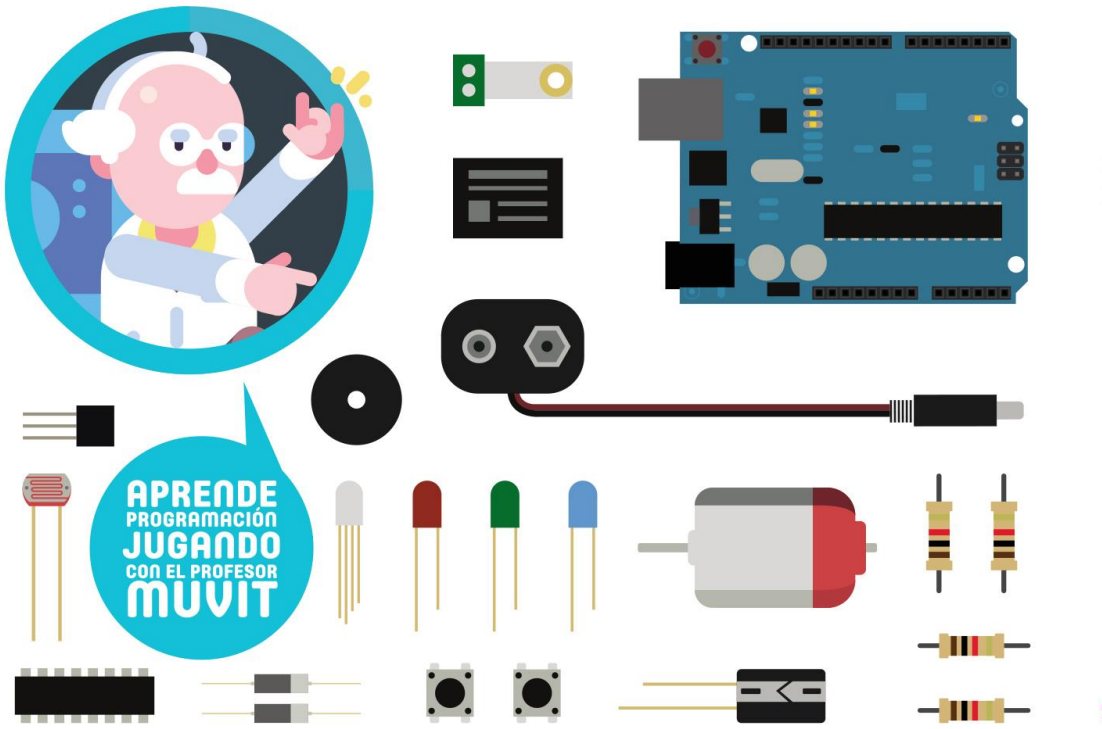
Manual

ARDX Starter Kit

Arduino para principiantes



muvit Brain

seeed studio

RoHS COMPLIANCE

Made in P.R.C.

Made for MAKERS

Introducción

Sobre este Kit

Este kit ha sido diseñado para aprender jugando. Permitirá que te familiarices con una amplia gama de Componentes Electrónicos a través de circuitos pequeños, simples y fáciles.
El objetivo es hacer funcionar cada circuito y luego darte las herramientas para que puedas descubrir el por qué.

Sobre Arduino:

Arduino es una plataforma electrónica de código abierto basada en hardware y software fácil de usar. Está pensada para que cualquiera pueda crear proyectos interactivos.

Durante años, ha sido el cerebro de cientos de proyectos, desde objetos cotidianos hasta instrumentos científicos complejos. Una comunidad mundial de creadores que comprende estudiantes, artistas, aficionados, programadores y profesionales se ha reunido en torno a esta plataforma y sus contribuciones se han añadido a un gran volumen de conocimientos accesible para todo el mundo que puede ser de ayuda tanto para novatos como para expertos.

Las placas Arduino detectan el entorno recibiendo información de varios sensores y responden al entorno mediante control de luces, motores y otros actuadores.

Sobre muvit I/O:

Muvit es una marca registrada del grupo ascendeo, líder en España y Portugal en el desarrollo y la distribución de accesorios para dispositivos móviles y objetos conectados.
Los productos muvit están diseñados en Europa y producidos en ascendeo China (Shenzhen), donde se realizan los controles de calidad así como todos los embalajes de los productos.

¿Quieres saber más? https://iot.ascendeoiberia.net

Sobre Seeed Studio:

Seeed Studio es un facilitador de hardware abierto con sede en China, que ayuda a los diseñadores a convertir sus ideas en productos. Al trabajar estrechamente con proveedores de tecnología de todas las escalas, Seeed proporciona tecnologías accesibles con calidad, velocidad y garantía.

¿Quieres saber más? https://www.seeedstudio.io

Servicio Post Venta:

Nos esforzamos por ofrecer el más alto nivel de calidad en cada cosa que producimos. Si alguna vez encuentra una instrucción ambigua, una pieza faltante, o simplemente desea hacer una pregunta, estamos a su disposición:

IOT@ascendeoiberia.com

Tabla de contenido

.: Dónde Encontrarlo Todo :.

Antes de empezar

{ASEM} Unir las piezas 02

{INST} Instalar el software 03

{PROG} Pequeña introducción a la programación 04

{ELEC} Pequeña introducción a la electrónica 06

Los circuitos

{CIRC01} ¡Empezamos! - (Iluminar un LED) 08

{CIRC02} Diversión para 8 - (LEDs múltiples) 10

{CIRC03} Gira Motor, gira - (Transistor y Motor) 12

{CIRC04} Servo - (Servos) 14

{CIRC05} 8 LEDs más - (74HC595 Registro de desplazamiento) 16

{CIRC06} Música - (Piezoeléctrico) 18

{CIRC07} Pulsar botones - (Pulsadores) 20

{CIRC08} Girando - (Potenciómetros) 22

{CIRC09} Luz - (Resistencias) 24

{CIRC10} Temperatura - (Sensor de temperatura TMP36) 26

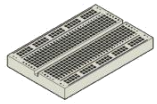
{CIRC11} Cargas más grandes - (Relés) 28

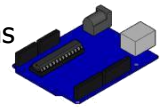
{CIRC12} Luces de colores - (LEDs RGB) 30

{CIRC13} Vibración - (Sensor) 32

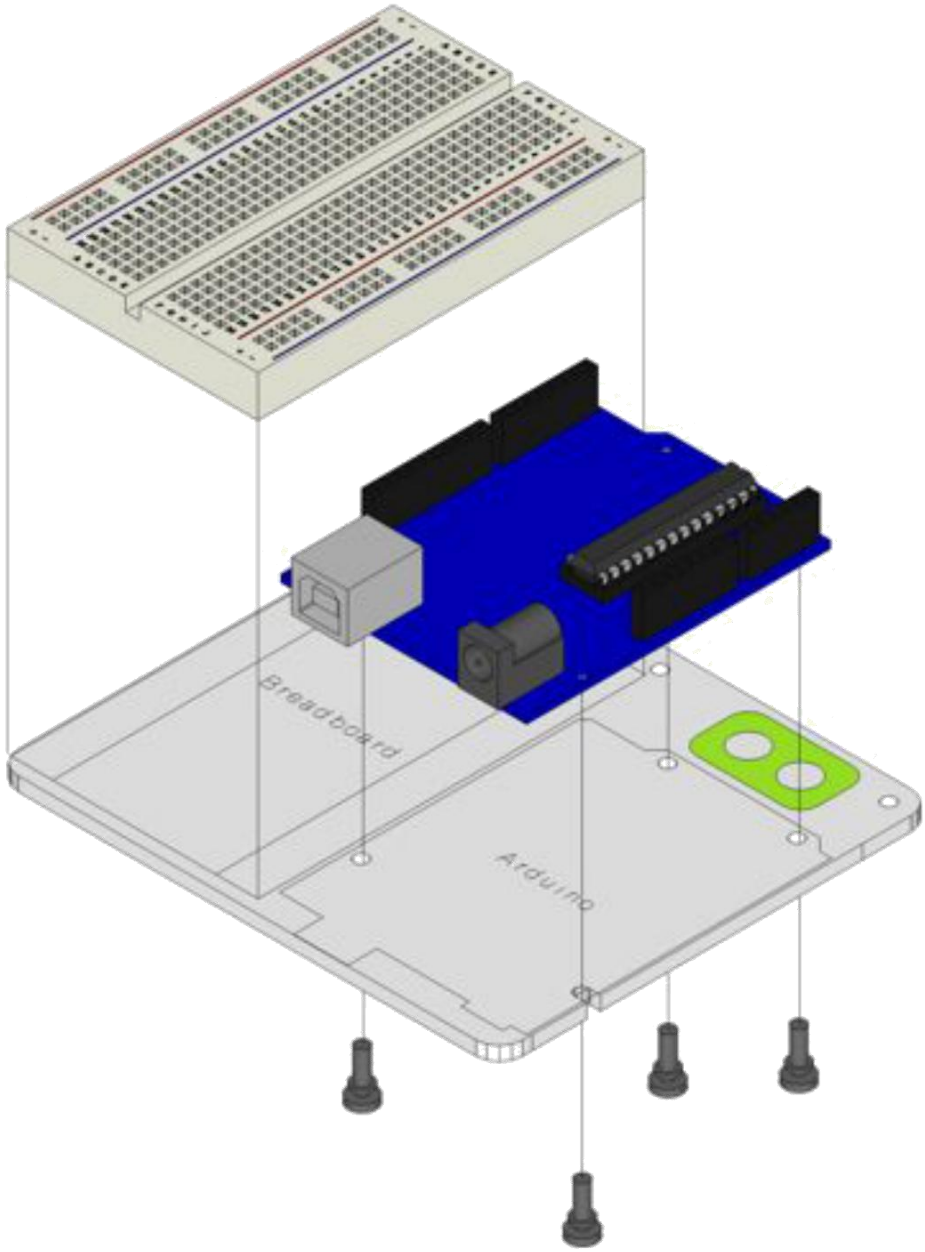
.: ¡Vamos a Montarlo! :.

Soporte Arduino
x1


Placa de pruebas
x1


Arduino
x1

Remaches de plástico 3mm
x4



.: Si quieres saber más sobre Arduino, entra aquí :.

.: Instalar el Software IDE :.

Usaremos este programa para escribir el código Arduino. Puede parecer un poco difícil pero una vez lo hayas instalado y lo uses, verás que es muy intuitivo.

Paso 1: Descarga el Software

Haz clic en
http://arduino.cc/en/Main/Software
descarga el software para tu sistema operativo

Windows® XP

Paso 2: Descomprime

Unzip
arduino-1.X-windows.zip (X - version #)
Ruta recomendada
c:\Program Files\

Paso 3: Acceso Directo

Abre
c:\program files\arduino-1.X\ (X – version #)
Haz clic con el botón derecho
Arduino.exe (enviar a>escritorio (crear acceso)

Paso 4: Conecta tu Arduino

Conecta tu Arduino:
Usa el cable USB incluido, conecta tu placa Arduino a un puerto USB libre.
Espera a que se abra un cuadro de diálogo

Paso 5: Añade nuevo Hardware

Evita buscar en internet
(haz clic en "siguiente")
(haz clic en "**Instalar desde una lista o localización específica** (Avanzado))
Elige la localización
c:\program files\arduino-00XX\drivers\
¡Ya estàs listo!

Windows Vista®
Windows 7

Paso 5: Añade nuevo Hardware

Abre el gestor de dispositivos
Inicio > Ejecutar > devmgmt.msc
Elige Arduino
Otros dispositivos > Arduino Uno (Uno)
Actualiza el Driver
Haz clic en "Actualizar Driver"
Selecciona el Driver
Haz clic en "Buscar en mi ordenador"
c:\program files\arduino-1.X\drivers\
¡Ya estás listo!

Mac OS X®

Paso 2: Abre The .dmg

Abrir (mount)
arduino-1.X-macosx.zip (X version#)

Paso 3: Copia la aplicación

Dirígete a
"Arduino" (en la sección Finder del dispositivo)
Mueve
La aplicación "Arduino" a la Carpeta "Aplicaciones"

Paso 4: Conecta tu Arduino

Conecta tu Arduino:
Usa el cable USB incluido, conecta tu placa Arduino a un puerto USB libre.
¡Ya estás listo!

.: NOTA: :.
.: ¿Tienes problemes? :.
.: ¿Necesitas más datos? ¿Usas Linux? :.
.: Haz clic aquí :.

.: Introducción a la Programación :.

PROGRAMACIÓN ARDUINO EN POCAS PALABRAS

Arduino se programa en lenguaje C. Esto es una pequeña introducción para aquellos que tienen un poco de experiencia en programación y sólo necesitan un resumen de las idiosincrasias de C y Arduino IDE. Si los conceptos son un poco liosos, no te preocupes, puedes empezar haciendo los circuitos y adquiriendo experiencia por el camino. Para una introducción más detallada, la web Arduino.cc es un gran recurso.

ESTRUCTURA

Cada programa Arduino (también llamado sketch) requiere dos funciones (también llamadas rutinas).

CONFIGURACIÓN VACÍA (){}

void setup ( ) { }
El código entre los paréntesis se ejecutará una vez cuando el programa Arduino se inicie por primera vez.

BUCLE VACÍO () {}

void loop ( ) { }
Esta función se ejecutará después de la configuración. Se repetirá una y otra vez hasta que se le quite poder.

SINTAXIS

Uno de los elementos más frustrantes de C son los requisitos de formato (lo que también lo hace muy poderoso). Si recuerdas lo siguiente, todo debería salir bien

// (Comentarios de una sola línea). Es útil para escribir notas para ti mismo para describir qué hace cada línea. Escribe dos barras al principio y todo lo que escribas a continuación será ignorado por el programa.

{ } (Llaves) Se usan para definir cuándo empieza y cuando termina un bloque de código. (También se usa en funciones y bucles).

/* */ (Comentarios de varias líneas). Si tienes mucho que decir, puedes expandir el comentario a varias líneas. Todo lo que escribas entre estos símbolos será ignorado por el programa.

; (punto y coma). Cada línea de código debe acabar con un punto y coma (omitirlo es a menudo la razón por la que un programa no compila)

VARIABLES

Un programa no es más que instrucciones para mover números de un modo inteligente. Las variables se crean para configurar el movimiento.

int (números enteros)
La unidad principal, almacena un número en bytes (16 bits). No admite decimales y almacenará un valor entre -32,768 y 32,767.

long (longitud)
Se usa cuando un número entero no es suficientemente largo. Ocupa 4 bytes (32 bits) de RAM y tiene un rango entre -2,147,483,648 y 2,147,483,647

boolean (booleano)
Una simple variable de verdadero o falso. Útil porque solo usa un bit de RAM.

float (flotar)
Usado para números con coma flotante (decimales). Usa 4 bytes (32 bits) de RAM y tiene un rango entre -3.4028235E+38 y 3.4028235E+38.

Char (caracteres)
Almacena un carácter usando código ASCII (ejemplo 'A' = 65). Usa un byte (8 bits) de RAM. Arduino maneja cadenas (strings) como conjuntos de caracteres (array).

.: Para una introducción más extensa :.
Haz clic aquí

OPERADORES MATEMÁTICOS

Son los operadores usados para manipular números. (Funcionan como matemáticas simples).

= (asignación) Hace que algo sea igual a otra cosa. (Ej. x = 10 * 2 (x es igual a 20))
% (módulo) Da el resto cuando un número se divide por otro. (Ej. 12 % 10 (da 2))
+ (Suma)
- (Resta)
***** (Multiplicación)
/ (División)

OPERADORES DE COMPARACIÓN

Son los operadores usados para comparaciones lógicas.

== (Igual a) (Ej. 12 == 10 es FALSO o 12 == 12 es VERDADERO)
! = (No igual a) (Ej. 12 ! = 10 es VERDADERO o 12 != 12 es FALSO)
< (Menor que) (Ej. 12 < 10 es FALSO o 12 < 12 es FALSO o 12 < 14 es VERDADERO)
> (Mayor que) (Ej. 12 > 10 es VERDADERO o 12 > 12 es FALSO o 12 > 14 es FALSO)

ESTRUCTURA DE CONTROL

Los programas dependen de controlar qué se ejecuta a continuación, aquí están los elementos básicos de control. (Hay muchos más en internet)

if (condición){ }
else if(condición){ }
else { }

Ejecutará el código entre los paréntesis si la condición es verdadera. Si no lo es, probará la condición 'else if'. Si también es falso ejecutará el código 'else'

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

Se utiliza cuando se quiere repetir un fragmento de código un número de veces (puede contar hacia arriba i++ o hacia abajo i— o usar cualquier variable)

DIGITAL

PinMode (pin, modo);
Se usa para especificar el modo de un pin, pin es el número de pin al que quieres dirigirte 0-19. (Analógicos 0-5 son 14-19). El modo puede ser de ENTRADA (input) o SALIDA (output).

digitalWrite (pin, valor);
Cuando se configura un pin como SALIDA, puede configurarse como HIGH (hasta +5 voltios) o LOW (hasta tierra).

int digitalRead (pin);
Una vez se ha configurado un pin como ENTRADA puedes usarlo para volver a HIGH (hasta +5 voltios) o LOW (hasta tierra).

ANALÓGICO

Arduino es una máquina digital que tiene la habilidad de operar en un mundo analógico. (A través de trucos). Así es como puedes lidiar con cosas que no son digitales.

int analogWrite(pin, value);
Algunos de los pins Arduino admiten modulación de ancho de pulso (3, 5, 6, 9, 10, 11). Esto enciende el pin y lo hace actuar rápidamente como una salida analógica. El valor es cualquier número entre 0 (0% ciclo de trabajo ~0v) y 255 (100% ciclo de trabajo ~5 voltios).

int analogRead(pin);
Cuando los pins de entrada analógica están configurados para entrada se puede leer su voltaje. Devolverán un valor entre 0 (para 0 voltios) y 1024 (para 5 voltios)

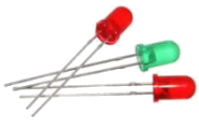
.: Introducción a la Electrónica :.

LA ELECTRÓNICA EN POCAS PALABRAS

No se necesita experiencia previa en electrónica para divertirse con este kit. Aquí hay algunos detalles sobre cada componente para que podáis identificarlos y quizás, entenderlos un poco mejor. Si te preocupa saber cómo se usa un componente o por qué no funciona, internet ofrece infinidad de consejos. También puedes contactarnos en IOT@ascendeoiberia.com

DETALLE DE LOS COMPONENTES

LED
(Diodo emisor de luz)



¿Qué hace?:
Emite luz cuando pasa una pequeña cantidad de corriente a través suyo (en una sola dirección

¿Cómo identificarlo?
Parece una bombilla pequeña

No. de conexiones:
2 (una larga que se conecta a positivo)

¿Qué hay que tener en cuenta?
Funciona en una sola dirección
Requiere una resistencia para limitar la corriente

Más detalles aquí

Diodo

¿Qué hace?:
Es el equivalente electrónico a una válvula de una vía. Permite que la corriente fluya en una dirección pero no en la otra.

¿Cómo identificarlo?
Generalmente es un cilindro con cables que se extienden hacia ambos lados.
(Y una línea descentrada que indica la polaridad)

No. de conexiones:
2

¿Qué hay que tener en cuenta?
Funciona en una sola dirección
(la corriente fluirá si el final con la línea está conectado a tierra)

Más detalles aquí

Resistencia

¿Qué hace?
Restringe la cantidad de corriente que puede fluir a través de un circuito.

¿Cómo identificarlo?:
Es un cilindro con cables que se extienden hacia ambos lados. El valor se muestra usando un sistema de código de colores.

No. de conexiones:
2

¿Qué hay que tener en cuenta?
Es fácil coger un valor incorrecto
(Fíjate en los colores antes de usarlo)

Más detalles aquí

Transistor

¿Qué hace?:
Usa una pequeña corriente para cambiar o amplificar una corriente mucho más grande.

¿Cómo identificarlo?:
Viene en muchos paquetes diferentes pero Puedes leer la referencia en el paquete.
(Coge P2N2222AG en este kit y busca info online)

No. de conexiones:
3 (Base, Colector, Emisor)

¿Qué hay que tener en cuenta?
Enchúfalo en el sentido correcto (puedes necesitar una resistencia en la base)

Más detalles aquí

Servo

¿Qué hace?:
Toma un pulso temporizado y lo convierte en una posición angular del eje de salida.

¿Cómo identificarlo?:
Una caja de plástico con tres cables saliendo de un lado y un eje con un aspas de plástico encima.

No. de conexiones:
3

¿Qué hay que tener en cuenta?
El enchufe no está polarizado así que asegúrate que lo conectas en la posición correcta.

Más detalles aquí

Motor DC

¿Qué hace?:
Gira cuando la corriente pasa a través suyo.

¿Cómo identificarlo?:
Es fácil, parece un motor.
Generalmente es un cilindro con un eje saliendo de uno de los lados.

No. de conexiones:
2

¿Qué hay que tener en cuenta?
Usar un transistor o relé adecuado para el tamaño de motor que estés usando.

Más detalles aquí

DETALLE DE LOS COMPONENTES (CONT.)

Piezoeléctrico

¿Qué hace?:
Un pulso de corriente hará que haga clic. Varios pulsos seguidos harán que emita un tono.

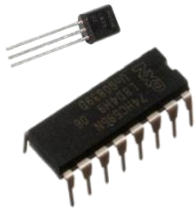
¿Cómo identificarlo?
En este kit viene en una carcasa negra pero a menudo es simplemente un disco dorado

No. de conexiones:
2

¿Qué hay que tener en cuenta?
Es difícil darle un mal uso

Más detalles aquí

Circuito Integrado



¿Qué hace?:
Agrupa cualquier conjunto de electrónica complicada en un formato más sencillo.

¿Cómo identificarlo?:
La referencia está escrita en el exterior de la pieza. (Generalmente requiere mucha luz o una lupa para leerlo).

No. de conexiones:
2 - 100s (en este kit hay uno con 3 (TMP36) y uno con 16 (74HC595)

¿Qué hay que tener en cuenta?
Orientación adecuada. (Busca las marcas que indican el pin 1)

Más detalles aquí

Pulsador

¿Qué hace?:
Completa un circuito cuando se pulsa.

¿Cómo identificarlo?:
Un pequeño cuadrado con conexiones en la base y un botón encima.

No. de conexiones:
4

¿Qué hay que tener en cuenta?:
Son casi cuadrados por lo que se pueden insertar en ángulo de 90 grados.

Más detalles aquí

Potenciómetro

¿Qué hace?:
Produce una resistencia variable en función de la posición angular del eje.

¿Cómo identificarlo?:
Pueden venir en muchos formatos diferentes. Busca la escala numérica para identificarlo.

No. de conexiones:
3

¿Qué hay que tener en cuenta?:
Comprar por error una escala logarítmica.

Más detalles aquí

Foto resistencia

¿Qué hace?:
Produce una resistencia variable en función de la luz incidente.

¿Cómo identificarlo?
Generalmente es un disco pequeño con una parte superior clara y una línea curva debajo.

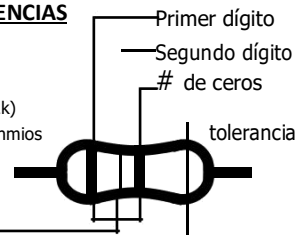
No. de conexiones:
2

¿Qué hay que tener en cuenta?:
Recuerde que tiene que estar en un divisor de voltaje antes de dar una entrada válida.

Más detalles aquí

COLORES DE LAS RESISTENCIAS

Ejemplos:
verde-azul-marrón - 560 ohmios
rojo-rojo-rojo - 2 200 ohmios (2.2k)
marrón-negro-naranja - 10 000 ohmios (10k)



0 - Negro	5 - Verde	20%	- nada
1 - Marrón	6 - Azul	10%	- plata
2 - Rojo	7 - Lila	5%	- oro
3 - Naranja	8 - Gris		
4 - Amarillo	9 - Blanco		

AJUSTE DE CONEXIONES:

Algunos componentes de este kit vienen con los cables de conexión muy largos. Para hacerlos más compatibles con la placa de pruebas es necesario hacer un par de cambios.

LEDs:
Acorta las patas para que la larga mida ~10mm (3/8") y la corta mida ~7mm (9/32").

Resistencias:
Dobla las patas para que queden a 90 grados del cilindro. Recorta el sobrante para que midan ~6mm (1/4").

Otros componentes:
Otros componentes pueden necesitar recortes. Hazlo a tu gusto.

CIRC-01

.: ¡Empezamos! :.
.: (Iluminar un LED) :.

¿QUÉ ESTAMOS HACIENDO?:

Los LED (diodos emisores de luz) se usan en todo tipo de cosas inteligentes y por eso los hemos incluido en este kit. Empezaremos con algo muy simple, encendiendo y apagando uno repetidamente, produciendo un efecto intermitente. Para empezar coge las piezas que detallamos a continuación, coloca la hoja plantilla en tu placa de pruebas y conéctalo todo. Una vez hayas ensamblado el circuito, tendrás que cargar el programa. Para hacerlo, conecta la placa Arduino en tu puerto USB. Luego selecciona el puerto correcto en **Tools (Herramientas) > Serial Port (Puerto Serie) > (el puerto donde ha conectado la placa Arduino)**. Después carga el programa yendo a **File (Archivo) > Upload to (cargar a) I/O Board (ctrl+U)**. Finalmente, disfruta de la gloria y las posibilidades que ofrece el control de luces.

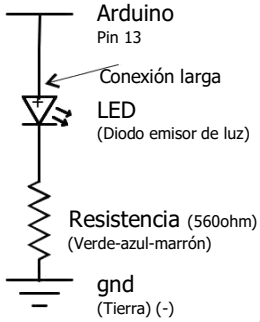
Si tienes problemas para cargar el software, puedes encontrar la solución aquí

EL CIRCUITO:

Piezas:

- Hoja plantilla CIRC-01 x 1
- Conectores de dos pines x 4
- LED 10mm x1
- Cables
- Resistencia 560 Ohm Verde-Azul-Marrón x1

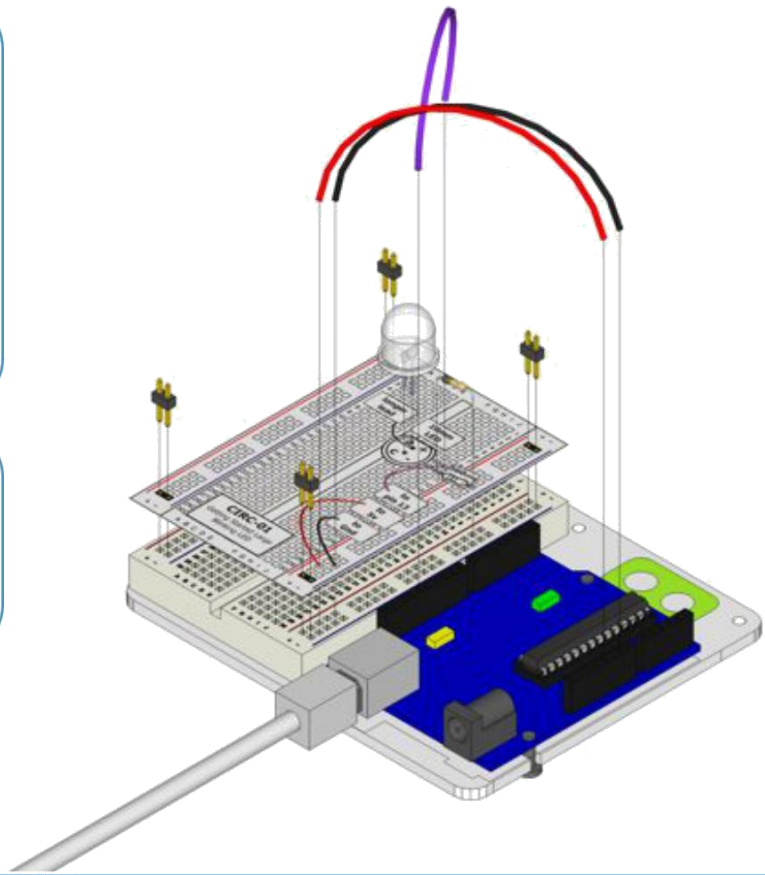
Esquema



Internet

Descarga la hoja plantilla Para la placa de pruebas aquí

Video explicativo aquí



CÓDIGO (No hace falta copiarlo, solo haz clic)

File (archivo) > Examples (ejemplos) > 1.Basic > Blink

(Ejemplo de la página arduino.cc, visítala para encontrar más ideas)

```
/* Blink (parpadear)
 *  enciende un LED durante un segundo, luego lo apaga durante un segundo
 *  repetidamente
 *  creado el 1 de Junio de 2005 por David Cuartielles
 *  https://www.arduino.cc/en/Tutorial/Blink
 *  basado en un original de H. Barragan para la placa Wiring i/o
 */

int ledPin =          13;         // LED conectado al pin digital 13


// El modo setup() se ejecuta una vez al inicio
void setup()          {                         // inicializa el pin digital como una salida:
   pinMode(ledPin, OUTPUT);              }

//    el método loop() se repite una y otra vez,
//    mientras Arduino tenga energía
void loop()                                     {
   digitalWrite(ledPin, HIGH);                  // enciende el LED
   delay(1000);                                 // espera un segundo
   digitalWrite(ledPin, LOW);                   // apaga el LED
   delay(1000);                                 // espera un segundo
}
```

¿NO FUNCIONA? (Intenta estas tres cosas)

¿El LED no se enciende?

Los LEDs funcionan en una sola dirección. Intenta sacarlo y girarlo 180 grados. (No te preocupes. Instalarlo del revés no representa un daño permanente).

El programa no carga

Ocurre a veces, lo más probable es que hayas confundido el puerto serie.

Puedes cambiarlo en:

tools>serial port>

¿Sigue sin funcionar?

Un circuito roto no es divertido.

Mándanos un e-mail y nos pondremos en contacto

IOT@ascendeoiberia.com

¿CÓMO MEJORARLO?

Cambiando el pin:

El LED está conectado al pin 13 pero podemos usar cualquier pin de la placa Arduino. Para cambiarlo, quita el cable conectado al pin 13 y muévelo a otro pin de tu elección (de 0-13) (también puedes usar los analógicos 0-5, analógico 0 es 14...) Entonces en el código deberás cambiar la línea:

int ledPin = 13; -> int ledPin = newpin;

Luego carga el programa: (ctrl-u)

Cambia el tiempo de parpadeo:

¿No te gusta el intervalo de un segundo?

En el código, cambia las líneas:

```
digitalWrite(ledPin, HIGH);
delay(time on);       //(segundos * 1000)
digitalWrite(ledPin, LOW);
delay(time off); //(segundos * 1000)
```

Control de brillo:

Junto con el control digital de Arduino (encendido/apagado) puedes controlar algunos pins en analógico (brillo). (Más detalles en otros circuitos). Si quieres probarlo:

Cambia el LED al pin 9: (cambia también el cable)

ledPin = 13; -> int ledPin = 9;

Reemplaza el código entre { } de loop() con esto:

analogWrite(ledPin, nuevo número);

(Nuevo número) = cualquiera entre 0 y 255.
0 = apagado, 255 = encendido. En medio = brillos distintos.

Desvanecimiento:

Usaremos otro programa de ejemplo incluido. Para abrirlo dirígete a **File > Examples > 3.Analog > Fading**

Cárgalo en tu placa y observa como el LED se desvanece y se vuelve a encender.

MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?:

IOT@ascendeoiberia.com

.:Diversión para 8:.
.:LEDs Múltiples:.

¿QUÉ ESTAMOS HACIENDO?

Hemos hecho que un LED brillara, ahora es hora de ir a por más. Vamos a conectar ocho. También tendremos la posibilidad de experimentar un poco con Arduino creando varias secuencias de luces. Este circuito es también un buen experimento de puesta a punto con la escritura de programas propios y entender un poco más cómo funciona Arduino.

Along with controlling the LEDs we start looking into a few simple programming methods to keep your programs small.

Para ( ) bucles – se usa cuando quieres repetir parte del código varias veces.

Conjuntos de caracteres (arrays) [ ] – Se usan para manejar las variables con más facilidad. (Es un grupo de variables)

EL CIRCUITO:

Piezas:

CIRC-02

Hoja Plantilla x 1

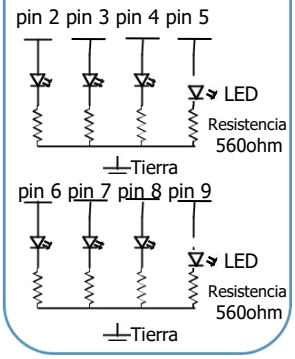
Conectores de dos pines x 4

LEDs verdes 5mm x8

Cable

Resistencia 560 Ohm
Verde-azul-marrón x 8

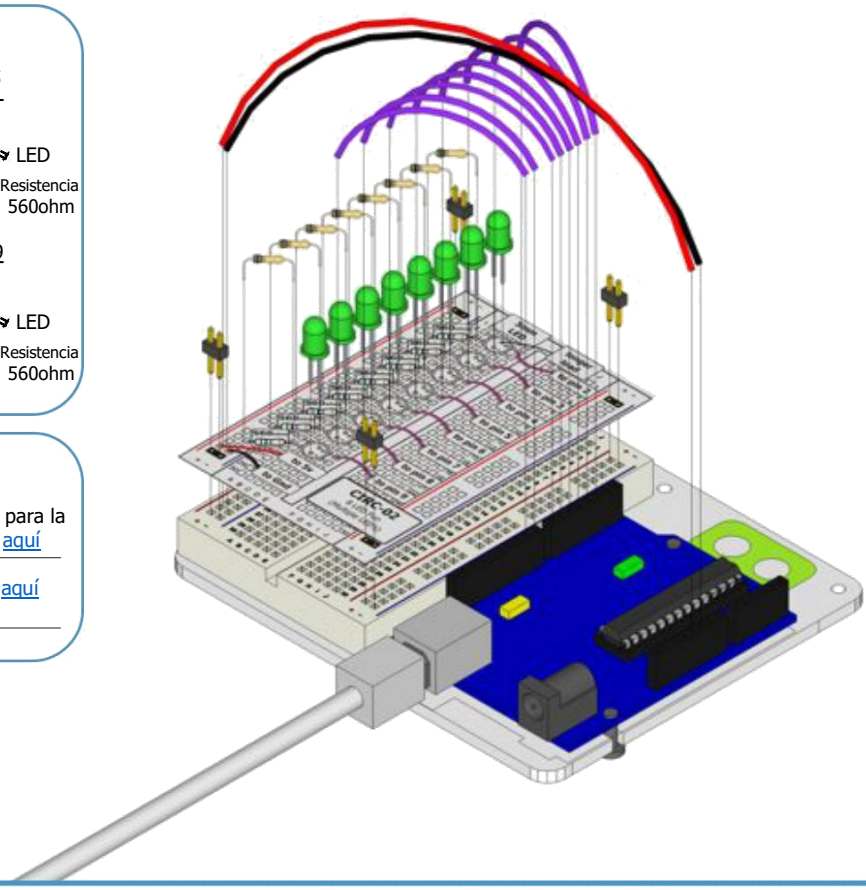
Esquema



Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí



CÓDIGO (No es necesario escribir todo, solo haz clic)

Descarga el código aquí

(Luego copia el texto y pégalo en una hoja Arduino)

```
//Variables de pin LED
int ledPins[] = {2,3,4,5,6,7,8,9};
                    //Un array para englobar el pin
                    //al que está conectado cada LED
                    //p.e. LED #0 está conectado a pin 2

void setup()
{
   for(int i = 0; i < 8; i++){
        //esto es un bucle y se repetirá ocho veces
      pinMode(ledPins[i],OUTPUT);
        //usamos esto para asignar pins LED a salida
   }
}

void loop()                    // se repite una y otra vez
{
   oneAfterAnotherNoLoop();
         //esto encenderá cada LED uno a uno
         //y después los apagará
   //oneAfterAnotherLoop();
        //esto hace lo mismo que oneAfterAnotherNoLoop
        //pero con mucha menos escritura
   //oneOnAtATime();
   //inAndOut();
}

/*
 *  oneAfterAnotherNoLoop() – Esto encenderá uno y luego
 *  esperará el tiempo de intervalo, luego encenderá el siguiente
 * LED y luego los apagará.

void oneAfterAnotherNoLoop(){
   int delayTime = 100;
        //el tiempo de pausa (en milisegundos)
        //entre los LEDs
   digitalWrite(ledPins[0], HIGH);                    //Enciende el LED #0
                                                      //(conectado a pin 2)
   delay(delayTime);            //espera el intervalo de tiempo (milisegundos)
   ...
   ...
   digitalWrite(ledPins[7], HIGH);                    //Enciende el LED #7
                                                      //(conectado a pin 9)
   delay(delayTime);            //espera el intervalo de tiempo (milisegundos)
//Apaga cada LED
   digitalWrite(ledPins[7], LOW);                     //Apaga el LED #7
   delay(delayTime);            //espera el intervalo de tiempo (milisegundos)
   ...

     -----Más código en la versión descargable------
```

¿NO FUNCIONA? (intenta estas tres cosas)

Algunos LEDs no se encienden

Es fácil colocar los LEDs del revés. Comprueba los LEDs que no funcionan y asegúrate que están bien colocados

Funcionan fuera de secuencia

Con ocho cables es fácil cruzar un par de ellos. Comprueba que el primer LED está conectado en el pin 2 y cada pin de ahí en adelante.

Empezar de nuevo

Es fácil colocar un cable mal accidentalmente sin darte cuenta. Desmontarlo todo y empezar de nuevo suele ser la manera más fácil de solucionar el problema.

¿CÓMO MEJORARLO?

Cambiando a bucles:

En la función bucle ( ) hay cuatro líneas. Las tres últimas empiezan con '//'. Significa que la línea es tratada como un comentario (no se ejecuta). Para cambiar el programa e introducir bucles cambia el código del bucle vacío ( ) a:

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

Carga el programa y verás que nada ha cambiado. Echa un vistazo a ambas funciones. Las dos hacen lo mismo pero desde distintas perspectivas (Pista: la segunda usa un bucle).

Animaciones extra:

¿Cansado de esta animación? Entonces intenta estas otras. Borra la marca de comentario y carga el programa a tu placa para disfrutar de las nuevas animaciones de luces. (Borra las barras de delante de las líneas 3 y 4)

Prueba tus propias animaciones:

Utiliza el código incluido y empieza a cambiar cosas. Lo importante es encender un LED digitalWrite(pinNumber, HIGH); y luego apagarlo digitalWrite(pinNumber, LOW); .

Anímate. Cambies lo que cambies, no vas a romper nada.

MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?:

IOT@ascendeoiberia.com

CIRC-03

.:Gira motor, gira:.
.:Transistor y Motor:.

¿QUÉ ESTAMOS HACIENDO?:

Los pins Arduino son ideales para controlar directamente pequeñas piezas eléctricas como los LEDs. No obstante, cuando trabajamos con piezas más grandes (como un micro motor o una lavadora), se necesita un transistor externo. Un transistor es increíblemente útil. Maneja gran cantidad de corriente usando una corriente mucho más pequeña. Un transistor tiene 3 pins. Para un transistor de tipo negativo (NPN), conectas tu carga al colector y el emisor a tierra. Entonces, cuando una corriente pequeña fluye de la base al emisor, la corriente fluirá a través del transistor y el motor girará. (Esto pasa cuando configuramos nuestro pin Arduino como HIGH). Hay literalmente miles de tipos distintos de transistores, permitiendo una solución para cada situación particular. Hemos elegido un P2N2222AG que es un transistor bastante común. Los factores importantes en nuestro caso son un voltaje máximo (40v) y su corriente máxima (600 miliamperios). Ambos son suficientemente altos para nuestro micro motor. (más detalles en nuestra ficha técnica aquí).

(El diodo 1N4001 está actuando como un diodo flyback. Para saber más entra aquí)

EL CIRCUITO:

Piezas:

CIRC-03 Hoja Plantilla x1

Micro Motor x 1

Conectores de dos pines x4

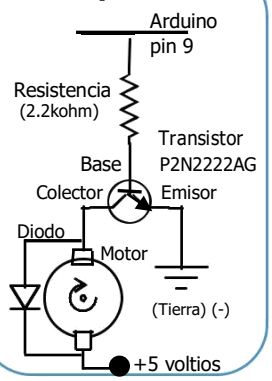
Diodo (1N4001) x 1

Transistor P2N2222AG (TO92) x1

Resistencia 2.2k Ohm Rojo-Rojo-Rojo x 1

Cable

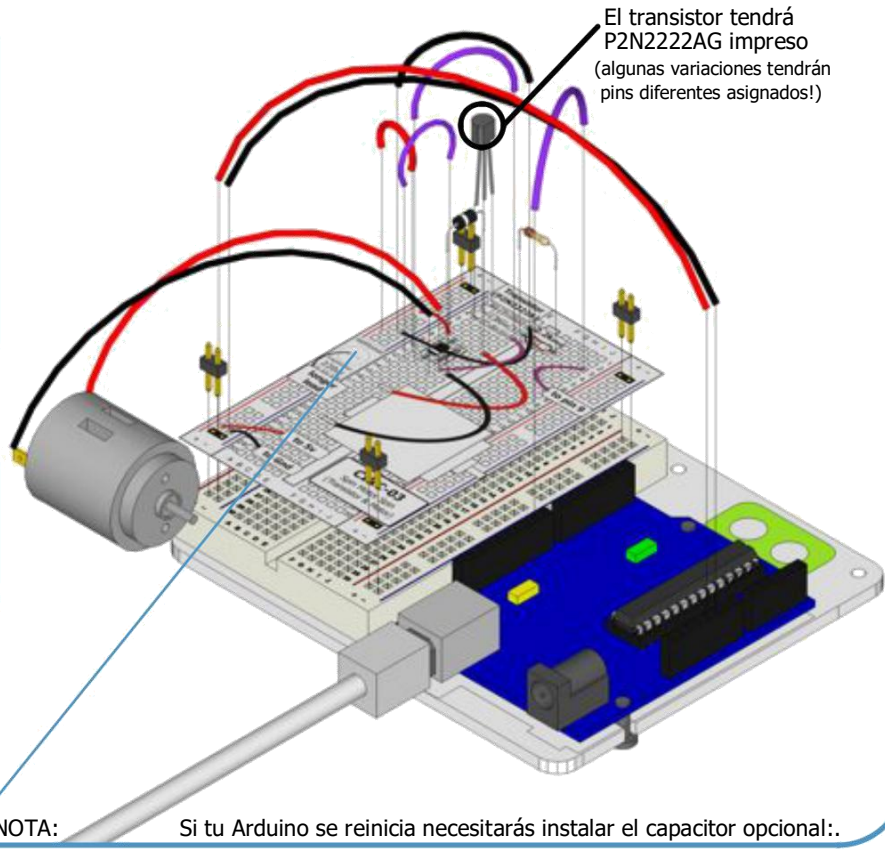
Esquema



Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí



.:NOTA: Si tu Arduino se reinicia necesitarás instalar el capacitor opcional:.

CÓDIGO (No es necesario escribir todo. Sólo haz clic)

Descarga el código aquí
(Luego copia el texto y pégalo en una hoja Arduino)

```
int motorPin = 9; //pin al que el motor está conectado

void setup() //Se repite una vez
{
  pinMode(motorPin, OUTPUT);
}

void loop()                    // Se repite una y otra vez
{
  motorOnThenOff();
  //motorOnThenOffWithSpeed();
  //motorAcceleration();
}

/*
  *   motorOnThenOff() – Enciende el motor y luego lo apaga
  *   (date cuenta de que este código es idéntico al que usamos
      para iluminar un LED)
  */
void motorOnThenOff(){
    int onTime = 2500; // encendido, intervalo de tiempo, apagado =
    1000; //off time digitalWrite(motorPin, HIGH);
                        //    Enciende el motor
    delay(onTime); // tiempo que espera para encender (milisegundos)
     digitalWrite(motorPin, LOW);
                        //    Apaga el motor
    delay(offTime);// tiempo que espera para apagar (milisegundos)
}
```

```
void motorOnThenOffWithSpeed(){
    int onSpeed = 200;// un número entre
                                    //0 (parado) and 255 (máxima velocidad)
    int onTime = 2500;
    int offSpeed = 50;// un número entre
                                //0 (parado) and 255 (máxima velocidad)
    int offTime = 1000;
    analogWrite(motorPin, onSpeed);
          // Enciende el motor
    delay(onTime);                // tiempo que espera para encender (milisegundos)
    analogWrite(motorPin, offSpeed);
          // turns the motor Off
    delay(offTime);               // tiempo que espera para apagar (milisegundos)
}

void motorAcceleration(){
    int delayTime = 50; //tiempo entre cada paso de velocidad(int i = 0; i
    < 256; i++){
            //pasa a través de cada velocidad de 0 a 255
     analogWrite(motorPin, i); //establece el nuevo retraso de velocidad
     (delayTime);// tiempo de espera (milisegundos)
    }
    for(int i = 255; i >= 0; i--){
            //pasa a través de cada velocidad de 255 a 0
       analogWrite(motorPin, i); //establece el nuevo retraso de velocidad
       delay(delayTime);//tiempo de espera (milisegundos)
    }
}
```

¿NO FUNCIONA? (Intenta estas tres cosas)

¿El motor no gira?

Si has comprado tu propio transistor, comprueba en la ficha técnica que el pin de salida es compatible con un P2N2222A
(Muchos son invertidos).

¿No ha habido suerte?

Si has comprado tu propio motor, comprueba que funciona con 5 voltios y que no arrastra mucha energía.

¿Sigue sin funcionar?

En ocasiones la placa Arduino se desconecta del ordenador. Intenta desconectarla y volver a conectarla a tu puerto USB.

¿CÓMO MEJORARLO?

Controlando la velocidad:

Antes hemos jugado con la habilidad de Arduino para controlar el brillo de un LED, ahora usaremos la misma característica para controlar la velocidad del motor. Arduino lo consigue usando algo llamado Modulación por Ancho de Pulso (PWM). Está basado en la capacidad de Arduino de trabajar muy, muy rápido. En lugar de controlar directamente el voltaje que llega del pin, Arduino encenderá y apagará el pin muy rápido. En el mundo de los ordenadores va de 0 a 5 voltios muchas veces por segundo, pero en el mundo humano lo vemos como un voltaje. Por ejemplo, si el Arduino modula por ancho de pulso al 50% veremos la luz atenuada un 50% porque nuestros ojos no son suficientemente rápidos para ver cómo se enciende y se apaga. Lo mismo ocurre con los transistores. ¿No te lo crees? ¡Pruébalo!

Cambia la sección del bucle () por esto:

```
//    motorOnThenOff();
      motorOnThenOffWithSpeed();
//    motorAcceleration();
```

Luego carga el programa. Puedes cambiar las velocidades cambiando las variables onSpeed y offSpeed.

Acelerar y decelerar:

¿Por qué parar en dos velocidades?, ¿por qué no acelerar y decelerar el motor? Para conseguirlo, simplemente cambia el código del bucle ( ) con lo siguiente:

```
//    motorOnThenOff();
//    motorOnThenOffWithSpeed();
      motorAcceleration();
```

Después carga el programa y observa cómo tu motor acelera lentamente hasta velocidad máxima y luego se ralentiza otra vez. Si quieres cambiar la velocidad de aceleración cambia la variable delayTime (más grande significa mayor tiempo de aceleración).

MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?
IOT@ascendeoiberia.com

.:Servo:.
.:Servos:.

¿QUÉ ESTAMOS HACIENDO?

Girar un motor es muy divertido pero cuando se trata de proyectos donde se requiere control de movimiento nos solemos quedar con ganes de más ¿La respuesta? Servos. Se producen en masa, se encuentra fácilmente y cuestan desde dos euros a cientos. Dentro hay una pequeña caja de engranajes (para hacer el movimiento más potente) y piezas electrónicas (para que sea más fácil de controlar). Un servo estándar es posicionable desde 0 a 180 grados. El posicionamiento se controla a través de un pulso temporizado, entre 1.25 milisegundos (0 grados) y 1.75 milisegundos (180 grados) (1.5 milisegundos para 90 grados). El tiempo varía según el fabricante. Si el pulso se envía cada 25-50 milisegundos, el servo funcionará con suavidad. Una de las grandes características de Arduino es que dispone de una librería de software que permite controlar dos servos (conectados al pin 9 ó 10), en una única línea de código.

EL CIRCUITO

Piezas:

CIRC-04
Hoja plantilla x 1

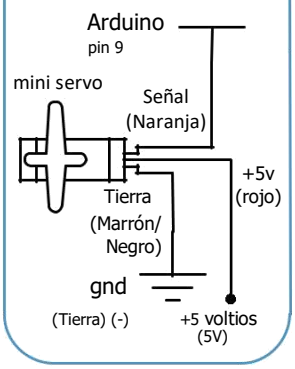
Conectores de dos pines x4

Conector de tres pines x1

Cable

Mini Servo x 1

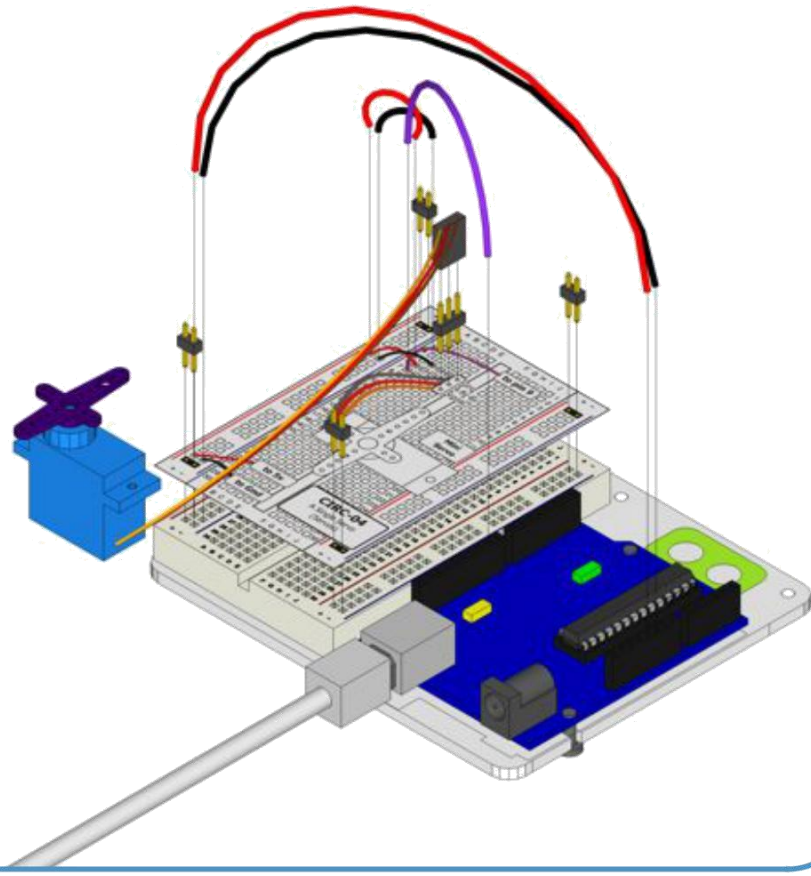
Esquema



Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí



CÓDIGO (No es necesario escribir todo, solo haz clic)

Archivo > Ejemplos > Servo > Oscilar (Sweep)

(Ejemplo de la página arduino.cc ¡Echa un vistazo para más ideas geniales!)

```
//    Sweep
//    Por BARRAGAN <http://barraganstudio.com>

#include <Servo.h>
Servo myservo;        // Crea un objeto servo para controlar el servo
int pos = 0;          // Variable para almacenar la posición del servo

void setup() {
myservo.attach(9);              // Conecta el servo en el pin 9 al objeto servo
}

 void loop() {
    for(pos = 0; pos < 180; pos += 1)              // va desde 0 grados a 180 grados
    {                                              // en pasos de 1 grado
          myservo.write(pos);                      // Le dice al servo que vaya a la posición en la variable 'pos'
       delay(15);                                  // Espera 15 ms para que el servo alcance la posición
    }
    for(pos = 180; pos>=1; pos-=1)                 // va desde 180 grados a 0 grados
    {
          myservo.write(pos);                      // Le dice al servo que vaya a la posición en la variable 'pos'
       delay(15);                                  // Espera 15 ms para que el servo alcance la posición
    }
}
```

¿NO FUNCIONA? (Intenta estas tres cosas)

¿El servo no gira?

Incluso con cables de colores sigue siendo muy fácil conectar un servo del revés. Este podría ser el caso.

¿Sigue sin funcionar?

Un error que hemos cometido una o dos veces es simplemente olvidar conectar la corriente. (Cables rojo y marrón) a +5 voltios y tierra.

Ajustes y Arranques

Si el servo empieza a moverse y luego se para, y hay una luz que parpadea en la placa Arduino, la fuente de alimentación que estás usando no está a la altura del desafío. El USB debería resolver este problema.

¿CÓMO MEJORARLO?

Control de potenciómetro:

Todavía tenemos que experimentar con las entradas, pero si quieres adelantarte, hay un programa de ejemplo en **Archivo > Servo > Pomo (Knob)**. Usa un potenciómetro (CIRC08) para controlar el servo. Puedes encontrar instrucciones en línea aquí

Auto sincronización:

A pesar de que es fácil controlar un servo usando la librería de Arduino incluida, a veces es divertido pensar cómo programar algo por ti mismo. Inténtalo. Estamos controlando el pulso directamente así que puedes utilizar este método para controlar servos en cualquiera de los 20 pins disponibles de Arduino. (Necesitarás optimizar este código antes de hacerlo)

```
   int servoPin = 9;

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

void loop() {
   int pulseTime = 2100; //(el número de microsegundos
                         //de pausa (1500  90 grados
                         //    900  0 grados 2100  180 grados)
   digitalWrite(servoPin, HIGH); delayMicroseconds(pulseTime);
   digitalWrite(servoPin, LOW);

   delay(25);
}
```

Grandes ideas:

Los servos se pueden utilizar para hacer todo tipo de cosas geniales. Aquí hay algunas de nuestras favoritas:

Contador de visitas de Navidad
http://ardx.org/XMAS

Brazo robótico de código abierto (usa un controlador servo y Arduino)
http://ardx.org/RARM

Caminador servo
http://ardx.org/SEWA

MÁS, MÁS, MÁS:

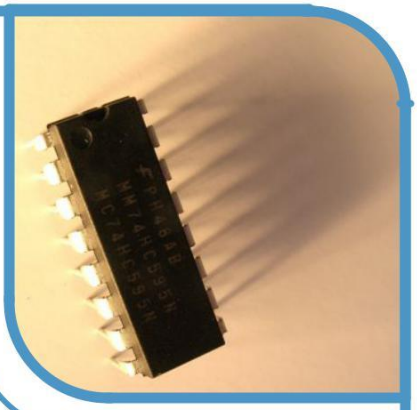
¿Necesitas más detalles o tienes alguna pregunta?:

IOT@ascendeoiberia.com

CIRC-05

.:8 LEDs más:.

.:Registro de desplazamiento 74HC595:.



¿QUÉ ESTAMOS HACIENDO?

Es hora de empezar a jugar con chips, o circuitos integrados (CI) como les gusta ser llamados. El aspecto exterior de un chip puede ser muy engañoso. Por ejemplo, el chip de la placa Arduino (un micro controlador) y el que utilizaremos en este circuito (un registro de desplazamiento o shift register) parecen muy similares pero en realidad son muy diferentes. El precio del ATMega chip en la placa Arduino cuesta unos pocos euros, mientras que el 74HC595 cuesta unos pocos céntimos. Es un buen chip introductorio, y una vez que te sientas cómodo experimentando con él y su ficha técnica (disponible aquí) el mundo de los chips estará a tus pies. El registro de desplazamiento (también llamado convertidor de serie a paralelo), te dará 8 salidas adicionales (para controlar LEDs y similares) usando tan sólo tres pins Arduino. También se pueden unir entre sí para darte un número casi ilimitado de salidas usando los mismos cuatro pins. Para usarlo "registra" los datos y luego bloquéalo (ciérralo). Para esto, tienes que configurar los datos como HIGH o LOW, pulsar el reloj (clock), luego ajustar el pin de datos nuevamente y pulsar el reloj (clock) repitiendo hasta que se hayan configurado 8 bits de datos. Por último, pulsa el cierre (latch) y los 8 bits serán transferidos a los pins del registro de desplazamiento 74HC595. Suena complicado pero es muy simple una vez te pones a ello. Tienes más información aquí y un video tutorial aquí.

EL CIRCUITO

Piezas:

CIRC-05
Hoja plantilla x 1

Conectores de dos pines x4

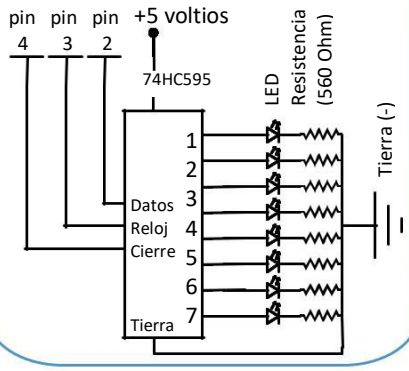
Registro de desplazamiento 74HC595 x1

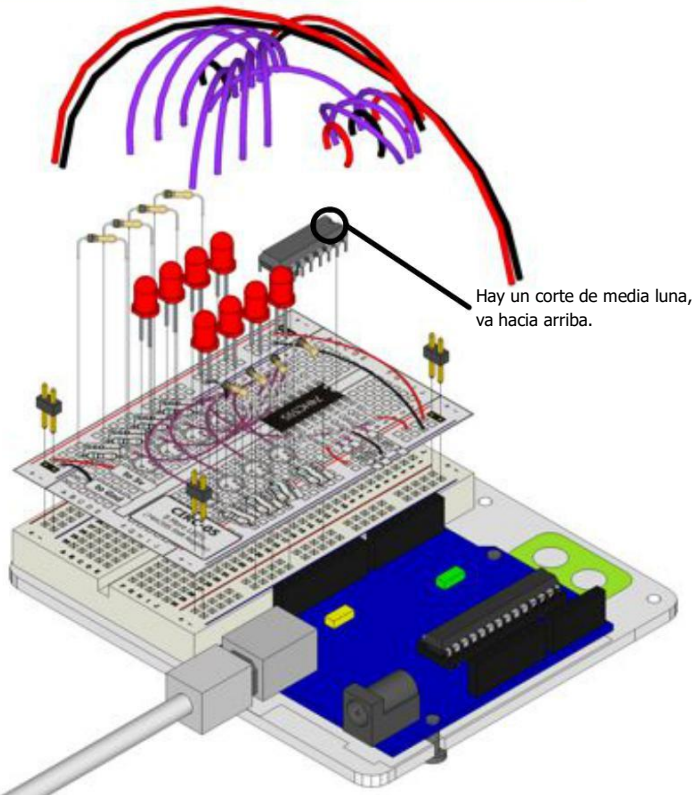
Cable

LED rojo x 8

Resistencia 560 Ohm
Verde-Azul-Marrón x 8

Esquema





Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí

CÓDIGO (No es necesario escribir todo, solo haz clic)

Descarga el código aquí
(Luego copia el texto y pégalo en una hoja Arduino)

```
//Definición de los pines
//El 74HC595 usa un enlace de comunicación
//en serie que tiene tres pins
int data = 2;
int clock = 3;
int latch = 4;

void setup() //Se ejecuta una vez {

    pinMode(data, OUTPUT);
    pinMode(clock, OUTPUT);
    pinMode(latch, OUTPUT); }

void loop() // se ejecuta una y otra vez {

    int delayTime = 100;
        //retraso entre las actualizaciones LED
    for(int i = 0; i < 256; i++){
      updateLEDs(i);
      delay(delayTime);              }
}
/*
  *   updateLEDs() – Envía los estados LED configurados
  *   en valor a la secuencia 74HC595
  */
void updateLEDs(int value){
          digitalWrite(latch, LOW);

//Marca el cierre (latch) del chip como LOW
    shiftOut(data, clock, MSBFIRST, value);
              //Desplaza 8 bits al registro de desplazamiento


digitalWrite(latch, HIGH);
          //Marca el cierre (latch) como HIGH mostrando los datos
}
```

---------- Más código en internet ----------

¿NO FUNCIONA? (Intenta estas tres cosas)

El LED de corriente del Arduino se apaga

Esto nos ha ocurrido un par de veces, pasa cuando el chip se inserta del revés. Si lo arreglas rápido no se romperá nada.

No acaba de funcionar

Sentimos sonar como un disco rayado, pero probablemente se trate de un cable cruzado.

¿Estás frustrado?

Envíanos un e-mail. Este circuito es simple y complicado a la vez. Queremos saber qué problemas has tenido para corregirlos en futuras ediciones.
IOT@ascendeoiberia.com

¿CÓMO MEJORARLO?

Haciéndolo más difícil

Arduino hace que las acciones más complejas sean fáciles, el desplazamiento de datos es uno de esos casos. No obstante, una de las mejores características de Arduino es que puedes hacer las cosas tan fáciles o difíciles como quieras. Vamos a ver un ejemplo. Cambia la línea de tu bucle ( ) por esto:

updateLEDs(i) -> updateLEDsLong(i);

Carga el programa y verás que nada ha cambiado. Si te fijas en el código, verás que nos estamos comunicando con el chip con un bit cada vez. Puedes encontrar más información aquí

Controlando LEDs individuales:

Es hora de empezar a controlar los LEDs de un modo similar al que probamos en el CIRC02 ya que el estado de los 8 LEDs está almacenado en un byte (un valor de 8 bit). Para más detalles sobre cómo funciona haz clic aquí. Arduino es muy bueno manipulando bits y hay un conjunto entero de operadores que nos pueden ayudar. Puedes encontrar más detalles sobre las matemáticas de bits aquí.

Nuestra implementación.

Sustituye el código del bucle ( ) con:

```
int delayTime = 100; //el número de milisegundos
                     //de retraso
                     //entre actualizaciones de LED
  for(int i = 0; i < 8; i++){
     changeLED(i,ON);
     delay(delayTime);
  }
  for(int i = 0; i < 8; i++){
     changeLED(i,OFF);
     delay(delayTime);
  }
```

Cargando esto conseguirás que los LEDs se enciendan uno detrás de otro y se apaguen del mismo modo. Revisa el código y la Wikipedia para ver cómo funciona, o envíanos un mail si tienes dudas.

Más animaciones:

Ahora las cosas se ponen más interesantes. Si te fijas en el código de CIRC02 (Diversión para 8) verás que cambiamos los LEDs usando digitalWrite(led, state). Este es el mismo formato que la rutina que hemos escrito changeLED(led, state). Puedes usar las animaciones que has escrito para CIRC02 copiando el código dentro de éste y cambiando todos los digitalWrite() a changeLED(). (Tendrás que cambiar algunas otras cosas, pero revisa los errores de compilación y se arreglará solo).

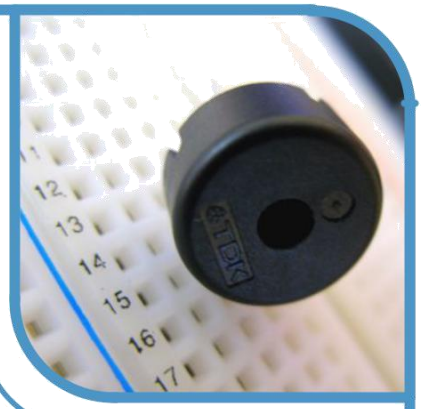
MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?

IOT@ascendeoiberia.com

CIRC-06

.:Música:.
.:Piezoeléctrico:.



¿QUÉ ESTAMOS HACIENDO?

Hasta este momento hemos controlado la luz, el movimiento y la electricidad. Vamos a probar con el sonido. Pero el sonido es un fenómeno analógico así que… ¿Cómo podrá dominarlo nuestro Arduino digital?

Nos basaremos de nuevo en esa velocidad increíble que hará posible imitar un comportamiento analógico. Para hacer esto, incorporaremos un piezoeléctrico a uno de los pines digitales de Arduino. Un piezoeléctrico emite un sonido cada vez que recibe un pulso de corriente. Si enviamos pulsos de la frecuencia correcta (por ejemplo 440 veces por segundo para conseguir un LA) estos clics se ejecutarán juntos para producir notas. Vamos a experimentar con ello y hacer que tu Arduino toque "Estrellita, ¿Dónde estás?".

EL CIRCUITO:

Piezas:

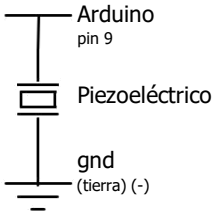
CIRC-06
Hoja plantilla
x 1

Conectores de dos pines x4

Piezoeléctrico
x1

Cable

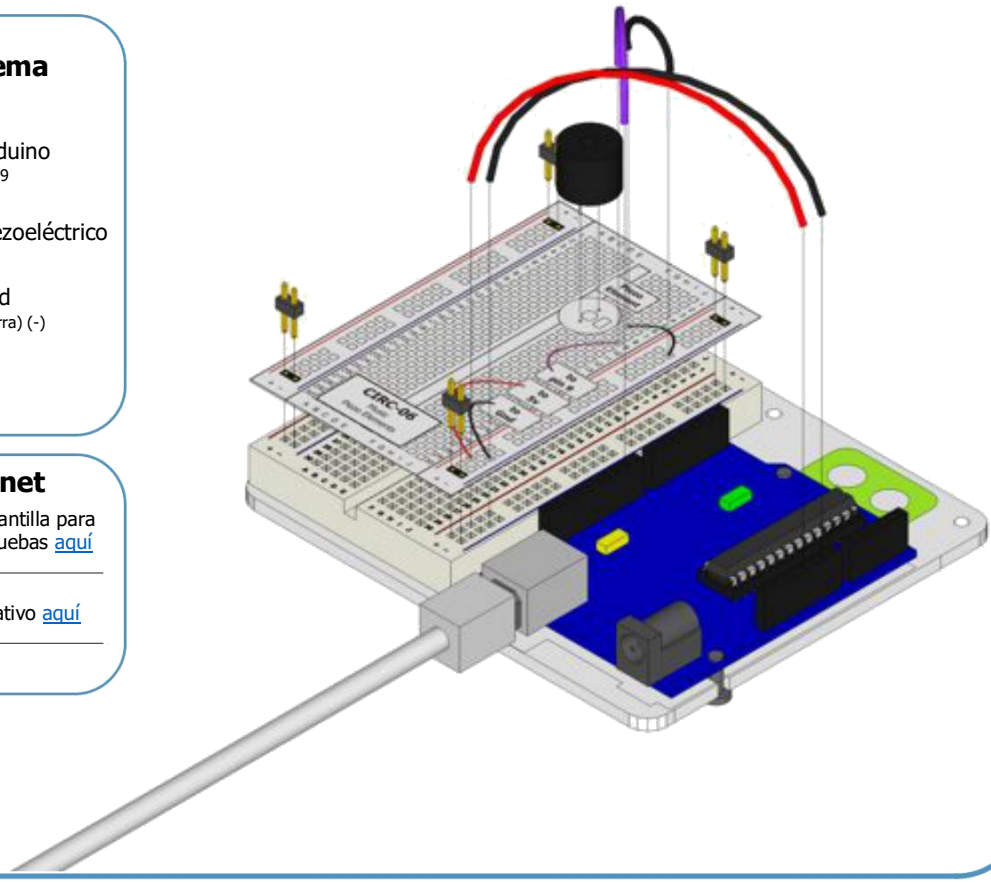
Esquema



Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí



CÓDIGO (No es necesario escribir todo, solo haz clic)

Descarga el código aquí

(Luego copia el texto y pégalo en una hoja Arduino)

```
/* Melodía
 *  (adaptación) 2005 D. Cuartielles para K3
 *
 *  Este ejemplo utiliza un piezo altavoz para reproducir melodías. Envía una
 *  onda cuadrada de la frecuencia apropiada al piezo
 *  generando el tono correspondiente.
 *
 *  El cálculo de los tonos se lleva a cabo siguiendo
 *  operaciones matemáticas:
 *
 *          timeHigh = period / 2 = 1 / (2 * toneFrequency)
 *
 *  donde los diferentes tonos se describen en la siguiente tabla:
 *
 * note          frequency  period          timeHigh
 * c (do)          261 Hz          3830          1915
 * d (re)          294 Hz          3400          1700
 * e (mi)          329 Hz          3038          1519
 * f (fa)          349 Hz          2864          1432
 * g (sol)         392 Hz          2550          1275
 * a (la)          440 Hz          2272          1136
 * b (si)          493 Hz          2028          1014
 * c (do)          523 Hz          1912           956
 *
 *  http://www.arduino.cc/en/Tutorial/Melody
 */
int speakerPin = 9;
int length = 15; // el número de notas
char notes[] = "ccggaagffeeddc "; // un espacio representa un descanso
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 }; int tempo = 300;


void playTone(int tone, int duration) {
    for (long i = 0; i < duration * 1000L; i += tone * 2) {
        digitalWrite(speakerPin, HIGH);
        delayMicroseconds(tone);
        digitalWrite(speakerPin,
LOW);
        delayMicroseconds(tone);
    }
}

void playNote(char note, int duration) {
    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956
};
        //    Reproduce el tono correspondiente al nombre de la nota
    for (int i = 0; i < 8; i++) {
        if (names[i] == note) {
            playTone(tones[i], duration);
        }
    }
}

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

void loop() {
    for (int i = 0; i < length; i++) { if (notes[i] == ' ') {

            delay(beats[i] * tempo); // rest
        } else {
            playNote(notes[i], beats[i] * tempo);
        }
                // pausa entre las notas
        delay(tempo / 2);                }
}
```

¿NO FUNCIONA? (Intenta estas tres cosas)

No suena

Dado el tamaño y forma del piezoeléctrico, es fácil omitir los agujeros correctos en la placa de pruebas. Comprueba su posición

¿No puedes pensar mientras suena la melodía?

Retira el piezoeléctrico mientras piensas, carga el programa y vuelve a colocarlo.

¿Te has cansado de "Estrellita, donde estás"?

El código está escrito así que puedes añadir tus propias canciones fácilmente. Fíjate en el código de abajo para empezar.

¿CÓMO MEJORARLO?

Jugando con la velocidad:

El tiempo para cada nota está calculado en base a variables y por lo tanto podemos ajustar el sonido de cada nota o el tiempo. Para cambiar la velocidad de la melodía necesitas cambiar solo una línea.

int tempo = 300; ---> int tempo = (new #)

Cámbiala a un número más alto para ralentizar la melodía, o a un número más pequeño para acelerarla.

Afinar las notas:

Si te preocupa que las notas estén un poco desafinadas, se puede arreglar. Las notas se han calculado basadas en una formula en el bloque de comentarios al principio del programa. Pero para afinar notas individuales solo tienes que ajustar sus valores en los tonos []. Prueba hacia arriba y hacia abajo hasta que suenen bien.

Cada nota se corresponde con su nombre en los nombres [] (array p.e. c = 1915)

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

Compón tus propias melodías:

El programa está pre configurado para tocar "Estrellita, ¿dónde estás?", pero la manera en la que está programado hace que cambiar la canción sea sencillo. Cada canción está definida en un *int* y dos *arrays*, la longitud del *int* define el número de notas, el primer *array notas []* define cada nota, y el segundo *beats []* define durante cuánto tiempo se toca cada nota. Algunos ejemplos:

Estrellita, ¿Dónde estás?

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

Cumpleaños Feliz (primera línea)

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

MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?

IOT@ascendeoiberia.com

.:Pulsar Botones:.
.:Pulsadores:.

¿QUÉ ESTAMOS HACIENDO?

Hasta este punto nos hemos centrado íntegramente en las salidas, es hora de hacer que nuestro Arduino escuche, vea y sienta. Empezaremos con un simple pulsador. El cableado del pulsador es simple. Hay un componente, la resistencia pull up, que puede parecer fuera de lugar. Está incluida porque un Arduino no siente del mismo modo que lo hacemos nosotros (p.e. botón pulsado, botón sin pulsar). Por el contrario, se fija en el voltaje del pin y decide si es HIGH o LOW. El botón está configurado para marcar el pin Arduino como LOW cuando se pulsa, sin embargo, cuando el botón no está pulsado, el voltaje del pin flotará (causando errores ocasionales). Para conseguir que el Arduino lea el pin como HIGH cuando el botón no está pulsado, añadimos la resistencia pull up.

(Nota: el primer programa de ejemplo usa solamente uno de los dos botones)

EL CIRCUITO:

Piezas:

CIRC-07
Hoja plantilla x1

Conectores de dos pines x4

Pulsador x2

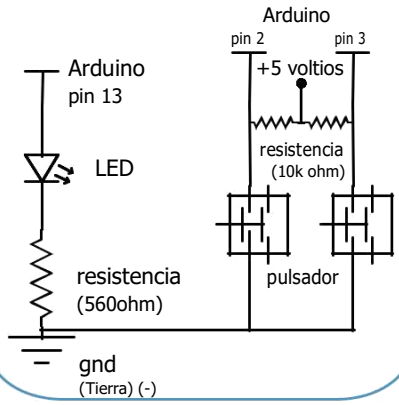
Cable

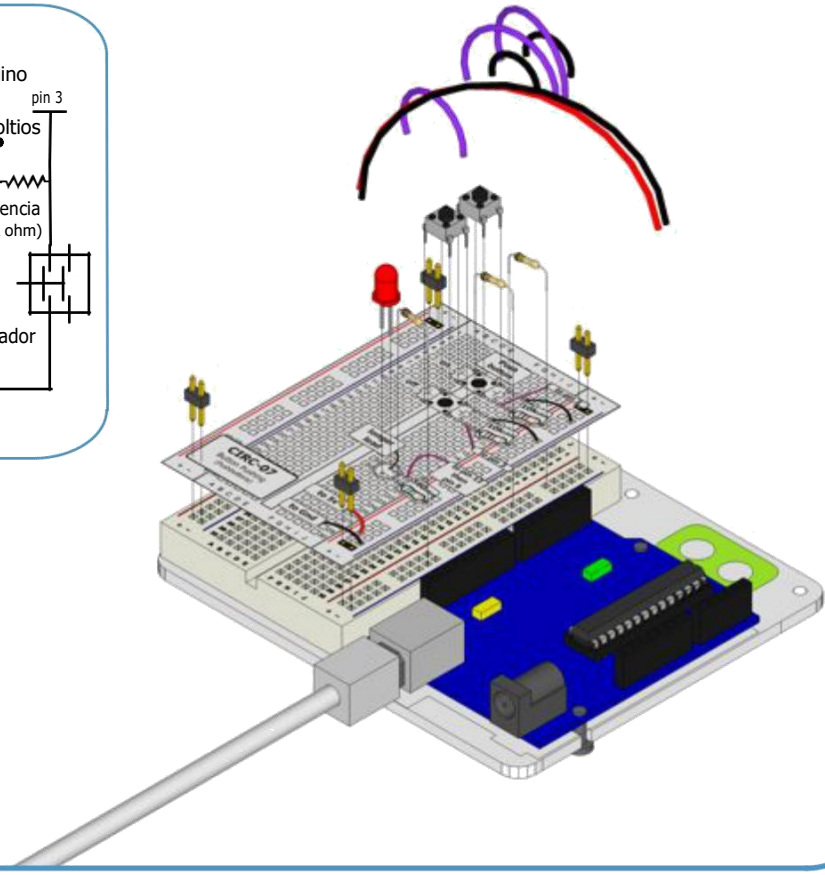
Resistencia de 10k Ohm
Marrón-Negro-Naranja x2

Resistencia de 560 Ohm
Verde-Azul-Marrón x1

Led Rojo x1

Esquema





Internet

Descarga la plantilla para la placa de pruebas aquí

Video explicativo aquí

CÓDIGO (No es necesario escribir todo, solo haz clic)

Archivo > Ejemplos > 2.Digital > Pulsador (Button)

(Ejemplo de la página web arduino.cc, visítala para otras grandes ideas)

```
/*
  Pulsador (Button)
  Por DojoDave <http://www.0j0.org>

  Enciende y apaga un diodo emisor de luz (LED) conectado al pin digital
  13, cuando se presiona un pulsador conectado al pin 7.
  http://www.arduino.cc/en/Tutorial/Button
 */
  int ledPin = 13;                    // elige el pin para el LED
int inputPin = 2;                     // elige el pin de entrada (para el pulsador)
int val = 0;                          // variable para leer el estado del pin

void setup() {
   pinMode(ledPin, OUTPUT);           // declarar LED como salida
   pinMode(inputPin, INPUT);          // declarar pulsador como entrada
}

void loop(){
   val = digitalRead(inputPin);       // leer el valor de entrada
   if (val == HIGH) {                 // comprobar que la entrada sea HIGH
      digitalWrite(ledPin, LOW);      // apagar el LED
   } else {
      digitalWrite(ledPin, HIGH); // encender el LED
   }
}
```

¿NO FUNCIONA? (Intenta estas tres cosas)

La luz no se enciende

El pulsador es cuadrado y por ese motivo es fácil colocarlo mal. Gíralo 90 grados y a ver si empieza a funcionar.

La luz no se desvanece

Un error tonto que repetimos constantemente es que, cuando cambiamos de un simple encendido/apagado a bajar la intensidad, hay que cambiar el cable del LED del pin 13 al pin 9

¿Decepcionado?

No te preocupes. Todos los circuitos están muy simplificados para hacer que jugar con los componentes sea fácil. Pero si los combinas entre ellos, el cielo es el límite.

¿CÓMO MEJORARLO?

Pulsador de encendido, pulsador de apagado:

El ejemplo inicial puede ser un poco decepcionante (En realidad no necesitas un Arduino para hacer esto), así que vamos a complicarlo un poco más. Un pulsador encenderá el LED y el otro lo apagará. Cambia el código a:

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

void setup() {
pinMode(ledPin, OUTPUT);   //                Declarar LED como salida
pinMode(inputPin1, INPUT); //              Hacer pulsador 1 una entrada
pinMode(inputPin2, INPUT); //              Hacer pulsador 2 una entrada
}

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

Carga el programa en tu placa y empieza a encender y apagar el LED.

Regular la intensidad de la luz:

Vamos a usar los pulsadores para controlar la señal analógica. Para esto necesitarás cambiar el cable de conexión del LED del pin 13 al pin 9, y también cambiar este código:

int ledPin = 13; ----> int ledPin = 9;

Luego cambia el código del bucle () con este.

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

Cambiar la velocidad de cambio de intensidad:

Si quieres que el LED cambie a más o menos intensidad de luz más rápido o más lento, sólo tienes que cambiar una línea del código:

```
delay(10);     ----> delay(new #);
```

Irá más rápido con un número mayor y más lento con un número menor.

MÁS, MÁS, MÁS:

¿Necesitas más detalles o tienes alguna pregunta?

IOT@ascendeoiberia.com