Una ventaja obvia de este enfoque es que es fácil cambiar el mensaje simplemente alterando la matriz de duraciones (durations[i]). En el Proyecto 3, llevaremos el uso de los arrays un paso más allá para crear un destello de código Morse de propósito más general.

Proyecto 3
Traductor de código Morse

En este proyecto vamos a usar el mismo hardware que en los Proyectos 1 y 2, pero vamos a escribir un nuevo **sketch** que nos va a permitir escribir una frase en nuestro ordenador y hacer que la placa Arduino lo convierta en los puntos y rayas apropiados del código Morse.

La Figura 3-1 muestra el traductor de código Morse en acción. El contenido del cuadro de mensaje será transmitido como puntos y rayas mediante el LED.

Para ello, haremos uso de lo que hemos aprendido acerca de **arrays** (matrices) y **strings** (cadenas), y también aprenderemos algo acerca de cómo enviar mensajes desde el ordenador a la placa Arduino a través del cable USB.

Para este proyecto, necesitará los mismos componentes que para el Proyecto 1 y 2. De hecho, el hardware es exactamente el mismo; vamos simplemente a modificar el **sketch** del Proyecto 1.

COMPONENTES Y EQUIPO		
	Descripción	**Apéndice A**
	Placa Arduino UNO o Duemilanove o similar	1
D1	LED rojo de 5 mm	23
R1	Resistencia 270 Ω 0,5W	6

Hardware

Por favor, remítase al Proyecto 1 para el conexionado de este proyecto.

Puede simplemente conectar la resistencia y el LED directamente en los conectores de Arduino o utilizar la placa de pruebas (**protoboard**) (véase el Capítulo 1). Puede incluso cambiar sólo la variable

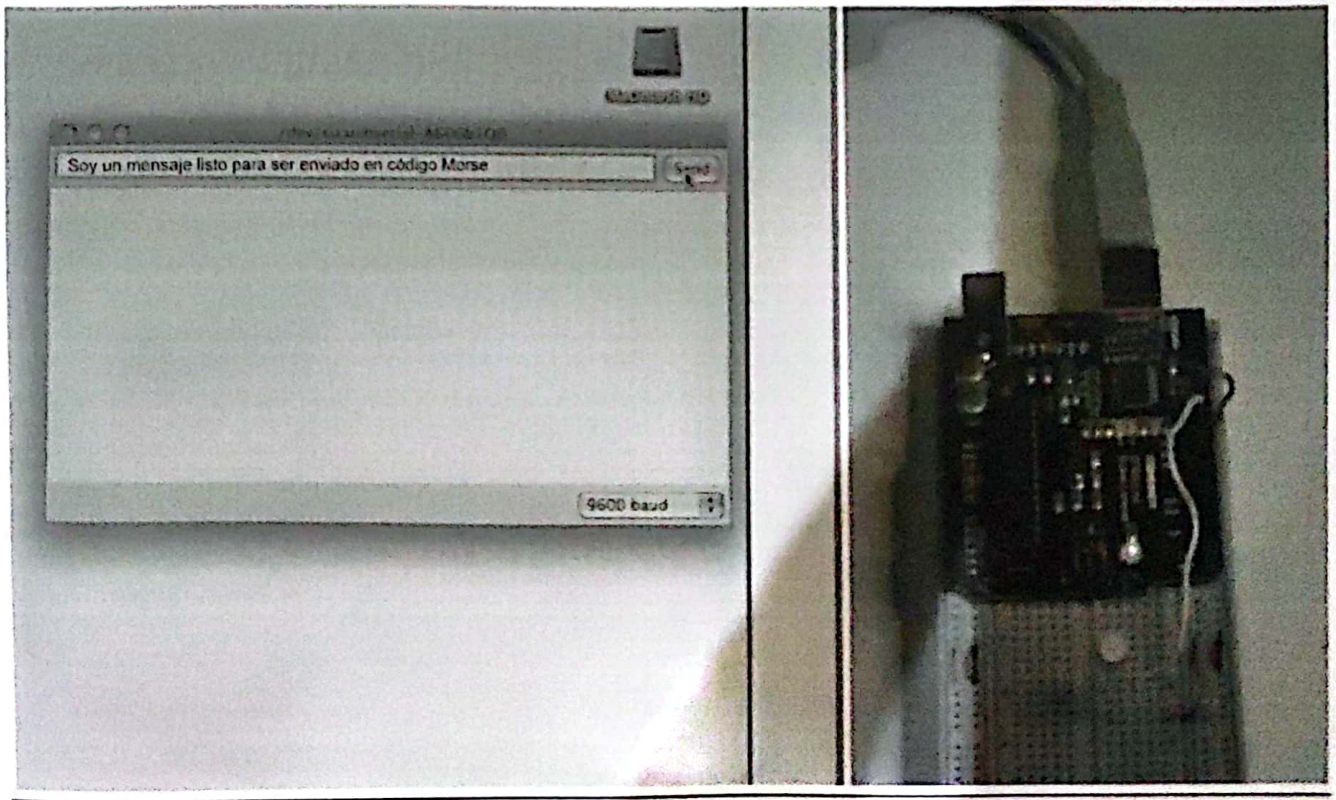

Figura 3-1 Traductor de código Morse

ledPin en el **sketch** para que sea el pin 13, y así utilizar el LED integrado en la placa, no necesitando entonces ningún componente externo.

Software

Las letras del código Morse se muestran en la Tabla 3-1.

Algunas de las normas del código Morse son que la duración de una raya es tres veces la de un punto, el tiempo que transcurre entre cada raya o punto es igual a la duración de un punto, el espacio entre dos letras tiene la misma longitud que la raya, y el espacio entre dos palabras tiene la misma duración que siete puntos.

En aras de este proyecto, no nos vamos a preocupar de la puntuación, aunque sería un interesante ejercicio el que intentara añadir esto al **sketch**. Para obtener una lista completa de todos los caracteres Morse, véase: http://en.wikipedia.org/wiki/Morse_code.

TABLA 3-1 Alfabeto del código Morse

A	.-	N	-.	0	——-
B	-...	O	—-	1	.——
C	-.-.	P	.--.	2	..---
D	-..	Q	--.-	3	...--
E	.	R	.-.	4	....-
F	..-.	S	...	5	.....
G	--.	T	-	6	-....
H	....	U	..-	7	--...
I	..	V	...-	8	---..
J	.---	W	.--	9	----.
K	-.-	X	-..-		
L	.-..	Y	-.--		
M	--	Z	--..		

Puede ver el **sketch** en el Listado del Proyecto 3. A continuación damos una explicación de su funcionamiento.

LISTADO DE PROYECTO 3

```
int ledPin = 12;

char* letters[] = {
  ".-", "-...", "-.-.", "-..", ".", "..-.", "--.", "....", "..",    // A-I
  ".---", "-.-", ".-..", "--", "-.", "---", ".--.", "--.-", ".-.",  // J-R
  "...", "-", "..-", "...-", ".--", "-..-", "-.--", "--.."          // S-Z
};

char* numbers[] = {"-----", ".----", "..---", "...--", "....-", ".....", "-....",
    "--...", "---..", "----."};

int dotDelay = 200;

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

void loop()
{
  char ch;
  if (Serial.available())      // ¿hay algo que leer del USB?
```

LISTADO DE PROYECTO 3 (continuación)

```
  {
    ch = Serial.read();              // leer una letra
    if (ch >= 'a' && ch <= 'z')
    {
      flashSequence(letters[ch - 'a']);
    }
    else if (ch >= 'A' && ch <= 'Z')
    {
      flashSequence(letters[ch - 'A']);
    }
    else if (ch >= '0' && ch <= '9')
    {
      flashSequence(numbers[ch - '0']);
    }
    else if (ch == ' ')
    {
     delay(dotDelay * 4);            // espacio entre palabras
    }
  }
}

void flashSequence(char* sequence)
{
   int i = 0;
   while (sequence[i] != NULL)
   {
       flashDotOrDash(sequence[i]);
       i++;
   }
   delay(dotDelay * 3);              // espacio entre letras
}

void flashDotOrDash(char dotOrDash)
{
  digitalWrite(ledPin, HIGH);
  if (dotOrDash == '.')
  {
    delay(dotDelay);
  }
  else // debe ser una -
  {
    delay(dotDelay * 3);
  }
  digitalWrite(ledPin, LOW);
  delay(dotDelay);                   // espacio entre destellos
}
```

Hacemos el seguimiento de nuestros puntos y rayas usando matrices de cadenas (**arrays** de **strings**). Contamos con dos de estos, uno para las letras (**letters**) y otro para los números. Por lo tanto, para averiguar lo que necesitamos para hacer parpadear el LED con la primera letra del alfabeto (A), obtendremos la primera posición de **string letters[0]** -recuerde, el primer elemento de una matriz es el elemento 0, no el 1.

Hemos definido la variable **dotDelay**, por lo tanto, si queremos hacer que nuestro código Morse parpadee más rápido o más lento, podemos cambiar este valor, ya que todas las duraciones se han definido como múltiplos del tiempo de duración de un punto.

La función **setup** es prácticamente igual a la de nuestros proyectos anteriores; sin embargo, esta vez estamos recibiendo comunicaciones del puerto USB, por lo que debemos añadir el comando:

```
Serial.begin(9600);
```

Esto indica a la placa Arduino que configure la velocidad de comunicación a través de USB para que sean **9600 baudios**. Esto no es que sea muy rápido, pero es suficiente para nuestros mensajes de código Morse. También es una buena velocidad de configuración porque ésa es la velocidad predeterminada utilizada por el software Arduino en el ordenador.

En la función **loop**, vamos a ver repetidamente si hemos enviado letras a la conexión USB y si tenemos que procesar la letra. La función de **Arduino Serial.available()** será verdadera si hay un carácter para convertir en código Morse y la función **Serial.read()** nos dará ese carácter, que asignaremos a una variable llamada **ch** que acabamos de definir dentro del **loop**.

Luego tenemos una serie de sentencias **if** (si..) que determinan si el carácter es una letra mayúscula, minúscula, o un carácter espacio de separación de dos palabras. Si miramos a la primera instrucción **if**, estamos comprobando si el valor del carácter es mayor o igual a "a" e inferior o igual a "z". Si es el caso, podemos buscar la secuencia de rayas y puntos de destello utilizando la matriz de letras (**letters[ ]**) que hemos definido en la parte superior del **sketch**.

Debemos determinar qué secuencia utilizar de la matriz restando "a" del carácter de **ch**. A primera vista, podría parecer extraño restar una letra de otra, pero es perfectamente aceptable hacer esto en C. Así, por ejemplo, "a" - "a" es igual a 0, mientras que "d" - "a" nos dará una respuesta de 3. Por lo tanto, si la letra que leemos en las conexiones USB fuera **f**, calculamos "f" - "a", lo que nos da 5 como la posición en el array **letters[ ]**. Si ahora miramos **letters[5]**, nos dará la cadena "..-.". A continuación pasamos esta cadena (string) a una función llamada **flashSequence**.

La función **flashSequence** lo que hace es recorrer cada una de las partes de la secuencia y realizar un destello en forma de raya o punto. En C, las **strings** (cadenas) tienen un código especial en el extremo que indica el final de la cadena, y esto se conoce como **NULL**. Por tanto, la primera cosa que hace **flashSequence** consiste en definir una variable llamada "i". Esto indicará la posición actual en la cadena de puntos y rayas, comenzando en la posición 0. El bucle **while** seguirá ejecutándose hasta que llegue a **NULL**, al final de la cadena.

Dentro del bucle **while**, primero hacemos que destelle el punto o raya en el que nos encontremos utilizando una función que vamos a tratar en un momento y, a continuación, agregaremos 1 a **"i"** para, sin interrupción, continuar dando vueltas en el **loop**, haciendo parpadear cada punto o raya en su momento hasta que lleguemos al final de la cadena.

La función final que hemos definido es **flashDotOrDash**; esto sólo enciende el LED y luego utiliza una instrucción **if** para: o bien crear un retardo durante la duración de un solo punto (si el carácter es un punto), o un retardo de un período tres veces superior a esa duración (si el carácter es una raya), antes de volver a apagar el LED de nuevo.

Pongamos todo junto

Cargue el **sketch** terminado del Proyecto 3 desde su **Arduino Sketchbook** y descárguelo en su placa (véase el Capítulo 1).

Para utilizar el traductor de código Morse, necesitamos emplear una parte del software Arduino llamado **Serial Monitor**. Esta ventana le permite escribir mensajes que se envían a la placa Arduino, así como ver los mensajes con que la placa Arduino

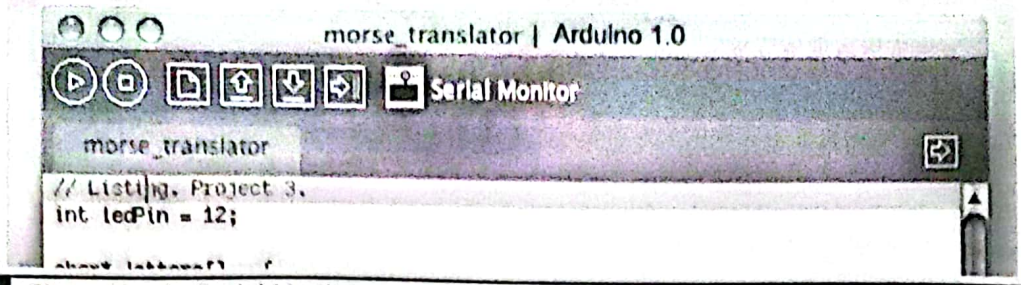

Figura 3-2 Ejecución de Serial Monitor

pueda responder.

Serial Monitor se inicia haciendo clic en el icono más a la derecha que aparece resaltado en la Figura 3-2.

Serial Monitor (consulte la Figura 3-3) consta de dos partes. En la parte superior, hay un campo en el que se puede escribir una línea de texto, que se enviará a la placa cuando se hace clic en **Send** o se pulse **RETURN**.

Debajo se encuentra un área mayor en el que se mostrará cualquier mensaje procedente de la placa Arduino. En la parte inferior de la ventana hay una lista desplegable en la que puede seleccionar la velocidad a la que se deben enviar los datos. Cualquier cosa que seleccione aquí debe coincidir con la velocidad en baudios que se especifique en el mensaje **startup** del programa de comandos (**script**). Nosotros utilizamos 9600, que es el valor predeterminado, con lo que no hay necesidad de cambiar nada.

Por tanto, todo lo que tenemos que hacer es iniciar **Serial Monitor**, escribir algún texto en el campo **Send** y pulsar la tecla **RETURN**. Deberíamos recibir nuestro mensaje en forma de destellos en código Morse.

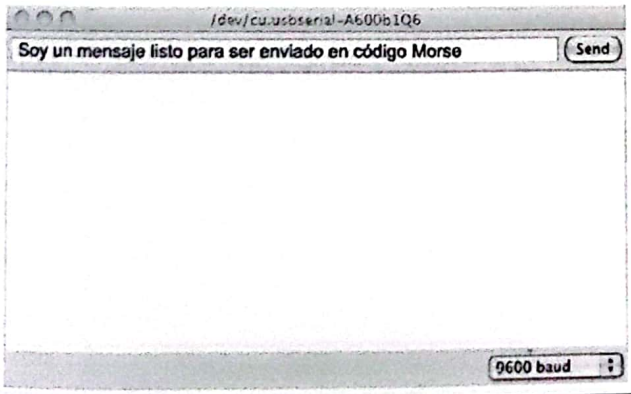

Figure 3-3 La ventana de Serial Monitor

Proyecto 4
Traductor de código Morse de gran intensidad

El pequeño LED del Proyecto 3 es poco probable que sea visible desde un buque en el horizonte que se ve atraído por nuestro falso mensaje de socorro de aficionado a la robótica. Por lo tanto, en este proyecto vamos a incrementar la potencia y utilizar un LED Luxeon de 1 W. Estos LED son extremadamente brillantes y toda la luz proviene de una pequeña zona en el centro, por lo que, para evitar cualquier posibilidad de daño en la retina, no debe mirarlo directamente.

También echaremos un vistazo para averiguar cómo, empleando un soldador, podemos convertir este proyecto en una **shield** que se pueda conectar a nuestra placa Arduino.

COMPONENTES Y EQUIPO		
	Descripción	**Apéndice A**
	Placa Arduino UNO o Duemilanove o similar	1
D1	LED Luxeon de 1 W	30
R1	Resistencia 270 Ω 0,5W	6
R2	Resistencia 1 W 4 Ω	16
T1	Transistor potencia BD139	41
	Kit de protoshield (opcional)	3