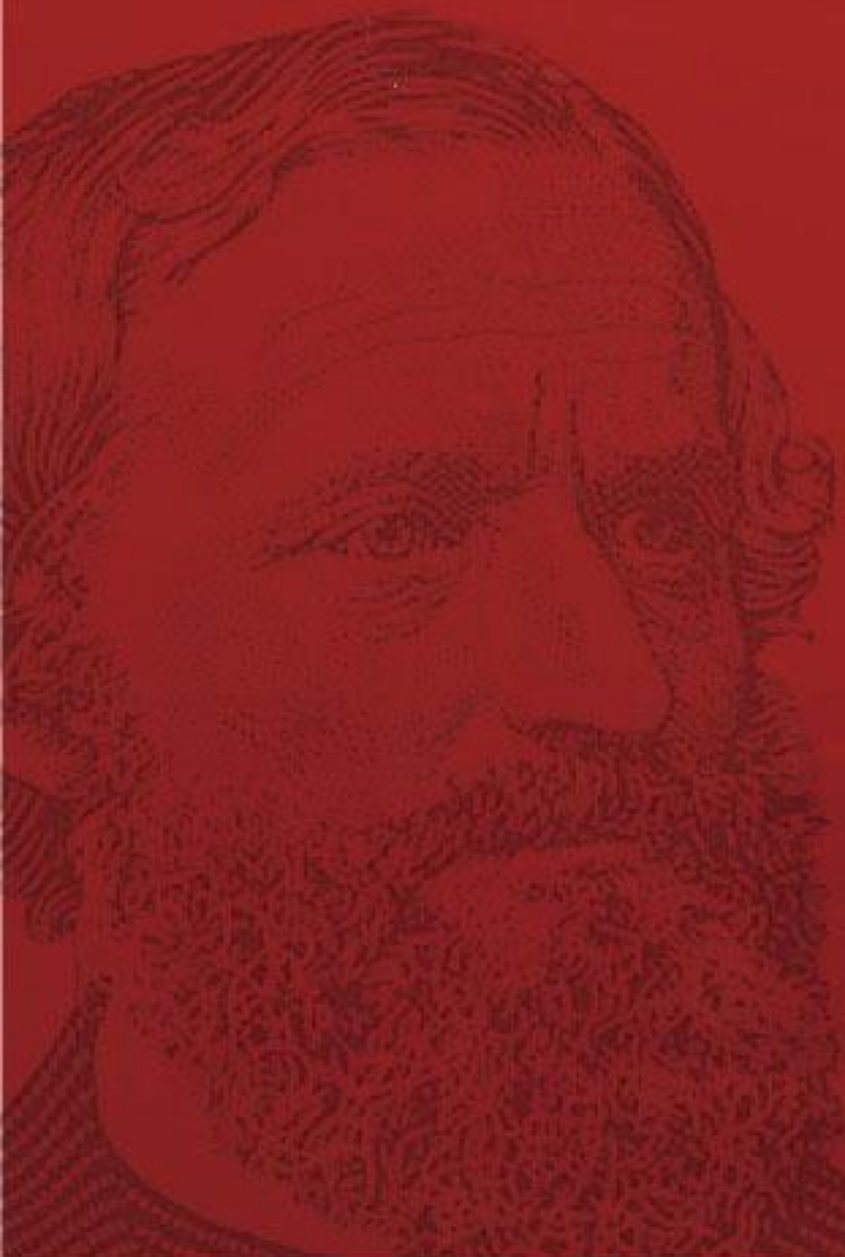
PROSERQUISA de C.V.

EQUIPO DE LABORATORIO DIDÁCTICO

"Excelencia en la experimentación científica"

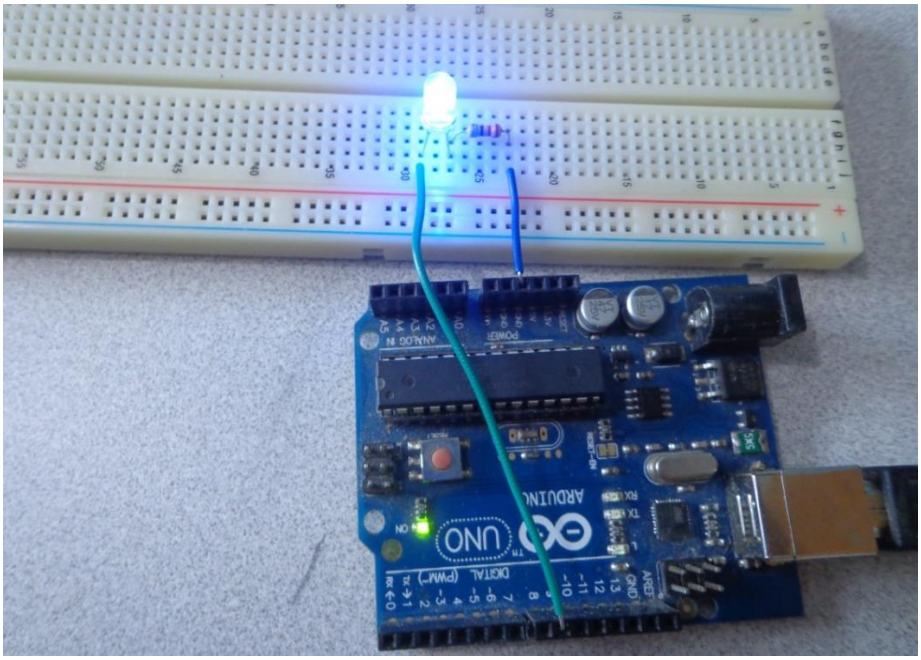
Curso de Arduino Leccion 1



Tel.: (503) 2273-2018
Fax: (503) 2273-4770
gerencia@proserquisa.net

Reparto y Calle Los Héroes No. 26-A,
San Salvador, El Salvador, Centroamérica

Tutorial 1. Introducción a Arduino
Controlando un LED.

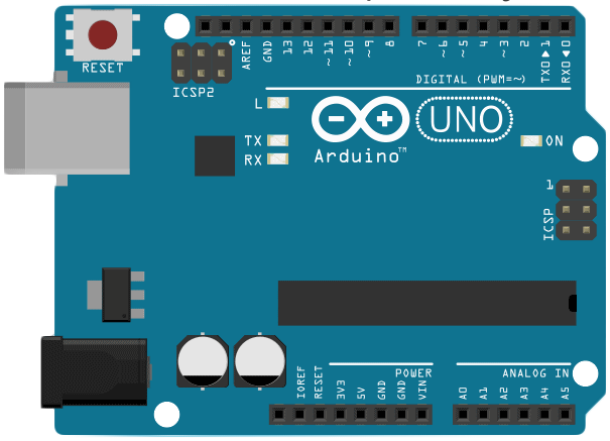


Objetivo general.

Conocer los conceptos básicos de Arduino para aplicarlos, encender y apagar un LED como primera práctica**.**

¿Qué es Arduino?

Arduino es un micro controlador de código abierto, una sencilla y económica placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo en el lenguaje de programación Processing basado en Java con una fácil curva de aprendizaje.



¿Qué puedo hacer con una placa Arduino?

Arduino nos permite crear proyectos de electrónica de una manera sencilla y eficaz. Siendo el limitante de su capacidad uno mismo referente a la gran gama de posibilidades para "crear" que nos permite Arduino. Actualmente hay una fuerte comunidad de desarrollo que presenta proyectos novedosos de código abierto y hay mucha documentación que nos respaldara a la hora de programar

Para Comenzar.

Nuestro "hola mundo en Arduino" será encender un LED haciendo una conexión simple y escribiendo un código muy sencillo.

Materiales

1 Arduino uno

1 Cable AB

1 Breadboard

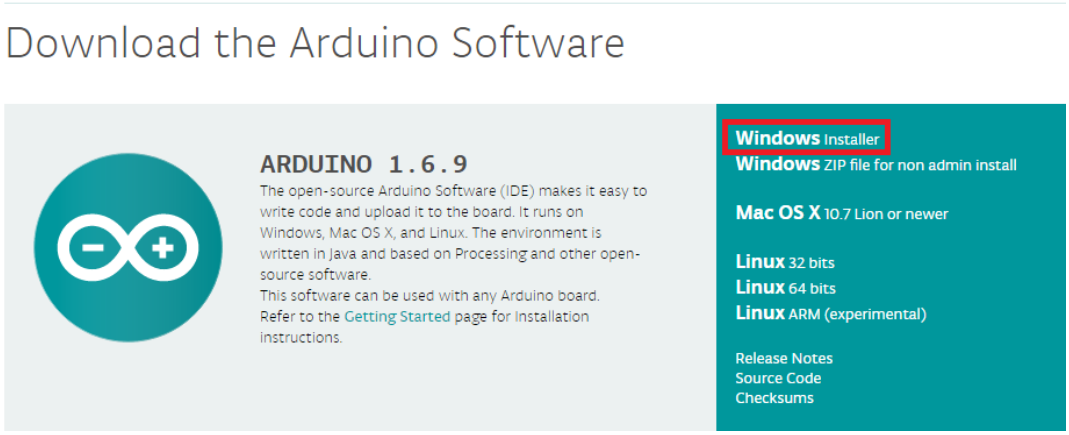
1 LED

Cables de conexión para Arduino (podemos usar UTP)

1 Resistencia 220 ohm

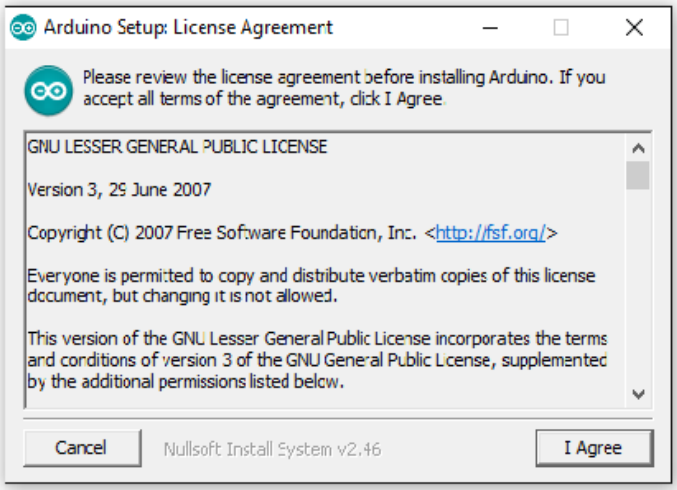
Paso 1: Instalación del IDE de Arduino

Lo primero que tenemos que hacer es instalar el software de Arduino en nuestra computadora, puedes descargarlo entrando a este link Arduino Software . Luego damos click en Windows installer (en mi caso uso Windows) usaremos el instalador porque trae algunas ventajas extra.

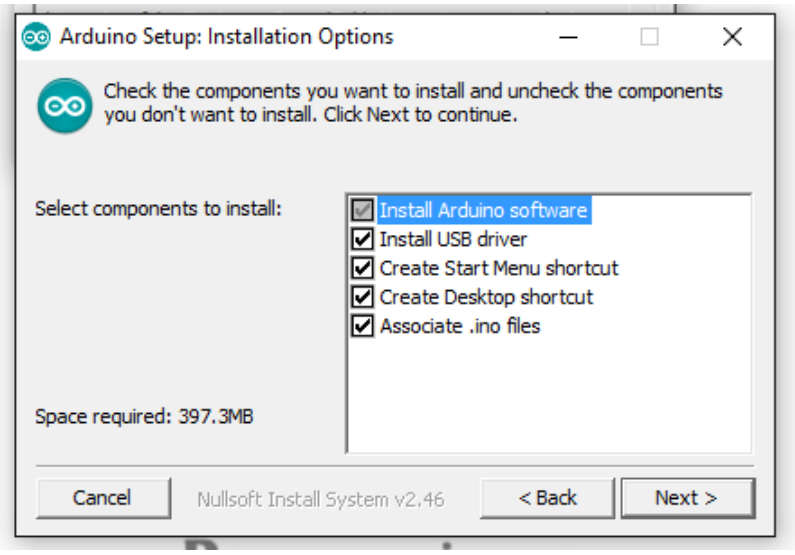


Luego click en "just download" y esperamos que se descargue.

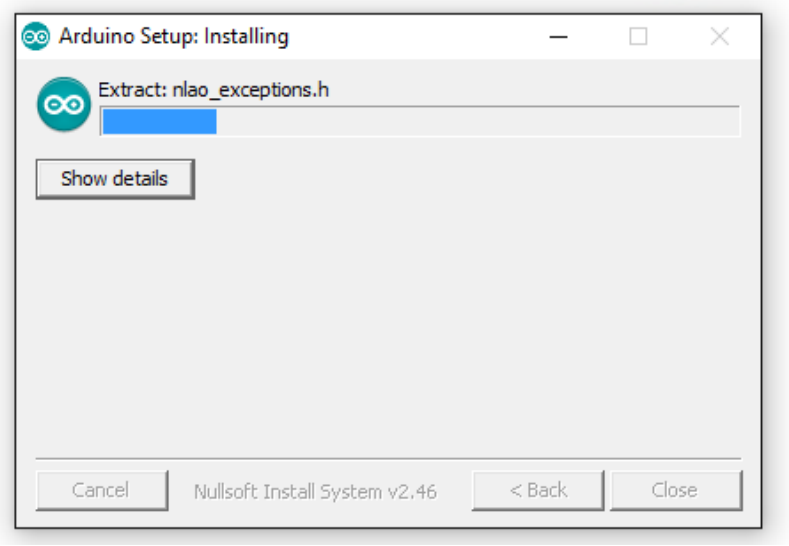
Luego de descargarlo, vamos al ejecutable arduino-1.6.9-windows.exe y damos Click derecho, y ejecutar como administrador.



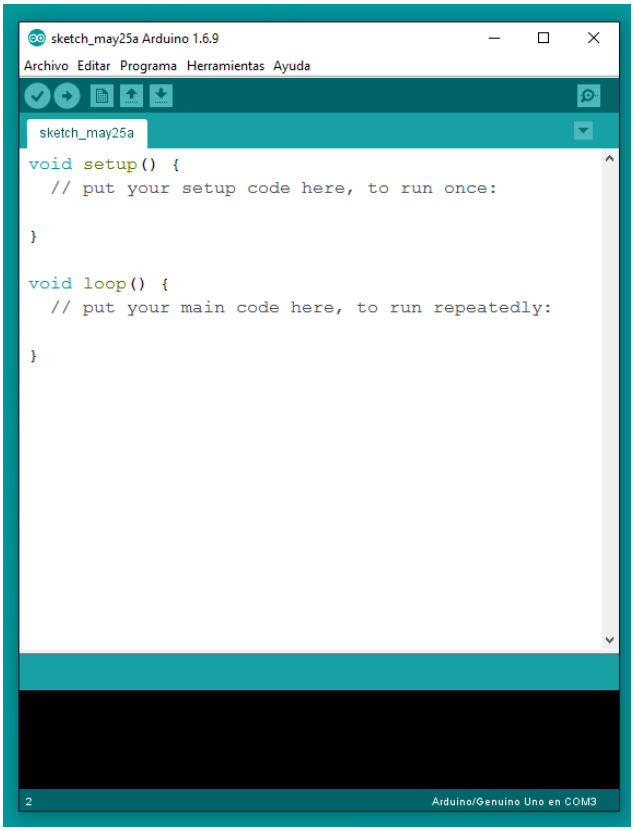
Click en I Agree, luego click en Next en esta parte observamos lo que está marcado, y podemos ver que de una vez nos instala los drivers USB de una vez.



Damos click en Next una vez más y esperamos a que termine la instalación y al final damos click en Close.



Terminamos la Instalación, ahora podemos abrir el IDE de Arduino para comenzar a programar. Continuamos abriendo el programa Arduino con el acceso directo creado en el escritorio



Paso 2; Empezar a programar.

Para poder empezar primero debemos entender como es la estructura básica de Arduino, que parámetros necesitamos escribir antes de escribir el código para nuestro proyecto.

La estructura básica de programación de Arduino es bastante simple y divide la ejecución en dos partes: setup y loop. Setup() constituye la preparación del programa y loop() es la ejecución. En la función Setup() se incluye la declaración de variables y se trata de la primera función que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar el pinMode (p. ej. si un determinado pin digital es de entrada o salida) e inicializar la comunicación serie. La función loop() incluye el código a ser ejecutado continuamente (leyendo las entradas de la placa, salidas, etc.).

```
void setup() {
  pinMode(pin, OUTPUT);       // Establece 'pin' como salida
}
void loop() {
   digitalWrite(pin, HIGH);       // Activa 'pin'
   delay(1000);             // Pausa un segundo
   digitalWrite(pin, LOW);        // Desactiva 'pin'
   delay(1000);
}
```

Como se observa en este bloque de código cada instrucción acaba con ; y los comentarios se indican con //. Al igual que en C se pueden introducir bloques de comentarios con /* ... */.

Variables

Una variable debe ser declarada y opcionalmente asignada a un determinado valor. En la declaración de la variable se indica el tipo de datos que almacenará (int, float, long)

int inputVariable = 0;

Una variable puede ser declarada en el inicio del programa antes de setup(), localmente a una determinada función e incluso dentro de un bloque como pueda ser un bucle. El sitio en el que la variable es declarada determina el ámbito de la misma. Una variable global es aquella que puede ser empleada en cualquier función del programa. Estas variables deben ser declaradas al inicio del programa (antes de la función setup()).

```
int  v=9;                          // 'v' es visible en todo el programa.
void setup() {
 pinMode(v,OUTPUT).       //configuramos la variable entera "v) o el pin 8, como una salida
}
void loop() {
digitalWrite(v,HIGH);          //Escribimos que "v" envíe un pulso atreves de la constante HIGH.
}
```

Ya en práctica

Lo primero que haremos será declarar la variable entera donde nombraremos "Led" al pin 9 de Arduino. De esta manera cada vez que queramos utilizar el pin 9 lo llamaremos "Led"

int Led=9;

Luego necesitamos configurar a "Led" como un pin de Salida en el Void Setup. Utilizando la función pinMode, **"Ojo tiene que estar escrita de la misma manera con M mayúscula"**, cuando se escribe de la forma correcta la función se colorea de naranja. Y tienes que fijarte que la variable este escrita exactamente de la misma forma o cuando compiles tu código, se generara un error.

Importante: "Ten en cuenta las mayúsculas y minúsculas que escribes".

```
void setup()
{
 pinMode(Led, OUTPUT);
}
```

El siguiente paso es configurar el void loop este es el código que se repetirá constantemente y aquí es donde programas lo que necesitas para que tu proyecto funcione. En nuestro caso necesitamos enviar un pulso al pin llamado Led para que pueda encender, usaremos la función **digitalWrite(Led,HIGH);** con este código le indicamos que Led escriba HIGH y de paso a los 5v. Seguidamente usaremos un **Delay** este nos servirá para agregar tiempo entre las funciones **se mide en milisegundos,** si ponemos 500ms, estaremos dándole medio segundo de duración. Acontinuacion apagaremos el Led cerrando el paso de voltaje con la función **digitalWrite(Led,LOW);** y agregamos otra espera de 500ms.

Esta programación se repetirá cíclicamente hasta que se desconecte el arduino, creamos un "Bucle" al cual le podemos seguir agregando más especificaciones de como funcionara.

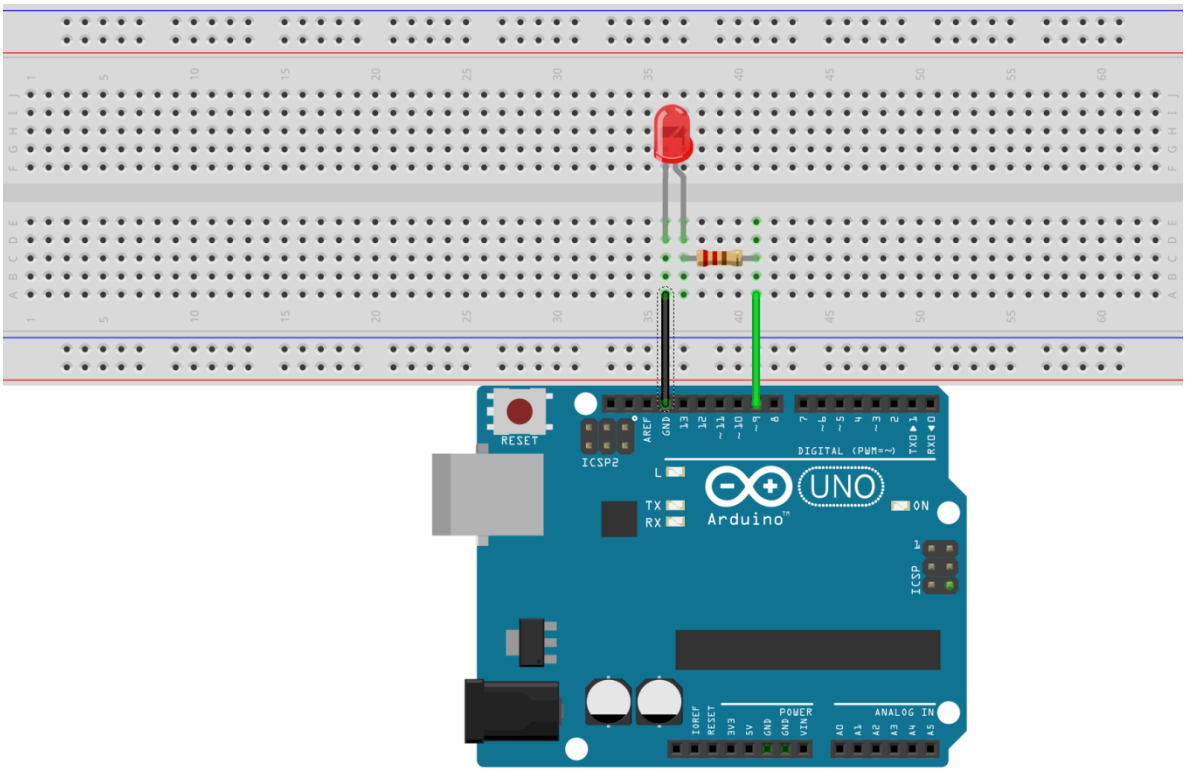
```
void loop() {
  digitalWrite(Led,HIGH);
  delay(500);
  digitalWrite(Led,LOW);
  delay(500);}
```

Al final el código se vera de esta manera.

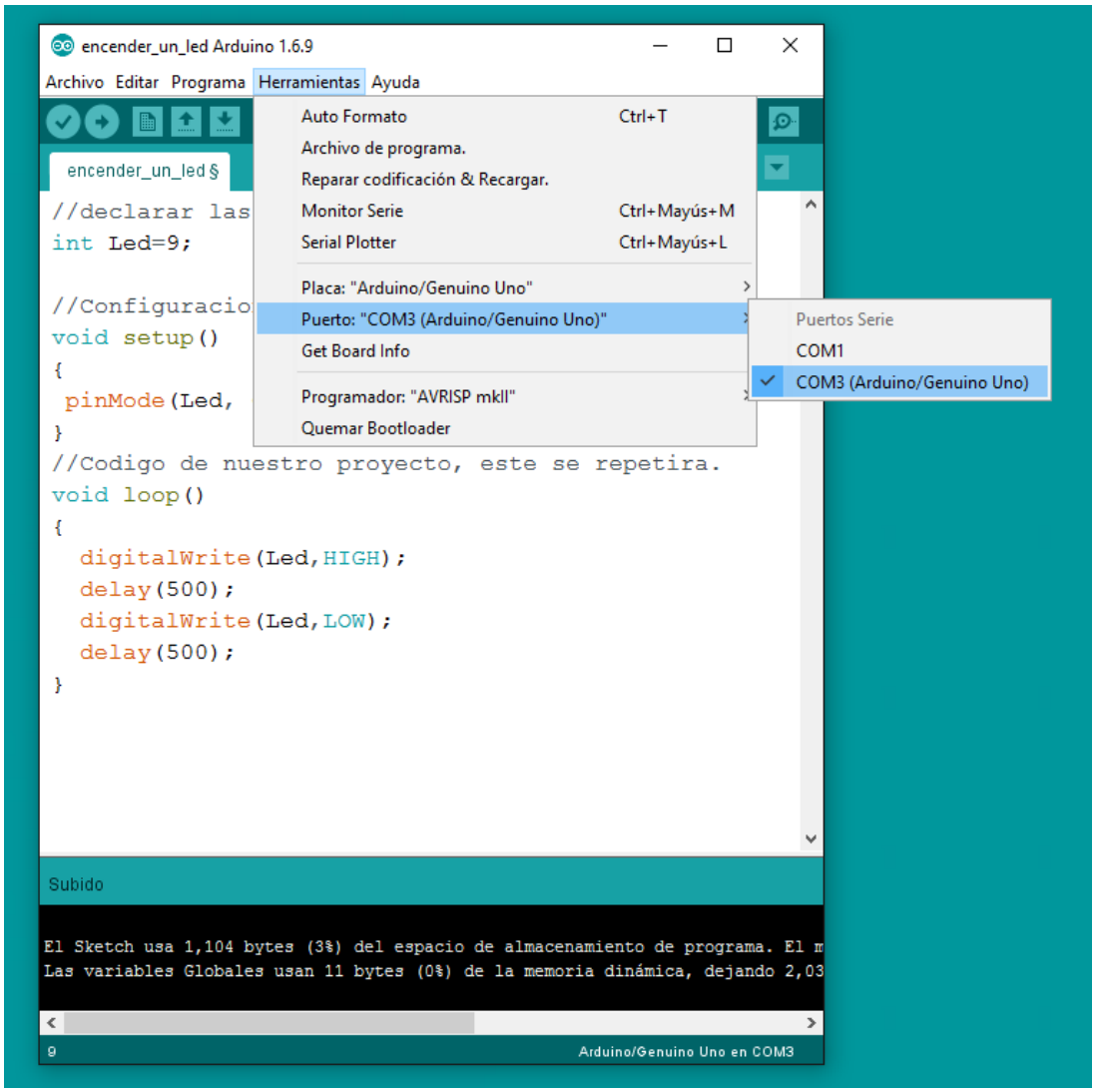
Importante: Tenemos que fijarnos que en la sintaxis del lenguaje que utiliza Arduino, es necesario siempre terminar las funciones con ";" indicando el cierre de esta para poder pasar a la siguiente, si no se cierran esto puede generar errores.

Paso-3: Diagrama de conexión.

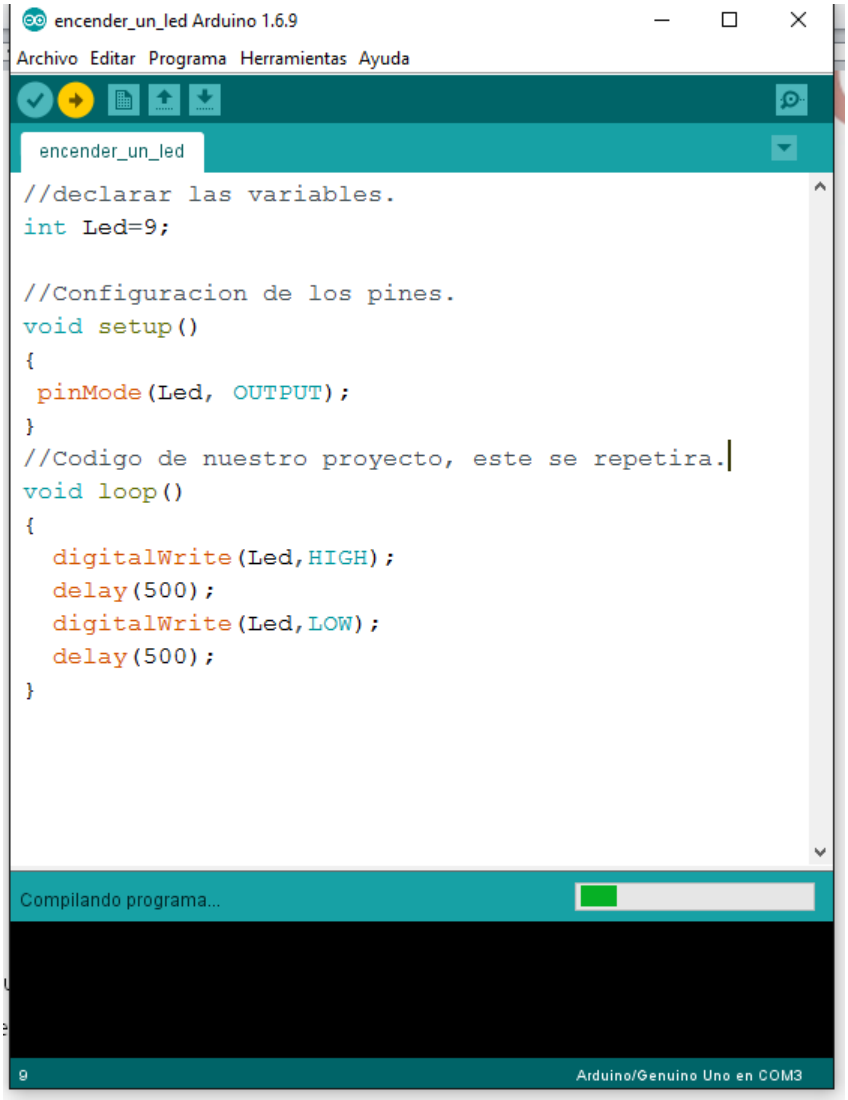
Lo que haremos en primer lugar es conectar el led a la Breadboard, los puntos verdes de vertical están conectados entre si, tomando en cuenta que la resistencia tiene que estar conectado al ánodo del led (la patita más corta) y continuamente conectamos al pin 9 en arduino, luego conectamos el cátodo del led al GND de arduino (Arduino uno cuenta con 3 GND)



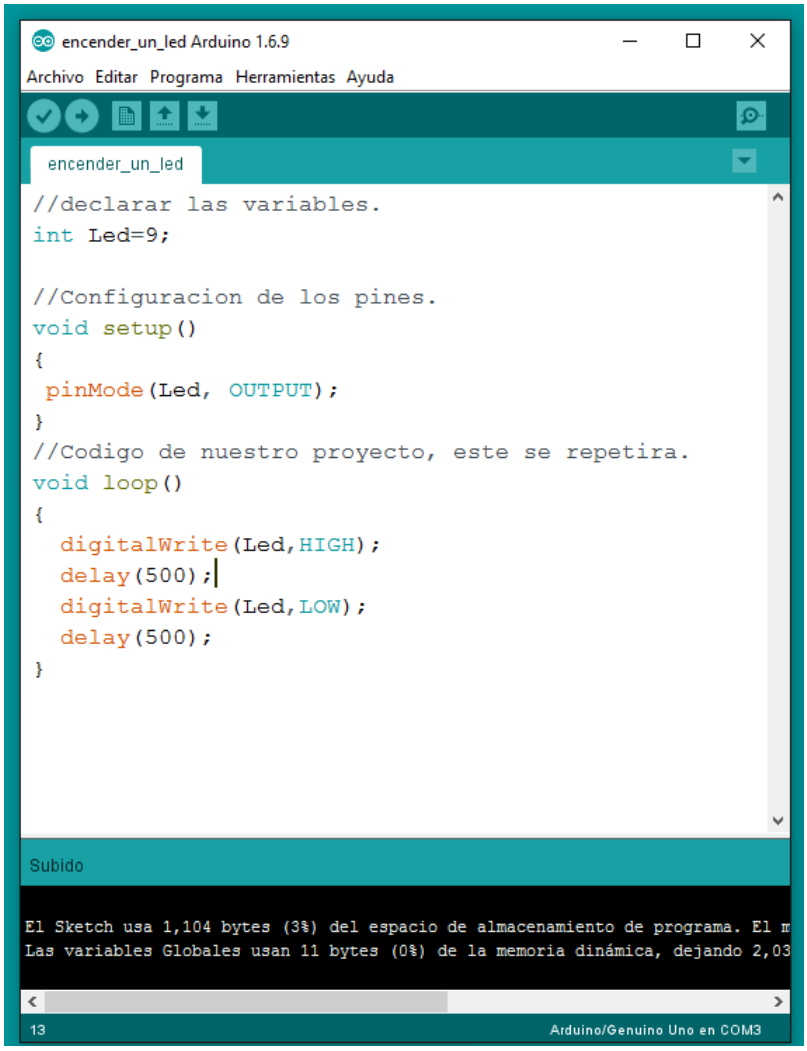
Luego de haber conectado correctamente todo, procedemos a cargar el código, para esto debemos conectar el Arduino a la computadora y seleccionar el puerto desde el IDE



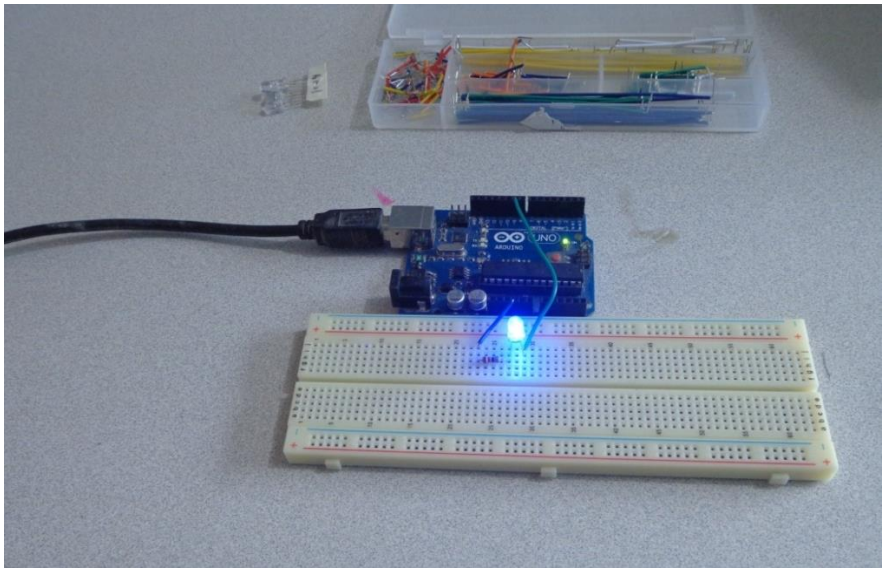
El puerto puede variar, en mi caso es el COM3. Luego procedemos a cargar el programa en el microcontrolador, dando click en **Subir** y esperamos a que se cargue el programa antes compilado. Se pueden producir diferentes errores en el proceso, es allí donde tenemos que revisar que todo esté bien escrito, si cerramos con ";" cada función, si las funciones de la estructura principal están cerradas con sus llaves, si las variables están bien escritas, el IDE nos muestra un mensaje en la parte inferior, donde podemos revisar en que línea se detectó un error.



Cuando el programas está listo, automáticamente el led comienza a encender y apagar tal y como se lo programamos.



Como práctica adicional, podemos jugar un poco con los delays. Intenta configurar los tiempos, que vaya más rápido o más lento, incluso puedes hacer diferentes secuencias, y agregar más leds. Acuérdate que tienes que declarar las variables primero luego configurar tus pines como salidas y por ultimo configurar uno por uno para que realicen las funciones que le programes.



Sección de preguntas.

1. **¿Para qué sirve la función Delay en el lenguaje de Arduino?**
2. **¿Para qué sirve la resistencia conectada con el LED a Arduino?**
3. **¿Cuáles son las ventajas de que Arduino sea de código abierto?**

1 La función delay nos permite hacer una pausa en el código, con la que podemos determinar la duración de cada función.

2 Para proteger al led, ya que no todos los leds pueden resistir 5v por si solos.

3 entre algunas están: puedes usarlo gratuitamente y modificar lo que sea, hay una fuerte comunidad de desarrollo que comparte conocimiento de arduino gratis.