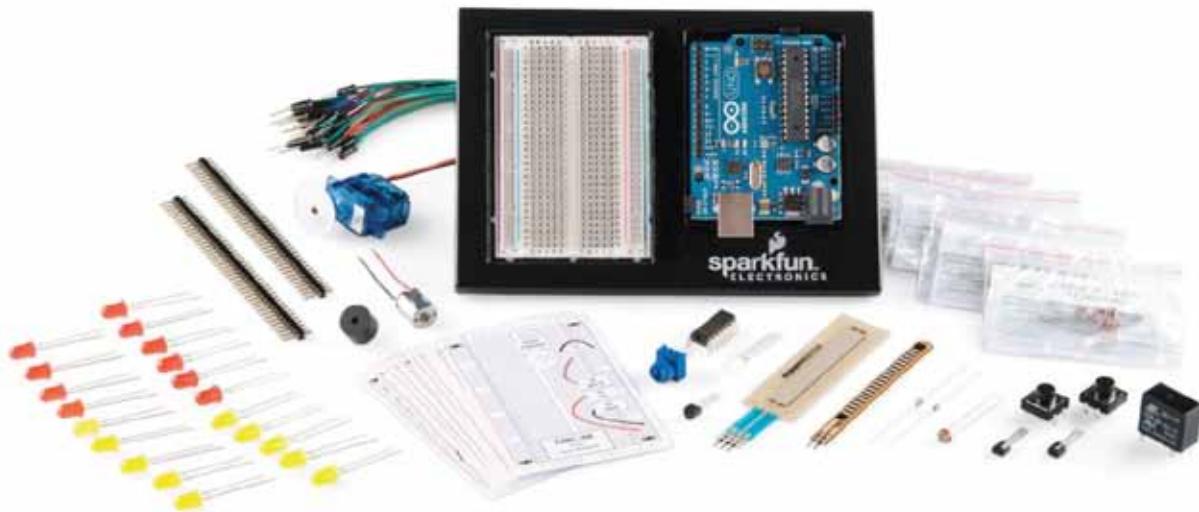
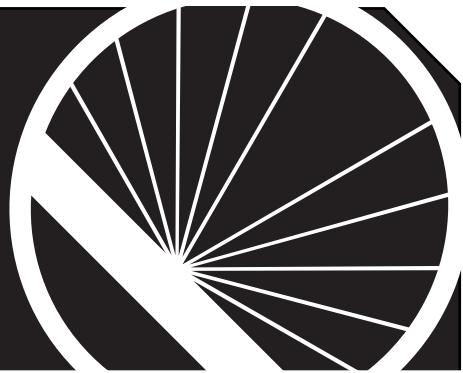




 Reflexion®



Aprende electrónica por ti mismo con la **GUÍA DEL INVENTOR DE REFLEXIÓN**



UNAS POCAS PALABRAS



:: SOBRE ESTE KIT

El objetivo general de este kit es pasárselo bien. Partiendo de esa base, nuestro mayor deseo es hacer que te resulte lo más placentero posible el utilizar una amplia variedad de componentes electrónicos, a través de unos pequeños circuitos muy fáciles y sencillos de montar. La fórmula consiste en hacer funcionar estos circuitos, facilitándote las herramientas para que puedas entender como funcionan todos ellos. Si encuentras algún tipo de problema,quieres hacer alguna pregunta o te gustaría saber más sobre algún tema en particular siempre puedes ponerte en contacto con nosotros mandando un correo electrónico a shop@reflexiona.biz.

:: SOBRE EL HARDWARE DE CÓDIGO ABIERTO

Todos los proyectos presentados en esta guía son de código abierto. Esto significa que todo lo relacionado con el desarrollo de este kit, ya sea la propia guía, los modelos 3D o el código fuente, están disponibles para que te los descargues de forma gratuita. Pero la cosa va más lejos todavía, también eres libre de reproducir y modificar cualquier parte de este material, y de distribuirlo tú mismo. ¿A que te preguntas donde está el truco o la trampa? Nada más lejos, todo lo anterior tiene una explicación muy sencilla. Todo este material está publicado bajo licencia Creative Commons Reconocimiento-CompartirIgual 3.0 España (CC BY-SA 3.0). Esto significa que debes nombrar a sus desarrolladores iniciales en tu diseño y compartir tus desarrollos del mismo modo. ¿Y por qué? Porque todos nosotros hemos crecido aprendiendo y jugando con software de código abierto y la experiencia ha sido realmente divertida. Pensamos que sería igual de genial si puede volver a repetirse la experiencia con objetos físicos.

Puedes encontrar más detalles sobre las licencias Creative Commons en <http://wiki.creativecommons.org/spain>

:: SOBRE OOMLOUT

Oomlout es una pequeña pero valiente empresa de diseño, orientada a la producción de productos de código abierto exquisitamente originales y divertidos. Más información en <http://www.oomlout.com>

:: SOBRE SPARKFUN

Sparkfun es una empresa joven y dinámica que pretende convertir la electrónica en algo divertido, accesible y cercano para todo el mundo, desde chavales que se encuentran en educación elemental a ingenieros superiores. Más información en <http://www.sparkfun.com>

:: SOBRE REFLEXIONA

Reflexiona es un estudio creativo multidisciplinar especializado en Diseño de Iluminación, Integración Audiovisual y Consultoría en Tecnología del Espectáculo, que considera el software y el hardware de código abierto como una fórmula de aprendizaje y desarrollo de proyectos muy atractiva. Más información en <http://www.reflexiona.biz>

:: SOBRE PROBLEMAS

Nos esforzamos por ofrecer la mayor calidad posible en todos y cada uno de los productos que desarrollamos y distribuimos. Si encuentras alguna instrucción ambigua, faltan piezas o, simplemente, quieres formular una consulta, mándanos un correo electrónico a shop@reflexiona.biz. (nos gusta escuchar tus problemas para mejorar las próximas versiones)

MUCHAS GRACIAS POR ESCOGER A OOMLOUT, SPARKFUN Y REFLEXIONA

:: ANTES DE EMPEZAR

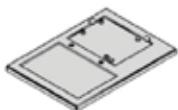
[ENSAMBLAJE] Uniendo las piezas	02
[INSTALACIÓN] Instalando el IDE	03
[PROGRAMACIÓN] Fundamentos de programación	04
[ELECTRÓNICA] Fundamentos de electrónica	07

:: LOS CIRCUITOS

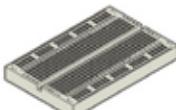
[CIRC - 01] Blinking LED	12
[CIRC - 02] Múltiples LEDs	14
[CIRC - 03] Transistor + Motor	16
[CIRC - 04] Servomotor	18
[CIRC - 05] Registro electrónico	20
[CIRC - 06] Piezo elemento	22
[CIRC - 07] Pulsador	24
[CIRC - 08] Potenciómetro	26
[CIRC - 09] Fotorresistencia	28
[CIRC - 10] Sensor de temperatura	30
[CIRC - 11] Relé	32
[CIRC - 12] LED RGB	34
[CIRC - 13] Sensor flexible	36
[CIRC - 14] Potenciómetro de membrana	38
[NOTAS] Espacio para tus apuntes	40

ENSAMBLAJE

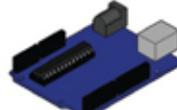
Uniendo las piezas



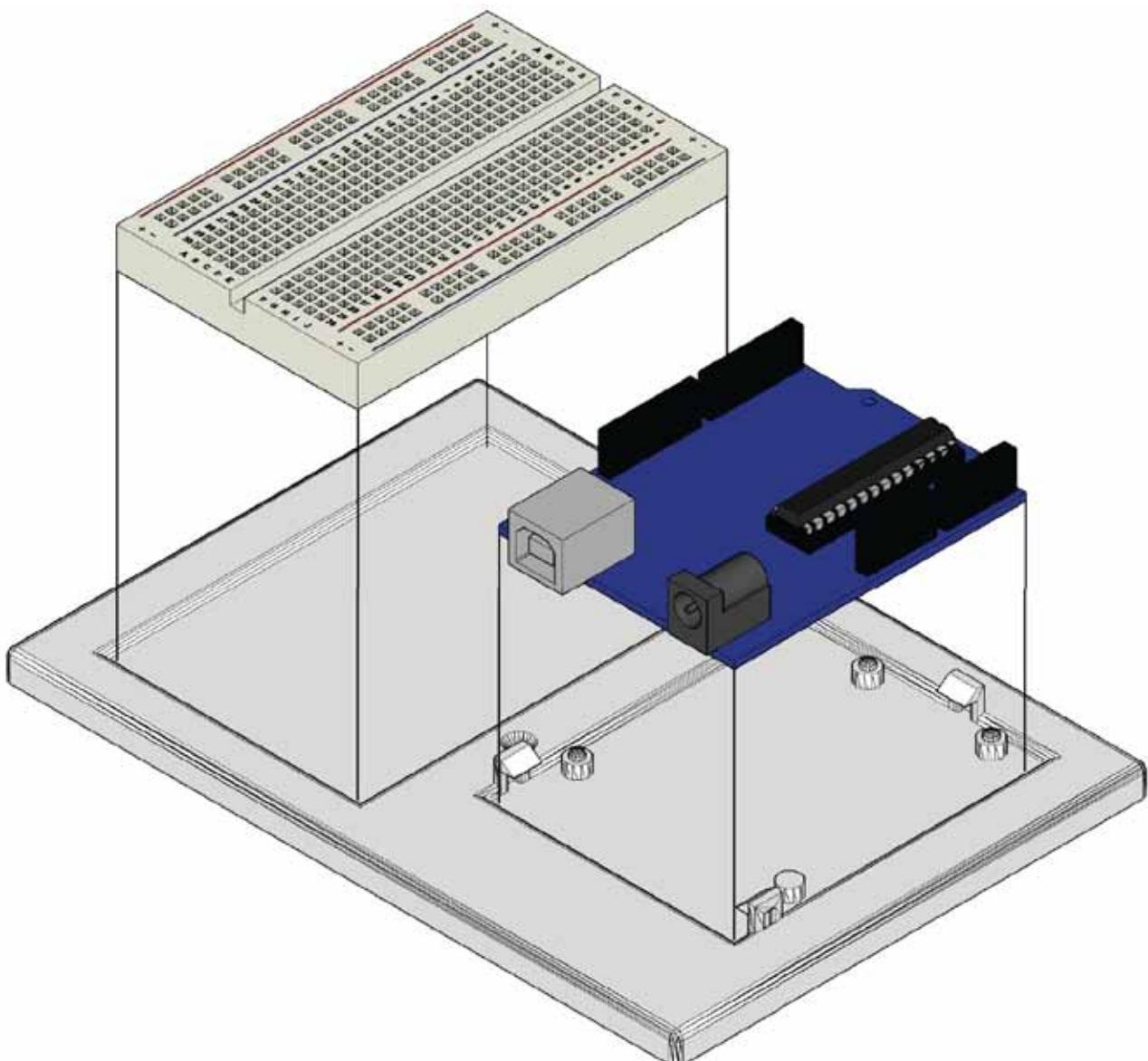
1 x
Soporte
para Arduino



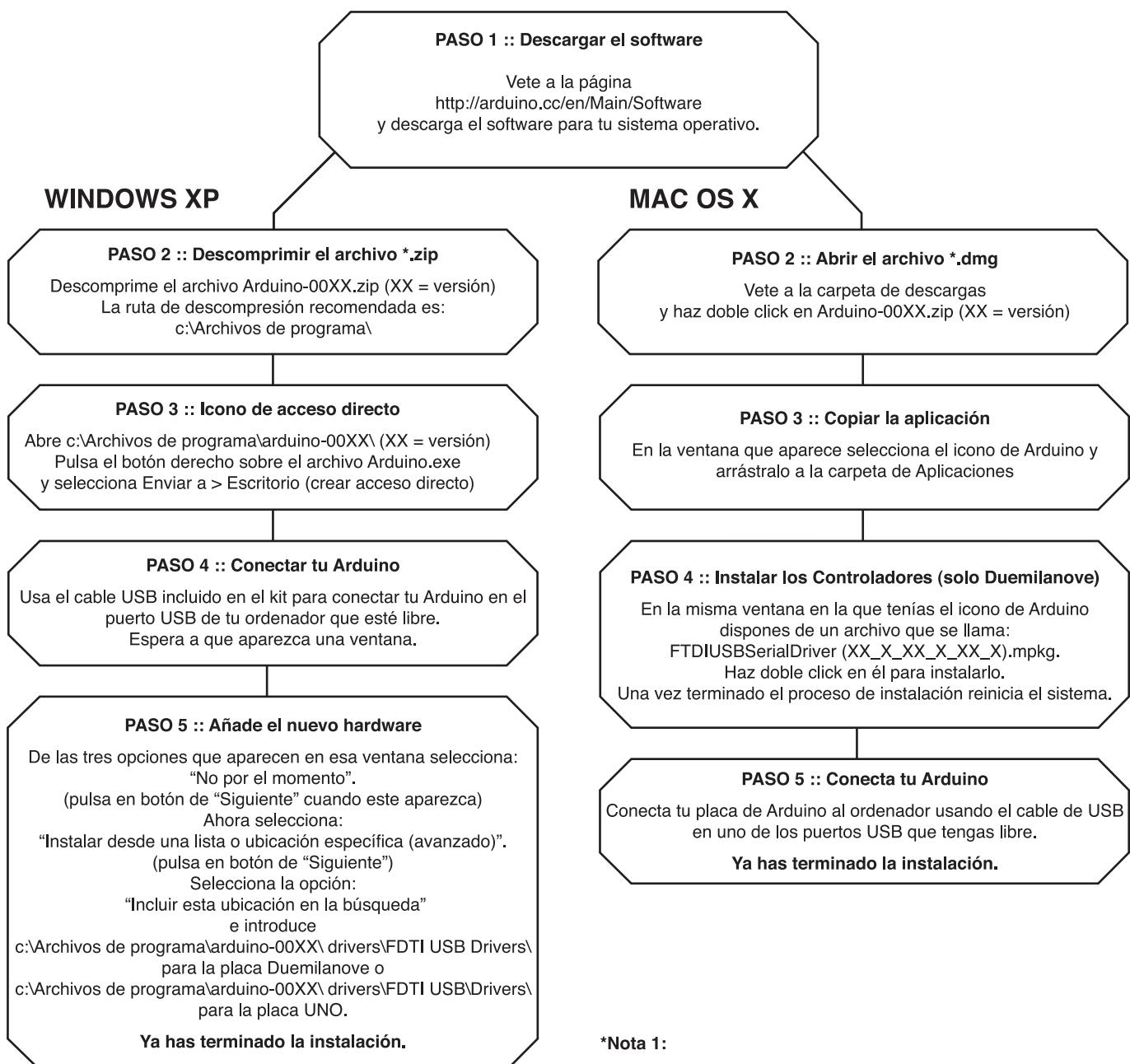
1 x
Tablero
de circuitos



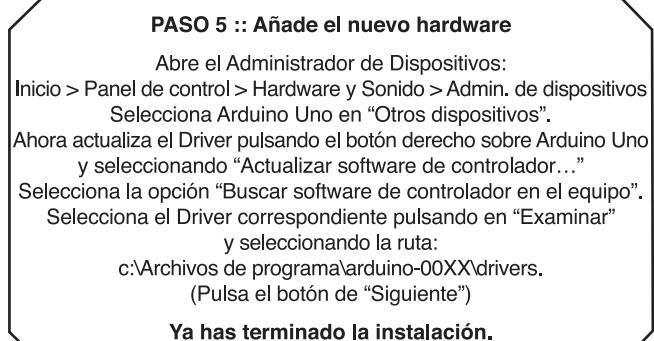
1 x
Arduino



El IDE o Entorno de Desarrollo Integrado es el software que se utiliza para escribir el código con Arduino. Al principio puede parecer un poco desalentador, pero una vez que lo tienes instalado y empiezas a trabajar con él, irás poco a poco descubriendo todos los secretos que esconde.



WINDOWS VISTA / WINDOWS 7



*Nota 1:

Si has tenido problemas a la hora de la instalación, o tienes curiosidad por algún detalle en particular, puedes solicitar ayuda mandando un correo electrónico a shop@reflexiona.biz

*Nota 2:

Si usas el sistema operativo Linux puedes encontrar más información en <http://www.arduino.cc/playground/Learning/Linux>

:: PRINCIPIOS DE PROGRAMACIÓN CON ARDUINO

Arduino se programa con lenguaje C. A continuación se explican algunas nociones básicas para aquellas personas que tienen ciertos conocimientos de programación y que solo necesitan una pequeña introducción en la idiosincrasia de C y del IDE de Arduino. Si te parece que estos contenidos son un poco desalentadores, no te preocupes. Puedes empezar directamente a trabajar con los circuitos, e ir adquiriendo estos conocimientos de programación a medida que vas practicando.

Para una introducción más a fondo, la web <http://arduino.cc> es una gran fuente de recursos.

:: ESTRUCTURA

Cada uno de los programas de Arduino (a menudo llamados sketch) tiene dos funciones fundamentales en su estructura (también conocidas como rutinas).

void setup ()

(inicialización) Esta función se establece cuando se inicia un sketch. Se emplea para iniciar variables, establecer el estado de los pines, inicializar librerías, etc. Esta función se ejecutará una única vez después de que se conecte la placa Arduino a la fuente de alimentación, o cuando se pulse el botón de reinicio de la placa.

void loop ()

(bucle) Esta función arranca cuando el “setup” ha terminado. Después de reproducirse la primera vez volverá a reproducirse una y otra vez hasta que se desconecte la fuente de alimentación.

:: SINTAXIS

Uno de los aspectos más frustrantes de C (pero que también lo hace realmente potente) son los requisitos de formato. Si eres capaz de acordarte de lo siguiente, todo irá a la perfección.

//

(comentarios en una línea) Estas dos barras son muy útiles para que, a medida que avanzas por cada línea de código, escribas notas sobre los que estás haciendo. Estas líneas son ignoradas por el compilador y no se exportan al procesador. Por lo tanto, no ocupan espacio en el chip ATmega.

/* */

(comentarios en múltiples líneas) Si tienes mucho que escribir puedes extender los comentarios a lo largo de varias líneas. Todo lo que escribas entre estos dos símbolos será ignorado por el compilador y no se exporta al procesador. Por lo tanto, no ocupa espacio en el chip ATmega.

{ }

(llaves) Se usan para definir cuando empieza y cuando acaba un bloque de código (también se usa en funciones como **void loop**)

;

(punto y coma) Cada línea de código debe terminar con un punto y coma (la falta de punto y coma al final de una línea de código es muchas veces el motivo de que un programa rechace la compilación).

:: VARIABLES

Un programa no es más que una serie de instrucciones que permiten mover los números en un sentido o en otro, de manera inteligente. Las variables son las funciones que se utilizan para realizar todos esos movimientos.

int

(entero) El caballo de batalla principal. Almacena un número en 2 bytes (16 bits). No tiene decimales y almacenará un valor entre -32.768 y 32.767.

long

(entero 32b) Se utiliza cuando un entero no es lo suficientemente grande. Tiene 4 bytes (32 bits) de RAM y su rango se encuentra entre -2.147.483.648 y 2147.483.647.

boolean

(booleano) Se trata de una simple variable Verdadero o Falso. Muy útil por que solo utiliza 1 bit de memoria RAM.

float

(en coma flotante) Se utiliza con las matemáticas de punto flotante (decimales). Tiene 4 bytes (32 bits) de RAM y su rango se encuentra entre -3.4028235E+38 y 3.4028235E+38.

char

(carácter) Almacena un carácter utilizado en código ASCII (por ejemplo 'A'=65). Utiliza 1 byte (8 bits) de RAM. Arduino maneja las cadenas como una matriz de caracteres.

:: OPERADORES ARITMÉTICOS

Los operadores aritméticos se utilizan para manipular los números (funcionan como en las matemáticas simples)

= (asignación)

Hace que algo sea igual a algo otro (ej. **x = 10** x es por lo tanto igual a 20)

% (resto)

Calcula el resto de la división entre dos enteros. Es muy útil para mantener una variable dentro de un rango particular (por ejemplo, el tamaño de un array).

+ (suma)

- (resta)

* (multiplicación)

/ (división)

:: OPERADORES COMPARATIVOS

Los operadores comparativos se utilizan para realizar comparaciones lógicas

== (igual a) (ej. **12==10** es FALSE ó **12==12** es TRUE)

!= (distinto de) (ej. **12!=10** es TRUE ó **12!=12** es FALSE)

< (menor que) (ej. **12<10** es FALSE ó **12<12** es FALSE ó **12<14** es TRUE)

> (mayor que) (ej. **12>10** es TRUE ó **12>12** es FALSE ó **12>14** es FALSE)

PROGRAMACIÓN

:: ESTRUCTURAS DE CONTROL

Los programas dependen de lo que se va a reproducir después. A continuación se muestran los elementos básicos de control (hay unos cuantos más en internet).

```
if (comparador) {      }
else if (comparador) {    }
else {      }
```

Esto ejecutará el código entre las llaves si el comparador es **TRUE**, y si es **FALSE** ejecutará en comparador **else if**, en caso de que este sea también **FALSE** se ejecutará el código **else**.

```
For (int i = 0; i< #repeats; i++) {      }
```

Se usa cuando quieras repetir un trozo de código un número determinado de veces (puede contar hasta **i++** o hasta **i--**, o usar cualquier otra variable)

:: FUNCIONES DIGITALES

```
pinMode(pin, mode);
```

Se usa para configurar el modo de un pin, donde **pin** es el número de salida que quieras direccionar del 1 al 19 (los pines analógicos 0 a 5 se corresponden con los números del 14 al 19). El parametro **mode** (modo) puede ser **INPUT** (entrada) o **OUTPUT** (salida).

```
digitalRead(pin);
```

Una vez que el pin a sido establecido como **INPUT** (entrada), puede ser configurado para que responda como **HIGH** (valor del voltaje a 5V) o **LOW** (valor del voltaje a 0V).

```
digitalWrite(pin);
```

Una vez que el pin ha sido establecido como **OUTPUT** (salida), puede ser configurado para que responda como **HIGH** (valor del voltaje a 5V) o **LOW** (valor del voltaje a 0V).

:: FUNCIONES ANALÓGICAS

Arduino es una máquina digital pero tiene la habilidad de funcionar también en el ámbito analógico (por medio de ciertos trucos). A continuación se explica como se trabaja con dispositivos que no sean digitales.

```
analogRead(pin);
```

Cuando los pines de entrada analógica son configurados como **INPUT** (entrada) puedes leer su voltaje. Estos pines pueden leer un valor entre 0 (para 0 voltios) y 1024 (para 5 voltios).

```
analogWrite(pin, value);
```

Algunos pines de Arduino soportan PWM (o Modulación por Ancho de Pulso), concretamente los pines 3, 5, 6, 9, 10 y 11. Esta técnica enciende y apaga el pin de forma muy rápida para que funcione como una salida digital. El valor es cualquier número entre 0 (ciclo de trabajo al 0%, siempre apagado, 0 voltios) y 255 (ciclo de trabajo al 100%, siempre encendido, 5 voltios).

*Nota: Para una referencia de programación completa visita <http://arduino.cc/es/Reference/HomePage>

:: PRINCIPIOS DE ELECTRÓNICA

No es necesaria experiencia previa con la electrónica para divertirse con este kit. A continuación, puedes encontrar algunos detalles sobre cada uno de los componentes electrónicos que incluye el kit, para que lo puedas identificar y entiendas en qué consiste de manera más sencilla. Si en algún momento te preocupa el funcionamiento de algún componente en particular, o si este no funciona, internet resulta una verdadera fuente de conocimientos. Asimismo, siempre puedes ponerte en contacto con nosotros enviando un email a shop@reflexiona.biz

:: DETALLES DE LOS COMPONENTES

LED (Light Emitting Diode)



Lo que hace: Emite luz cuando una pequeña corriente lo atraviesa.

Identificación: Parece una pequeña bombilla.

Número de conductores: 2 (Uno más largo que el otro. El más largo se conecta al positivo)

A tener en cuenta: Solo funciona en una dirección. Requiere una resistencia (de 330 ohmios) que limite el paso de la corriente.

Más detalles: <http://es.wikipedia.org/wiki/Led>

Resistencia



Lo que hace: Restringe la cantidad de corriente eléctrica que puede circular a través del circuito.

Identificación: Es un cilindro con alambres que salen de cada uno de sus extremos. Su valor se muestra utilizando un código de color (ver página 11 para más detalles).

Número de conductores: 2

A tener en cuenta: Es muy fácil equivocarse con el código de color. Antes de utilizarlas, conviene comprobar siempre por segunda vez que se trata de la resistencia correcta .

Más detalles: <http://es.wikipedia.org/wiki/Resistor>

Diodo



Lo que hace: Es el equivalente electrónico a una válvula de un solo sentido (o antiretorno), permitiendo que la corriente circule en una dirección pero no en la contraria.

Identificación: Normalmente, es un cilindro con alambres que salen de cada uno de los extremos y una franja negra indicando la polaridad (la franja está en el lado del positivo).

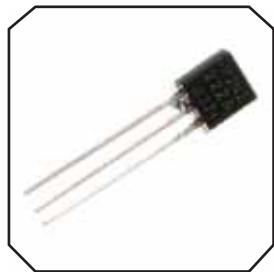
Número de conductores: 2

A tener en cuenta: Solo funciona en una dirección (la corriente circulará a través del diodo si el final de la línea está conectado a tierra).

Más detalles: <http://es.wikipedia.org/wiki/Diodo>

ELECTRÓNICA

Transistor



Lo que hace: Utiliza una pequeña corriente para conmutar o amplificar una corriente mayor.

Identificación: Viene en diferentes formatos pero puedes leer el número de pieza sobre su superficie. El de este kit es el P2N2222AG y su hoja de especificaciones está en Internet.

Número de conductores: 3 (base, colector, emisor)

A tener en cuenta: Conectarlo correctamente (también suele ser necesario una resistencia limitadora de corriente en el pin base).

Más detalles: <http://es.wikipedia.org/wiki/Transistor>

Motor eléctrico



Lo que hace: Su eje rota cuando lo atraviesa una corriente eléctrica.

Identificación: Este es fácil, tiene pinta de motor. Normalmente es cilíndrico con un eje que sobresale de uno de los extremos.

Número de conductores: 2

A tener en cuenta: Utilizar un transistor o un relé que sea el adecuado para el tamaño del motor eléctrico que estés utilizando.

Más detalles: http://es.wikipedia.org/wiki/Motor_el%C3%A9ctrico

Servo motor



Lo que hace: Recibe un tiempo de pulso y lo convierte en una posición angular del eje.

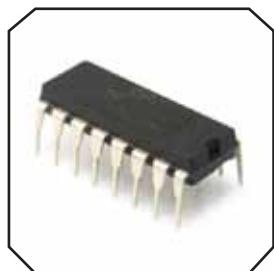
Identificación: Es una pequeña caja negra con tres cables que salen de uno de sus lados y un eje con una hélice de plástico que sobresale de su parte superior.

Número de conductores: 3

A tener en cuenta: El conector no está polarizado así que tienes que asegurarte que lo conectas de forma correcta.

Más detalles: <http://es.wikipedia.org/wiki/Servo>

Registro electrónico



Lo que hace: Almacena cualquier tipo de electrónica compleja dentro de un envoltorio que lo hace muy fácil de usar.

Identificación: La ID (Identificación) de la pieza está escrita en el exterior del envoltorio (para poder leer esta ID se requiere, a veces, de mucha luz o una lupa).

Número de conductores: De 2 a 100s. En este kit hay un con 3 pines (TMP36) y otro con 16 pines (74HC595).

A tener en cuenta: Orientarlo de manera correcta (hay que fijarse en las marcas que indican cuál es el pin número 1).

Más detalles: http://es.wikipedia.org/wiki/Circuito_integrado

Sensor piezoelectrónico



Lo que hace: Un pulso de corriente le hará emitir un “click” y una serie de pulsos de corriente le harán emitir un tono.

Identificación: En este kit su forma es la de un pequeño barrilete negro, pero muchas veces son tan solo un disco dorado.

Número de conductores: 2

A tener en cuenta: Muy difícil de usar mal.

Más detalles: http://es.wikipedia.org/wiki/Sensor_piezoelectrónico

Botón pulsador



Lo que hace: Completa el circuito cuando es pulsado.

Identificación: En este caso es un pequeño cuadrado con 4 patillas en su parte inferior y un botón redondo en su parte superior.

Número de conductores: 4

A tener en cuenta: La forma de estos pulsadores es cuadrada por lo que hay que tener cuidado de no conectarlos girados 90°.

Más detalles: [http://es.wikipedia.org/wiki/Botón_\(dispositivo\)](http://es.wikipedia.org/wiki/Botón_(dispositivo))

Potenciómetro



Lo que hace: Produce una resistencia variable en función de la posición angular del eje.

Identificación: Pueden tener un montón de formas diferentes. En este caso, para identificarlo tienes que buscar un selector azul con una flecha en bajo relieve.

Número de conductores: 3

A tener en cuenta: Existen potenciómetros lineales y logarítmicos. Ten cuidado de no comprar accidentalmente uno de escala logarítmica.

Más detalles: <http://es.wikipedia.org/wiki/Potencímetro>

Fotorresistencia



Lo que hace: Produce una resistencia variable en función de la cantidad de luz incidente.

Identificación: Normalmente es un pequeño disco con la parte superior blanca y una línea curva en su parte inferior.

Número de conductores: 2

A tener en cuenta: Recordar que necesita estar situado en un divisor de voltaje para poder proporcionar una entrada útil.

Más detalles: <http://es.wikipedia.org/wiki/Fotorresistencia>

ELECTRÓNICA

Sensor de temperatura



Lo que hace: Proporciona un voltaje proporcional a la temperatura medida en grados centígrados.

Identificación: Viene en diferentes formatos pero puedes leer el número de pieza que viene escrito en su superficie. El de este kit es el TMP36 y su hoja de especificaciones está en Internet.

Número de conductores: 2

A tener en cuenta: Conectarlo de la forma correcta.

Más detalles: <http://es.wikipedia.org/wiki/Sensor>

Relé



Lo que hace: Funciona como un interruptor controlado por un circuito eléctrico que permite abrir o cerrar otros circuitos eléctricos independientes.

Identificación: Viene en diferentes tamaños y formatos, pero normalmente puedes leer las características del relé que vienen escritas en su superficie.

Número de conductores: 5

A tener en cuenta: Conectarlo de la forma correcta. Es un modelo de relé para soldar así que puede que tengas que presionarlo contra la breadboard para que haga contacto correctamente.

Más detalles: <http://es.wikipedia.org/wiki/Relé>

LED RGB



Lo que hace: Emite luz de cuando una pequeña corriente lo atraviesa. Un LED RGB es en realidad 3 LEDs en uno: Rojo+Verde+Azul. Cuando enciendes dos o más de ellos, se mezclan para conseguir el resto de colores del espectro.

Identificación: Parece una pequeña bombilla.

Número de conductores: 4 (uno más largo que el resto que se conecta al positivo)

A tener en cuenta: Requiere de resistencias que limiten el paso de la corriente.

Más detalles: <http://bit.ly/rQEHch>

Sensor flexible



Lo que hace: Cuando se dobla, las láminas de color oscuro se separan y la resistencia a través del sensor aumenta.

Identificación: Hay de diferentes tamaños. El de este kit es una lámina de plástico de 8 cm. de largo x 0,7 cm. de ancho con franjas blancas y negras en uno de sus lados.

Número de conductores: 2

A tener en cuenta: La resistencia del sensor flexible varía cuando las láminas de metal están en el exterior de la curva.

Más detalles: <http://bit.ly/sImPsv>

Potenciómetro de membrana



Lo que hace: Se trata de una resistencia variable en la que la resistencia viene determinada por el lugar de la superficie sobre el que se aplica una presión.

Identificación: Existen de diferentes tamaños. El de este kit es una lámina de plástico de 6,5 cm. de largo x 2 cm. de ancho con una lengüeta de color azul.

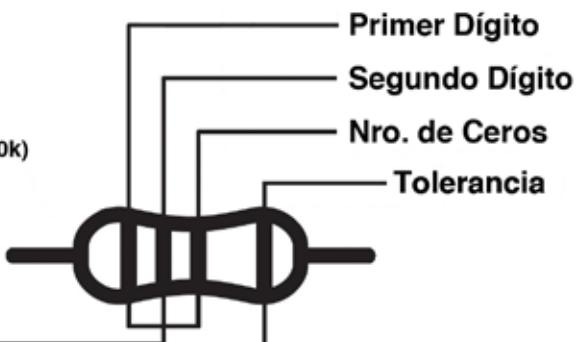
Número de conductores: 3

A tener en cuenta: Se le puede aplicar presión con un dedo, un bolígrafo o un trozo de plástico.

Más detalles: <http://bit.ly/s68Klr>

:: CÓDIGO DE COLOR DE LAS RESISTENCIAS

Ejemplos:
verde-azul-marrón = 560 Ohmios
rojo-rojo-rojo = 2200 Ohmios (2.2K)
marrón-negro-naranja = 10000 Ohmios (10k)



■ 0 - Negro
■ 1 - Marrón
■ 2 - Rojo
■ 3 - Naranja
■ 4 - Amarillo

■ 5 - Verde
■ 6 - Azul
■ 7 - Morado
■ 8 - Gris
■ 9 - Blanco

20%	- Ninguna
10%	- Plateado
5%	- Dorado

:: RECORTAR LAS PATILLAS

Algunos de los componentes de este kit vienen con unos cables o patillas realmente largos. Para hacerlos más compatibles con la breadboard (o tablero de circuitos) son necesarios un par de cambios.

LEDs

Corta las patillas de los LEDs para que la patilla más larga (positivo) tenga aproximadamente 10mm. y la patilla más corta (negativo) tenga aproximadamente 7mm.

Resistencias

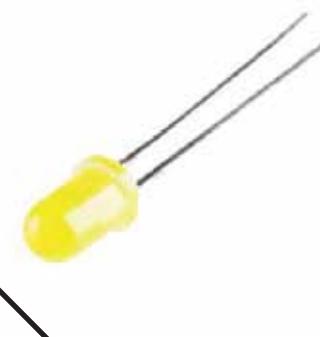
Dobra las patillas hacia abajo para colocarlas a 90° respecto del cilindro. Después recórtalas para que tengan una longitud de aproximadamente 6 mm.

Otros componentes

Algún otro de los componentes puede que también necesite ser cortado. Usa tu propio criterio.

:: LO QUE ESTAMOS HACIENDO

Los LEDs se utilizan en infinidad de ingeniosos objetos y ese es el principal motivo por el cual han sido incluidos en este kit. Vamos a empezar con algo realmente simple, encendiendo y apagando uno de estos LEDs repetidamente para generar un agradable efecto de parpadeo. Para empezar tienes que coger los componentes que se muestran más abajo. Despues, coloca la plantilla en el tablero de circuitos y conecta todo tal y como te indica la plantilla. Una vez que el circuito está ensamblado tendrás que cargar el programa. Para llevar a acabo esta operación tienes que conectar tu placa Arduino al puerto USB de tu ordenador. Ahora tienes que seleccionar el puerto correcto en **Herramientas > Puerto Serial > (el puerto de comunicación de tu Arduino)**. Vamos a utilizar un ejemplo de programa incluido en el IDE. Para abrirlo vete a **Archivo > Ejemplos > 01.Basics > Blink**. Lo siguiente es cargar el programa yendo a **Archivo > Cargar** (o pulsando cmd+U) (ctrl+U para PC). ¡Por fin! ¡Ya puedes disfrutar de la gloria! (y de las posibilidades que ofrece la capacidad de controlar la luz).



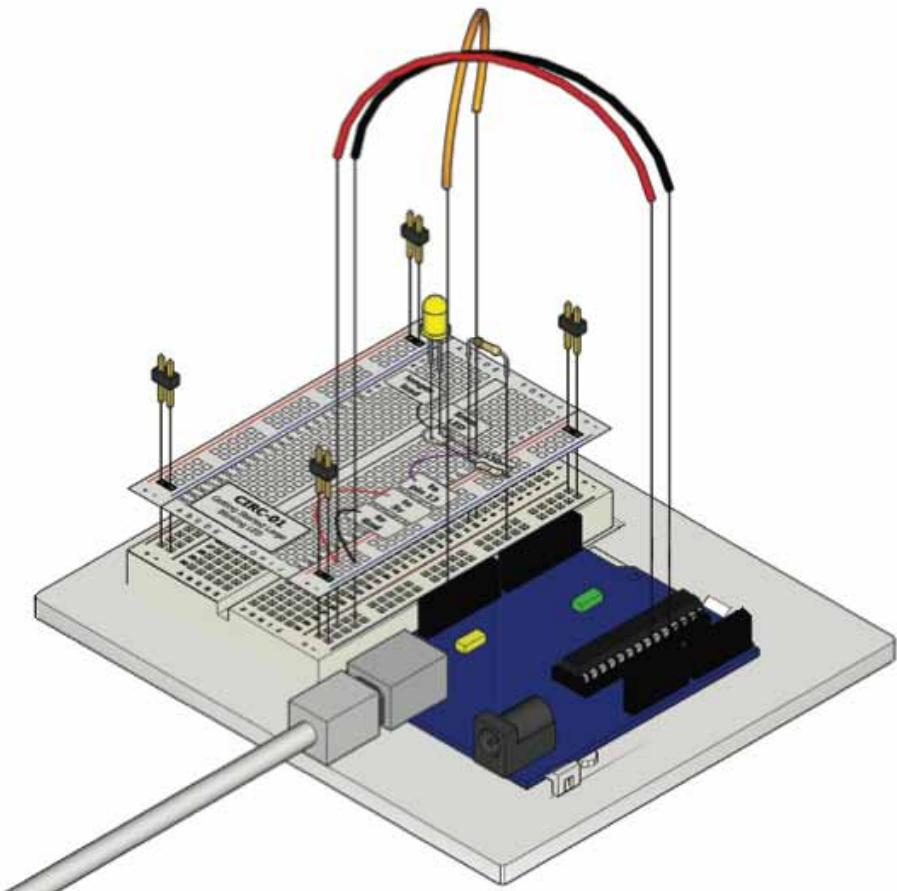
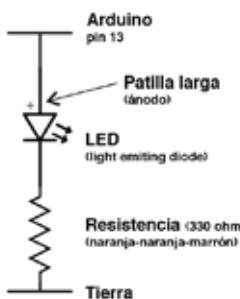
Si tienes algún problema cargando el programa, dispones de una completa guía de solución de problemas en la página <http://arduino.cc/es/Guide/Troubleshooting>

:: EL CIRCUITO

_Componentes



_Esquema



_Internet

Descarga una plantilla para el **CIRC-01** en
http://reflexiona.biz/plantilla_circ01

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ01

:: EL CÓDIGO

No es necesario que escribas todo el texto, solo tienes que pulsar en: **Archivo > Ejemplos > 01.Basics > Blink** (ejemplo extraído de la página <http://Arduino.cc>. Échale un vistazo para ver otras grandes ideas)

:: NO FUNCIONA (3 cosas que puedes probar)

_No se enciende el LED

Los LEDs solo funcionan en una dirección. Prueba a desconectarlo y conectarlo de nuevo pero girado 180° (no te preocupes, conectarlo al revés no causa daños permanentes)

_No se carga el programa

Esto sucede a veces, lo más común es que no hayas elegido correctamente el puerto serie.

Puedes cambiarlo en **Herramientas > Puerto Serial (el puerto de comunicación de tu Arduino)**

_Sigue sin funcionar

Un circuito roto no es nada divertido. Mándanos un email a shop@reflexiona.biz y nos pondremos en contacto contigo lo antes posible para que, en caso necesario, podamos reemplazar tu placa.

:: MEJORANDO EL CIRCUITO

_Cambiar el pin

El LED está conectado al pin 13 pero puedes usar cualquiera de los pines de Arduino. Para cambiarlo, coge el cable conectado al pin 13 y muévelo al pin que tu elijas (de 0 a 13) (también puedes conectarlo a las salidas analógicas: 3, 5, 6, 9, 10 ,11)

Ahora, en el código, cambia la línea:

```
int led = 13; => int led = nuevo pin;
```

Y, después, carga el sketch (cdm+U para Mac OS o ctrl + U para Windows)

_Cambiar el tiempo de parpadeo

¿No te gusta que este 1 segundo encendido y 1 segundo apagado?

Cambia las siguientes líneas de código:

```
digitalWrite(ledPin, HIGH);
delay(tiempo de encendido); // (segundos * 1000)
digitalWrite(ledPin, LOW);
delay(tiempo de apagado); // (segundos * 1000)
```

_Controlar el brillo

Además del control digital (on/off), Arduino puede controlar algunos pines de forma analógica (control del brillo). Habrá más detalles sobre este tema en posteriores circuitos pero vamos a jugar un poco con esta característica tan estupenda.

En el código, cambia el LED de salida del pin 13 al pin 9 (cambia también el cable que va a la breadboard):

```
int led = 13; => int led = 9;
```

Sustituye el código **digitalWrite()** en **loop{ }** por esto: **analogWrite(9, número 8 bits);**

(número 8 bits) = cualquier número entre 0 y 255. (0 = off; 255 = on; Entre 0 y 255 = diferente brillo)

_Controlar la intensidad:

Vamos a utilizar otro ejemplo de programa incluido en el IDE. Para abrirlo vete a **Archivo > Ejemplos > 03.Analog > Fading**. Ahora carga este programa en la placa y observa como el LED se enciende y se apaga gradualmente.

CIRC - 02

:: LO QUE ESTAMOS HACIENDO

Ya hemos hecho parpadear un LED. Es momento de ir un paso más allá. Ahora, vamos a conectar 8 LEDs. También tendremos la oportunidad de forzar un poco más la placa Arduino creando varias secuencias de iluminación. El circuito que vamos a utilizar a continuación es muy útil para que experimentes escribiendo tus propios programas y consigas hacerte una mejor idea de cómo funciona Arduino.



Al mismo tiempo que controlamos los LEDs vamos a empezar a utilizar unos métodos muy simples de programación que nos ayudarán a mantener los programas más cortos.

for () loops : Se usa cuando quieras reproducir una parte del código varias veces.

Arrays [] : Se usa para manejar variables de forma sencilla (se trata de un grupo de variables).

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-02
para tablero de circuitos



4 x
Clavija de 2 pines



8 x
LED amarillo de 5 mm.

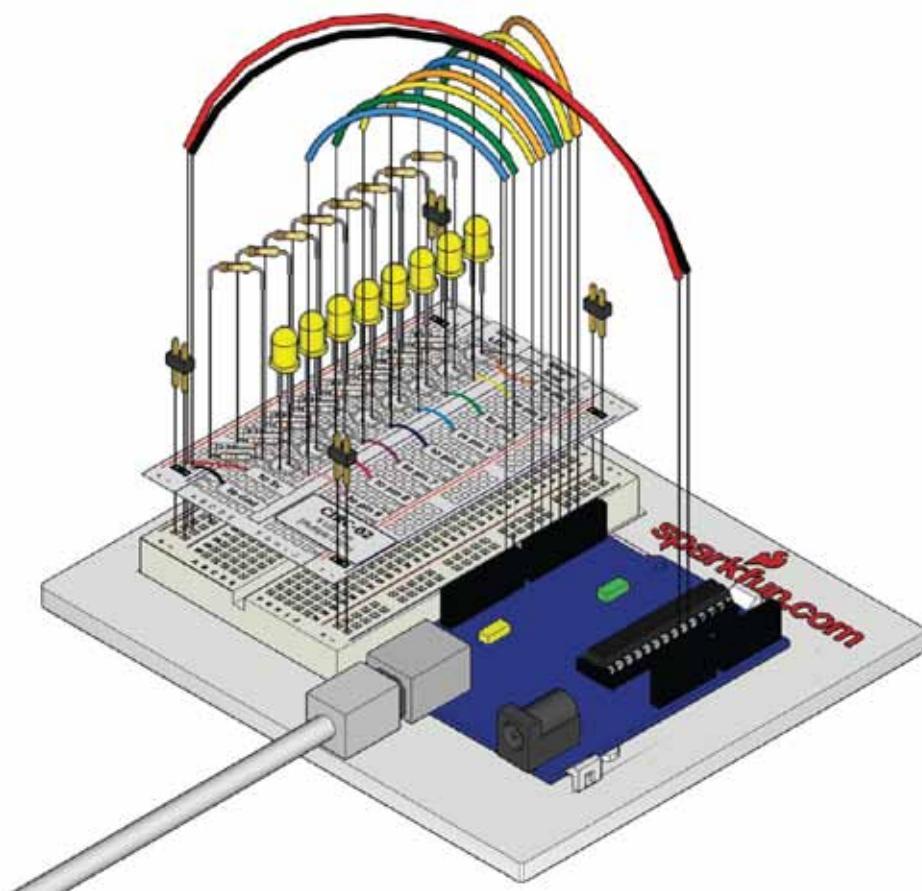
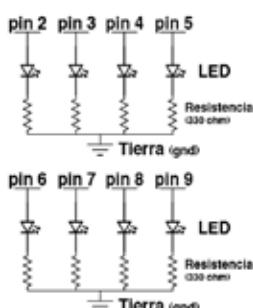


10 x
Cable con terminales



8 x
Resistencia 330 Ohmios
(Naranja-Naranja-Marrón)

_Esquema



_Internet

Descarga una plantilla para el **CIRC-02** en
http://reflexiona.biz/plantilla_circ02

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ02

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ02. Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_Algunos LEDs no se encienden

Es muy fácil conectar un LED al revés. Comprueba todos los LEDs que no se encienden y asegúrate de que están colocados de forma correcta.

_Funcionan de forma desordenada

Con ocho cables atravesando el circuito es muy fácil que se te haya cruzado alguno. Comprueba que el primer LED está conectado al pin 2 y que el resto de LEDS están conectados a los pines que siguen a este.

_Empezar de nuevo

Es muy fácil conectar mal un cable de forma accidental. Mucha veces, es mejor desconectar todo y empezar otra vez con una nueva plantilla que intentar encontrar donde está problema.

:: MEJORANDO EL CIRCUITO

_Ajustando los ciclos

En la función `loop ()` hay 4 líneas de código. Las tres últimas empiezan con `//`, esto significa que el compilador considera esa línea como texto y no como programación. Para ajustar el programa a la utilización de los diferentes ciclos disponibles, hay que modificar el contenido de la función `loop ()` del código:

```
// oneAfterAnotherNoLoop();
oneAfterAnotherLoop();
// oneOnAtATime();
// inAndOut();
```

Carga el programa y te darás cuenta de que nada ha cambiado. Echa un vistazo a las dos primeras funciones, las dos hacen lo mismo pero utilizan aproximaciones distintas (la segunda función utiliza un `for loop`)

_Animaciones extra

¿Te has cansado de esta animación? Entonces, prueba las otras dos animaciones. Borra las barras que dan a cada línea el formato de comentario, carga de nuevo el programa en tu placa y disfruta de las nuevas animaciones de iluminación. Prueba primero, borrando las barras solo delante de las línea 3, y después haz lo mismo con las de la línea 4. Al principio, solo tendría que haber una línea de las cuatro sin barras cada vez que cargas el programa. Más tarde puedes probar a quitar las barras de todas las líneas.

_Comprobando tus propias animaciones

Introdúcete en el código y empieza a cambiar cosas. El objetivo es encender un LED usando `digitalWrite(pinNumber, HIGH)`; después apagar ese mismo LED usando `digitalWrite(pinNumber], LOW)`; Escribe todo lo que te dé la gana. Aunque cambies un montón de líneas de código no vas a romper nada.

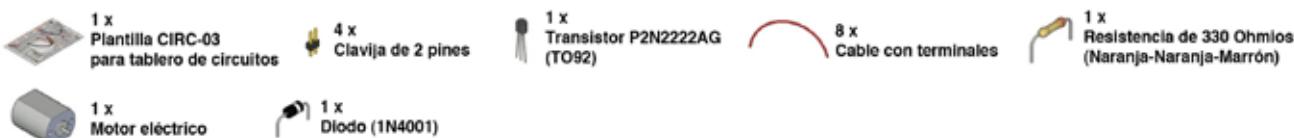
:: LO QUE ESTAMOS HACIENDO

Los pines de Arduino son estupendos para controlar directamente pequeños dispositivos eléctricos como los LEDs. En cualquier caso, cuando hay que lidiar con dispositivos más grandes (como un motor eléctrico o una lavadora) es necesario un transistor externo. Los transistores son increíblemente útiles, ya que proporcionan mucha corriente usando una corriente mucho menor. Los transistores tienen 3 pines (base, colector y emisor). En los transistores en modo amplificador (NPN), la carga se conecta al colector, y la tierra al emisor. Cuando una pequeña corriente pasa desde la base al emisor, la corriente fluye a través del transistor y hace girar el motor (esto sucede cuando configuras el pin de tu Arduino como **HIGH**). Hay miles de tipos diferentes de transistores, lo que permite cumplir a la perfección con cualquier cometido. Para este circuito se ha escogido un transistor P2N2222AG, se trata de un transistor común y de uso general. Lo que hay que tener en cuenta, en este caso, es que su tensión máxima (40V) y su corriente máxima (200mA) sean las suficientemente altas para el motor eléctrico. (Puedes encontrar la hoja de especificaciones de este transistor visitando la página <http://reflexiona.biz/p2n2222ag>). En este circuito, el diodo 1N4001 actúa como diodo de retorno. Puedes encontrar más detalles sobre porque está ahí en http://en.wikipedia.org/wiki/Flyback_diode.

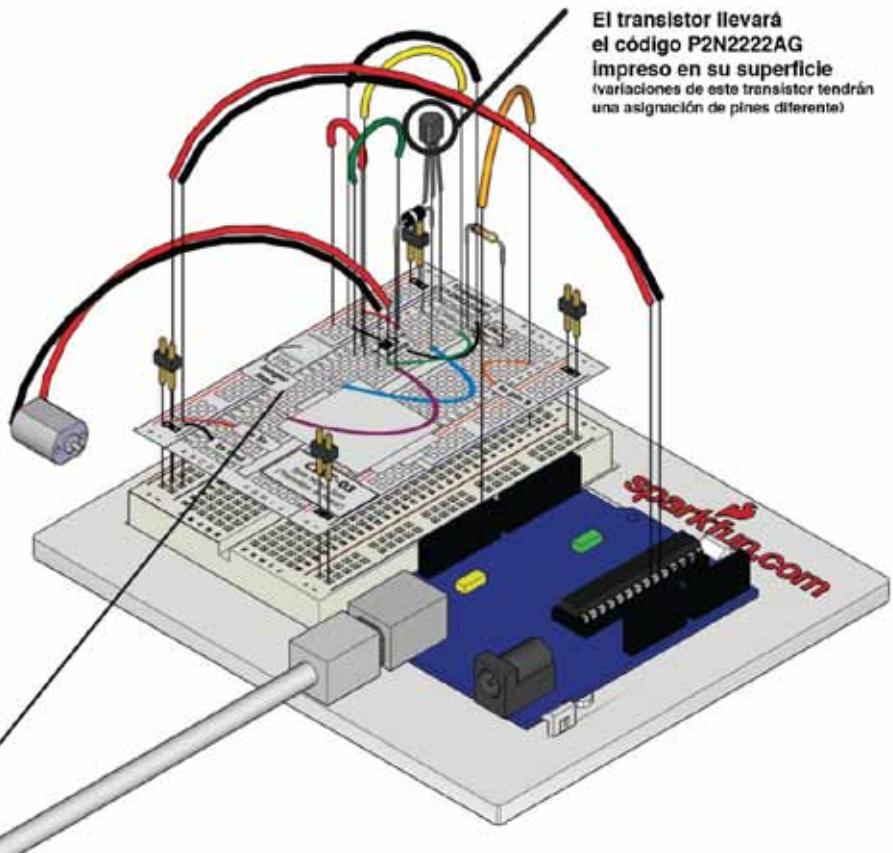
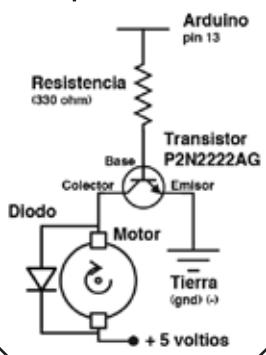


:: EL CIRCUITO

_Componentes



_Esquema



_Internet

Descarga una plantilla para el **CIRC-03** en
http://reflexiona.biz/plantilla_circ03

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ03

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ03. Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_El motor no da vueltas

Si has utilizado un transistor que no es el que incluye este kit, vuelve a comprobar en su hoja de especificaciones que los pines son compatibles con los de un P2N2222AG (en muchos casos pueden estar invertidos)

_Sigue sin haber suerte

Si has utilizado un motor que no es el suministrado con el kit, vuelve a comprobar en su hoja de especificaciones que funcione a 5V y que no necesite más potencia.

_Sigue sin funcionar

A veces, la conexión entre la placa Arduino y el ordenador, se puede interrumpir. Prueba a desconectar y volver a conectar el cable USB.

:: MEJORANDO EL CIRCUITO

_Controlando la velocidad

Ya hemos practicado anteriormente con la capacidad de Arduino para controlar la intensidad de un LED. Ahora, usaremos esa misma característica para controlar la velocidad de un motor. Arduino realiza esta operación haciendo uso de una técnica llamada PWM o Power Wide Modulation (en castellano Modulación de Ancho de Pulso). Esta técnica está basada en la capacidad que tiene un Arduino para funcionar realmente rápido. En lugar de controlar el voltaje que llega desde un pin, Arduino encenderá y apagará repetidamente ese pin, y de forma extremadamente veloz. En el mundo de los ordenadores, esto supone ir de 0 a 5 voltios mucha veces por segundo, pero en el mundo de los seres humanos, lo vemos como si fuera voltaje. Por ejemplo, si el ciclo de Modulación de Ancho de Pulso está al 50% vemos la luz emitida por un LED regulada al 50%, ya que nuestros ojos no son lo suficientemente rápidos como para ver parpadear el LED. Esta teoría funciona del mismo modo con los transistores. ¿Qué no te lo crees? ¡Pues haz la prueba!

En la función **loop()** cambia el contenido por:

```
// motorOnThenOff();
motorOnThenOffwithSpeed();
// motorAcceleration();
```

Ahora carga el programa. Puedes cambiar la velocidad, cambiando las variables en **onSpeed** y **offSpeed**.

_Acelerando y decelerando

¿Por qué dejarlo solo en dos velocidades cuando puedes acelerar y desacelerar el motor? Para hacer esto solo tienes que cambiar el código en la función **loop()** para que ponga:

```
// motorOnThenOff();
// motorOnThenOffwithSpeed();
motorAcceleration();
```

Ahora carga el programa y observa como tu motor acelera poco a poco hasta alcanzar la máxima velocidad y luego vuelve a decelerar. Si quieres cambiar en tiempo que tarda en acelerar y decelerar, cambia el parámetro **delayTime** (un número más alto significa mayor tiempo de aceleración).

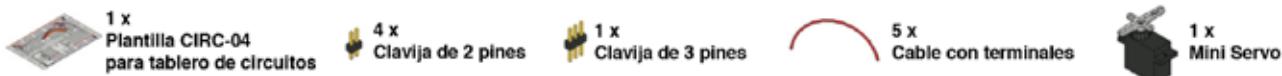
:: LO QUE ESTAMOS HACIENDO

Hacer que un motor eléctrico gire es muy divertido, pero cuando hay que llevar a cabo un proyecto en el que es necesario un control de movimiento más preciso, estos motores suelen quedarse un poco cortos. Una solución adecuada en este tipo de situaciones es la utilización de servomotores de modelismo. Estos motores se fabrican al por mayor, están disponibles en la mayoría de los establecimientos de electrónica y tienen precios que van desde los dos euros a los cientos de euros. En el interior de estos motores podemos encontrar un pequeño mecanismo (que permite movimientos más potentes) y algo de electrónica (que facilita el control). Un servomotor estándar se puede posicionar de 0 a 180 grados. La posición del servomotor se controla mediante un tiempo de pulso que va de 1,25 milisegundos (pulso equivalente a la posición 0°) a 1,75 milisegundos (pulso equivalente a la posición 180°). Siguiendo esta lógica, para conseguir posicionarlo a 90° haría falta un pulso de 1,5 milisegundos. Estos tiempos varían en función del fabricante. Si el pulso se envía cada 25-50 milisegundos el servomotor funcionará suavemente. Otra de las maravillosas características de Arduino es que dispone de una librería que permite controlar 2 servomotores (conectados a los pines 9 ó 10) utilizando una sola línea de código.

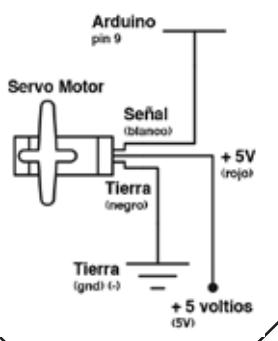


:: EL CIRCUITO

_Componentes



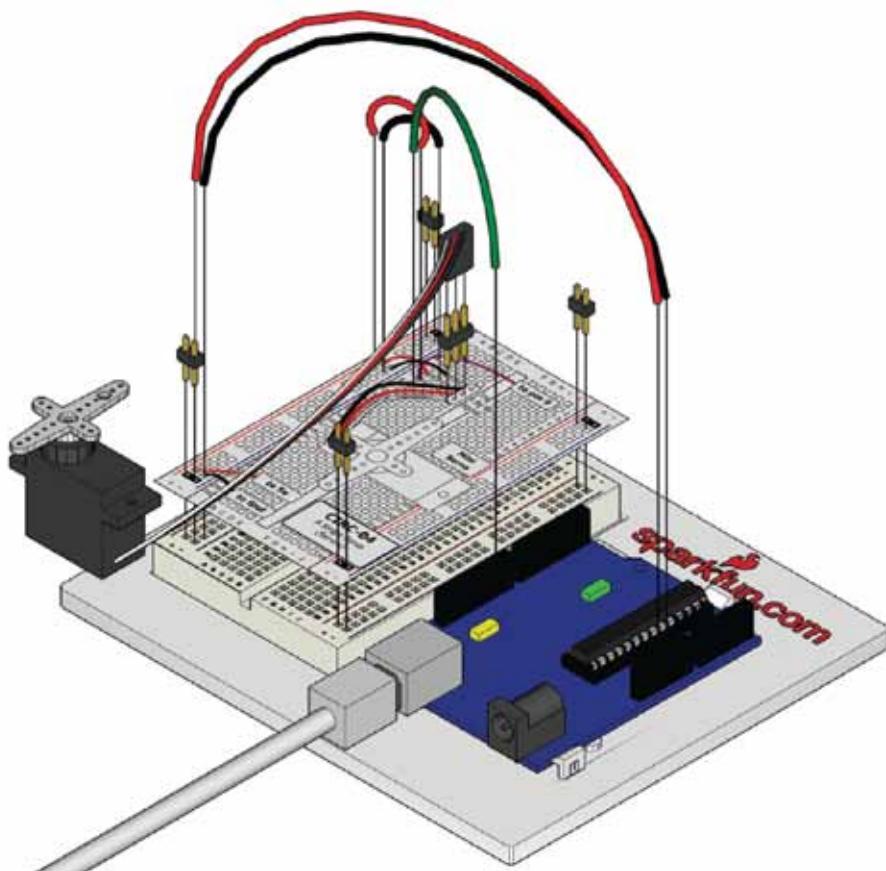
_Esquema



_Internet

Descarga una plantilla para el CIRC-04 en
http://reflexiona.biz/plantilla_circ04

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ04



:: EL CÓDIGO

No es necesario que escribas todo el texto, solo tienes que pulsar en **Archivo > Ejemplos > Servo > Sweep** (ejemplo extraído de página <http://Arduino.cc>). Échale un vistazo para ver otras grandes ideas)

:: NO FUNCIONA (3 cosas que puedes probar)

El servo no da vueltas

Incluso con los cables de colores sigue siendo fácil equivocarse y conectar un servomotor del revés. Este podría ser tu caso.

Sigue sin funcionar

Otro error que se suele cometer habitualmente es haber olvidado conectar la corriente a +5V y a la tierra (GND), cable rojo y cable negro respectivamente. ¿Has comprobado si están conectados?

Ajuste de la alimentación

Si el servomotor empieza a moverse pero acto seguido va a trompicones y parpadea una luz en tu placa de Arduino, esto significa que la fuente de alimentación que estás usando no es lo suficientemente potente. Usar una pila nueva (o una batería) en vez del puerto USB, debería resolver este problema.

:: MEJORANDO EL CIRCUITO

Control mediante un potenciómetro

Todavía tenemos que experimentar con las entradas de Arduino pero si quieras adelantarte unos pasos hay un programa pulsando en **Archivo > Ejemplos > Servo > Knob** donde se utiliza un potenciómetro (ver CIRC-08) para controlar el servomotor. Puedes encontrar las instrucciones en <http://www.arduino.cc/es/Tutorial/Knob>

Auto-sincronización

A pesar de que es muy fácil controlar un motor servo con la librería que incluye la IDE de Arduino, a veces es más divertido tratar de averiguar como programar algo por ti mismo ¡INTÉNTALO! En anteriores ejercicios, ya hemos controlado directamente el pulso enviado desde un pin, así que puedes utilizar este mismo método para controlar servomotores desde cualquiera de los 20 pines disponibles de tu Arduino (para hacer esto necesitas cambiar el parámetro **pulseTime**).

```
int servoPin = 9;

void setup(){
pinMode (servoPin, OUTPUT);
}

void loop(){
int pulseTime = 2100;      // el número de microsegundos
                          // para posicionar el servo (1500 es aprox. 90 grados,
                          // 900 es aprox. 0 grados y 2100 es aprox. 180 grados)

digitalWrite(servoPin, HIGH);
delayMicroseconds(pulseTime);
digitalWrite(servoPin, LOW);
delay(25);
}
```

Grandes ideas

Los servomotores se pueden usar para hacer un montón de cosas interesantes. A continuación, puedes encontrar algunos proyectos basados en el uso de servomotores que son realmente interesantes:

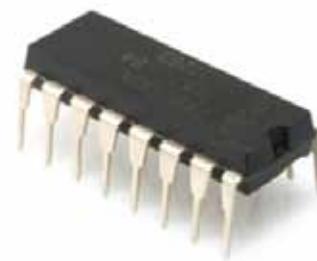
Xmas Hit Counter: <http://ardx.org/XMAS>

Open Source Robotic Arm (utiliza tanto un controlador de servo como un Arduino): <http://ardx.org/RARM>

Servo Walker: <http://ardx.org/SEWA>

:: LO QUE ESTAMOS HACIENDO

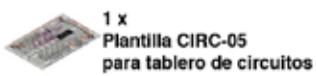
Es hora de empezar a jugar con chips o circuitos integrados (como se les llama habitualmente). La parte externa de un chip puede resultar realmente decepcionante. Para que te hagas una idea, el chip de la placa de Arduino (un microcontrolador) y el que vamos a utilizar en este circuito (un registro electrónico) se parecen mucho exteriormente, pero en realidad son completamente diferentes. El precio del chip ATmega de la placa de Arduino es de unos cuantos euros, mientras que el chip de registro electrónico 74HC595 cuesta tan solo unos céntimos. Este último, es un buen chip de aprendizaje y en el momento que te encuentres cómodo con su funcionamiento y seas capaz de entender su hoja de especificaciones (ver en <http://reflexiona.biz/74hc595>), el mundo de los chips será coser y cantar. El registro electrónico (también conocido como convertidor de serie a paralelo) te proporciona 8 salidas adicionales para controlar LEDs (o similares), usando solo 3 pines de Arduino. También se pueden conectar varios chips 74HC595 juntos, para disponer de un número de salidas casi ilimitado usando esos mismos tres pines. Para hacer uso de este chip, registras los datos y luego los bloqueas (los aseguras). Esto se consigue configurando el pin de datos, ya sea a **HIGH** o a **LOW**, y registrándolo; configurando el pin de nuevo y volviéndolo a registrar; y así, repitiendo esta operación hasta que has cambiado los 8 bits de datos. Despues pulsas el bloqueo y los 8 bits se transfieren a los pines del chip de registro electrónico. Suena complicado pero, una vez que le coges el truco, es realmente fácil.



Para disponer de más información sobre como funciona un registro eléctrico visita http://es.wikipedia.org/wiki/Registro_electrónico

:: EL CIRCUITO

_Componentes



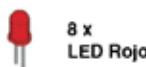
1 x
Plantilla CIRC-05
para tablero de circuitos



4 x
Clavija de 2 pines



1 x
Registro electrónico
74HC595



8 x
LED Rojo

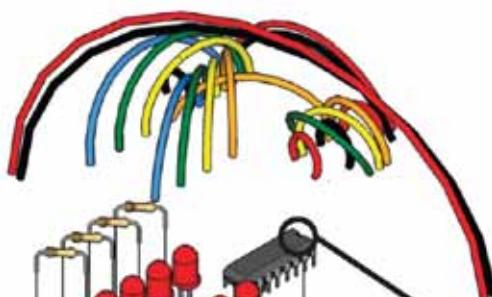
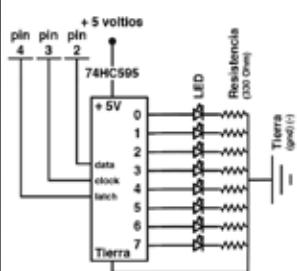


8 x
Resistencia de 330 Ohmios
(Naranja-Naranja-Marrón)



19 x
Cable con terminales

_Esquema



Hay una endidura con forma de media luna.
Esta va en la parte de arriba.

_Internet

Descarga una plantilla para el CIRC-05 en
http://reflexiona.biz/plantilla_circ05

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ05

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ05
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_El LED de encendido de Arduino se apaga

Esto pasa algunas veces y es debido a que el chip de registro electrónico está conectado al revés. Si lo cambias rápidamente de posición no tiene por qué romperse nada.

_Sigue sin funcionar

Sentimos mucho que este manual suene en ocasiones como un disco rallado, pero es que es muy probable que el problema sea algún cable que esté cruzado o mal conectado.

_Te invade la frustración

Este circuito es muy simple y muy complejo al mismo tiempo. Mándanos un correo electrónico a shop@reflexiona.biz. Queremos saber que problema has tenido con este circuito para poder añadirlo en las próximas ediciones de esta guía.

:: MEJORANDO EL CIRCUITO

_Hacerlo de la manera más dura

Un Arduino convierte en fáciles las acciones más difíciles. El chip registro electrónico es un claro ejemplo de esto. En cualquier caso, una de las mejores características de Arduino es que puedes hacer las cosas tan simples o tan complejas como deseas. Vamos a ver un ejemplo de esto. En tu línea de `loop()` haz el siguiente cambio:

```
updateLEDs(i) =====> updateLEDsLong(i);
```

Carga el programa y fíjate que nada ha cambiado. Si te fijas en el código se puede ver como nos estamos comunicando con el chip bit a bit. Para obtener más detalles visita http://es.wikipedia.org/wiki/Serial_Peripheral_Interface.

_Controlando LEDs individuales

Es hora de controlar los LEDs de forma similar a como lo hacímos en el ejercicio CIRC-02. Como los estados de los LEDs están almacenados en un solo Byte (un valor de 8 bits), para ver al detalle cómo funciona vete a http://es.wikipedia.org/wiki/Sistema_binario. Un microcontrolador Arduino es muy bueno manipulando bits y tiene un montón de operadores que nos pueden ayudar con esta tarea. Puedes encontrar más detalles sobre matemática de nivel de bits en http://es.wikipedia.org/wiki/Operador_a_nivel_de_bits.

Nuestra implementación. Sustituye el código de `loop()` por:

```
int delayTime = 100;           //el número de milisegundos de retardo
                               //entre la actualización de los LEDs

for (int i = 0; i < 8; i++){
    changeLED(i, ON);
    delay(delayTime);
}

for (int i = 0; i < 8; i++){
    changeLED(i, OFF);
    delay(delayTime);
}
```

Cargar este código hará que las luces se enciendan una detrás de otra, para apagarse después del mismo modo.

_Más animaciones

Ahora es cuando las cosas se ponen interesantes. Si retomas el código utilizado en el CIR-02 (Múltiples LEDs) puedes comprobar que cambiábamos los LEDs usando `digitalWrite(led, State)`. Este código es igual que la rutina que hemos escrito con `changeLED(led, State)`. Puedes utilizar las animaciones que creaste para el CIRC-02 con tan solo copiar el código en este sketch y cambiando todos los `digitalWrite()` por `changeLED()`. ¡Realmente potente! ¿No te parece? ¡Atención! Vas a tener que cambiar otras cosas del código pero sigue las indicaciones de los errores de compilación y terminarás haciendo que funcione correctamente).

:: LO QUE ESTAMOS HACIENDO

Hasta el momento hemos controlado luz, movimiento y electrones. Ha llegado la hora de abordar el sonido. Pero... ¡El sonido es un fenómeno analógico! ¿Cómo hará nuestro Arduino para enfrentarse a este reto?



Una vez más, nos aprovecharemos de la increíble velocidad de procesado de nuestra placa para simular un comportamiento analógico. Para poner de nuevo en práctica esta estupenda habilidad de nuestro microcontrolador, conectaremos un zumbador piezoelectrónico a uno de los pines digitales de nuestro Arduino.

Un zumbador piezoelectrónico emite un “click” cada vez que es accionado por la corriente que lo atraviesa. Si lo accionamos con la frecuencia adecuada, estos “clicks” sonarán de forma continua y nos permitirán reproducir diferentes notas. Por ejemplo, si enviamos pulsos con una frecuencia de 440 Hz (veces por segundo) conseguiremos la nota “La”, o si enviamos pulsos con una frecuencia de 261 Hz conseguiremos la nota “Do”. Vamos a experimentar con esto y hacer que nuestro Arduino toque “Twinkle Twinkle Little Star”.

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-06
para tablero de circuitos



4 x
Clavija de 2 pinos

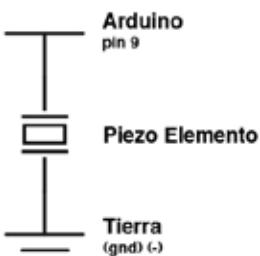


1 x
Zumbador Piezoelectrónico



4 x
Cable con terminales

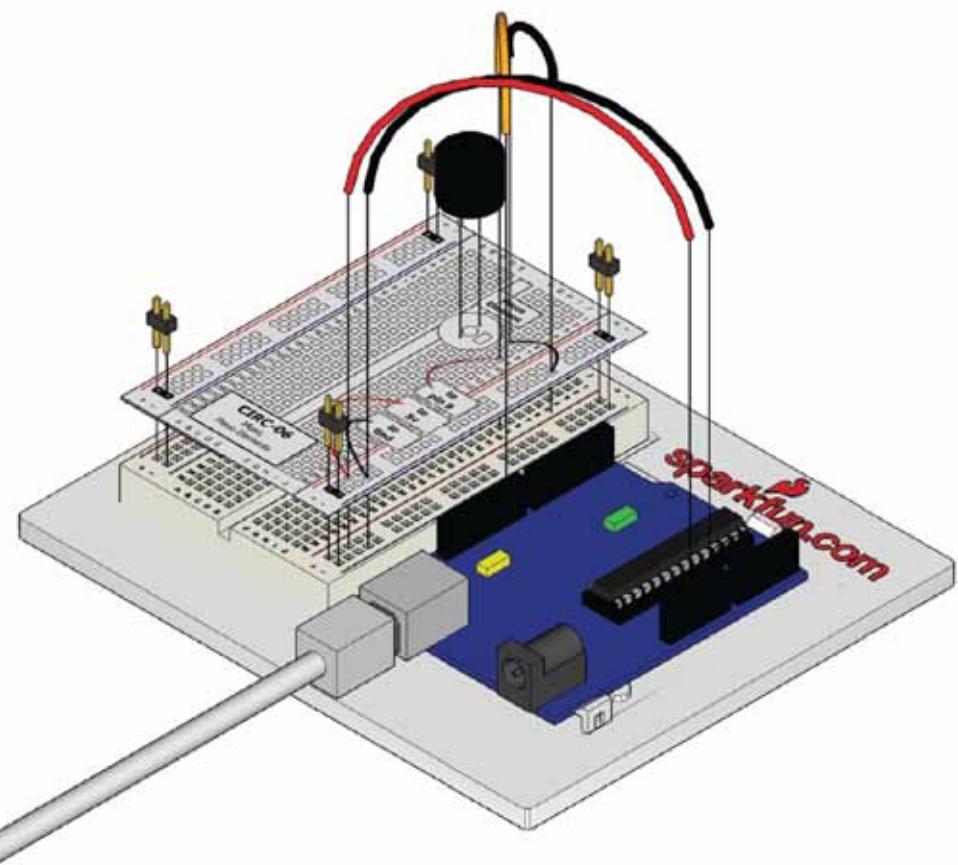
_Esquema



_Internet

Descarga una plantilla
para el CIRC-06 en
http://reflexiona.biz/plantilla_circ06

Echa un vistazo
al video de ensamblaje en
http://reflexiona.biz/video_circ06



:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ06
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_No hay sonido

En función del tamaño y la forma del zumbador piezoelectrónico es fácil confundir cuáles son los pines de la breadboard en los que has introducido los pines del zumbador. Comprueba de nuevo donde has pinchado estos pines.

_No puedes pensar cuando oyes la melodía

La melodía del zumbador piezoelectrónico puede llegar a ser un poco molesta. Solo tienes que desconectar el zumbador piezoelectrónico mientras piensas, carga tu nuevo programa y conecta de nuevo el zumbador.

_Te has cansado de escuchar “Twinkle Twinkle Little Star”

El código que has copiado ha sido escrito por “alguien cualquiera” así que tú también puedes escribir tus propias canciones. Échale un vistazo a los ejemplos de código que hay más abajo antes de ponerte con ello.

:: MEJORANDO EL CIRCUITO

_Jugando con la velocidad

El tiempo para cada nota está calculado basándose en variables, de tal forma que podamos cambiar el sonido de cada nota o el tiempo que dura. Para cambiar la velocidad de la melodía solo tienes que cambiar una línea:

```
int tempo = 300; =====> int tempo = (nuevo número)
```

Cámbialo a un número mayor para que la melodía suene mas lento, o a un número menor para que vaya más rápido.

_Ajustando las notas

Si estás preocupado por que las notas suenan un poco desafinadas, esto también tiene remedio. Las notas han sido calculadas basándose en la fórmula situada en el bloque de comentarios de la parte superior del programa. Pero para ajustar las notas de forma individual solo tienes que ajustar sus valores hacia arriba o hacia abajo en la matriz de **tones[]** hasta que suenen correctamente. Cada nota se corresponde con su nombre en la matriz de **names[]** (p. Ej.: C = 1915)

```
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
```

_Componiendo tu propias melodías

El programa esta configurado para que suene “Twinkle Twinkle Little Star” pero la forma en la que he sido programado permite cambiar la canción de forma rápida y sencilla. Cada canción se define mediante un **int** y dos matrices, la longitud del **int** define el número de notas, la primera matriz **notes[]** define cada nota y la segunda **beats[]** define cuanto tiempo suena cada nota.

Algunos ejemplos:

Twinkle Twinkle Little Star

```
int length = 15;
char notes[] = "ccggaagffeeddc ";
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 4 };
```

Feliz Cumpleaños (primera línea)

```
int length = 13;
char notes[] = "ccdcfeccdcgf ";
int beats[] = { 1, 1, 2, 2, 4, 1, 1, 2, 2, 2, 4 };
```

:: LO QUE ESTAMOS HACIENDO

Hasta el momento nos hemos centrado exclusivamente en las salidas. Ha llegado la hora de hacer que Arduino “escuche, vea y sienta”. Empezaremos con un simple botón pulsador. No te procures, el cableado del botón pulsador es realmente sencillo. En todo el circuito, solo hay un componente que puede dar la sensación de estar fuera de lugar: la resistencia de polarización (en este caso la resistencia de 10k Ohmios). Esta resistencia ha sido incluida por que Arduino no siente de la misma forma que lo hacemos los seres humanos (p. ej.: botón pulsado, botón no pulsado), si no que atiende al voltaje que hay en el pin y decide si es **HIGH** o **LOW**. El botón pulsador esta configurado para forzar el pin de Arduino a **LOW** cuando es pulsado. En cualquier caso, cuando el botón no está pulsado el voltaje del pin “flotará” (pudiendo provocar fallos ocasionales). Para conseguir que Arduino lea con exactitud el modo **HIGH** cuando el botón no está pulsado, se añade la resistencia de polarización.



*Nota: el primer programa de ejemplo solo utiliza uno de los dos botones.

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-07
para tablero de circuitos



4 x
Clavija de 2 pines



2 x
Botón
Pulsador



7 x
Cable con terminales



1 x
LED Rojo

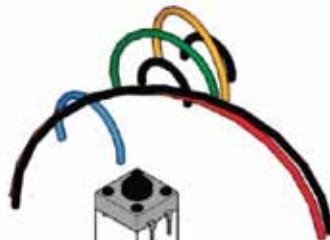
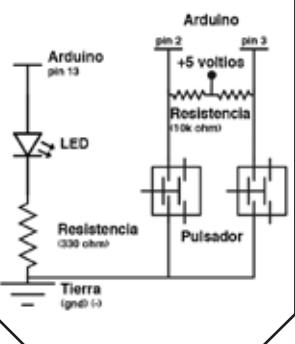


1 x
Resistencia de 330 Ohmios
(Naranja-Naranja-Marrón)



2 x
Resistencia de 10k Ohmios
(Marrón-Negro-Naranja)

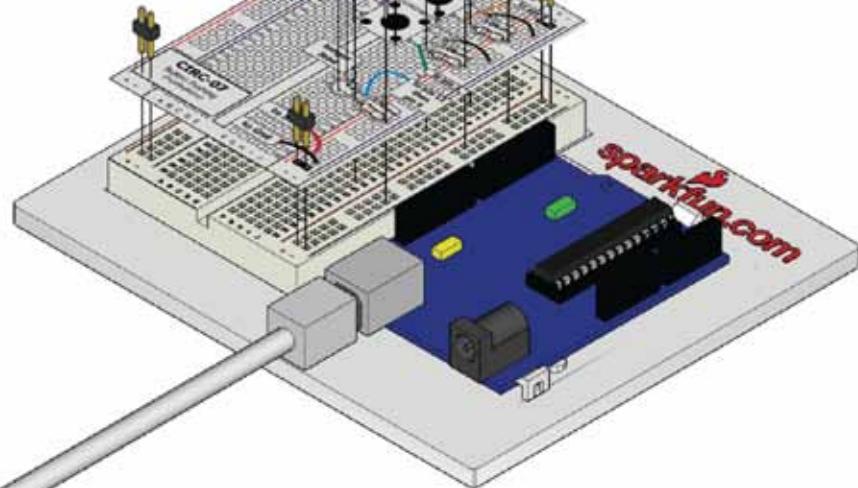
_Esquema



_Internet

Descarga una plantilla
para el **CIRC-07** en
http://reflexiona.biz/plantilla_circ07

Echa un vistazo
al video de ensamblaje en
http://reflexiona.biz/video_circ07



:: EL CÓDIGO

No es necesario que escribas todo el texto, solo tienes que pulsar en: **Archivo > Ejemplos > 2.Digital > Button** (ejemplo extraído de la página <http://Arduino.cc>. Échale un vistazo para ver otras grandes ideas)

:: NO FUNCIONA (3 cosas que puedes probar)

La luz no se enciende

El botón que usamos en este kit tiene forma cuadrada. Por este motivo, es muy fácil equivocarse y colocarlo de forma incorrecta. Gira el pulsador 90° sobre la breadboard y comprueba si ahora funciona.

La luz no se apaga gradualmente (cuando mejoramos el circuito)

Un fallo que comentemos constantemente es que cuando cambias de un simple encendido/apagado a un apagado gradual no cambiamos del pin 13 al pin 9 el cable conectado al LED.

No estás impresionado

No te inquietes, todos los circuitos que hemos visto en esta guía son elementales para que aprender jugando con los componentes te resulte muy fácil, pero una vez que juntas en un solo circuito todos estos componentes que hemos visto (y que vamos a ver)... ¡el límite es tu imaginación!

:: MEJORANDO EL CIRCUITO

Botón de encendido, botón de apagado

El primer ejemplo con los botones pulsadores puede ser un poco decepcionante. Vamos a complicarlo un poco más haciendo que un botón encienda el LED y el otro botón lo apague. Sustituye el código por:

```
int ledPin = 13;           //Escoge el pin para el LED
int inputPin1 = 3;          //Botón 1
int inputPin2 = 2;          //Botón 2

void setup() {
    pinMode(ledPin, OUTPUT);           //Declarar el LED como salida
    pinMode(inputPin1, INPUT);         //Botón 1
    pinMode(inputPin2, INPUT);         //Botón 2
}

void loop() {
    if (digitalRead(inputPin1) == LOW)
        digitalWrite(ledPin, LOW);      //Enciende el LED
    else if (digitalRead(inputPin2) == LOW)
        digitalWrite(ledPin, HIGH);     //Apaga el LED
}
```

Y carga este programa en tu placa de Arduino para empezar a encender y a apagar el LED.

Apagando y encendiendo gradualmente

Utilicemos ahora los botones pulsadores para controlar una señal analógica. Para hacer esto, necesitas cambiar el cable que conecta el pin 13 con la patilla positiva del LED al pin 9, y cambiar también el código siguiente:

```
int ledPin = 13; =====> int ledPin = 9;
```

Ahora añades al principio del sketch:

```
int value = 0;
```

y cambias el código de **loop()** para que ponga:

```
void loop(){
    if (digitalRead(inputPin1) == LOW) { value--; }
    else if (digitalRead(inputPin2) == LOW) { value++; }
    value = constrain(value, 0, 255);
    analogWrite(ledPin, value);
    delay(10);
}
```

Cambiando la velocidad de regulación

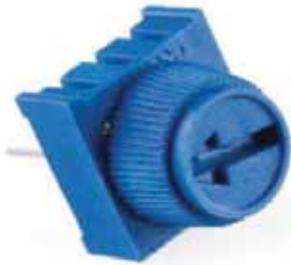
Si quieres que el LED se apague y se encienda de forma gradual pero más deprisa o más despacio de lo establecido en el código anterior, solo es necesario que cambies una línea de código:

```
delay(10); =====> delay(nuevo número);
```

Para que la regulación sea más rápida tienes que escribir un número más bajo y para que más lenta tienes que escribir un número más alto.

:: LO QUE ESTAMOS HACIENDO

Además de los pines digitales, Arduino también dispone de 6 pines que pueden ser utilizados para señales de entrada analógicas. Estas entradas analógicas admiten un voltaje de 0 a 5 voltios y lo convierten en un número digital entre 0 (equivalente a 0 voltios) y 1024 (equivalente a 5 voltios), o lo que es lo mismo, con una resolución de 10 Bits. Un componente que hace muy buen uso de esta característica de Arduino es el potenciómetro (también conocido como resistencia variable). Cuando un potenciómetro está conectado a la placa con 5 voltios atravesando los pines de sus extremos, el pin central (conocido también como cursor) reconocerá un valor entre 0 y 5 voltios en función del ángulo al que haya sido girado (p. ej.: con el potenciómetro girado a la mitad serían 2,5 voltios). En nuestro sketch de Arduino, podemos utilizar como una variable los valores que son enviados desde el potenciómetro al pin analógico al que este conectado el cursor.



:: EL CIRCUITO

_Componentes

1 x Plantilla CIRC-08 para tablero de circuitos

4 x Clavija de 2 pines

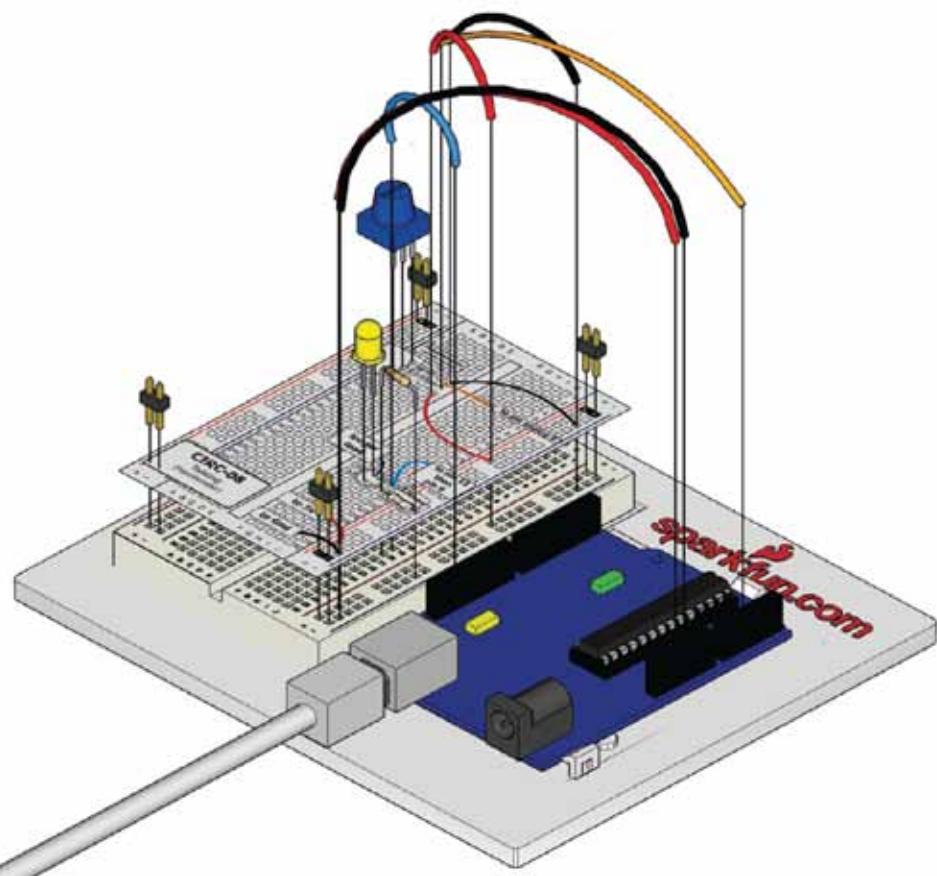
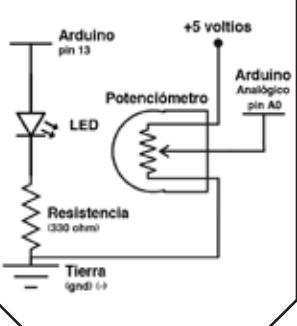
1 x Potenciómetro 10k Ohmios

6 x Cable con terminales

1 x LED Amarillo

1 x Resistencia de 330 Ohmios (Naranja-Naranja-Marrón)

_Esquema



_Internet

Descarga una plantilla para el CIRC-08 en
http://reflexiona.biz/plantilla_circ08

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ08

:: EL CÓDIGO

No es necesario que escribas todo el texto, solo tienes que pulsar en: **Archivo > Ejemplos > 3.Analog > AnalogInput** (ejemplo extraído de la página <http://Arduino.cc>. Échale un vistazo para ver otras grandes ideas)

:: NO FUNCIONA (3 cosas que puedes probar)

_Funciona de forma intermitente

Probablemente, esto es debido a una conexión dudosa con los pines del potenciómetro. Podemos solucionar este problema fácilmente. Basta con apretar el potenciómetro un poco en del tablero de circuitos.

_No funciona

Asegúrate de no haber conectado accidentalmente el pin central del potenciómetro al pin digital 2 en lugar de conectarlo al pin analógico 2. (Los pines analógicos son la línea de terminales que está debajo de los terminales de alimentación)

_Sigue sin funcionar

Intenta hacer funcionar el circuito dándole la vuelta (poniéndolo boca abajo). A veces, así sí funciona.

:: MEJORANDO EL CIRCUITO

_Interruptor de umbral

Puede que en algún momento quieras encender o apagar la señal de salida cuando un valor exceda cierto umbral. Para hacer esto con un potenciómetro cambia el código de **loop()** por:

```
void loop(){
    int threshold = 512;
    if (analogRead(sensorPin) > threshold)
        digitalWrite (ledPin, HIGH);
    else (digitalWrite(ledPin, LOW));
}
```

Esto hará que el LED se encienda cuando el valor sea superior a 512 (más o menos la mitad del recorrido del potenciómetro). Puedes ajustar la sensibilidad cambiando el valor de **threshold**.

_Regulación

Ahora vamos a controlar el brillo del LED directamente desde el potenciómetro. Para hacer esto primero tenemos que cambiar el pin al que está conectado el LED. Traslada el cable desde el pin 13 al pin 9 y cambia la siguiente línea de código:

```
int ledPin = 13; =====> int ledPin = 9;
```

Ahora cambias el código de **loop()** para que ponga:

```
void loop(){
    int potPin = 1024;
    int value = analogRead(potPin) / 4;
    analogWrite(ledPin, value);
}
```

Carga el código y observa como tu LED se regula en función del giro del potenciómetro.

***Nota:** el motivo por el que dividimos el valor **value** entre 4 es que la función **analogRead()** devuelve un valor entre 0 y 1024 (10 bits), y **analogWrite()** recibe un valor entre 0 y 255 (8 bits).

_Controlando un servo

Este es un excelente ejemplo en el que se funden un par de circuitos. Conecta el servo como hiciste en el CIRC-04, después abre el programa de ejemplo en **Archivo > Ejemplos > Servo > Knob**

Carga el programa en tu placa de Arduino. Observa como el eje del servo gira a la vez que giras el potenciómetro.

:: LO QUE ESTAMOS HACIENDO

Mientras que recibir una señal de entrada puede ser realmente útil para los experimentos que están controlados por personas... ¿Qué es lo que podemos utilizar cuando queremos que sea el propio entorno el que controle nuestros experimentos? Pues se utilizan los mismos principios, pero en vez de un potenciómetro (resistencia variable basada en la rotación de un eje) utilizamos, por ejemplo, una fotorresistencia (resistencia variable basada en la cantidad de luz).



Arduino no puede recibir directamente el valor de la resistencia (puede interpretar el voltaje), por lo que tenemos que configurar un divisor de tensión (http://es.wikipedia.org/wiki/Divisor_de_tensión). Se puede calcular el voltaje exacto en el pin de entrada pero en nuestro caso (solo queremos distinguir la iluminación relativa) podemos experimentar con los valores y ver que es lo que mejor funciona para nuestro caso. Obtendremos un valor bajo cuando el sensor está bien iluminado y un valor alto cuando está a oscuras.

:: EL CIRCUITO

_Componentes

1 x Plantilla CIRC-09 para tablero de circuitos

4 x Clavija de 2 pines

1 x Fotorresistencia

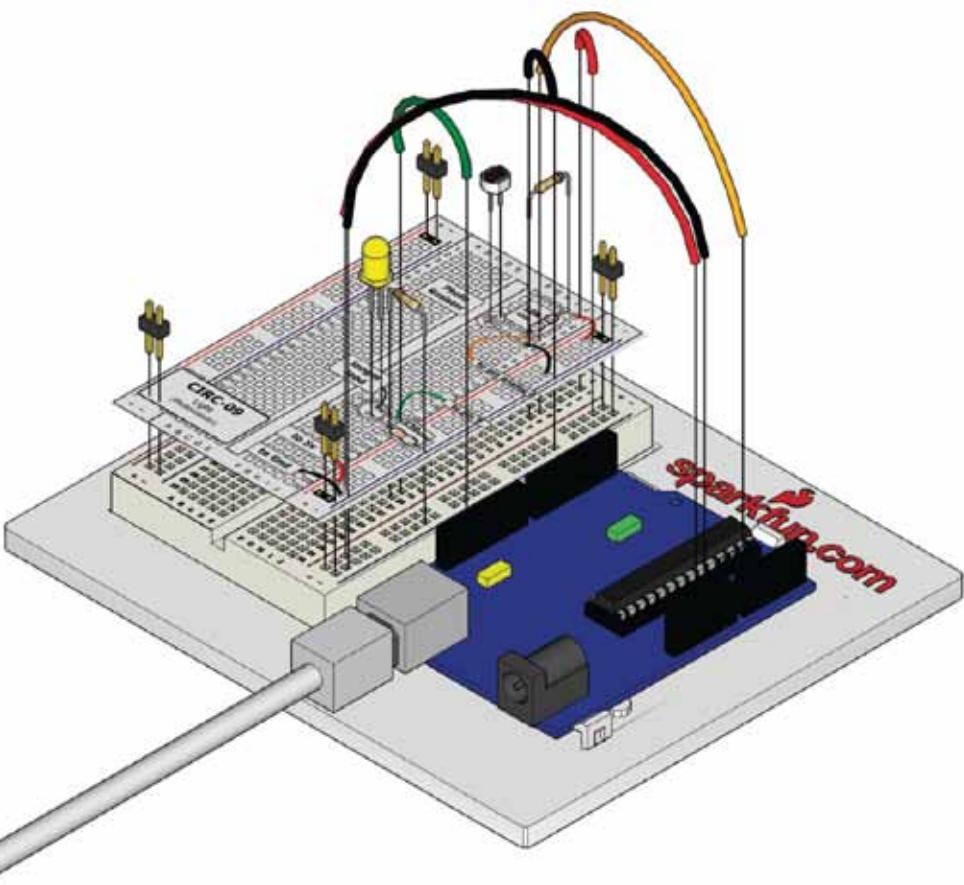
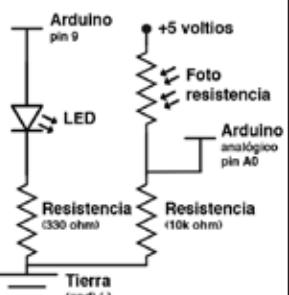
6 x Cable con terminales

1 x LED Amarillo

1 x Resistencia de 330 Ohmios (Naranja-Naranja-Marrón)

1 x Resistencia de 10k Ohmios (Marrón-Negro-Naranja)

_Esquema



_Internet

Descarga una plantilla para el CIRC-09 en http://reflexiona.biz/plantilla_circ09

Echa un vistazo al video de ensamblaje en http://reflexiona.biz/video_circ09

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ09.
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_El LED se mantiene apagado

Este es el típico error que se repite una y otra vez, por lo menos hasta que inventen un LED que funcione en los dos sentidos. Desconéctalo del tablero de circuitos y conéctalo al revés.

_No responde a los cambios de luz

Debido a que los espacios entre las patillas de las fotorresistencias no son estándar, es muy fácil colocarlas mal. Comprueba que has conectado cada patilla en su sitio.

_Sigue sin funcionar

Puede que te encuentres en una habitación que es demasiado clara o demasiado oscura. Prueba a apagar o encender las luces para intentar que funcione. Si dispones de una linterna a mano prueba con ella.

:: MEJORANDO EL CIRCUITO

_Invertir la respuesta

Es posible que quieras que el LED responda a la contra. No te preocupes, es muy fácil conseguir esto con tan solo cambiar:

```
analogWrite(ledPin, lightLevel); =====> analogWrite(ledPin, 255 - lightLevel);
```

Carga el programa y observa como se produce el cambio.

_Luz nocturna

Ahora vamos a usar nuestras recién adquiridas capacidades de percepción de la luz para controlar un servo (y al mismo tiempo realizar un pequeño pirateo del código de Arduino). Tienes que conectar un motor servo al pin 9 (como en el CIRC-04). Despues abre el ejemplo de KNOB (el mismo que usábamos en el CIRC-08) **Archivo > Ejemplos > Servo > Knob**. Carga el programa y observa como funciona sin necesidad de modificar prácticamente nada.

_Utilizar todo el recorrido del servo

Habráis observado que el motor servo solo funciona en una porción limitada de todo el recorrido. Esto es debido a que con el circuito divisor de tensión que usamos el voltaje en el pin analógico A0 no recibe tensión de 0 a 5V, sino que recibe dos valores intermedios (estos valores varían en función de tu configuración).

Para arreglar esto juega con la línea:

```
val = map (val, 0, 1023, 0, 179);
```

Para obtener más pistas sobre lo que tienes que hacer visita: <http://arduino.cc/es/Reference/Map>

:: LO QUE ESTAMOS HACIENDO

¿Cuál va a ser el próximo fenómeno que vamos a medir con nuestro Arduino? ¡TEMPERATURA! Para hacer esto vamos a utilizar un complejo circuito integrado que está oculto dentro de una envolvente idéntica a la de nuestro transistor P2N222AG. Este circuito integrado dispone de tres pines (tierra, señal, +5 voltios) y es realmente fácil de usar. Emite una señal de salida de 10mV (milivoltios) por grado centígrado a través del pin de señal. Para permitir temperaturas bajo cero hay una compensación de 500mV ($25^{\circ}\text{C} = 750\text{mV}$, $0^{\circ}\text{C} = 500\text{mV}$). Para convertir los valores digitales en grados centígrados haremos uso de algunas de las habilidades matemáticas de Arduino. Después, para mostrar los resultados, utilizaremos una de las características más potentes de la IDE de Arduino, la ventana de depuración. Extraeremos el valor obtenido a través de una conexión serie para visualizarlo en la pantalla. ¡VAMOS A POR ELLO! Solo una cosa más. Este circuito hace uso del monitor serie de la IDE de Arduino. Para abrirlo, primero tienes que arrancar el programa y pulsar el botón que parece una antena.



Puedes encontrar la hoja de especificaciones del TMP36 en <http://reflexiona.biz/tmp36>

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-10
para tablero de circuitos



4 x
Clavija de 2 pines

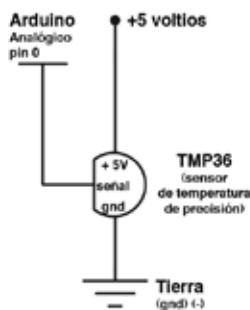


1 x
Sensor de temperatura
TMP36



5 x
Cable con terminales

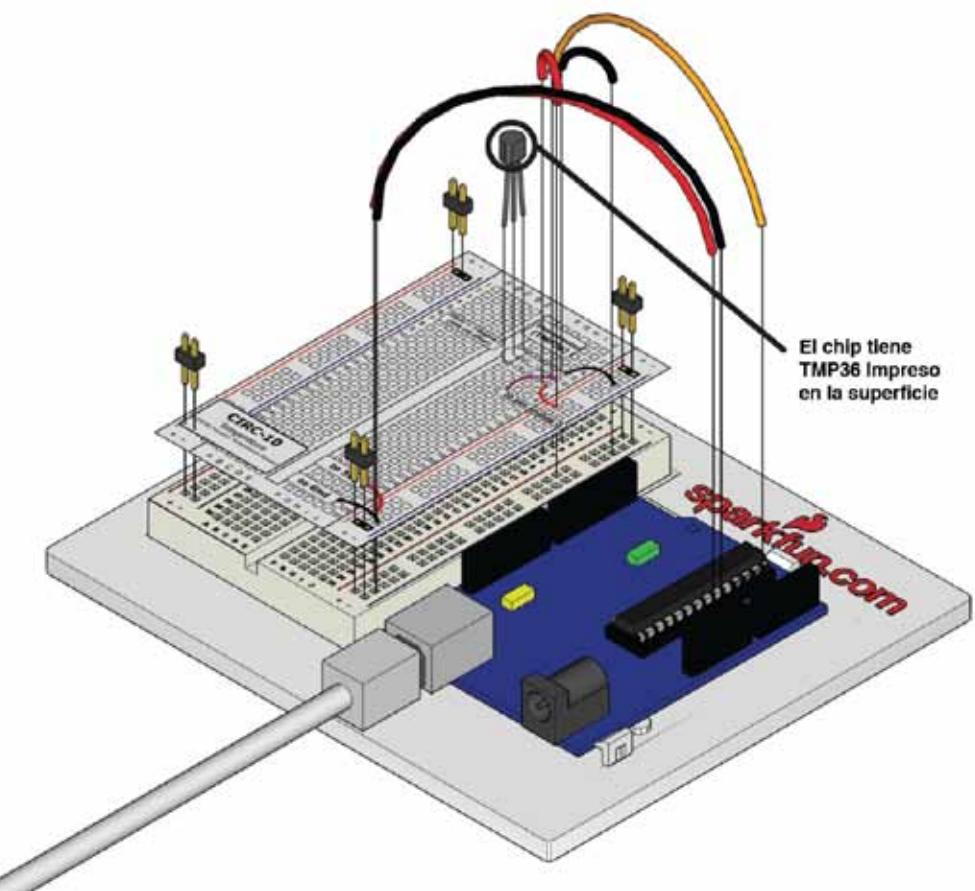
_Esquema



_Internet

Descarga una plantilla para el CIRC-10 en http://reflexiona.biz/plantilla_circ10

Echa un vistazo al video de ensamblaje en http://reflexiona.biz/video_circ10



:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ10.
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_Parece que no pasa nada

Este programa no dispone de un indicador que te diga que está funcionando. Para ver los resultados tienes que monitorizarlo desde la IDE de Arduino (instrucciones en el apartado LO QUE ESTAMOS HACIENDO).

_Lo que se ve en el monitor serie no tiene sentido

Esto sucede por que la monitorización en serie recibe los datos a una velocidad diferente de lo esperado. Para solucionar esto pulsa el cuadro desplegable donde pone "XXXX baudio" y cámbialo a "9600 baudio".

_El valor de temperatura no cambia

Intenta generar calor apretando el sensor con tus dedos, o colocando una bolsa de hielo encima para enfriarlo.

:: MEJORANDO EL CIRCUITO

_Enviando voltaje

Esto es sencillamente cuestión de cambiar una línea. Nuestro sensor emite 10mv (milivoltios) por grado centígrado, así que para conseguir una tensión sencillamente mostramos el resultado de `getVoltage()`.

Borra la línea `temperature = (temperature - .5) *100;`

_Enviando grados Farenheit

Una vez más con hacer un simple cambio que requiere algo de matemáticas.

Para cambiar de °C a °F utilizamos la fórmula: $(F = C \times 1.8 + 32)$

Añade la línea `temperature = (((temperature - .5) *100)*1.8) + 32;`

antes de `Serial.println(temperature);`

_Una salida con más información

Añadimos un mensaje a la salida serie para hacer que lo que aparece en el monitor serie sea más informativo. Para hacer esto tienes que volver al código original. Después, haz el siguiente cambio:

```
Serial.println(temperature); =====> Serial.print(temperature);
                                         Serial.println(" grados centigrados");
```

Ese cambio en la primera línea significa que cuando recibamos la siguiente salida aparecerá en la misma línea, entonces añadimos el texto informativo y una nueva línea.

_Cambiar la velocidad de los datos serie

Si alguna vez quieras recibir muchos datos a través del puerto serie, la esencia es la línea de tiempo. Estamos transmitiendo a 9600 baudios pero es posible utilizar velocidades de transmisión mucho mayores. Para cambiar esto tienes hacer el siguiente cambio:

```
serial.begin(9600); =====> serial.begin(115200);
```

Carga el programa y enciende el monitor serie, después cambia la velocidad de 9600 baudios a 115200 baudios en el menú desplegable. Ahora está transmitiendo los datos 12 veces más rápido.

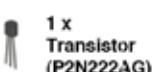
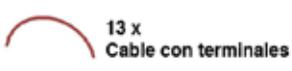
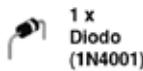
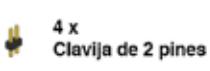
:: LO QUE ESTAMOS HACIENDO

El próximo circuito que vamos a hacer es una especie de test. Vamos a combinar los conocimientos adquiridos sobre transistores en el CIRC-03 para controlar un relé. Un relé es un interruptor mecánico controlado eléctricamente. Dentro de esa pequeña caja negra hay un electroimán que al recibir energía, abre o cierra otros circuitos eléctricos independientes (a menudo con un sonido característico que produce mucha satisfacción). Puedes encontrar relés de distintos tamaños, tan pequeños como la cuarta parte del que incluye este kit o tan grandes como un frigorífico, cada uno capaz de admitir una cantidad determinada de tensión de control. Es muy divertido trabajar con ellos por que incorporan cierto elemento físico. Mientras que toda la silicona con la que hemos jugado hasta el momento es divertida eventualmente, puede que lo que realmente deseas sea cablear cientos de interruptores para controlar una instalación espectacular. Los relés te permiten soñar con lograr esto, además de permitirte controlar todo con tu Arduino. Utilizando tecnología del presente controlamos la tecnología del pasado. En este circuito, el diodo 1N4001 actúa como diodo de retorno. Más detalles sobre porque está ahí en http://en.wikipedia.org/wiki/Flyback_diode.

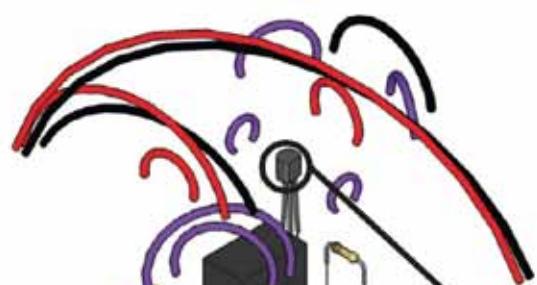
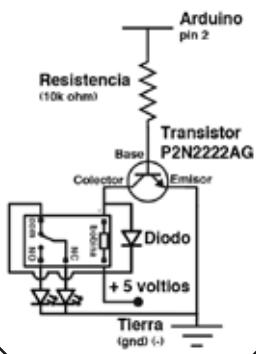


:: EL CIRCUITO

_Componentes



_Esquema



El transistor llevará el código P2N222AG impreso en su superficie (variaciones de este transistor tendrán una asignación de pines diferente)

_Internet

Descarga una plantilla para el CIRC-11 en http://reflexiona.biz/plantilla_circ11

Echa un vistazo al video de ensamblaje en http://reflexiona.biz/video_circ11

:: EL CÓDIGO

No es necesario que escribas todo el texto, solo tienes que pulsar en: **Archivo > Ejemplos > 1.Basic > Blink** (ejemplo extraído de la página <http://Arduino.cc>. Échale un vistazo para ver otras grandes ideas).

:: NO FUNCIONA (3 cosas que puedes probar)

_No pasa nada

El código del ejemplo utiliza el pin 13 y tenemos el relé conectado al pin 2. Asegúrate de haber echo este cambio en el código.

_No hace “click”

El transistor o la bobina del circuito no trabaja lo suficiente. Comprueba que el transistor está conectado de forma correcta.

_Sigue sin funcionar

El relé que incluye este kit esta pensado para ser soldado más que para ser utilizado con un tablero de circuitos. Es por este motivo que tienes que presionarlo bien para asegurarte de que funcione (puede que incluso se suelte en alguna ocasión).

:: MEJORANDO EL CIRCUITO

_Viendo el Pulso Electromagnético (EMP)

Sustituye el diodo por un LED. Podrás observar como parpadea cada vez que “rechaza” la tensión de la bobina cuando esta se apaga.

_Controlando un motor

En el CIR-03 controlábamos un motor utilizando un transistor. Si quisieras controlar un motor mayor, la mejor opción es utilizar un relé. Para poder hacer esto bastaría con retirar el LED rojo y conectar el motor en su lugar (acuérdate de derivar la resistencia de 330 ohmios)

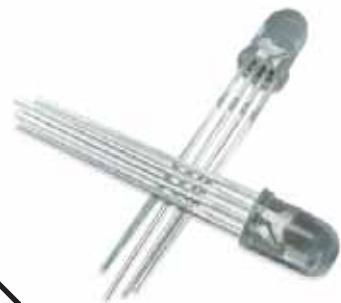
_Controlando la dirección de un motor

Una mejora algo más complicada para terminar. Para controlar la dirección de rotación de un motor eléctrico debemos ser capaces de cambiar de sentido el paso de la corriente por el mismo. Para hacer esto de forma manual usamos los cables. Para hacerlo de forma eléctrica necesitamos una cosa llamada Puente-h. Esto puede lograrse, usando un relé DPDT para controlar la dirección de rotación del motor. Cablea el circuito. Así de soplón, puede parecer complicado, pero es posible realizar esta prueba usando tan solo un par de cables extra. ¡INTÉNTALO!

CIRC - 12

:: LO QUE ESTAMOS HACIENDO

Cuando empezaste con el CIRC-01, seguro que te alegraste un montón al hacer que un LED rojo parpadease, pero... ¿A qué ya has superado esa fase? ¡Ahora los que quieras es desarrollar proyectos con todo tipo de colores: naranja, rosa, turquesa, morado y mucho más! :-)



Afortunadamente, existe una sencilla fórmula para conseguir todos esos colores a partir de un único LED, es decir, sin necesidad de disponer de un LED de cada color para realizar la mezcla, y se trata de un LED RGB. Un LED RGB es en realidad 3 LEDs dentro de una pequeña envolvente: uno LED rojo, un LED verde y un LED azul (Red-Green-Blue). Cuando enciendes a la vez dos o más colores, estos se mezclan para conseguir el resto de colores del espectro. El color que obtienes es la mezcla de las intensidades de los LEDs individuales de color rojo, verde y azul.

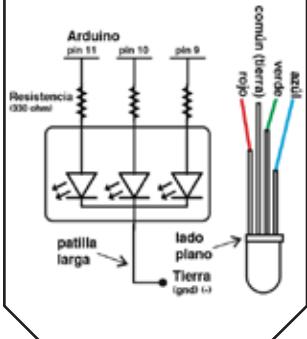
La intensidad de cada uno de ellos se controla mediante modulación de ancho de pulso (PWM), técnica que ya habíamos utilizado para controlar el brillo de un LED y la velocidad de un motor.

:: EL CIRCUITO

_Componentes



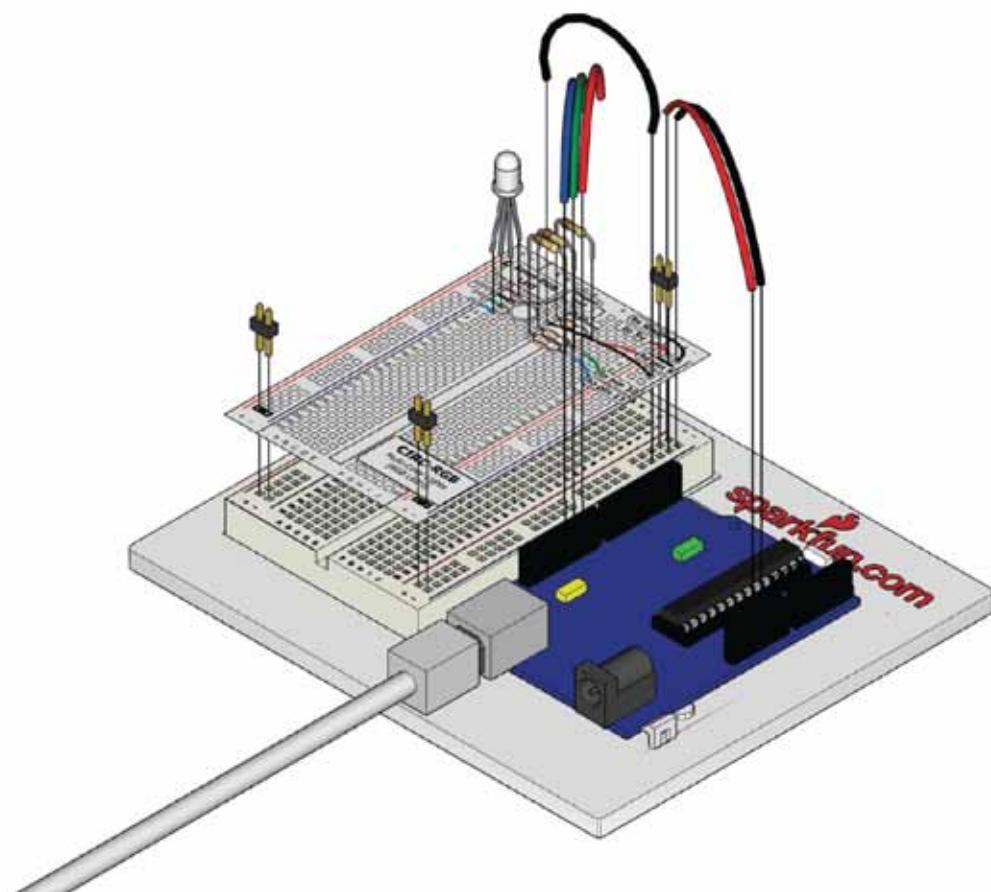
_Esquema



_Internet

Descarga una plantilla para el CIRC-12 en
http://reflexiona.biz/plantilla_circ12

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ12



:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ12
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_El LED no se enciende o muestra un color incorrecto

Como los cuatro pines del LED están muy pegados los unos a los otros, es bastante común conectar uno de ellos mal. Comprueba que cada uno de los pines está conectado donde debe de estarlo.

_Se ve todo rojo

Es muy probable que el diodo rojo del LED RGB sea mucho más brillante que los otros dos. Para conseguir que los colores estén más balanceados, conviene que utilices una resistencia mayor. También puedes ajustar esto mediante código:

```
analogWrite(RED_LED_PIN, redIntensity); =====> analogWrite(RED_LED_PIN, redIntensity/3);
```

_Buscas algo más

(Esto es un poco de propaganda) Si quieras ir un paso más allá con el tema de los LEDs RGB deberías echar un vistazo a todos los dispositivos disponibles en www.reflexiona.biz/shop

:: MEJORANDO EL CIRCUITO

_Utilizar códigos de color HTML

Si estás habituado a trabajar con páginas web es posible que prefieras especificar los colores de forma hexadecimal, igual que haces cuando estás usando HTML y CSS. El modo hexadecimal especifica un color utilizando una serie de letras y de números como "#FF0000" para el rojo ó "#800080" para el morado. Puedes aprender más sobre el funcionamiento de este tema en la Wikipedia (http://es.wikipedia.org/wiki/Colores_HTML). También encontrarás una lista de números hexadecimales para no tener que hacerlo tu mismo.

Descárgate el código desde http://reflexiona.biz/codigo_circ12b

_Usar un difusor

Una de las desventajas de usar un LED RGB (echo a partir de tres LEDs individuales) a la hora de generar colores es que, a veces, es posible distinguir claramente los colores de cada una de las distintas fuentes de luz. Un método para solucionar esto es buscar la forma de hacer que la luz sea más difusa, de forma que la mezcla de color sea mejor.

Para mejorar la efectividad de la mezcla de color, el LED suministrado con este kit dispone de una envolvente difusa (en vez de transparente). Si te parece que la luz sigue sin ser lo suficientemente difusa, puedes probar a colocar el LED detrás de un trozo de papel o de acrílico, o incluso dentro de una bola de ping pong o de poliestireno.

:: LO QUE ESTAMOS HACIENDO

Dicen que en la vida es importante ser flexible pero... ¿Cómo te las apañas cuando quieres medir la flexibilidad de un objeto? ¡Pues utilizas un sensor flexible!



Un sensor flexible utiliza una tira de carbón (o de plástico) para actuar como un potenciómetro o resistencia variable (acuerdate del CIRC-08) pero en vez de cambiar la resistencia girando un dispositivo rotatorio, lo cambias flexionando (doblando) el componente. Usamos de nuevo un divisor de tensión (acuerdate de los CIRC-08 y CIR-09) para detectar un cambio en la resistencia. El sensor se dobla en una dirección, y contra más se dobla, la resistencia es mayor. Dispone de un intervalo que va de los 10k Ohmios a los 35k Ohmios. En este circuito vamos a utilizar la curva de flexión del sensor flexible para controlar la posición de un servo.

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-13
para tablero de circuitos



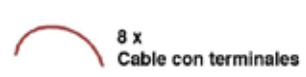
4 x
Clavija de 2 pines



1 x
Clavija de 3 pines



1 x
Sensor Flexible



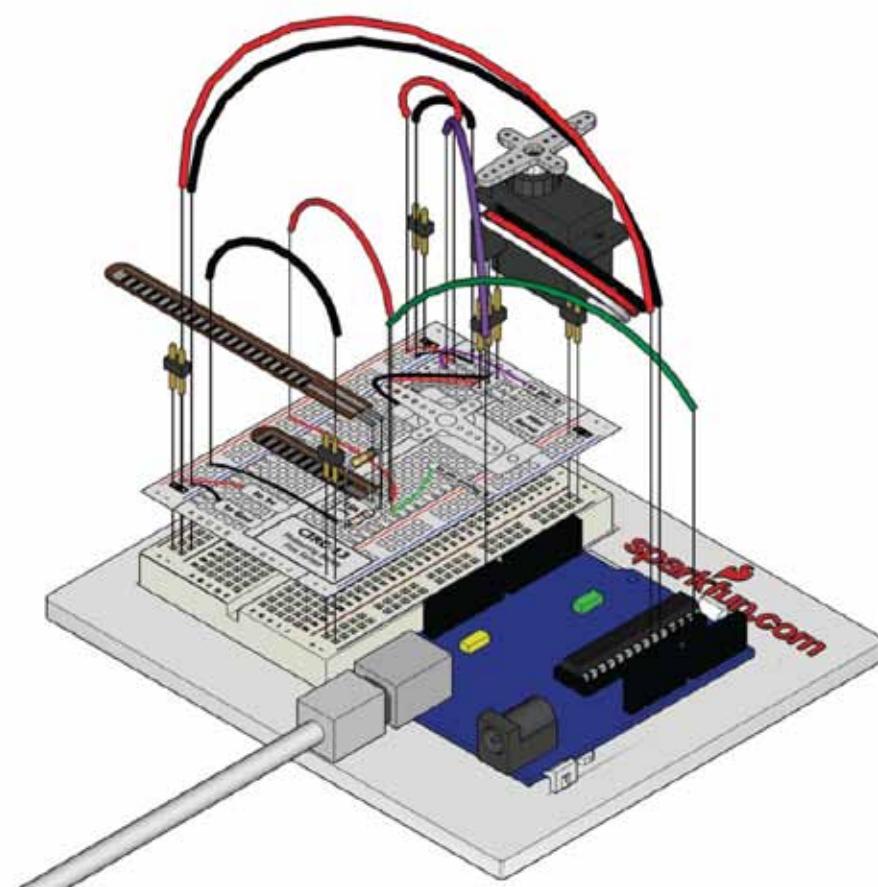
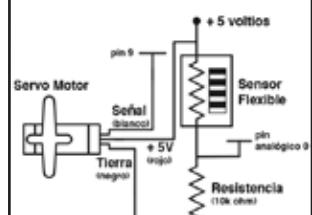
8 x
Cable con terminales

1 x
Resistencia 10k Ohmios
(Marrón-Negro-Naranja)



1 x
Mini Servomotor

_Esquema



_Internet

Descarga una plantilla para el CIRC-13 en
http://reflexiona.biz/plantilla_circ13

Echa un vistazo al video de ensamblaje en
http://reflexiona.biz/video_circ13

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ13. Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

El servo no da vueltas

Incluso con los cables de colores sigue siendo extraordinariamente fácil conectar un servomotor del revés. Este podría ser tu caso.

El servo no se mueve como esperabas

El sensor ha sido diseñado para trabajar en una única dirección. Prueba a doblarlo en sentido contrario (de forma que la cara con la superficie rallada quede en el lado convexo).

El servo se mueve solo una vez

Es posible que tengas que modificar el intervalo de valores en la llamada a la función `map()` (puedes encontrar más detalles en la sección MEJORANDO EL CIRCUITO)

:: MEJORANDO EL CIRCUITO

Calibrando el intervalo

A pesar de que el servomotor se está moviendo, es muy probable que el intervalo no sea demasiado perfecto. Para ajustar este intervalo tenemos que cambiar los valores en la función `map()`:

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Para ver todos los detalles sobre como funciona visita <http://www.arduino.cc/es/Reference/Map>

Para calibrar nuestro sensor podemos utilizar la ventana de depuración (como en el CIRC-11). Abre la ventana del monitor serie y sustituye el valor `fromLow` (por defecto a 50) con el valor que sale cuando el sensor está sin doblar. Después sustituye el valor `fromHigh` (por defecto a 300) con el valor que obtienes al doblar el sensor por completo

Aplicaciones

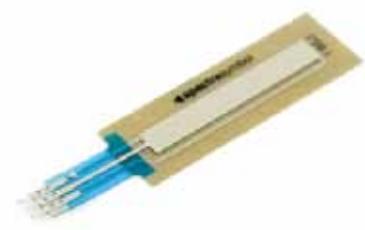
Con los sensores flexibles la verdadera diversión llega cuando los utilizas de forma ingeniosa e inesperada. A continuación, os presentamos algunas de nuestras aplicaciones favoritas:

Guante para jugar un solitario de piedra papel o tijera: <http://ardx.org/RPS>

Abrazadera electrónica para plantas: <http://ardx.org/BRACE>

:: LO QUE ESTAMOS HACIENDO

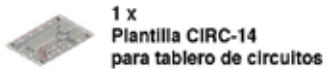
Un potenciómetro de membrana funciona como un potenciómetro convencional (el que hemos visto en el CIRC-08) pero con la diferencia de que es plano, muy fino, flexible y no dispone de mando rotatorio. Como ya hemos comentado anteriormente, al potenciómetro también se le conoce como resistencia variable. El caso de un potenciómetro de membrana la resistencia viene determinada por el lugar sobre el que se aplica una presión. Esta presión puede aplicarse con un dedo, un bolígrafo o un trozo de plástico. Pulsando en diferentes zonas de la membrana la resistencia varía de 100 a 10k Ohmios, permitiéndote calcular la posición relativa del punto de presión sobre la membrana. Puedes utilizar esta característica para seguir el movimiento de algún dispositivo sobre la membrana o como pulsaciones en un botón discreto.



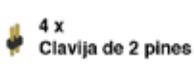
En este circuito, usaremos el potenciómetro de membrana para controlar el cambio de color de un LED RGB.

:: EL CIRCUITO

_Componentes



1 x
Plantilla CIRC-14
para tablero de circuitos



4 x
Clavija de 2 pines

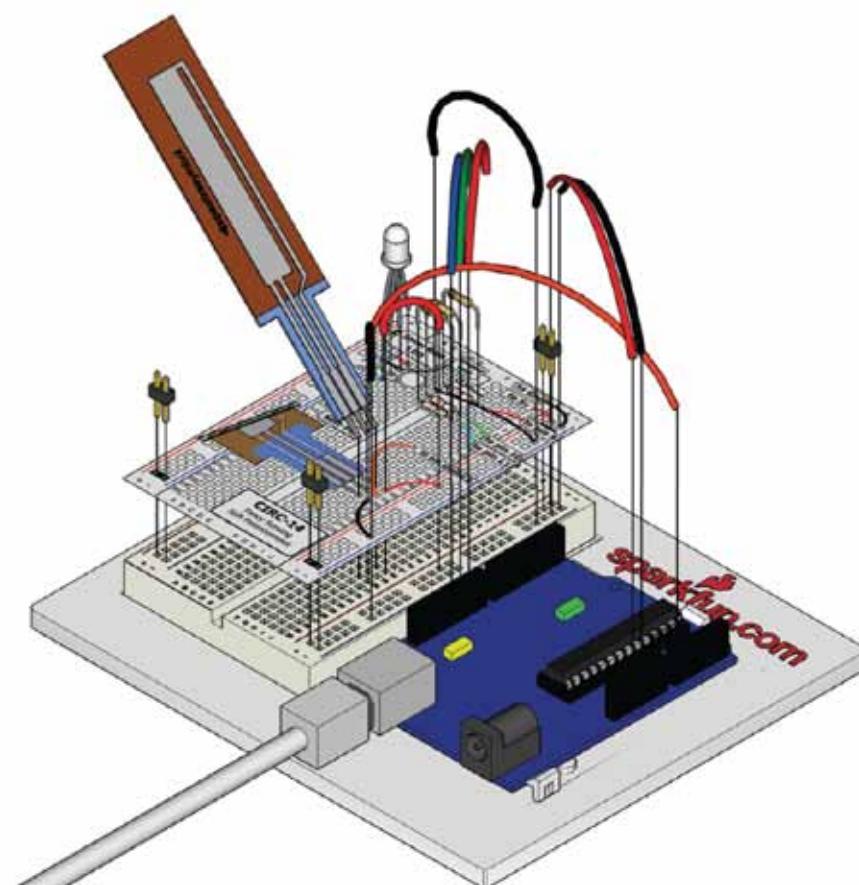
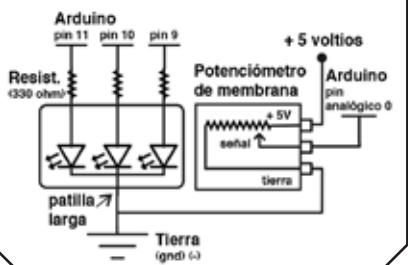


1 x
Sensor Flexible



9 x
Cable con terminales

_Esquema



_Internet

Descarga una plantilla
para el CIRC-14 en
http://reflexiona.biz/plantilla_circ14

Echa un vistazo
al video de ensamblaje en
http://reflexiona.biz/video_circ14

:: EL CÓDIGO

No es necesario que escribas todo el texto. Descarga el código desde http://reflexiona.biz/codigo_circ14.
Copia el texto y pégalo en un sketch vacío de Arduino.

:: NO FUNCIONA (3 cosas que puedes probar)

_El LED no se enciende o muestra un color incorrecto

Como los cuatro pines del LED están muy pegados los unos a los otros, es bastante común conectar uno de ellos mal.
Comprueba que cada uno de los pines está conectado donde debe de estarlo.

_Se ve todo rojo

Es muy probable que el diodo rojo del LED RGB sea mucho más brillante que los otros dos. Para conseguir que los colores estén más balanceados, conviene que utilices una resistencia mayor. También puedes ajustar esto mediante código:

```
analogWrite(RED_LED_PIN, redIntensity); => analogWrite(RED_LED_PIN, redIntensity/3);
```

_Resultados bizarros

La causa mas probable de que suceda esto, es que estés presionando el potenciómetro en más de un punto a la vez. Que suceda esto es bastante normal, y la verdad es que puede usarse para crear algunos resultados estupendos.

:: MEJORANDO EL CIRCUITO

_Modelo de color HSB (Tonalidad, Saturación, Brillo)

Nuestro LED RGB muestra un color utilizando códigos de color RGB. Sin embargo, esta no es siempre la forma más fácil de trabajar con colores. Un modelo de color mucho más intuitivo es el HSB. Si quieras obtener más detalles sobre este modelo de color visita http://es.wikipedia.org/wiki/Modelo_de_color_HSV

Para convertir de RGB a HSB todo lo que hace falta es un poco de matemáticas “ligeramente complicadas”. Para ver un programa de ejemplo visita http://reflexiona.biz/codigo_circ14b

Este es un programa basado en el código original de www.kasperkamperman.com

Ahora, cuando utilices el potenciómetro de membrana, notarás que la transición del rojo al violeta es mucho mas clara y completa.

_Botones de imitación

Tal y como funciona el potenciómetro de membrana, este puede ser usado para hacer botones personalizados. Para hacer esto tienes que definir un intervalo de valores que se correspondan con un botón discreto.

Utiliza el trozo de código de abajo y la ventana del monitor serie para determinar los valores deseados:

```
If(analogRead(0) > minValue && analogRead(0) < maxValue){  
    buttonAction()  
}
```

Ahora puedes cubrir el potenciómetro de membrana con el diseño de un botón dibujado o impreso.

NOTAS

w w w . r e f l e x i o n a . b i z / s h o p



Los contenidos de esta obra están bajo licencia de Creative Commons Reconocimiento-CompartirIgual 3.0 Unported (CC BY-SA 3.0).

Para ver una copia de esta licencia visita <http://es.creativecommons.org/licencia/>

También puedes enviar un correo electrónico con tu consulta a procomun@gmail.com