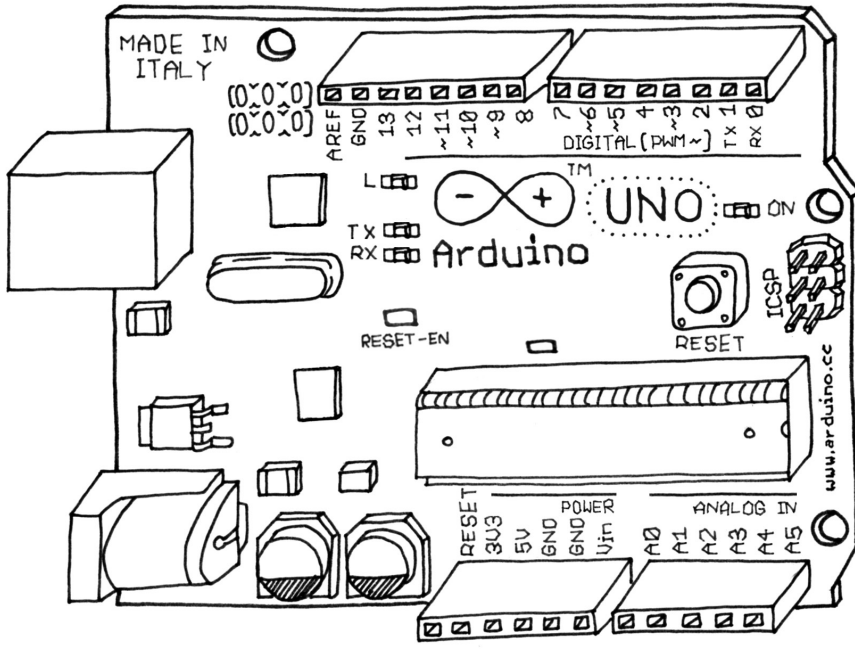
Introducción a Arduino

4.ª edición



La plataforma para prototipos electrónicos de código abierto

por **Massimo Banzi,** cofundador de Arduino,
y **Michael Shiloh**

ANAYA MULTIMEDIA

Índice de contenidos

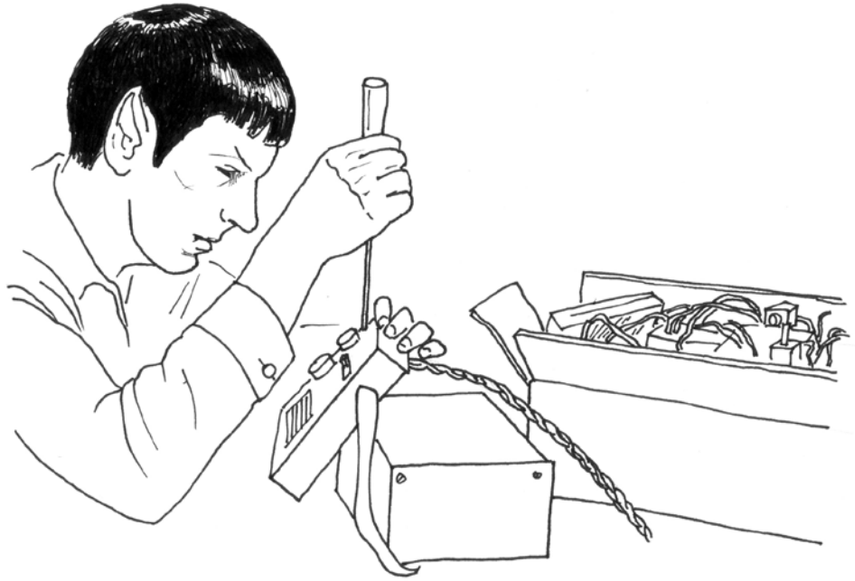
con mucha rapidez para poder motivarnos para dar el siguiente paso o, quizá, motivar a otra persona para que nos proporcione mucho dinero para dar el siguiente paso.

Por eso hemos desarrollado la creación de prototipos oportunista: ¿por qué dedicar tiempo y energía a construir desde cero, un proceso que requiere tiempo y un conocimiento técnico profundo, cuando podemos usar dispositivos ya preparados y transformarlos para aprovechar el duro trabajo que han hecho empresas grandes y buenos ingenieros?

Modificación experimental

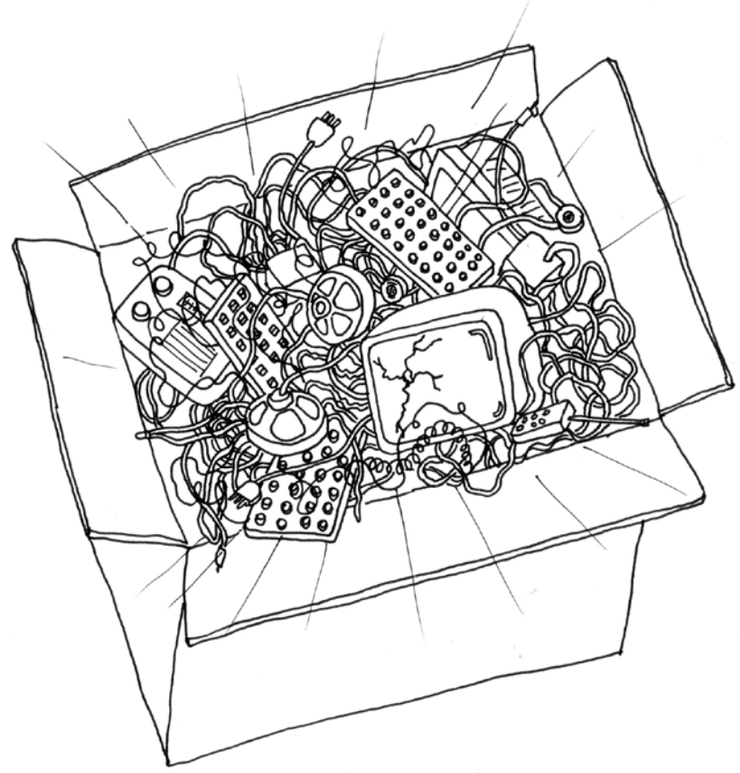
Creemos que es esencial jugar con la tecnología, explorar diferentes posibilidades directamente en el hardware y el software, a veces sin un objetivo muy definido.

Reutilizar la tecnología existente es una de las mejores formas de llevar a cabo modificaciones experimentales. Adquirir juguetes baratos o equipos antiguos desechados y transformarlos para que hagan algo nuevo es una de las mejores formas de obtener resultados geniales.



¡Nos encantan los trastos!

Hoy en día, la gente tira todo tipo de tecnología: viejas impresoras, ordenadores, máquinas de oficina raras, equipos técnicos e incluso muchos artículos militares. Siempre ha habido un gran mercado para esta tecnología sobrante, sobre todo entre los creadores más jóvenes o pobres y aquellos que están empezando. Este mercado resultó evidente en Ivrea, donde desarrollamos Arduino. En la ciudad se encontraba la sede de la empresa Olivetti, que fabricaba ordenadores desde los años sesenta; a mediados de los noventa, tiraron todo en desguaces de la zona, que se llenaron de piezas de ordenadores, componentes electrónicos y dispositivos raros de todo tipo. Pasamos innumerables horas allí, comprando toda clase de aparatos por unos pocos euros y adaptándolos para crear nuestros prototipos. Cuando uno puede comprar miles de altavoces por muy poco dinero, es muy probable que al final se le ocurra alguna idea. Acumule trastos y revíselos antes de empezar a construir algo de cero.



Sensores y actuadores

Los sensores y los actuadores son componentes electrónicos que permiten que un elemento electrónico interactúe con el mundo real.

Puesto que el microcontrolador es un ordenador muy simple, solo puede procesar señales eléctricas (algo parecido a los impulsos eléctricos que se envían entre las neuronas de nuestro cerebro). Para sentir la luz, la temperatura u otras cantidades físicas, necesita algo que las convierta en electricidad. En nuestro cuerpo, por ejemplo, el ojo convierte la luz en señales que se envían a nuestro cerebro utilizando nervios. En la electrónica, podemos utilizar un dispositivo simple llamado resistencia dependiente de la luz (LDR, *light-dependent resistor*), también conocido como fotorresistencia, que puede medir la cantidad de luz que incide en él y transmitirla como una señal que el microcontrolador puede entender.

Una vez que se han leído los sensores, el dispositivo tiene la información necesaria para decidir cómo reaccionar. El proceso de toma de decisiones está gestionado por el microcontrolador, y la reacción la llevan a cabo los actuadores. En nuestros cuerpos, por ejemplo, los músculos reciben señales eléctricas del cerebro y las convierten en movimiento. En el mundo de la electrónica, estas funciones puede realizarlas una luz o un motor eléctrico.

En las siguientes secciones, veremos cómo leer sensores de diferentes tipos y cómo controlar distintas clases de actuadores.

Hacer parpadear un LED

El *sketch* del LED parpadeante es el primer programa que debería ejecutar para comprobar si su placa Arduino funciona y está configurada correctamente. También suele ser el primer ejercicio de programación que realiza una persona que está aprendiendo a programar un microcontrolador. Un diodo emisor de luz (*light-emitting diode*, LED) es un pequeño componente electrónico que se parece un poco a una bombilla, pero es más eficiente y requiere un voltaje más bajo para funcionar.

La placa Arduino viene con un LED preinstalado. Está marcado con una L en la placa. Este LED preinstalado está conectado al pin número 13. Recuerde ese número, porque necesitaremos utilizarlo más adelante. También puede utilizar su propio LED; conéctelo como se muestra en la figura 4.2. Observe que está conectado al orificio etiquetado con el 13.

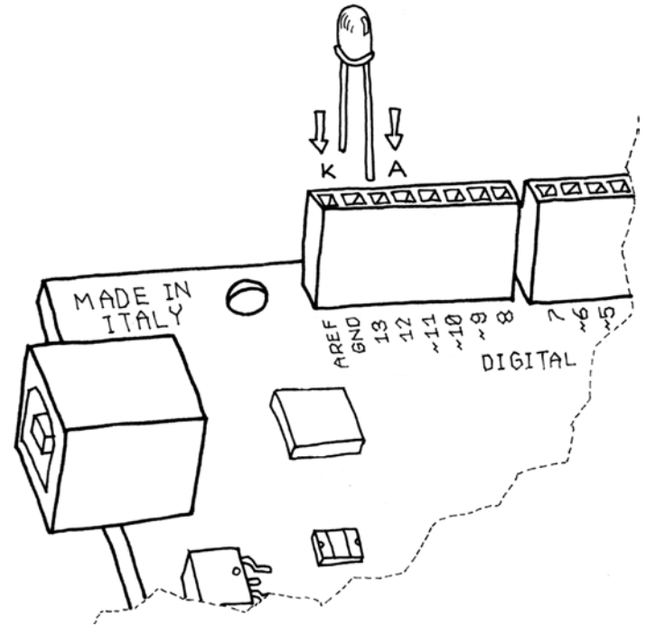


Figura 4.2. Conexión de un LED a Arduino.

Nota: Si planea dejar el LED encendido durante un periodo de tiempo largo, debería utilizar una resistencia, como se explicará en el capítulo 5.

La K indica el cátodo (negativo) o terminal corto; la A indica el ánodo (positivo) o terminal largo.

Una vez que el LED esté conectado, debe indicar a Arduino lo que tiene que hacer. Esto se hace mediante código: una lista de instrucciones que damos al microcontrolador para que haga lo que queremos. (Los términos código, programa y *sketch* hacen referencia a la misma lista de instrucciones).

En su ordenador, ejecute el IDE de Arduino (en el Mac, debería estar en la carpeta Aplicaciones; en Windows, el acceso directo debería estar en el escritorio o en el menú Inicio). Seleccione Archivo>Nuevo y se le pedirá que elija un nombre de archivo para el *sketch*: aquí es donde se guardará su *sketch* de Arduino. Llámelo Blinking_LED (LED parpadeante) y haga clic en **OK**. Después, escriba el *sketch*

Además, la mayoría de los LED tienen un borde aplanado en el lado del cátodo, como muestra la figura. Un modo sencillo de recordar esto es que la parte plana parece un signo menos, y que al terminal corto se le resta algo.

Como hemos mencionado en "Hacer parpadear un LED", en el capítulo 4, debería utilizar siempre una resistencia con un LED para evitar que este se queme. Una resistencia de cualquier valor entre 220 ohmios (rojo-rojo-marrón) y 1000 ohmios (marrón-negro-rojo) debería servir.

Después, cree un nuevo *sketch* en Arduino con el código que se muestra en el ejemplo 5.1. También puede descargarlo desde el enlace de códigos de ejemplo en la página del libro.[1]

Ejemplo 5.1. Disminuir y aumentar la intensidad de un LED, como en un ordenador Apple en reposo.

```
const int LED = 9; // el pin para el LED
int i = 0;         // usaremos esto para contar hacia arriba y hacia abajo

void setup() {
  pinMode(LED, OUTPUT); // indica a Arduino que LED es una salida
}

void loop(){

  for (i = 0; i < 255; i++) { // bucle de 0 a 254 (aumento de intensidad)
    analogWrite(LED, i);      // configura el brillo del LED
    delay(10); // Espera 10 ms porque analogWrite
               // es instantánea y no veríamos
               // ningún cambio si no hay retraso
  }

  for (i = 255; i > 0; i--) { // bucle de 255 a 1 (disminución de intensidad)
    analogWrite(LED, i); // configura el brillo del LED
    delay(10);           // Espera 10 ms
  }

}
```

Cargue el *sketch* y la intensidad del LED aumentará y disminuirá de manera continua. ¡Enhorabuena! Ha replicado una característica sofisticada de un ordenador portátil.

Puede que sea un pequeño desperdicio utilizar Arduino para algo tan simple, pero se puede aprender mucho de este ejemplo.

1. https://makezine.com/go/arduino-4e-github/.

Como hemos visto antes, `analogWrite()` cambia el brillo del LED. La otra parte importante es el bucle `for`: repite `analogWrite()` y `delay()` una y otra vez, usando cada vez un valor diferente para la variable `i`, de la siguiente manera.

El primer bucle `for` inicia la variable `i` con el valor de 0, y la aumenta hasta 255, lo que incrementa la intensidad del LED al máximo brillo.

El segundo bucle `for` inicia la variable `i` con el valor de 255, y lo reduce hasta 0, lo que disminuye la intensidad del LED hasta que está apagado por completo.

Después del segundo bucle `for`, Arduino empieza nuestra función `loop()` de nuevo.

`delay()` permite ralentizar un poco el proceso para que se pueda apreciar el cambio en el brillo; de lo contrario, se produciría demasiado rápido.

Vamos a usar esta información para mejorar nuestra lámpara.

Añada el circuito que hemos utilizado para leer un botón (en el capítulo 4) a esta placa de pruebas. Compruebe si puede hacerlo sin leer más allá de este párrafo, porque queremos que empiece a pensar en el hecho de que cada circuito elemental que mostramos aquí es un bloque de construcción para crear proyectos cada vez más grandes. Si necesita echar un vistazo a lo que viene después, no se preocupe; lo más importante es que dedique algún tiempo a pensar en cómo quedaría.

Para crear este circuito, necesita combinar el circuito que acaba de construir (mostrado en la figura 5.4) con el circuito del pulsador que se mostraba en la figura 4.6. Si quiere, puede construir ambos circuitos en partes diferentes de la placa de pruebas; tiene espacio de sobra.

Eche un vistazo al apéndice A para aprender más acerca de la placa de pruebas sin soldadura.

Si no está listo para probar esto, no se preocupe: solo tiene que conectar ambos circuitos a su Arduino como muestran las figura 4.6 y 5.4.

Ahora, hablemos del siguiente ejemplo: si solo tiene un pulsador, ¿cómo controla el brillo de la lámpara? Vamos a mostrarle otra técnica de diseño de interacción: detectar durante cuánto tiempo se ha pulsado el botón. Para ello, necesitamos actualizar el ejemplo 4.5 del capítulo 4 para añadir la atenuación. La idea es crear una interfaz en la que la acción de pulsar y soltar encienda y apague la luz, y la acción de pulsar y mantener cambie el brillo.

Eche un vistazo al *sketch* del ejemplo 5.2. Enciende el LED cuando se pulsa el botón y lo mantiene encendido después de soltarlo. Si el botón se mantiene pulsado, el brillo cambia.

7. Arduino Cloud

Arduino Cloud es un servicio en línea desarrollado por Arduino que permite a cualquier persona crear y gestionar dispositivos conectados utilizando solo un navegador. Sus módulos principales son:

- Web Editor, un IDE de Arduino completamente funcional implementado como un sitio web. Solo necesitamos un navegador web para escribir, compilar y cargar código de Arduino.
- IoT Cloud, un servicio que nos permite crear, programar y gestionar dispositivos conectados con una cantidad mínima de código (lo que hoy en día se denomina "Low Code"). Por ejemplo, podemos crear con facilidad un dispositivo para regar las plantas y controlarlo desde nuestro *smartphone* mientras nos tostamos al sol en la playa.
- Project Hub, un repositorio con miles de proyectos y tutoriales creados por la comunidad. Es un punto de partida estupendo si busca un gran proyecto para empezar.

Vamos a ver con más detalle cada módulo.

IDE Arduino Cloud

El IDE Arduino Cloud (conocido anteriormente como Arduino Create) es un entorno de desarrollo basado en la nube para Arduino que puede utilizarse con cualquier navegador de Internet moderno. Podemos iniciar sesión de forma muy simple desde cualquier parte del mundo en un IDE de Arduino totalmente funcional que almacena nuestro código en la nube. Resulta muy útil sobre todo

si usamos un Chromebook o utilizamos diferentes ordenadores y queremos tener la misma configuración en todas partes. En caso de emergencia, podemos tomar prestado el ordenador de otra persona y encontrar ahí todos nuestros archivos y bibliotecas. Una característica especial del IDE Cloud es que la carpeta de *sketches* de Arduino también puede albergar diagramas esquemáticos y diagramas de disposición. Solo tiene que meter una imagen `schematic.png` y `layout.png` en la carpeta y aparecerán como pestañas en el IDE. ¡Fácil! Otra ventaja interesante del IDE Cloud es que todas y cada una de las bibliotecas de Arduino que conocemos (¡miles!) están preinstaladas, así que no hay que dedicar tiempo a buscar bibliotecas e instalarlas; ya están ahí. Para empezar, solo tenemos que ir a `https://cloud.arduino.cc` y se nos pedirá que iniciemos sesión o creemos una cuenta de Arduino. Una vez dentro, vemos esta pantalla donde podemos encontrar el *sketchbook* y cualquier otra cosa que necesitemos.

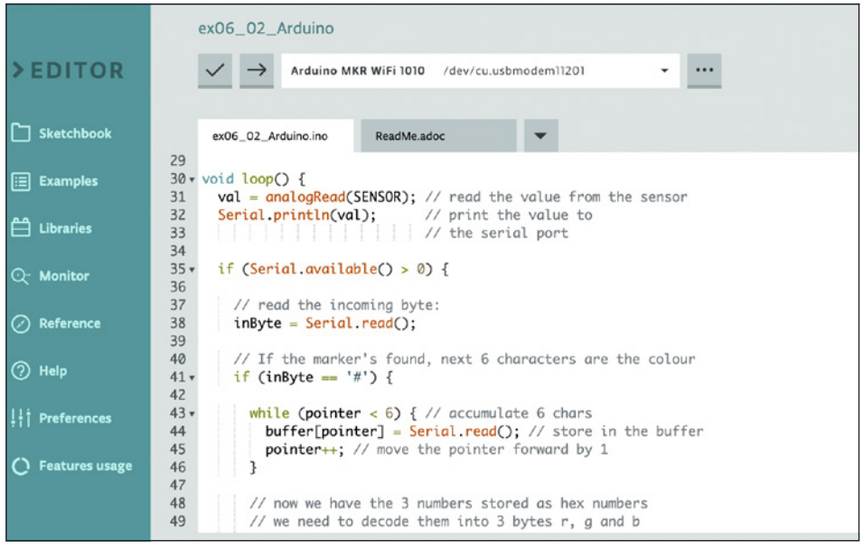


Figura 7.1. IDE Arduino Cloud.

Si utilizamos el nuevo Arduino IDE 2.0, podemos sincronizar el *sketchbook* que vemos en la nube con el de nuestro ordenador (más o menos similar a lo que ocurre con Dropbox y servicios parecidos). Si es la primera vez que utilizamos el IDE Cloud, se nos pedirá que instalemos un programa muy pequeño, el Arduino Create Agent, que permite a nuestro navegador comunicarse con puertos serie, de manera que podamos cargar nuestros *sketches* en placas reales.

Si usamos tipos determinados de placas, como MKR, Nano 33 IoT y similares, podemos incluir una función llamada OTA (actualizaciones *Over the Air*) que nos permite cargar código nuevo en la placa con una conexión a Internet. Bastante chulo, ¿no? Ofrecer una descripción completa y detallada de cómo funciona el IDE Cloud queda fuera del ámbito de este libro introductorio, pero puede encontrar más información en `https://cloud.arduino.cc`.

Project Hub

Una característica muy potente de Arduino Cloud es "Project Hub", un lugar en el que pueden encontrarse literalmente miles de tutoriales y proyectos para cualquier placa Arduino, que abarcan todo tipo de temas: de música a instalaciones, de domótica a jardinería, de dispensadores de pienso para mascotas a robots. Algunos de los proyectos son muy sofisticados y están muy bien documentados. Si busca un proyecto para empezar a crear con Arduino, ¡aquí es donde debe buscar!

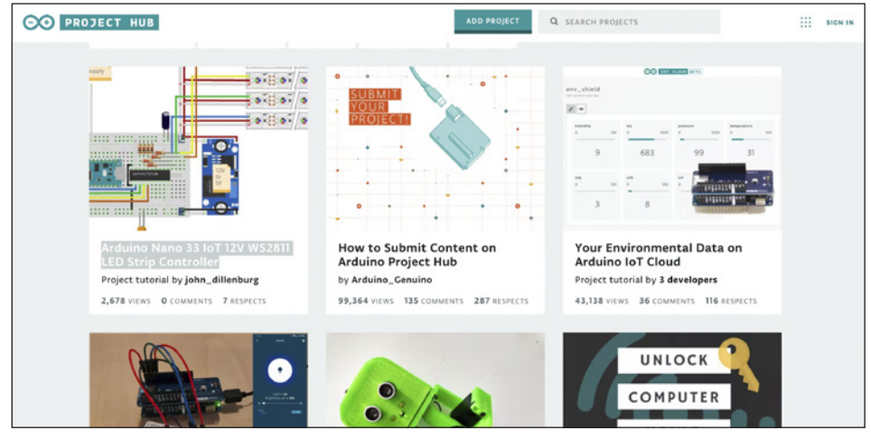


Figura 7.2. El Arduino Project Hub.

IoT Cloud

Un IoT Cloud es un servicio en línea que puede actuar como puente entre los dispositivos conectados y, por ejemplo, un cuadro de mandos web o incluso otros dispositivos. Si tiene una Arduino o placa similar compatible con IoT, el servicio IoT Cloud la detectará cuando se conecte.

Probar el sensor de temperatura y humedad

El DHT11 es un sensor de temperatura y humedad popular. Al igual que el RTC, es barato y fácil de usar con la Arduino. Según su ficha técnica, el DHT11 se conecta como muestra la figura 8.13. Fíjese en la resistencia *pull-up* en el pin de datos.

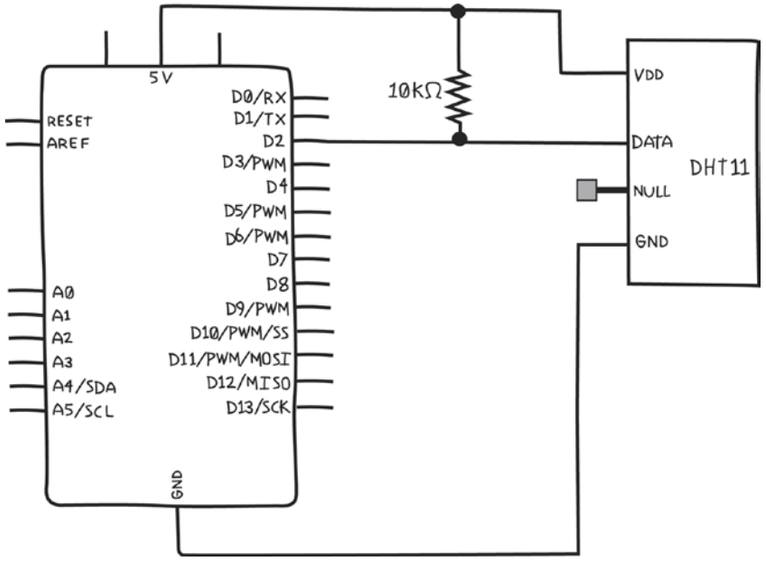


Figura 8.13. Diagrama esquemático para probar el sensor de temperatura y humedad DHT11.

Puesto que estamos añadiendo un componente que lo necesita, vamos a añadir otra resistencia de 10K ohmios a nuestra lista de la compra. Ya vamos por la versión 0.4:

- Añada una resistencia, 10K ohmios (para el sensor de temperatura y humedad).

Debido a la resistencia *pull-up*, no podemos usar el mismo truco que con el RTC (ponerlo directamente en la placa Arduino), así que vamos a tener que colocarlo en la placa de pruebas (figura 8.14).

Nota: El diagrama esquemático de un circuito es el mismo, al margen de si este se construye en una placa de pruebas o de otro modo.

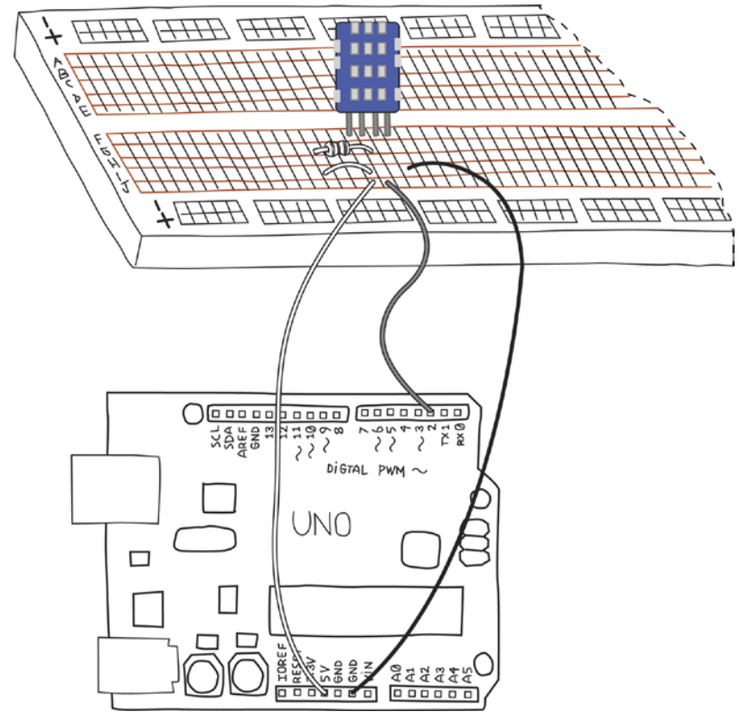


Figura 8.14. Diagrama pictórico del circuito para probar el sensor de temperatura y humedad DHT11.

Puede instalar la biblioteca DHT11 de Adafruit igual que ha hecho con la biblioteca RTClib.

Compruebe que ha instalado bien la biblioteca abriendo el ejemplo DHTtester en la categoría DHT sensor library de los ejemplos y, a continuación, haga clic en el botón Verificar (consulte la figura 4.3). Si recibe el mensaje "Compilado", la biblioteca se ha instalado correctamente.

Antes de cargar el *sketch* en la Arduino, fíjese en que el ejemplo es compatible con tres modelos diferentes de sensores DHT: DHT11, DHT21 y DHT22. Para seleccionar de manera correcta el modelo adecuado, se define una constante llamada DHTTYPE para que sea DHT11, DHT21 o DHT22:

```
// ¡Descomente el tipo que esté utilizando!
//#define DHTTYPE DHT11   // DHT 11
#define DHTTYPE DHT22     // DHT 22  (AM2302)
//#define DHTTYPE DHT21   // DHT 21 (AM2301)
```

```
void printMenu() {
  Serial.println(
    "Please enter P to print the current settings");
  Serial.println(
    "Please enter S2N13:45 to set valve 2 ON time to 13:34");
}
```

Nota: Siempre que un bloque de código vaya a utilizarse más de una vez, es un buen candidato para convertirse en una función, no importa lo corto que sea.

Por fin, el ejemplo 8.4 muestra el *sketch* completo.

Ejemplo 8.4. El *sketch* del sistema de riego.

```
/* Ejemplo 8.4. El sketch del sistema de riego */

#include <Wire.h>   // Biblioteca Wire, usada por la biblioteca RTC
#include "RTClib.h" // Biblioteca RTC
#include "DHT.h"    // Biblioteca del sensor de temperatura/humedad DHT

// Uso de pines analógicos
const int RTC_5V_PIN = A3;
const int RTC_GND_PIN = A2;

// Uso de pines digitales
const int DHT_PIN  = 2;     // sensor de temperatura/humedad
const int WATER_VALVE_0_PIN = 8;
const int WATER_VALVE_1_PIN = 7;
const int WATER_VALVE_2_PIN = 4;

const int NUMBEROFVALVES = 3; // Cuántas válvulas tenemos
const int NUMBEROFTIMES = 2;  // Cuántas horas tenemos

// Matriz para guardar horas ON y OFF para cada válvula
// Guarda esta hora como el número de minutos transcurridos desde medianoche
// para facilitar los cálculos
int onOffTimes [NUMBEROFVALVES][NUMBEROFTIMES];
int valvePinNumbers[NUMBEROFVALVES];

// Qué columna es la hora ON y cuál es la hora OFF
const int ONTIME = 0;
const int OFFTIME = 1;

#define DHTTYPE DHT11
DHT dht(DHT_PIN, DHTTYPE); // Crea un objeto DHT

RTC_DS1307 rtc;    // Crea un objeto RTC

// Variables globales establecidas y usadas en diferentes funciones
```



```
DateTime dateTimeNow; // para guardar resultados del RTC

float humidityNow; // resultado de humedad del sensor DHT11

void setup(){

  // Alimentación y tierra al RTC
  pinMode(RTC_5V_PIN, OUTPUT);
  pinMode(RTC_GND_PIN, OUTPUT);
  digitalWrite(RTC_5V_PIN, HIGH);
  digitalWrite(RTC_GND_PIN, LOW);

  // Inicializa la biblioteca Wire
  #ifdef AVR
    Wire.begin();
  #else
    // Los pines I2C de la shield se conectan al bus I2C alternativo en
Arduino Due
    Wire1.begin();
  #endif

  rtc.begin();        // Inicializa el objeto RTC
  dht.begin();        // Inicializa el objeto DHT
  Serial.begin(9600); // Inicializa el objeto Serial

  // Establece los números de pines de las válvulas de agua en la matriz
  valvePinNumbers[0] = WATER_VALVE_0_PIN;
  valvePinNumbers[1] = WATER_VALVE_1_PIN;
  valvePinNumbers[2] = WATER_VALVE_2_PIN;

  // y establece todos esos pines como salidas
  for (int valve = 0; valve < NUMBEROFVALVES; valve++) {
    pinMode(valvePinNumbers[valve], OUTPUT);
  }

};

void loop() {

  // Recuerda brevemente al usuario los posibles comandos
  Serial.print("Type 'P' to print settings or ");
  Serial.println("'S2N13:45' to set valve 2 ON time to 13:34");

  // Obtiene (e imprime) la fecha, la hora,
  // la temperatura y la humedad actuales
  getTimeTempHumidity();

  checkUserInteraction(); // Comprueba si hay solicitud del usuario

  // Comprueba si es hora de activar o desactivar la válvula
  checkTimeControlValves();

  delay(5000); // No es necesario hacer esto con demasiada frecuencia
}
```

Algunas placas de prueba sin soldadura tienen filas adicionales: dos en la parte superior y dos en la inferior, a menudo señaladas con franjas rojas y azules y, a veces, marcadas con + y -. Estas filas se conectan en horizontal y están pensadas para cualquier señal eléctrica que se utilice con frecuencia. Son perfectas para 5V o GND, que también son las conexiones más comunes en los proyectos de este libro, y en casi todos los proyectos electrónicos. Estas filas suelen denominarse rieles o *buses*.

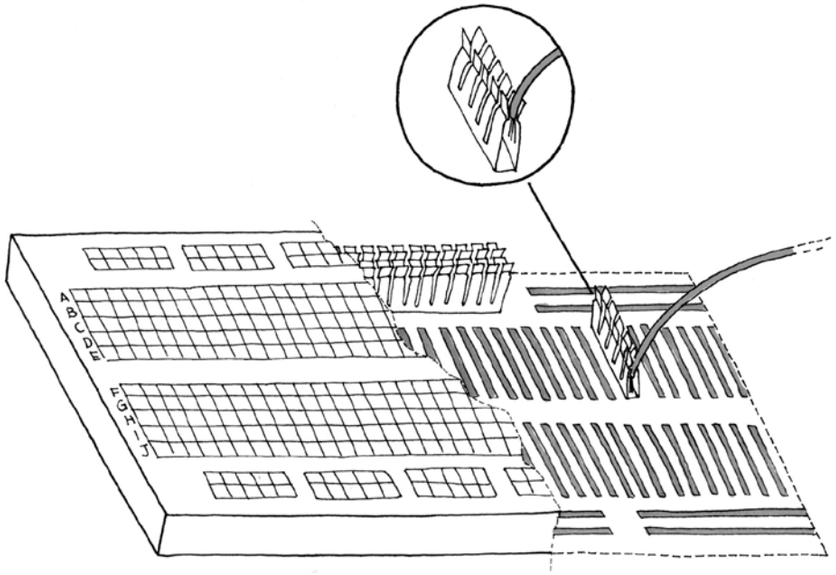


Figura A.1. La placa de pruebas sin soldadura.

Si conectamos la fila roja (o la marcada como +) a 5V en nuestra placa Arduino y la azul (o la marcada como -) a GND en nuestra placa Arduino, siempre tendremos 5V y GND cerca de cualquier punto de la placa de pruebas.

Aparece un buen ejemplo de estos rieles en el capítulo 6.

Nota: En algunas placas de pruebas, los rieles no son continuos y se dividen en la parte central. A veces, esto se indica mediante un hueco en la franja roja o azul y, a veces, mediante un espacio entre pines un poco más grande de lo habitual. Como es fácil olvidar esto, mucha gente deja un cable puente de forma permanente para cerrar este espacio en cada fila.

Algunos componentes, como las resistencias, los condensadores y los LED, tienen patillas largas y flexibles que pueden doblarse para llegar a orificios en diferentes puntos.

Sin embargo, otros componentes, como los chips, tienen patillas (conocidas como pines entre los amantes de la tecnología) que no pueden moverse. Estos pines tienen casi siempre un espacio de 2,54 mm, así que los orificios de la placa de pruebas sin soldadura utilizan esa distancia.

La mayoría de los chips tienen dos filas de pines y, si las columnas de la placa de pruebas estuviesen conectadas de forma continua hasta el final los pines de un lado del chip estarían conectados (mediante la placa de pruebas) a los pines del otro lado. Esa es la razón de que haya un espacio en el medio, que interrumpe cada línea vertical de orificios. Si coloca un chip de manera que pase por este hueco, los pines de un lado no se conectarán con los del otro. Ingenioso, ¿verdad?

Nota: Algunas placas de pruebas tienen letras que indican las filas y números que indican las columnas. No haremos referencia a esas etiquetas, ya que no todas las placas de pruebas son iguales. Cuando mencionemos un número de pin, estaremos refiriéndonos al pin en la Arduino, no en la placa de pruebas.

Introducción a Arduino

4.ª edición

Arduino es la plataforma para prototipos electrónicos de código abierto que se encuentra en el corazón del mundo Maker. Esta introducción exhaustiva, actualizada para el lanzamiento del IDE de Arduino más reciente y las nuevas placas basadas en ARM, le ayudará a empezar a crear prototipos de inmediato. Desde la obtención de los componentes requeridos a la adición de los últimos toques al proyecto, ¡toda la información que necesita está aquí!

¡Empezar a usar Arduino es muy fácil! Para utilizar los ejemplos introductorios de esta guía, solo necesita la Arduino Uno o Nano, junto con un cable USB y un LED. El entorno de desarrollo de Arduino, gratuito y fácil de utilizar, puede ejecutarse en Mac, Windows y Linux.

Únase a los cientos de miles de aficionados que han descubierto esta increíble (y educativa) plataforma.

En este libro aprenderá acerca de:

- Diseño de interacción e informática física.
- La placa Arduino y su entorno de software.
- Conceptos básicos de electricidad y electrónica.
- Creación de prototipos en una placa de pruebas sin soldadura.
- Dibujo de un diagrama esquemático.
- Uso del IDE Cloud y la nueva Arduino IoT Cloud.
- Construcción de un sistema de riego de plantas personalizado.
- Creación de un "choque de puños" por Internet con Wi-Fi.

Escrito por Massimo Banzi, cofundador de Arduino, y Michael Shiloh, líder educativo de Arduino.

ISBN 978-84-415-4705-6

2315183

9 788441 547056

ANAYA MULTIMEDIA

Make:

www.anayamultimedia.es