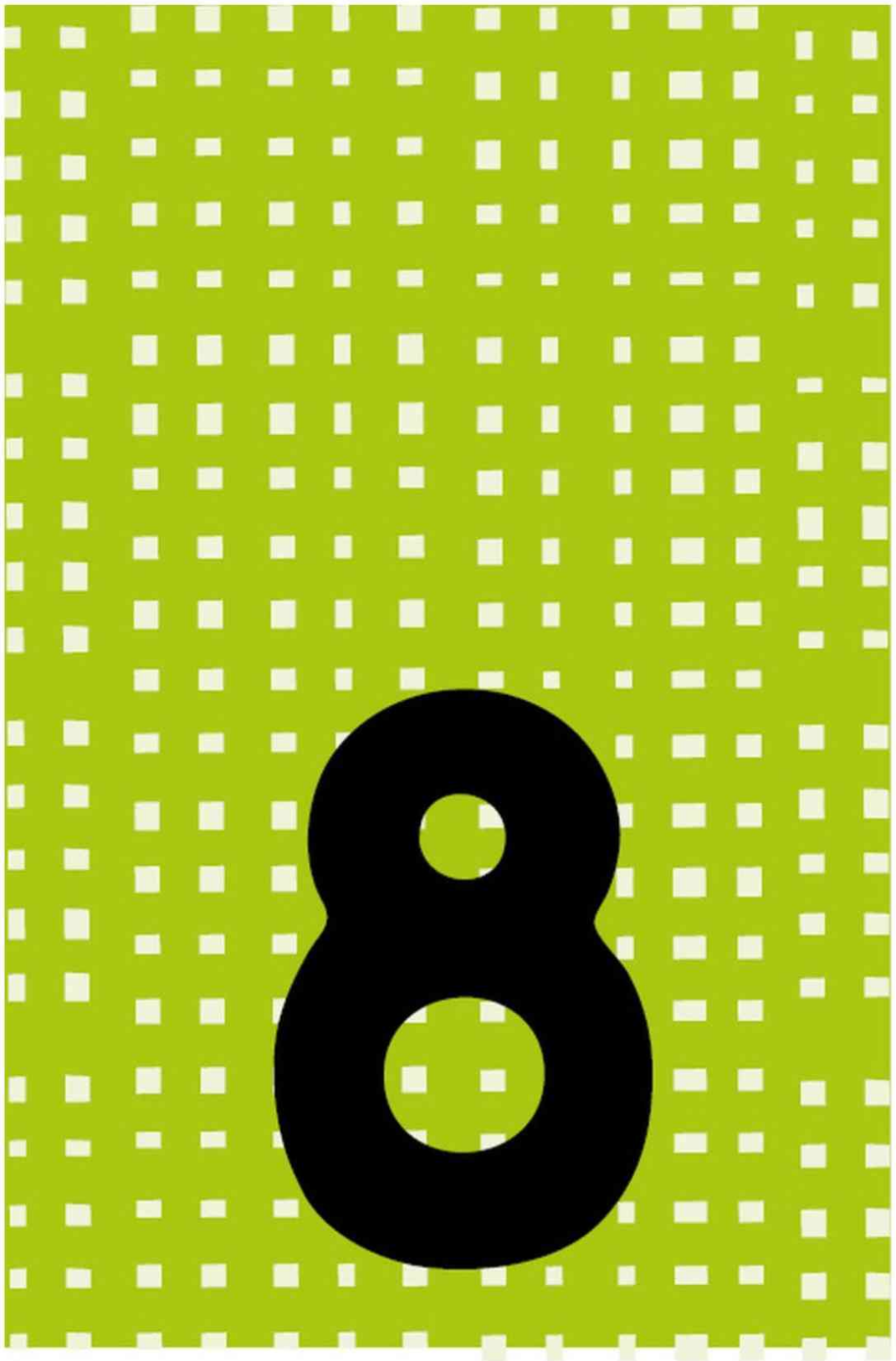


FIGURA 7.37 Menos luz supone un mayor valor de la resistencia.

Es una buena práctica acostumbrarte a revisar la información mostrada en el monitor serie. Es posible que la necesites para solucionar posibles problemas.

resumen

En este capítulo, has aprendido a conectar un potenciómetro y una fotorresistencia a los pines de entrada analógica del Arduino para obtener un rango de valores que utilizas en los sketches. Has aprendido lo que significa PWM y cómo el Arduino utiliza los pines PWM con analogWrite() para simular una salida analógica. Ahora sabes cómo asignar valores del rango que recibes de las entradas a un rango más apropiado a las salidas que estás usando. También has aprendido a utilizar el monitor serie en el IDE de Arduino para leer los valores de las entradas. En el siguiente capítulo, te apoyarás en estos conocimientos para montar un circuito que hace girar un motor. Puedes descargar el código para LEA7_VariableResistorTone en: github.com/arduinotogo/LEA/blob/master/LEA7_VariableResistorTone.ino.


8

Servomotores

En este capítulo añadirás movimiento a los proyectos con Arduino. Utilizarás servomotores, como los que se muestran en la figura 8.1.

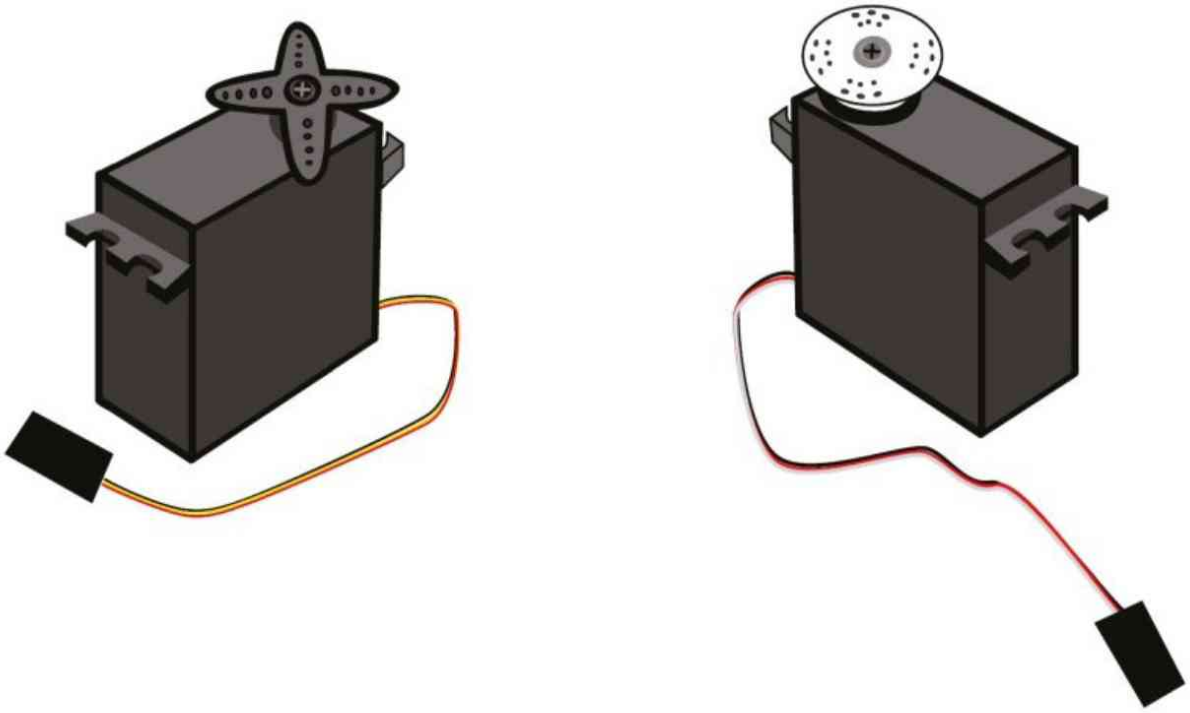


FIGURA 8.1 Servomotores para aficionados.

Los servomotores son un tipo especial de motores que se pueden programar fácilmente para que giren hasta una posición precisa. Un servomotor contiene un conjunto de engranajes y un mecanismo de control que hace girar un eje un número específico de grados. Debido a que los servos son relativamente fáciles de controlar, son una buena introducción al uso de motores en tus proyectos. Aunque hay muchos tipos de servos, los que te recomendamos que utilices pueden girar entre 0 y 180 grados.

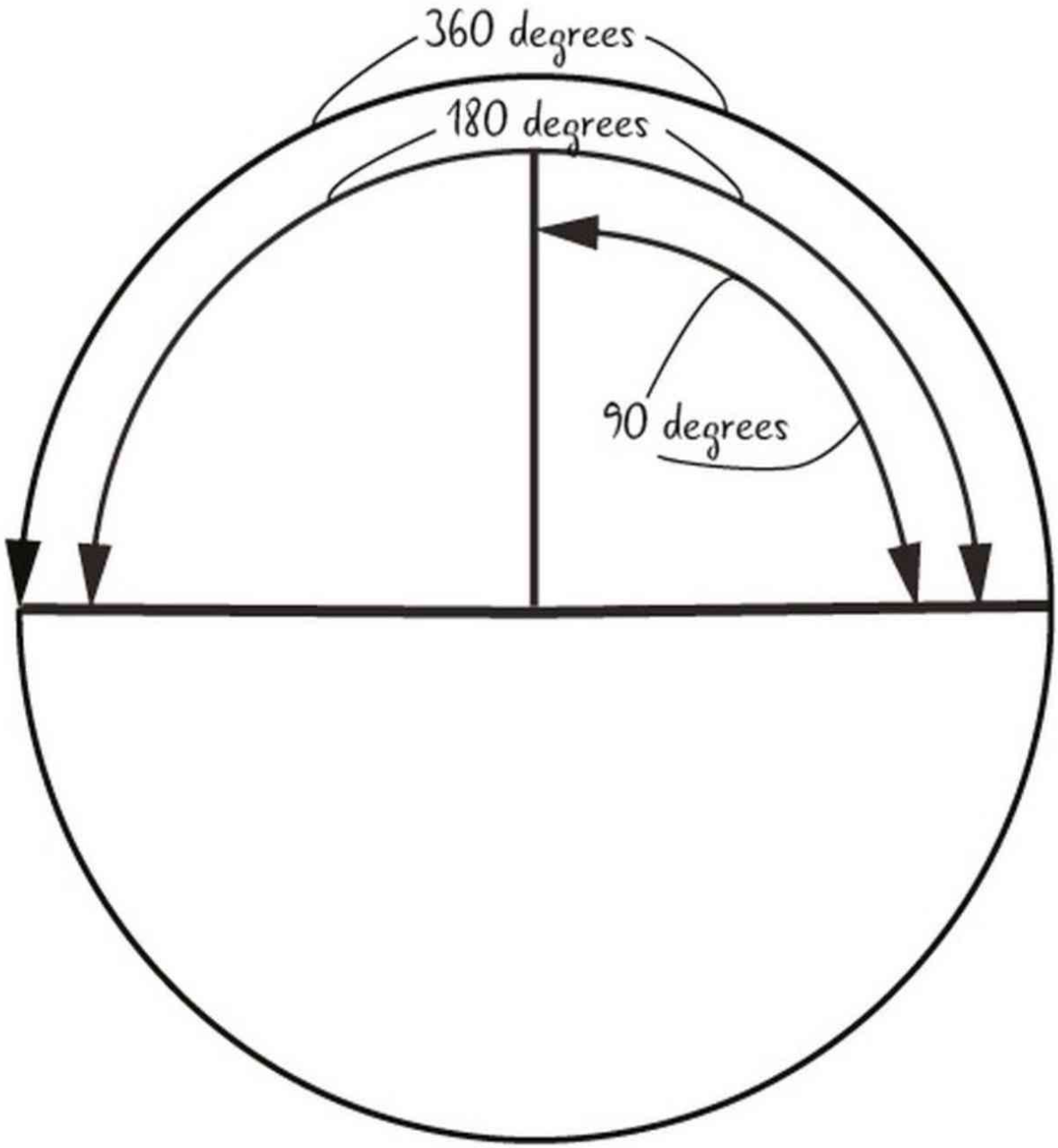


Figura 8.2: **Diagrama de grados de rotación.**

Primero, harás girar el servo de forma continua con un sketch de ejemplo del IDE de Arduino. Después, controlarás un servo con un potenciómetro. Por último, añadirás un segundo servo al circuito y adaptarás el sketch para que el movimiento de ambos servos se controle girando el potenciómetro.

También trataremos algunos conceptos de programación que no has visto antes, incluyendo bucles y funciones personalizadas.

El tipo de servomotores con los que trabajarás se denominan servomotores de rotación posicional. Están limitados a 180 grados, o dicho de otra forma, a la mitad de una rotación completa; los grados de rotación se muestran en la figura 8.2 (página anterior). Dentro de esta gama, son precisos, lo que significa que si necesitas que el eje del motor apunte a un punto exacto, permiten un gran ajuste.

Los servomotores se utilizan en una amplia variedad de aplicaciones, entre las que se encuentran modelos de aviones de aficionados, robótica y proyectos de arte de todas las formas y tamaños.

Cómo ondear banderas

La Figura 8.3 muestra un dibujo del primer proyecto que vas a realizar. Revisarás por encima los datos analógicos, estudiarás más detenidamente los servos y luego empezarás el montaje.

revisión de los datos analógicos

En el capítulo anterior viste que los datos analógicos pueden referirse a cualquier información que tenga más de los dos valores posibles que puede contener la información digital (descritos alternativamente como 1 o 0, verdadero o falso, HIGH o LOW). En los sketches del Arduino, has visto que el número de valores posibles estaba asignado a menudo a un rango particular (para entradas, un valor entre 0 y 1.023, y para salidas, un valor entre 0 y 255). Disponer de un rango más amplio de valores te permite hacer algo más que simplemente encender o apagar los componentes del circuito.

Los servomotores utilizan un posicionamiento preciso. En los proyectos que montes en este capítulo, usarás datos analógicos para establecer la dirección en la que está orientado el eje del motor.

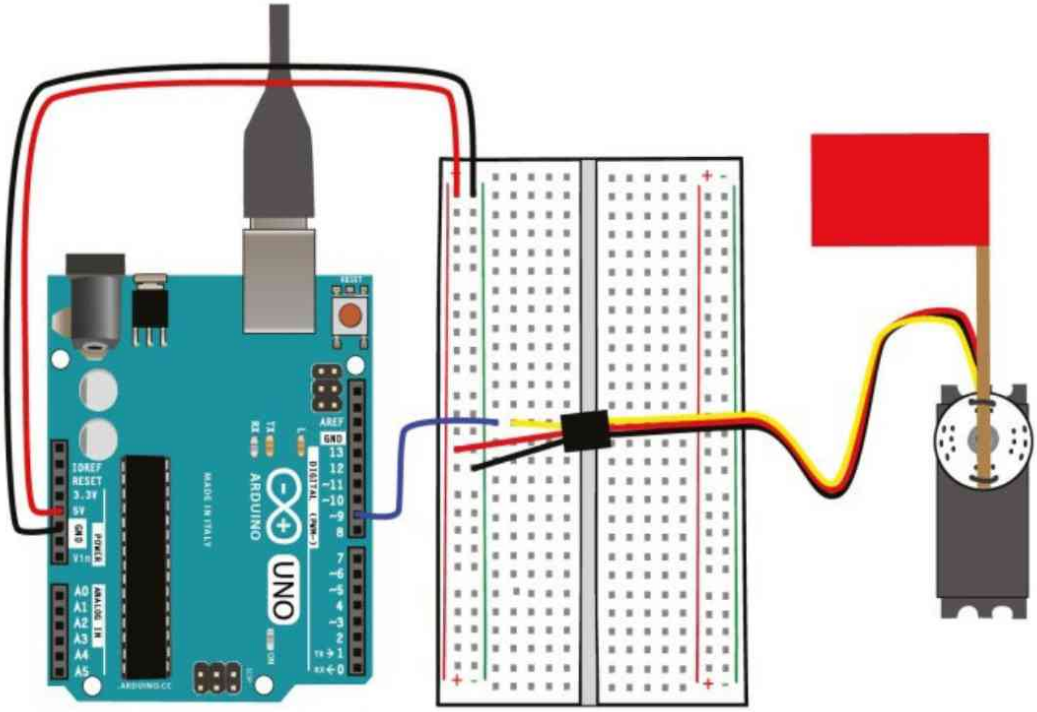


FIGURA 8.3 El servo gira y hace ondear la bandera.

los Servos de Cerca

Como hemos dicho, hay muchos tipos de servomotores disponibles. Te recomendamos utilizar un motor con un rango de 180 grados que funcione de 4.8 V a 6 V. Este tipo de servo estándar lo tienen normalmente disponible muchos vendedores en línea, tiendas de *hobbies* o tiendas que venden componentes electrónicos.

Partes de un servo

Los mecanismos que hacen girar el servo (motor, engranajes y circuito) están contenidos en una carcasa. La lengüeta es la parte del eje móvil que se extiende fuera de la caja. El cuerno, o brazo, se fija a la lengüeta. Un tornillo sujeta el cuerno en su posición a la lengüeta. El conjunto de materiales de montaje que viene con el servo generalmente tendrá distintos tipos de cuernos que se sujetan con un tornillo para poderlos cambiar dependiendo de las necesidades del proyecto, así como algunos tornillos y otros tipos de sujeciones. Los servos están diseñados para poder desatornillar fácilmente el cuerno y reemplazarlo por otro. También suelen tener bridas de montaje en la parte delantera y trasera, lo que hace que el servo sea fácil de montar en tus proyectos.

VENDEDORES EN LÍNEA

Entre los proveedores en línea se encuentran los siguientes:

adafruit.com
sparkfun.com
makershed.com
microcenter.com
servocity.com

Cuando compras un servomotor, además del motor viene un paquete que contiene las herramientas de montaje, como se ve aquí:

En la parte delantera de la carcasa, cerca de la parte inferior, hay conectado un cable. Este está formado por tres cables codificados por colores; el cable negro se conectará a tierra, el cable rojo se conectará a la alimentación y el tercer cable, a veces amarillo, a veces azul y a veces blanco, es el cable de control. Conectarás el cable de control a un pin del Arduino. El servo tiene un enchufe, o conector, en el extremo del cable para conectarlo a un circuito. La figura 8.4 muestra un servo con y sin cuerno.

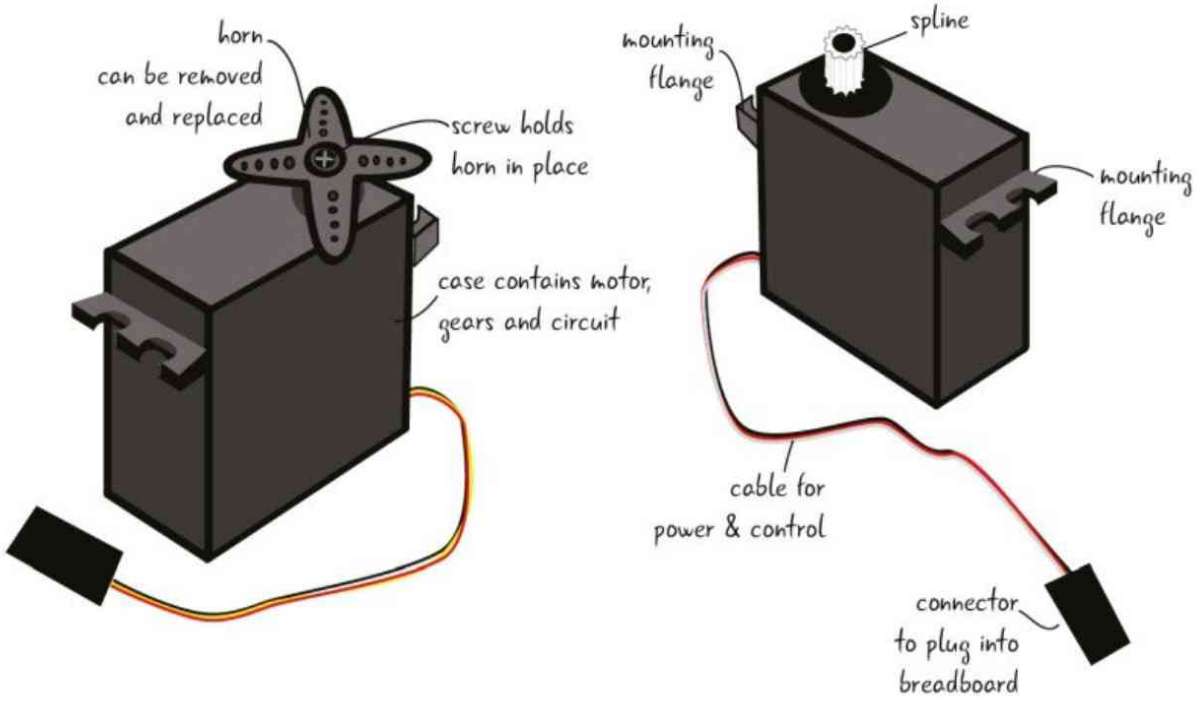


FIGURA 8.4 Dos servos, uno con cuerno y el otro con el cuerno separado.

Los diferentes estilos de cuernos que vienen con el servo te permiten conectar el adecuado para el proyecto que estés llevando a cabo (figura 8.5).

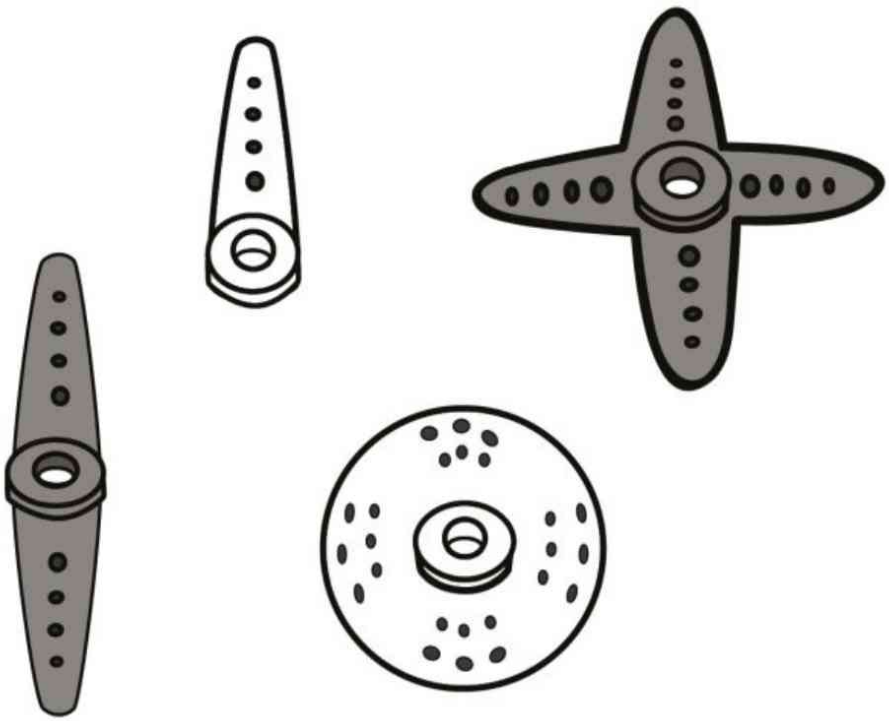


FIGURA 8.5 Algunos tipos de cuernos del servo.

¿PREGUNTAS?

P: ¿Por qué empezamos con los servomotores en lugar de hacerlo con otro tipo de motor?

R: Empezamos con los servomotores porque son fáciles de controlar y de cablear.

P: ¿Necesitaré otros tipos de motores para mis proyectos?

R: Sí, aunque los servos son útiles, no funcionarán en todos los proyectos. A veces es apropiado utilizar un motor de corriente continua o un motor paso a paso debido a los requisitos de potencia o al cometido concreto que deban realizar. Se conectan al circuito y se programan de forma diferente, pero no los vamos a tratar en este libro.

P: Los colores de los cables de mi servo no coinciden con los que se han indicado aquí. ¿Qué cables van a la alimentación y a tierra?

R: En algunos servos, el cable de tierra es marrón; generalmente, el de alimentación es rojo en la mayoría de los servos para aficionados. Hay que mirar en la parte delantera del servo para ver la posición de los conductores del cable. Generalmente, el conductor de tierra está a la derecha, el conductor de alimentación, en medio, y el conductor de control, a la izquierda.

montaje paso a paso de un circuito con servo

Necesitarás las siguientes piezas:

- ▨Servomotor estándar
- ▨Placa de pruebas
- ▨Cables de puente
- ▨Agitador de café, de madera, o tira de cartón
- ▨Cinta
- ▨Papel de color
- ▨Arduino Uno
- ▨Cable USB tipo A-B
- ▨Ordenador con el IDE de Arduino instalado

La figura 8.6 muestra el esquema y el dibujo del primer circuito que vas a montar. Como de costumbre, los buses de alimentación y tierra de la placa de pruebas están conectados a 5 V y GND en el Arduino. Puedes ver que el servo tiene tres conductores: uno conectado a la alimentación, otro a tierra y otro a un pin en el Arduino.

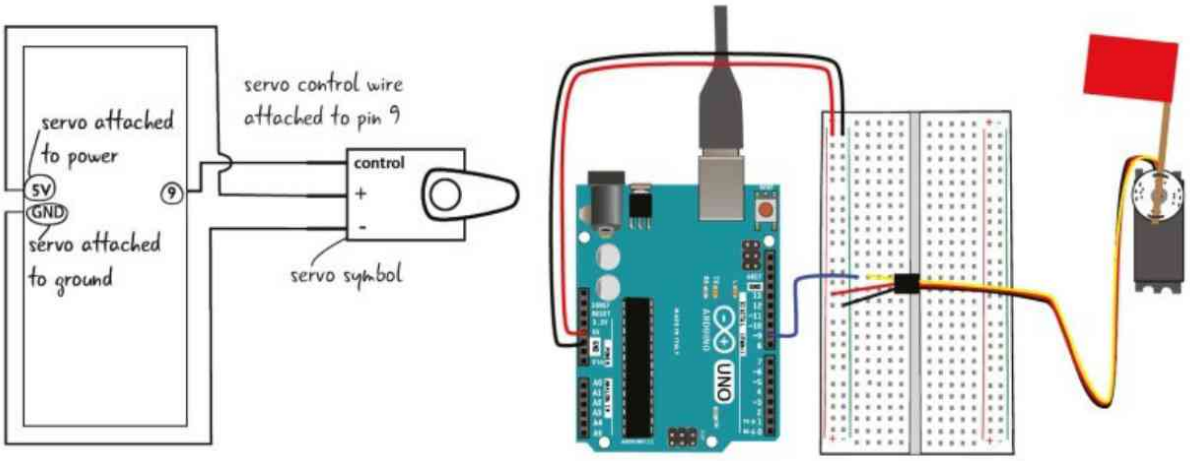


FIGURA 8.6 Esquema y dibujo del primer circuito con un servo.

Hay algunas cosas que debes saber sobre el servo que te permitirán configurarlo con mayor facilidad.

Preparación del Servo

Ya has visto que el servo viene con un paquete con diferentes cuernos. Es posible que quieras cambiar el cuerno que lleva instalado el servo de fábrica. Usa un destornillador pequeño para quitar el tornillo que sujeta el cuerno y sustitúyelo por otro, como se muestra en la figura 8.7. En nuestros ejemplos vamos a utilizar un cuerno circular.

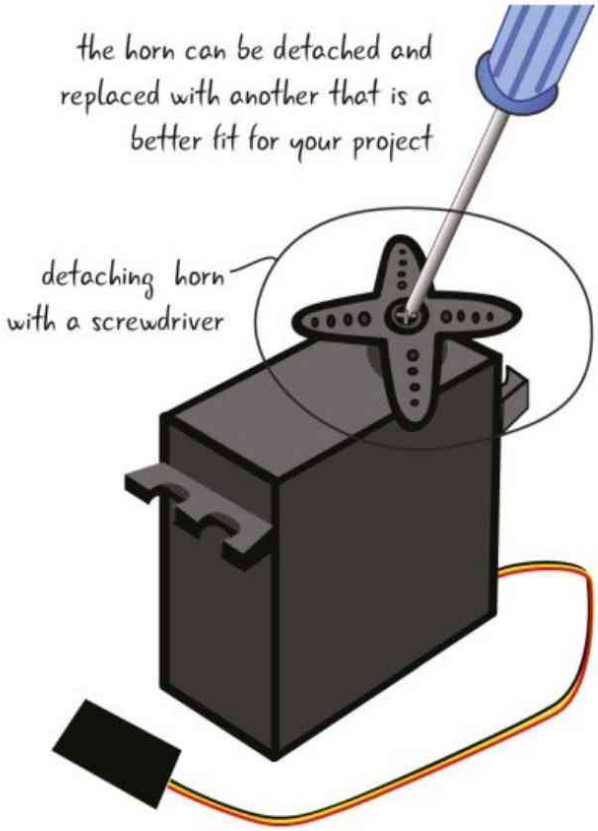


FIGURA 8.7 Desmontaje del cuerno.

Hemos hecho una bandera con un agitador de café y un trozo de papel de colores. Una tira de cartón con un trozo de espuma de color también funcionaría. Confecciona una bandera con cualquier material que tengas a tu alrededor y fíjala al cuerno con alambre, como se muestra en la figura 8.8.

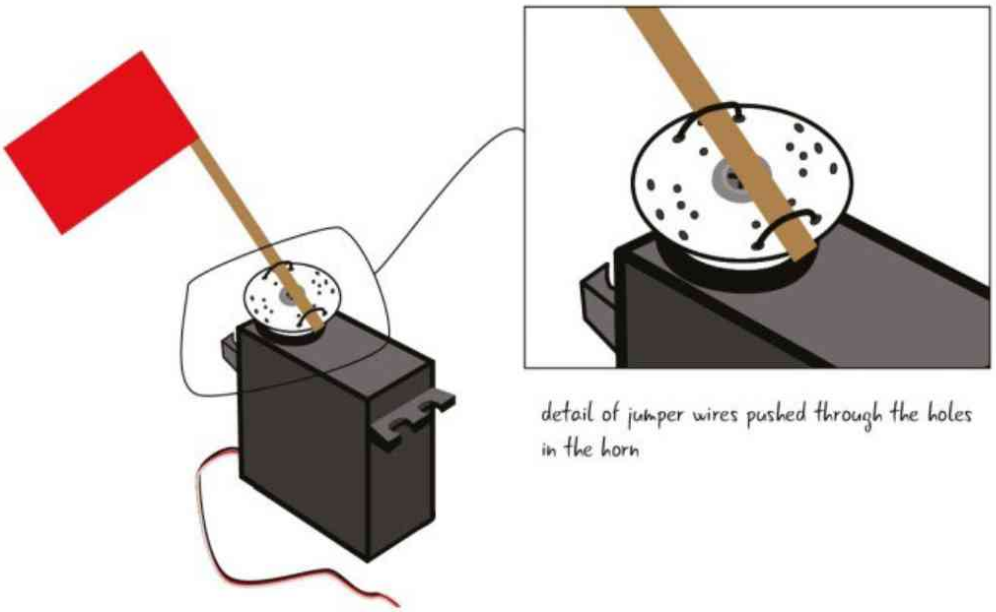


FIGURA 8.8 Fijación de la bandera al cuerno del servo.

Antes de conectar el servo a la placa de pruebas, tendrás que colocar cables de puente en el enchufe/conector del servo. Como ya sabes, hay un conductor de control, un conductor que va a la alimentación y otro que va a tierra. Sigue las convenciones de color (cable rojo conectado a la red eléctrica; negro, a tierra) como de costumbre. Si tienes un puente del mismo color que el cable de control, úsalo, o utiliza un color que se distinga de los cables rojo y negro. En nuestro ejemplo (figura 8.9), el cable de control es amarillo, pero a veces ese cable puede ser de otro color, como por ejemplo blanco.

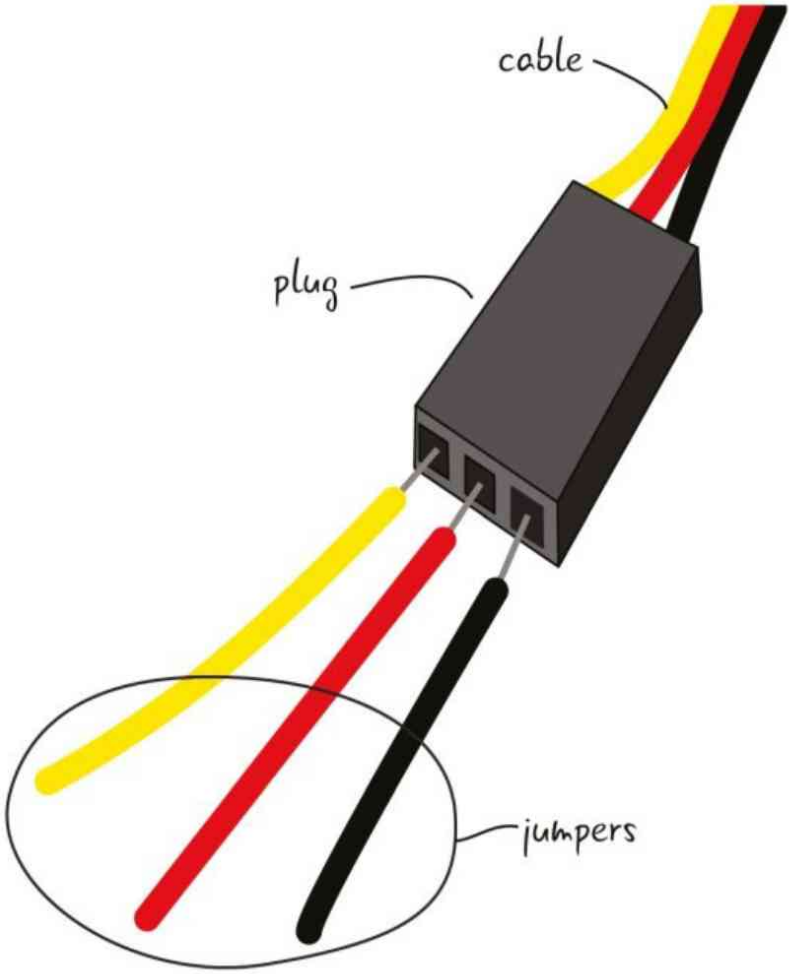


FIGURA 8.9 Conductores del servo de cerca.

conexión del servo

Conecta puentes desde la GND del Arduino al bus de tierra en la placa de pruebas y desde 5 V en el Arduino al bus de alimentación.

Ahora conecta el servo a la placa de pruebas. Conecta el puente de color rojo al bus de alimentación, y el negro de tierra al bus de tierra. Luego coloca el puente del cable de control en su fila de puntos de conexión (figura 8.10).

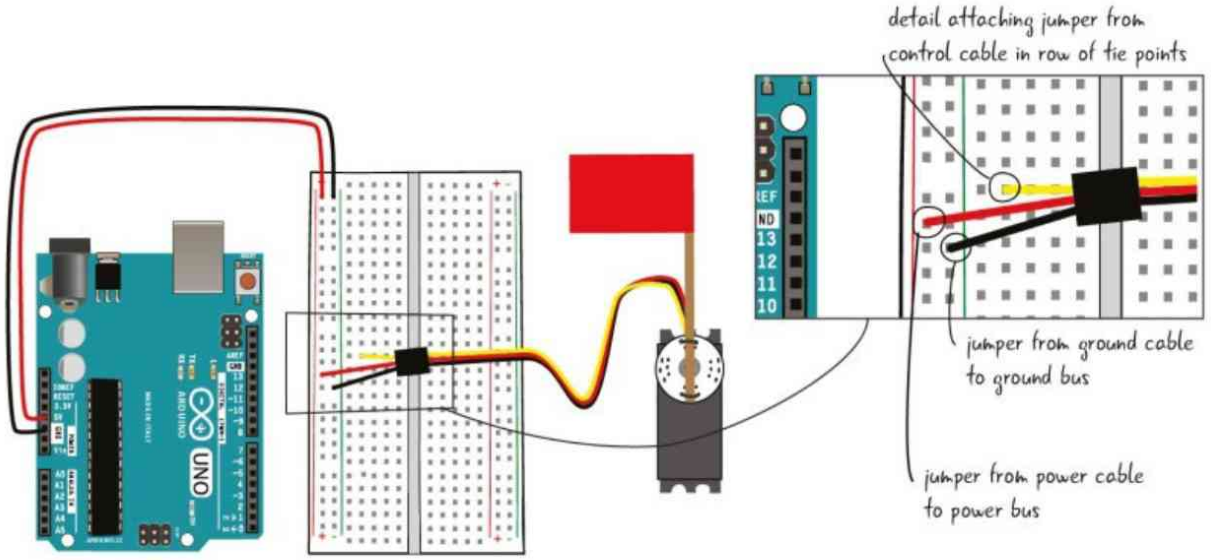


FIGURA 8.10 Conexión del servo a la placa de pruebas.

A continuación, conecta un puente del pin 9 a la misma fila de puntos de unión que el puente del cable de control, como se muestra en la figura 8.11. Conecta el servo al pin 9, porque ese es el pin al que está conectado en el sketch que vas a descargar. Se puede conectar a cualquiera de los pines digitales, del 2 al 13.

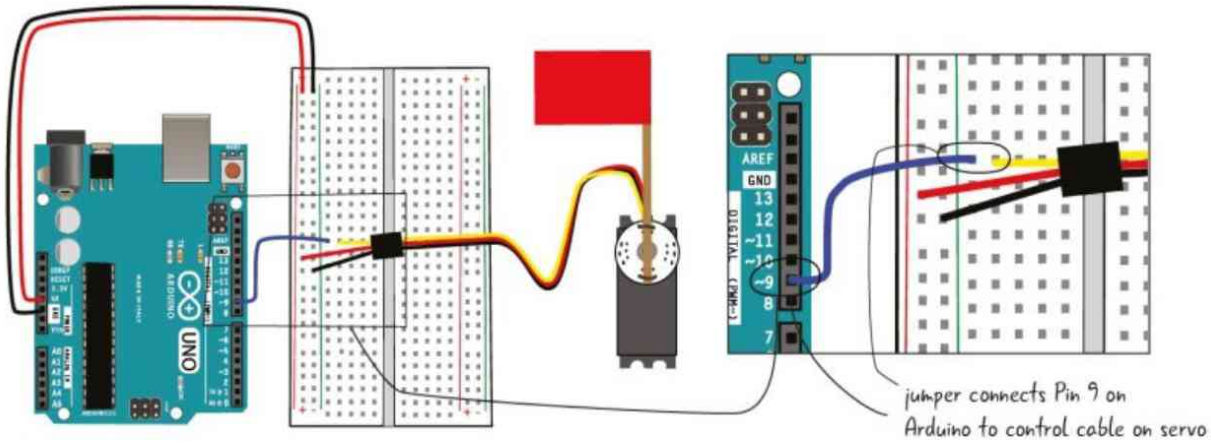


FIGURA 8.11 Conexión del conductor de control del servo mediante un puente al pin 9 del Arduino.

Ya estás listo para descargar el sketch del IDE de Arduino.

conexión del ordenador y descarga del sketch sweep

Ahora que has completado el cableado del circuito, necesitas descargar el sketch al Arduino para hacer funcionar el servo. El Arduino incluye algunos sketches sobre el uso de servomotores, y para este primer ejemplo, utilizarás el sketch Sweep incluido en la carpeta Servo de ejemplos de sketches ("Archivo > Ejemplos > Servo > Sweep").

Cuando hayas abierto el sketch, lo guardas como **LEA8_Sweep**. Si todavía no lo has hecho, conecta el ordenador al Arduino y sube el sketch.

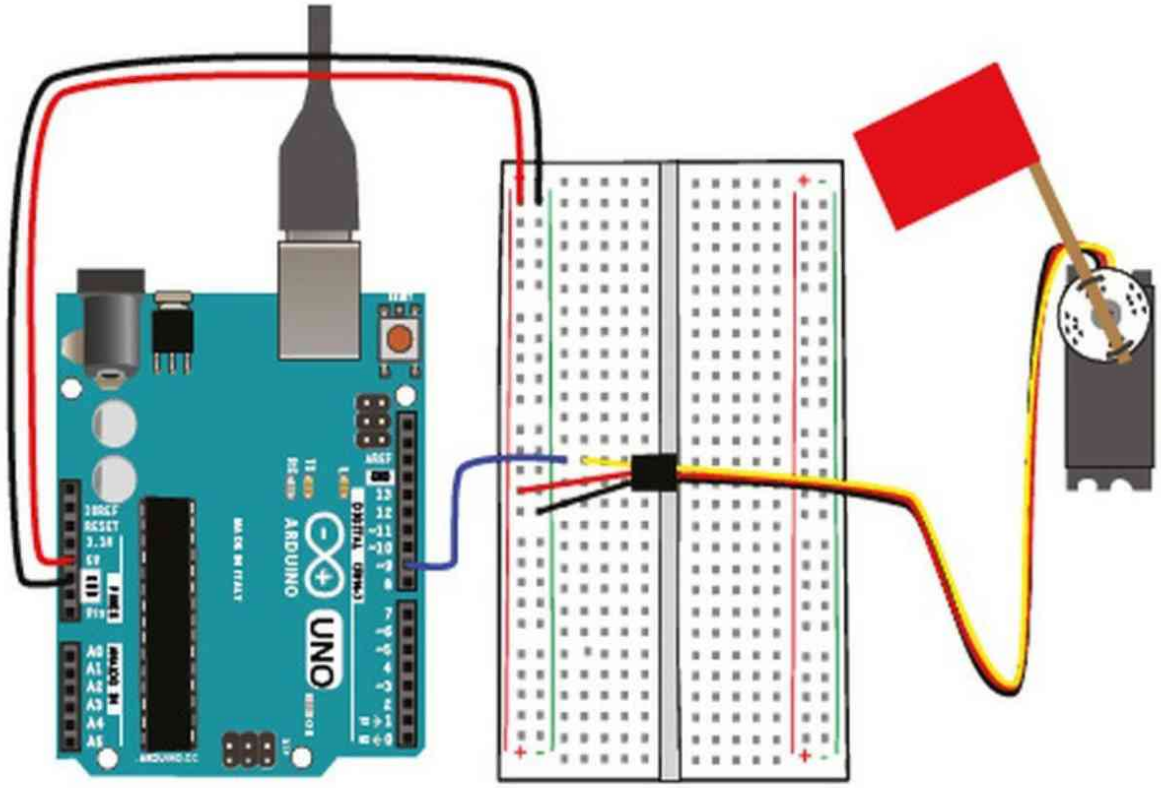


Figura 8.12:**La bandera ondea.**

¡Agitar la bandera!

Debes empezar a ver al servomotor mover la bandera sujeta al cuerno a 180 grados en una dirección, y luego invertir la dirección y volver a su posición inicial (figura 8.12). El servo continuará enlazando estos movimientos, uno tras otro, mientras el Arduino esté conectado a la alimentación. Echemos un vistazo más de cerca al código y trataremos de explicar lo que hace cada línea.

resumen de LEA8_Sweep

En el desglose de algunos de los sketches de este capítulo hemos eliminado la sección de comentarios para facilitar la lectura. La figura 8.13 muestra un rápido vistazo al sketch.

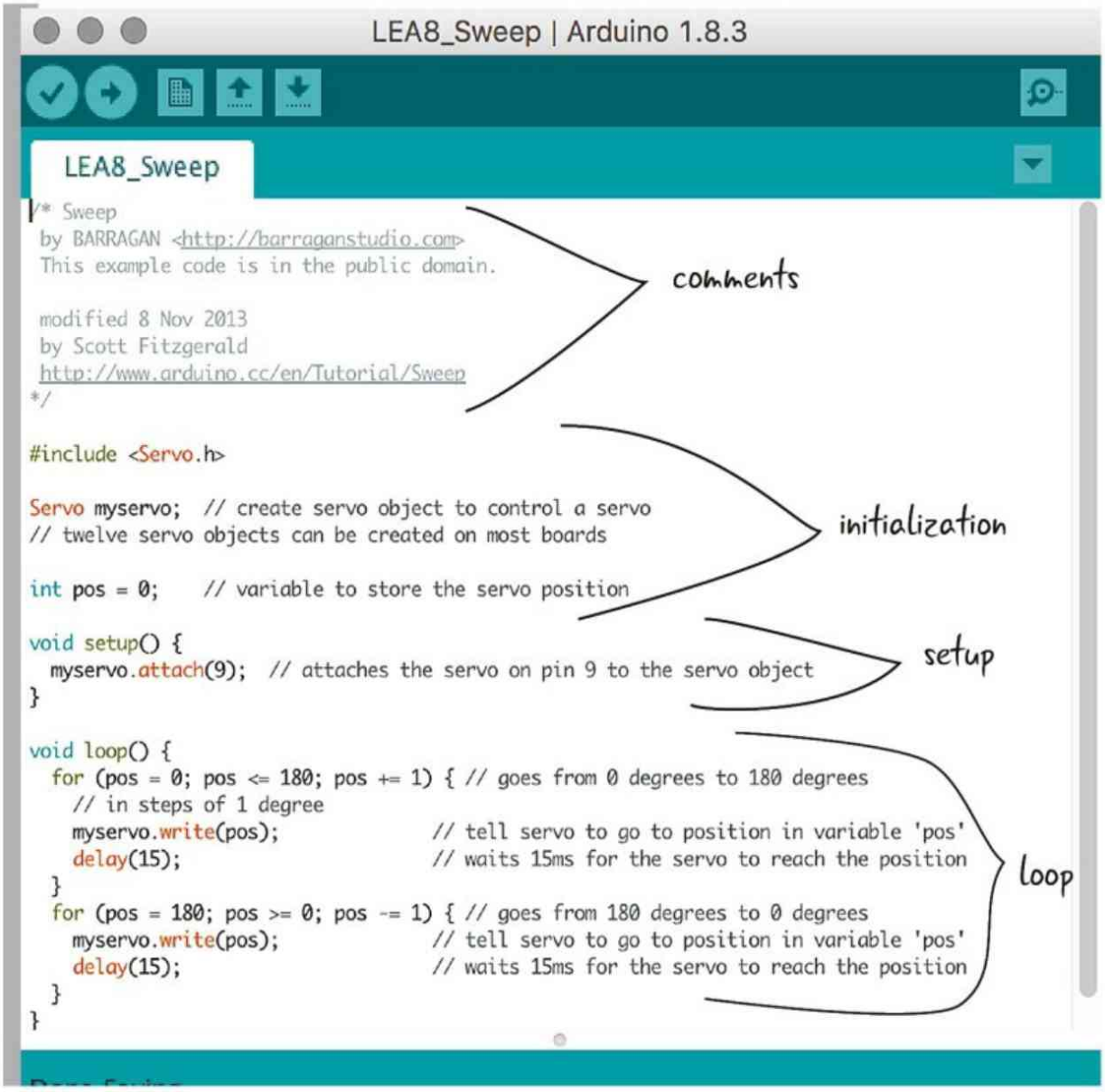


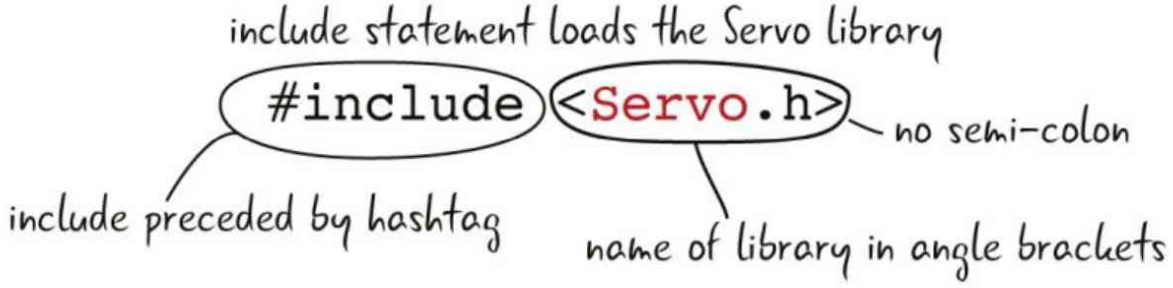
FIGURA 8.13 Resumen de LEA8_Sweep.

Inicialización

Lo primero que ves en la sección de inicialización es una línea de código que

va a añadir funcionalidad al Arduino. La sentencia "include" le dice al Arduino que cargue una *biblioteca,* con la que se ampliarán las capacidades de tu Arduino. En vez de tener que escribir todo el código tú mismo, la inclusión de bibliotecas proporciona acceso a funciones extra que han escrito otras personas y que amplían las posibilidades del Arduino.

¿Cómo añades una biblioteca? Utiliza la sentencia "include", que comienza con un # seguido de la palabra include. A continuación se abren paréntesis angulares, con el nombre de la biblioteca, en este caso Servo, y la extensión .h a continuación. Un paréntesis angular de cierre completa la declaración y no hay punto y coma en este caso.



Note **Una *biblioteca* es un conjunto de código que extiende la funcionalidad de Arduino. La biblioteca debe también incluir específicamente tu sketch para utilizarlo.**

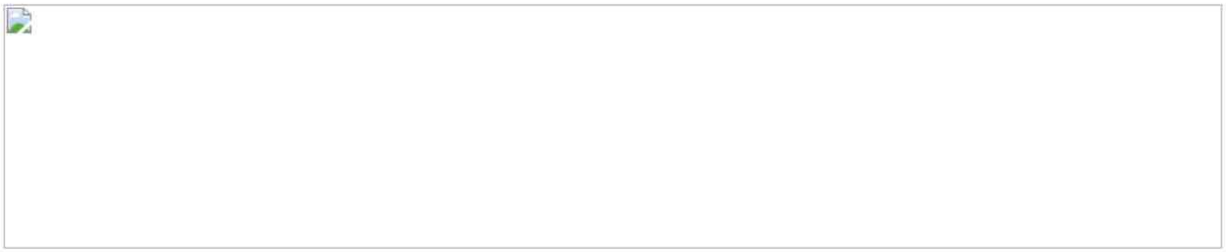
El IDE de Arduino tiene muchas bibliotecas ya cargadas, y también permite cargar nuevas bibliotecas si quieres tener acceso a ellas. Por ahora solo nos preocupa la biblioteca Servo y lo que hace.

Si observas la siguiente línea de la sección de inicialización, verás un nuevo tipo llamado Servo. Después de cargar la biblioteca Servo, puedes crear el objeto servo, que tiene funciones que le permiten controlar servomotores. Esta

línea crea el objeto servo y lo almacena en una variable llamada myservo.

```
create servo object        store it in a variable
        Servo myservo;  // create servo object to control a servo
```

No hemos comentado antes nada sobre los objetos, y una discusión a fondo sobre objetos está más allá del alcance de este libro. Piensa en un objeto como una plantilla con un conjunto de funciones y propiedades adjuntas, es decir, puedes crear varios servo objetos diferentes en el sketch basados en la plantilla. Aunque cada uno sigue la misma estructura básica, puedes modificar tanto sus propiedades como la posición.



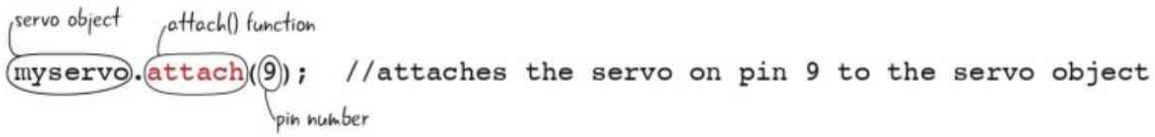
La última línea en nuestro sketch de inicialización crea una variable llamada pos, que se inicializa a 0. Esta variable se utilizará para definir la posición del servo. Si cambias este valor y lo envías de nuevo al servo, el motor se actualizará a su nueva posición. Veremos el código que cambia estos valores dentro del código loop().

```
variable named pos is declared, typed and set to 0, it will hold the position of the servo
        int pos = 0;  // variable to store the servo position
```

el interior de setup()

La sección setup() incluye solo una línea en este sketch. attach() es una nueva función disponible para el Arduino desde la librería Servo que te permite

conectar el objeto servo, al que le pusiste el nombre de myservo, a un pin del Arduino. De esta manera, siempre que te refieras a myservo estás haciendo referencia al pin al que has conectado myservo, y serás capaz de controlar el servo conectado a ese pin. En este proyecto, el servomotor se conecta al pin 9.



dentro de loop()

Este código loop() es algo diferente a lo que has visto antes, y te introducirá en el siguiente concepto de programación: el *bucle for*. Echemos un vistazo al código; luego lo desglosaremos.

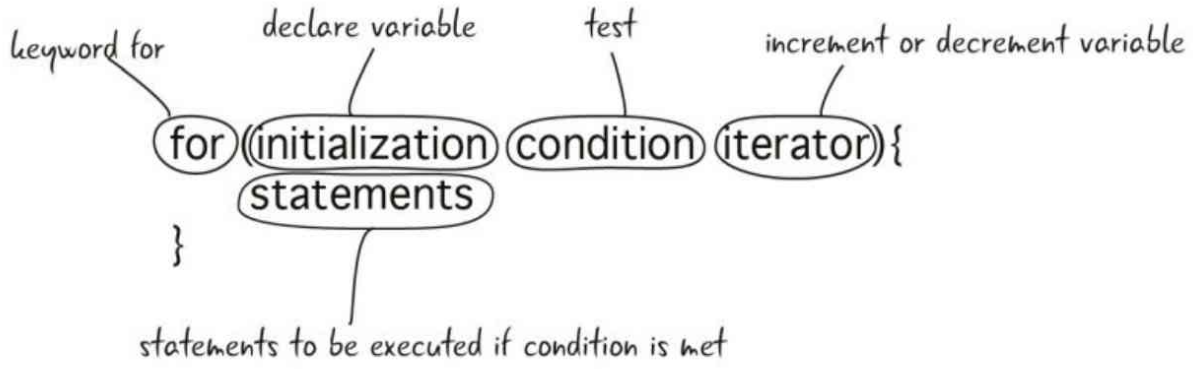
the for loop: code inside of loop()

```
for (pos = 0; pos <= 180; pos += 1) //goes from 0 degrees to 180 degrees
{                                   // in steps of 1 degree
    myservo.write(pos);     //tell servo to go to position in variable 'pos'
    delay(15);              //waits 15ms for the servo to reach the position
}
for (pos = 180; pos >= 0; pos -= 1) //goes from 180 degrees to 0 degrees
{
    myservo.write(pos);     //tell servo to go to position in variable 'pos'
    delay(15);              //waits 15ms for the servo to reach the position
}
```

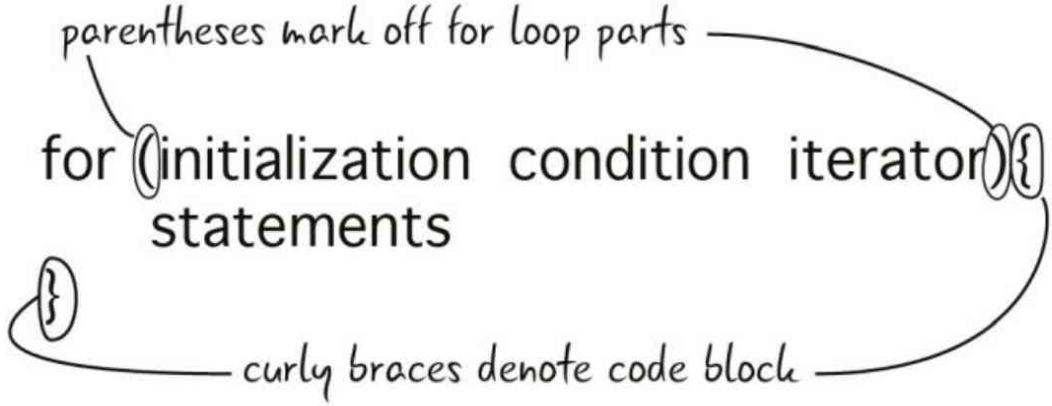
¿qué es un bucle for?

Hay ocasiones en las que puedes querer repetir algo un cierto número de veces, o hasta que se cumpla una condición en particular. El bucle for te permite repetir algo un número de veces teniendo en cuenta ciertas condiciones. En tu sketch, lo que haces es ajustar la posición del servomotor con un bucle for.

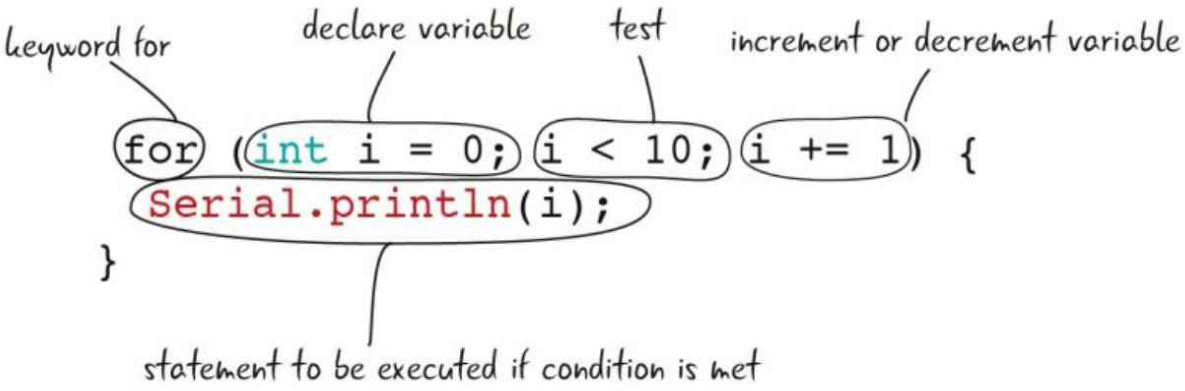
Veamos primero un ejemplo de bucle for antes de ver qué hace exactamente en tu sketch. En el lenguaje Arduino, después de la palabra clave for, el bucle for tiene tres partes: la *inicialización*, la *condición* o prueba, y el *iterador*.



Así es como se usan los paréntesis y las llaves en un bucle. Los paréntesis delimitan la sección de inicialización, condición e iterador. Las llaves delimitan el bloque de código, o las sentencias a ejecutar si la condición es verdadera.



Ahora veamos un ejemplo de bucle for en la sintaxis del lenguaje Arduino. Este bucle for presenta en el monitor serie los enteros del 0 al 9.



PIENSA SOBRE ESTO...

¿Qué diferencia habría en su funcionamiento si pones el bucle `for` en `setup()`? ¿Qué ocurre dentro de `loop()`?

¿Cómo funciona el bucle for?

¿En qué orden se ejecutan las partes del bucle for? Echemos un vistazo a la

figura 8.14.

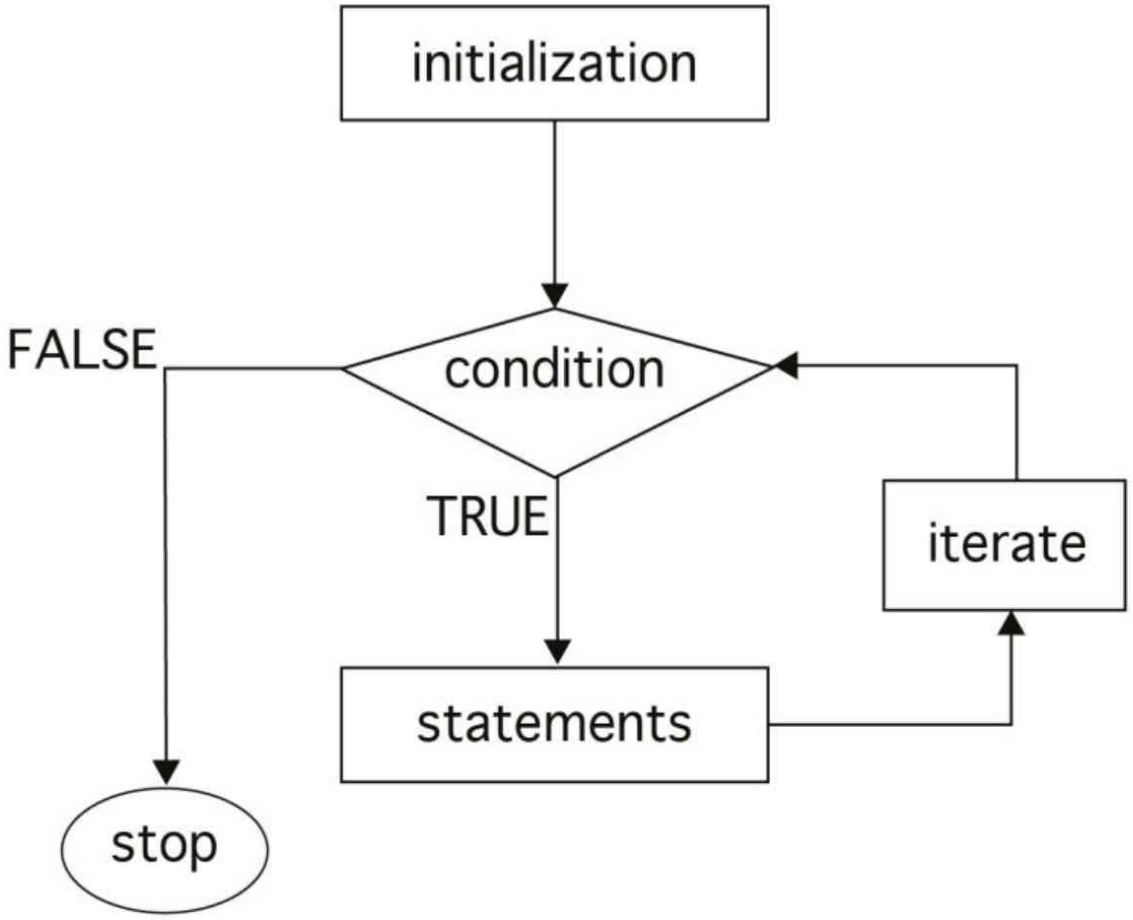


FIGURA 8.14 Diagrama de flujo del bucle `for`.

Lo primero que ocurre en nuestro bucle for es la inicialización (figura 8.15). Crea una variable temporal para contar cuántas veces se ejecuta el bucle for. El bucle for ocurrirá un cierto número de veces.

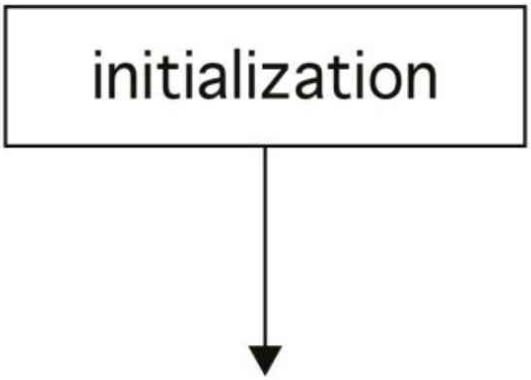


FIGURA 8.15 La inicialiación es el primer paso.

¿Cuántas veces ocurrirá el bucle for? Eso depende de la siguiente parte de tu bucle for: la prueba, que se muestra en la figura 8.16. Si la condición en la prueba es verdadera, entonces se ejecutarán las declaraciones dentro de las llaves. Si el resultado de la prueba no es verdadero, el bucle for dejará de ejecutarse. Hablaremos muy pronto sobre los diferentes tipos de condiciones que crearás para la prueba.

FIGURA 8.16 Se prueba la condición.

Si la prueba se evalúa como verdadera (figura 8.17), se ejecutan las declaraciones/in-strucciones. Entonces, el valor se itera. Esto significa a menudo que aumentas la cuenta de la variable en uno, pero puedes también alterar la variable de muchas otras formas para continuar definiendo el bucle for. Una vez has iterado la variable, el bucle for vuelve a ejecutar la condición. Si la prueba continúa siendo verdadera, las sentencias dentro de las llaves se ejecutan de nuevo, y el valor se itera.

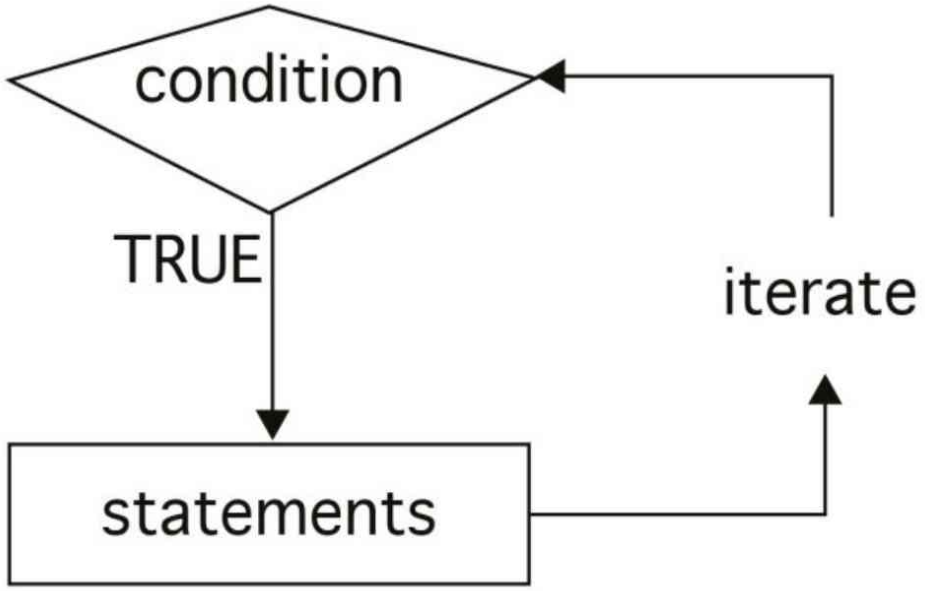


FIGURA 8.17 Si la condición es verdadera, se ejecutan las instrucciones y se itera.

Solamente en el caso de que la prueba sea falsa, como se muestra en la figura 8.18, el bucle for termina.

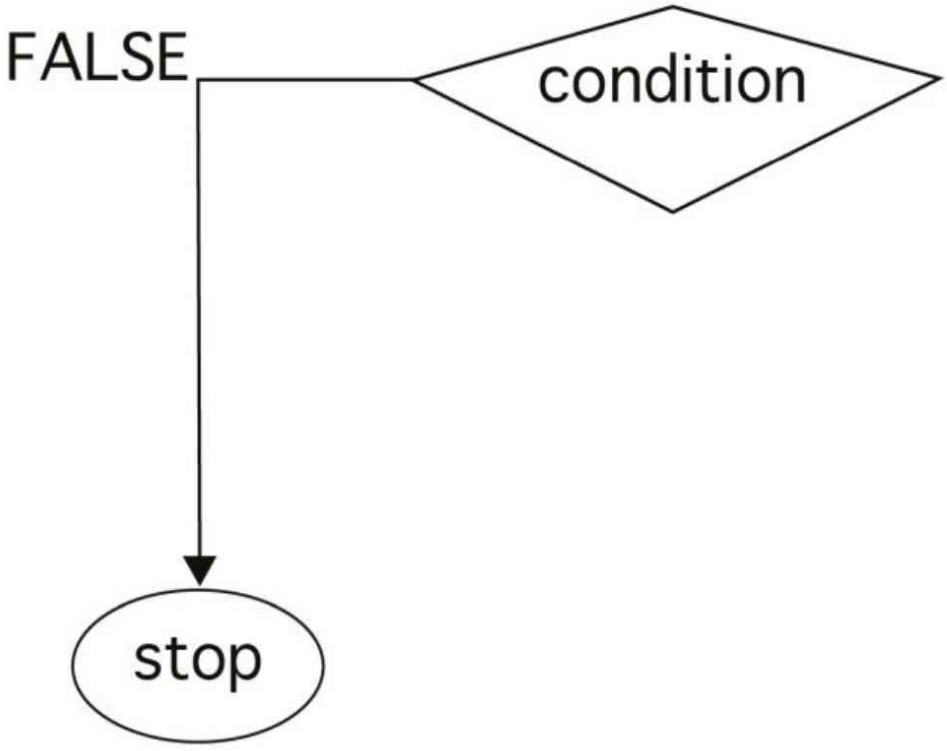


FIGURA 8.18 El bucle `for` termina cuando la prueba se valora como falsa.

Veamos de nuevo el ciclo con el código de nuestro ejemplo (figura 8.19).

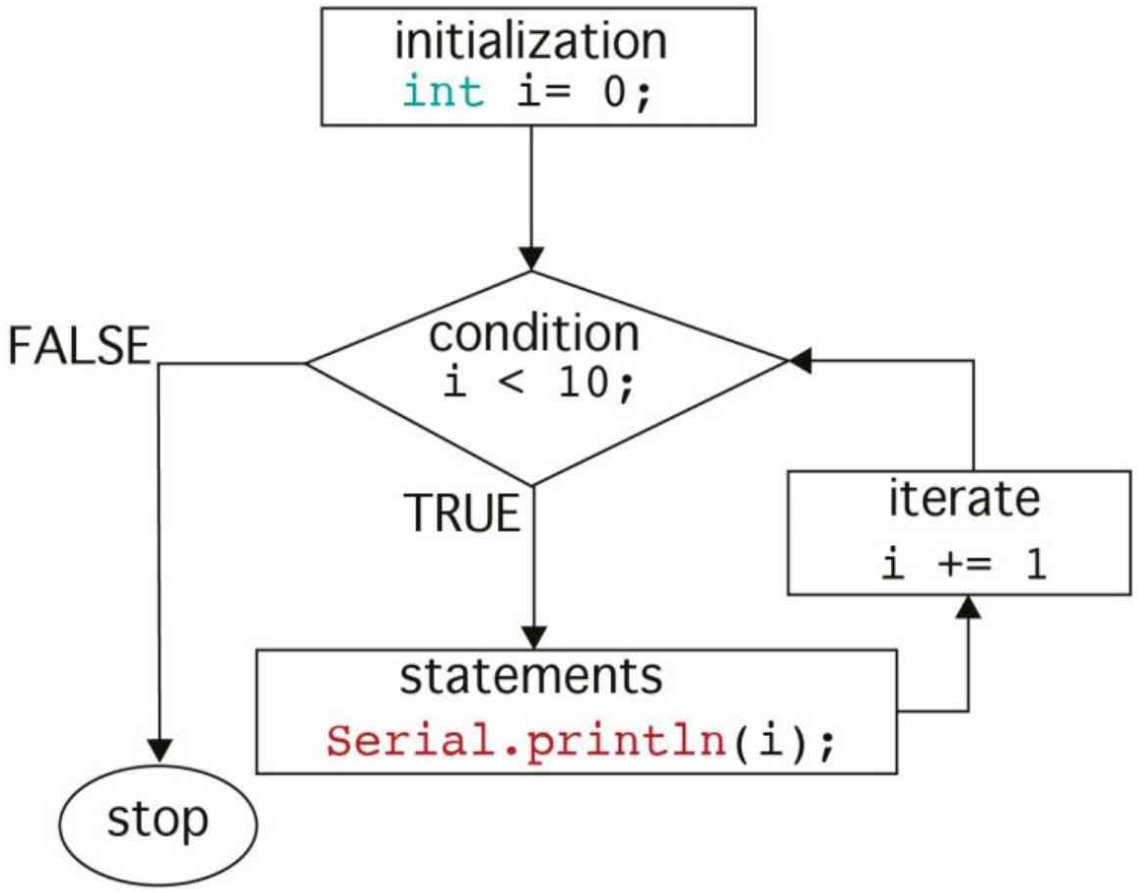


FIGURA 8.19 Diagrama de flujo del bucle `for` con el código.

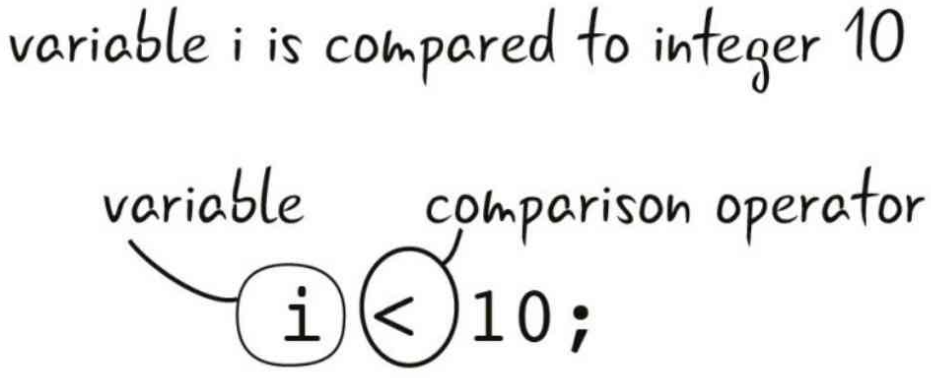
Antes de seguir adelante, veamos más de cerca la sección de la condición, o prueba, del bucle for, que requiere analizar la idea de *operador*.

Operadores

Un *operador* es una evaluación matemática o lógica de valores que son útiles para valorar la prueba en el bucle for. Las operaciones de aritmética básica —suma, resta, multiplicación y división— son ejemplos de operadores. Hay otros tipos diferentes de operadores.

Operadores de comparación

Echemos un vistazo más de cerca a la prueba. Ves la variable i, luego el símbolo < y a continuación el número 10. ¿Qué quiere decir esto? En castellano, esto significa *¿es la variable i menor que el número entero 10?* Sabemos que la variable i se puso a 0 cuando se inicializó el bucle for. El símbolo < quiere decir "es menor que"; comprueba el resultado de comparar el valor de i con el número 10. En este contexto se lo denomina *operador de comparación*. Los operadores de comparación se utilizan en expresiones lógicas, como es el caso de la prueba en un bucle for, o en una expresión condicional, para determinar si una expresión es verdadera o falsa.



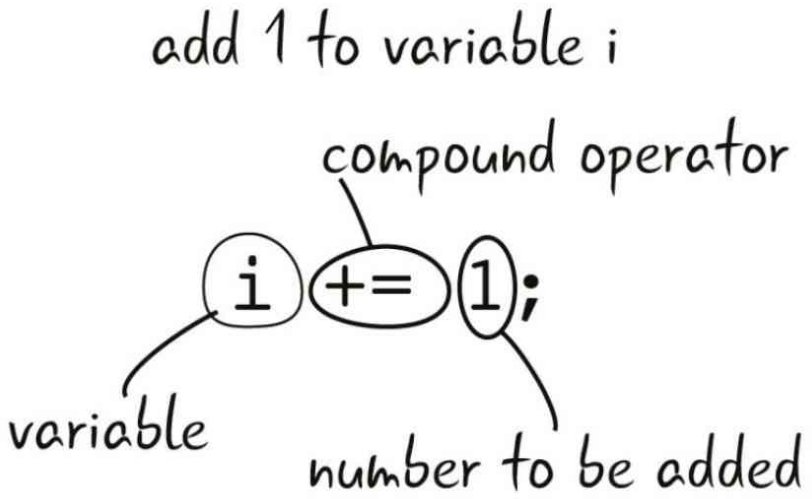
La tabla 8.1 muestra los operadores de comparación más utilizados en el lenguaje Arduino.

Tabla 8.1:**Operadores lógicos de comparación**

operador de comparación	qué significa	Ejemplo	Qué significa el ejemplo
>	Mayor que	x > 0	x es mayor que 0
<	Menor que	x < 10	x es menor que 10
>=	Mayor o igual que	x >= 0	x es mayor o igual que 0
<=	Menor o igual que	x <= 10	x es menor o igual que 10
==	Es igual a	x == 10	x es igual a 10
!=	No es igual a	x != 10	x no es igual a 10

Operadores compuestos

Ya que estamos con el tema de los operadores, observa que se utiliza un tipo de operador distinto en la sección del iterador de nuestro bucle for. A la variable i le sigue +=, seguido de 1. Esto significa que estamos añadiendo 1 al valor de la variable. Los símbolos += indican que queremos añadir lo que está en el lado derecho a la variable que está en el lado izquierdo. En nuestro ejemplo, esto significa añadir 1 a la variable i.



A este tipo de operador se le llama *operador compuesto*. Los operadores compuestos realizan una operación matemática de algún tipo. La tabla 8.2 lista los operadores compuestos comúnmente usados en el lenguaje Arduino. En cada uno de los ejemplos de la tabla 8.2, x se fija inicialmente en 10.

Tabla 8.2: **Resultados al utilizar un operador compuesto cuando x es inicialmente igual a 10**

operador compuesto	qué significa	Ejemplo	qué significa el ejemplo
++	Suma 1	x++	x es ahora igual a 11
--	Resta 1	x--	x es ahora igual a 9
+=	Suma el valor de la derecha al valor de la izquierda	x += 2	x es ahora igual a 12
-=	Resta el valor de la derecha al valor de la izquierda	x -= 2	x es ahora igual a 8

*=	Multiplica el valor de la izquierda por el valor de la derecha	x *= 5	x es ahora igual a 50
/=	Divide el valor de la izquierda por el valor de la derecha	x /= 2	x es ahora igual a 5

el bucle for en el Sketch

Entonces, ¿cómo puedes utilizar los bucles for para ayudarte a mover el servo? Echemos un vistazo al primer bucle for de nuestro código. Si lo desglosamos, en la inicialización se establece la variable pos a 0. La condición verifica si la variable pos es menor de 180, y si es así, se añade un 1 (iterado) a pos. Aquí no has tenido que usar int para indicar el tipo de pos, porque pos se había declarado en la sección de inicialización.

```
for (pos = 0; pos <= 180; pos += 1) //goes from 0 degrees to 180 degrees
{                                   // in steps of 1 degree
    myservo.write(pos);     //tell servo to go to position in variable 'pos'
    delay(15);              //waits 15ms for the servo to reach the position
}
```

¿Qué instrucciones se ejecutan cada vez que se pasa por el bucle for? Mientras el valor de i sea inferior a 180, el Arduino escribirá el valor de pos al motor, lo que moverá el servomotor a alguna posición entre 0 y 180 grados. Después de que el Arduino haya escrito esa posición, hay un retardo de 15 milisegundos.

Puesto que el bucle for continúa para cada valor entre 0 y 180, el servomotor se moverá desde 0 hasta 180 grados en el primer bucle for. Esto tardará unos segundos (hay una pausa muy corta entre cada movimiento), pero el movimiento en general se apreciará relativamente suave. Si eliminas o ajustas la duración del retardo, cambiará la suavidad del movimiento.

¿Por qué el bucle for cuenta entre 0 y 180? Porque eso representa el margen del movimiento en el que se puede mover el eje de tu servomotor estándar. Piensa que el margen es de 0 a 180 grados.

El segundo bucle for en este sketch funciona de la misma manera. En lugar de empezar en 0, empieza con la variable pos igual a 180. ¿Qué es 180? El segundo bucle necesita empezar con la última posición del primer bucle, que

es también la posición final del servo.

```
for (pos = 180; pos >= 0; pos -= 1) //goes from 180 degrees to 0 degrees
{
    myservo.write(pos);     //tell servo to go to position in variable 'pos'
    delay(15);              //waits 15ms for the servo to reach the position
}
```

Además de tener un punto de partida diferente, este segundo bucle for también disminuye una unidad cada vez que se pasa por él. De esta forma, el servomotor arranca en 180 grados y gira lentamente hacia los 0 grados.

Una vez que el segundo bucle for ha terminado, la función completa loop() de Arduino ha terminado también. El Arduino volverá al principio del loop() y repetirá los pasos, como has visto con otras funciones de loop().

¿PREGUNTAS?

P: Los bucles `for` se utilizan en otros lenguajes de programación ¿no es así?

R: Sí, los bucles `for` se utilizan normalmente en muchos y diferentes lenguajes de programación. A menudo se utilizan cuando tiene que suceder algo un cierto número de veces.

P: ¿Hay otros tipos de bucles además del bucle `for` en el lenguaje de programación Arduino?

R: Sí, están los bucles `while` y los bucles `do`. Puedes ampliar la información en: arduino.cc/en/Reference/While y en arduino.cc/en/Reference/DoWhile.

PIENSA SOBRE ESTO...

Ahora que sabes cómo funciona un servomotor, piensa en algunos usos del mismo. ¿Qué tipos de dispositivos has visto que utilicen servomotores? ¿Qué proyectos te gustaría montar en los que se produjera este tipo de movimiento?

añadir Interactividad: ondear la bandera

Ahora tienes una comprensión básica de cómo funcionan los servomotores con el código del Arduino, así que vamos a hacer que el circuito con el servo sea interactivo. En lugar de que el Arduino mueva el servo a un ritmo constante y de forma permanente, el siguiente sketch utiliza información del potenciómetro para posicionar el eje del servomotor. Cuando gires el mando, el eje del servomotor se moverá.

Utilizarás el sketch Knob, que también está incluido en los ejemplos de servomotores en el IDE. Antes de subir el sketch, sin embargo, vas a añadir el potenciómetro al circuito.

Adición del Potenciómetro paso a paso

La adición de un potenciómetro al circuito te permite controlar cómo ondea la bandera y ajustarla a una posición precisa. Si completaste el primer circuito con el servo, puedes dejar el cable de control del servo conectado al pin 9 del Arduino, el cable de alimentación conectado al bus de alimentación, y el cable de tierra al bus de tierra. Solo añadirás el potenciómetro al circuito.

Necesitarás las siguientes piezas:

- Potenciómetro de 110 K
- Cables de puente

La figura 8.20 muestra el circuito terminado junto al esquema.

> **Tip** Como siempre, debes tener la seguridad de haber desconectado Arduino antes de hacer cualquier cambio en el circuito.

Coloca el potenciómetro en la placa de pruebas como se muestra en la figura 8.21.

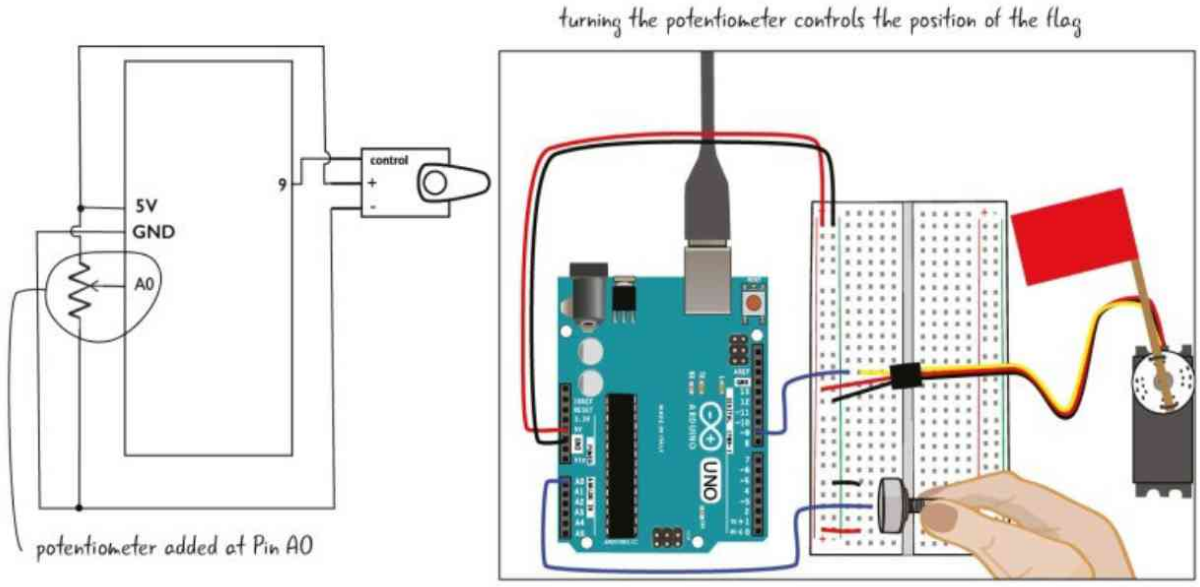


FIGURA 8.20 Circuito con el potenciómetro añadido.

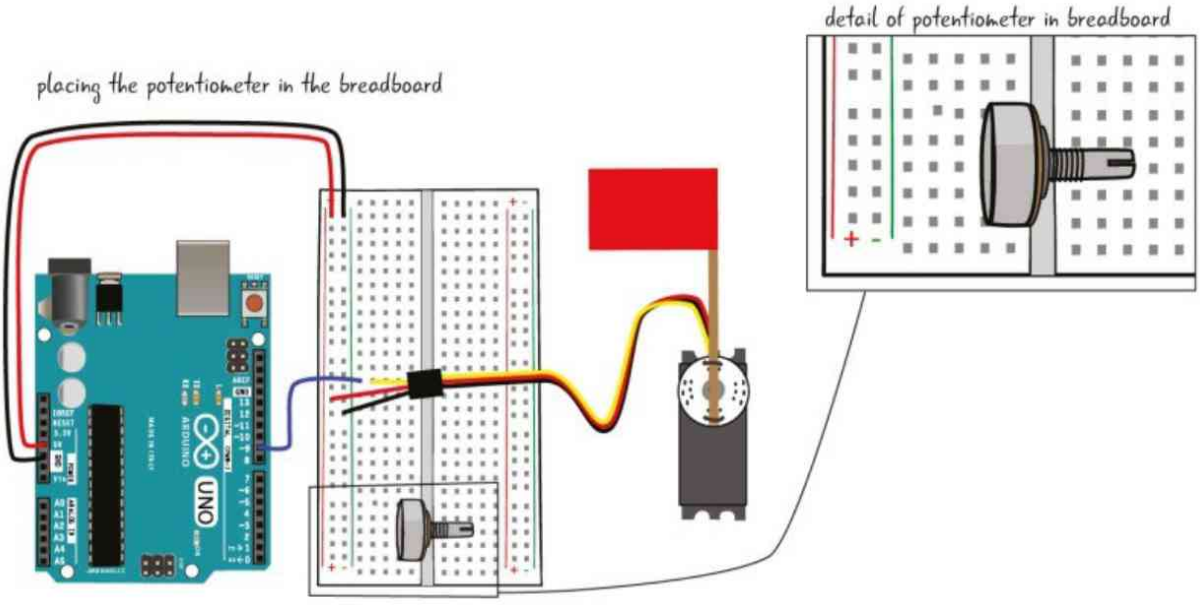


FIGURA 8.21 Adición del potenciómetro a la placa de pruebas.

Conecta un extremo del potenciómetro al bus de tierra con un puente. Conecta el otro extremo del potenciómetro al bus de alimentación con un puente. Conecta el pin intermedio del potenciómetro al pin A0, uno de los pines de entrada analógica (figura 8.22).

Ahora que has cableado el circuito, es el momento de abrir el siguiente sketch en el IDE de Arduino. Este sketch se localiza en "Archivo >

Ejemplos > Servo > Knob". Una vez abierto, lo guardas como **LEA8_Knob**.

Conecta el ordenador al Arduino. Haz clic en "Verify" para comprobar el código y después haz clic en **Subir** para cargarlo en el Arduino.

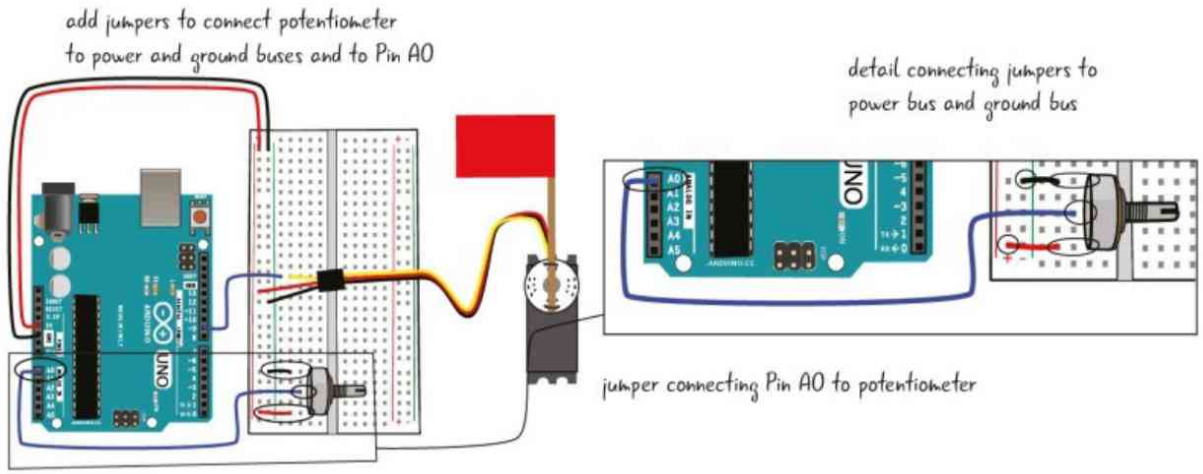


FIGURA 8.22 Adición de puentes al potenciómetro.

Ahora, cuando gires el potenciómetro, la bandera también debe girar.

¿cómo cambia el Sketch Cuando utilizamos un Potenciómetro?

Echemos un vistazo al sketch. Es similar a LEA8_Sweep sketch, con un par de diferencias importantes que veremos detenidamente.

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo

int potpin = 0;  // analog pin used to connect the potentiometer
int val;    // variable to read the value from the analog pin

void setup()
{
  myservo.attach(9);
  // attaches the servo on pin 9 to the servo object
}

void loop()
{
  // reads the value of the potentiometer (value between 0 and 1023)
  val = analogRead(potpin);
  // scale it to use it with the servo (value between 0 and 180)
  val = map(val, 0, 1023, 0, 180);
  // sets the servo position according to the scaled value
  myservo.write(val);
  // waits for the servo to get there
  delay(15);
}
```

initialization

attach servo in setup

read value, map value, write value to servo in loop

explicación de LEA8_Knob

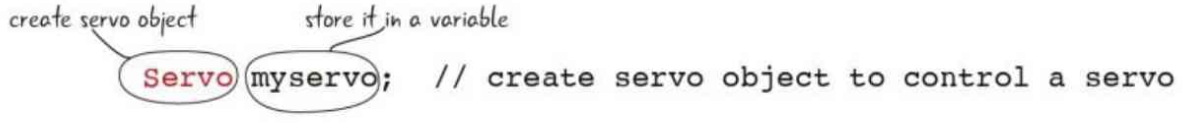
Al igual que con LEA8_Sweep, este sketch controla la posición del cuerno conectado al eje del servomotor, en función del valor que le envíe el Arduino. Sin embargo, esta vez puedes controlar cuánto gira, ya que a medida que giras el potenciómetro, cambias el valor que el Arduino recibe y envía al servo.

Inicialización

En este sketch del servo, al igual que en el anterior, lo primero que se ve en la sección de inicialización es la sentencia "include" que carga la librería Servo. Como has visto, las librerías amplían las capacidades del Arduino para realizar funciones específicas o interactuar con algunos tipos de tecnología de un modo sencillo, de manera que puedas escribir código de forma simplificada.

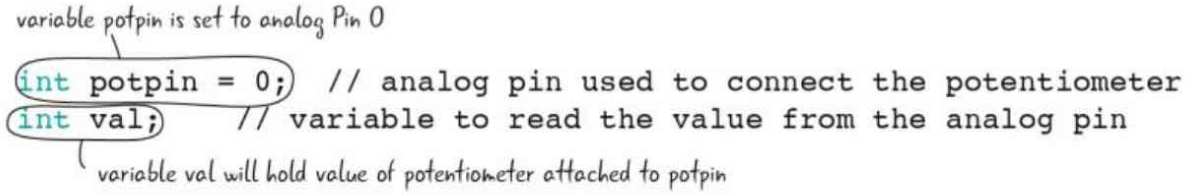
```
#include <Servo.h>
```

Si observas la segunda línea de la sección de inicialización, ves que, como en LEA8_Sweep, estás creando un nuevo servo objeto llamado myservo. Este objeto podrá acceder a las funciones de la librería Servo para comunicarse con el servomotor.



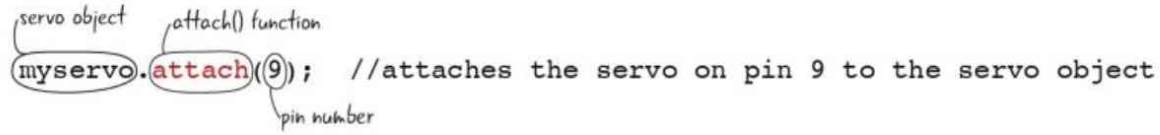
La sección de inicialización también contiene una variable para el pin

analógico al que está conectado el potenciómetro. Hemos tratado este tema en el capítulo 7; si conectas el potenciómetro al pin analógico 0 puedes tomar lecturas entre 0 y 1.023 en lugar de la lectura HIGH o LOW que obtienes de un pin digital. Finalmente, la última línea en el esquema de inicialización crea una variable llamada val, que después utilizarás para almacenar el valor que viene del potenciómetro y enviarlo al servo.



código en setup()

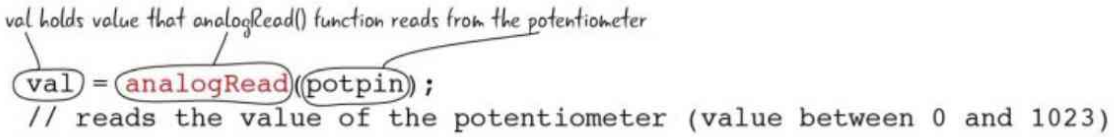
La sección setup() contiene solamente una línea para este sketch. Utiliza de nuevo attach() para conectar el objeto servo que denominaste myservo a un pin en el Arduino. De esta forma, cada vez que te refieras a myservo, estarás haciendo referencia a un pin en particular, de la misma manera que has visto con digitalWrite() y un nombre de pin. En este caso, el servo debe estar conectado al pin 9.



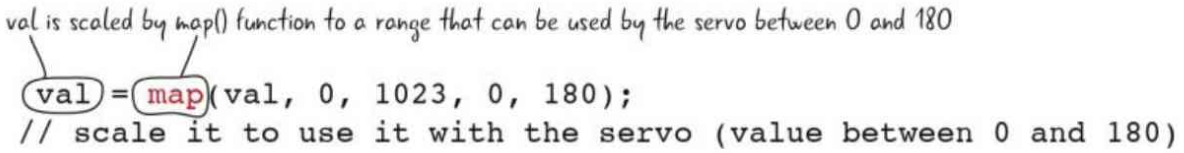
código en loop()

La sección del código loop()es similar a la que elaboraste en el capítulo 7. El primer paso es utilizar la función analogRead() para leer el valor del potenciómetro en el pin A0 y almacenarlo en el valor de la variable val. Esto

ajustará val a un valor entre 0 y 1.023, que como sabes, es el rango de valores posibles en un pin analógico.



A continuación, utiliza la función map() para ajustar el valor de la lectura del potenciómetro y hacerla coincidir con los grados de movimiento del eje del servomotor. Dado que el servomotor puede moverse 180 grados, asignarás el valor de 0 a 180 grados. De esta manera, cuando se envía el valor al servomotor, ya estará en un valor dado en grados. Este nuevo valor asignado se guarda de nuevo en la variable val.



El siguiente paso es escribir la variable escalada val al servo conectado al pin 9 usando la función write()del objeto servo. Vale la pena hacer de nuevo hincapié aquí en que el servo no se moverá el número de grados adicionales que indica val, sino que se moverá el número de grados de val a partir de 0. Por ejemplo, si val es igual a 90, siempre moverá el eje del servo al punto medio.

```
// sets the servo position according to the scaled value
myservo.write(val);
```

La última línea en el código loop() retarda el programa de Arduino durante

15 milisegundos. Este corto periodo de retardo permitirá que el servo se mueva a la posición correcta, ya que el movimiento no es instantáneo. También le dará al Arduino un poco más de tiempo entre las lecturas del potenciómetro para asegurar una lectura más precisa.

```
// waits for the servo to get there
 delay(15);
```

dos banderas ondeando: MONTA otro ServoMotor

Vas a añadir otro servomotor al circuito. Utilizarás la información del potenciómetro para ajustar la posición de ambos servos. Vas a montar un sistema de señales con banderas.

En el sketch de este proyecto, aprenderás a escribir una función personalizada, también aprenderás algo más sobre el uso de la lógica en las sentencias condicionales.

Si montaste el circuito del sketch LEA8_Knob, este circuito será casi idéntico, lo único nuevo es la adición del segundo servomotor.

Necesitarás lo siguiente:

- ▨ El servomotor
- ▨ Puentes
- ▨ Un agitador de café o una tira de cartón
- ▨ Papel de color
- ▨ Una cinta

La figura 8.23 muestra el esquema del circuito, así como un dibujo del proyecto terminado.

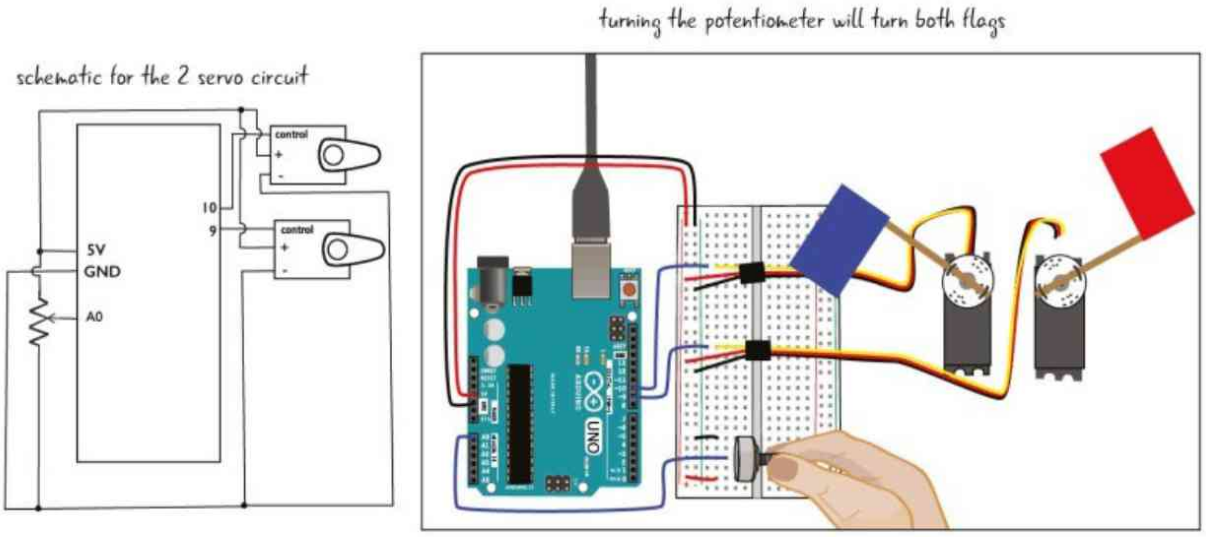


FIGURA 8.23 Circuito con dos servos y su esquema.

Figura 8.24:**Fijación de la bandera al cuerno del servo.**

Primero conecta el agitador de madera con la bandera de papel al cuerno del servo, como hiciste con el primer servomotor (figura 8.24).

Conecta los puentes al conector del servo (figura 8.25), de modo que coincidan con los colores de la alimentación energía y de tierra, y conecta el cable de control.

Primero conecta el agitador de madera con la bandera de papel al cuerno del servo, como hiciste con el primer servomotor (figura 8.24).

Conecta los puentes al conector del servomotor (figura 8.25), de acuerdo con los colores de alimentación y tierra, y el cable de control.

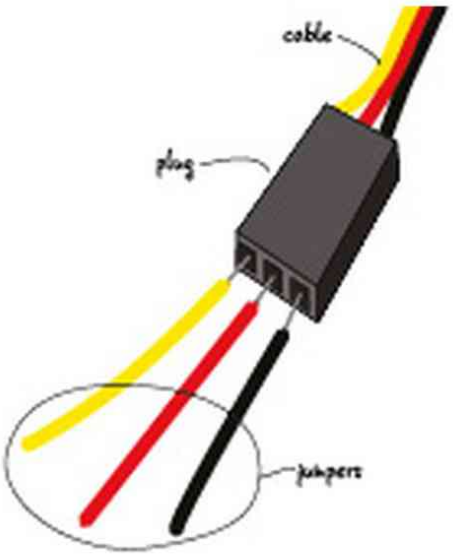


Figura 8.25:**Conexión de los puentes al conector**

Ahora conecta los puentes a la placa de control, como se muestra en la

figura 8.26. Como hiciste antes, conecta el puente prolongación del cable de alimentación al bus de alimentación y el puente prolongación de tierra al bus de tierra. El cable de control se conecta a una fila de puntos de contacto. Finalmente, conecta el pin 10 a la misma fila de puntos de contacto que el puente del cable de control.

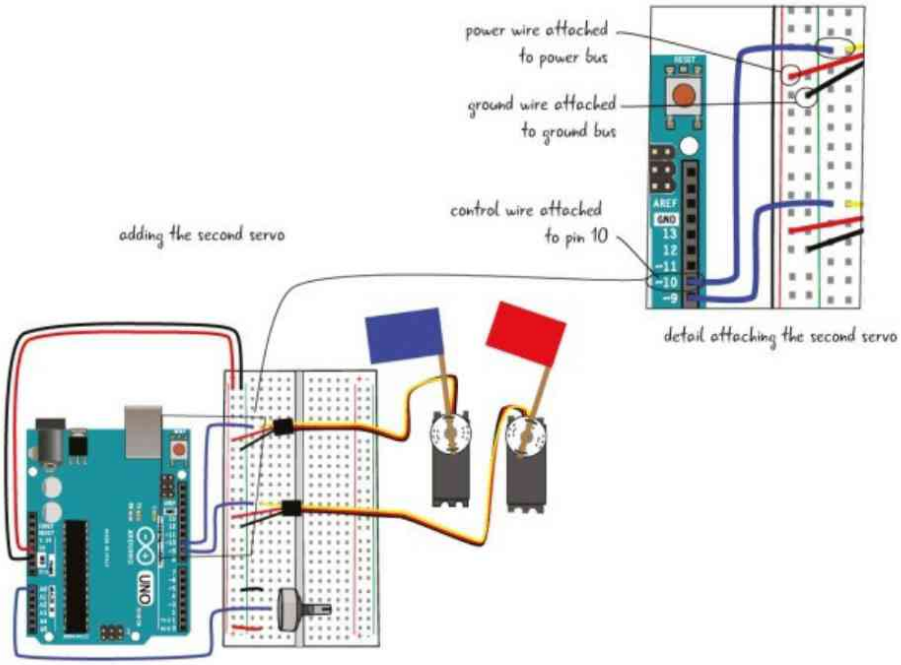


FIGURA 8.26 Adición del segundo servo.

Antes de conectar el Arduino al ordenador, debes hacer algunos ajustes en el código. Veamos el sketch.

LEA8_2_servos, primer contacto

Guarda LEA8_Knob como **LEA8_2_servos**. Vas a adaptar el código. En el sketch LEA8_2_servos, la sección de inicialización es similar a los otros sketches, y lo mismo ocurre con setup(), pero fíjate en que hay algo nuevo en loop(), que explicaremos.

La figura 8.27 es una primera vista del código.

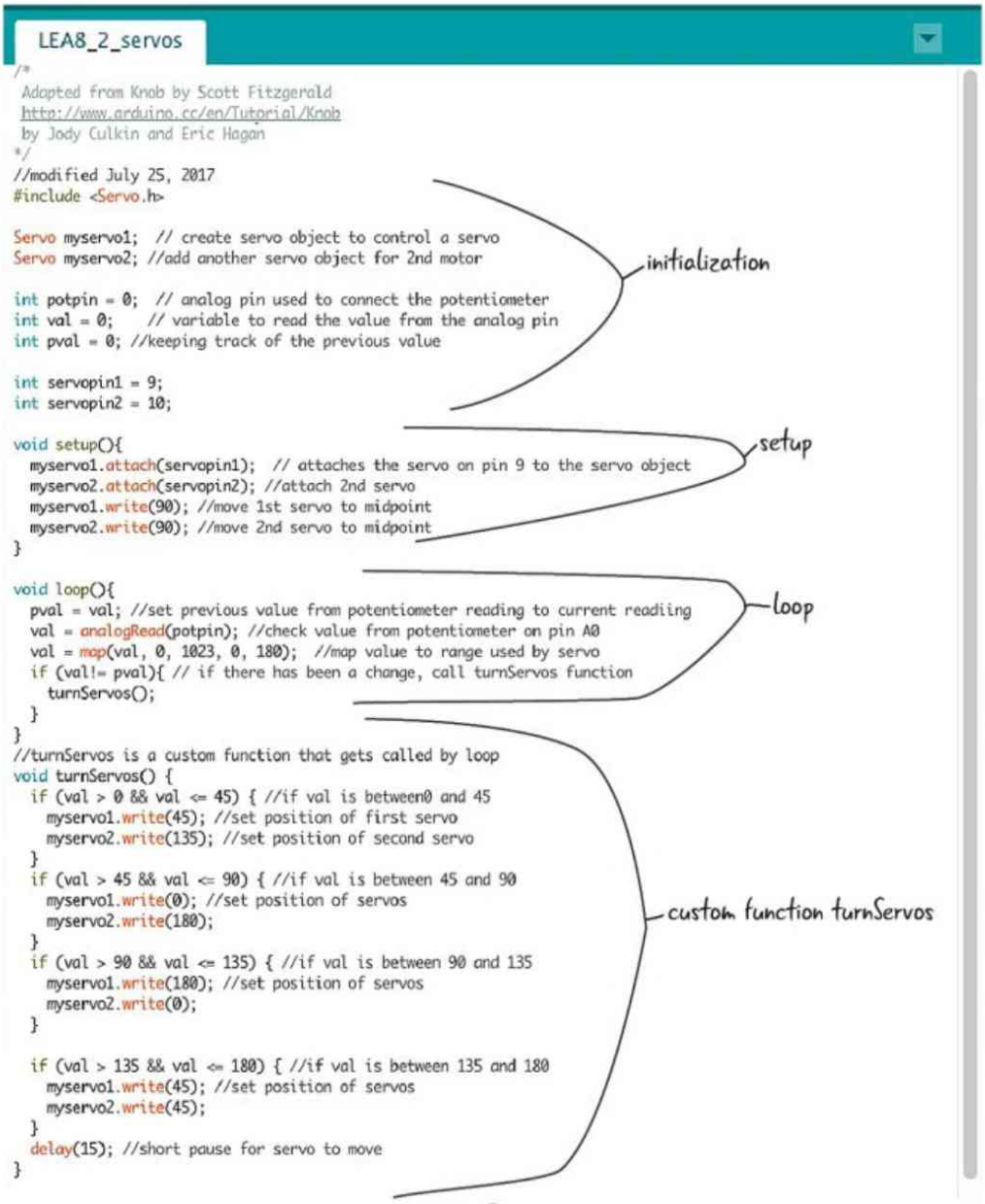


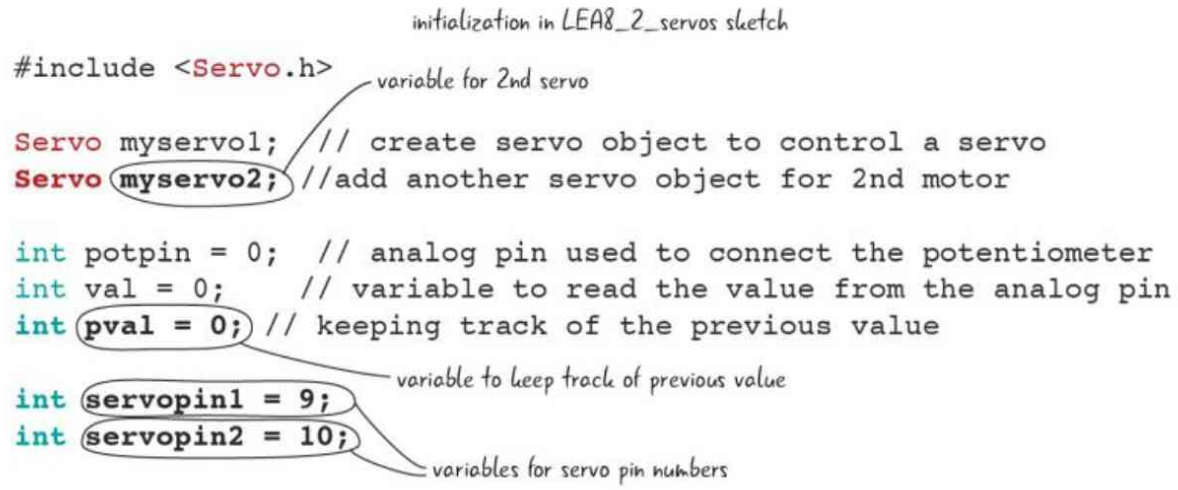
FIGURA 8.27 El sketch LEA8_2_servos comentado.

Al desglosar el código se han eliminado algunos comentarios para hacerlo

más legible.

Inicialización

Como hemos dicho, la sección de inicialización es bastante similar a la sección de LEA8_Knob. Hay un par de cosas nuevas: incluiste una variable para el segundo objeto servo, y almacenaste en las variables los números de pin a los que se conectan los servos. También añadiste una variable (pval) que almacena el valor previo.



la función setup() en LEA8_2_servos

La función setup()tiene un par de cambios en relación con el sketch LEA8_-Knob. Enlazas los objetos servo a los pines, usando las variables servopin1 y servopin2. Entonces utilizas la función de la librería Servo para mover el primer servo al punto medio y a continuación mueves el segundo servo al punto medio utilizando la función write() de nuevo.

code in setup()

```
void setup(){
  myservo1.attach(servopin1);
    // attaches the servo on pin 9 to the servo object
  myservo2.attach(servopin2);
    //attach 2nd servo
  myservo1.write(90);
    //move 1st servo to midpoint
  myservo2.write(90);
    //move 2nd servo to midpoint
}
```

attaches servo on pin 9 to servo object

attaches servo on pin 10 to servo object

set both servos to midpoint

resumen del código loop()

En el código loop(), la primera línea guarda el valor anterior leído del potenciómetro en la variable pval. A continuación lees el valor actual del potenciómetro y lo asignas. Finalmente, utilizas una sentencia condicional para comprobar si val no es igual a pval, en otras palabras, comprobar si ha habido un cambio en la lectura del pin. Si ha habido un cambio, llama a la función turnServos().

Notarás algunas cosas que no has visto antes en este código. Vamos a desglosarlo.

loop() code

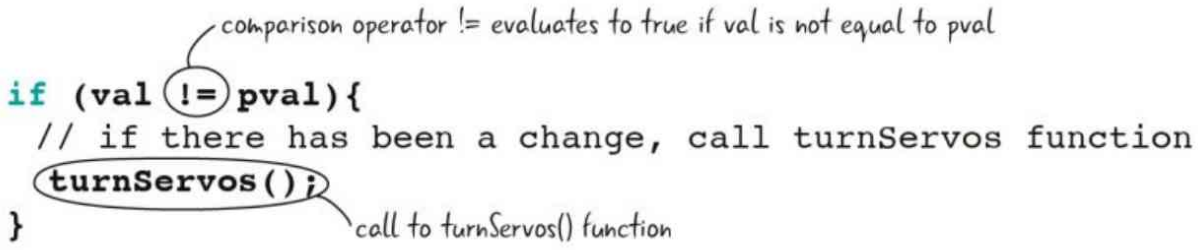
```
void loop(){
  pval = val;
    //set previous value from potentiometer reading to current reading
  val = analogRead(potpin);
    //check value from potentiometer on pin A0
  val = map(val, 0, 1023, 0, 180);
    //map value to range used by servo
  if (val!=pval){
    // if there has been a change, call turnServos function
    turnServos();
  }
}
```

store previous value

check if value of potentiometer has changed

if changed, call function

Utilizas una sentencia condicional para comparar el valor anterior leído del potenciómetro con el nuevo valor que has leído. Este condicional utiliza un *operador de comparación*: el símbolo !=. Este operador se utiliza para comparar dos valores, y si un valor no es igual al otro, se evalúa a verdadero. (La tabla 8.1 contiene estos operadores de comparación y lo que significa cada uno).



Si se ha producido un cambio, se llama a la función turnServos(), que es una *función personalizada*, el tema de nuestra próxima sección.

Creación de una función personalizada

Hemos introducido otro concepto nuevo en el código para los servos actualizado: las funciones personalizadas. ¿Por qué querrías hacer tus propias funciones? Primero repasemos rápidamente lo que es una función.

> **Note** Una *función* es un bloque de código que realiza una acción específica o una serie de acciones que se pueden utilizar una y otra vez.

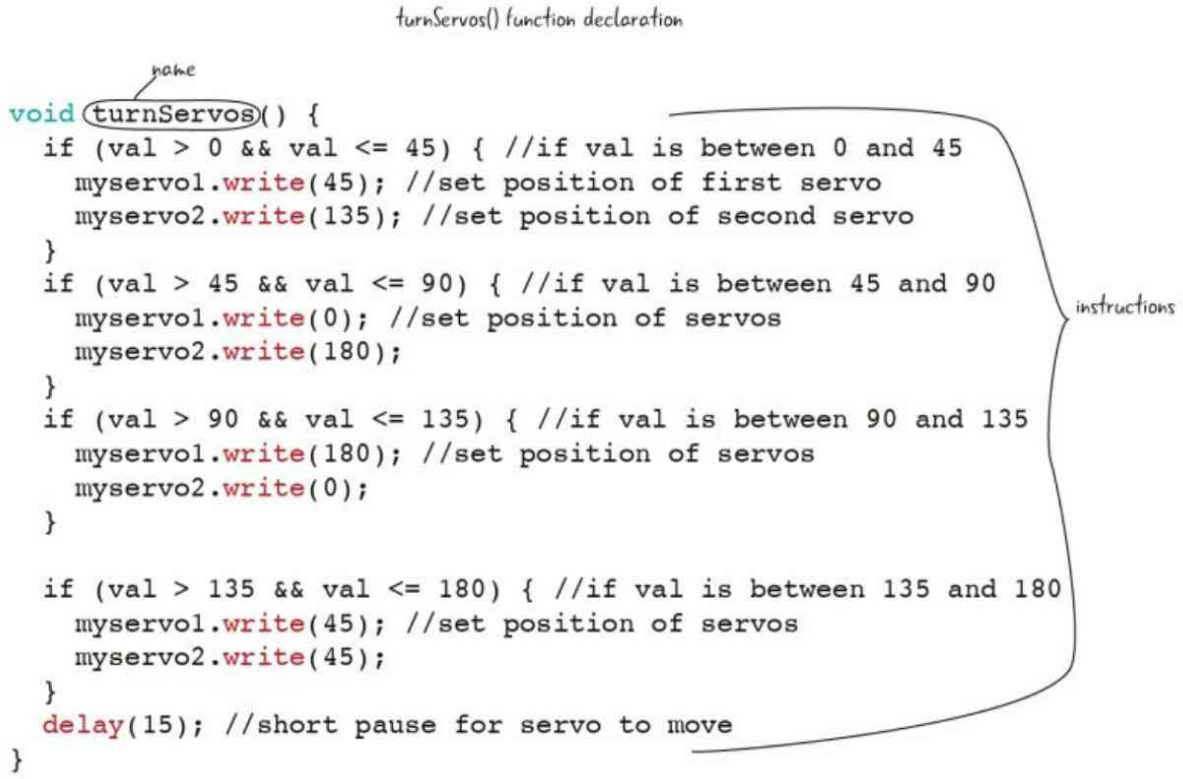
Has usado muchas funciones de Arduino: delay(), digitalWrite() y analogRead() son funciones que se utilizan para realizar algunas tareas específicas con el Uno. Has escrito la mayor parte del código dentro de las funciones setup()y loop() para los sketches.

¿Cuál es la ventaja de escribir tus propias funciones? Puedes agrupar acciones fuera del código loop()y llamar a esta nueva función solo cuando

quieras que estas acciones ocurran. Esto hace que el código sea más legible y más fácil de comprender.

¿Qué hace la línea de código turnServos();? Es la llamada a la función personalizada que has escrito. Como has visto, se le llamará si ha habido un cambio en la variable val, es decir, si alguien ha girado el potenciómetro. turnServos() solo ocurrirá cuando el valor del potenciómetro haya cambiado, y no cada vez que el Arduino pase por la función loop().

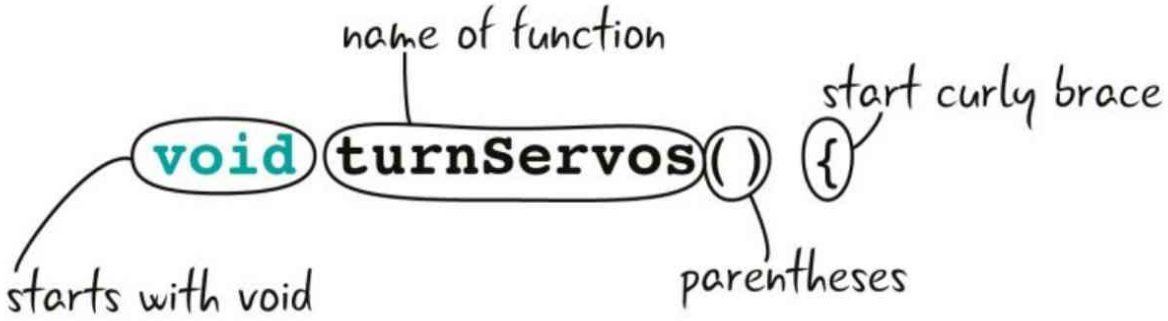
¿Cómo es tu función turnServos()? Primero empieza con void. A esto le sigue turnServos, que es el nombre de la función, seguido de paréntesis y una llave de apertura. Las instrucciones que se ejecutarán cuando se llame a turnServos() son las que van a continuación, y la última línea contiene solamente una llave de cierre.




```
void turnServos() {
  if (val > 0 && val <= 45) { //if val is between 0 and 45
    myservo1.write(45); //set position of first servo
    myservo2.write(135); //set position of second servo
  }
  if (val > 45 && val <= 90) { //if val is between 45 and 90
    myservo1.write(0); //set position of servos
    myservo2.write(180);
  }
  if (val > 90 && val <= 135) { //if val is between 90 and 135
    myservo1.write(180); //set position of servos
    myservo2.write(0);
  }

  if (val > 135 && val <= 180) { //if val is between 135 and 180
    myservo1.write(45); //set position of servos
    myservo2.write(45);
  }
  delay(15); //short pause for servo to move
}
```

A la acción de crear una función personalizada se la llama *declarar una función*, y sigue ciertas reglas en el lenguaje de programación Arduino.



Note Hay un par de reglas sencillas para para dar nombres a funciones. Las funciones deben empezar con una letra y sus nombres no deben ser palabras reservadas a Arduino. Y es conveniente que el nombre de la función indique claramente qué es lo que va a hacer exactamente.

Note ¿Por qué la función comienza con `void`? Tiene que ver con cómo operan algunas funciones; en algunas funciones puedes ver `int` o `string`, por ejemplo. Está más allá del alcance de este libro explorar este concepto, pero puedes encontrar una explicación en el sitio de Arduino: arduino.cc/en/Reference/FunctionDeclaration.

Lo que ves a continuación es un conjunto de paréntesis. Algunas funciones tienen parámetros o información que pasarán a ser argumentos de la misma cuando sea llamada. Estos se colocan dentro de los paréntesis. Como turnServos() no tiene ningún parámetro, los paréntesis están vacíos. A los paréntesis les sigue una llave de apertura, el signo que has usado antes para marcar un bloque de código.

Esta exposición sobre la declaración de una función te resultará familiar, porque ya has visto algunas de estas convenciones antes. ¿Dónde? ¡En

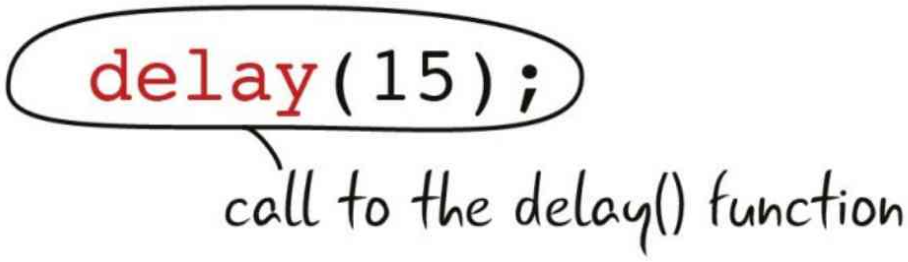
setup()y en loop()! La diferencia es que ahora eres tú el que está creando la función y dándole un nombre.

Puedes escribir tus propias funciones en cualquier momento, y tus funciones pueden incorporar cualquier código compatible con Arduino.

llamada a una función personalizada

Cuando deseas invocar una función, *llama a esa función*. La llamada a tu función personalizada está dentro de loop(). La llamada es simplemente el nombre de la función seguido de paréntesis y de punto y coma.

Has estado llamando a las funciones incorporadas a Arduino desde que subiste el primer sketch. Las funciones que has usado tienen normalmente parámetros, así que introdujiste argumentos dentro de los paréntesis. Cuando en su momento llamaste a la función delay(), por ejemplo, introdujiste la cantidad de tiempo en milisegundos que querías que durase el retardo.

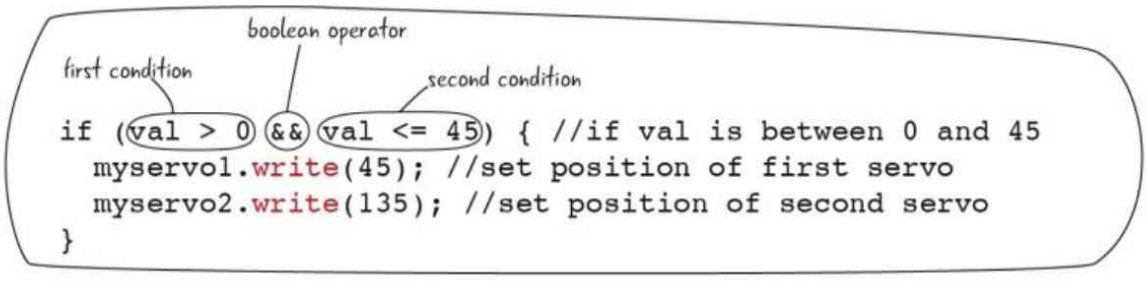


Las funciones personalizadas pueden ser muy potentes, ya que te permiten ampliar la funcionalidad de los sketches. La exposición que hacemos aquí es limitada, tiene el cometido de proporcionar una introducción al concepto y a las reglas generales para crearlas. Sin duda, explorarás muchas más cosas por tu cuenta.

dentro de turnServos()

Sabes que turnServos() se va a encargar de mover los motores, pero ¿cómo? Se trata de colocar ambas banderas siguiendo un patrón basado en hasta donde se gira el potenciómetro; a veces están enfrentadas y a veces se sitúan en paralelo. El código consiste en una serie de sentencias condicionales. Veamos de cerca la primera.

En este condicional, se están probando dos condiciones. ¿Es val mayor que 0 y es val también menor o igual que 45? El valor de val debe ser un número entre 0 y 45 para las instrucciones que se ejecutan dentro de la sentencia if.



Los símbolos && son un ejemplo de *operador booleano*. En este caso, tanto la primera como la segunda condición deben ser ciertas para que las posiciones del servo se puedan ajustar.

Operadores booleanos

Los operadores booleanos permiten realizar evaluaciones complejas para intentar decidir qué medidas se deben tomar. La tabla 8.3 enumera los

operadores booleanos y explica lo que significan, e incluye un ejemplo para cada uno.

TablA 8.3:**Operadores booleanos**

operador booleano	qué significa	Ejemplo	qué significa el ejemplo
&&	lógica and	if (a>0 && b<10)	Es verdadero si ambas condiciones son ciertas
||	lógica or	if (a>0 || b<10)	Es verdadero si cualquiera de las dos condiciones es cierta
!	not	if (!a)	Es verdadero si 'a' es falsa

PIENSA EN ELLO...

Aunque utilices && solo en este sketch, ¿puedes pensar en cómo podrías utilizar los otros operadores para cambiar la ejecución/ lógica del programa?

La función *turnServo()* y los operadores booleanos

Echemos un vistazo otra vez a la primera declaración if en turnServos(). ¿Qué ocurre si val es un número entre 0 y 45? El primer servomotor gira a una posición de 45 grados y el segundo servomotor gira a una posición de 135 grados. Sabes esto porque se llamará a ambas funciones del objeto servo write()y moverán myservo1 y myservo2 a la posición deseada.

boolean operator

first condition

second condition

```
if (val > 0 && val <= 45) { //if val is between 0 and 45
  myservo1.write(45); //set position of first servo
  myservo2.write(135); //set position of second servo
}
```

if first condition and second condition are true, execute instructions

Las otras tres sentencias condicionales en turnServos() funcionan de manera similar: probando el valor de val (cuánto se ha girado el potenciómetro) y, si está en un rango particular de números, el Arduino girará entonces cada uno de los servomotores a las nuevas posiciones especificadas en la función turnServos().

Ahora que has escrito el sketch, asegúrate de haberlo guardado (como **LEA8_2_servos**) si aún no lo has hecho. Haz clic en el botón **Verificar** para comprobar si hay errores, y si está libre de errores haz clic en el botón **Subir**. Las banderas cambiarán de posición a medida que gires el potenciómetro (figura 8.28).

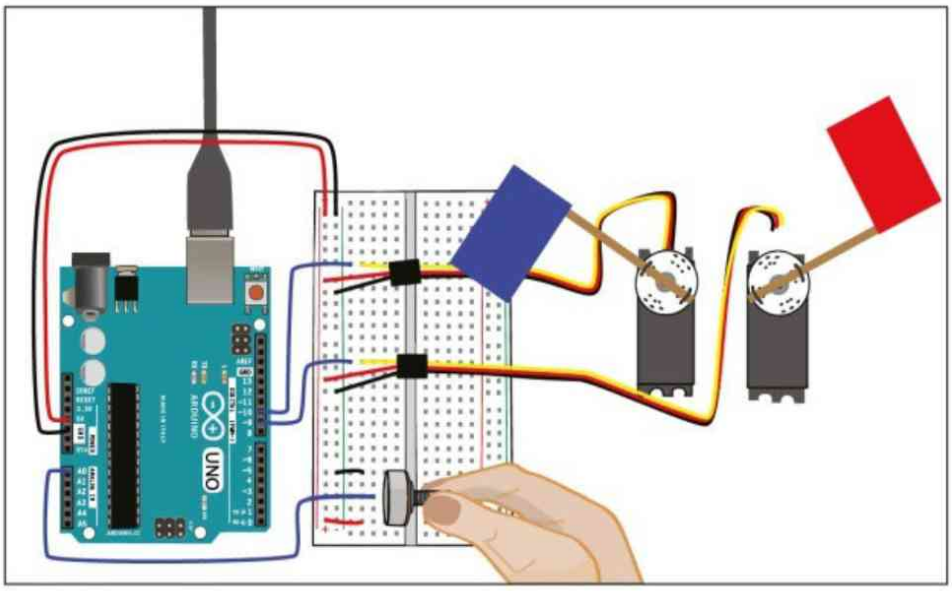


FIGURA 8.28 Los dos banderas ondean.

¿PREGUNTAS?

P: ¿Cuándo debo escribir mis propias funciones personalizadas?

R: Si sabes que hay un bloque de código que vas a necesitar a menudo en un sketch o si te encuentras repitiendo las mismas líneas, puede ser útil escribir una función personalizada para acortar el código y que sea más fácil de leer. Podríamos haber utilizado una función personalizada en el primer sketch de SOS. De hecho podríamos haber usado los bucles `for` allí también.

P: ¿Cómo sé qué operador booleano o de comparación debo utilizar?

R: Como con la mayoría de los conceptos de programación que se tratan en este libro, lo más fácil es decir en lenguaje sencillo lo que se intenta lograr para decidir cómo se estructurará la lógica en el sketch. Por ejemplo, si quieres que las condiciones sean verdaderas (se pulsa el botón 1 y se enciende un LED) para hacer que suceda algo, usarías `and`, o el signo `&&`, como has hecho en este sketch.

P: ¿Cuántos son demasiados condicionales?

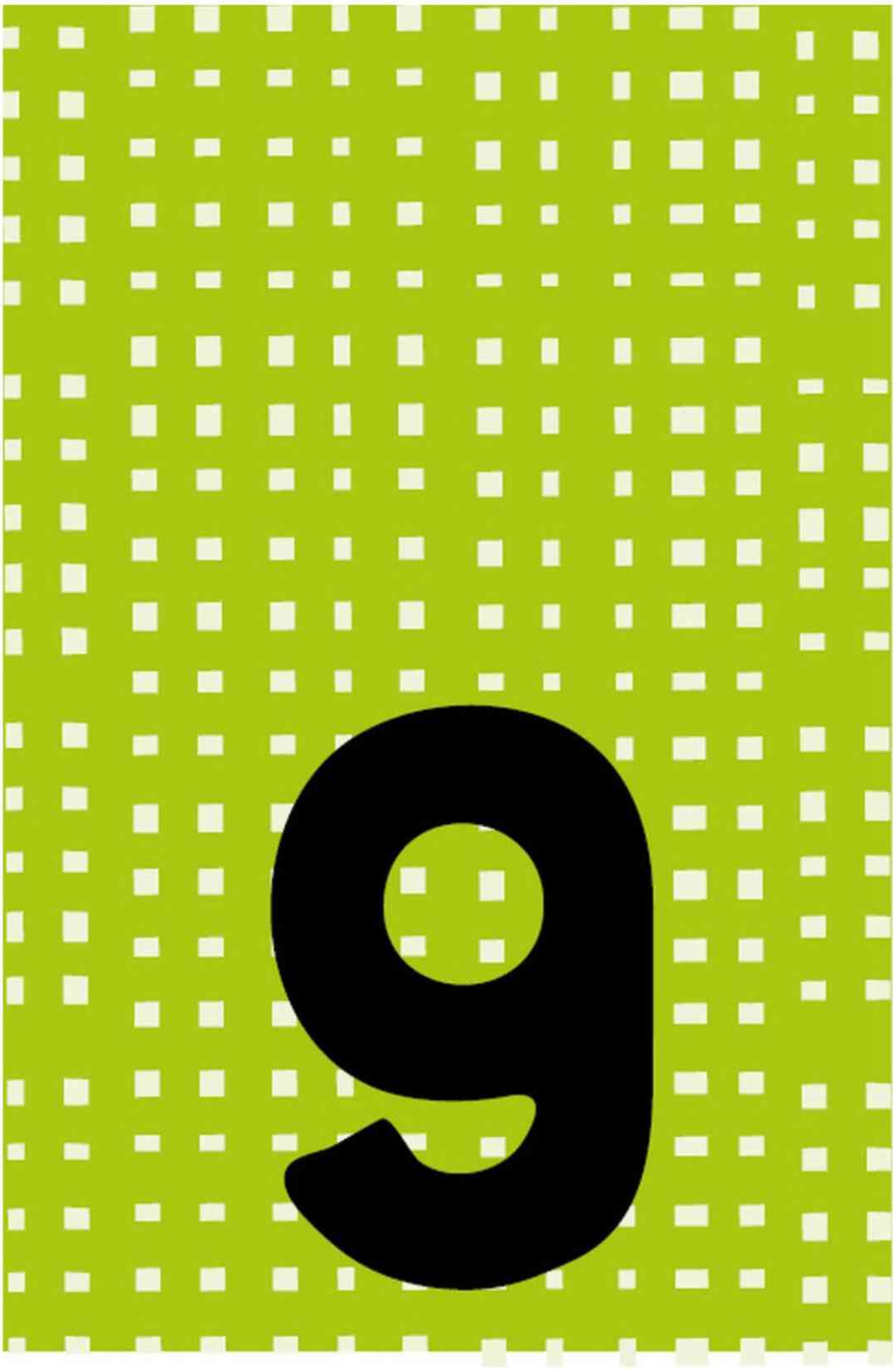
R: Hemos seleccionado cuatro posiciones clave en este sketch para crear una especie de coreografía de banderas. Puedes usar tantos condicionales como necesites para conseguir el comportamiento requerido en tu proyecto.

resumen

Nuestro objetivo principal en este capítulo ha sido mostrar cómo utilizar los servomotores. Los servomotores son versátiles y se pueden utilizar en muchos proyectos de Arduino, ya que un servo se puede hacer funcionar fácilmente de forma automática, como en el sketch Sweep, o se puede controlar mediante un sensor o un interruptor, como en el sketch Knob.

En este capítulo hemos estudiado una serie de importantes conceptos de programación. Has aprendido acerca de las bibliotecas, y has utilizado la biblioteca Servo para proporcionar a tu Arduino una serie de funciones servo que hacen más fácil el control de los servomotores.

También hemos explicado cómo utilizar los bucles for para colocar los servos en diferentes posiciones. Y has aprendido a utilizar operadores de comparación, compuestos y booleanos en el código. Puedes encontrar el sketch LEA8_2_servos en: github.com/arduinotogo/LEA/blob/master/LEA8_2_servos.ino.


g

MONTA TUS PROYECTOS

Ahora que has completado los proyectos de este libro, ¿qué sigue? Este capítulo será un breve resumen de consejos para la gestión de proyectos, comentaremos algunas ideas de proyectos y veremos rápidamente algunas de las otras placas Arduino disponibles y lo que pueden hacer.

GESTIÓN DE PROYECTOS

En este libro hemos seguido paso a paso las instrucciones sobre cómo trabajar con el Arduino. ¿Cómo puedes iniciar tus propios proyectos? El primer paso debe ser la investigación. Echa un vistazo a los recursos en línea. Muchos de los proveedores que hemos mencionado tienen sitios web repletos de tutoriales e ideas sobre proyectos. Además, navegar por los sitios para tener una idea de las entradas y salidas que están disponibles para ti, te debería proporcionar un montón de ideas. Aquí te indicamos algunos sitios:

makezine.com/category/technology/arduino/

learn.adafruit.com/category/learn-arduino

learn.sparkfun.com/tutorials/tags/arduino?page=all

playground.arduino.cc/Projects/Ideas

DEFINE EL PROYECTO

Cuando tengas una idea para un proyecto, intenta dibujar o escribir el sistema que piensas construir. Esto puede ser tan sencillo como hacer una lista que contenga los componentes que estás planeando usar y el tipo de comportamiento que necesitarás para el código. Normalmente ayuda poder dividir el proyecto en entradas, salidas y código. Recuerda, tu proyecto siempre será un sistema, con entradas, salidas y código que controla el comportamiento que se ejecuta en el equipo Arduino (figura 9.1).

DIVÍDELO EN PARTES

Dividir el proyecto en sus componentes, comenzando con la sección más sencilla que ya sabes cómo hacer, te ayudará a poner en marcha el trabajo. Abordar cada parte por separado, en lugar de enfrentarte a todo al mismo tiempo, hace que sea más fácil realizar el trabajo. Además, cuando empiezas a trabajar, simplificar la idea te ayudará a realizarla. Siempre es posible mejorarla y depurarla en versiones posteriores.

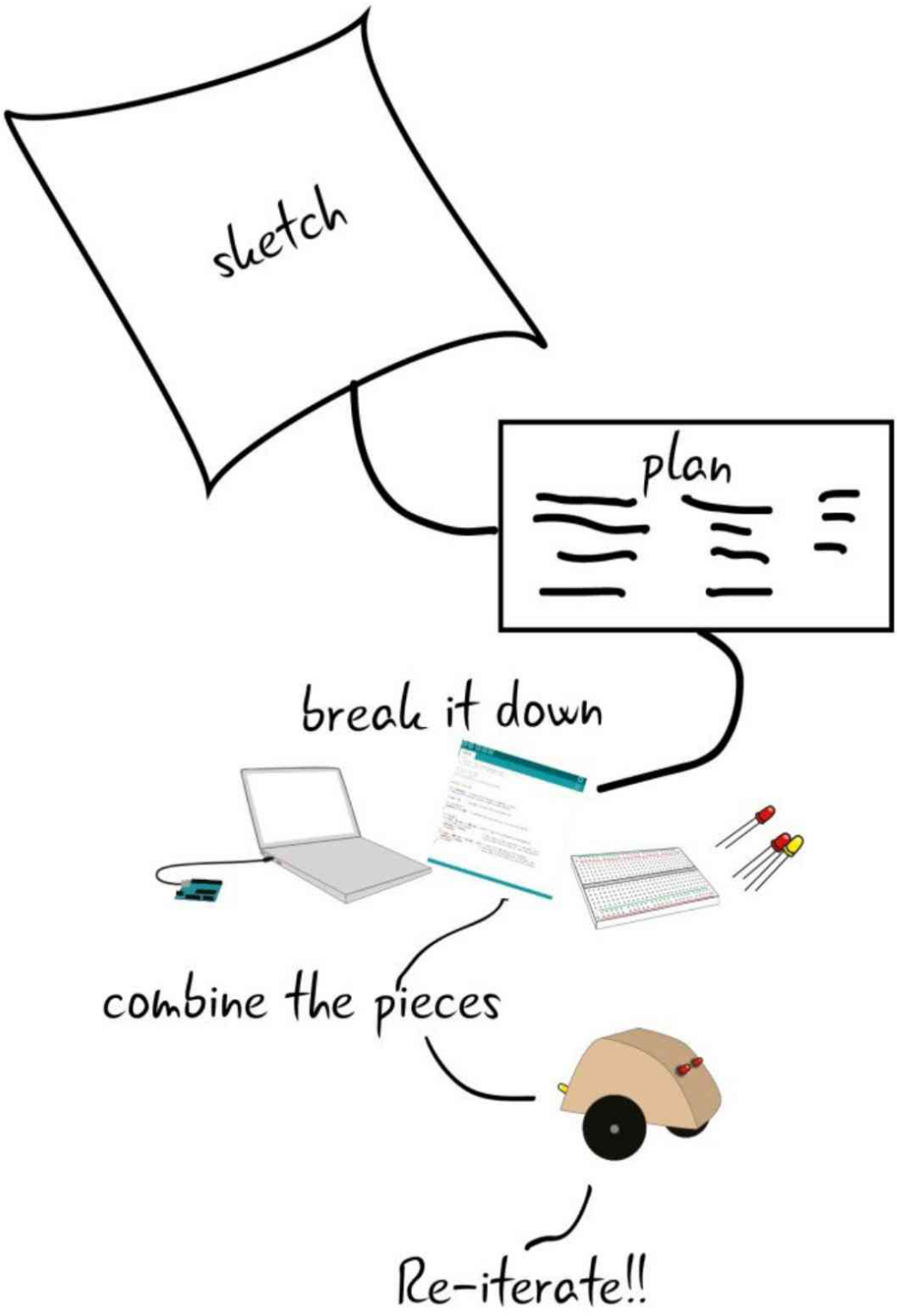


FIGURA 9.1 Las notas de planificación pueden ayudar a dirigir tu proyecto.

Cuando empiezas a montar tu proyecto, ¿qué haces si algo no funciona? A lo largo de este libro hemos resaltado la importancia de la depuración, tanto del código como de los circuitos (figura 9.2). Debes ser paciente y aplicar un

enfoque metódico para examinar cada elemento de tu proyecto. Si encuentras un error en el editor de código del IDE, anota el término exacto y escríbelo en un motor de búsqueda. Probablemente descubrirás que no eres la primera persona que tiene ese problema. Los foros en el sitio web de Arduino (forum.arduino.cc) son un gran sitio para buscar respuestas a problemas y hacer preguntas. Arduino Stack Exchange (arduino.stackexchange.com) es otro sitio en el que se puede buscar.

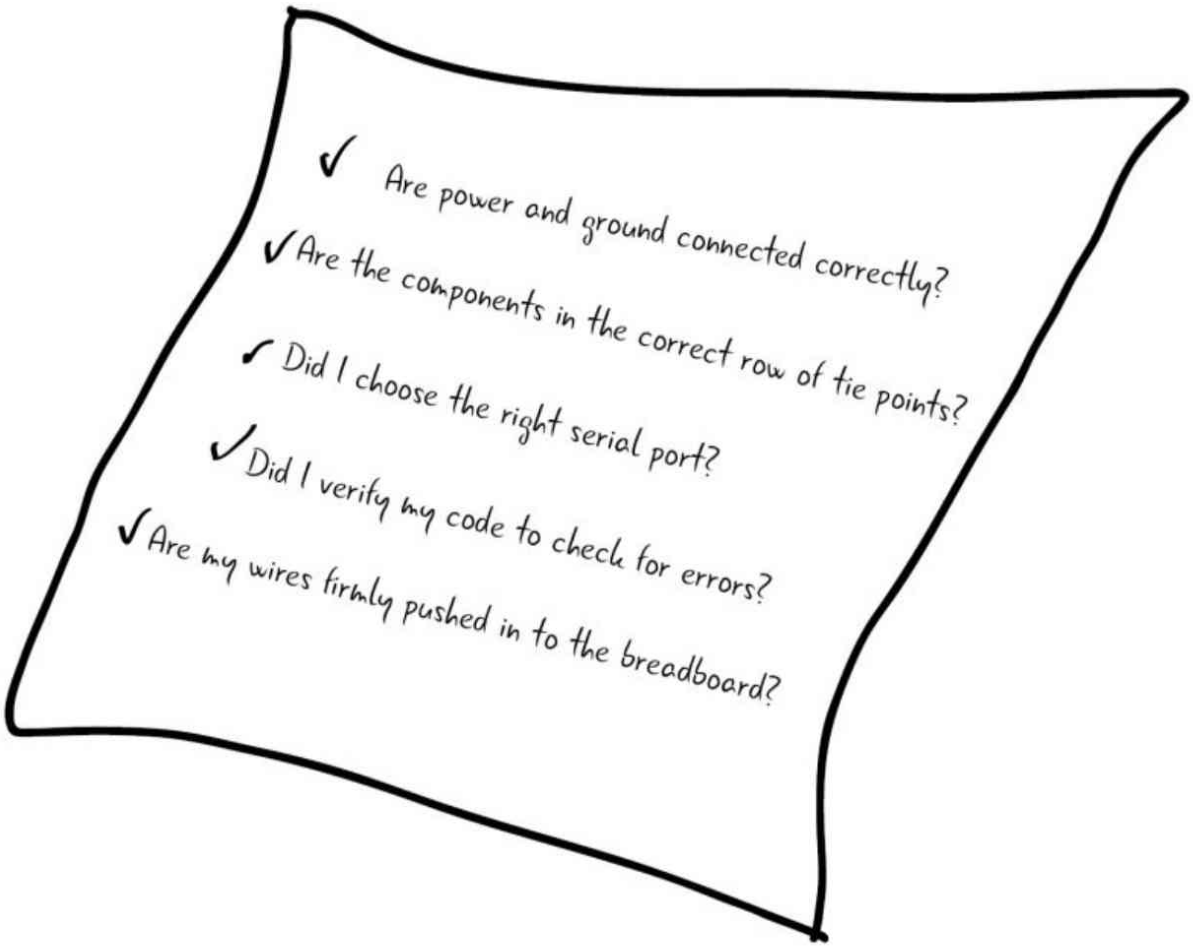


FIGURA 9.2 Evita las frustraciones de los proyectos fallidos: usa la depuración.

PRUEBAS DE USUARIO

Una vez que tengas una versión o prototipo funcional del proyecto,

compártelo con alguien. Explícale el proyecto y pídele que pruebe tu dispositivo. Al montar un proyecto, es fácil hacer muchas suposiciones sobre cómo lo verán o lo usarán los demás, y puede ayudar tener una perspectiva externa para romper algunas de tus suposiciones. Si es posible, tener un buen número de personas dedicadas a probar el proyecto te ayudará a hacer la mejor versión posible y a desarrollar tu idea. Si no estás seguro de a quién dirigirte, comienza con tus amigos y familiares (figura 9.3).

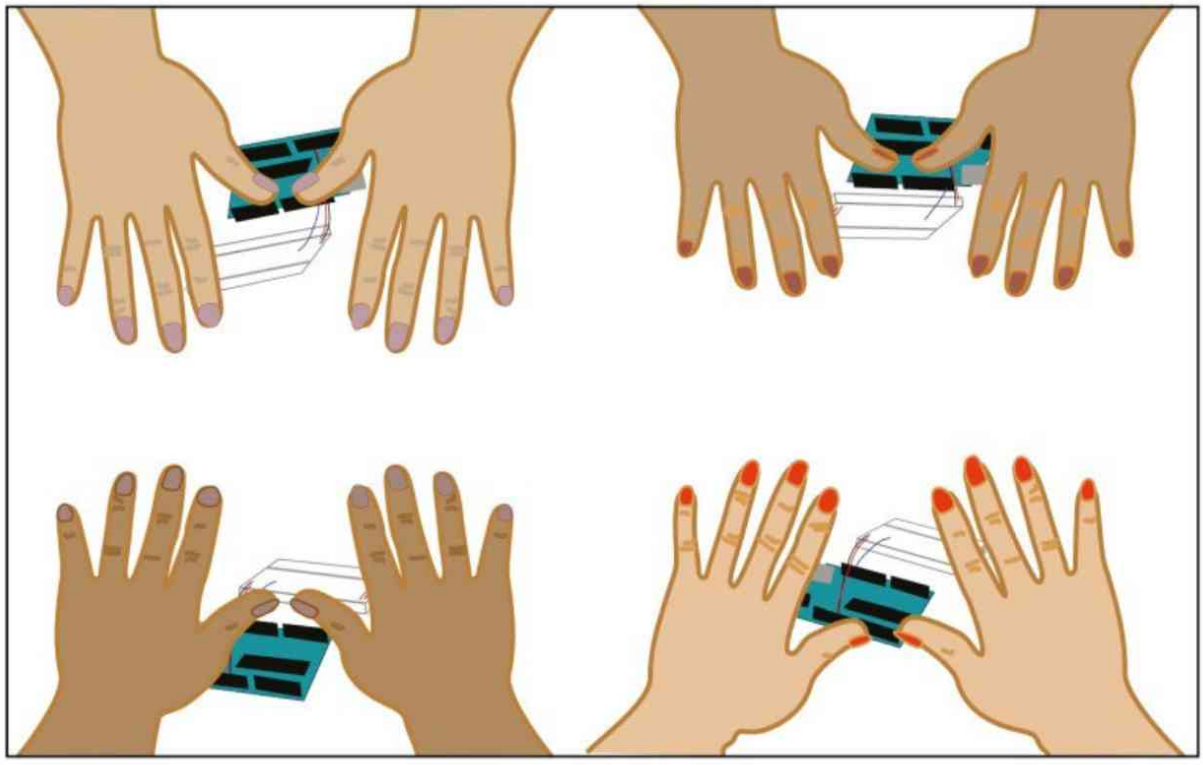


FIGURA 9.3 Haz que otros prueben tus proyectos.

Reflexionar y repetir

Ahora que has superado el primer paso de tu proyecto, no te debe costar trabajo escribir algunas notas. ¿Qué fue bien en el proyecto? ¿Qué mejoras podrían introducirse? Estas notas pueden ayudarte a iterar el proyecto y hacer mejores versiones en el futuro al mejorar errores pasados o suposiciones falsas.

Ahora que tienes una comprensión básica y el apoyo de algunas técnicas de gestión de proyectos, hablemos de los componentes más frecuentes en los proyectos de Arduino.

Componentes útiles

No tenemos suficiente espacio para entrar en todas las variedades de sensores y de salidas que existen en el mundo, pero sí queremos mencionar opciones populares que pueden ayudar a que tus proyectos cobren vida.

Sensores

Estos son algunos de los sensores más utilizados que puedes incorporar fácilmente a tus proyectos.

Detección de distancia y movimiento

Los sensores infrarrojos pasivos (PIR; figura 9.4) y los telémetros ultrasónicos (figura 9.5) se utilizan para determinar la distancia a la que están de tu equipo las personas u objetos. También se pueden utilizar para comprobar si alguien ha caminado delante del equipo. Dado que ambos componentes proporcionan a menudo valores analógicos, puedes utilizar estos sensores de forma similar a como empleaste la fotocélula en el capítulo 7, "Valores analógicos".

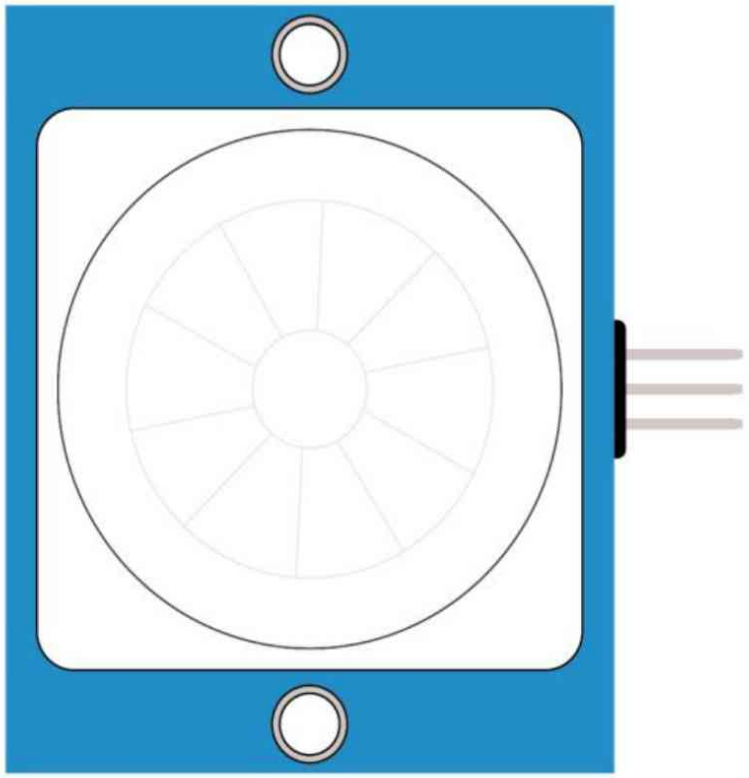


FIGURA 9.4 Sensor infrarrojo pasivo (PIR).

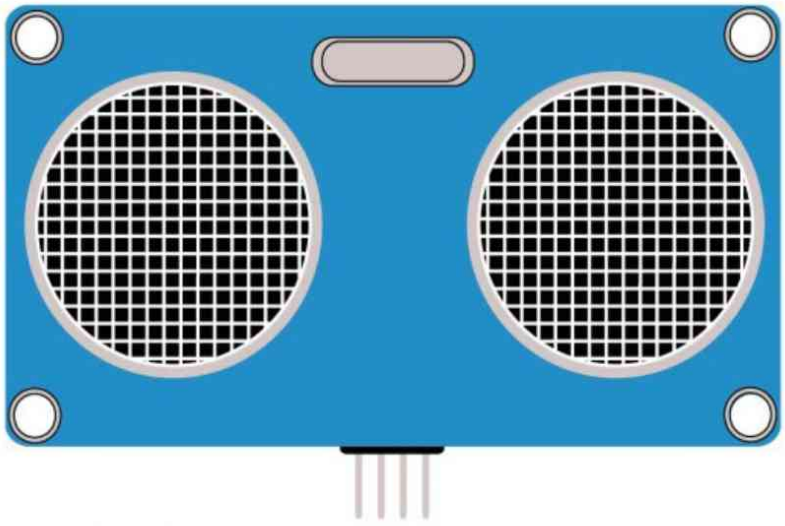


FIGURA 9.5 Sensor ultrasónico.

Resistencia de detección de fuerza

Las resistencias de detección de fuerza (FSR) permiten detectar diferentes valores de empuje o presión sobre un sensor (figura 9.6). Dado que

proporcionan lecturas analógicas, puedes escalar la respuesta para mover servomotores, iluminar diferentes secciones o reproducir sonidos de un altavoz. Las FSR se utilizan en controladores de juegos y otras interacciones prácticas. Los FSR se presentan en una gran variedad de formas (tanto cuadradas como redondas) y tamaños y distintos valores de sensibilidad.

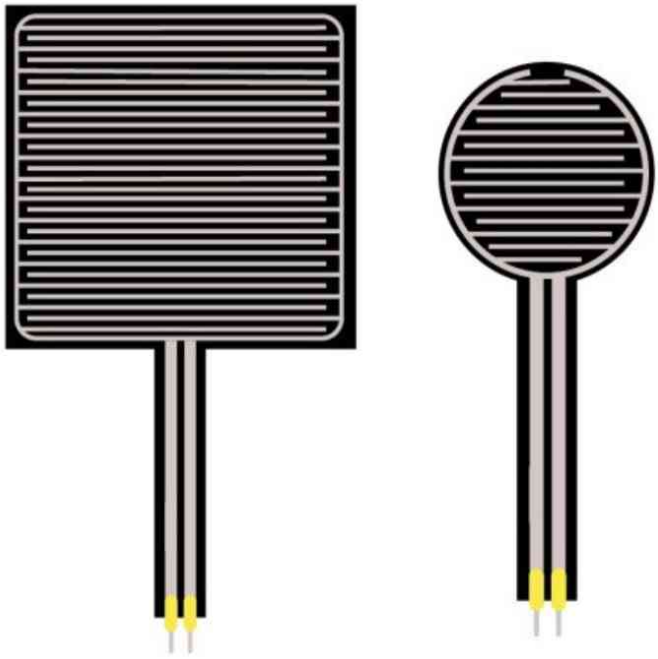


FIGURA 9.6 Las resistencias de detección de fuerza (FSR) se presentan en diferentes formas y tamaños.

Otros sensores

Como se mencionó anteriormente, hay muchos más sensores que pueden ayudarte a ampliar tus proyectos de Arduino. Desde sensores de temperatura hasta micrófonos para medir los niveles de volumen, pasando por monitores de pulso y frecuencia cardíaca; encontrar el sensor adecuado puede hacer que tu proyecto destaque.

Actuadores y Motores

Te hemos mostrado proyectos que incorporan movimiento usando

servomotores, pero hay otros tipos de actuadores (componentes que pueden mover algo) que pueden hacer que tu proyecto ejecute movimientos de varios tipos. Hemos destacado algunas opciones populares que indicamos a continuación.

Motores de corriente continua

Los motores de corriente continua se presentan en una gran variedad de tamaños y valores de resistencias para satisfacer incluso los proyectos más difíciles (figura 9.7). A menudo giran continuamente en una sola dirección y se mueven más rápido o más despacio, en función de cuánta energía se les aplique (dentro de un rango de seguridad). Los motores de C.C. se utilizan con pleno éxito en la conducción de ruedas, para levantar objetos pesados, etc.

FIGURA 9.7 Motor de corriente continua.

Motores paso a paso

Los motores paso a paso (figura 9.8) son un tipo de motor más controlable que el de corriente continua básico, lo que significa que también requieren más potencia de computación del equipo Arduino para funcionar. En lugar

de girar continuamente, los motores paso a paso realizan "paso a paso" un porcentaje de la rotación total. Esto significa que se pueden utilizar para conseguir un posicionamiento preciso y se inician y se detienen cuando se les ordena. Los motores paso a paso funcionan bastante bien con el Arduino, aunque a menudo requieren un chip de circuito integrado H-bridge o un controlador de este tipo de motores para actuar de forma más compleja.

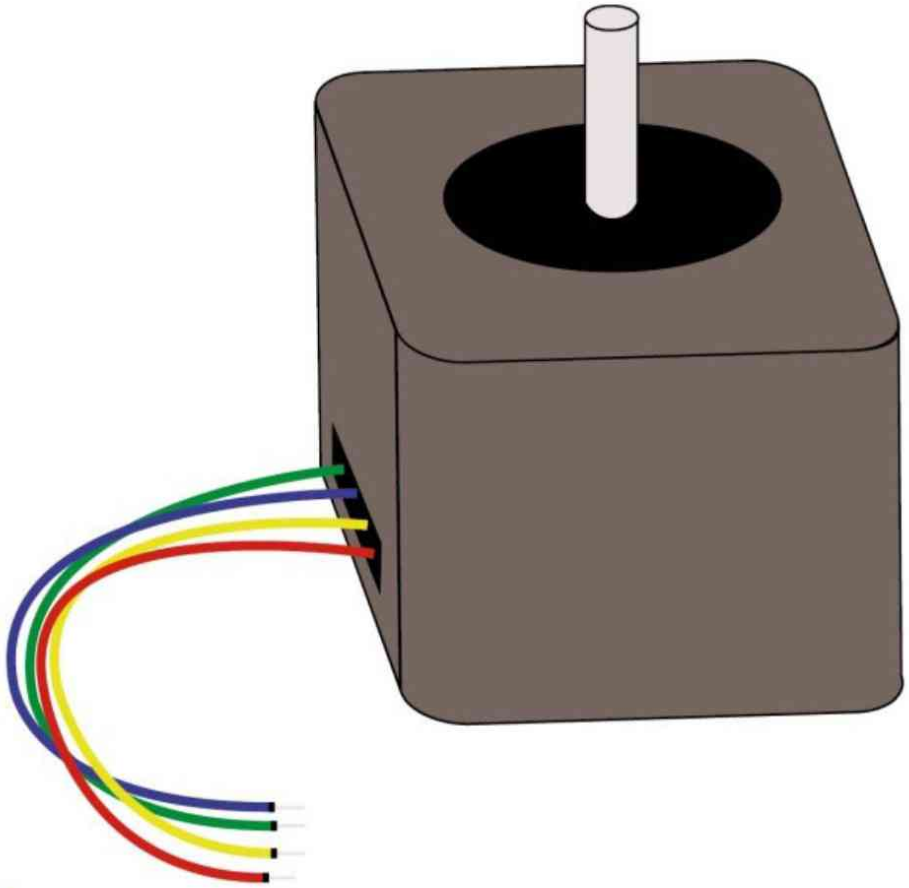


FIGURA 9.8 Motor paso a paso.

Solenoide

Los solenoides (figura 9.9) son muy diferentes de los otros actuadores de los que hemos hablado. En lugar de crear una rotación, los solenoides se "disparan" en línea recta. Tienen un resorte unido a un eje metálico que empuja o tira del cuerpo central del motor, dependiendo del tipo que sea. A menudo se utilizan en instrumentos musicales para tocar elementos de percusión o

instrumentos similares a campanas con el fin de crear nuevos sonidos.

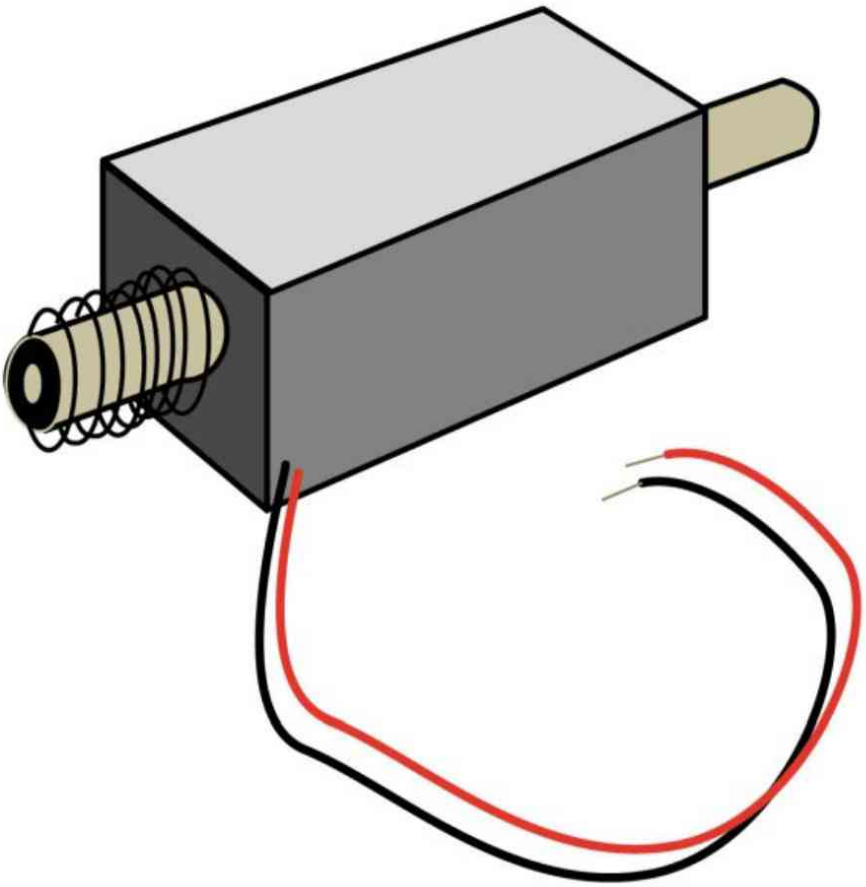


FIGURA 9.9 Solenoide.

Tipos de Proyectos

Hemos hablado de una gran variedad de proyectos que puedes montar con Arduino, pero queríamos sugerirte un par de tipos de proyectos más y algunas ideas para ayudarte a empezar.

DOMÓTICA

Aunque hay varios productos en el mercado, puedes montar tus propios proyectos de domótica usando Arduino. Las opciones más populares para los proyectos de automatización del hogar incluyen el encendido de luces, la activación de ventiladores o el apagado de electrodomésticos.

Robots

Los robots son siempre una opción popular para los proyectos de Arduino. Con unos pocos motores y sensores, puedes tener un robot mascota en poco tiempo. Los robots de una sola tarea también son una gran opción, desde los robots que cortan mantequilla hasta los que rastrean objetos en el suelo. ¡Incluso se pueden construir con cartón! (figura 9.10).

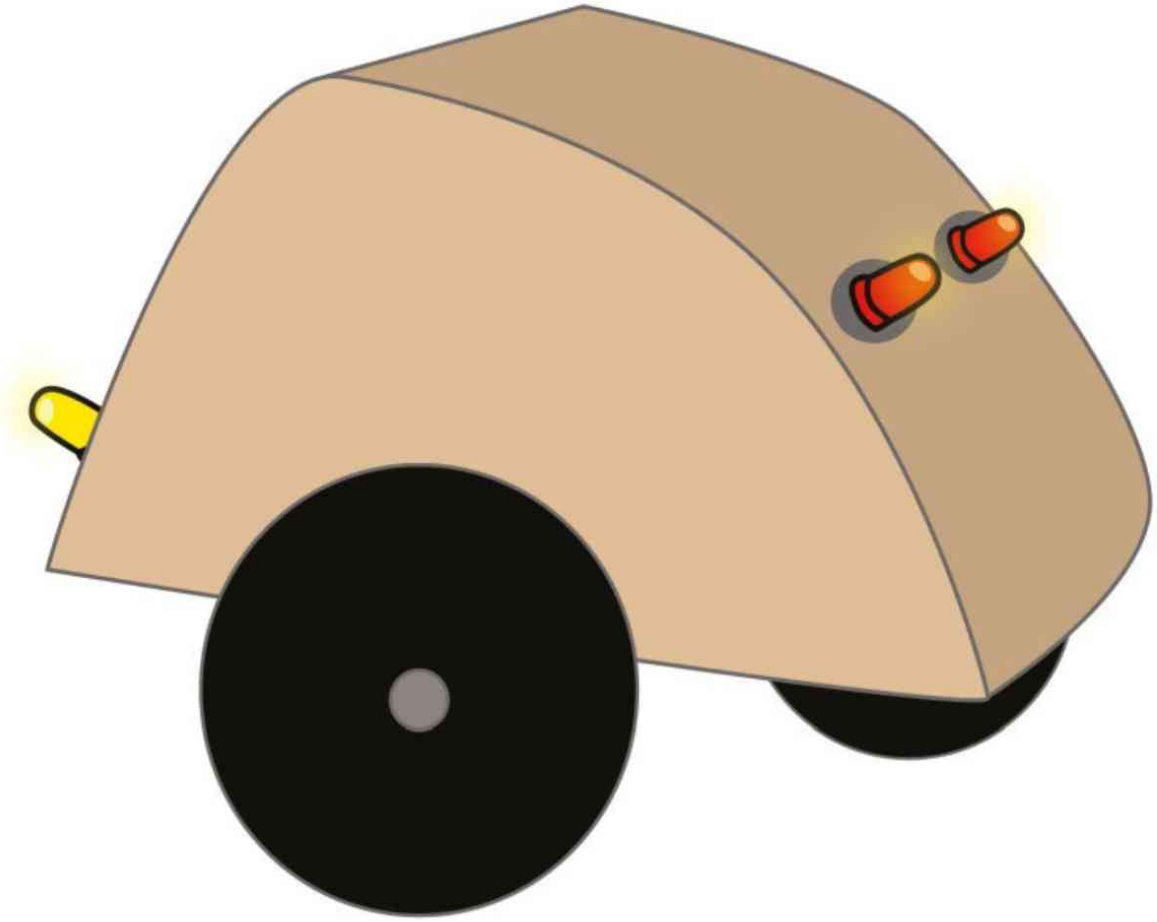


FIGURA 9.10 Robot mascota de cartón.

Proyectos ponibles

Los proyectos ponibles incluyen cualquier tipo de ropa, joyas o accesorios que combinan la capacidad de la informática física con la portabilidad y la accesibilidad. Puedes utilizar sensores para obtener datos sobre el pulso de sus usuarios o incorporar botones directamente en la ropa que llevan puesta. Los proyectos con más éxito utilizan guantes, gorras, camisetas o joyas y sensores para activar instrumentos musicales o pantallas de visualización (figura 9.11). ¿Qué tipo de proyectos se te ocurren que utilicen accesorios corrientes?

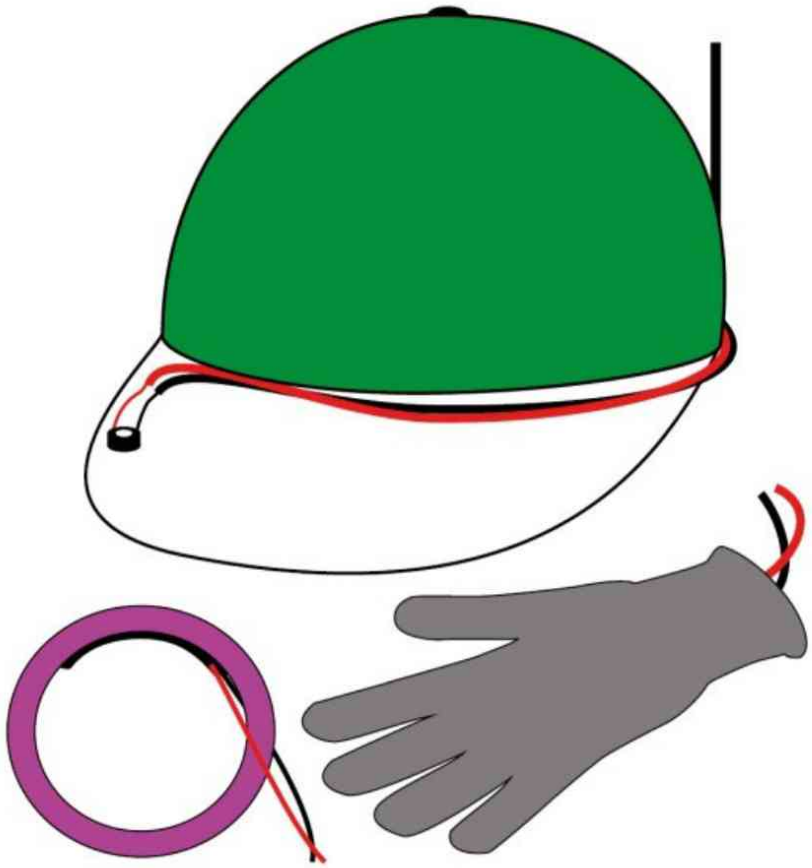


FIGURA 9.11 Pulseras, sombreros y prendas de vestir son todas opciones populares.

Proyectos de arte

Más allá de las categorías que ya hemos mencionado, puedes realizar cualquier tipo de proyecto artístico que tengas en mente. Desde dispositivos de pintura autogenerados hasta esculturas en movimiento y libros interactivos, la única limitación de un proyecto de arte es tu imaginación.

Otras Versiones de placas de Arduino

Hemos comentado que hay muchas otras versiones de Arduino, que tienen diferentes funcionalidades. Vamos a ver brevemente algunas de estas placas y lo que hacen. Hay muchas más.

Arduino 101

El Arduino 101 (figura 9.12) es una excelente opción para sustituir a la Uno, ya que es del mismo tamaño y tiene la misma disposición general que la Uno. También tiene conectividad bluetooth de baja energía (BLE) y un acelerómetro/giro de seis ejes. Si quieres que tu proyecto reconozca los gestos, esta puede ser una buena opción. Puedes leer más sobre esto en: store.arduino.cc/usa/arduino-101.

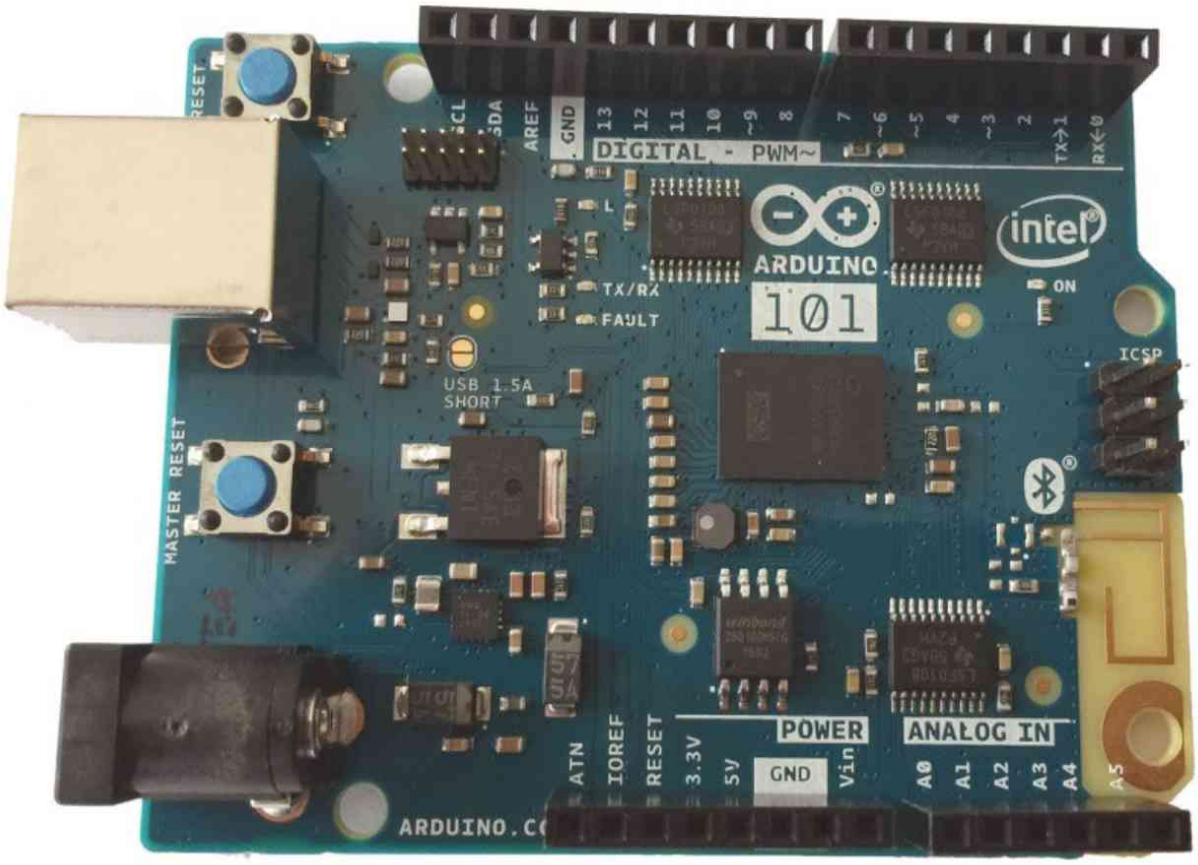


FIGURA 9.12 Arduino 101.

Arduino YÚN

Mitad Arduino y mitad Linux, el Arduino YÚN (figura 9.13) te permitirá usar wifi y la potencia de un sistema operativo con el que poder realizar tareas informáticas complicadas. En YÚN se pueden ejecutar scripts Python para analizar datos en la placa controlada por Linux, con el Arduino manejando entradas y salidas que responden a esa información. Tiene una ranura para una tarjeta SD y conectividad wifi y Ethernet integrada. Para ampliar la información puedes entrar en: store.arduino.cc/usa/arduino-yun.

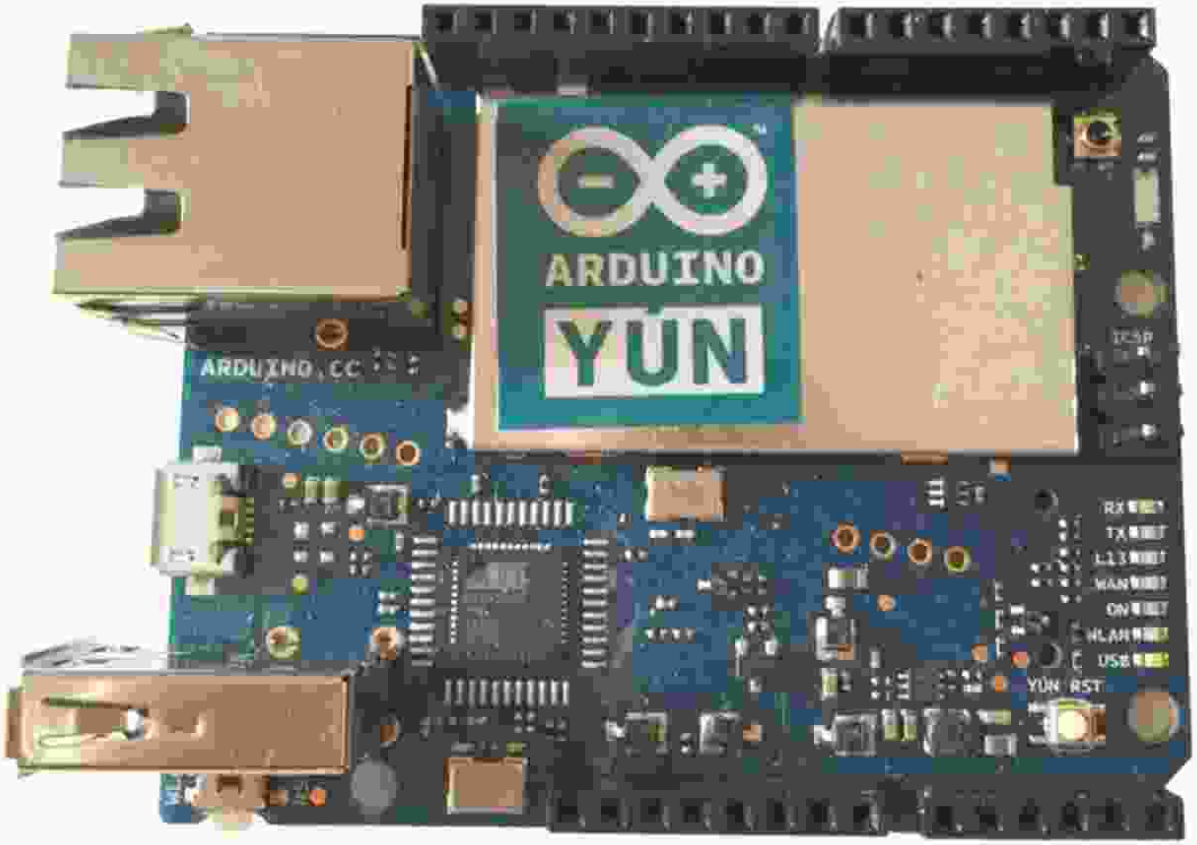


FIGURA 9.13 Arduino YÚN

Arduino lilypad

Como se ha mencionado en la sección "Proyectos ponibles", a veces quieres tener la posibilidad de conectar el Arduino a una prenda de vestir, y el Arduino Uno puede no ser el más recomendable. Arduino Lilypad es genial

porque no solo es plano y menos llamativo, sino que también puede utilizar hilo conductor en lugar de cables. Esto te permitirá coser tanto los sensores como el Arduino directamente en el tejido. Existen varias versiones de Lilypad; la figura 9.14 muestra la placa principal de Arduino Lilypad.

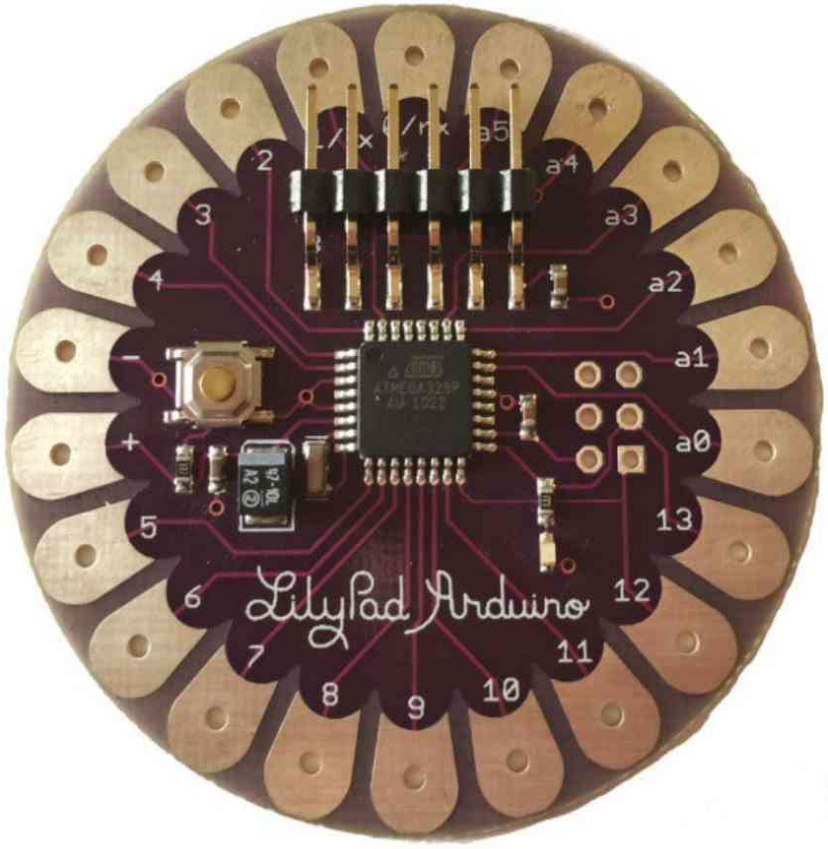


FIGURA 9.14 Arduino Lilypad. Placa principal.

Otras placas Arduino

Aunque entrar en cada modelo en detalle no es el objetivo de este libro, nos gustaría mencionar algunas placas de interés. Mega 2560 tiene 54 pines de entrada/salida digital y 16 pines de entrada analógica; es adecuado para proyectos grandes. Leonardo tiene incorporada la comunicación USB, por lo que puedes conectar un teclado y un ratón directamente. Micro es la placa más pequeña de la familia Arduino, lo que la hace apropiada para su integración en proyectos. Al igual que Leonardo, soporta la comunicación USB.

MKR ZERO es una placa más pequeña diseñada para trabajar con aplicaciones de audio. MKR1000 tiene conectividad wifi y una batería recargable de polímero de litio integrada. Gemma, desarrollada por Adafruit, es otra placa diseñada para utilizarla en prendas de vestir.

escudos Arduino

Además de las distintas placas Arduino, hay una amplia variedad de "escudos" de marca y de terceros que se adhieren a la parte superior del Arduino y amplían su funcionalidad. Estos incluyen lo siguiente:

- Compatibilidad con tarjetas SD para guardar datos
- Compatibilidad con archivos de sonido para reproducir grabaciones de audio
- Posibilidad de controlar motores

y mucho más

En **arduino.cc/en/Main/Products**, encontrarás un gráfico que enlaza con los detalles y especificaciones técnicas de cada modelo y con algunos de los escudos disponibles.

¡Documenta el proyecto y compártelo!

Una de las mejores cosas de los proyectos de código abierto es ver lo que otros han inventado, y ahora es tu turno de compartir tus proyectos con el mundo. Estos son algunos consejos que pueden diferenciar tu proyecto de otros proyectos en línea.

Toma buenas fotos

Una frustración que a menudo experimentamos con los proyectos de informática física *hazlo tú mismo* es que puede ser difícil documentar el proyecto con fotografías. Te recomendamos que tengas una iluminación intensa y homogénea y un fondo liso debajo del montaje.

Si piensas tomar fotografías del cableado, es muy importante que codifiques los cables por colores y evites cruzarlos con demasiada frecuencia. De lo contrario, ¡corres el riesgo de que el montaje se parezca a unos espaguetis! (figura 9.15).

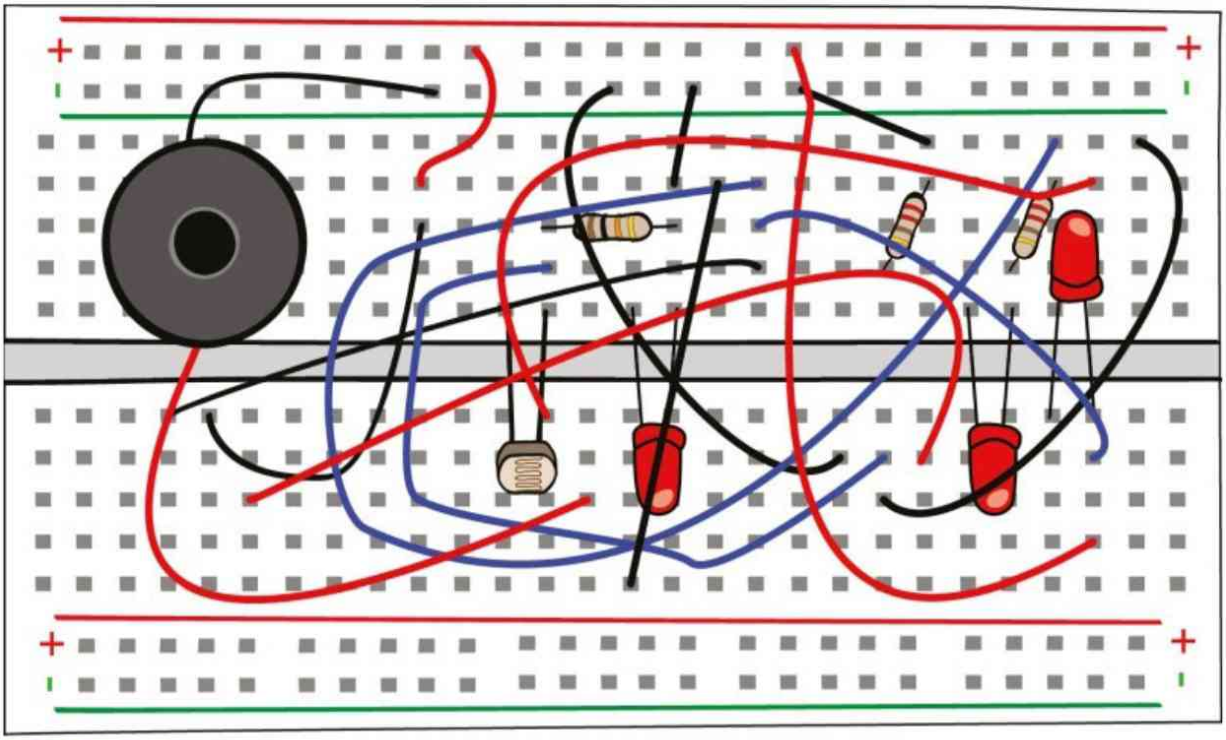


FIGURA 9.15 Cableado tipo espaguetis; ¡¡esto no es un circuito real!!

documenta el Proyecto

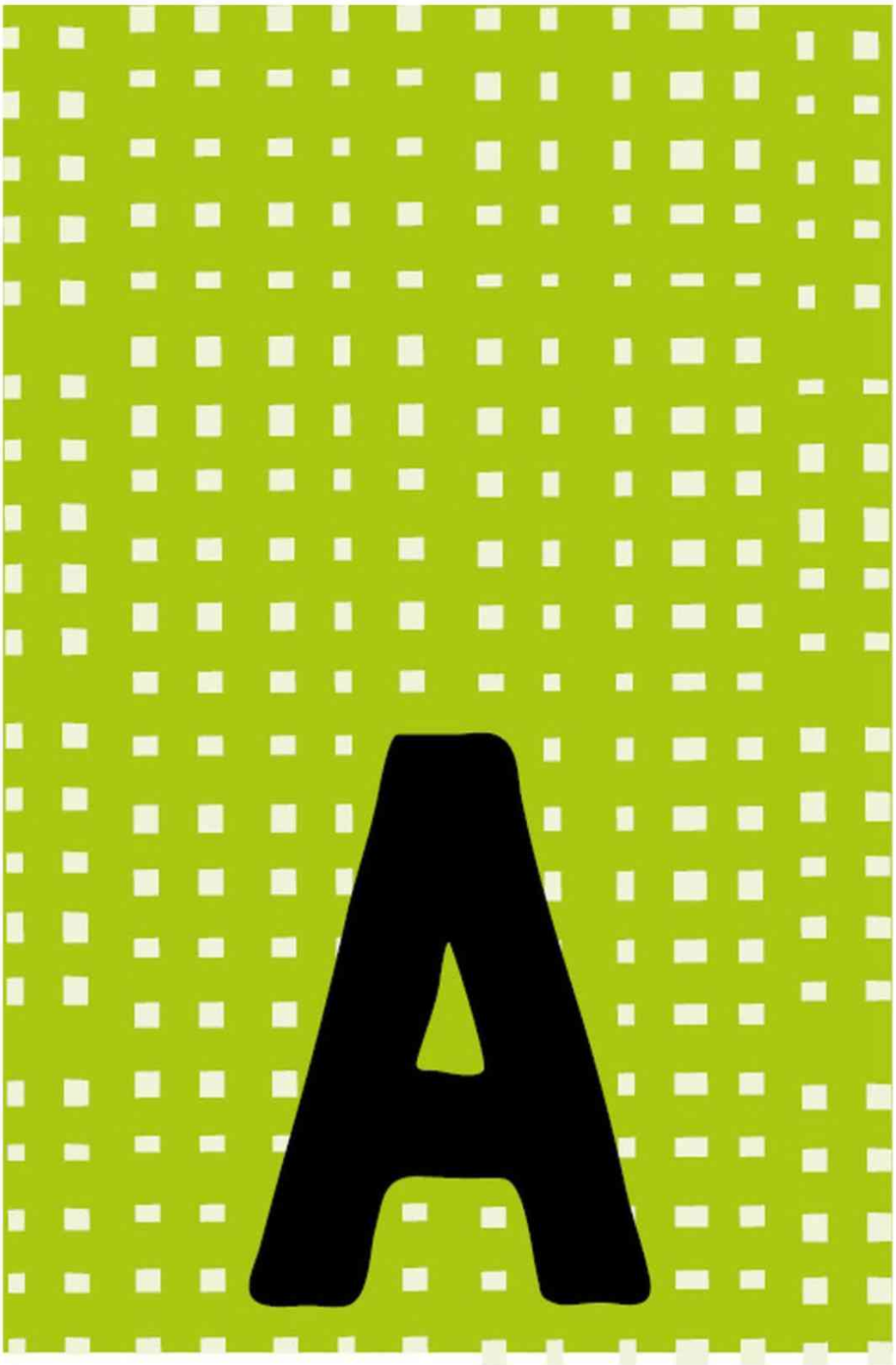
Si has tenido problemas con alguna sección del código o con cierto concepto, lo más probable es que la persona que intente reproducir la misma idea (o algo similar) se tropiece con los mismos problemas que tú. Escribir un resumen de tu experiencia con el montaje de un proyecto, o los pasos que seguiste para realizarlo, te ayudará a recordar los trucos que has aprendido para tenerlos en cuenta en futuros proyectos, y puedes evitarle a otro proyectista dolores de cabeza.

comparte el Proyecto

Aunque no es obligatorio, puede ser de gran ayuda compartir lo que se te ocurra para que otros lo vean. En muchos sitios web, como makershare.com y instructables.com, tienes la opción de publicar tus propios proyectos e incluir instrucciones paso a paso para realizarlos. Esta es uno de los aspectos más importantes de Arduino, que es de código abierto: el conocimiento se puede compartir con todos.

RESUMEN

Hemos llegado al final del libro. En los capítulos anteriores, te hemos hecho una introducción a la teoría y práctica de electrónica básica, así como a los conceptos de programación. En este capítulo te hemos dado algunos consejos para seguir adelante con tus propios proyectos. Ahora estás en disposición de montar tus propios y fabulosos proyectos Arduino.


A

Apéndice: lectura de códigos de resistencias

Si fueras a comprar resistencias y vinieran identificadas con algún tipo de etiqueta, tendrías un problema si perdieras la etiqueta indicando su valor. Afortunadamente, cada resistencia tiene un conjunto de bandas de color impresas en su superficie que te indican su valor. Mientras que hay resistencias con seis, tres o incluso una banda, las resistencias con las que se trabaja normalmente tienen cuatro bandas, y estas son las que vamos a ver en este apéndice.

Identificación de resistencias por las bandas de color

Veamos más de cerca una resistencia en la figura A.1. Las resistencias tienen dos terminales y un cuerpo con bandas de colores sobre él.

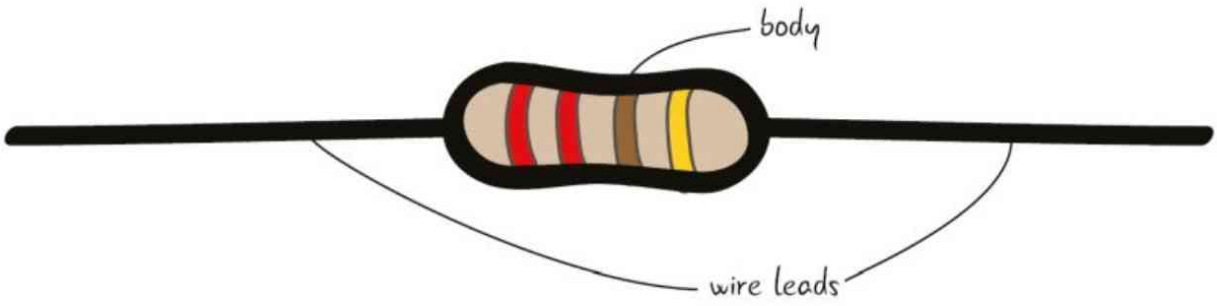


FIGURA A.1 Resistencia.

Orientación de la Resistencia

No solo importan los colores de las bandas, sino también el orden en el que aparecen los colores. ¿Cómo sabes lo que significa cada color? Lo primero que hay que hacer es orientar la resistencia en la dirección correcta, como se muestra en la figura A.2. En un lado de la resistencia el color de la banda será dorado o plateado. Esta banda debe estar situada en el lado derecho de la resistencia. Hay que identificar la banda dorada o plateada y colocar la resistencia de manera que quede a la derecha.

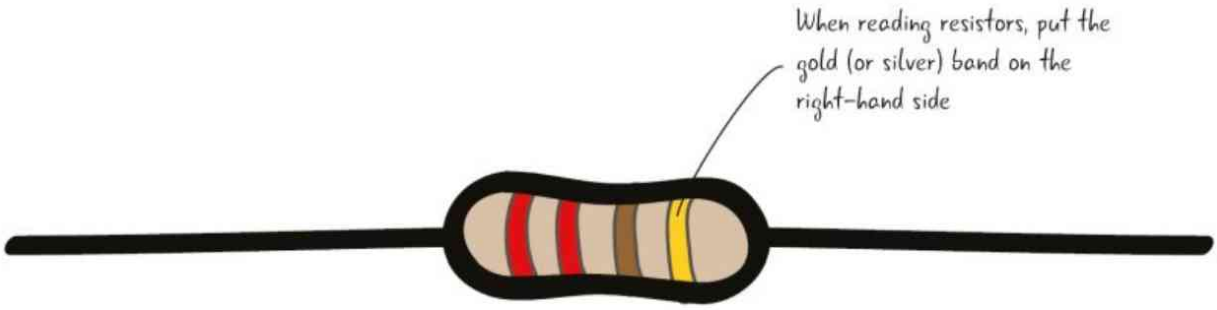


FIGURA A.2 Orientación de la resistencia.

Ahora que la resistencia está orientada correctamente, puedes identificar las otras bandas de color en el cuerpo de la resistencia. En la figura A.3

hemos etiquetado las bandas por orden. El color de cada banda tiene un significado particular.

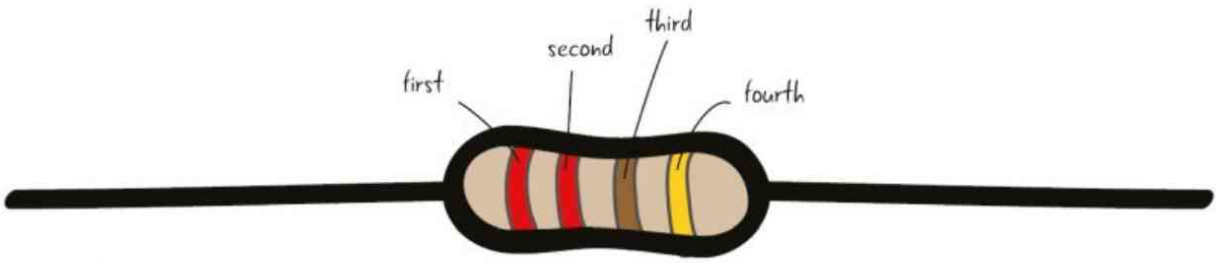


FIGURA A.3 Numeración de las bandas en la resistencia.

Tabla de colores de las resistencias

La figura A.4 es una tabla de colores estándar a la que se adaptan todas las resistencias. Podemos encontrar tablas similares en línea. Repasaremos en detalle lo que significa cada banda. Los colores significan lo mismo para todas las resistencias.

i

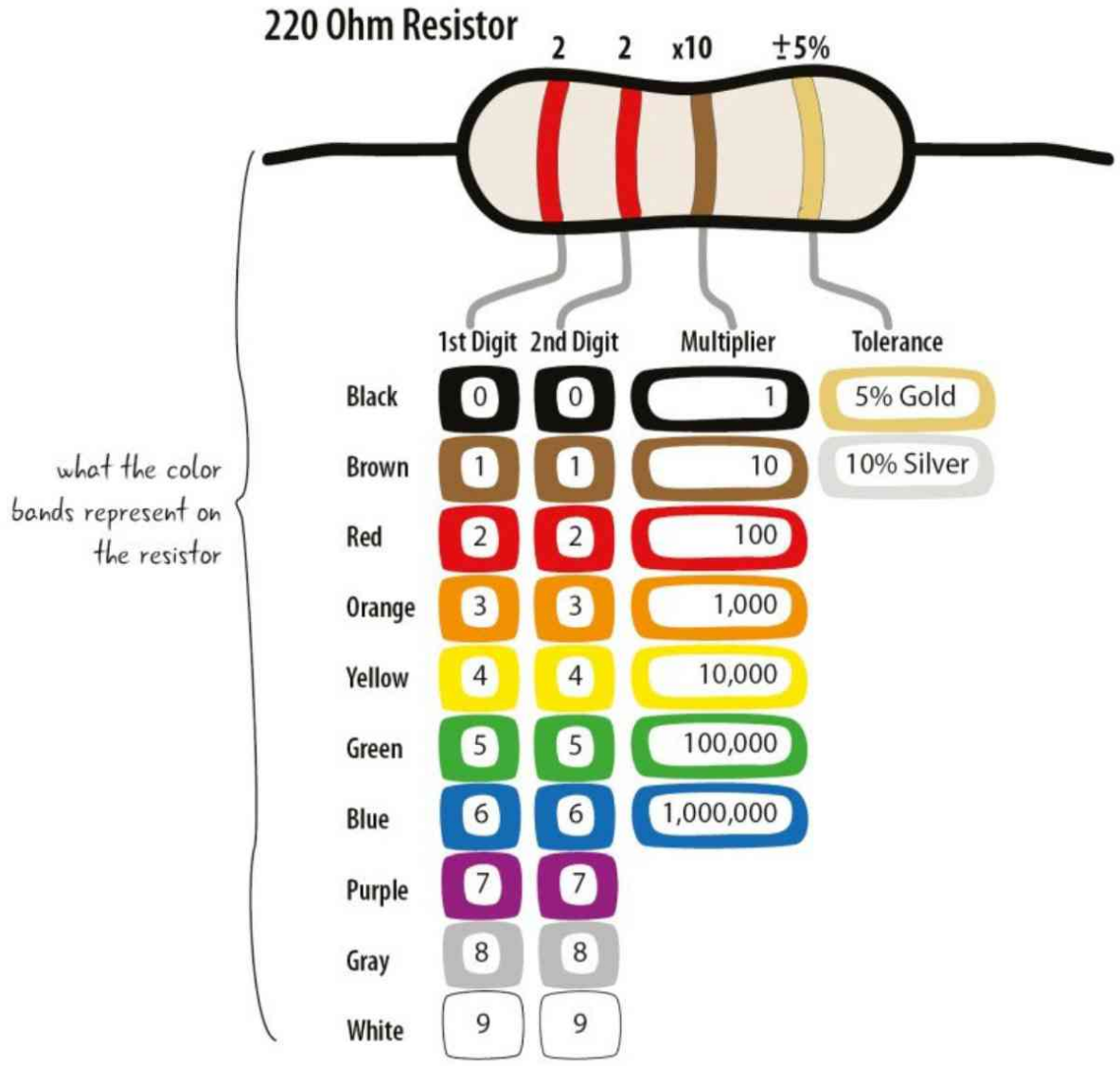


FIGURA A.4 Tabla de colores de las bandas de las resistencias.

Decodificación de las resistencias

Ahora que ya conoces la tabla de colores, vamos a mostrarte cómo se utiliza con una resistencia.

La primera banda representa el dígito más significativo, o el primer dígito de su valor. Por ejemplo, en la resistencia de la figura A.5, la primera banda es roja. Si consultas la tabla de colores, ves que el rojo de la primera banda es igual al número 2.

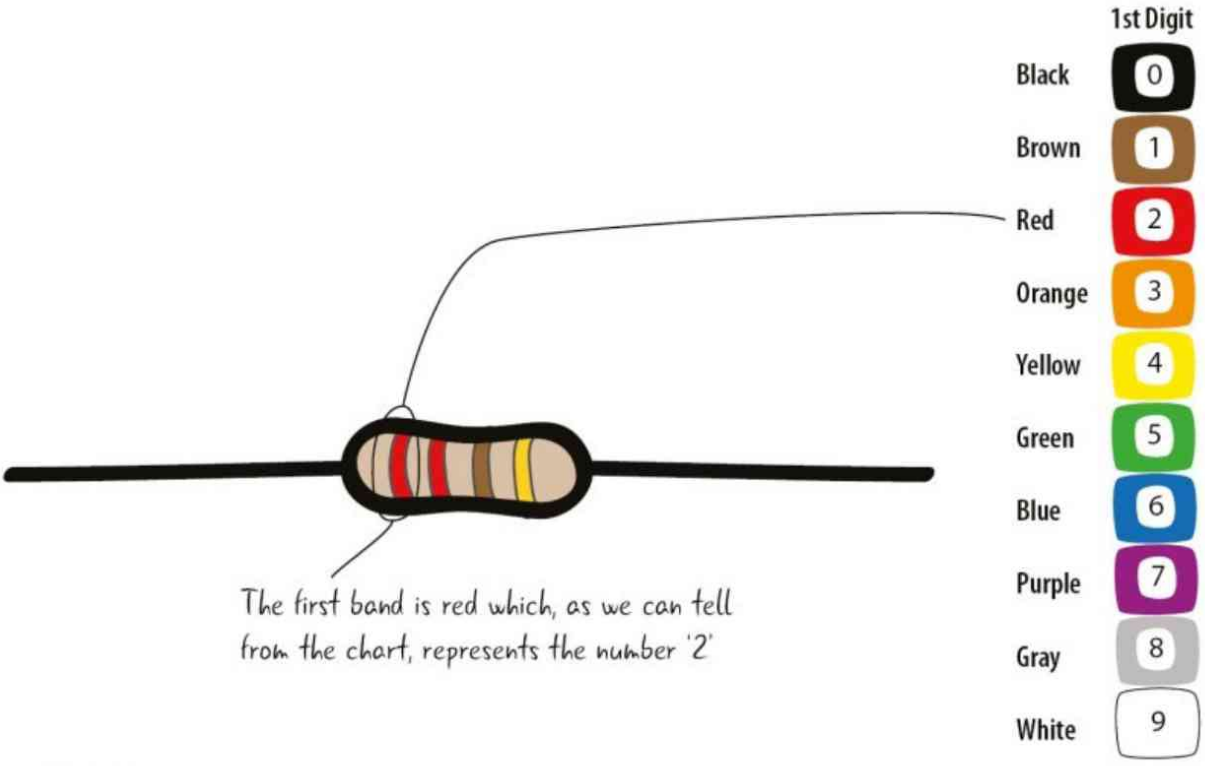


FIGURA A.5 Primera banda.

La segunda banda representa el segundo dígito más significativo. En esta resistencia, la segunda banda es también de color rojo. Como ves en la figura A.6, la tabla indica de nuevo que el color rojo de la segunda banda representa el número 2.

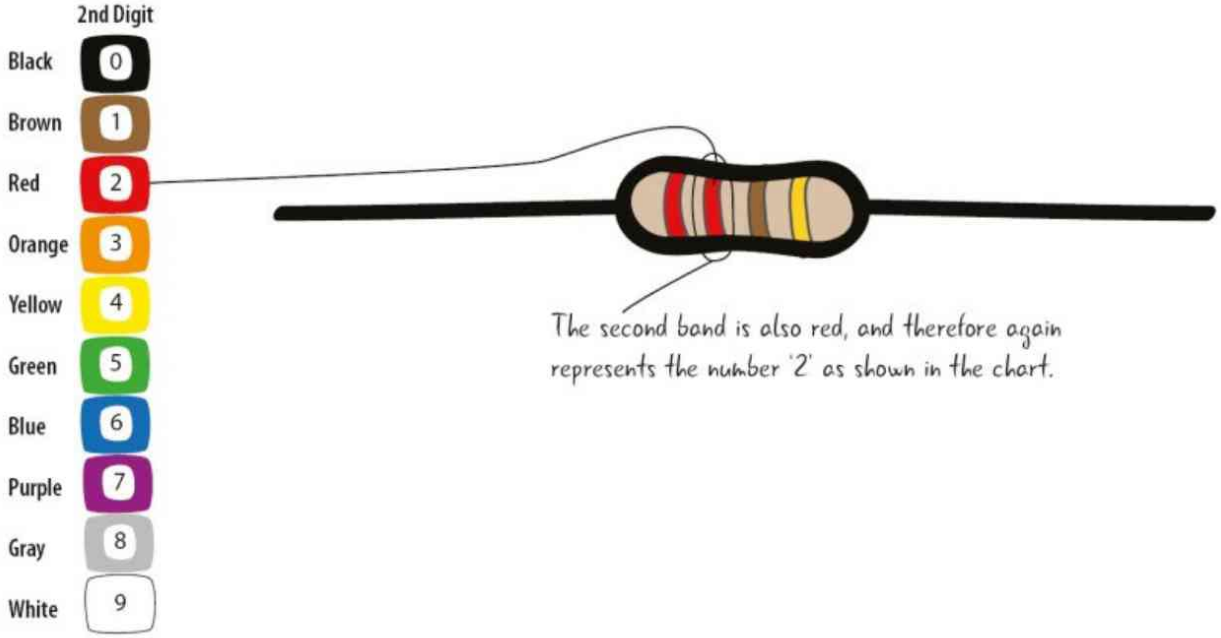


FIGURA A.6 Segunda banda.

Las dos primeras bandas juntas nos dan el número 22. Las dos primeras bandas de una resistencia siempre representarán un número entre 10 y 99. (Explicaremos qué significan estos números en breve). El tercer dígito es un poco diferente.

La tercera banda, mostrada en la figura A.7, tiene otro significado. En lugar de representar un número, como las dos primeras bandas, representa un multiplicador. Esta banda multiplica los valores de las dos primeras bandas por una potencia de 10. Puedes ver esto en la tercera fila de la tabla de la figura A.4. Para esta resistencia, la banda es marrón, y la tabla nos dice que se trata de un multiplicador por 10, o 10 a la primera potencia. Ahora que conoces estos tres valores, puedes calcular el valor total de la resistencia utilizando la sencilla fórmula que se muestra en la figura A.8: los dos primeros dígitos multiplicados por el multiplicador equivalen a la resistencia (en ohmios).

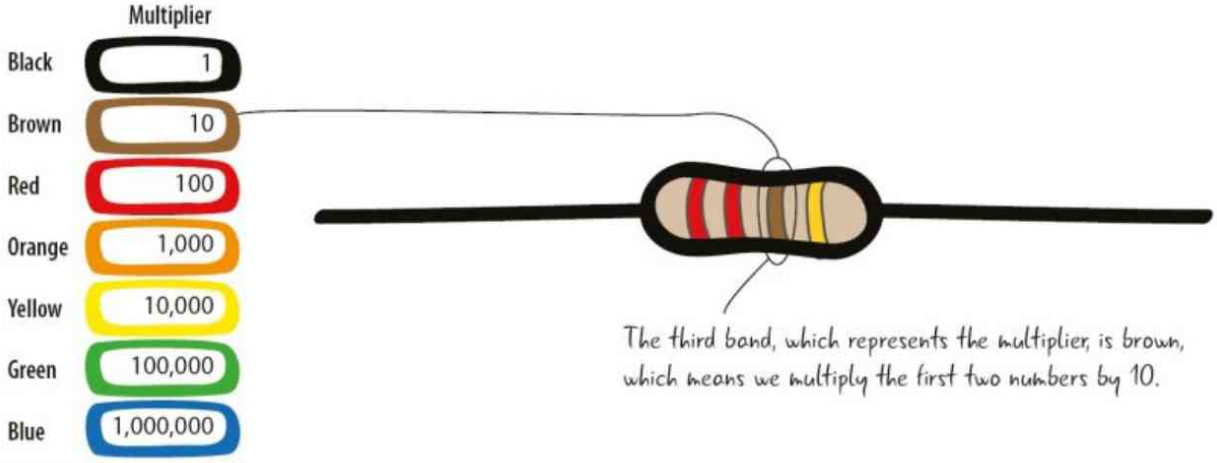


FIGURA A.7 Tercera banda.

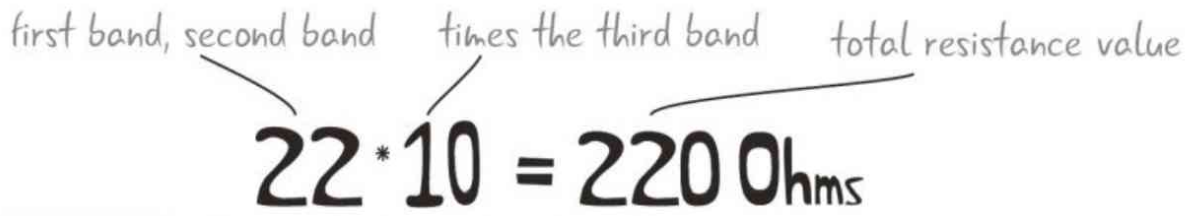


FIGURA A.8 Cálculo del valor de la resistencia.

Esto significa que la resistencia rojo-rojo-marrón tiene un valor de 220 ohmios. De hecho, todas las resistencias rojo-rojo-marrón tienen un valor de 220 ohmios.

La cuarta banda de la resistencia, en la figura A.9, representa la tolerancia del valor de la resistencia, o su margen de precisión. Con una banda dorada, la precisión es de más o menos el 5 %, lo que significa que la resistencia podría llegar a valer como máximo 231 ohmios (220x1,05) y como mínimo 209 (220x0,95). (Esta variación la causan las imperfecciones en el proceso de fabricación de la resistencia).

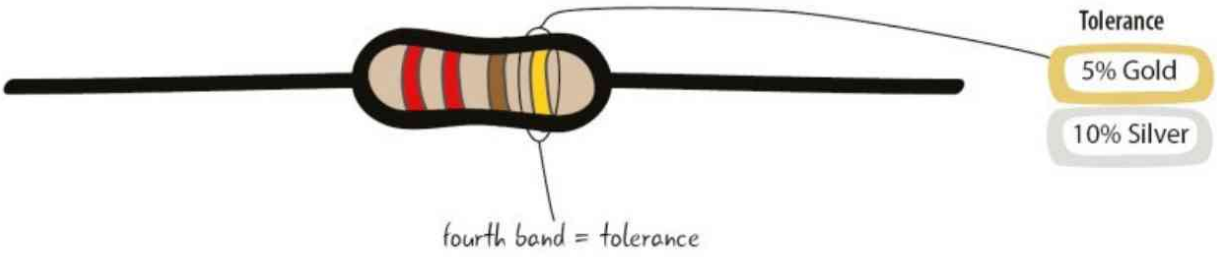


FIGURA A.9 Cuarta banda.

Ya que la cuarta banda va a ser siempre dorada o plateada, y esos colores no los utilizan las otras bandas, puedes utilizar la cuarta banda para orientar la resistencia correctamente.

¿PREGUNTAS?

P: ¿Los colores de las bandas son universales, o tendré que recordar lo que significa cada color?

R: Todas las resistencias utllizan los mismos códigos de colores estándar que hemos tratado aquí, independientemente del fabricante. No es necesario memorizarlos. Puedes encontrar fácilmente en línea la información de los colores, y existen varias aplicaciones gratuítas para teléfonos inteligentes y para todas las plataformas.

P: ¿Qué ocurre si las bandas son ilegibles o han pintado encima de ellas o las han borrado?

R: Si no se pueden leer las bandas de color, siempre puedes utilizar un multímetro para saber el valor de la resistencia.

P: ¿Con cuánta precisión debo utilizar las resistencias?

R: Es una buena pregunta. La electrónica y los componentes eléctricos para aficionados no son muy sensibles a pequeñas variaciones de las resistencias. La diferencia entre 209 ohmios y 231 ohmios no va a causar ningún problema al led. Sin embargo, usar una resistencia de valor mucho mayor (el doble o más) o mucho menor (la mitad o menos) dará lugar a problemas.

Note Aunque las resistencias con cuatro bandas son muy comunes, hay algunas que tienen un número diferente de bandas. Los colores indican los mismos números en las tres primeras bandas, pero los valores de tolerancia se calculan de otra forma.

Análisis de las bandas de color en otras resistencias

Veamos otra resistencia y vamos a evaluar las bandas de color para determinar el valor total de la resistencia. La resistencia de la figura A.10 tiene las bandas con los colores marrón, negro, naranja, y dorado.

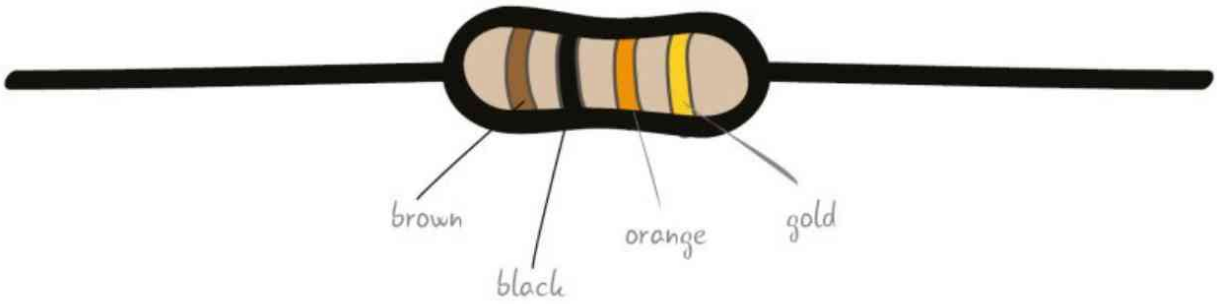


FIGURA A.10 Resistencia con las bandas etiquetadas.

El primer paso es orientar correctamente la resistencia. Para ello hay que situar la banda dorada en el lado derecho (figura A.11).

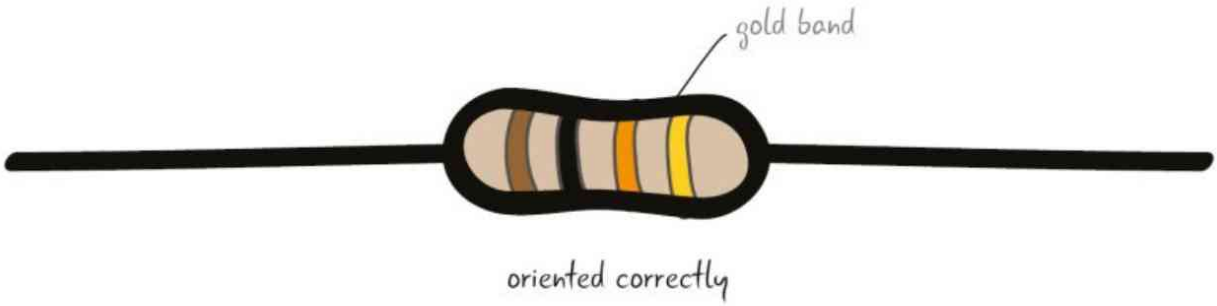


FIGURA A.11 Orientación de la resistencia.

consulta de la tabla de colores

Vuelve a consultar la figura A.4 para referenciar los valores de los colores. Siempre puedes consultar la tabla cuando necesites calcular el valor de una resistencia.

Lectura de las bandas

La primera banda es marrón, así que puedes consultar la tabla de colores y ver que el primer dígito es un 1 (figura A.12).

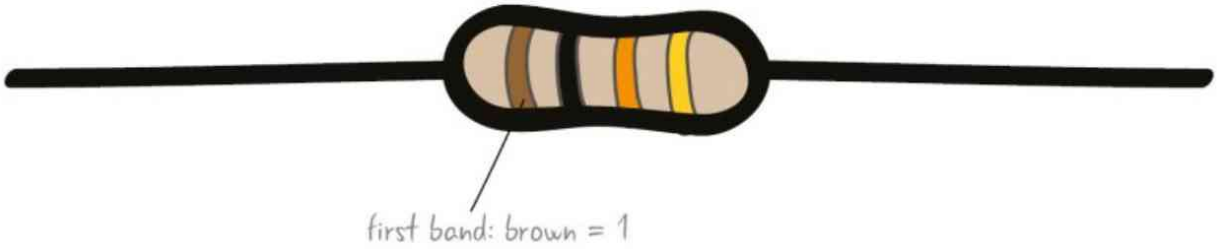


FIGURA A.12 Primera banda.

La siguiente banda de la resistencia es de color negro, lo que significa que el segundo dígito es un 0 (figura A.13).

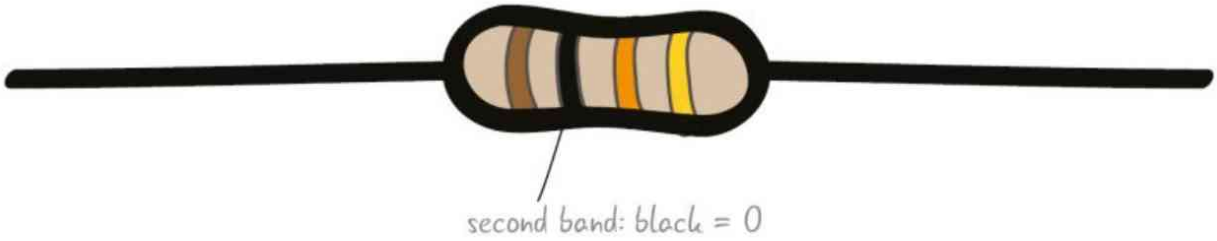


FIGURA A.13 Segunda banda.

La tercera banda es naranja, lo que quiere decir que el valor del multiplicador es 1 000 (figura A.14).

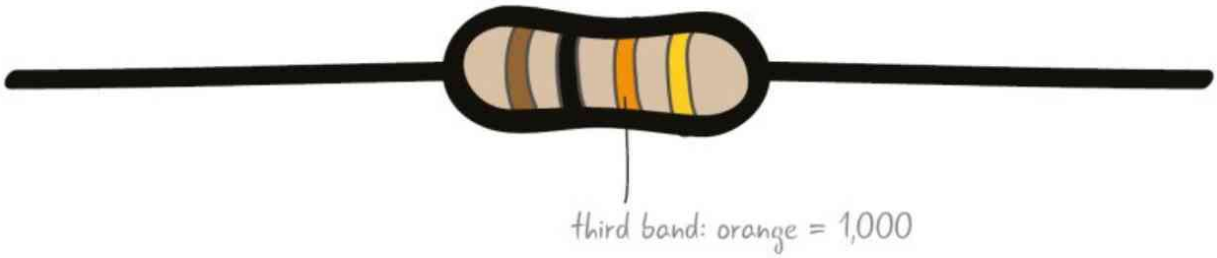


FIGURA A.14 Tercera banda.

Esto quiere decir que el valor de la resistencia es 10 veces 1 000. Es decir, la resistencia es de 10.000 ohmios, o como es más corriente verlo escrito, 10 KΩ (figura A.15).

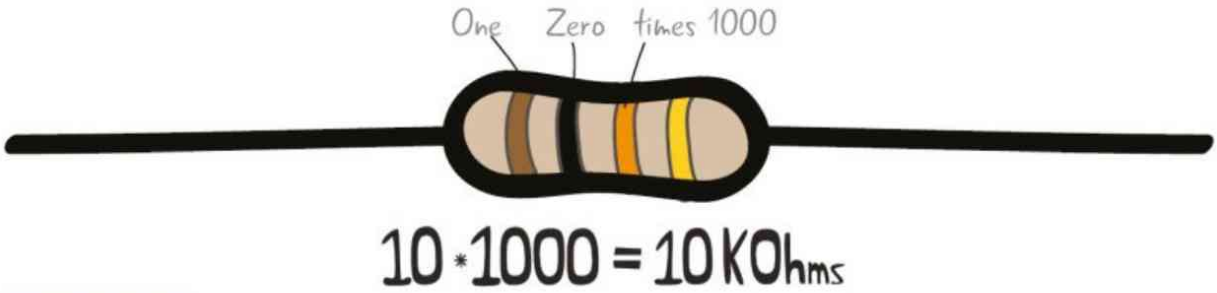


FIGURA A.15 Cálculo de la resistencia de 10 KΩ mediante las bandas de colores.

índice

Símbolos

D

J

K

L

U

V