

- [Arduino](#)
 - [Introducción](#)
 - [Placa Arduino](#)
 - [Instalación IDE](#)
- [Protoboard](#)
 - [Zonas](#)
- [Motores](#)
 - [Motor de corriente continua](#)
 - [Motor paso a paso mando](#)
 - [Motor paso a paso](#)
 - [Motor servo](#)
- [Luces](#)
 - [LED interno](#)
 - [LED](#)
 - [LED RGB](#)
 - [Tira de LEDs](#)
 - [7 segments \(1 dígito\)](#)
 - [7 segments \(4 dígitos\)](#)
 - [Pantalla LCD](#)
 - [LED con 74HC595](#)

Arduino

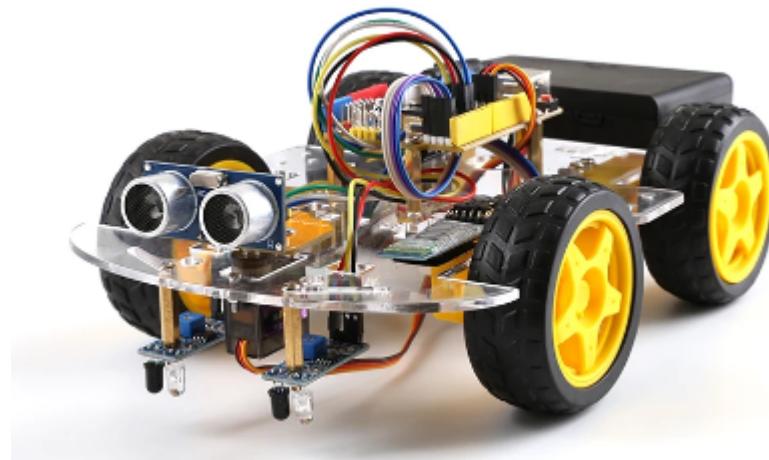
Introducción

Arduino es una plataforma de [hardware libre](#), basada en una placa con un microcontrolador y un entorno de desarrollo.

Hardware libre es aquel cuyas especificaciones y diagramas esquemáticos son de acceso público



Arduino está diseñado para permitir que los usuarios creen proyectos con una programación sencilla, sin necesidad de conocimientos previos de electrónica.



Placa Arduino

La placa **Arduino** contiene un microcontrolador, que es una pequeña computadora que puede leer los datos de los sensores que se conectan, realizar algunas operaciones matemáticas y controlar los dispositivos a través de los pines de salida.



El **microcontrolador** puede ser programado para recibir instrucciones. Esto significa que los usuarios pueden **programar** la placa para que realice cualquier tarea que deseen, desde controlar motores hasta leer datos de sensores.

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);       // turn the LED off
  delay(1000);                      // wait for a second
```

Instalación IDE

El **Entorno de desarrollo integrado (IDE) de Arduino** es el software de la plataforma **Arduino**. En esta lección, usted aprenderá cómo configurar tu ordenador para usar **Arduino** y cómo establecer sobre las lecciones que siguen.

Paso 1: Ir a <https://www.arduino.cc/en/Main/Software> y a continuación de la página.

The screenshot shows the Arduino Software (IDE) download page. At the top left is the Arduino logo, which is a teal circle containing a white infinity symbol with a minus sign on the left and a plus sign on the right. To the right of the logo, the text "ARDUINO 1.8.0" is displayed in bold. Below this, there is a brief description of the IDE: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the far right, there are download links for different operating systems: "Windows Installer", "Windows ZIP file for non admin install", "Windows app" (with a "Get" button), "Mac OS X 10.7 Lion or newer", "Linux 32 bits", "Linux 64 bits", "Linux ARM", "Release Notes", "Source Code", and "Checksums (sha512)".

La versión disponible en este sitio web es generalmente la última versión y la versión actual puede ser más reciente que la versión en el cuadro.

Paso 2 : Descargar el desarrollo software que es compatible con el sistema operativo del ordenador. Windows tomar como un ejemplo aquí

The screenshot shows the Arduino Software (IDE) download page. It features the same layout as the previous screenshot, with the Arduino logo, the "ARDUINO 1.8.0" heading, the product description, and the download links for various platforms. The "Windows app" section is highlighted with a larger "Get" button.

Haga click en Windows Installer.

Support the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). Learn more on how your contribution will be used.



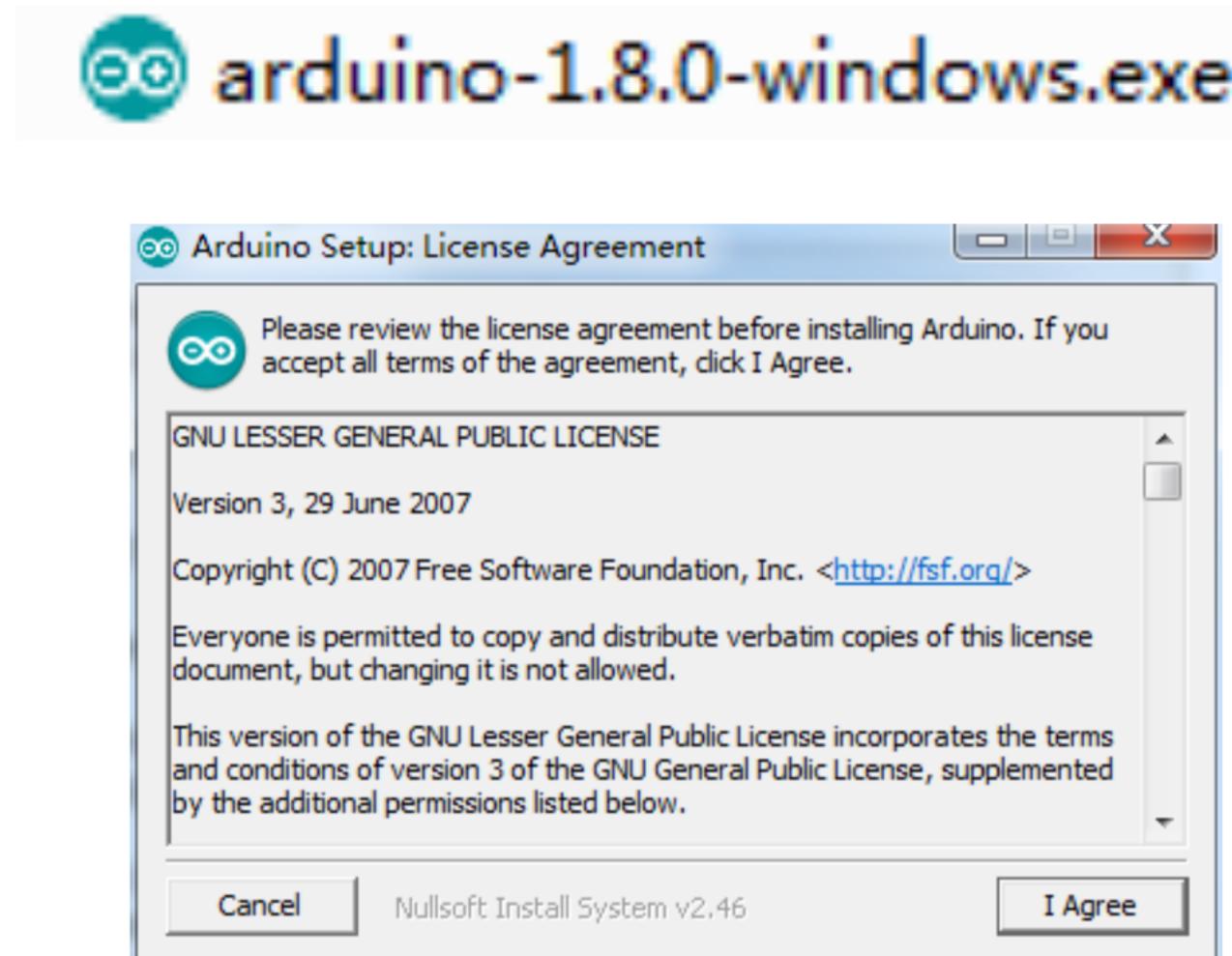
Haga click en DESCARGAR (JUST DOWNLOAD).

También está disponible en el material que nos proporciona la versión 1.8.0, y las versiones de nuestros materiales son las últimas versiones cuando se hizo este manual.

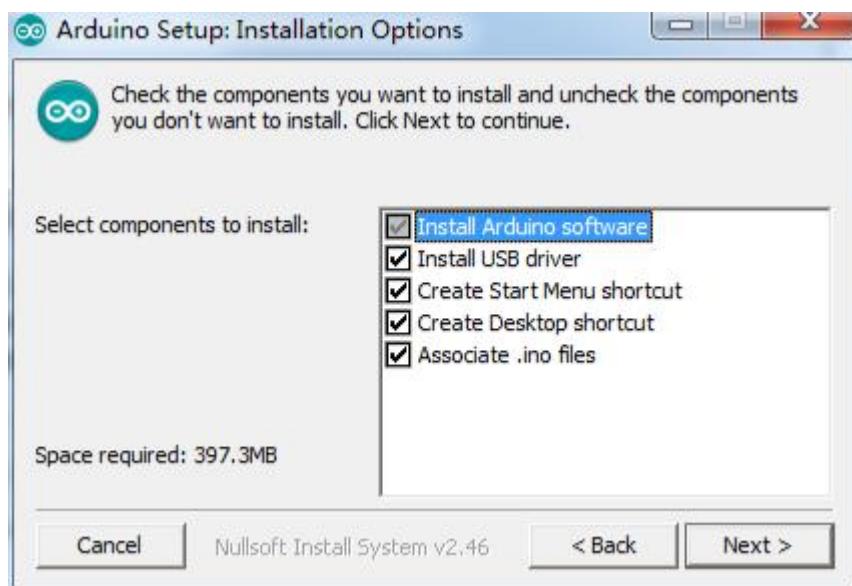
- arduino-1.8.0-linux32.tar.xz
- arduino-1.8.0-linux64.tar.xz
- arduino-1.8.0-macosx.zip
- arduino-1.8.0-windows.exe
- arduino-1.8.0-windows.zip

Instalación de **Arduino** (Windows)

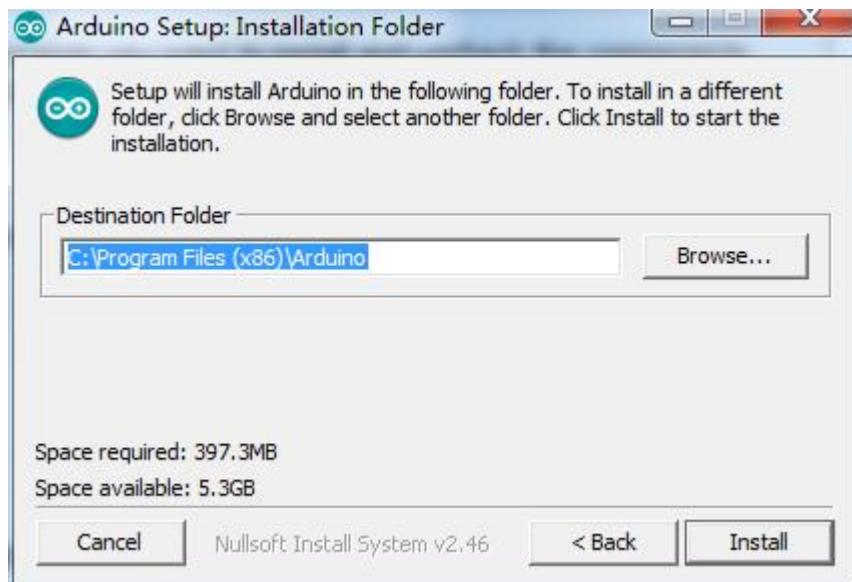
Instalar **Arduino** con el exe. Paquete de instalación.



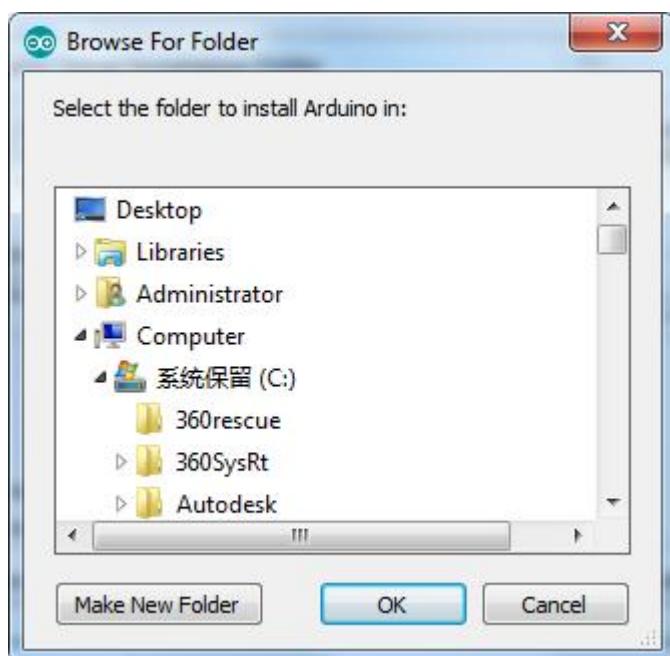
Haga click en I Agree to see de esta ventana



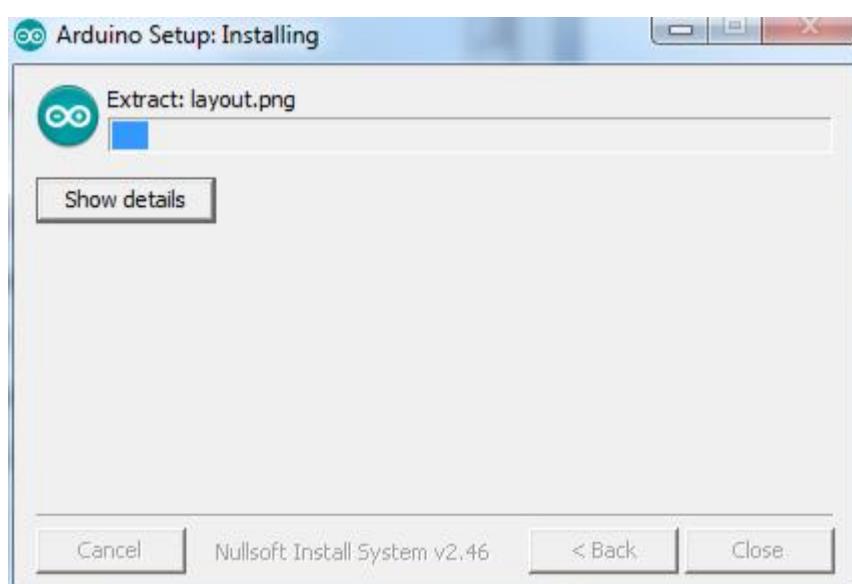
Click Next



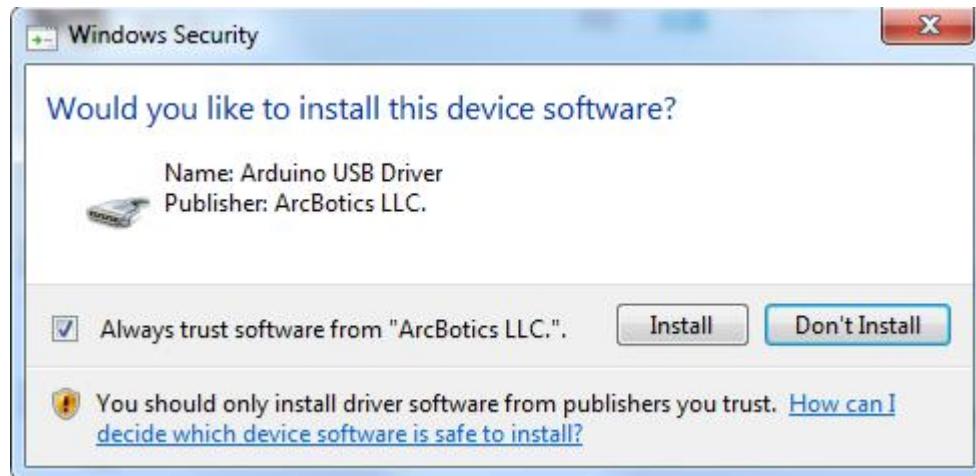
Puede pulsar examinar... elegir una ruta de instalación o directamente en el directorio que desee.



Haga Click en Install para comenzar la instalación



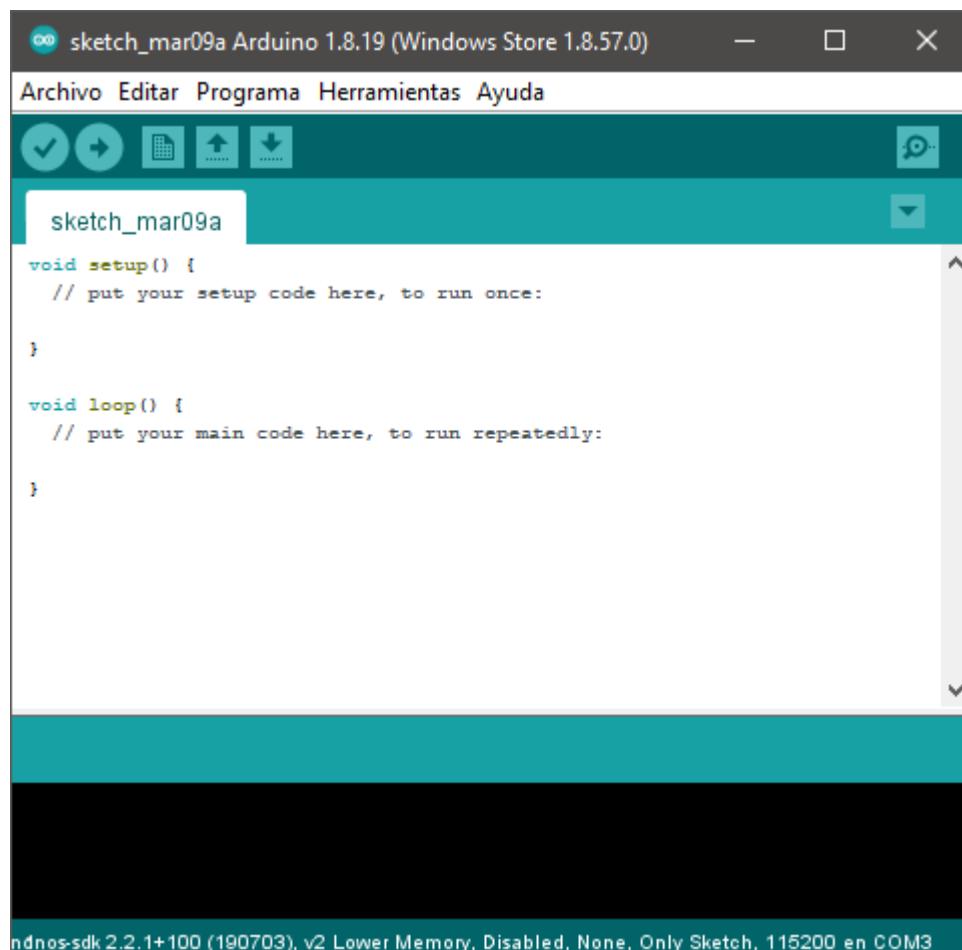
Por último, aparece la siguiente ventana, haga clic en Install para finalizar la instalación.



A continuación, aparece el siguiente ícono en el escritorio



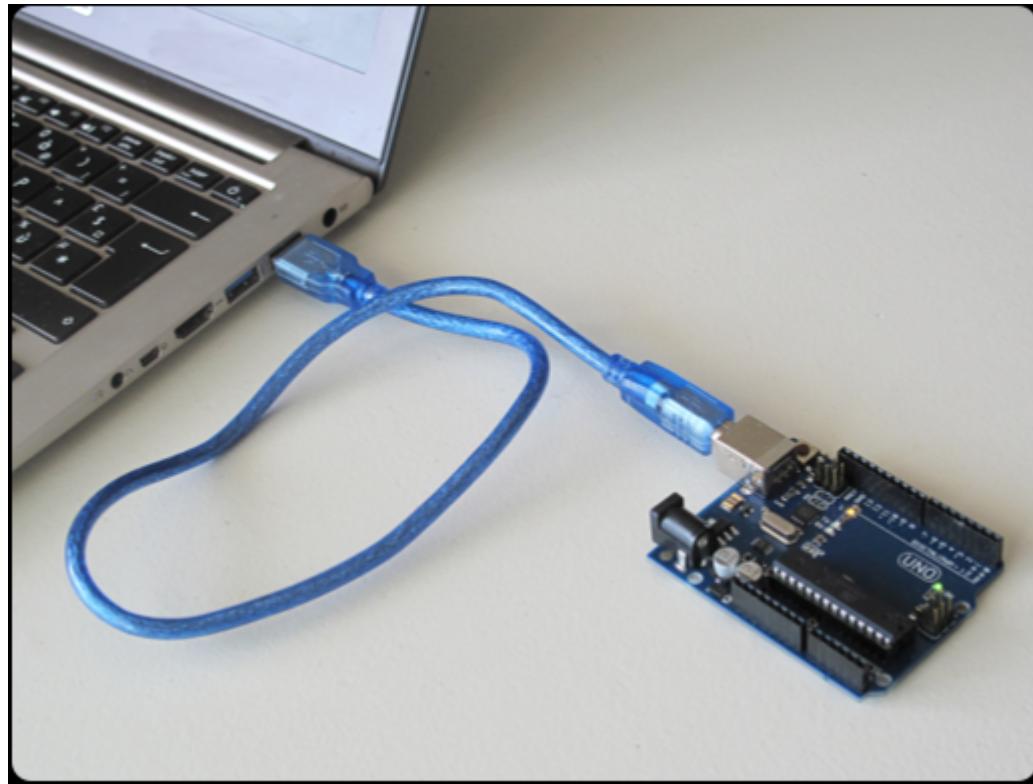
Haga doble clic para entrar en el entorno de desarrollo deseado



Conexión de placa [Arduino](#)

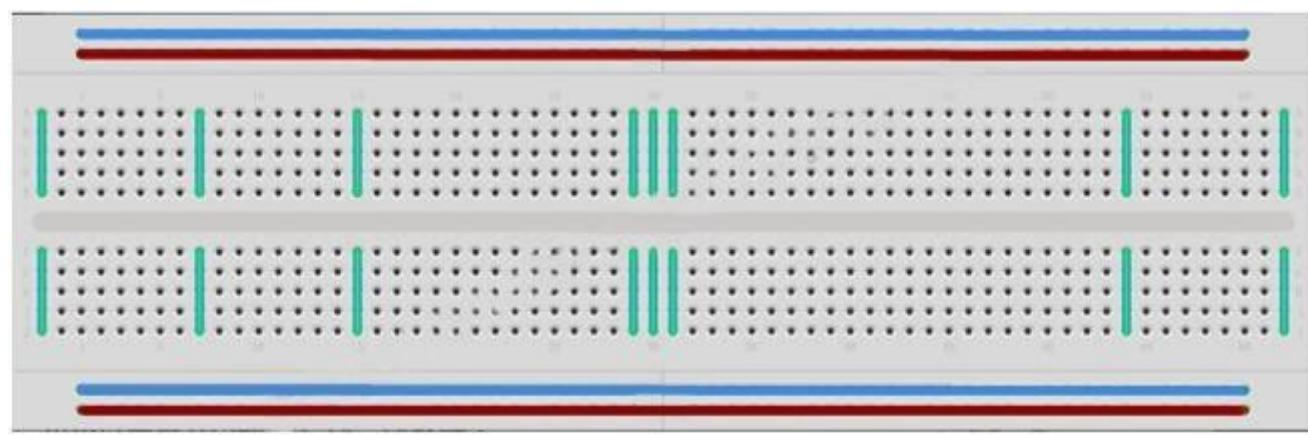
La carpeta de **Arduino** incluye el propio **programa de Arduino** y los **controladores** que permiten que el **Arduino** se conecte al ordenador mediante un cable USB.

Conecte su cable **USB** en el **Arduino** y en el USB ordenador.

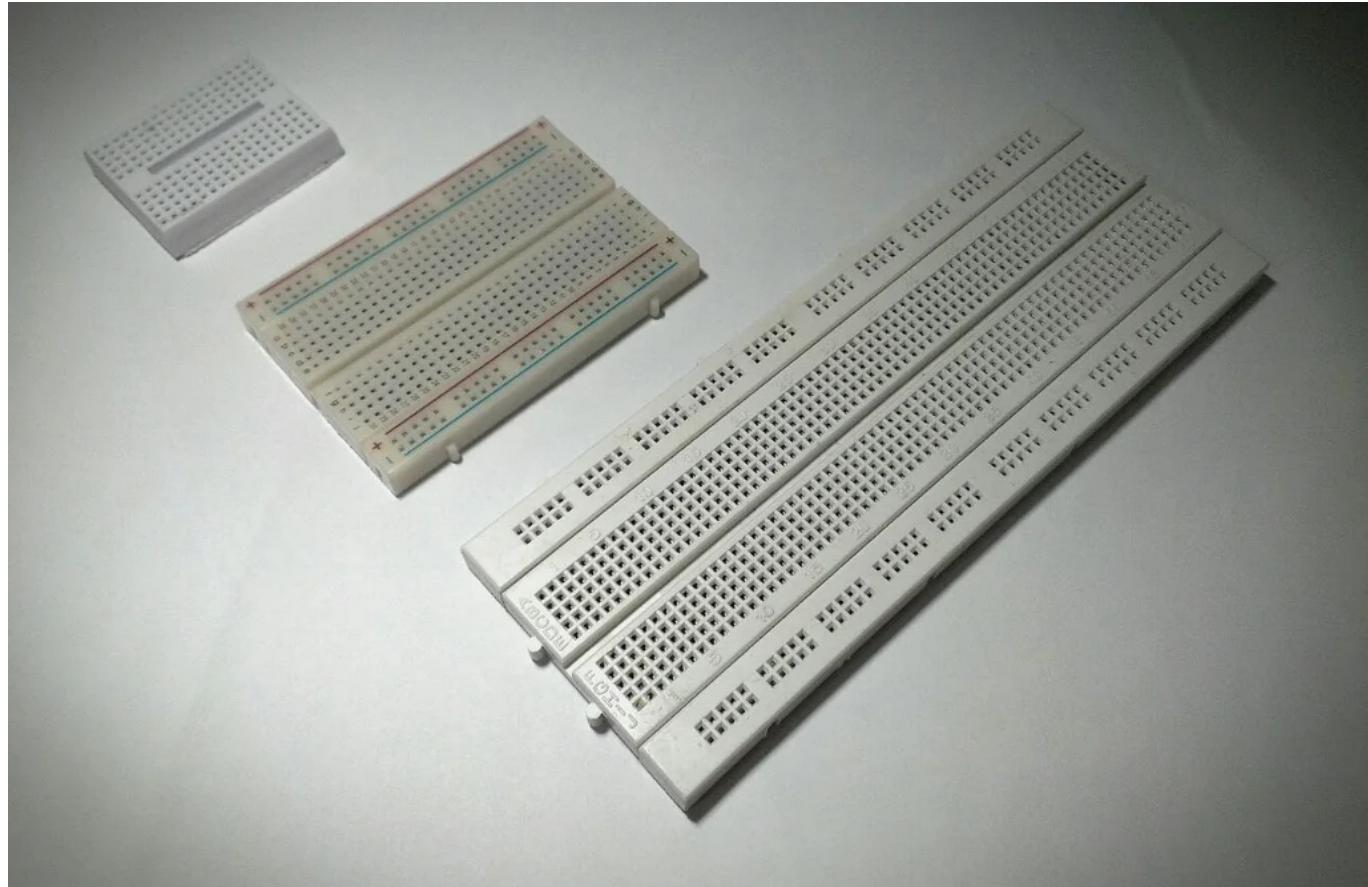


Protoboard

Un **protoboard** permite crear prototipos de circuitos de forma rápida, sin necesidad de soldar las conexiones. A continuación un ejemplo.



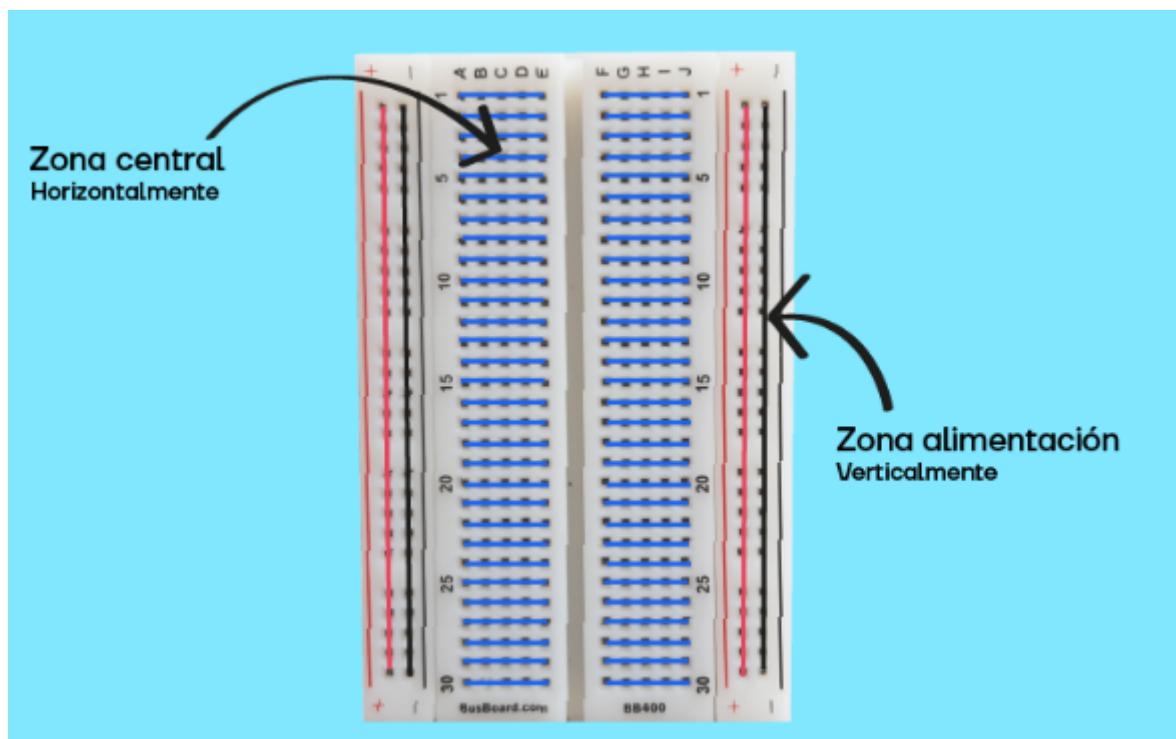
Las **protoboard** vienen en varios tamaños y configuraciones. La clase más simple es sólo una rejilla de agujeros en un bloque de plástico. En el interior son tiras de metal que proporcionan una conexión eléctrica entre los agujeros en las filas más cortas.



Conectando las patillas de dos componentes diferentes de la **misma fila** quedan unidos eléctricamente.

Zonas

Existen dos zonas principales. La zona de alimentación, y la zona central. Todo lo que pase por la misma línea, está interconectado eléctricamente.



#Canales centrales

Los canales centrales indican que no están conectados ambos lados. Lo que significa, puede insertar un chip con las patillas a ambos lados del canal sin conectarlos juntos.

#Tiras laterales

- Algunos **protoboards** tienen dos tiras de agujeros que corre a lo largo de los bordes laterales del tablero.
- Estas tiras se denominan **carriles** y le permiten alimentar a muchos componentes o puntos en el tablero.
- Proporcionan una manera para conectar una tensión común. Son generalmente en pares para + 5 voltios y tierra.

#Desventajas

Si bien son protoboards para prototipos, tienen algunas limitaciones. Porque las conexiones son temporales y de acople, no son tan fiables como conexiones soldadas. Si tienes problemas intermitentes con un circuito, puede ser debido a una mala conexión en una protoboard.

Motores

Els motors d'Arduino són dispositius que permeten a una placa Arduino controlar el moviment mecànic d'un sistema. Els motors poden ser de diferents tipus i formes, i poden ser controlats per la placa Arduino a través de diferents circuits i protocols.

Els motors més comuns utilitzats amb Arduino són els **motors de corrent continu** (DC) i els **servomotors**.

- Els **motors de corrent continu** són motors que giren en una direcció o altra dependent del sentit de la corrent que passa per ells, i poden ser controlats a través d'un circuit que permet variar la tensió aplicada al motor.
- Els **servomotors**, d'altra banda, són motors que poden ser controlats amb precisió per a posicionar-se en un determinat angle, i són utilitzats en molts projectes de robòtica i control de moviment.
- Els **motors pas a pas** són un tipus de motor que es caracteritza per moure's en increments precisos de posició en lloc de girar continuament

Motor de corriente continua

La fuerza máxima que puede generar un motor pequeño de corriente continua para proyectos de electrónica depende de varios factores, como el diseño y las especificaciones del motor. Sin embargo, en general, los motores pequeños de corriente continua para proyectos de electrónica suelen tener una fuerza máxima relativamente baja.

La fuerza generada por un motor de corriente continua está relacionada con su **torque**. El torque es una medida de la capacidad del motor para generar una fuerza de rotación. Los motores pequeños para proyectos de electrónica generalmente tienen un torque bajo y están diseñados para aplicaciones de baja carga, como mover pequeños mecanismos o generar movimiento en juguetes pequeños.

El **torque máximo** de un motor se especifica en su datasheet o hoja de datos proporcionada por el fabricante. Puede estar en unidades como gramos-centímetro ($\text{g}\cdot\text{cm}$) o milinewton-metro ($\text{mN}\cdot\text{m}$). Es

importante tener en cuenta que el torque máximo disminuye a medida que aumenta la velocidad de rotación del motor.

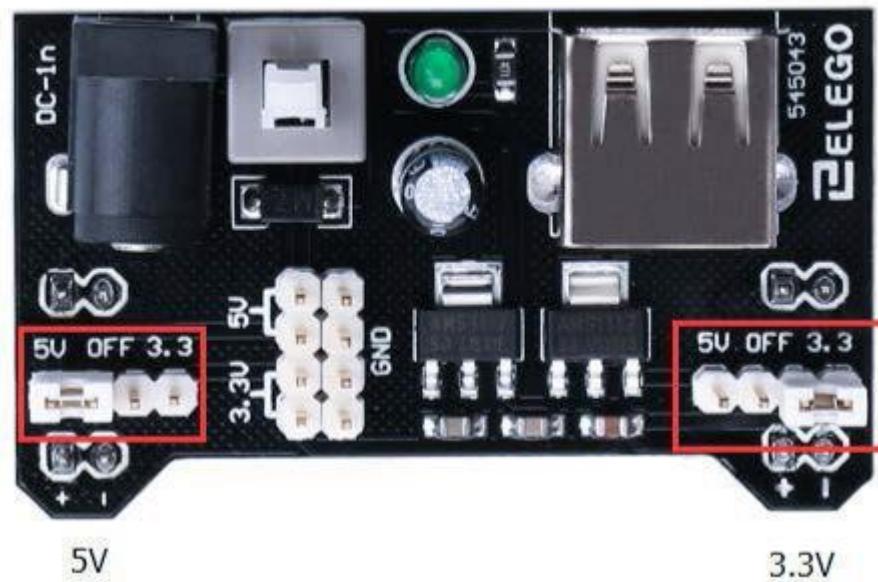
Fuente de alimentación de la placa

El pequeño motor de corriente continua es probable que use más energía que la que [Arduino](#) puede suministrar. Si tratamos de conectar el motor directamente a un pin, podríamos dañarlo. Para ello usar un **módulo de alimentación** que proporciona electricidad al motor.

Especificaciones

- Voltaje de entrada: 6.5-9v (CC)
- Voltaje de salida: 3.3V / 5v
- Máxima corriente de salida: 700 mA

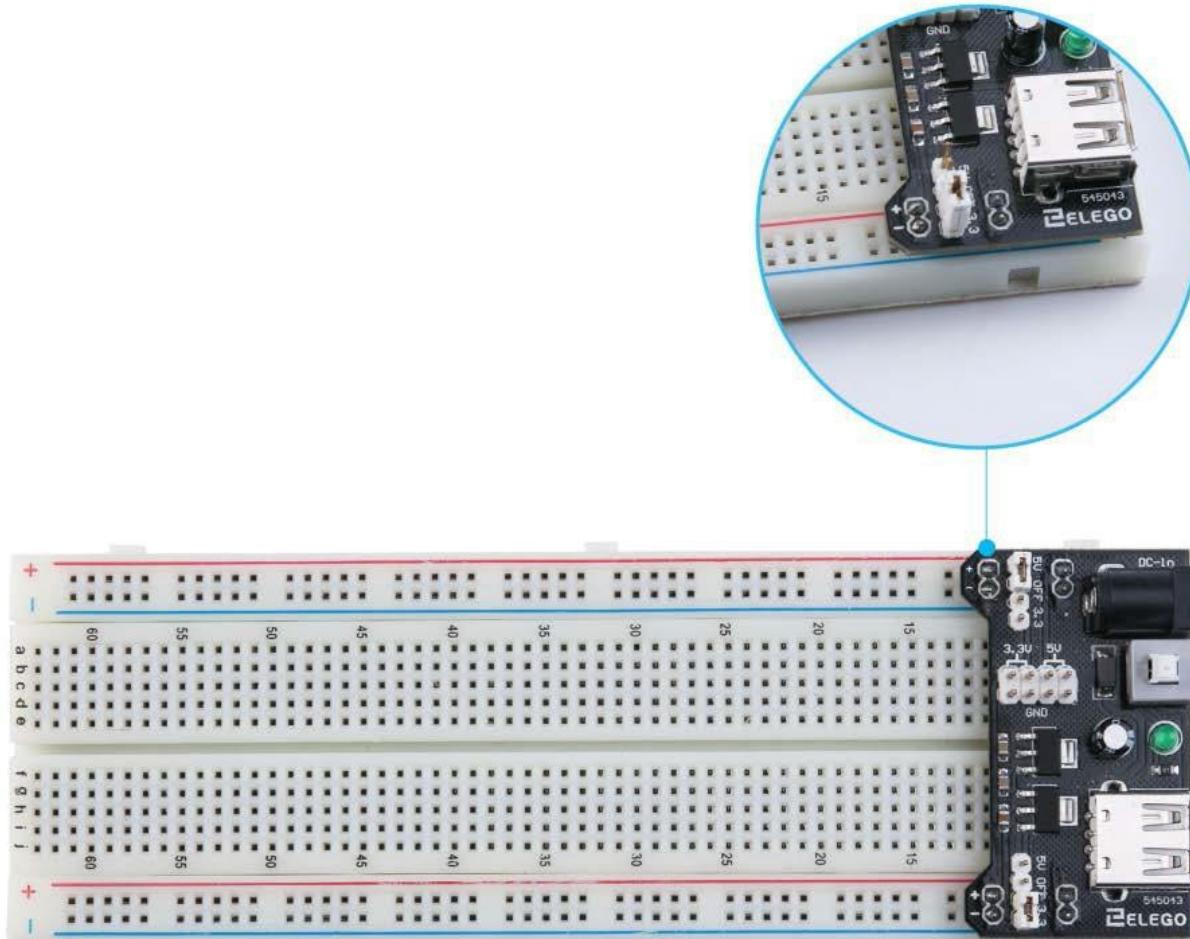
Configuración de voltaje



La izquierda y derecha de la tensión de salida puede configurarse independientemente. Para seleccionar la tensión de salida, mover el puente a los pines correspondientes. Nota: indicador de energía LED y los carriles de la energía de protoboard no se enciende si ambos puentes están en la posición "OFF".

Importante

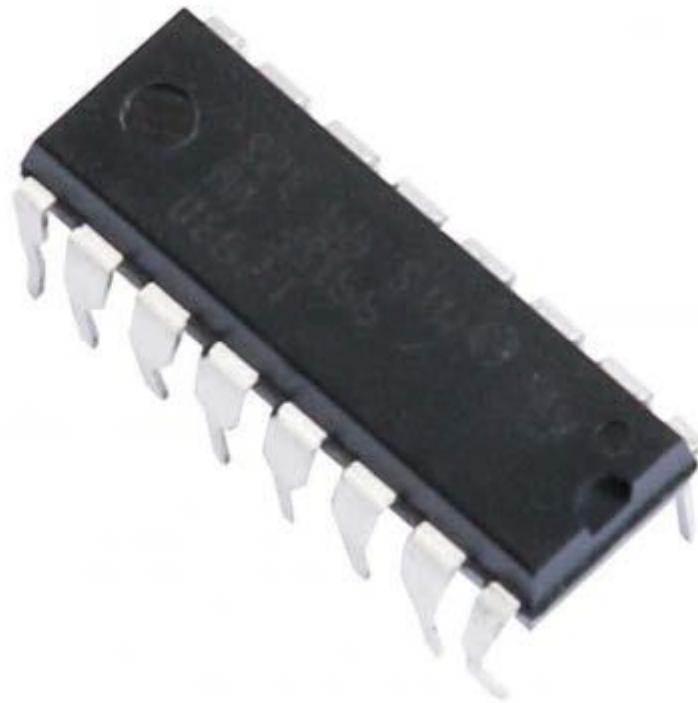
Asegúrese de alinear el módulo correctamente en la placa de pruebas.



L293D

El L293D és un circuit integrat que s'utilitza com a controlador de motor i permet controlar la direcció i la velocitat d'un motor DC.

El dispositiu inclou quatre drivers de pont H, que permeten controlar fins a dos motors DC de manera independent.



Especificaciones

| Característica | Valor | ----- || | Tensión de alimentación | 4,5 V a 36 V | | Salida de corriente | 1 A por canal (600 mA para el L293D) | | Máxima salida de corriente | 2 A por canal (1.2 A para L293D) |

Diagrama de pines

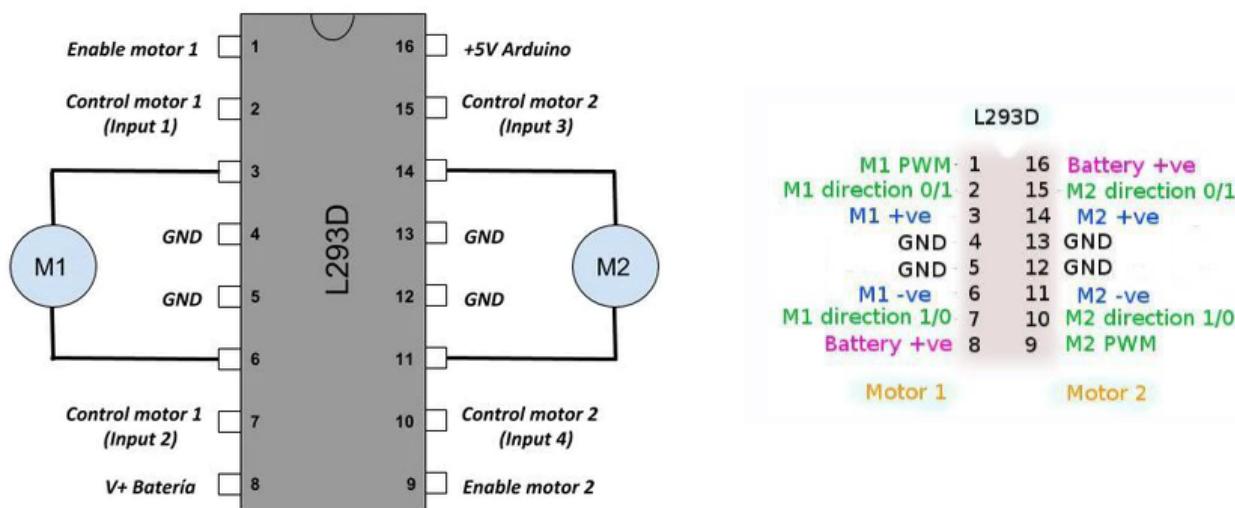


L293 y L293D

- El L293 está diseñado para proporcionar corrientes de transmisión bidireccional de hasta 1 A con tensiones de 4,5 V a 36 V.
- El L293D está diseñado para proporcionar bidireccional corrientes de impulsión de hasta 600 mA en tensiones de 4,5 V a 36 V.

Pines

- 4 pins per controlar la direcció dels motors
- 1 pin s'utilitza per controlar la velocitat.

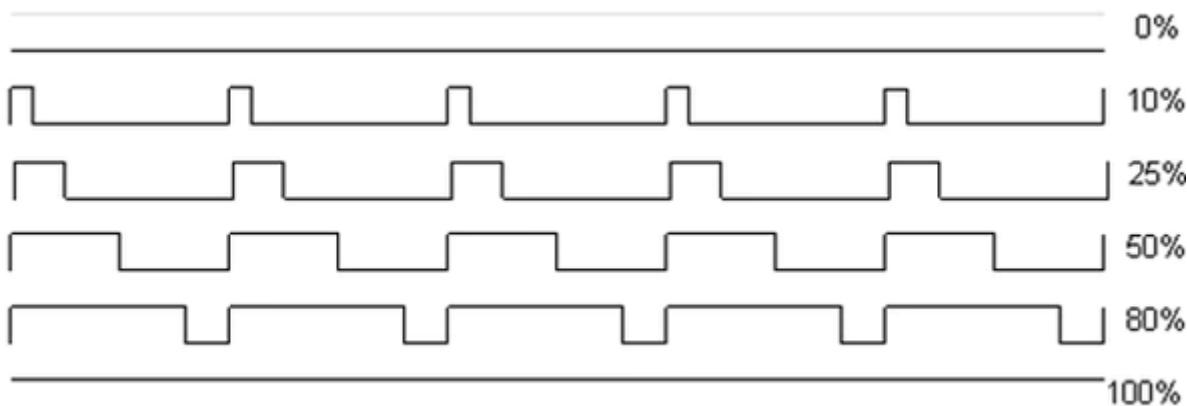


Control de la velocidad

M1 PWM lo conectaremos a un pin PWM de **Arduino**. Está marcados en la ONU, el pin 5 es un ejemplo. Cualquier número entero entre 0 y 255, donde:

- 0 significa velocidad 0 (no hay movimiento)
- 128 es la mitad de velocidad
- 255 es la velocidad máxima de salida.

Según el valor que escribamos, se generará una señal PWM diferente.



Dirección de giro

La dirección se controla a través de las entradas de dirección:

- **M1 0/1** y **M1 1/0** determinan el sentido de giro del motor 1
- **M2 0/1** y **M2 1/0** determinan el sentido de giro del motor 2

Dirección de giro

M1 PWM	1	16	Battery +ve
M1 direction 0/1	2	15	M2 direction 0/1
M1 +ve	3	14	M2 +ve
GND	4	13	GND
GND	5	12	GND
M1 -ve	6	11	M2 -ve
M1 direction 1/0	7	10	M2 direction 1/0
Battery +ve	8	9	M2 PWM

Motor 1

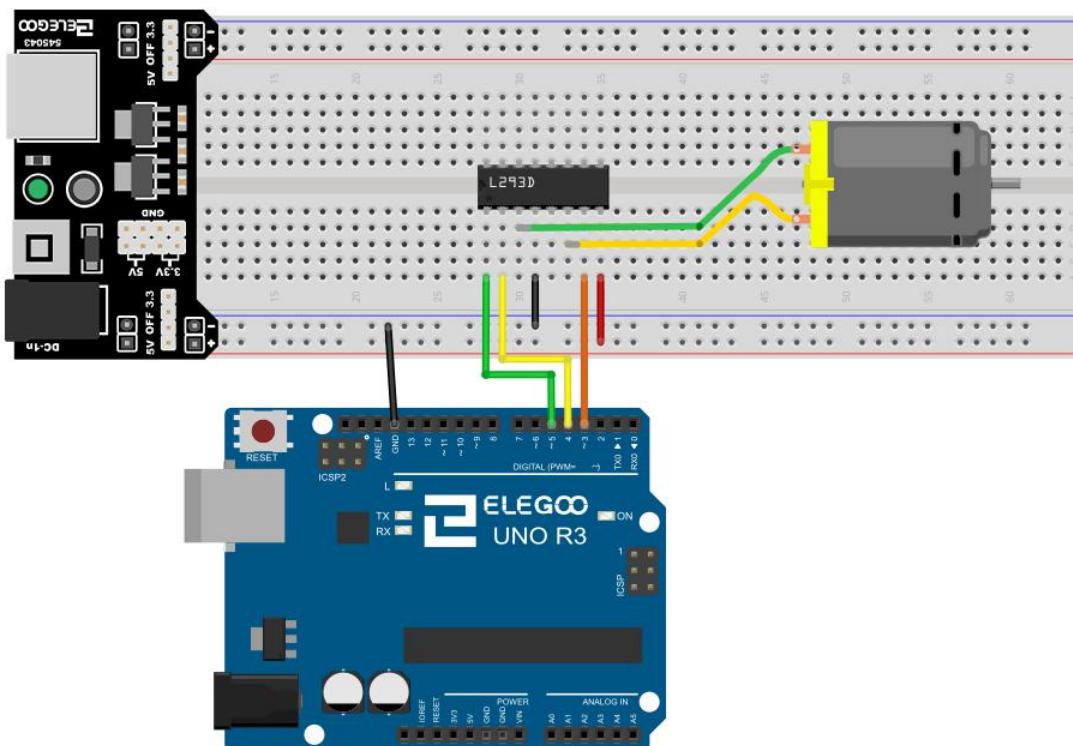
Motor 2

Dirección de giro

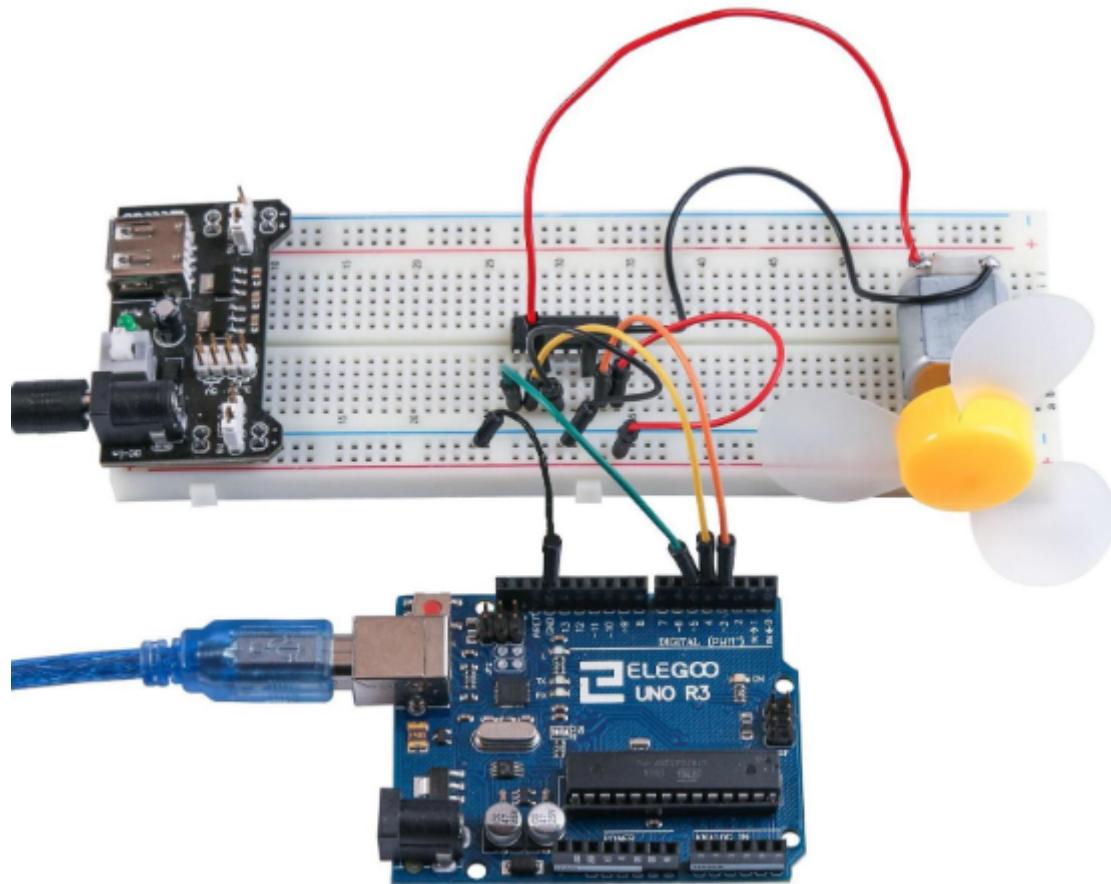
En la siguiente tabla veréis las 4 combinaciones posibles para el motor 1:

Pin 2	Pin 7	Salida
LOW	LOW	Detenido
LOW	HIGH	Derecha
HIGH	LOW	Izquierda
HIGH	HIGH	Detenido

Esquema



Montaje físico



Código

```
#define ENABLE 5
#define DIRA 3
#define DIRB 4

int i;

void setup() {
    //set pin direction
    pinMode(ENABLE,OUTPUT);
    pinMode(DIRA,OUTPUT);
    pinMode(DIRB,OUTPUT);
    Serial.begin(9600);
}

void loop() {
    //back and forth example
    Serial.println("One way, then reverse");
    digitalWrite(ENABLE,HIGH); // enable on
    for (i=0;i<5;i++) {
        digitalWrite(DIRA,HIGH); //one way
        digitalWrite(DIRB,LOW);
        delay(500);
        digitalWrite(DIRA,LOW); //reverse
        digitalWrite(DIRB,HIGH);
    }
}
```

```
    delay(500);
}
digitalWrite(ENABLE,LOW); // disable
delay(2000);

Serial.println("fast Slow example");
//fast/slow stop example
digitalWrite(ENABLE,HIGH); //enable on
digitalWrite(DIRA,HIGH); //one way
digitalWrite(DIRB,LOW);
delay(3000);
digitalWrite(ENABLE,LOW); //slow stop
delay(1000);
digitalWrite(ENABLE,HIGH); //enable on
digitalWrite(DIRA,LOW); //one way
digitalWrite(DIRB,HIGH);
delay(3000);
digitalWrite(DIRA,LOW); //fast stop
delay(2000);

Serial.println("PWM full then slow");
//PWM example, full speed then slow
analogWrite(ENABLE,255); //enable on
digitalWrite(DIRA,HIGH); //one way
digitalWrite(DIRB,LOW);
delay(2000);
analogWrite(ENABLE,180); //half speed
delay(2000);
analogWrite(ENABLE,128); //half speed
delay(2000);
analogWrite(ENABLE,50); //half speed
delay(2000);
analogWrite(ENABLE,128); //half speed
delay(2000);
analogWrite(ENABLE,180); //half speed
delay(2000);
analogWrite(ENABLE,255); //half speed
delay(2000);
digitalWrite(ENABLE,LOW); //all done
delay(10000);
}
```

Motor paso a paso mando

Veremos cómo funciona el motor de pasos de 4 fases **ULN2003 28BYJ-48** y el controlador de motor de 5V.

El motor

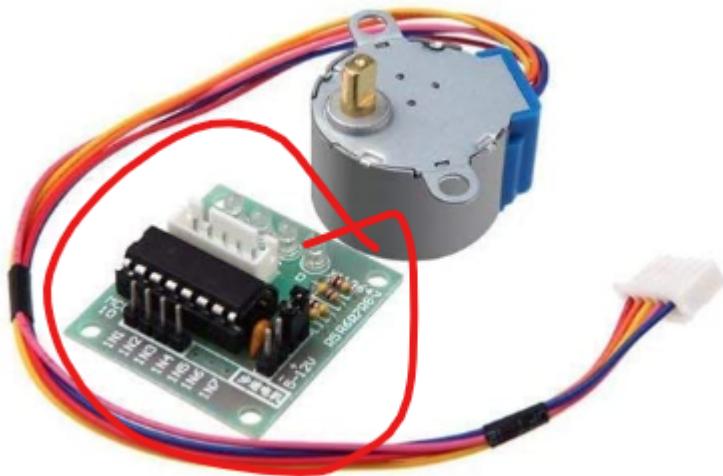
El motor de pasos de 4 fases ULN2003 28BYJ-48 es un motor de pasos pequeño y económico que se puede controlar con un microcontrolador. El motor tiene 4 fases, cada una con 2 polos. Cada fase requiere energía

para que el imán se atraiga o se repulse. Los 4 imanes del motor de pasos están dispuestos de forma que se atraigan y se repulsen en secuencia, lo que hace que el eje del motor gire.



Controlador

Para controlar el motor de pasos de 4 fases, se necesita un controlador de motor. El controlador de motor de 5V es un circuito integrado que se usa para controlar el motor de pasos.



El controlador de motor tiene 8 salidas, cada una conectada a una fase del motor. Para hacer que el motor gire, se activan las salidas en secuencia.

#Componentes necesarios

- (1) x Elegoo Uno R3
- x 830 tie-points breadboard
- x IR receiver module
- x IR remote
- x ULN2003 stepper motor driver module
- x Stepper motor
- x Power supply module
- x 9V1A Adaptador

- x F-M cables (cables de hembra a macho DuPont)
- x M-M cable (hilo puente de macho a macho)

##Esquema

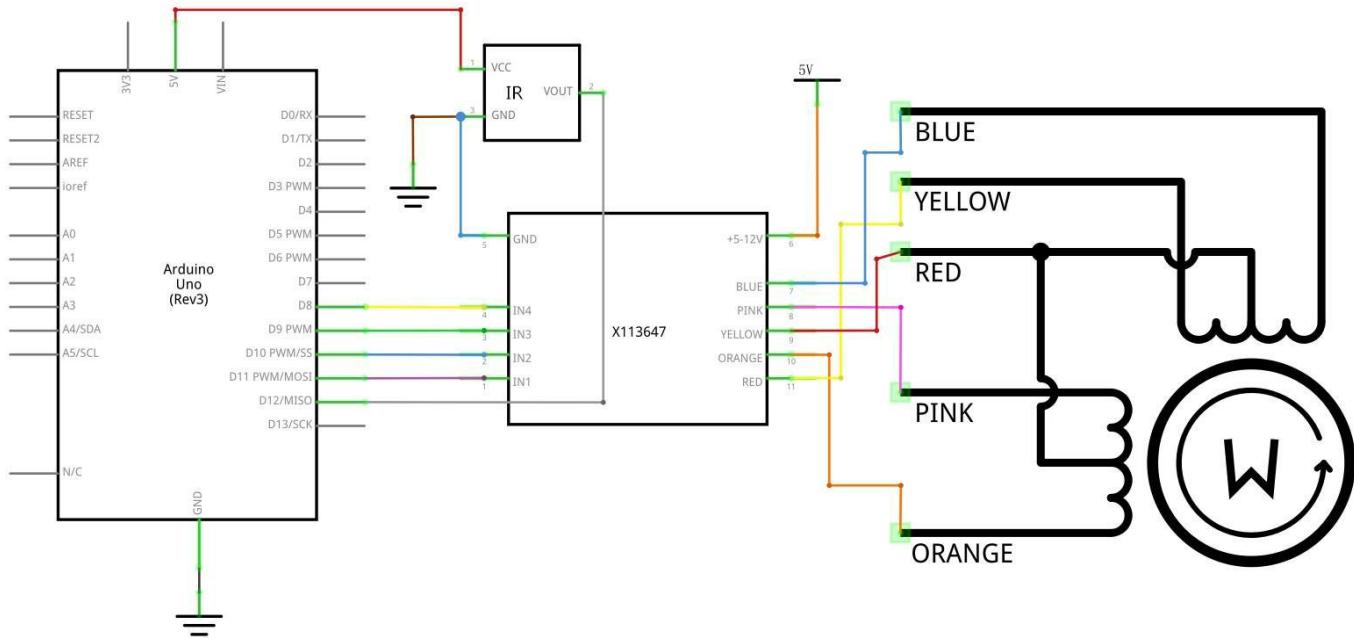
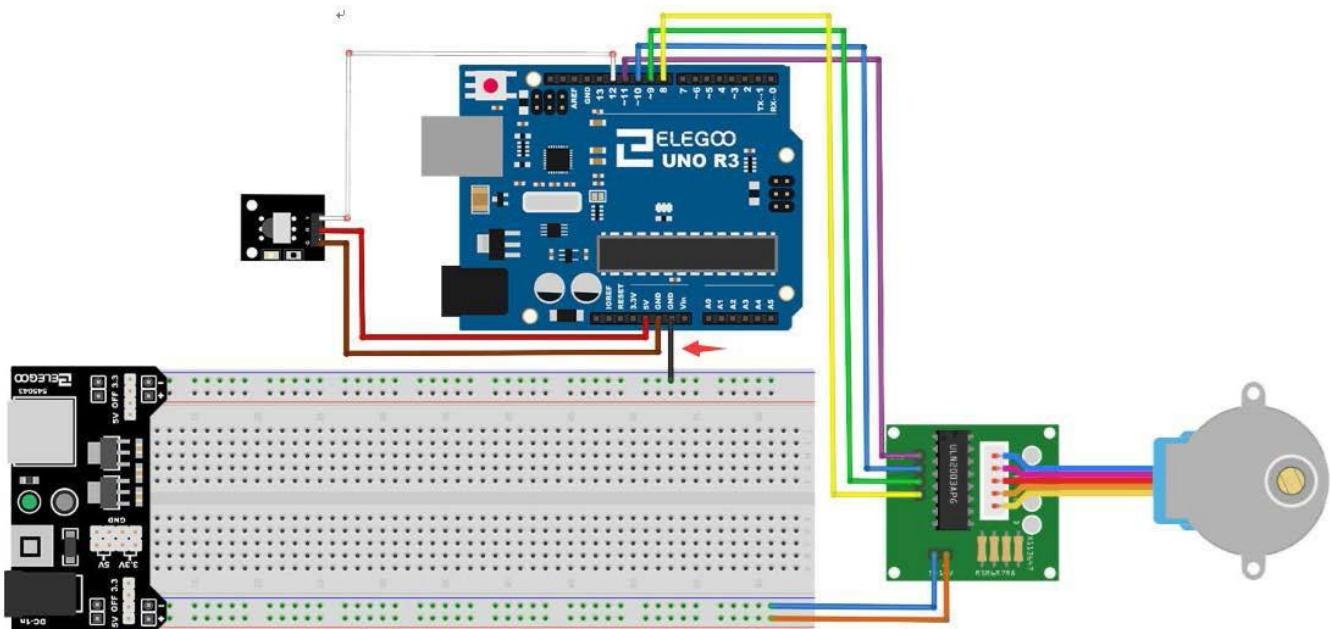


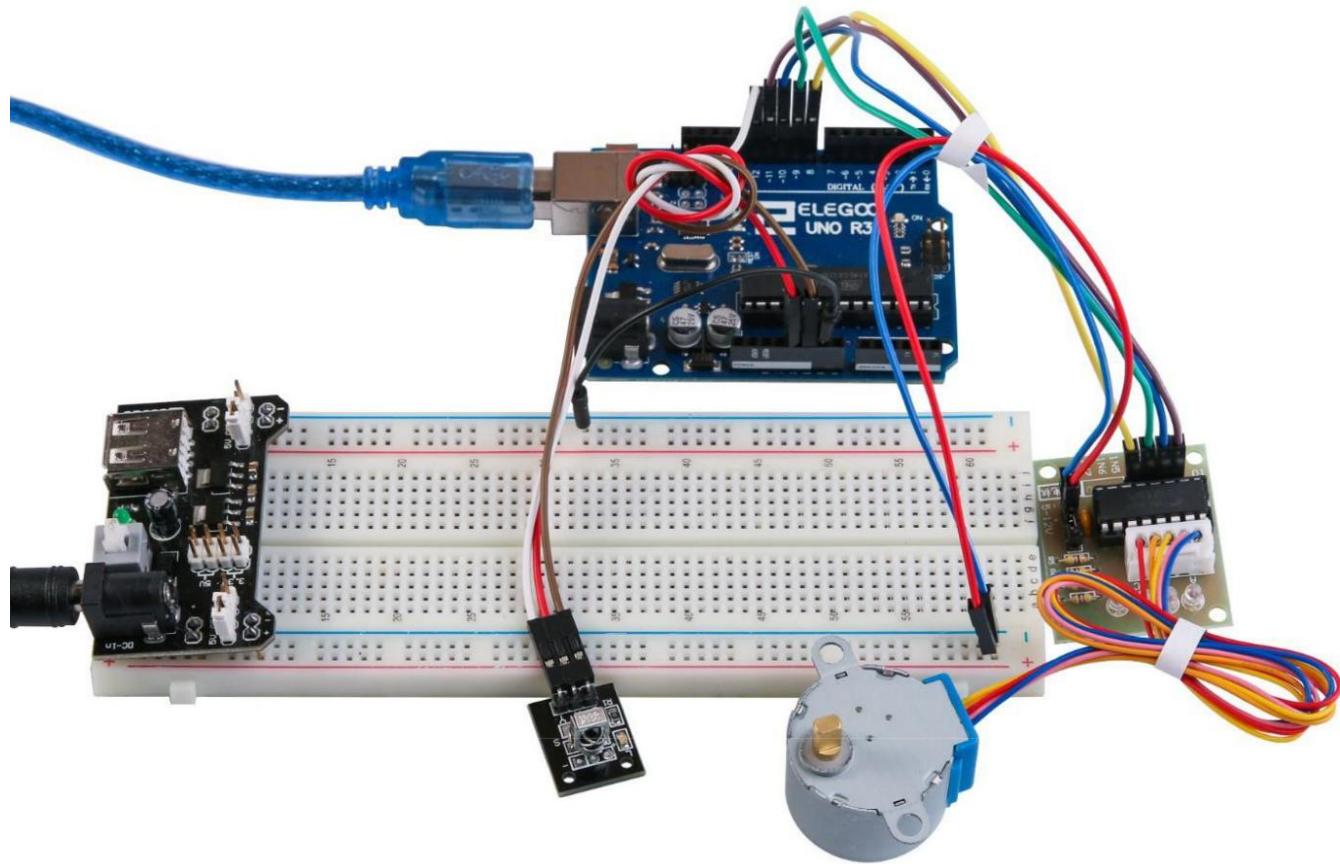
Diagrama de cableado



Estamos utilizando 4 pines para controlar el paso a paso y el 1 pin del sensor IR.

- Los pines 8-11 controlan el motor paso a paso
- El pin 12 recibe la información de IR.

Conectamos los 5V y la tierra al sensor. Como medida de precaución, usar un protoboard alimentación potencia el motor paso a paso ya que puede utilizar más energía y no queremos dañar la fuente de alimentación del Arduino.



Mando

El código reconoce sólo 2 valores desde el control remoto IR: VOL + y VOL-.

- Presionando VOL + del control remoto el motor hará un giro completo hacia la derecha.
- VOL- para hacer una rotación completa en sentido antihorario.

#Ejemplo 1

Este código hace que el motor gire en sentido horario y antihorario.

```
void setup()
{
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop()
{
```

```
//Gira el motor en sentido horario
digitalWrite(8, HIGH);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
digitalWrite(11, LOW);
delay(1000);

digitalWrite(8, LOW);
digitalWrite(9, HIGH);
digitalWrite(10, LOW);
digitalWrite(11, LOW);
delay(1000);

digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, HIGH);
digitalWrite(11, LOW);
delay(1000);

digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
digitalWrite(11, HIGH);
delay(1000);

//Gira el motor en sentido antihorario
digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
digitalWrite(11, HIGH);
delay(1000);

digitalWrite(8, LOW);
digitalWrite(9, LOW);
digitalWrite(10, HIGH);
digitalWrite(11, LOW);
delay(1000);

digitalWrite(8, LOW);
digitalWrite(9, HIGH);
digitalWrite(10, LOW);
digitalWrite(11, LOW);
delay(1000);

digitalWrite(8, HIGH);
digitalWrite(9, LOW);
digitalWrite(10, LOW);
digitalWrite(11, LOW);
delay(1000);

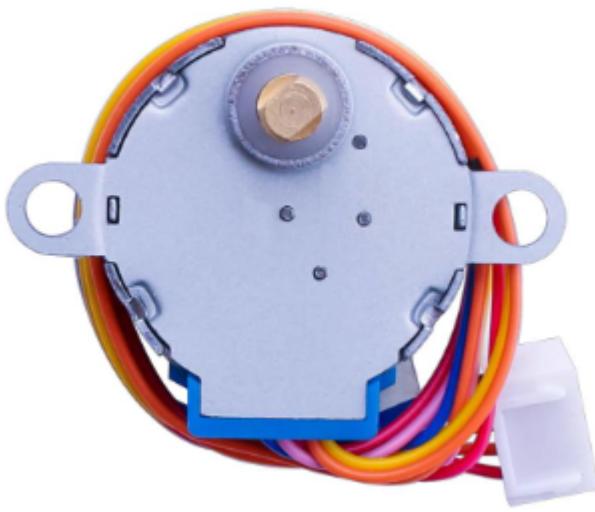
}
```

Motor paso a paso

Un motor paso a paso es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos discretos.

Componentes necesarios

- (1) x Elegoo Uno R3
- (1) x 830 tie-points breadboard
- x módulo de controlador de motor paso a paso de x ULN2003
- x Motor paso a paso
- x 9V1A Adaptador
- x Power supply module
- x F-M cables (cables de hembra a macho DuPont)
- M-M wire (hilo puente de macho a macho)



¿Cómo funciona un motor paso a paso?

El eje o eje de un motor paso a paso gira en incrementos discretos cuando impulsos de mando eléctrico se aplican a él en la secuencia correcta. La rotación de los motores tiene varias relaciones directas a estos pulsos de entrada aplicadas. La secuencia de los pulsos aplicados se relaciona directamente con la dirección de rotación de ejes motor. La velocidad de la rotación de los ejes motor está directamente relacionada con la frecuencia de los pulsos de entrada y la duración de la rotación está directamente relacionada con el número de pulsos de entrada aplicada. Una de las ventajas más importantes de un motor paso a paso es su capacidad para ser controlado con precisión en un sistema de lazo abierto. Control de lazo abierto significa que ninguna información de retroalimentación de posición es necesaria. Este tipo de control elimina la necesidad de costosos dispositivos de detección y regeneración como codificadores ópticos. Su posición es conocida simplemente por hacer el seguimiento de los pulsos de entrada de paso.

28BYJ-48 los parámetros del motor paso a paso

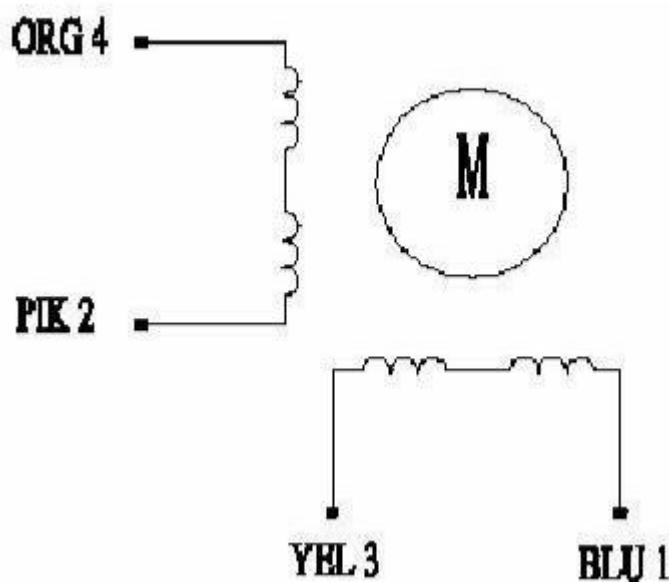
Modelo: 28BYJ-48

- Tensión nominal: 5 VDC
- Número de fase: 4
- Cociente de la variación de velocidad: 1/64
- Ángulo de paso: 5,625 ° 64

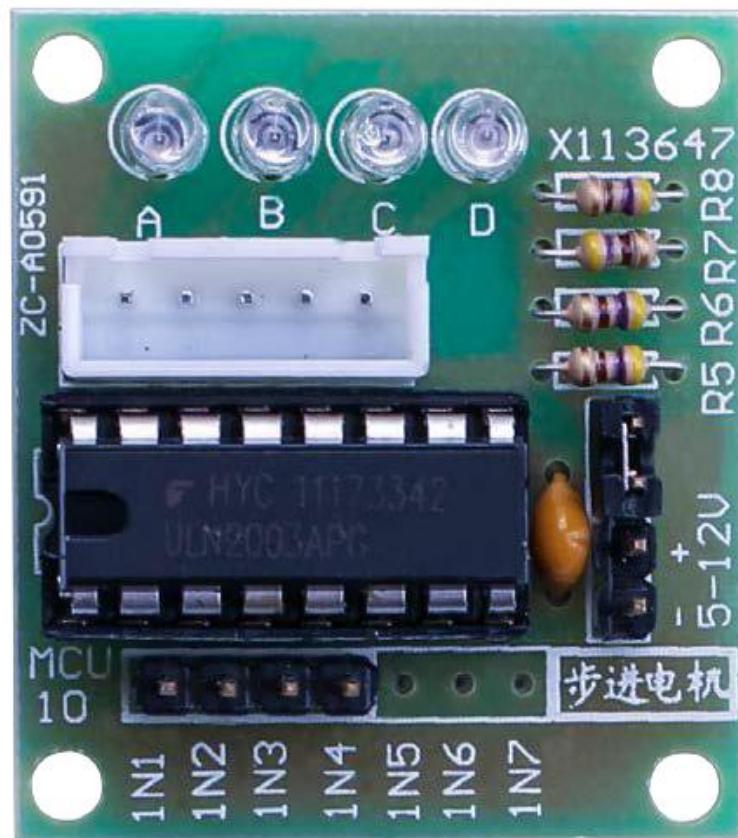
- Frecuencia: 100Hz
- Resistencia de la C.C.: $50\Omega \pm 7\%$ (25 ° C)
- Inactivo en tracción frecuencia: > 600Hz
- Frecuencia ociosa de hacia fuera-tracción: > 1000Hz En tracción par > 34.3mN.m(120Hz)
- Posicionamiento automático par > 34.3mN.m Par de fricción: 600-1200 gf.cm
- Tire un par: 300 gf.cm
- Resistencia de aislamiento > 10MΩ(500V) Aislantes de electricidad : 600VAC/1mA/1s Grado de aislamiento : A
- Subida de temperatura < 40K(120Hz) Ruido < 35dB (120Hz, No carga, 10cm

#Esquema de circuitos

WIRING DIAGRAM



El motor de pasos bipolar tiene generalmente cuatro cables que salen de él. A diferencia de los motores PAP unipolares, steppers bipolares no tienen ninguna conexión común de centro. Tienen dos juegos independientes de bobinas en lugar de otro. Se pueden distinguir de steppers unipolares midiendo la resistencia entre los cables. Debe encontrar dos pares de cables de igual resistencia. Si tienes las puntas de su medidor conectado a dos cables que no están conectados (es decir, no conectada a la bobina del mismo), debería ver resistencia infinita (o sin continuidad). #ULN2003 Placa conductora



Descripción del producto

- Tamaño: 42mmx30mm
- Chip de controlador de uso ULN2003, 500mA
- A. B. C. D LED que indica las cuatro fases las condiciones de trabajo motor paso a paso.
- Blanco jack es el conector estándar motor cuatro fase paso a paso.
- Pines de alimentación son separados
- Mantuvimos las clavijas del resto de la viruta del ULN2003 para sus prototipos más.

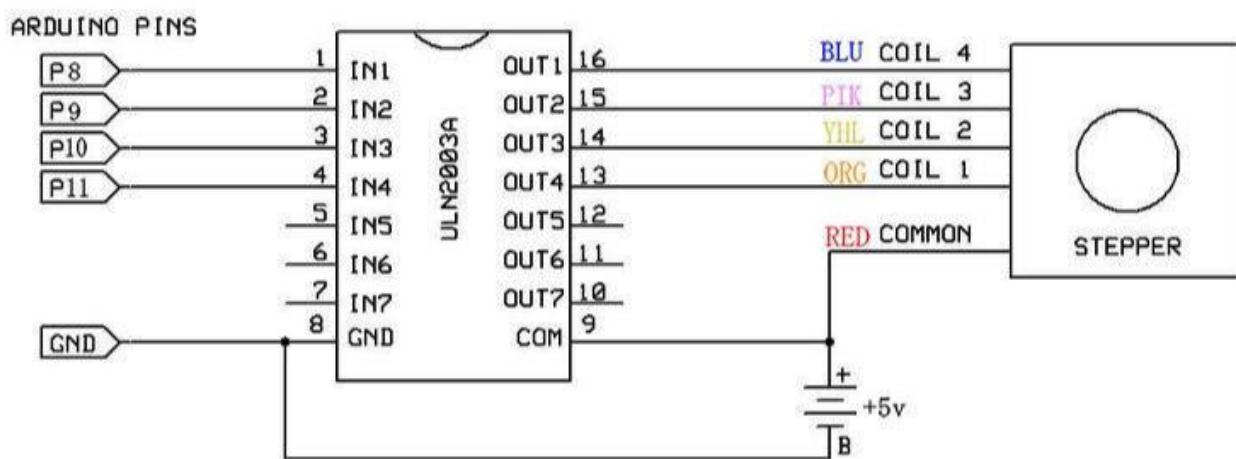
La forma más sencilla de conexión un paso a paso unipolar a [Arduino](#) es utilizar un desglose para chip de ULN2003A transistor array. El ULN2003A contiene siete controladores de transistor Darlington y es algo así como tener siete transistores TIP120 todo en un paquete. El ULN2003A puede pasar hasta 500 mA por canal y tiene una caída de tensión interna de 1V cuando en. También contiene diodos de abrazadera interna para disipar las puntas de tensión al manejar cargas inductivas.

Para controlar el paso a paso, aplicamos tensión a cada una de las bobinas en una secuencia específica.

La secuencia iría así:

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 ORG	-	-						-
3 YEL		-	-	-				
2 PIK				-	-	-		
1 BLU						-	-	-

Estos son esquemas que muestran cómo un paso a paso unipolar de interfaz motor a cuatro pins controlador utilizando un ULN2003A y mostrando cómo la interfaz usando cuatro com



#Conexión

#Esquema

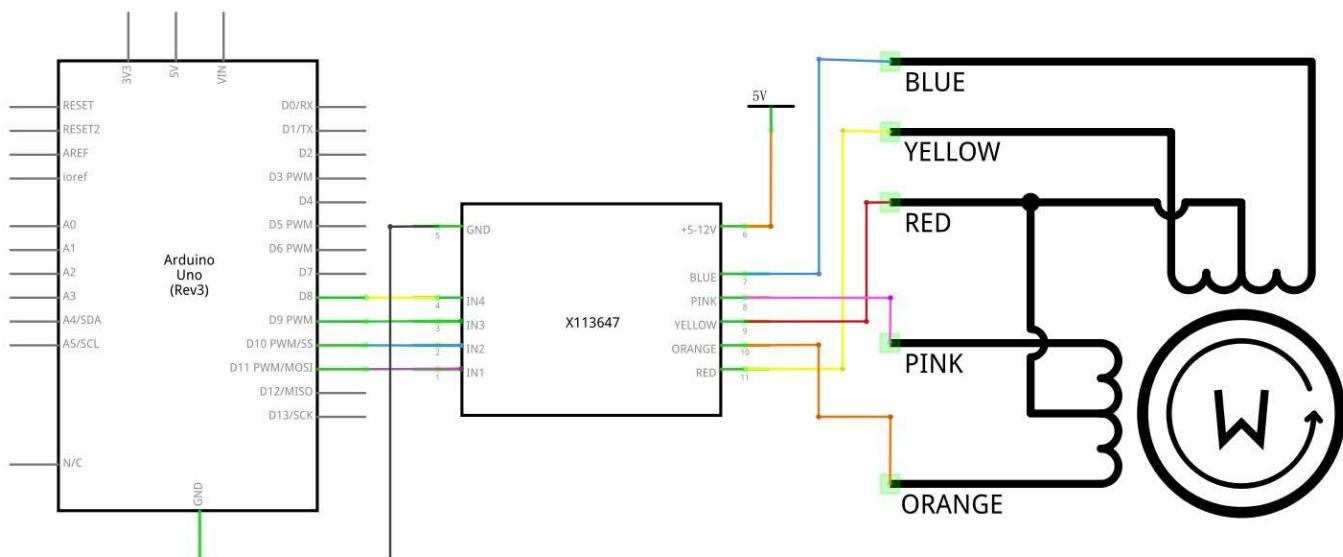
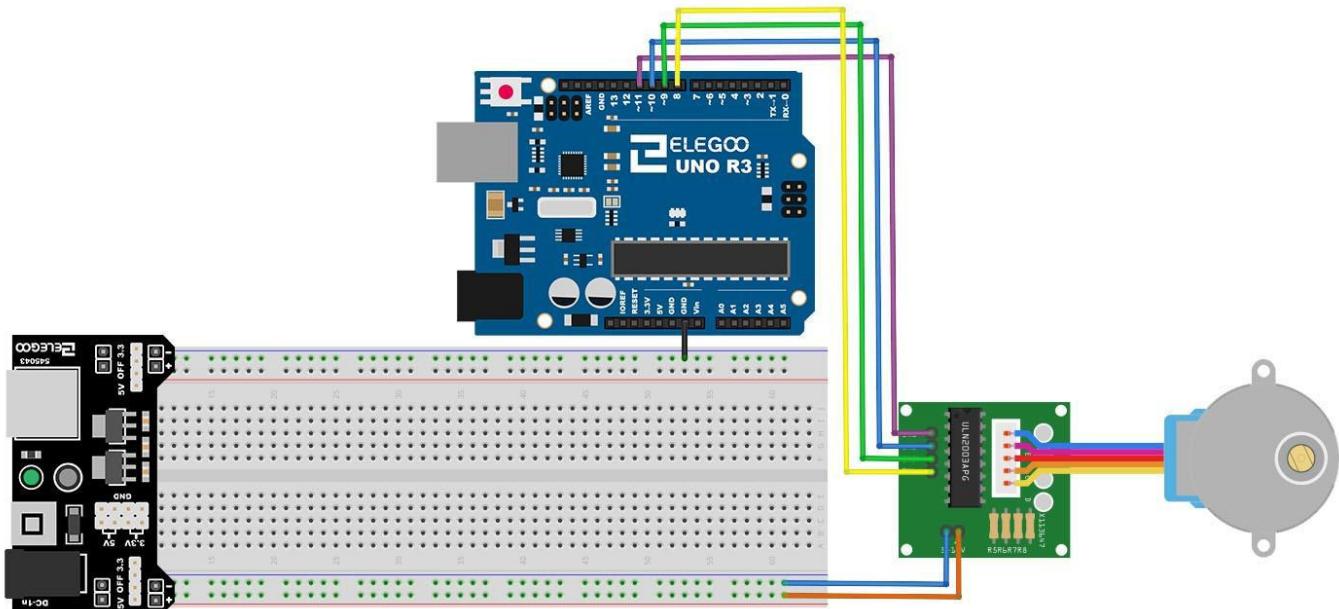


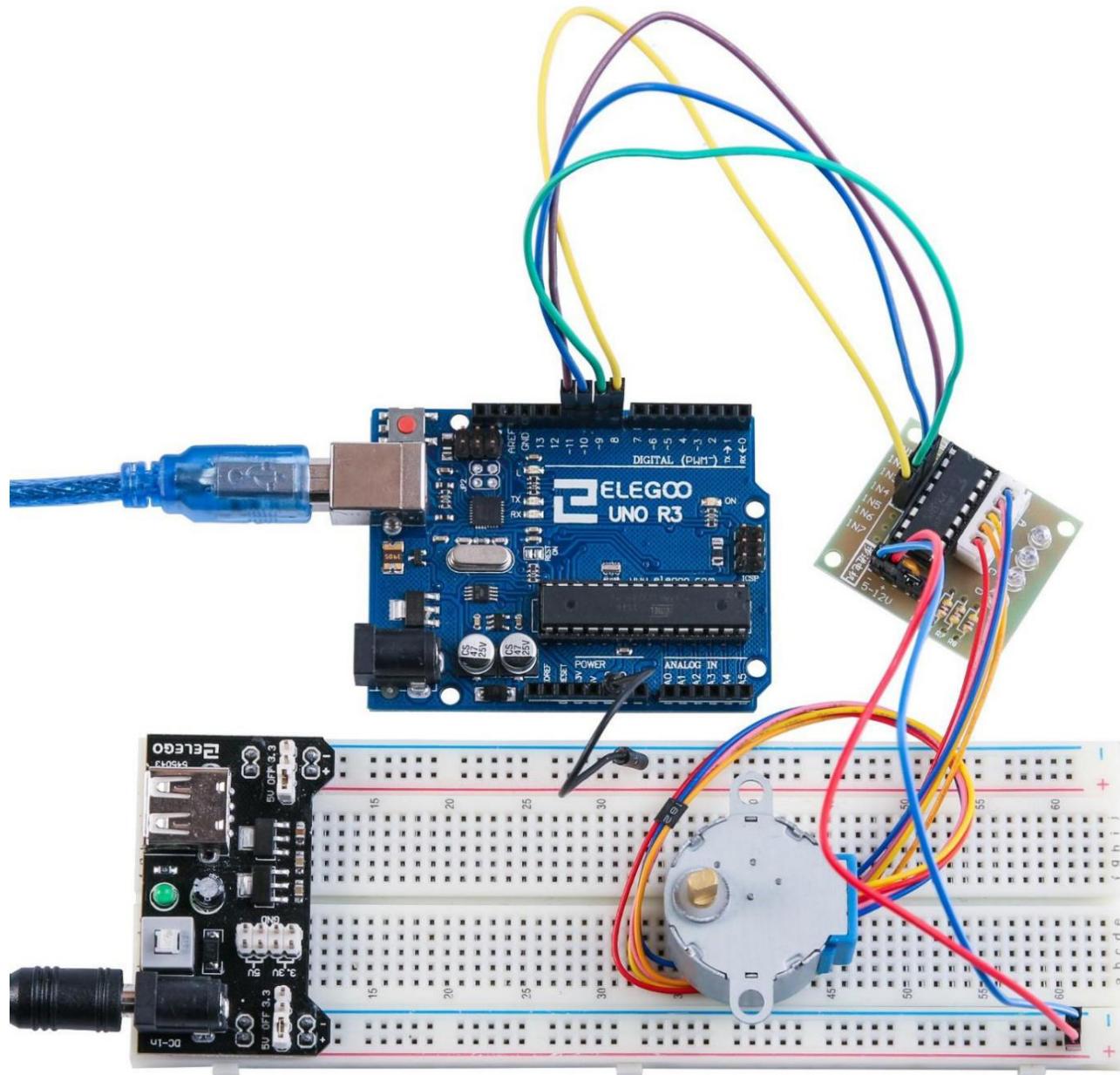
Diagrama de cableado



Estamos utilizando 4 pines para controlar el paso a paso.

- Los pines 8-11 controlan el motor paso a paso.
- Conectamos la tierra de a UNO para el motor paso a paso.

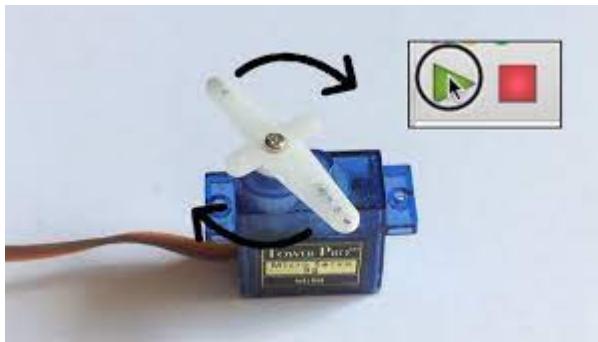
#Código



Motor servo

Los **servos** son un tipo especial de motor de c.c. que se caracterizan por su capacidad para posicionarse de forma inmediata en cualquier posición dentro de su intervalo de operación. Se mueven en una precisión de 180° como máximo.

El servo tiene un **eje** que puede girar y que es accionado por un motor. La posición del eje puede ser controlada con una señal analógica.



Para ello, el servomotor espera un tren de pulsos que se corresponde con el movimiento a realizar.

Cables

El Servo tiene tres cables:

- **Marrón** es el cable a tierra y deben conectarse a GND puerto UNO, el r
- **Rojo** es el cable de corriente y debe conectarse al puerto de 5v
- **Naranja** es el cable de señal y debe conectarse al puerto 9.

Servo MG995

El servomotor **MG995** es un servomotor digital de alta velocidad y alta precisión. El servomotor está construido con una carcasa de plástico reforzado y un eje de metal.



Tiene una salida de 5 V y una corriente máxima de 2 A. Puede alcanzar una velocidad de giro de 0,12 segundos por vuelta.

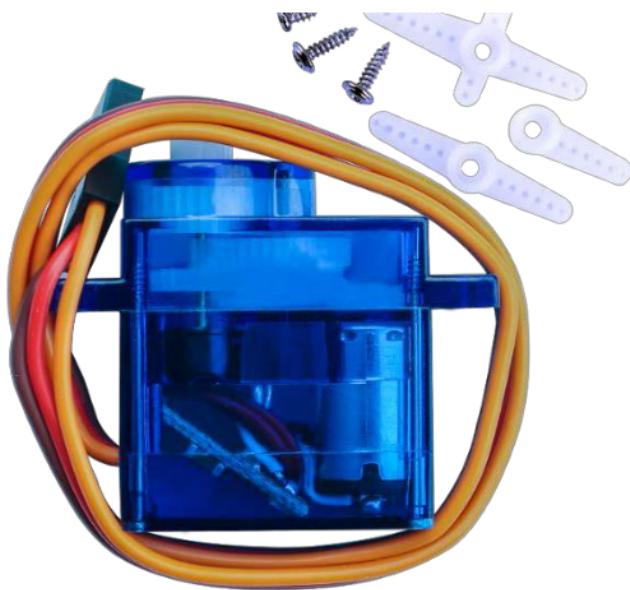
Servomotor SG90

- El **SG90** es un microservo más pequeño y ligero que el MG995.
- También es más económico.
- El **SG90** tiene un rango de movimiento de aproximadamente 180 grados, mientras que el MG995 tiene un rango de movimiento de aproximadamente 360 grados.

| Parámetro | Valor | | ----- || | Longitud del cable: | 25cm | | Sin carga; | Velocidad: 0,12 seg/60 degree (4.8V), 0.10 sec/60 grados (6.0V) | | Puesto de par (4.8V): | 1,6 kg/cm | | Temperatura: | -30 ~ 60° C | | Ancho de banda muerta: | 5 us | | Voltaje de funcionamiento: | 3.5 ~ 6V | | Dimensión: | 1.26 en x 1,18 en x 0,47 en (3,2 x 3 cm x 1.2 cm) | | Peso: | 4,73 onzas (134) |

Accesories

The servo comes with different accessories that can be used to mount it to other parts.



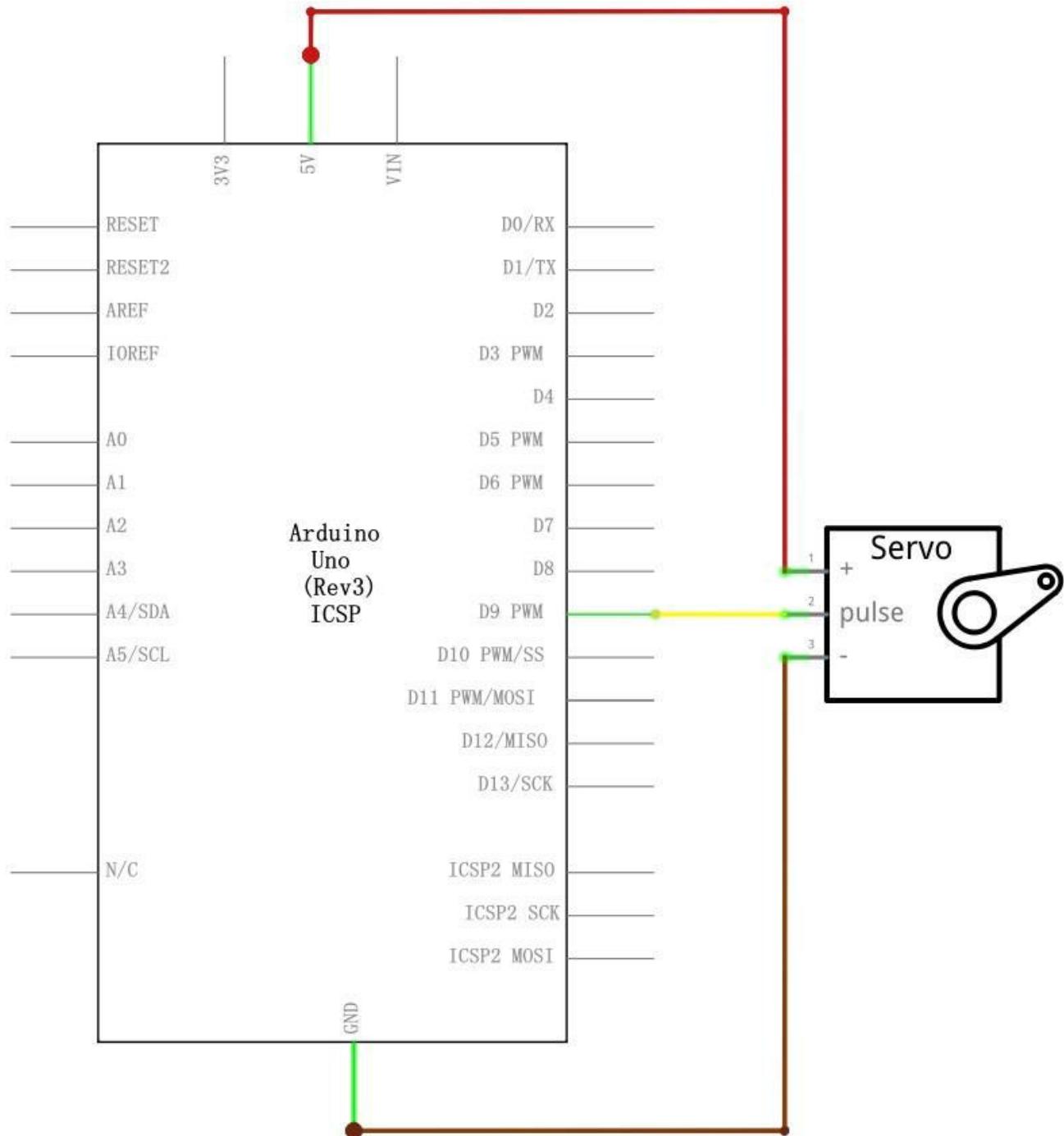
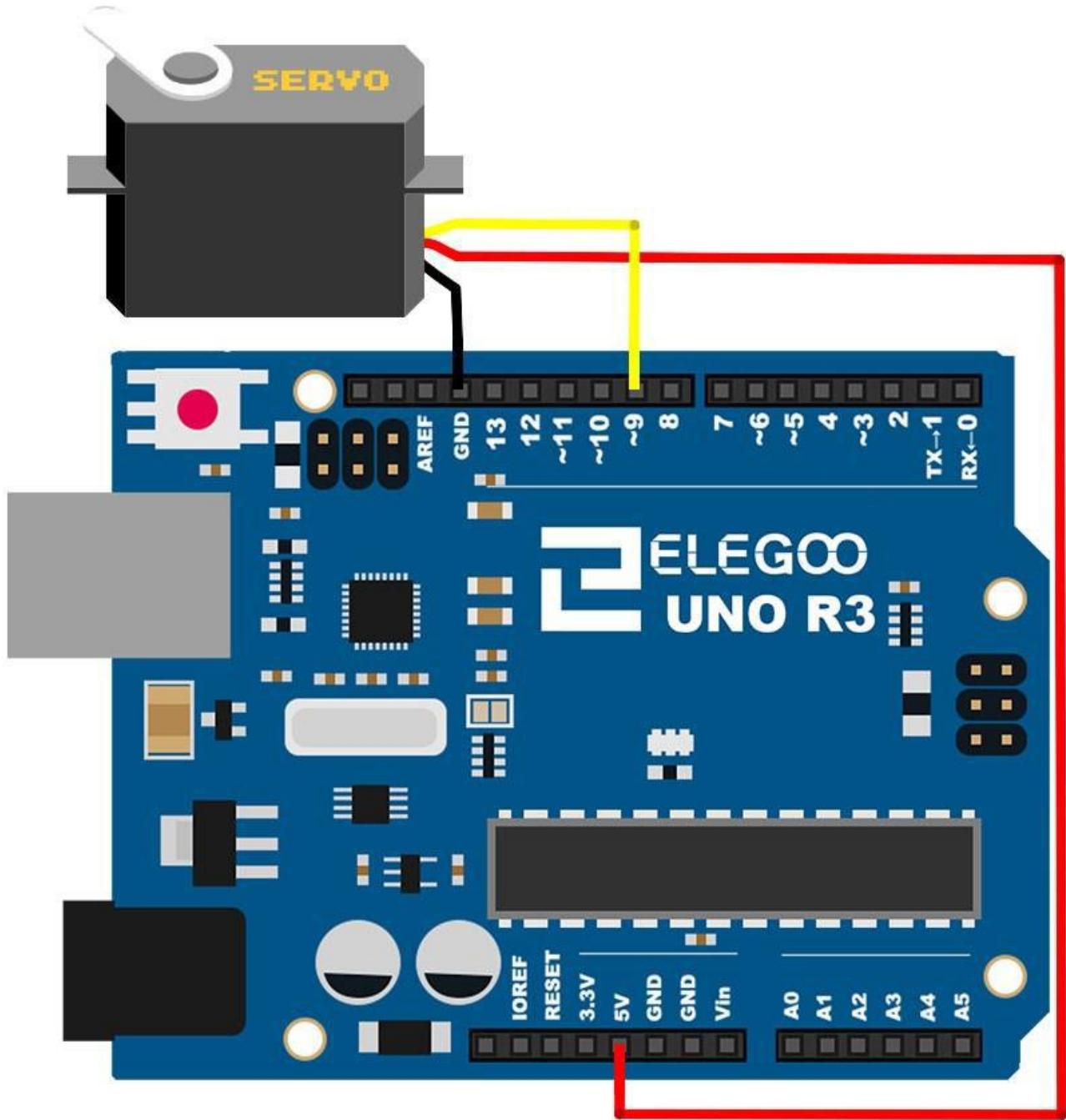
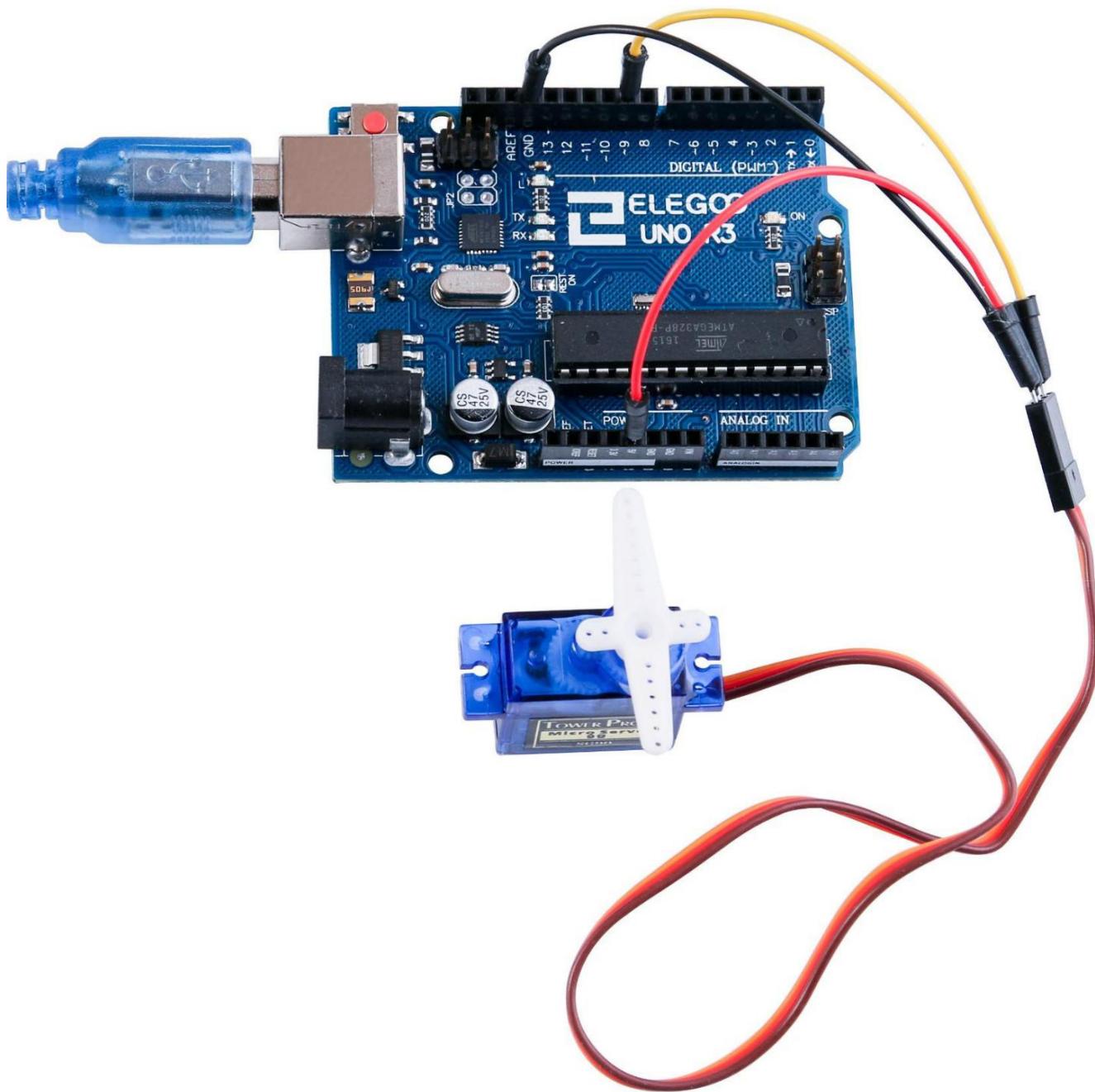


Diagrama de cableado



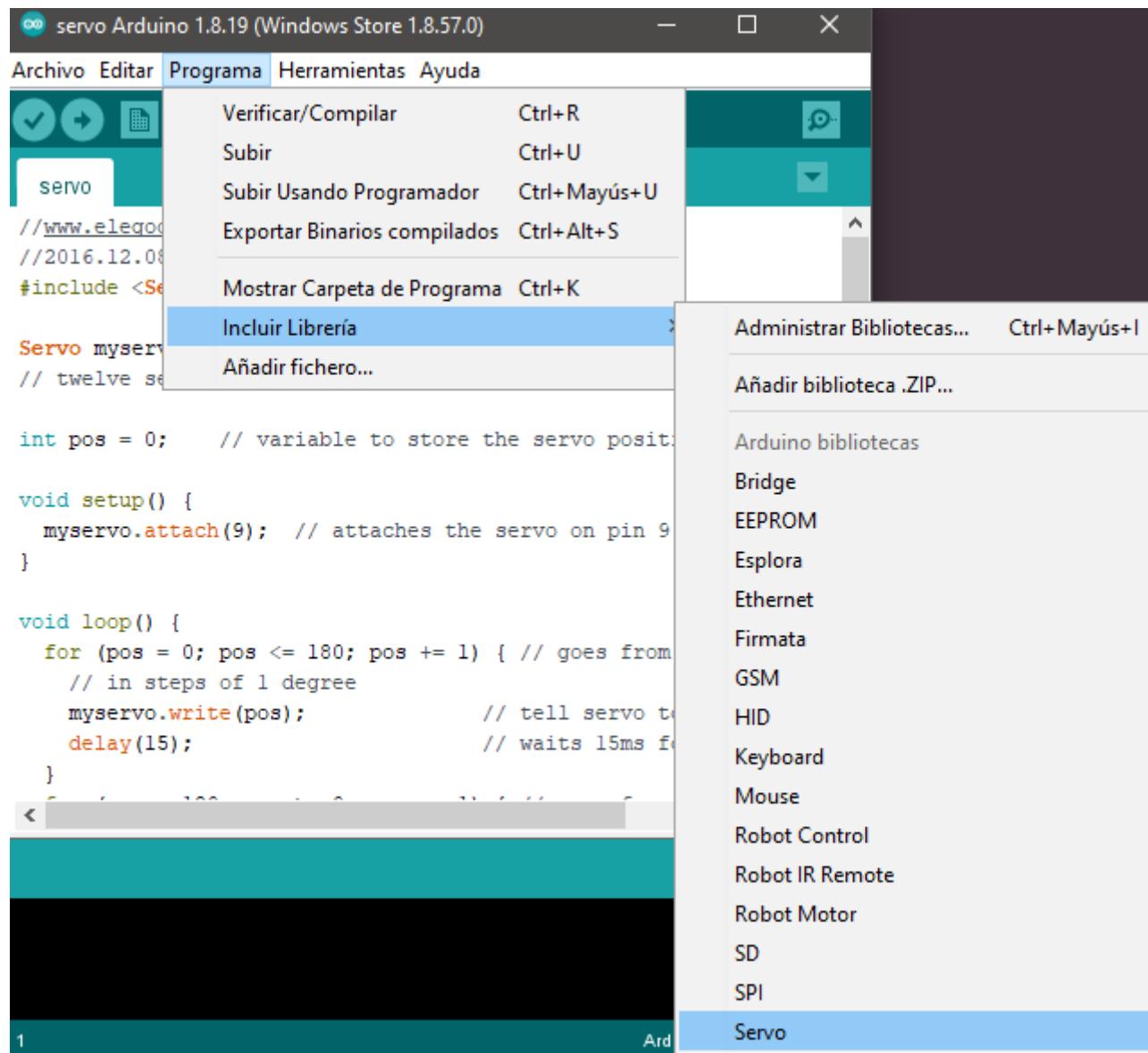
#Montaje

Necesitaremos 3 **jumpers** para conectar el servo a la placa.



#Código

Antes de ejecutar esto, debemos incluir la **biblioteca servo**. Esta librería incorpora funciones que nos permitirán manejar de forma más sencilla el comportamiento del motor.



Ejemplo 1

```
#include <Servo.h> // Incluimos la librería Servo

Servo miServo; // Creamos un objeto Servo

void setup() {
    miServo.attach(9); // Conectamos el servo al pin 9
}

void loop() {
    miServo.write(90); // Movemos el servo a 90 grados
    delay(1000); // Esperamos 1 segundo
    miServo.write(0); // Movemos el servo a 0 grados
    delay(1000); // Esperamos 1 segundo
}
```

Ejemplo 2 Este código mueve el eje del motor 180 grados en una dirección y luego en la contraria, indefinidamente.

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0; // variable to store the servo position

void setup() {
    myservo.attach(9); // Le asignamos el PIN 9.
}

void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
        // in steps of 1 degree
        myservo.write(pos); // tell servo to go to position in variable
        'pos'
        delay(15); // waits 15ms for the servo to reach the
        position
    }
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
        myservo.write(pos); // tell servo to go to position in variable
        'pos'
        delay(15); // waits 15ms for the servo to reach the
        position
    }
}
```

Luces

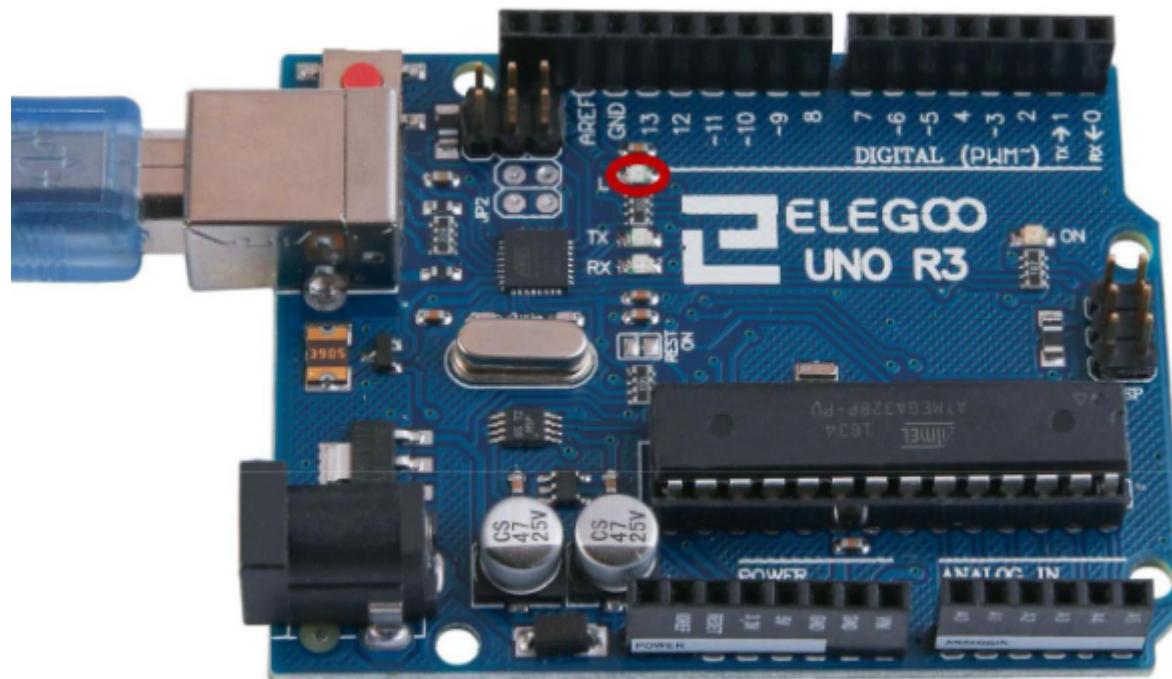
Els LEDs, els LCD i els displays de set segments són components electrònics que es poden controlar amb una placa Arduino per a mostrar informació o per a indicar l'estat d'un sistema. Cada un d'aquests dispositius té diferents característiques i funcions, i es poden utilitzar en diferents aplicacions.

- Els **LEDs** són diodes emissors de llum que es poden utilitzar per a indicar l'estat d'un sistema, per a il·luminar objectes o per a crear efectes de llum.
- Els **displays LCD** són pantalles de cristall líquid que es poden utilitzar per a mostrar text o gràfics. Aquests displays són útils en moltes aplicacions, com ara en sistemes de control de temperatura, temporitzadors, termostats, etc.
- Els **displays de set segments** són displays que mostren dígits o lletres utilitzant set segments de LEDs per a crear la forma desitjada. Aquests displays són utilitzats en moltes aplicacions, com ara rellotges digitals, termòmetres, indicadors de nivell, etc.

LED interno

En esta lección, haremos parpadear el **LED integrado de Arduino**. Para ello únicamente necesitaremos la placa **Arduino Uno R3**.

La placa de UNO R3 tiene unas filas de conectores a ambos lados que se utilizan para conectar varios dispositivos electrónicos y **shields** que amplían su capacidad.



LED integrado

También tiene un **LED** luminoso podemos controlar. Este LED está construida sobre la placa.

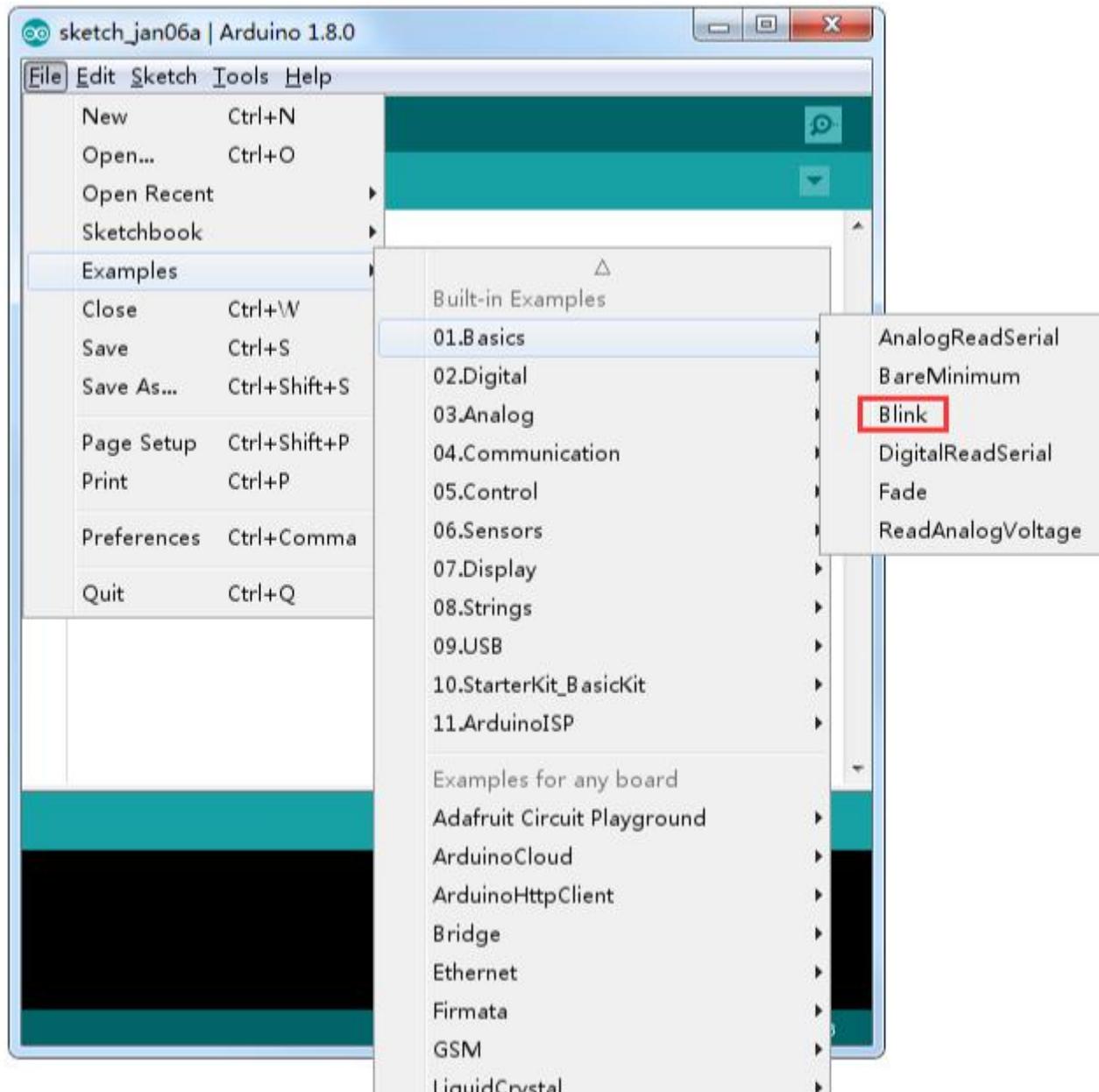
Este LED **parpadea** cuando se conecta a un enchufe del USB. Esto es porque las placas se envían generalmente con un programa llamado **Blink** pre-instalado.

Ejemplos

El IDE de **Arduino** incluye una gran colección de programas de ejemplo para utilizar directamente. Esto incluye un ejemplo para hacer el parpadeo del **LED**.

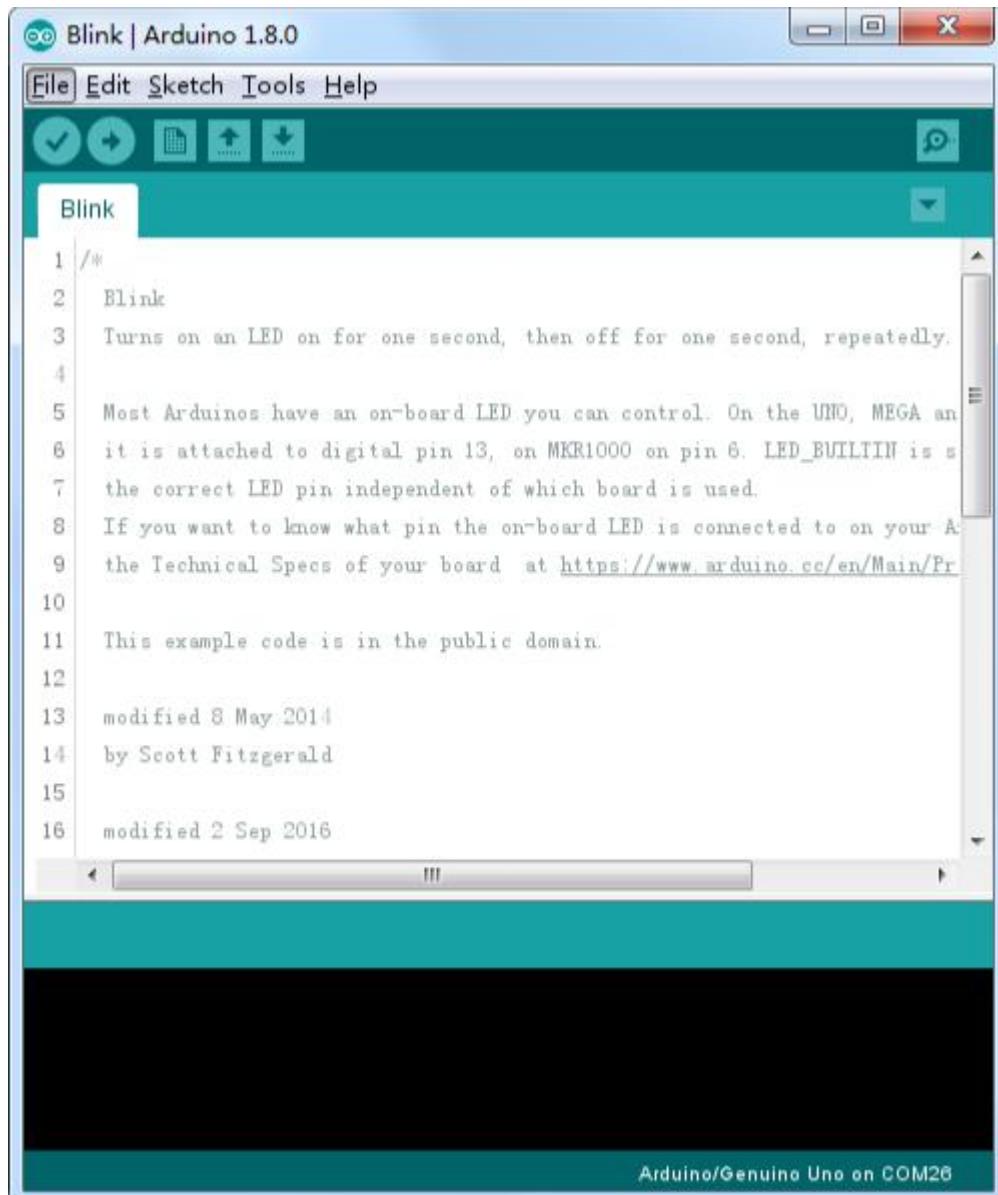
Ejemplo programa Blink

Cargar el programa de 'Blink' que encontrarás en el sistema de menús del IDE bajo **archivo > ejemplos > 01 conceptos básicos**



Blink

Cuando se abre la ventana de dibujo, agrandarla para que puedan ver el dibujo completo en la ventana.

A screenshot of the Arduino IDE version 1.8.0. The window title is "Blink | Arduino 1.8.0". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Print. The main area shows the "Blink" sketch code. The code is as follows:

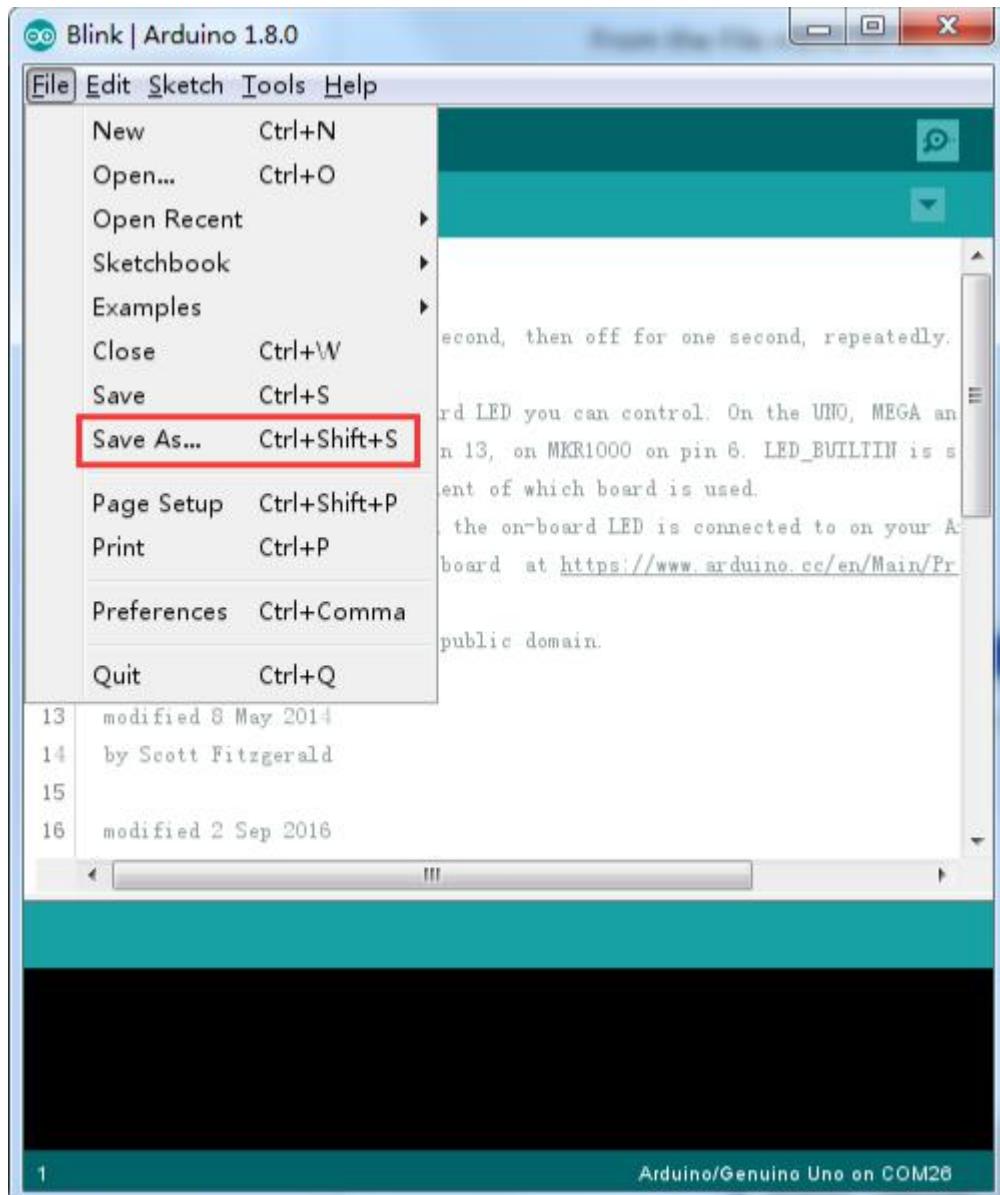
```
1 /*
2  * Blink
3  * Turns on an LED on for one second, then off for one second, repeatedly.
4  *
5  * Most Arduinos have an on-board LED you can control. On the UNO, MEGA and
6  * MKR1000 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set
7  * to the correct LED pin independent of which board is used.
8  * If you want to know what pin the on-board LED is connected to on your Arduino
9  * or on an Arduino-like board, refer to the Technical Specs of your board at https://www.arduino.cc/en/Main/Products
10 */
11
12 This example code is in the public domain.
13
14 modified 8 May 2014
15 by Scott Fitzgerald
16 modified 2 Sep 2016
```

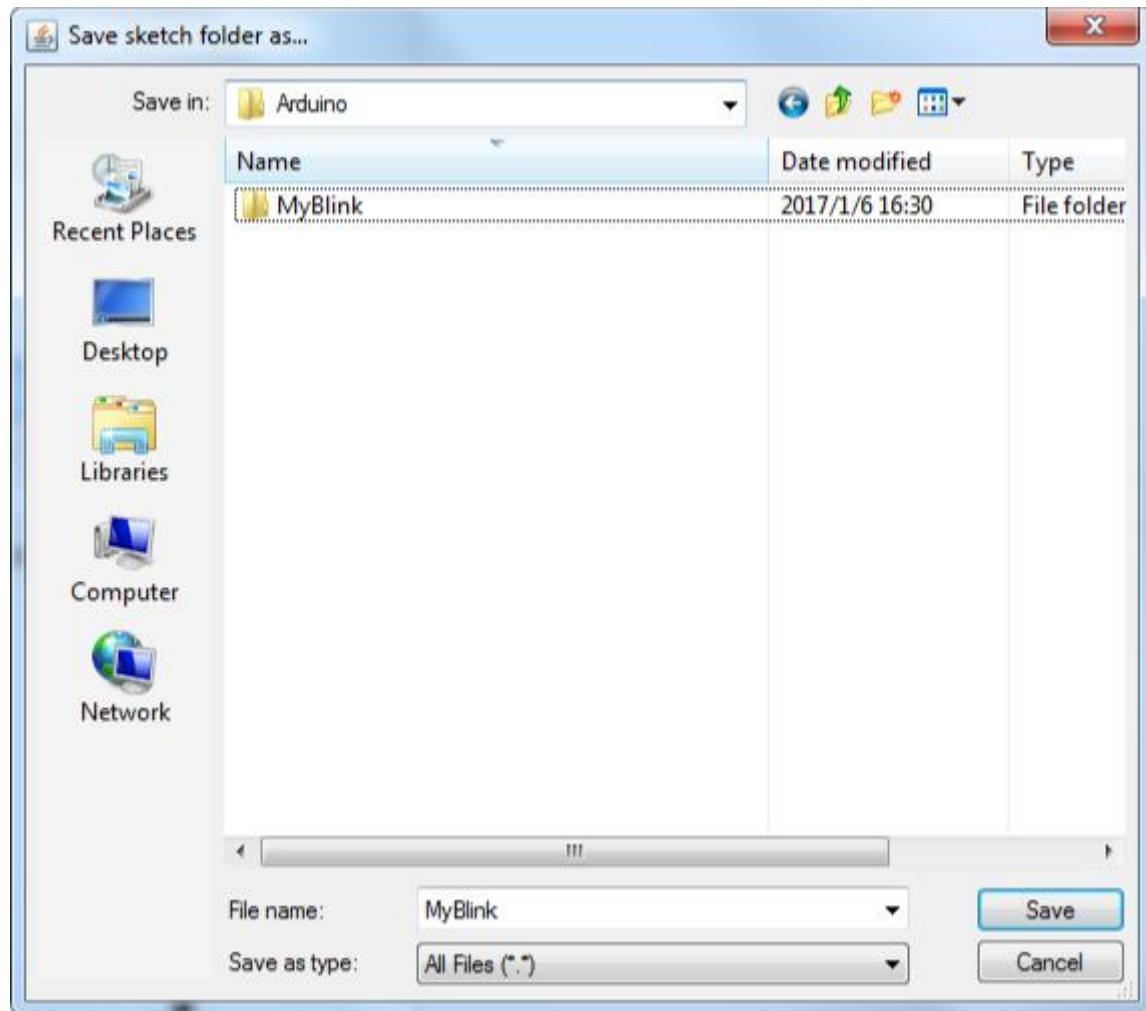
The status bar at the bottom right shows "Arduino/Genuino Uno on COM26".

Los **programas de ejemplo** incluidos con el IDE de **Arduino** son de 'sólo lectura'. Es decir, puedes subirlo a Arduino, pero no se pueden guardar una vez modificados.

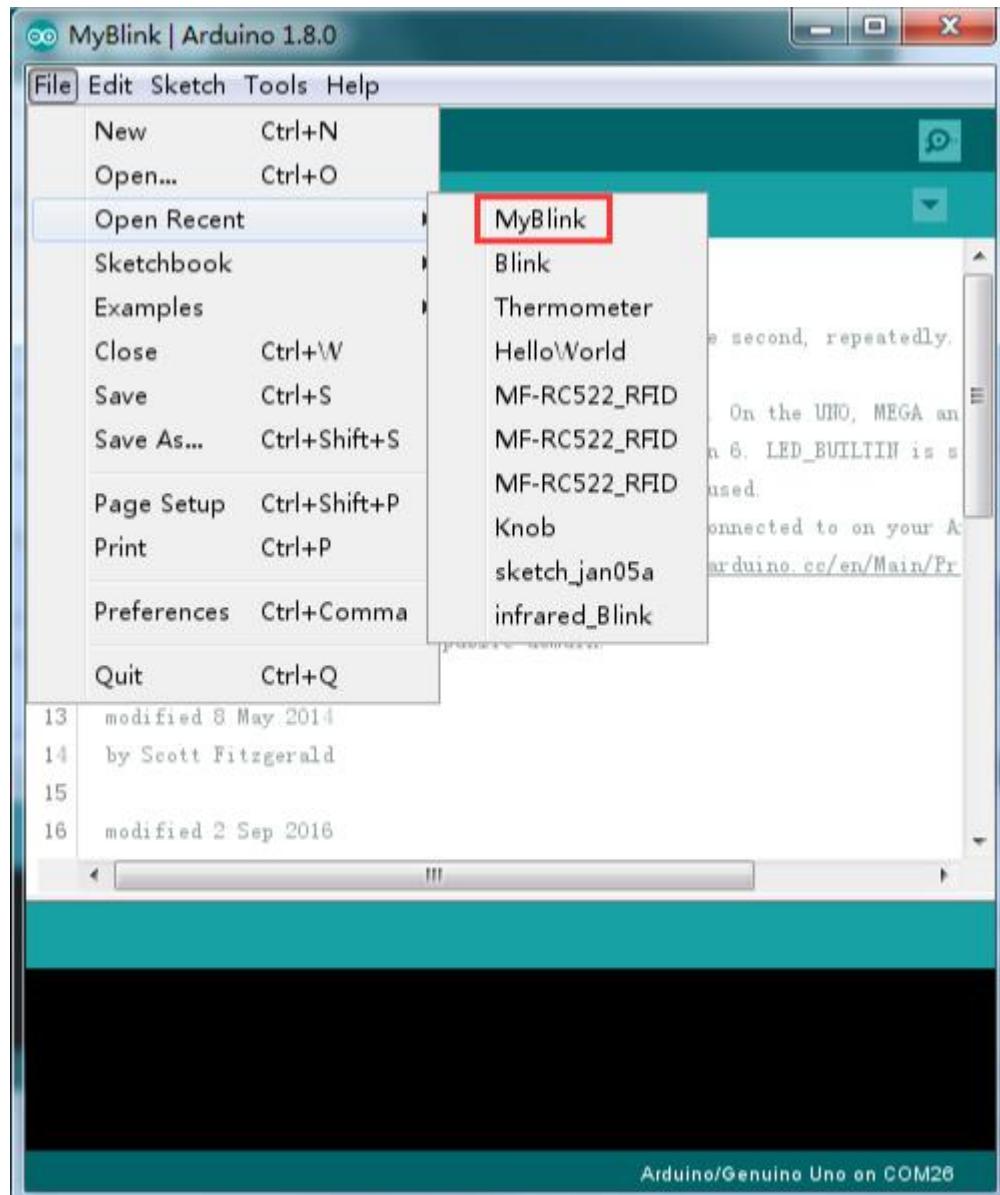
Guardar código en otro archivo

En el menú archivo en el IDE de **Arduino**, seleccione **Guardar como...** y guarde el dibujo con el nombre **parpadeo**





Para abrir un archivo que hemos guardado con anterioridad, podemos simplemente ir a [archivo > abrir](#) o también a [archivo > abrir reciente](#).



Conecitar placa al PC

Conecte la placa de **Arduino** al ordenador con el cable USB y compruebe que la **Board Type** y **Puerto serie** están ajustados correctamente.



MyBlink | Arduino 1.8.0

File Edit Sketch Tools Help

MyBlink

```
1 /*  
2  Blink  
3  Turns on an  
4  Most Arduino  
5  it is attach  
6  the correct  
7  If you want  
8  the Technic  
9  
10 This example  
11  
12  
13 modified 8 May 2014  
14 by Scott Fitzgerald  
15  
16 modified 2 Sep 2016
```

Board: "Arduino/Genuino Uno"

Port: "COM26 (Arduino/Genuino Uno)"

Get Board Info

Programmer: "AVRISP mkII"

Burn Bootloader

Boards Manager...

Arduino AVR Boards

Arduino Yún

Arduino/Genuino Uno

Arduino Duemilanove or Diecimila

Arduino Nano

Arduino/Genuino Mega or Mega 2560

Arduino Mega ADK

Arduino Leonardo

Arduino Leonardo ETH

Arduino/Genuino Micro

Arduino Esplora

Arduino Mini

Arduino Ethernet

Arduino Fio

Arduino BT

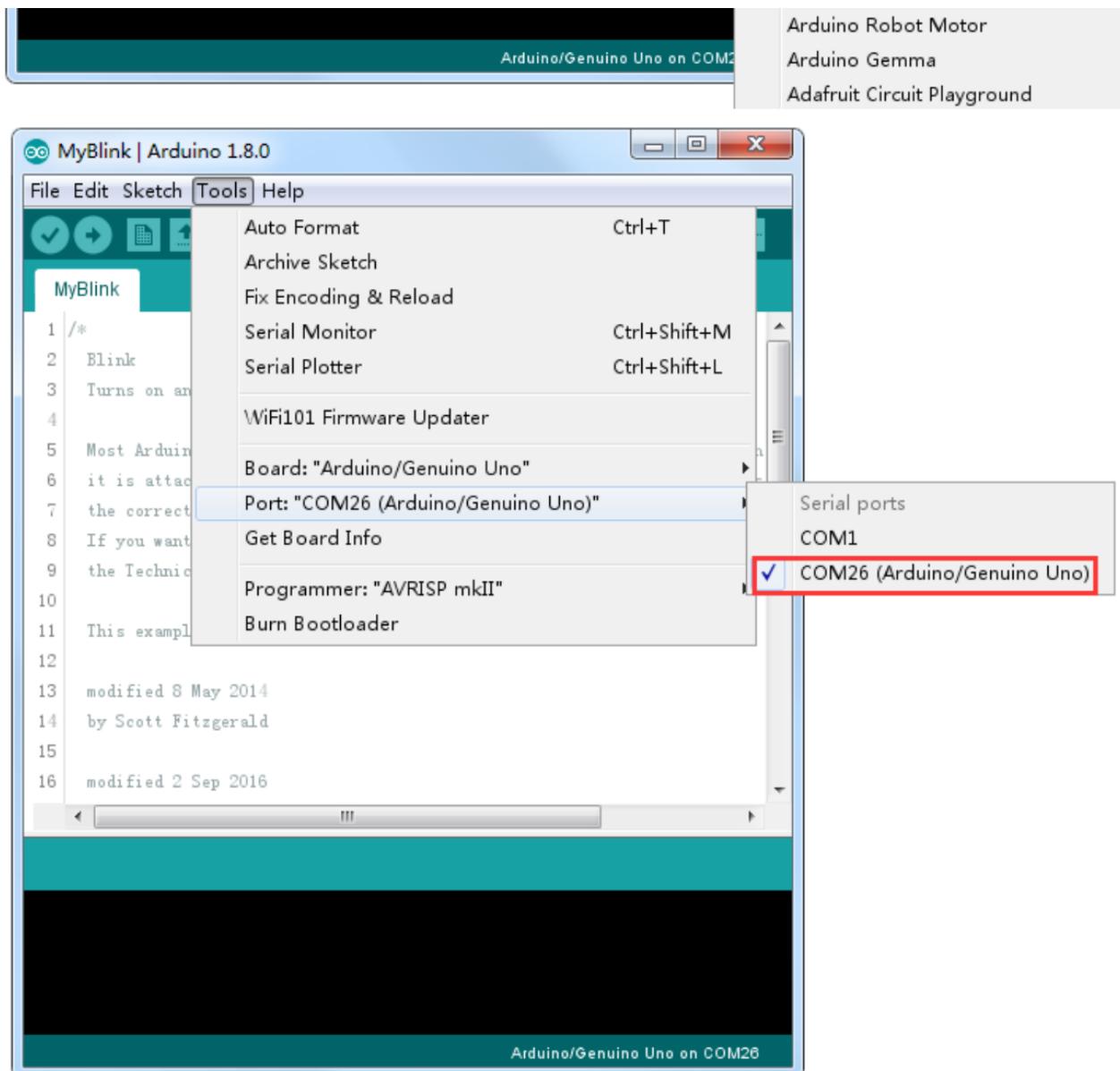
LilyPad Arduino USB

LilyPad Arduino

Arduino Pro or Pro Mini

Arduino NG or older

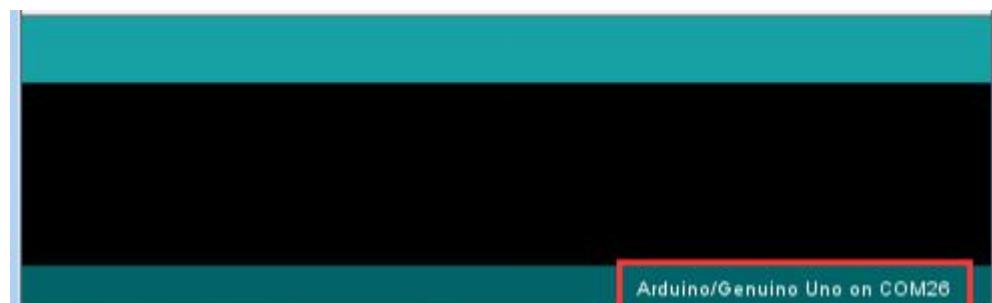
Arduino Robot Control



Nota

- El tipo de tarjeta y puerto Serial aquí no son necesariamente la misma que se muestra en la imagen.
- El puerto serie (**COM**) puede ser diferente, del tipo COM3 o COM4 en su ordenador.

El IDE de **Arduino** mostrará la configuración actual en la parte inferior de la ventana.



Subir código a Arduino

Para que Arduino lo ejecute, necesitamos enviarle a través del cable USB el código que queremos que haga.

Para ello, debemos hacer clic en el botón **subir**. El segundo botón de la izquierda en la barra de herramientas.

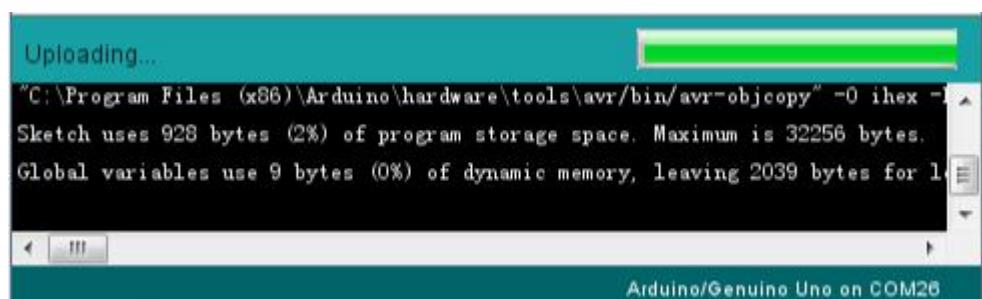


Subiendo código

Si usted mira el área de estado del IDE, verá una barra de progreso y una serie de mensajes. Al principio, que dice 'Bosquejo compilar...'. Esto convierte el dibujo en un formato adecuado para subir a la Junta.



A continuación, el estado cambiará a **subir**. En este punto, los LEDs de la **Arduino** deben comenzar a parpadear como se transfiere el dibujo.



Por último, el estado cambiará a 'Done'.



El otro mensaje nos dice que el **programa** está utilizando 928 bytes de 32.256 bytes disponibles. Después de la etapa de compilación Sketch... podría obtener el siguiente mensaje de error:



Puede significar que su Junta no está conectado a todos, o no se ha instalado los drivers (si es necesario) o que se ha seleccionado el puerto serial incorrecto.

Comprobar funcionamiento

Una vez completada la carga, la placa se debe reiniciar y el led comenzar a parpadear.

Comentarios

- Todo entre /* y */ en la parte superior del **programa** es un Comentario de bloque; explica lo que el **programa** es para.
- Los comentarios de una sola línea comienzan con // y hasta el final de esa línea se considera un comentario.

Crear variables

La primera línea de código es:

```
int led = 13;
```

Creamos una variable con un nombre y guardamos el número de pin al que el LED está conectado.

Función Setup

A continuación, tenemos la función de 'configuración'. Otra vez, como dice el comentario, este se ejecuta cuando se presiona el botón de **reset**. También se ejecuta cada vez que la Junta se reinicia por alguna razón, como poder primero se aplica a él, o después de un **programa** se ha subido

```
void setup() {  
  // Inicializa el pin digital como salida.  
  pinMode(led, OUTPUT);  
}
```

Función setup

Cada programaArduino debe tener una función de **setup** (configuración), y las instrucciones que contendrá se insertan entre las llaves { y }.

En este caso, es un comando, que, como dice el comentario dice la placaArduino que vamos a utilizar el pin LED como salida.

Función loop

También es obligatorio para un boceto tener una función de **loop**. A diferencia de la función de **setup** que se ejecuta sólo una vez, después de un reset, la función **loop**, después que haya terminado de ejecutar sus comandos, empezar inmediatamente otra vez.

Explicación loop

```
void loop() {
    digitalWrite(led, HIGH); // Encienda el LED (alto es el nivel de voltaje)
    delay(1000); // Espere un segundo
    digitalWrite(led, LOW); // Apagar el LED por lo que la tensión baja
    delay(1000); // Espere un segundo
}
```

Dentro de la función **loop**, los comandos en primer lugar activar el pin del LED (alto), girar a 'retraso' de 1000 milisegundos (1 segundo), entonces el pin LED apagado y pausa para otro segundo.

Cambiar la frecuencia de parpadeo

```
30 // the loop function runs over and over again forever
31 void loop() {
32     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the volt
33     delay(500) // wait for a second
34     digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the vo
35     delay(500) // wait for a second
36 }
```

Ahora vas a que el LED parpadee más rápido. Como puede haber adivinado, la clave de esto radica en cambiar el parámetro () para el comando **delay**.

Variar retardo

Este período de retardo en milisegundos, así que si desea que el LED parpadee dos veces tan rápidamente, cambiar el valor de 1000 a 500. Esto entonces pausa durante medio segundo cada retraso en lugar de un segundo entero.

Sube otra vez el **programa** y verás que el LED comienza a parpadear más rápidamente.

LED

Aprenderemos a cambiar el **brillo** de un LED usando diferentes valores de resistencia.

Componentes necesarios

- [x] (1) x Placa ``Arduino`` UNO
- [x] 1 LED rojo de 5 mm
- [x] (1) x resistencia de 220 ohmios
- [x] (1) x resistencia de 1 k ohm
- [x] (1) x resistencia de ohmio 10 k
- [x] (2) x M M cables (cables de puente de macho a macho)

¿Qué es un LED?

En esta lección, usarás tal vez el más común de todos los LEDs: un LED de 5mm de color rojo. 5mm se refiere al diámetro del LED. Otros tamaños comunes son 3mm y 10mm.

¿Cómo se conecta?**

Directamente no se puede conectar un LED a una batería o fuente de tensión porque:

1. El LED tiene un polo positivo y un negativo y no se encenderá si se conectan mal.
2. Un LED con una resistencia para limitar la corriente que circula a través de él.

Ejemplo de LED



Advertencia

Si no utilizas un resistencia con un LED, entonces se puede quemar casi de inmediato, como demasiada corriente fluirá a través, calienta y destruye al 'cruce' donde se produce la luz. Hay dos maneras de saber cual es el positivo del LED y que la negativa. En primer lugar, el positivo es más largo.

En segundo lugar, donde la pata del negativo entra en el cuerpo del LED, hay un borde plano para el caso del LED.

La patilla más larga es el **positivo**.

Resistencias

Como su nombre lo indica, resistencias de resisten el flujo de electricidad. Cuanto mayor sea el valor de la resistencia, resiste más y la menos corriente fluirá a través de él. Vamos a usar esto para controlar Cuánta electricidad fluye a través del LED y por lo tanto, como claramente brilla.

Resistencias: ejemplo



Resistencias: unidades

- La **unidad** de resistencia se denomina Ohm, que se abrevia generalmente a Ω la letra griega Omega.
- 1 Ohm es un valor bajo
- Valores de resistencias en $k\Omega$ (1.000 Ω) y $M\Omega$ (1.000.000 Ω). (kiloohmios y megaohmios).

Resistencias: valores

En esta lección, vamos a utilizar tres valores diferentes de resistencia:

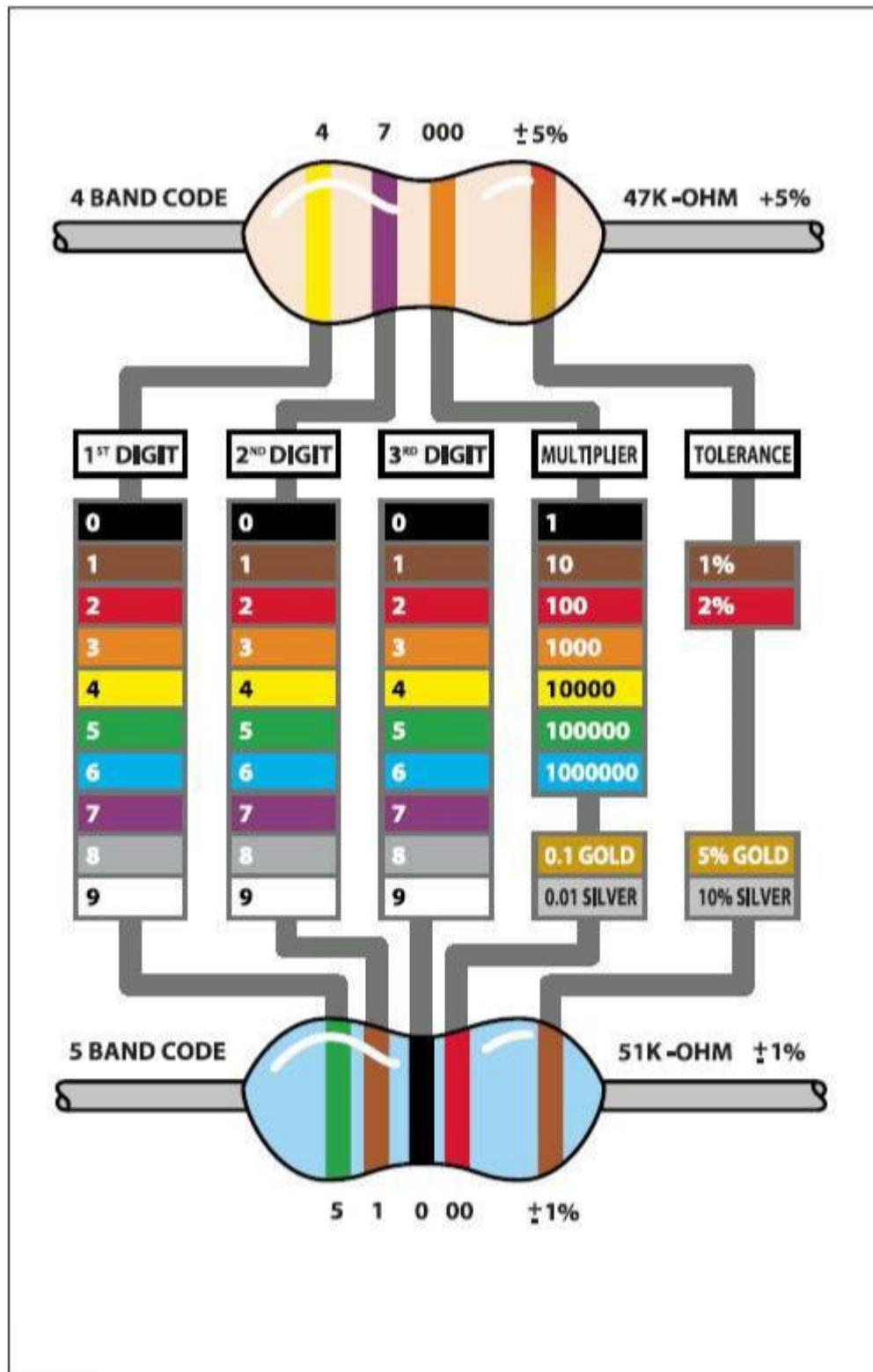
- 220 Ω
- 1 $k\Omega$
- 10 $k\Omega$

Resistencias: diferencias

Estas resistencias todas se ven iguales, excepto que tienen **rayas de colores** diferentes en ellos. Estas rayas decirte el valor de la resistencia.

El **código** de color resistor tiene tres franjas de colores y luego una banda de oro en un extremo.

Ejemplo identificación resistencia

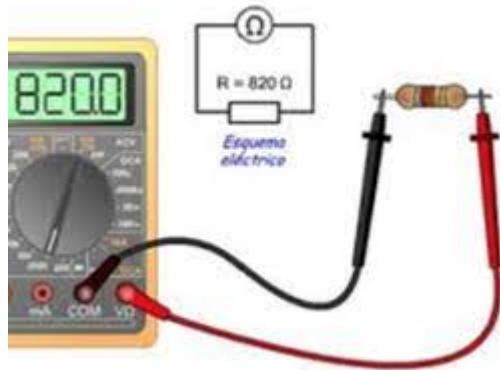


Resistencias: orientación

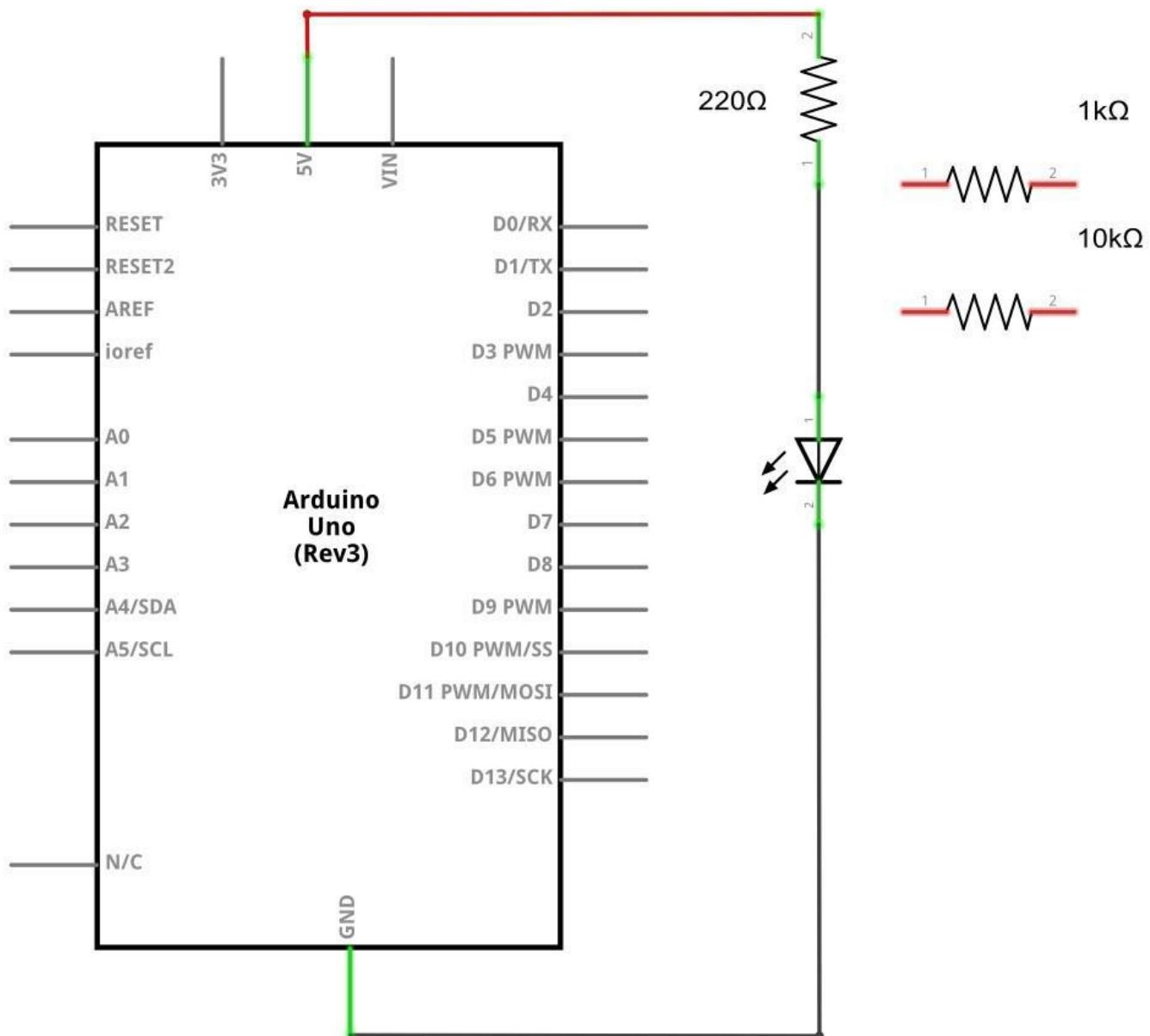
A diferencia de los **LED**, resistencias no tienen un cable positivo y negativo. Se puede conectar de cualquier manera alrededor.

Resistencias: medición

Si desconocemos el valor de una resistencia, también podemos medir su valor utilizando un **multímetro**.



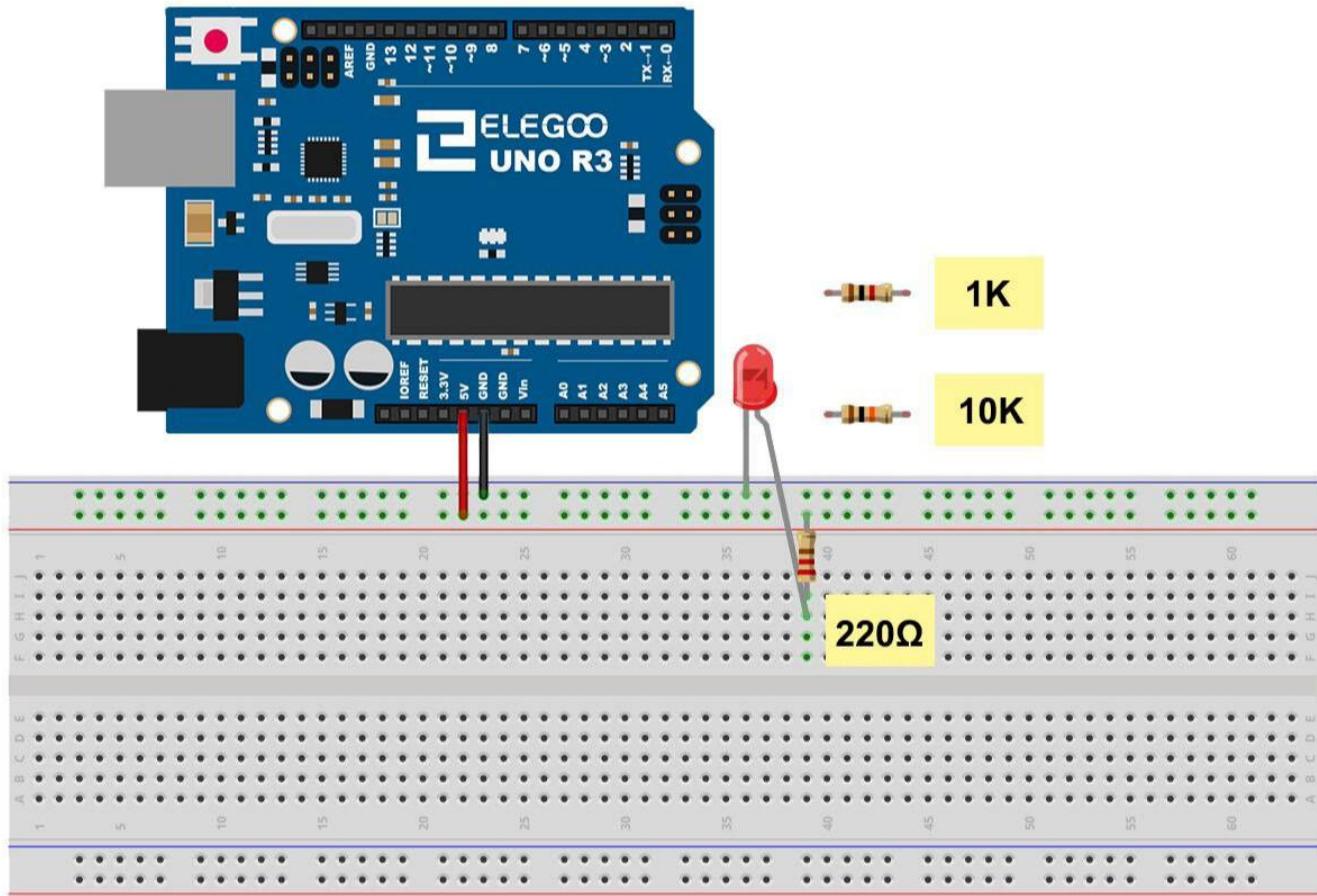
Esquema



Simulación

La **placa de desarrollo Arduino UNO** es una conveniente fuente de 5 voltios, que vamos a utilizar para alimentar el LED y la resistencia. No necesitas hacer nada con su UNO, salvo que lo conecte un cable USB.

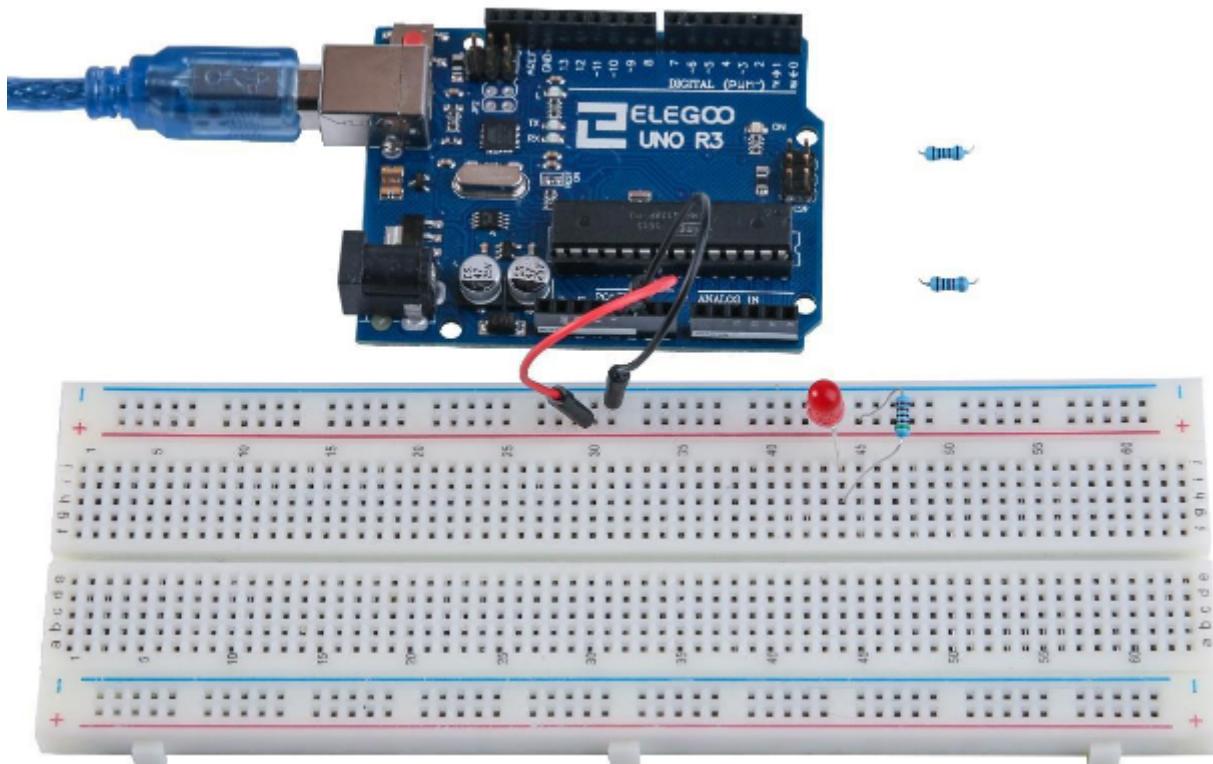
Simulación: ejemplo



Resistencias para LED

- Con la resistencia de $220\ \Omega$, el LED debe ser bastante brillante.
- Si cambia la resistencia $220\ \Omega$ para la resistencia de $1\ k\Omega$, brillará menos.
- Por último, con el resistor de $10\ k\Omega$ en su lugar, el LED será casi invisible.

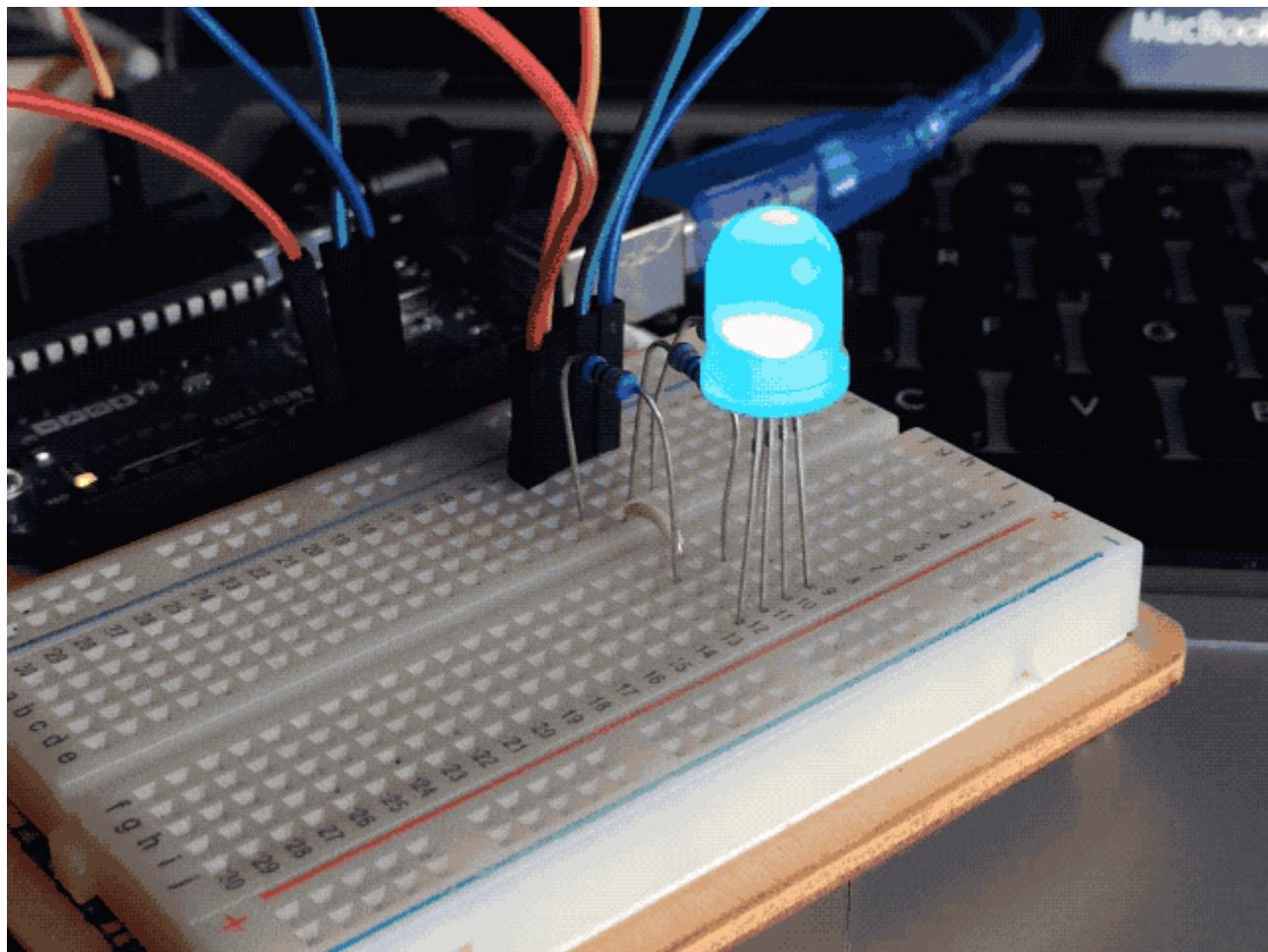
Montaje



LED RGB

En este proyecto aprendremos a controlar la iluminación de un [LED RGB](#) con [Arduino](#)

Los [LED RGB](#) permiten iluminar con cualquier color, a través de 3 leds que contiene en su interior: uno rojo, otro verde y otro azul.



Tipos de LED RGB

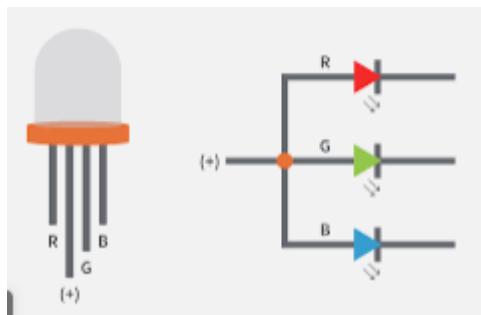
- Existen 2 versiones: Ánodo común y cátodo común.
- Ánodo común utiliza 5V en el pin común, mientras que el cátodo común se conecta a tierra.
- Como con cualquier LED, tenemos que conectar algunas resistencias en línea (3 total) para limitar la corriente.

Componentes necesarios

cantidad	componente
1	placa Arduino
1	protoboard
4	cables jumper
1	LED RGB
3	resistencias de 220 ohmios

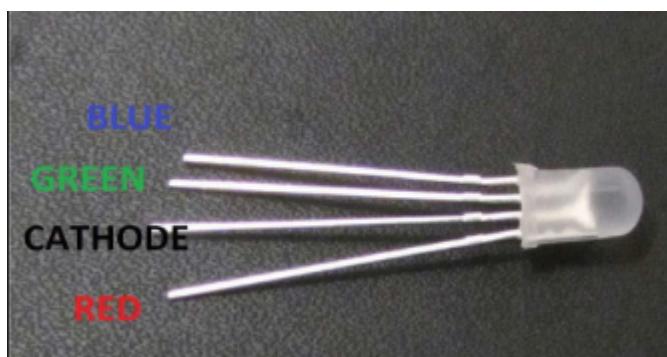
RGB

A primera vista, LEDs RGB (rojo, verde y azul) sólo parecen un LED. Sin embargo, dentro del paquete del LED generalmente, hay realmente tres LEDs, uno rojo, uno verde y sí, uno azul. Controlando el **brillo** de cada uno de los LEDs individuales, podemos mezclar prácticamente cualquier color.



Pines

El **LED RGB** tiene **cuatro pines**. Hay un cable a la conexión positiva de cada uno de los LEDs individuales dentro del paquete y un patilla única que está conectado a los tres lados negativos de los LEDs.



Cada pin separado de color verde o azul o de rojo se llama ánodo.

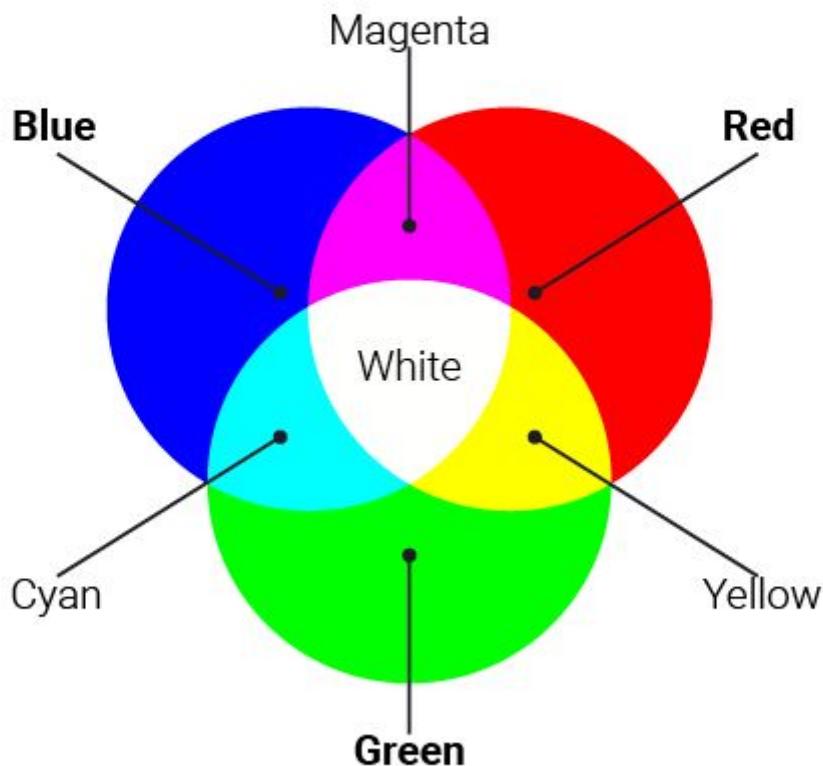
Color

Los colores los conseguiremos mezclando diferentes cantidades de cada color primario.

Color Chart	R	G	B	Color Name
■■■	0	0	0	Black
■■■	255	255	255	White
■■■	224	224	224	Light Gray
■■■	128	128	128	Gray
■■■	64	64	64	Dark Gray
■■■	255	0	0	Red
■■■	255	96	208	Pink
■■■	160	32	255	Purple
■■■	80	208	255	Light Blue
■■■	0	32	255	Blue
■■■	96	255	128	Yellow-Green
■■■	0	192	0	Green
■■■	255	224	32	Yellow
■■■	255	160	16	Orange
■■■	160	128	96	Brown
■■■	255	208	160	Pale Pink

La mezcla creará la **sensación** del color elegido. Podemos controlar el brillo de cada una de las partes de rojas, verdes y azules del LED por separado, lo que es posible mezclar cualquier color.

RGB



Ejemplos

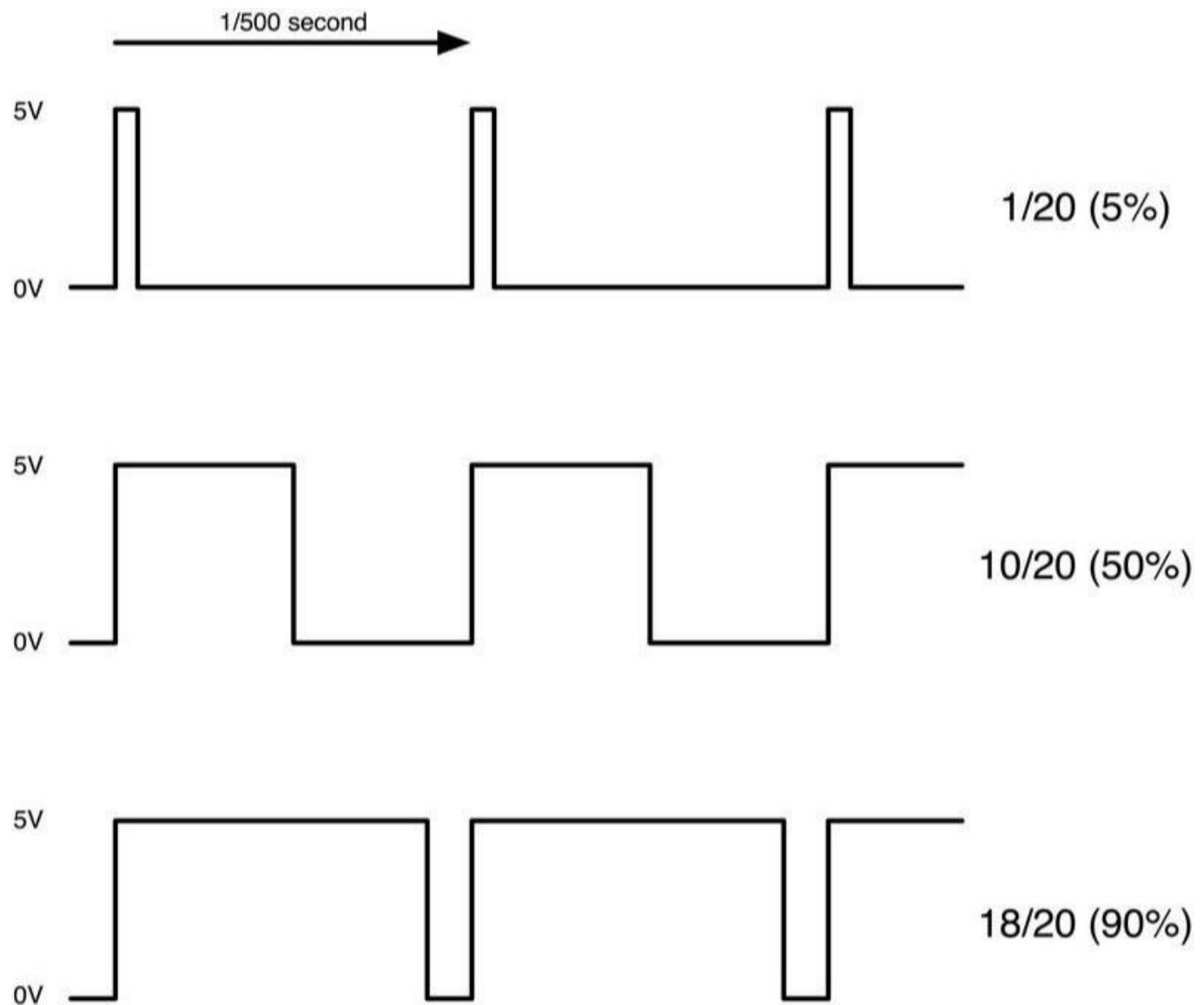
- Si establece el brillo de todos los tres LEDs al ser el mismo, el color general de la luz será blanco.
- Si apagamos el LED azul, para que sólo los LEDs rojo y verdes son el mismo brillo, la luz aparecerá amarillo.

¿Cómo consigo el color negro?

El color **Negro** no es tanto un color como una ausencia de luz. Por lo tanto, lo más cercano que podemos llegar a negro con el LED es apagar los tres colores, poniendo sus valores a 0.

Teoría (PWM)

- **Arduino** tiene una función **analogWrite** que se puede utilizar con pines marcados con un ~ a la salida de una cantidad variable de energía los LEDs apropiados.
- La forma de dar más o menos potencia a cada color es utilizando una señal del tipo **PWM**.
- La **modulación de ancho de pulso (PWM)** es una técnica para el control de potencia. La utilizamos aquí para controlar el brillo de cada uno de los LEDs.



Ciclo de trabajo

Aproximadamente cada 1/500 de segundo, la salida PWM producirá un pulso. La duración de este pulso es controlada por la función 'analogWrite'. Así:

- `analogWrite(0)` no producirá ningún pulso.
- `analogWrite(255)` producirá un pulso que dura todo el camino hasta el pulso siguiente vencimiento, para que la salida es en realidad todo el tiempo.

Si especificamos un valor en el **analogWrite** que está en algún lugar entre 0 y 255, se producir un pulso.

- Si el pulso de salida es alto para el 5% del tiempo, entonces lo que nosotros estamos manejando sólo recibirá el 5% de potencia.
- Si la salida es 5V para el 90% del tiempo, la carga recibirá el 90% .

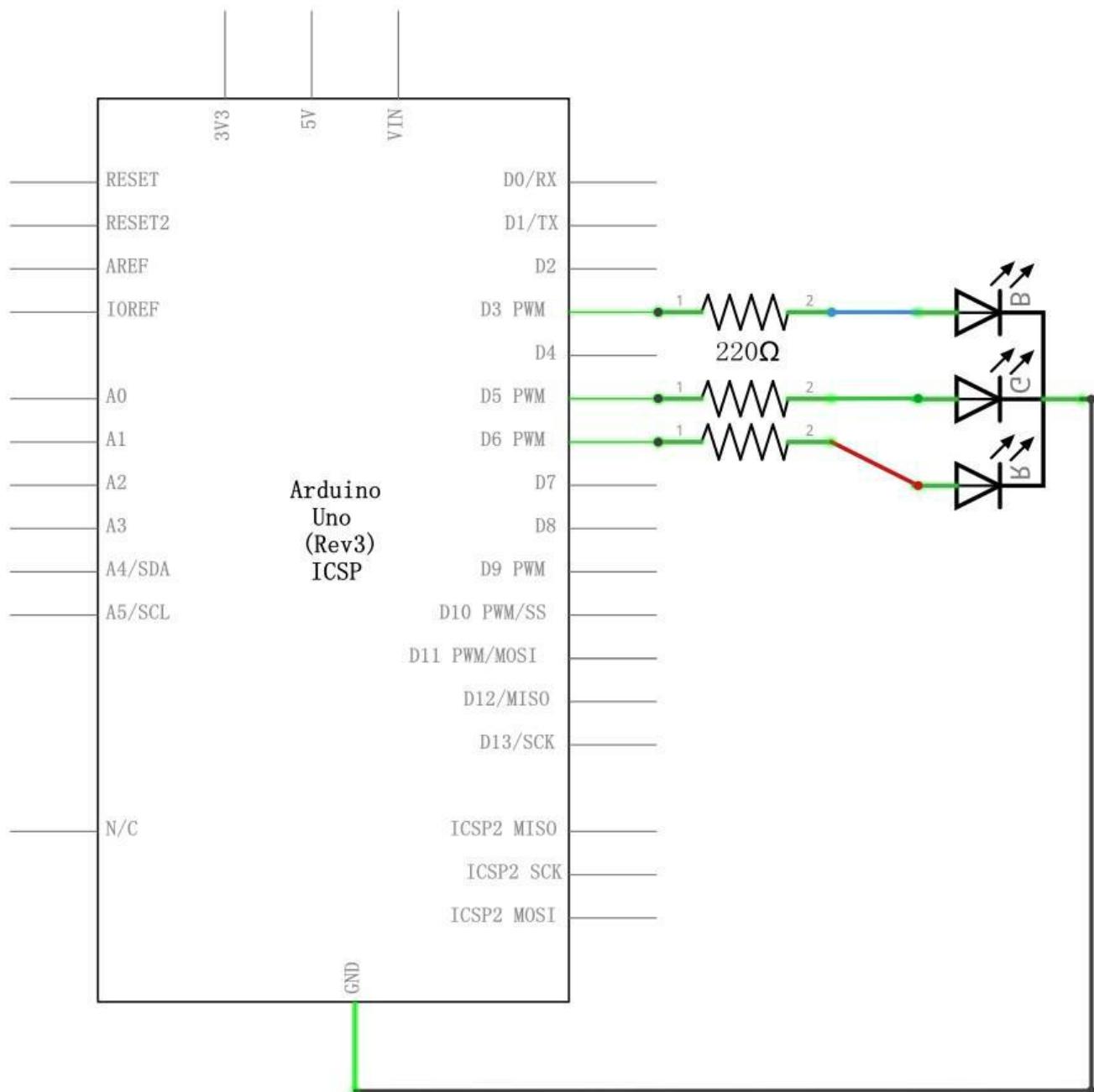
Los LED se encenderán y apagarán en esos períodos, pero nosotros percibiremos que el brillo del LED cambia.

Esquema

El esquema eléctrico que seguiremos es el siguiente:

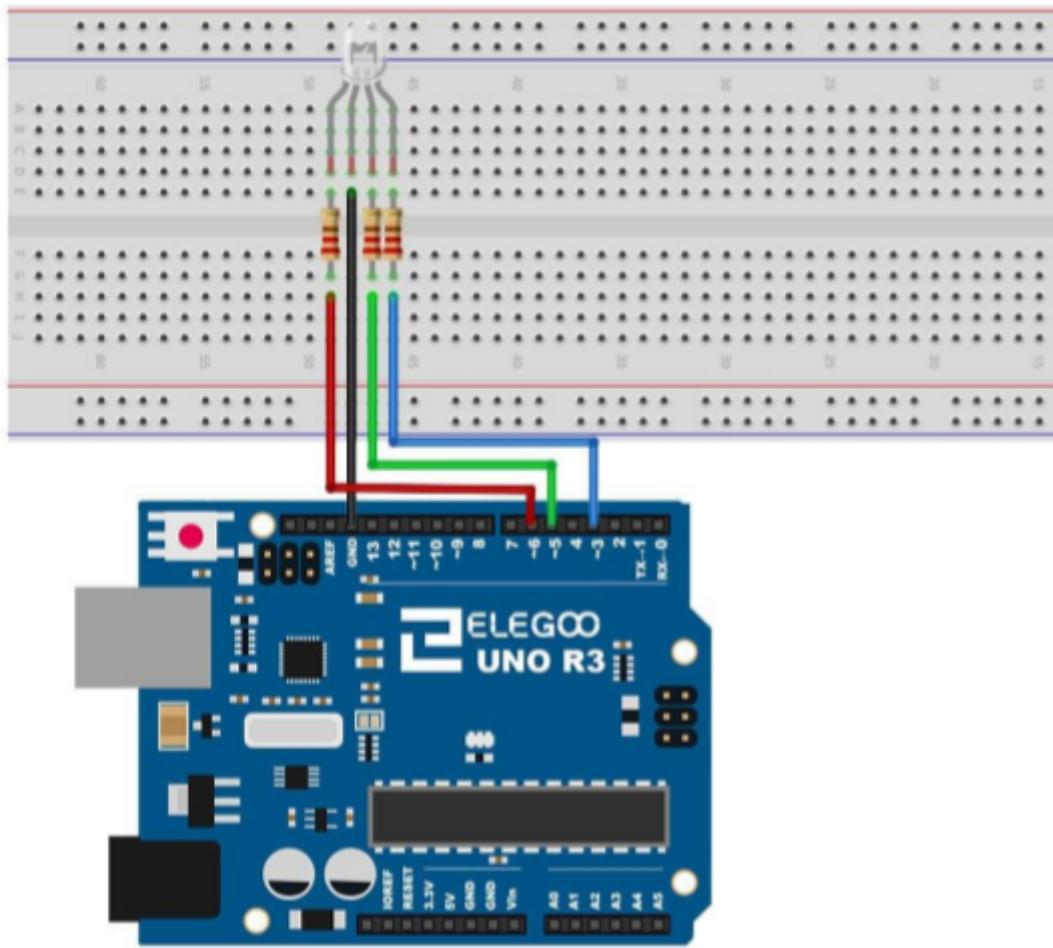
- Cada patilla de un color debe conectarse a una salida digital etiquetada como **PWM**

- La patilla común, irá conectada a un pin de tierra, etiquetado como **GND**

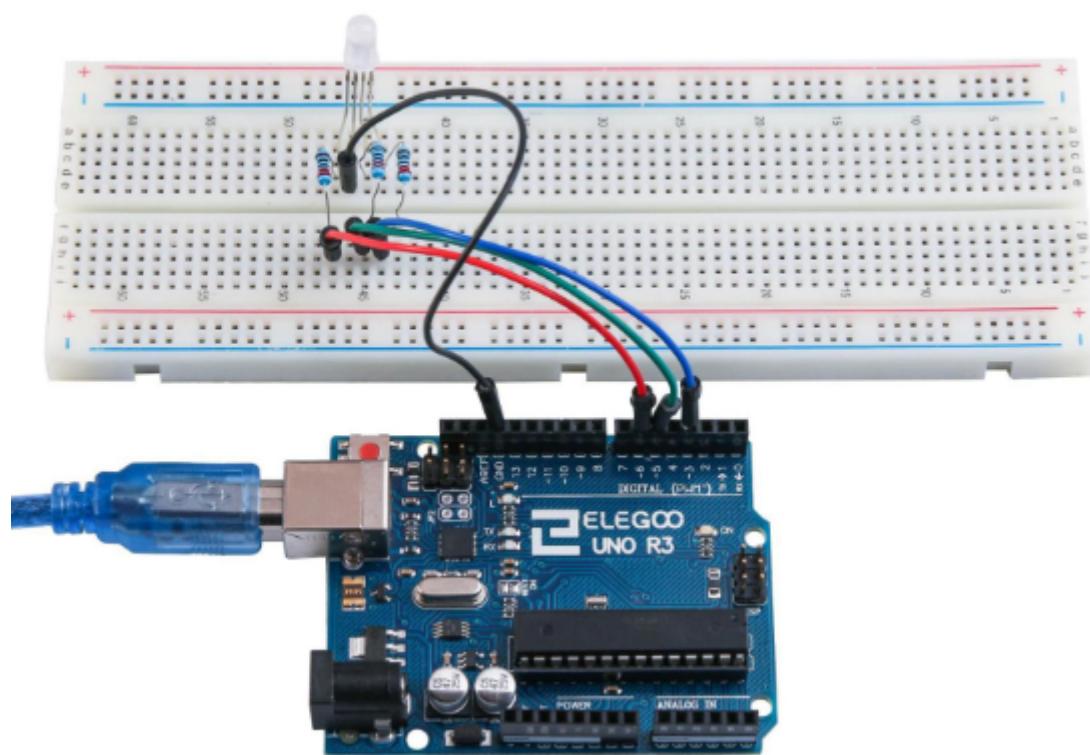


Conexión

1. El cátodo o conexión común es el segundo pin, que también es el **más largo** de las cuatro patas y se conectarán a la **tierra** (GND).
2. Cada LED requiere su propia **resistencia de 220 Ω** para prevenir demasiada corriente que fluye a través de él.
3. Los 3 pines de color (uno rojo, uno verde y uno azul) están conectados a los pines de salida UNO con estas resistencias.

Diagrama de cableado

Una vez conectado, debería quedar de la siguiente forma:



Código programa 1

```
// Define pines
#define BLUE 3
#define GREEN 5
#define RED 6

void setup()
{
    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(BLUE, OUTPUT);
}

void loop()
{
    analogWrite(RED, 0);
    analogWrite(GREEN, 255);
    analogWrite(BLUE, 0);
}
```

Una vez probado, puedes intentar estos ejercicios:

1. Combinar varios valores para conseguir colores diferentes
2. Crear un semáforo utilizando delays y cambiando los valores para producir las luces roja, verde y amarilla.

Código programa 2

```
// Define pines
#define BLUE 3
#define GREEN 5
#define RED 6

void setup()
{
    pinMode(RED, OUTPUT);
    pinMode(GREEN, OUTPUT);
    pinMode(BLUE, OUTPUT);
    digitalWrite(RED, HIGH);
    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, LOW);
}

// define variables
int redValue;
int greenValue;
int blueValue;

// main loop
void loop()
{
```

```
#define delayTime 10 // fading time between colors

redValue = 255; // choose a value between 1 and 255 to change the color.
greenValue = 0;
blueValue = 0;

// this is unnecessary as we've either turned on RED in SETUP
// or in the previous loop ... regardless, this turns RED off
// analogWrite(RED, 0);
// delay(1000);

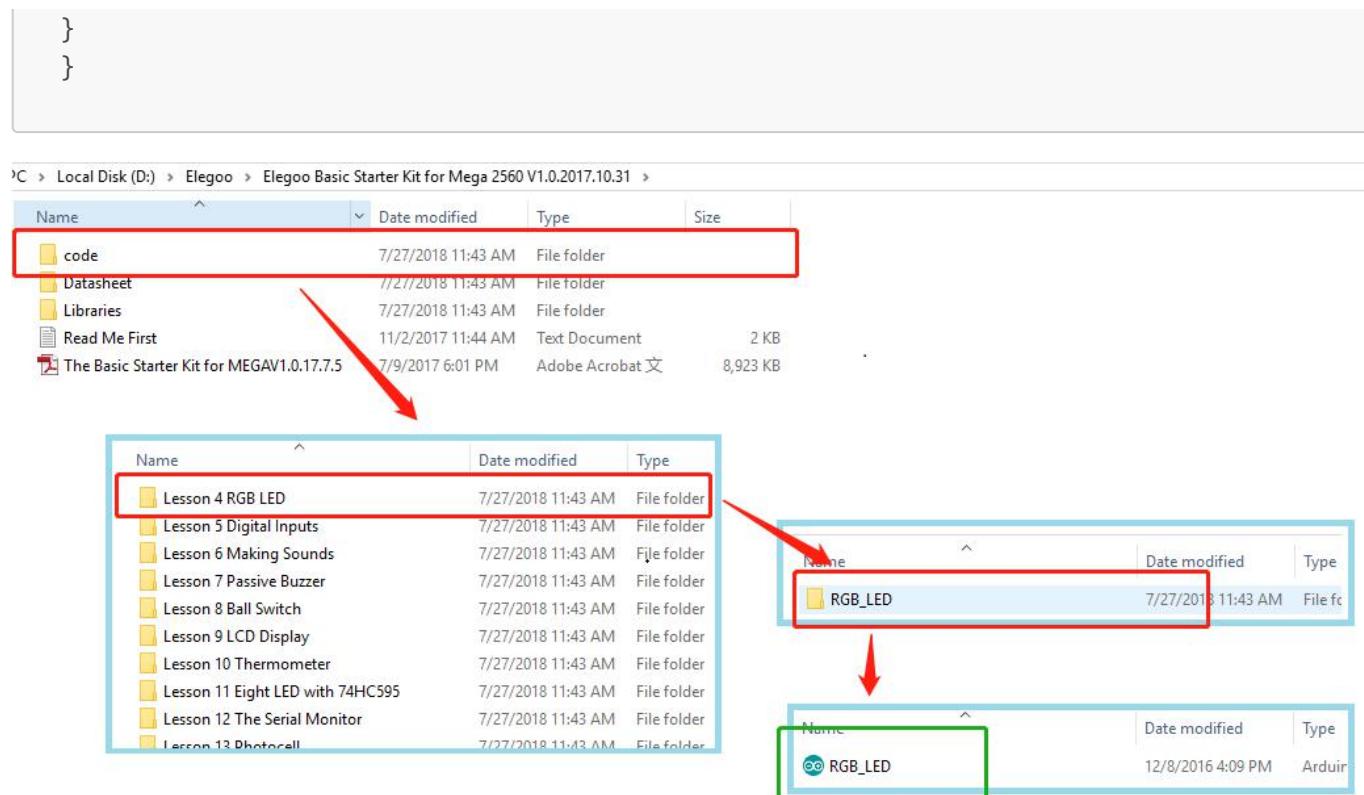
for(int i = 0; i < 255; i += 1) // fades out red bring green full when i=255
{
    redValue -= 1;
    greenValue += 1;
    // The following was reversed, counting in the wrong directions
    // analogWrite(RED, 255 - redValue);
    // analogWrite(GREEN, 255 - greenValue);
    analogWrite(RED, redValue);
    analogWrite(GREEN, greenValue);
    delay(delayTime);
}

redValue = 0;
greenValue = 255;
blueValue = 0;

for(int i = 0; i < 255; i += 1) // fades out green bring blue full when i=255
{
    greenValue -= 1;
    blueValue += 1;
    // The following was reversed, counting in the wrong directions
    // analogWrite(GREEN, 255 - greenValue);
    // analogWrite(BLUE, 255 - blueValue);
    analogWrite(GREEN, greenValue);
    analogWrite(BLUE, blueValue);
    delay(delayTime);
}

redValue = 0;
greenValue = 0;
blueValue = 255;

for(int i = 0; i < 255; i += 1) // fades out blue bring red full when i=255
{
    // The following code has been rearranged to match the other two similar sections
    blueValue -= 1;
    redValue += 1;
    // The following was reversed, counting in the wrong directions
    // analogWrite(BLUE, 255 - blueValue);
    // analogWrite(RED, 255 - redValue);1
    analogWrite(BLUE, blueValue);
    analogWrite(RED, redValue);
    delay(delayTime);
```



Primero especificamos a que pines de **Arduino** he conectado cada LED.

```
// Define Pins
#define BLUE 3
#define GREEN 5
#define RED 6
```

En el setup, declaramos estos pines como salidas (OUTPUT) para poder enviar corriente hacia los LED.

```
void setup()
{
pinMode(RED, OUTPUT);
pinMode(GREEN, OUTPUT);
pinMode(BLUE, OUTPUT);
digitalWrite(RED, HIGH);
digitalWrite(GREEN, LOW);
digitalWrite(BLUE, LOW);
}
```

Antes de echar un vistazo a la **función loop**, veamos la última función en el proyecto.

Las variables de definición:

```
redValue = 255; // choose a value between 1 and 255 to change the color.
greenValue = 0;
blueValue = 0;
```

Esta función tiene tres argumentos, uno para el brillo de los LEDs rojos, verdes y azules. En cada caso de que el número será en el rango 0 a 255, donde 0 significa apagado y 255 significa brillo máximo. La función entonces llama 'analogWrite' para ajustar el brillo de cada LED.

Si nos fijamos en la **función loop** se puede ver que ajuste la cantidad de luz roja, verde y azul que queremos mostrar y luego una pausa por un segundo antes de pasar al siguiente color.

```
#define delayTime 10 // fading time between colors  
Delay(delayTime);
```

Tira de LEDs

Los LEDs se están volviendo cada vez más populares como un medio para iluminar un espacio. Uno de los tipos de LEDs más nuevos y populares es el **WS2812B**. Estos LEDs ofrecen una serie de ventajas sobre los LEDs tradicionales, como la capacidad de cambiar de color y la capacidad de crear una gran variedad de efectos de iluminación.

El **WS2812B** es un tipo de LED direccionable. Esto significa que cada LED individual se puede controlar de forma **independiente**. Esto le da la capacidad de crear algunos efectos de iluminación realmente geniales. Por ejemplo, puede crear un espectáculo de luces en el que cada LED sea de un color diferente y todos cambien de color al mismo tiempo.

Alimentación

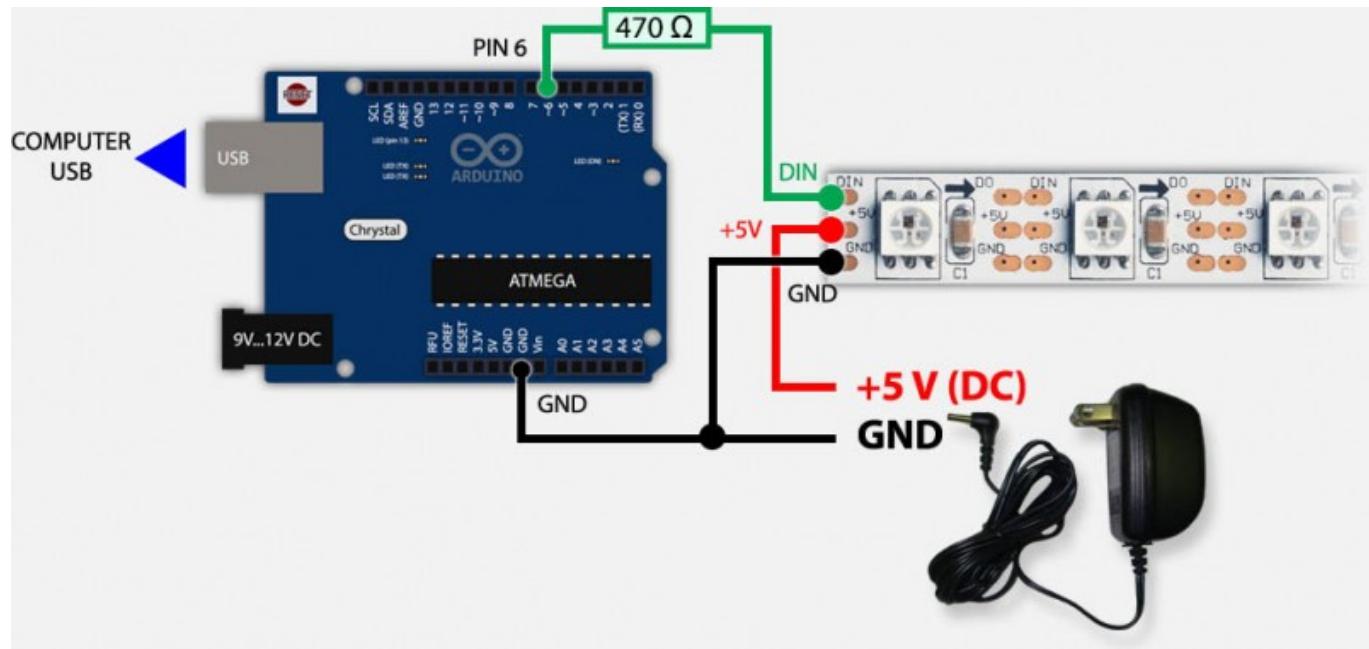
Las tiras **WS2812B** pueden alimentarse a través de una batería o de una fuente de alimentación externa.

- El voltaje necesario para que funcione correctamente la tira de led WS2812B es de 5V.
- Un led WS2812B necesita un mínimo de 60mA para funcionar.

La librería FASTLED

FastLED es una librería de código abierto para programar tira de LEDs RGB direccionables y controladores. Está diseñada para simplificar el proceso de crear efectos de iluminación complejos y se puede usar con una amplia variedad de hardware.

Conexión



Cabecera

```
#include<FastLED.h> // header file

#define NUM_LEDS 60 // number of led present in your strip
#define DATA_PIN 6 // digital pin of your ``Arduino``

CRGB leds[NUM_LEDS];

void setup() {
    FastLED.addLeds<WS2812B, LED_PIN, GRB>(leds, NUM_LEDS);
    FastLED.setBrightness(50);
}

void loop() {
    leds[0] = CRGB::Green; //glow 1st led as green
    leds[1] = CRGB::Blue; //glow 2nd led as blue
    FastLED.show(); // apply the function on led strip
    delay(30);
}
```

Parpadeo (blink)

```
void loop() {
    leds[0] = CRGB::Blue;
    FastLED.show();
    delay(200);
    leds[0] = CRGB::Black;
    FastLED.show();
    delay(200);
}
```

Para apagar la luz:

```
leds[0] = CRGB::Black;
```

Iluminar todos los LED de golpe

Para ello podemos utilizar la función `fill_solid`.

```
fill_solid(leds, NUM_LEDS, CRGB::Red);
```

Arcoiris

```
fill_rainbow(leds, NUM_LEDS, 0, 255 / NUM_LEDS);
```

LED Chaser

```
// chase forward
void loop()
{
    for(int dot = 0; dot < NUM_LEDS; dot++) {
        leds[dot] = CRGB::Red;
        FastLED.show();
        leds[dot] = CRGB::Black;
        delay(300);
    }
}

// chase backward

void loop()
{
    for(int dot=NUM_LEDS ; dot >=0 ; dot--) {
        leds[dot] = CRGB::Red;
        FastLED.show();
        leds[dot] = CRGB::Black;
        delay(300);
    }
}

// chase both

void loop() {
    for(int dot=(NUM_LEDS-1) ; dot >=0 ; dot--) {
        leds[dot] = CRGB::Green;
        FastLED.show();
        leds[dot] = CRGB::Black;
    }
}
```

```

    delay(300);
}

for(int dot = 0;dot < NUM_LEDS; dot++) {
    leds[dot] = CRGB::Red;
    FastLED.show();
    leds[dot] = CRGB::Black;
    delay(300);
}
}

```

Serial glow

```

void loop()
{
    for(int dot=(NUM_LEDS-1) ; dot >=0 ; dot--) {
        leds[dot] = CRGB::HotPink;
        FastLED.show();
        delay(300);
    }

    for(int dot = 0;dot < NUM_LEDS; dot++) {
        leds[dot] = CRGB::Blue;
        FastLED.show();
        delay(300);
    }
}

```

7 segments (1 dígito)

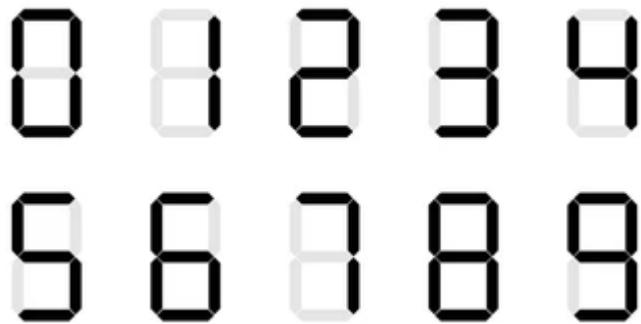
Un **seven segments** es un dispositivo de visualización formado por siete diodos LED dispuestos en forma de número 8.

Estas lámparas se utilizan para mostrar números, letras y caracteres especiales. Se usan en una variedad de dispositivos electrónicos, como relojes digitales, calculadoras, contadores, temporizadores, etc.



-

Estas son las combinaciones que podemos hacer para mostrar los distintos números:

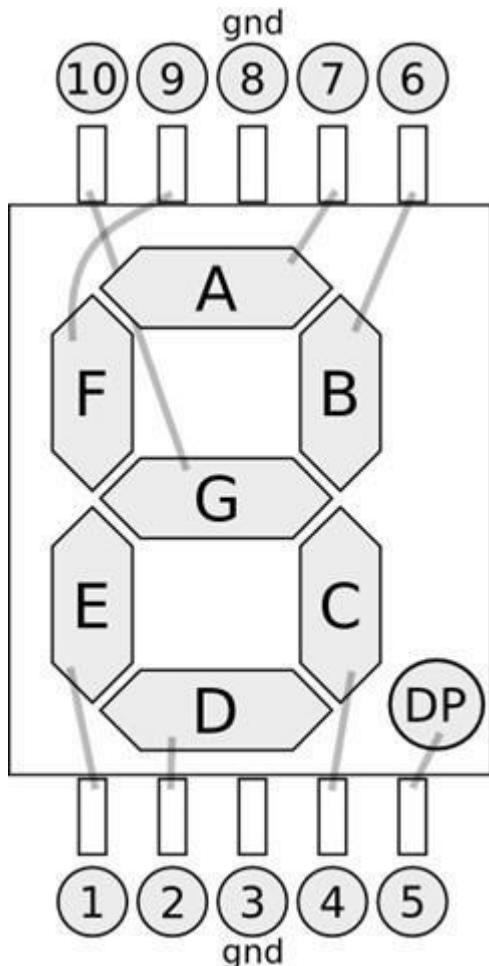


Componentes necesarios

cantidad	componente
1	placa Arduino
1	protoboard
1	circuito integrado 74HC595
1	Pantalla 7 segments
8	resistencias de 220 ohm
1	M-M cables (cables de puente de macho a macho)

Display de siete segmentos

Abajo está el diagrama de pines de siete segmentos

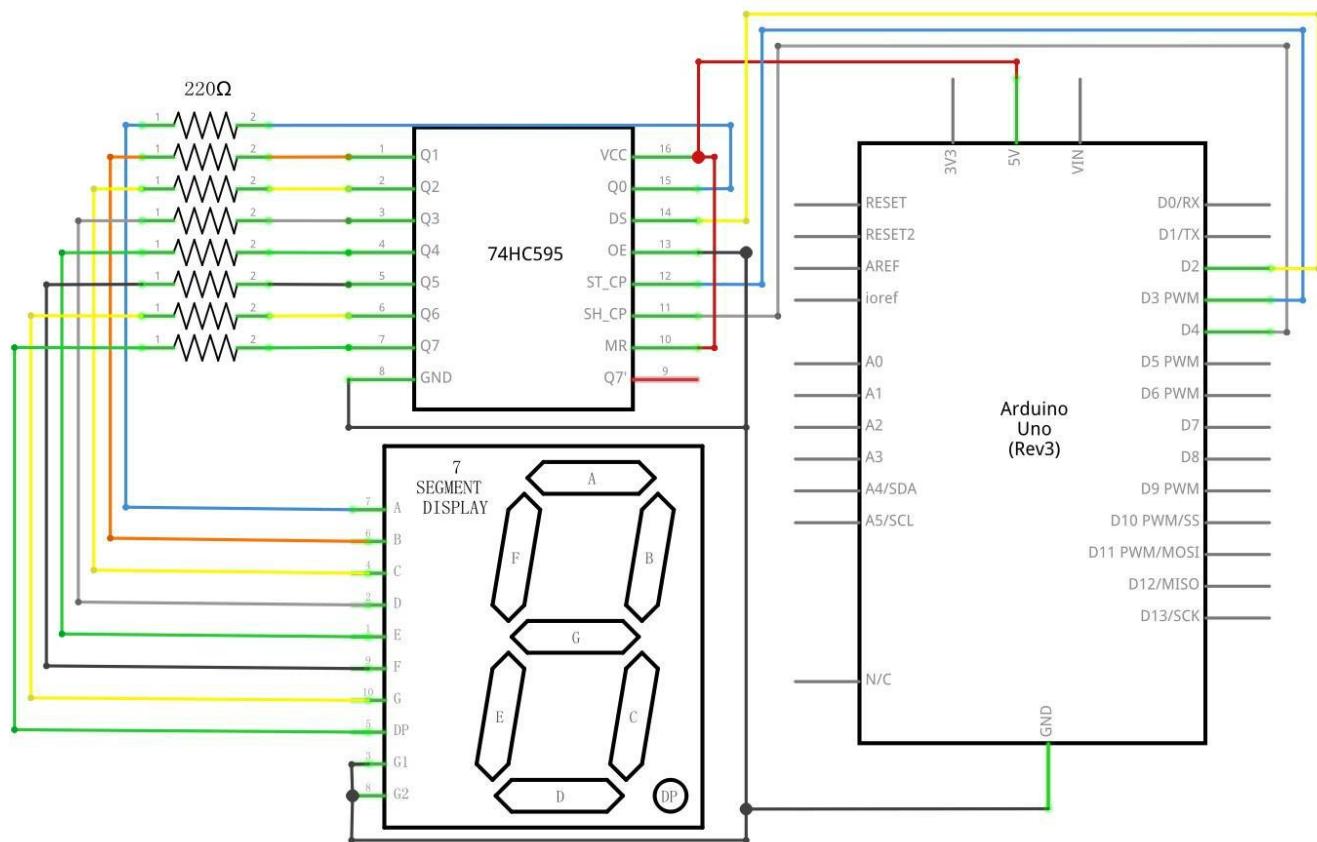


0-9 diez dígitos se corresponden con cada segmento es los siguientes (en la tabla siguiente se aplica común cátodo dispositivo de exhibición de segmento siete, si se utiliza un ánodo común, de la mesa debe ser reemplazado cada 1 0 0 si todos sustituidos por 1):

dp	a	b	c	d	e	f	g	0	1	2	3	4	5	6	7	8	9
	d	p		a	b	c	d	e	f	g							
0	1	1	0	0	1	1	1	1	1	1	0	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

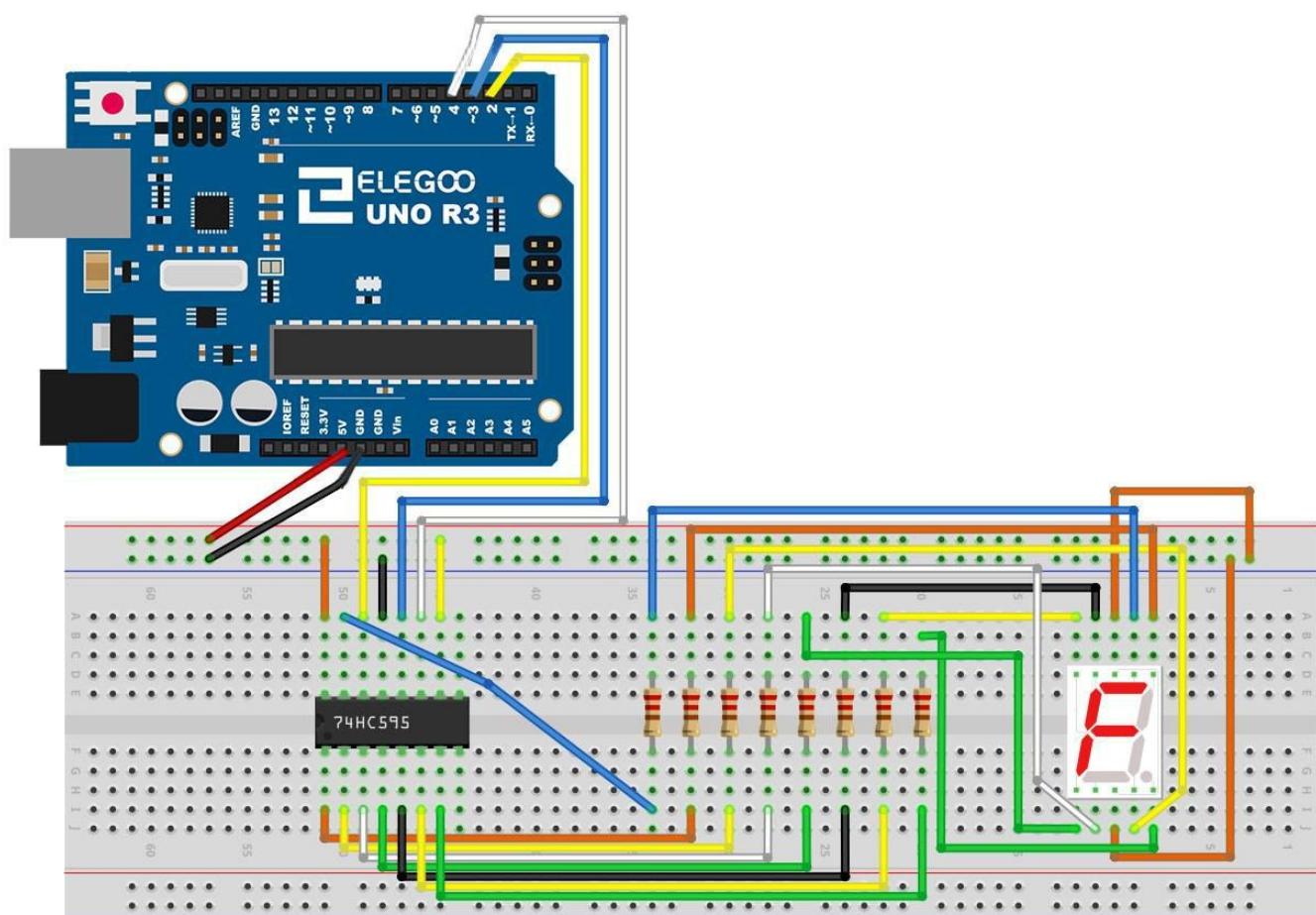
vamos a utilizar el registro de desplazamiento **74HC595** para controlar la visualización de un seven segments.

Conexión



Esquema

Diagrama de cableado



La siguiente tabla muestra la tabla de correspondencias pantalla de siete segmentos 74HC595 pin

Paso uno: conexión 74HC595

En primer lugar, el cableado está conectado a la alimentación y tierra:

- VCC (pin 16) y Señor (pin 10) conectado a 5V
- GND (pin 8) y OE (pin 13) a tierra

Pin conexión DS, ST_CP y SH_CP:

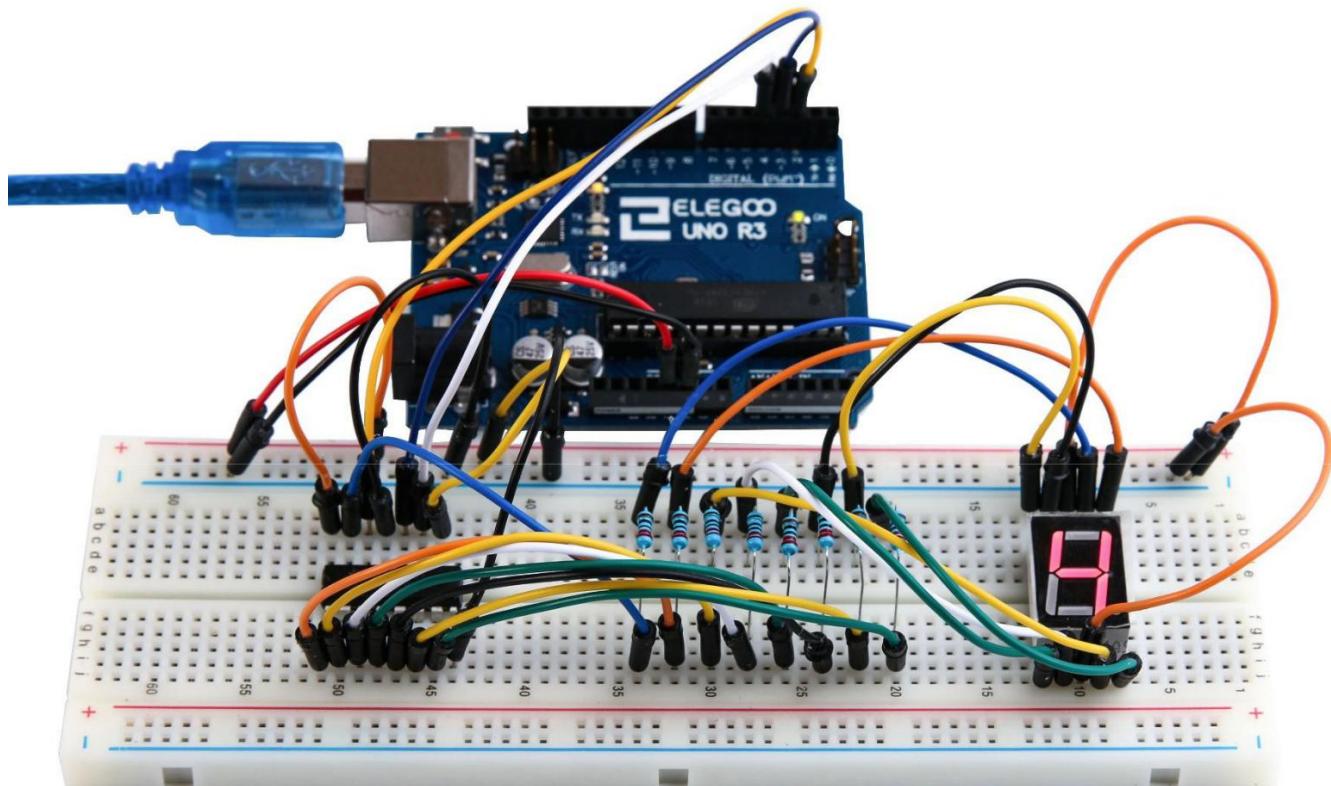
- DS (pin 14) conectado al pin de tablero UNO R3 2 (la cifra por debajo de la línea amarilla)
- ST_CP (pin 12, perno de pestillo) conectado al pin de tablero UNO R3 3 (línea azul de la figura abajo)
- SH_CP (pin 11, pin de reloj) conectado al pin de tablero UNO R3 4 (figura debajo de la línea blanca)

Paso 2: conectar el display de siete segmentos

El display de siete segmentos 3, 8 pin a UNO R3 Junta GND (este ejemplo utiliza el cátodo común, si se utiliza el ánodo común, por favor conecte el 3, 8 pines para tablero UNO R3 + 5V)

Según la tabla anterior, conecte el 74HC595 Q0 ~ Q7 a siete segmentos pantalla pin correspondiente (A ~ G y DP) y luego cada pie en una resistencia de 220 ohmios en serie.

Código



```
int tDelay = 100;
int latchPin = 11;      // (11) ST_CP [RCK] on 74HC595
int clockPin = 9;       // (9) SH_CP [SCK] on 74HC595
int dataPin = 12;       // (12) DS [S1] on 74HC595

byte leds = 0;

void updateShiftRegister()
{
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, LSBFIRST, leds);
    digitalWrite(latchPin, HIGH);
}
```

```

void setup()
{
    pinMode(latchPin, OUTPUT);
    pinMode(dataPin, OUTPUT);
    pinMode(clockPin, OUTPUT);
}

void loop()
{
    leds = 0;
    updateShiftRegister();
    delay(tDelay);
    for (int i = 0; i < 8; i++)
    {
        bitSet(leds, i);
        updateShiftRegister();
        delay(tDelay);
    }
}

```

[BACK](#) [Enrere](#) | [Pàgina principal](#) \n\n title: 7 segments - 4 dígitos footer: Daniel Moreno [🌐](#)
<github.com/danimrprofe> _footer: "" paginate: true _paginate: false _class: invert marp: true

7 segments (4 dígitos)



Resumen

En esta lección, aprendremos a utilizar una pantalla de 7 segmentos de 4 dígitos. Tenemos que tener en cuenta que:

- Si la pantalla es **ánodo común**, el pin común del ánodo se conecta a la fuente de energía
- Si es de **cátodo común**, el pin común del cátodo se conecta a la tierra.

Cuando se utilizan 4 dígitos de 7 segmentos, el ánodo común o pin de cátodo común se utiliza para controlar qué dígito aparece. A pesar de que hay sólo un dígito de trabajo, el principio de persistencia de la visión le permite ver todos los números de muestra ya que cada uno es tan rápida que apenas notará los intervalos de la velocidad de exploración.

Componentes necesarios

Cantidad	Componente
1	Elegoo Uno R3

Cantidad	Componente
----------	------------

1	protoboard
1	74HC595 IC
1	display de 4 dígitos de 7 segmentos
4	Resistencias de 220 ohm
1	M-M cables (cables de puente de macho a macho)

Muestra de 4 dígitos de 7 segmentos

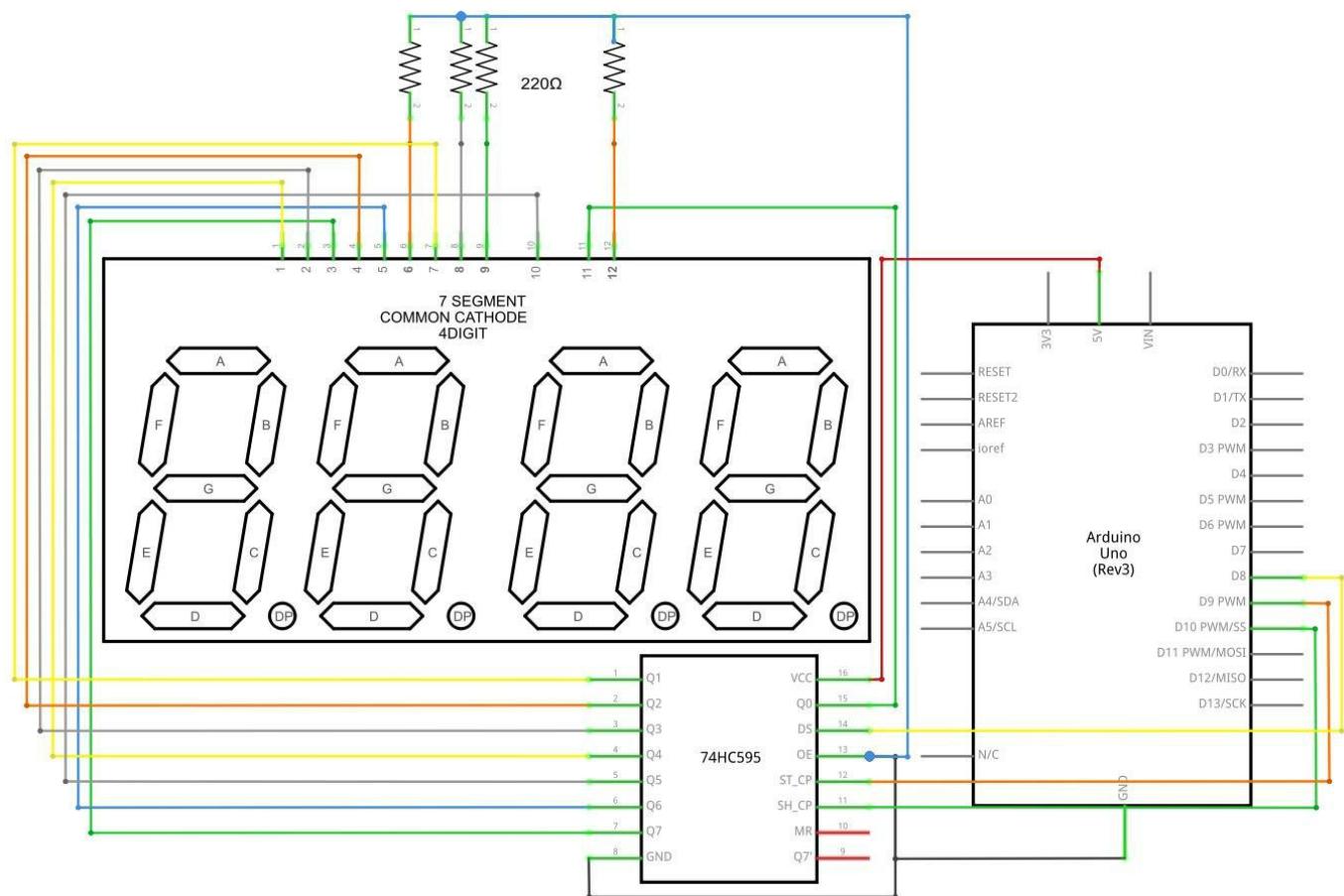
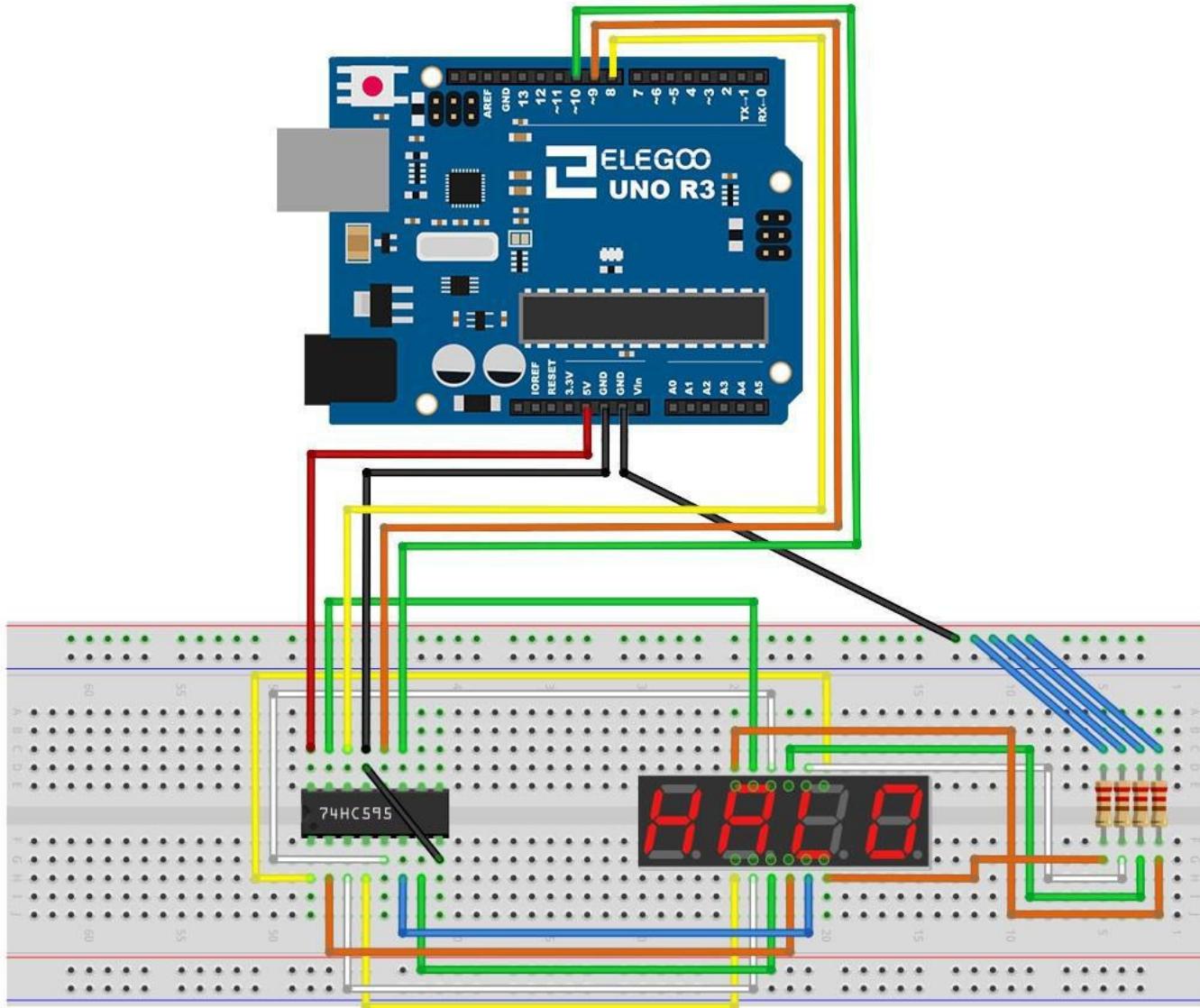
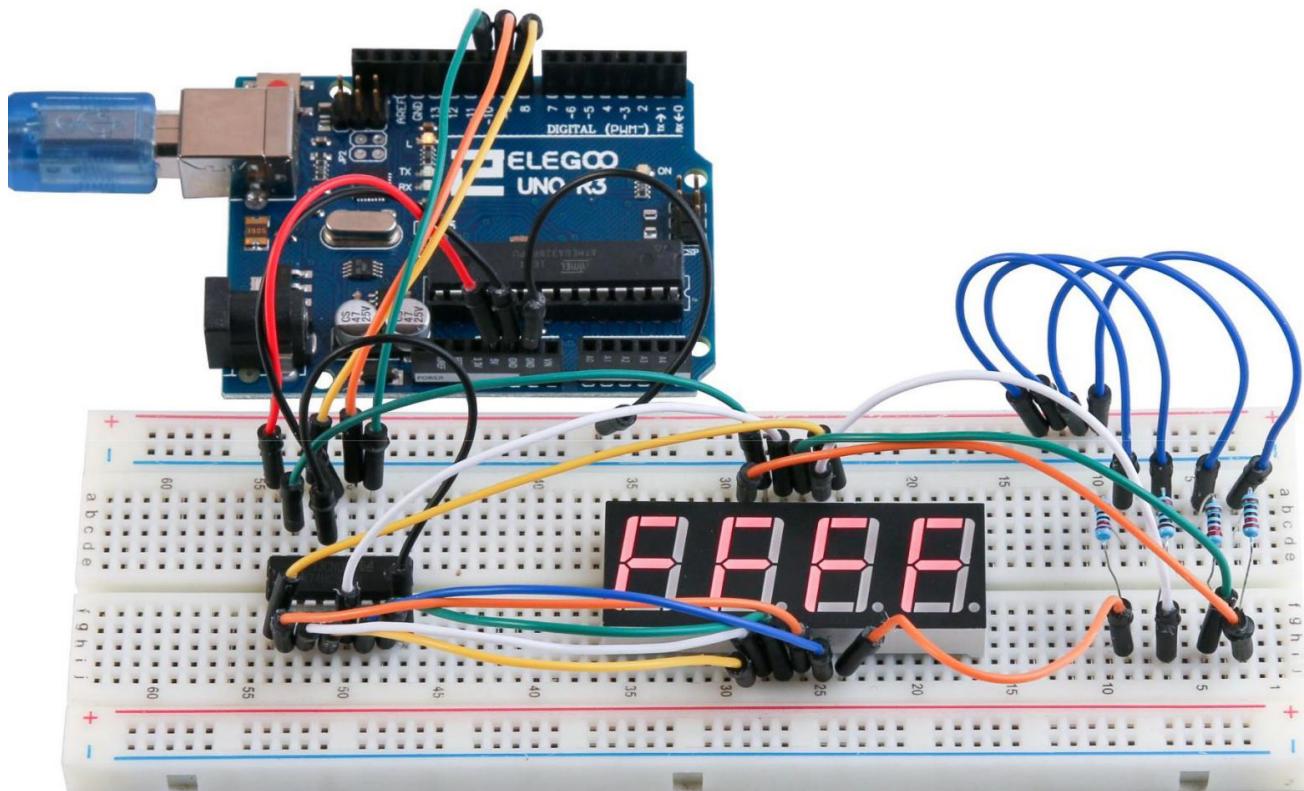


Diagrama de cableado

Cada dígito tiene 7 segmentos (A a G) y un punto decimal (D1 a D4).



Código



Pantalla LCD

La pantalla tiene una retroiluminación de LED y puede mostrar dos filas con hasta 16 caracteres en cada fila.



Circuito integrado LCD1602

La pantalla está incrustada en un **circuito integrado** que la controla, llamado **LCD1602**.

Pines

- **VSS** Un pin que se conecta a tierra
- **VDD** Un pin que se conecta a un + 5V fuente de alimentación
- **VO** ajusta el contraste.
- **RS** Un registro seleccione pin que controla donde en memoria de la pantalla LCD datos de escritura. Usted puede seleccionar el registro de datos, que es lo que pasa en la pantalla, o un registro de instrucción, que es donde busca controlador de LCD para obtener instrucciones sobre qué hacer.
- **R/W**: Pin A lectura y escritura que selecciona el modo de lectura o escritura a modo de E; Permitiendo a un perno con energía de bajo nivel, módulo causas la LDC para ejecutar instrucciones.
- **D0-D7** son los pines para escribir y leer datos.
- **A y K** controlan de la retroiluminación LED de los pernos

Esquema de conexión

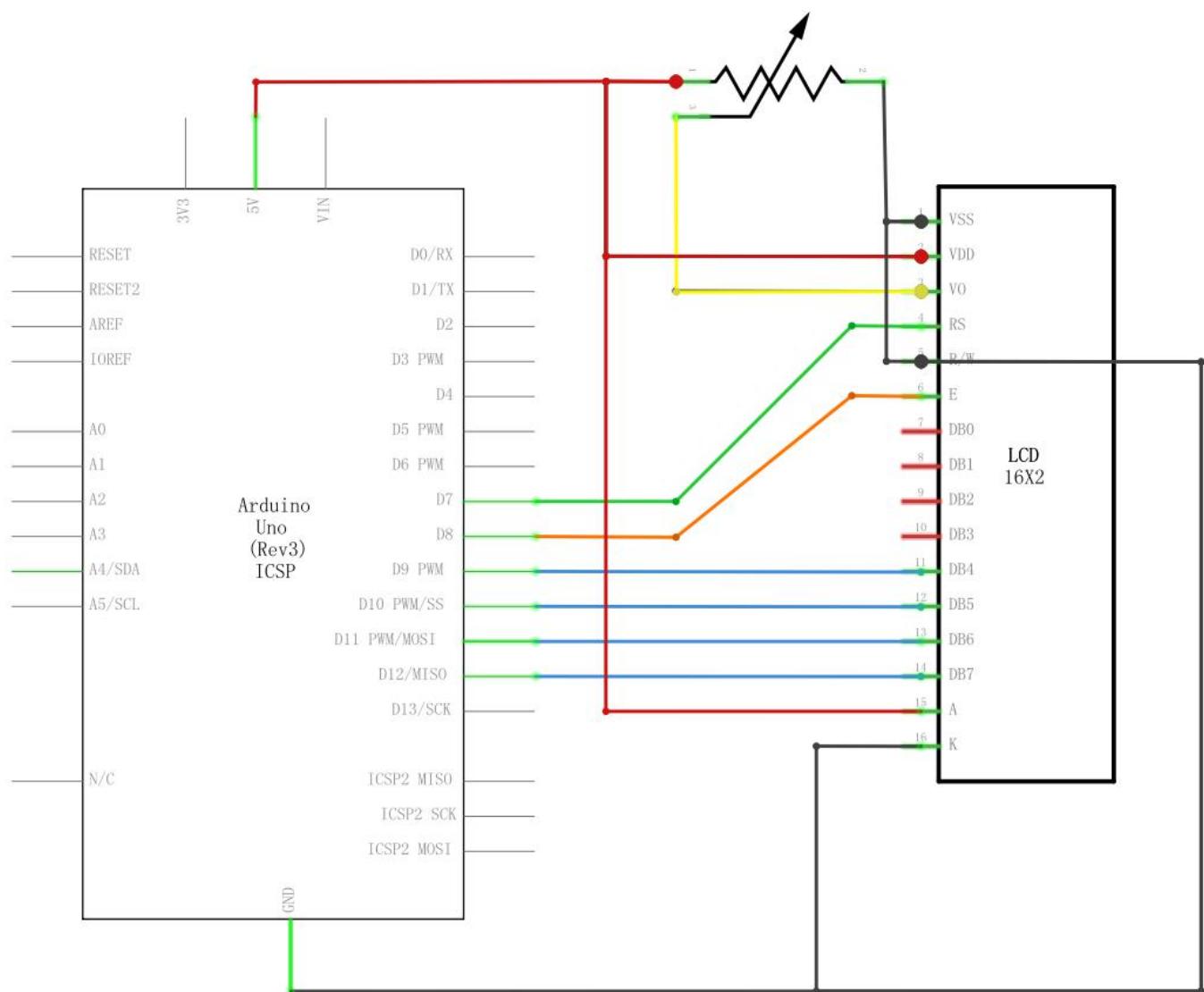
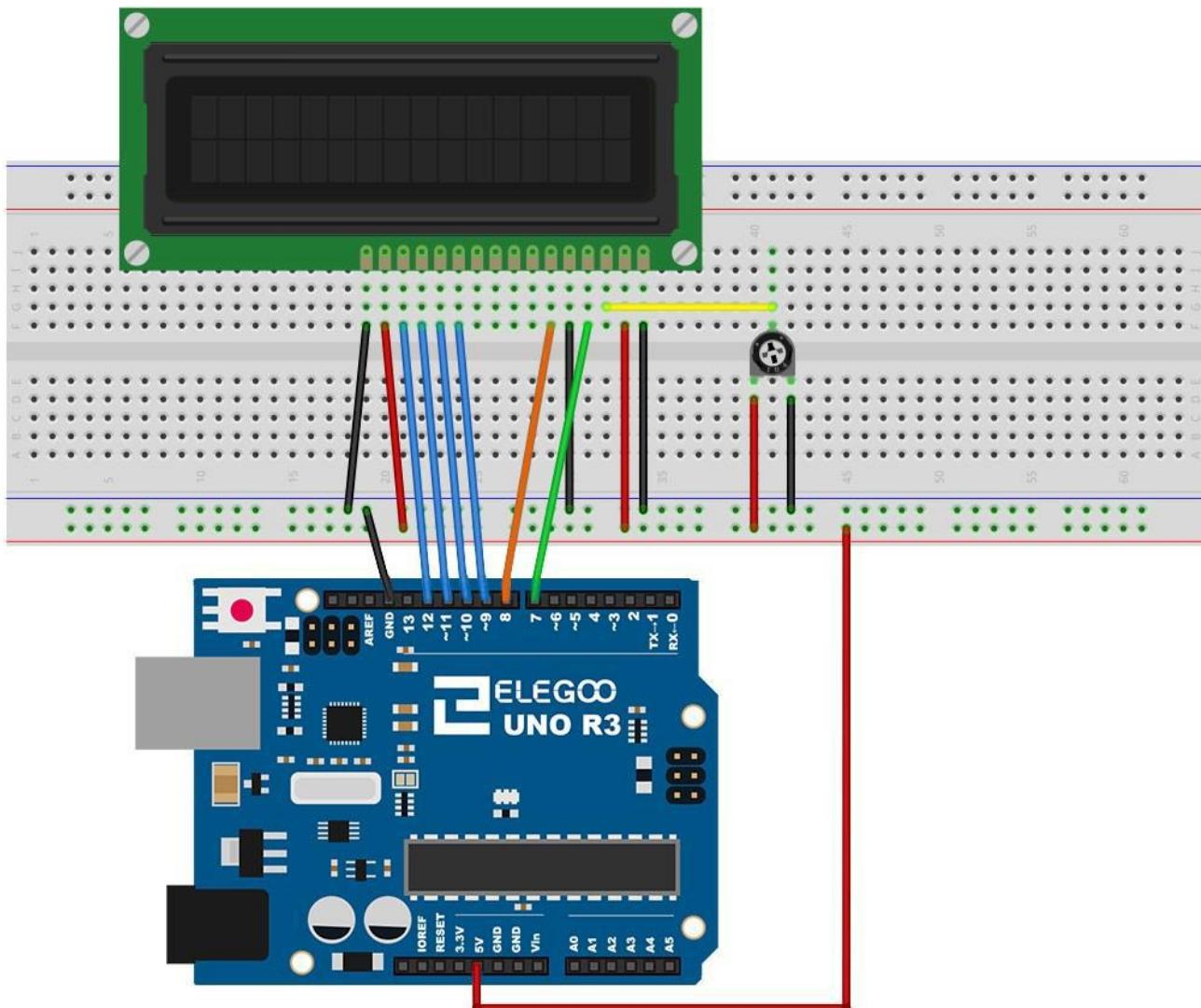


Diagrama de cableado



La pantalla LCD necesita:

- 6 pines digitales de datos de [Arduino](#)
- Conecciones de 5V y GND.

Potenciómetro

El **potenciómetro** se utiliza para controlar el [contraste](#) de la pantalla. En ocasiones se ajusta con un pequeño destornillador. El potenciómetro utilizado será de [10 KOhm](#)



Librería

Antes de ejecutar esto, asegúrese de que ha instalado la [librería < LiquidCrystal >](#) o volver a instalarlo, si es necesario. De lo contrario, el código no funcionará.

Lo primero que nota en el dibujo es la línea:

```
#include < LiquidCrystal.h >
```

Esto dice **Arduino** que queremos utilizar la **librería** de cristal líquido.

A continuación tenemos la línea que teníamos que modificar. Esto define qué pines de **Arduino** son para conectarse a que pines de la pantalla.

```
LiquidCrystal lcd (7, 8, 9, 10, 11, 12);
```

Después de subir este código, asegúrese de que se enciende la retroiluminación y ajustar el potenciómetro de toda la manera alrededor hasta que aparezca el mensaje de texto

En la función de **setup**, tenemos dos comandos:

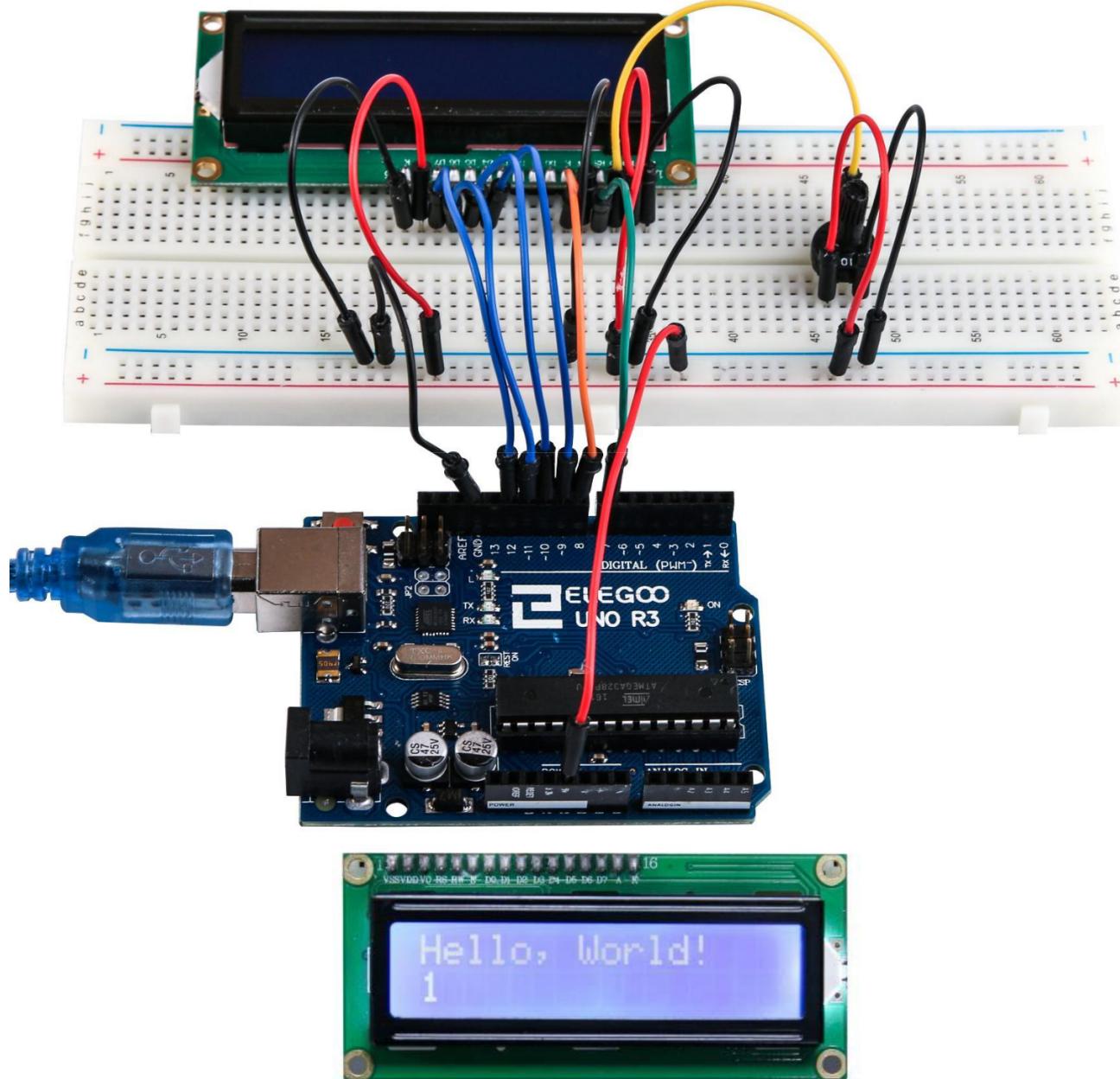
```
LCD.Begin (16, 2);
LCD.Print ("Hola, mundo!");
```

La primera cuenta la **librería** de cristal líquido cuántas columnas y filas tiene la pantalla. La segunda línea muestra el mensaje que vemos en la primera línea de la pantalla.

En la función de 'loop', aso tienen dos comandos:

```
lcd.setCursor (0, 1);
LCD.Print(Millis()/1000);
```

El primero establece la posición del cursor (donde aparecerá el siguiente texto) columna 0 y fila 1. Los números de columna y fila comienzan en 0 en lugar de 1.



La segunda línea muestra el número de milisegundos desde que se restableció el [Arduino](#).

```
// include the library code:  
#include <LiquidCrystal.h>  
  
// initialize the library with the numbers of the interface pins  
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);  
  
void setup() {  
    // set up the LCD's number of columns and rows:  
    lcd.begin(16, 2);  
    // Print a message to the LCD.  
}
```

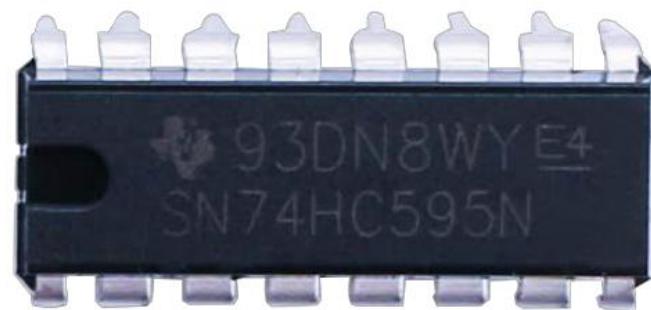
```
lcd.print("Hello, World!");  
}  
  
void loop() {  
    // set the cursor to column 0, line 1  
    // (note: line 1 is the second row, since counting begins with 0):  
    lcd.setCursor(0, 1);  
    // print the number of seconds since reset:  
    lcd.print(millis() / 1000);  
}
```

LED con 74HC595

Resumen

En esta lección, usted aprenderá cómo utilizar ocho LEDs rojo grandes con un UNO sin necesidad de renunciar a 8 patas de salida

Aunque usted podría conectar ocho LEDs con una resistencia a un pin UNO rápidamente empezaría a quedarse sin patas en su UNO. Si no tienes un montón de cosas conectadas a la ONU. Está bien hacerlo - pero a menudo tiempos queremos botones, sensores, servos, etc. y antes de saberlo que no tienes pernos de izquierda. Así, en lugar de hacer eso, vas a usar un chip llamado el 74HC595 Serial a paralelo convertidor. Este chip tiene ocho salidas (perfectos) y tres entradas que utilizas para alimentar datos en él un poco a la vez.



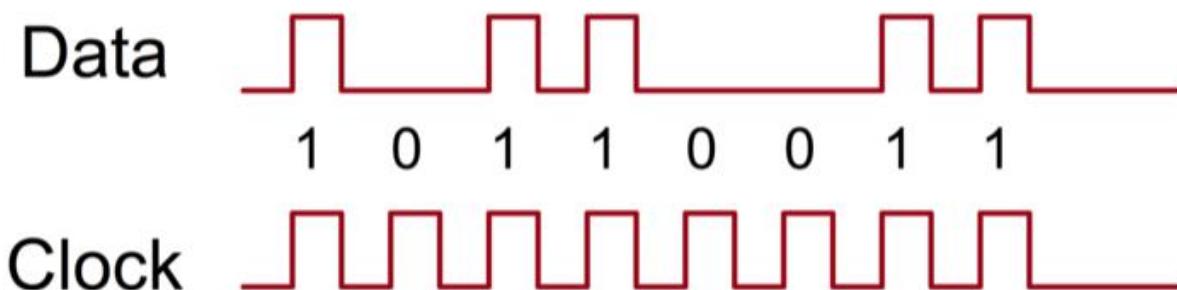
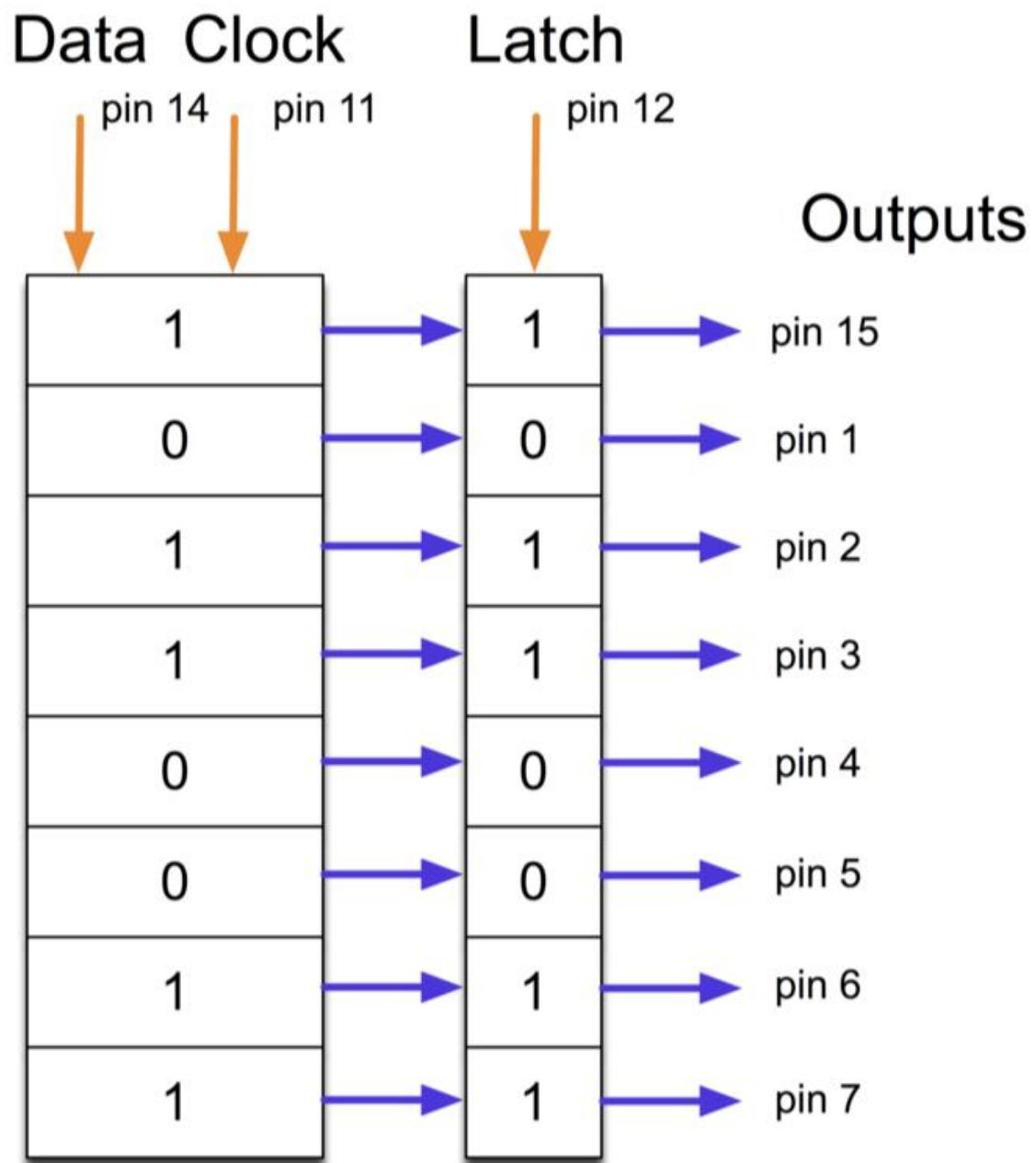
Este chip hace un poco más lento para los LEDs (sólo se puede cambiar el LED unos 500.000 veces por segundo en lugar de 8.000.000 por segundo) pero todavía es muy rápido, forma más rápido que los seres humanos puede detectar, asíque vale!

Componente necesario:

- (1) x Elegoo Uno R3
- Protoboard
- leds
- resistencias de 220 ohmios
- IC x 74hc595
- M M cables (cables de puente de macho a macho)

74HC595 Registro de desplazamiento

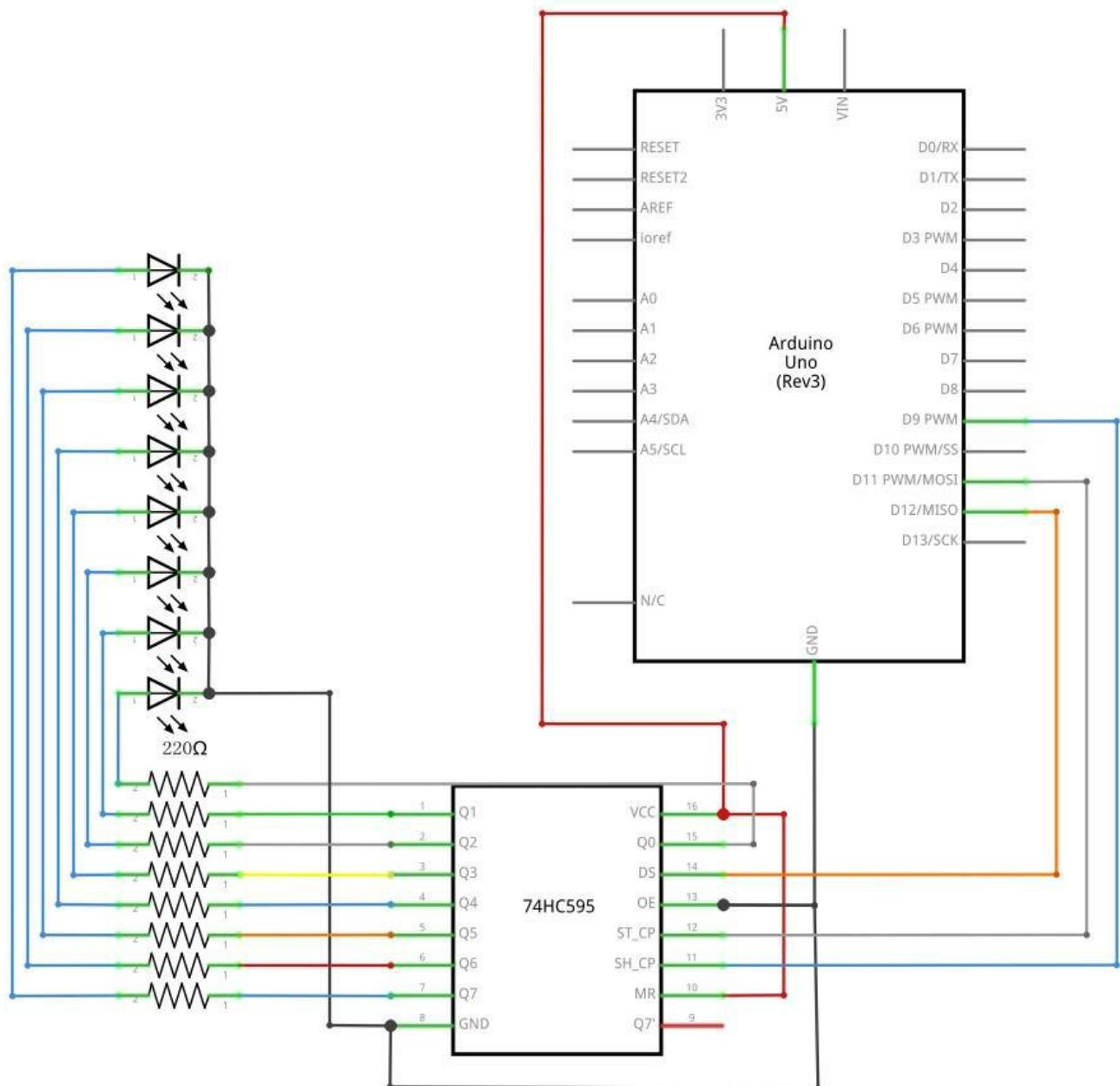
El registro de desplazamiento es un tipo de chip que tiene lo que puede considerarse como posiciones de memoria ocho, cada uno de ellos puede ser un 1 o un 0. Para definir cada uno de estos valores encendido o apagado, alimentamos en los datos mediante los pines del chip 'Datos' y 'El reloj'.



El pin de reloj debe recibir ocho pulsos. En cada pulso, si el pin de datos es alto, entonces un 1 obtiene empujado en el registro de desplazamiento; de lo contrario, un 0. Cuando se han recibido los ocho impulsos,

permitiendo el pin 'Pestillo' copia esos ocho valores en el registro de cierre. Esto es necesario; de lo contrario, parpadean mal los LEDs como se carga los datos en el registro de desplazamiento.

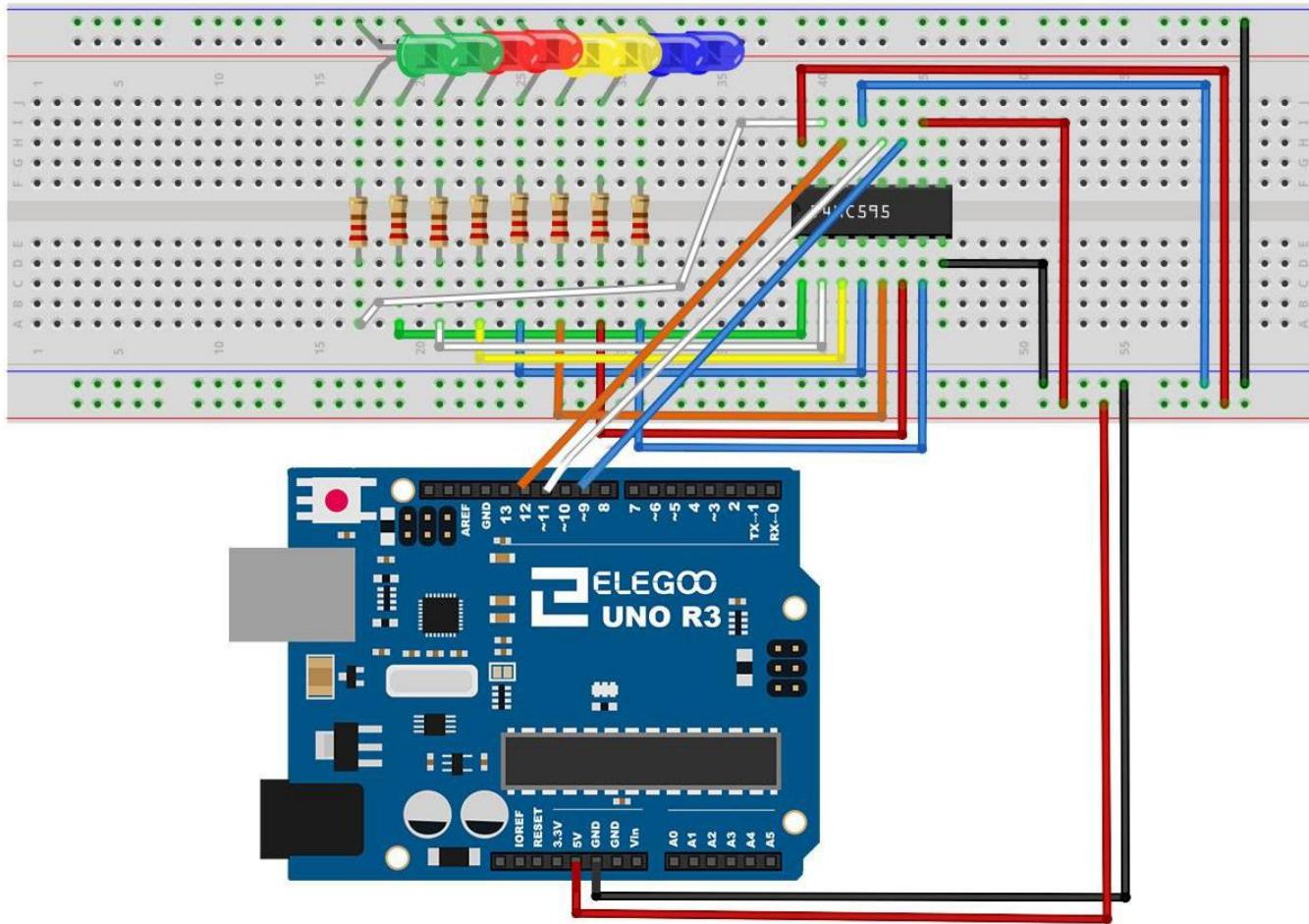
El chip también tiene un pin de salida activado (OE), que se utiliza para activar o desactivar las salidas a la vez. Podría conectar esto a un pin PWM capaz UNO y usar 'analogWrite' para controlar el brillo de los LEDs. Este pin es baja activa, por lo que nos aten a la tierra GND.



Conexión

Esquema

Diagrama de cableado



Ya que tenemos ocho LEDs y ocho resistencias para conectar, hay realmente muy pocas conexiones a realizar.

Es probablemente más fácil poner el **chip 74HC595** en primer lugar, como casi todo lo demás se conecta a él. Ponerlo de modo que la muesca en forma de U poco hacia la parte superior de la placa.

Pin 1 del chip es a la izquierda de esta muesca. Digital 12 del UNO va al pin #14 del registro de desplazamiento Digital 11 del UNO va al pin #12 del registro de desplazamiento

9 digital a partir de la UNO va al pin #11 del registro de desplazamiento

Todos sino una de las salidas de la IC está en el lado izquierdo del chip. Por lo tanto, para facilitar la conexión, es donde están los LEDs, también.

Después de la viruta, poner las resistencias en su lugar. Usted necesita tener cuidado de que ninguno de los cables de las resistencias tocan. Usted debe comprobar esto otra vez antes de conectar la energía a la ONU. Si le resulta difícil organizar las resistencias sin sus conductores tocando, entonces ayuda a acortar los cables que están mintiendo más cercanos a la superficie de la placa.

A continuación, coloque los LEDs en la protoboard. Cuanto más positivo lleva LED debe ser hacia el chip, de cualquier lado de la placa están en.

Conecte los conductores del puente como se muestra arriba. No olvide que va desde el pin 8 del IC a la columna GND de la placa.

Carga el bosquejo aparece un poco más adelante y probar. Cada LED debe encenderse alternadamente hasta que todos los LEDs están encendidos y luego se apagara y el ciclo se repite.

Código

Después de cableado, por favor, abra el programa en el código de carpeta lección 24 8 LED con 74HC595 y haga clic en UPLOAD para cargar el programa. Ver Lección 2 para más detalles sobre el programa cargar si hay algún error.

Lo primero que hacemos es definir los tres pernos que vamos a utilizar. Estos son los UNO salidas digitales que se conectarán a los pines de datos, reloj y cierre de los 74HC595.

```
int latchPin = 11;
clockPin int = 9;
int dataPin = 12;
```

A continuación, se define una variable llamada 'leds'. Esto se utiliza para sostener el patrón de que LED actualmente es activado o desactivados. Datos de tipo 'byte' representan números de ocho bits. Cada bit puede estar encendido o apagado, esto es perfecto para realizar un seguimiento de cuáles de nuestros ocho LEDs son on u off.

```
leds de byte = 0;
```

La función de **setup** sólo establece los tres pernos que estamos utilizando para ser de salidas digitales.

```
void setup()
{
pinMode (latchPin, salida);
pinMode (dataPin, salida);
pinMode (clockPin, salida);
}
```

La **función loop** inicialmente apaga todos los LEDs, al darle a los variable 'leds' el valor 0. A continuación, llama 'updateShiftRegister' que enviará el patrón de 'leds' para el registro de desplazamiento para que el LED se apague. Se tratará con 'updateShiftRegister' funcionamiento más adelante.

La función loop hace una pausa de medio segundo y entonces empieza a contar de 0 a 7 usando el bucle 'for' y la variable 'i'. Cada vez utiliza la función de [Arduino](#) 'verdadera' para establecer el bit que controla ese LED en la variable 'leds'. A continuación también llama 'updateShiftRegister' para que los leds actualizar para reflejar lo que está en la variable 'leds'.

Hay entonces medio segundo de retraso antes de 'i' se incrementa y se ilumina el LED próximo.

```
void loop()
{
LED = 0;
updateShiftRegister();
Delay(500);
for (int i = 0; i < 8; i++)
{
```

```
    bitSet(leds, i);
    updateShiftRegister();
    Delay(500);
}
}
```

La función 'updateShiftRegister', en primer lugar se establece la latchPin baja, entonces llama al UNO función 'shiftOut' antes de poner el 'latchPin' alta otra vez. Esto toma cuatro parámetros, los dos primeros son los pines para datos y el reloj respectivamente.

El tercer parámetro especifica que final de los datos que desea iniciar en el. Vamos a empezar con la derecha más poco, que se conoce como el 'Bit menos significativo' (LSB).

El último parámetro es los datos reales para ser cambiado de puesto en el registro de desplazamiento, que en este caso es 'leds'.

```
void updateShiftRegister()
{
    digitalWrite (latchPin, bajo);
    shiftOut (dataPin, clockPin, LSBFIRST, leds);
    digitalWrite (latchPin, HIGH);
}
```

Si usted deseó dar vuelta a uno de los LED apagado en lugar, llamaría una función similar de [Arduino](#) (bitClear) con la variable de 'leds'. Esto ajustará ese poco de 'leds' para ser 0 y entonces sólo necesitará seguir con una llamada a 'updateShiftRegister' para actualizar la actual LED.

