Tutorial de

ARDUINO

OpenWebinars.Net

Tutorial escrito por Fernando Martínez y Guillermo Pérez

ARDUINO

Tutorial Arduino

OpenWebinars

Introducción

Bienvenidos a este tutorial sobre Arduino. La primera entrega se compondrá de una serie de post's muy sencillos y fáciles de seguir con los que se pretende acercar y dar una introducción a Arduino.

Si eres de los que no tienes ni idea de programación y electrónica... éste es tu tutorial. Para seguirlos no necesitas conocimientos previos ya que se explica todo paso a paso apoyado con imágenes y vídeos. Además, cuando hayas seguido este tutorial, no sólo podrás realizar por ti mismo los ejemplos aquí realizados sino que también podrás llevar a cabo proyectos compartidos por la comunidad de usuarios de Arduino y, ¿porqué no? alguno de tu propia invención.

ACCEDE AHORA A LOS MEJORES CURSOS

¿Qué es Arduino?

Arduino es una placa controladora y su entorno de programación que permiten de manera rápida y sencilla realizar proyectos de electrónica, automatismo, control, domótica, etc. Arduino nació en el Instituto IVREA (Italia) para facilitar a sus alumnos las tareas de programación de microcontroladores. Es ese uno de los motivos de su éxito: cualquier persona con pocos conocimientos de informática y electrónica puede programarlo e implementarlo.

Existen varios modelos de Arduino como son Uno, Leonardo, Mega... pero este tutorial se limitará salvo que se diga lo contrario al modelo Uno por ser el más común y económico.

Hardware libre

El otro motivo del gran éxito de esta plataforma podemos encontrarlo en que es desarrollado como hardware libre. ¿qué es eso de hardware libre? Pues es lo análogo al software libre. Sus esquemas, especificaciones y planos están disponibles al público, por tanto cualquiera que acceda a ellos puede fabricarse una copia idéntica al original.

ACCEDE AHORA A LOS MEJORES CURSOS

Pero no sólo eso, además, cualquier usuario o fabricante puede crear accesorios compatibles con lo que las posibilidades de realizar nuevos proyectos se multiplican. Esto atrae nuevos usuarios, algunos de los cuales crean nuevos accesorios y tienen nuevas ideas con lo que estamos realimentando una comunidad de usuarios y fabricantes que no para de crecer.

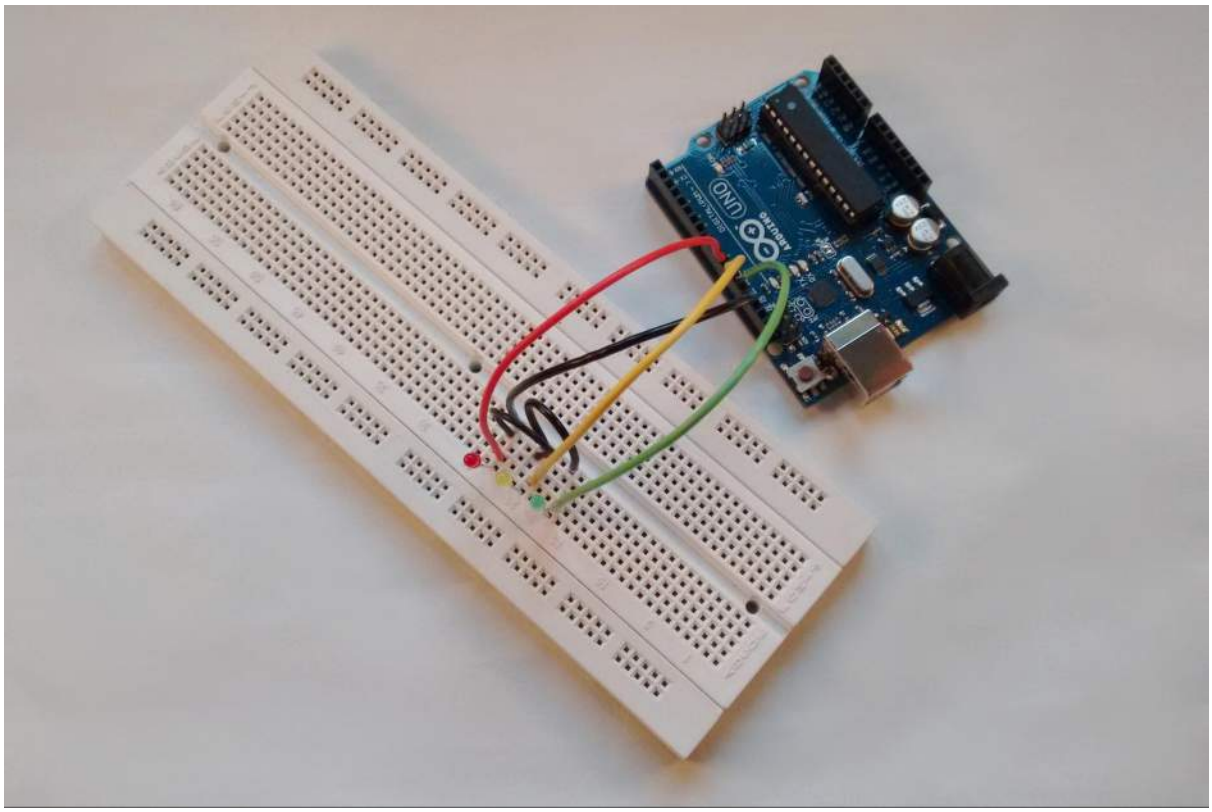
¿Para qué sirve?

Si estás leyendo esto seguramente es porque ya le has visto alguna aplicación y te ha picado el gusanillo. Aquí veremos algunas aplicaciones aunque las posibilidades son tantas como las que permite cualquier placa con microcontrolador: infinitas.

En general, podemos decir que un microcontrolador se utiliza para ordenar una serie de acciones en función de las entradas que le lleguen. Por ejemplo, encender un ventilador cuando la lectura de temperatura supere un cierto valor. Otras veces su tarea es realizar una serie de acciones sucesivamente sin hacer ninguna lectura. Éste caso sería el de utilizar Arduino para encender alternativamente las luces de un semáforo.

Cuando acabes el tutorial seguro que se te ocurren decenas de proyectos que podrás realizar.

ACCEDE AHORA A LOS MEJORES CURSOS



Partes de Arduino

En este apartado veremos las distintas partes que conformar nuestro Arduino como son entradas, salidas, alimentación, comunicación y shields.

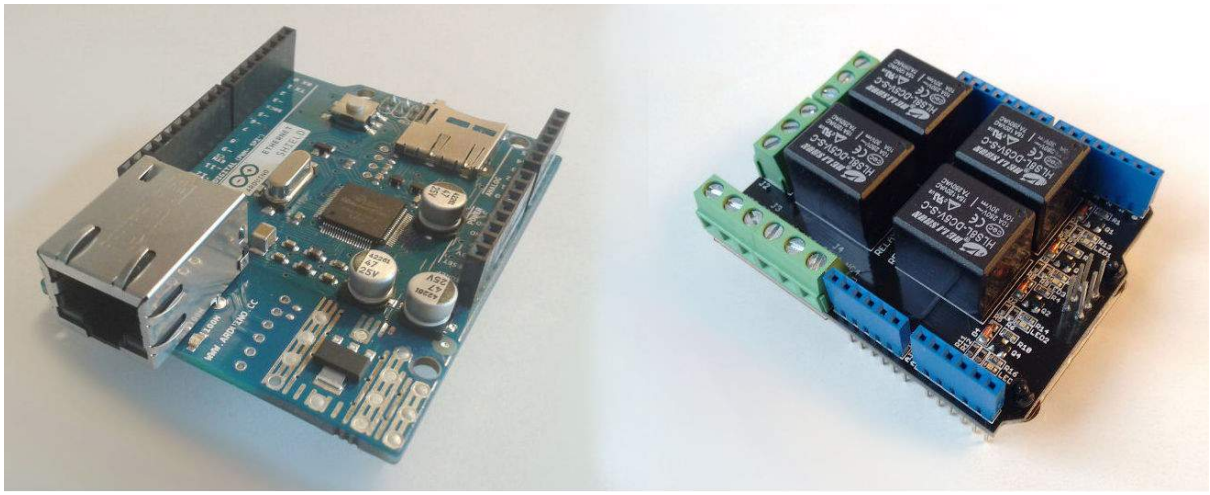
- Entradas: son los pines de nuestra placa que podemos utilizar para hacer lecturas. En la placa Uno son los pines digitales (del 0 al 13) y los analógicos (del A0 al A5).
- Salidas: los pines de salidas se utilizan para el envío de señales. En este caso los pines de salida son sólo los digitales (0 a 13).
- Otros pines: también tenemos otros pines como los GND (tierra), 5V que proporciona 5 Voltios, 3.3V que proporciona 3.3 Voltios, los pines REF de referencia de voltaje, TX (transmisión) y RX (lectura) también usados para

ACCEDE AHORA A LOS MEJORES CURSOS

comunicación serial, RESET para resetear, Vin para alimentar la placa y los pines ICSP para comunicación SPI.

- Alimentación: Como hemos visto el pin Vin sirve para alimentar la placa pero lo más normal es alimentarlo por el jack de alimentación usando una tensión de 7 a 12 Voltios. También podemos alimentarlo por el puerto USB pero en la mayoría de aplicaciones no lo tendremos conectado a un ordenador.
- Comunicación: En nuestros tutoriales nos comunicaremos con Arduino mediante USB para cargar los programas o enviar/recibir datos. Sin embargo no es la única forma que tiene Arduino de comunicarse. Cuando insertamos una shield ésta se comunica con nuestra placa utilizando los pines ICSP (comunicación ISP), los pines 10 a 13 (también usados para comunicación ISP), los pines TX/RX o cualquiera de los digitales ya que son capaces de configurarse como pines de entrada o salida y recibir o enviar pulsos digitales.
- Shields: traducido del inglés significa escudo. Se llama así a las placas que se insertan sobre Arduino a modo de escudo ampliando sus posibilidades de uso. En el mercado existen infinidad de shields para cada tipo de Arduino. Algunas de las más comunes son las de Ethernet, Wi-Fi, Ultrasonidos, Pantallas LCD, relés, matrices LED's, GPS...

ACCEDE AHORA A LOS MEJORES CURSOS



Todo lo que necesitas para empezar

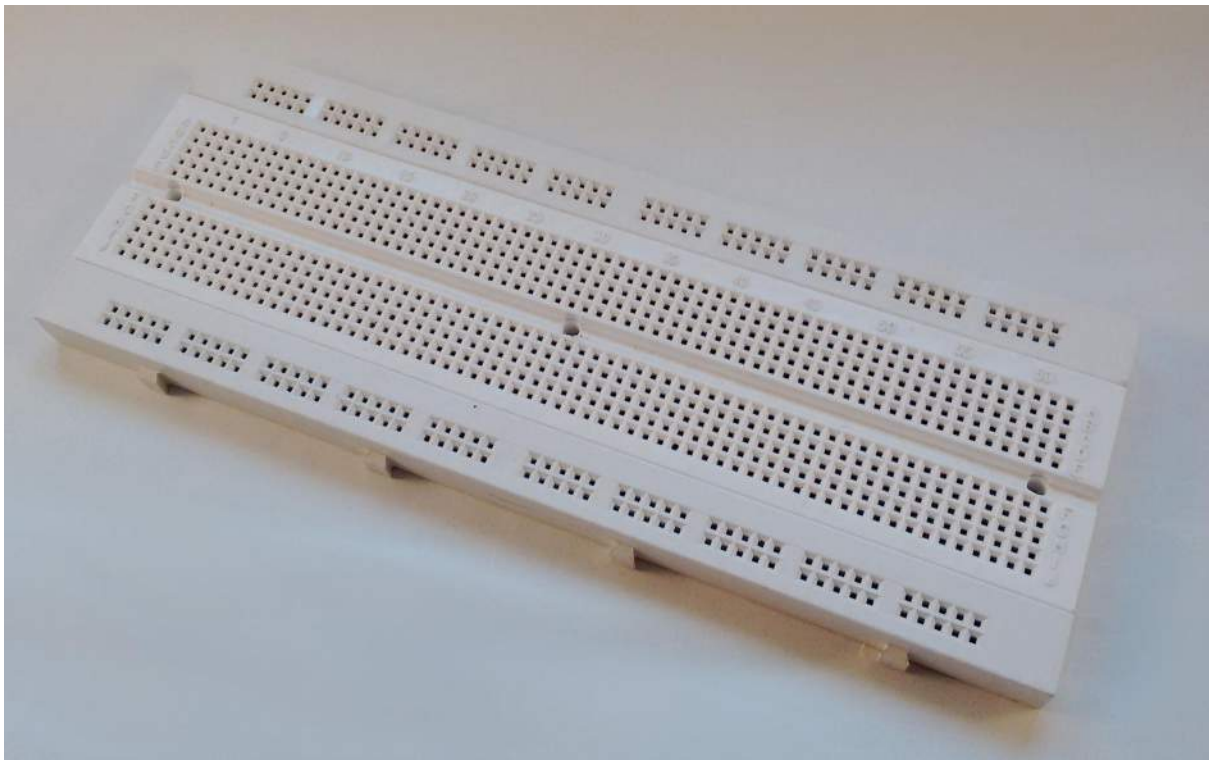
Una de las primeras cosas que se pregunta uno antes de empezar en este mundo es si el equipo es demasiado caro. La respuesta es que no, todo lo contrario.

Una placa Arduino Uno original cuesta 20,00€ y lo podéis pedir desde la propia web de Arduino. También tenemos placas de otros fabricantes por unos 11€ igualmente válidas gracias al carácter hardware libre de esta plataforma. Si no tenéis prisa, desde china también las podemos encontrar por unos 5€, pero pueden tardar más de un mes en llegarnos.

El entorno de programación IDE es totalmente gratuito y podemos descargarlo pinchando aquí.

Es muy aconsejable una protoboard o placa de prototipos para realizar las pruebas y montajes. Suelen valer entre 5€ y 10€.

ACCEDE AHORA A LOS MEJORES CURSOS



También nos facilitará el trabajo un juego de cables con pines macho para insertar en Arduino y en la protoboard. No suelen costar más de 3€. Una alternativa es pelar cables y estañarles las puntas para que no se deshilachen al meterlos en los agujeros.

Las shield tienen precios muy dispares y van desde los 5€ a 200€ o más, pero en este tutorial no haremos uso de ninguna de ellas.

Los componentes electrónicos adicionales de cada proyecto, como pueden ser resistencias, condensadores, fotoresistencias, led's, zumbadores... tienen en su mayoría precios de pocos céntimos y rara vez superan el euro.

En definitiva, por menos de 30€ podemos hacernos con un equipo que nos permitirá realizar multitud de proyectos.

Esto es todo de momento. En el siguiente post encontrarás toda la información para manejarte por el entorno de programación de Arduino y ejecutar tu primer programa.

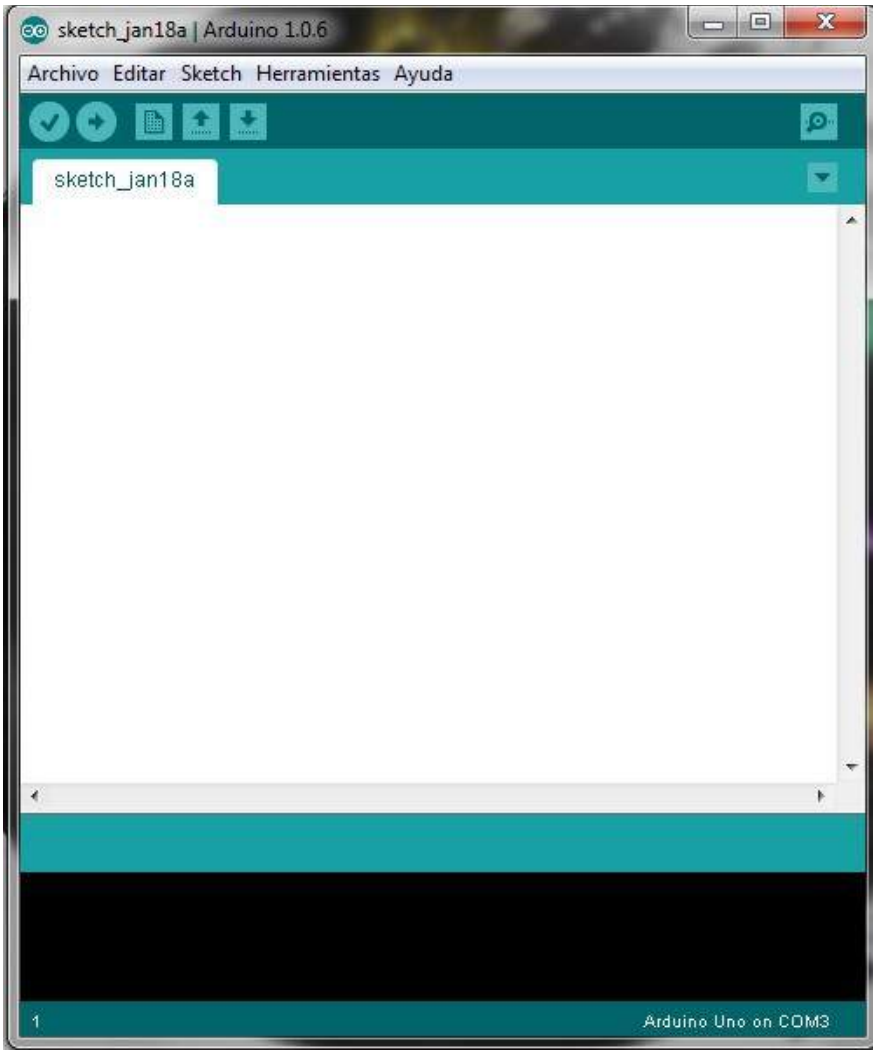
ACCEDE AHORA A LOS MEJORES CURSOS

IDE Aruduino

¿Qué es y para qué sirve?

Dado que el Arduino es como un pequeño ordenador que ejecuta una serie de códigos que previamente le hemos introducido, necesitaremos un programa para poder meter estos códigos a la propia placa. Este programa se llama IDE, que significa "Integrated Development Environment" ("Entorno de Desarrollo Integrado"). Este IDE estará instalado en nuestro PC, es un entorno muy sencillo de usar y en él escribiremos el programa que queramos que el Arduino ejecute. Una vez escrito, lo cargaremos a través del USB y Arduino comenzará a trabajar de forma autónoma.

ACCEDE AHORA A LOS MEJORES CURSOS

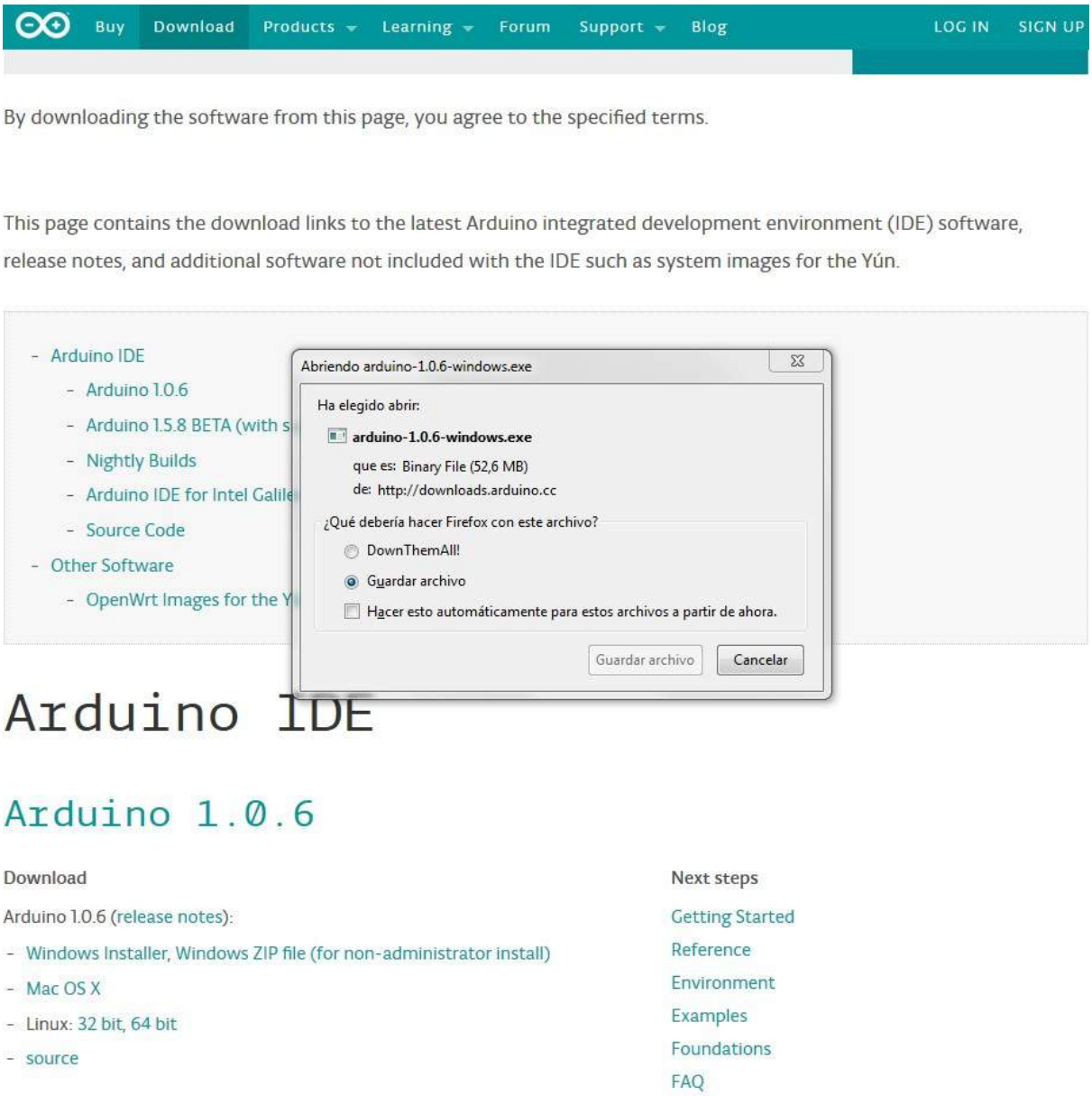


Descarga e instalación

Para instalar este IDE en vuestro sistema operativo solo tendréis que seguir unos sencillos pasos:

1. Descargar el programa gratuito "Arduino IDE" de su propia página web. Podéis hacer clic en este enlace o dirigíos en la web www.arduino.cc a la sección Donwnload.

ACCEDE AHORA A LOS MEJORES CURSOS



Hacéis clic en vuestro sistema operativo, después en guardar archivo y lo descargáis.

2. Instalación en Windows: (si vuestro sistema operativo es Linux podéis saltar al siguiente punto)

   Una vez descargado lo ejecutáis y podréis ver la siguiente ventana.

ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 3 Aceptación de condiciones.

Pulsamos en "I Agree". En este cuadro dejamos todas las opciones marcadas, pero hay que prestar especial atención a la opción de instalar los USB driver, ya que esto es muy importante para que la placa Arduino se pueda comunicar con el PC. Pulsamos en Next e Install

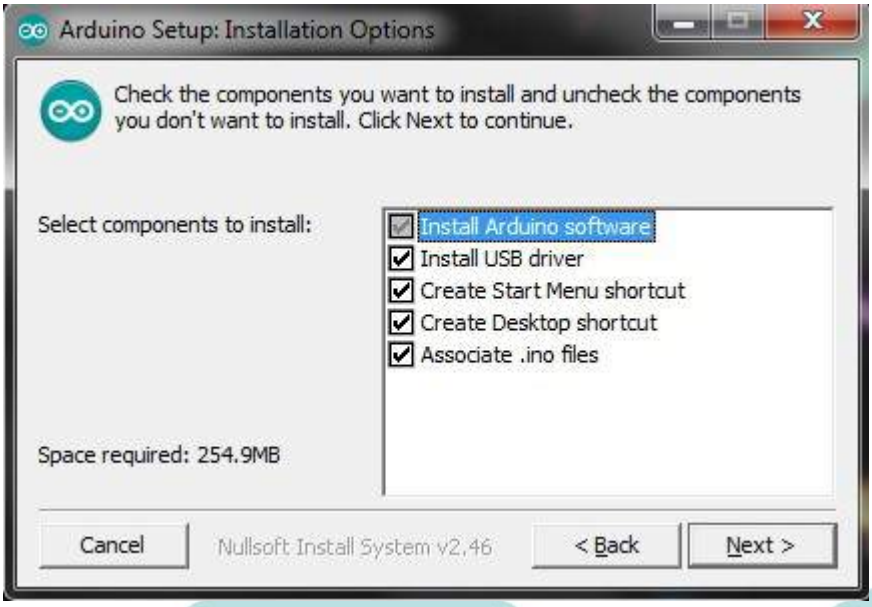


Ilustración 4 Selección de opciones de instalación.

ACCEDE AHORA A LOS MEJORES CURSOS

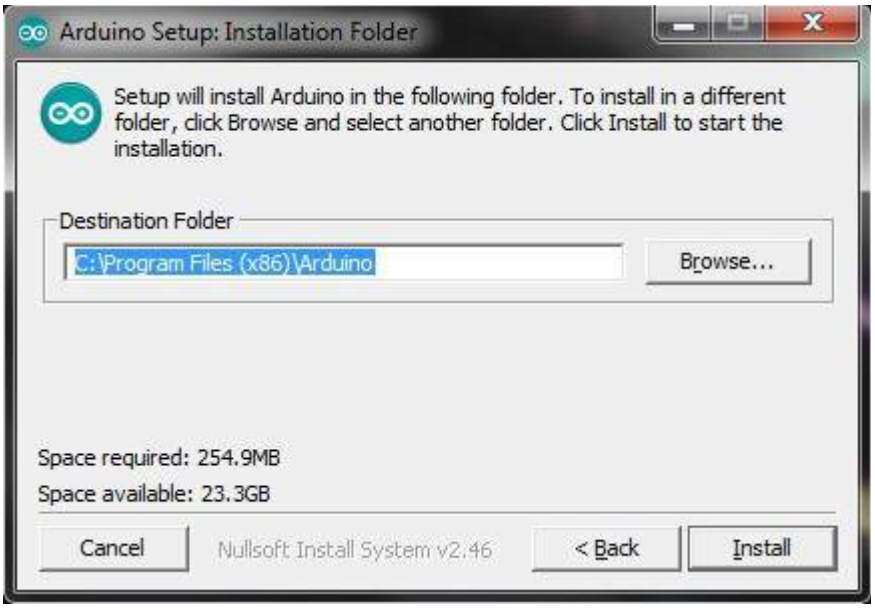


Ilustración 5 Selección de carpeta de instalación.

Y esperaremos que termine de instalar (si pregunta si deseamos instalar el software Arduino USB le damos a instalar)

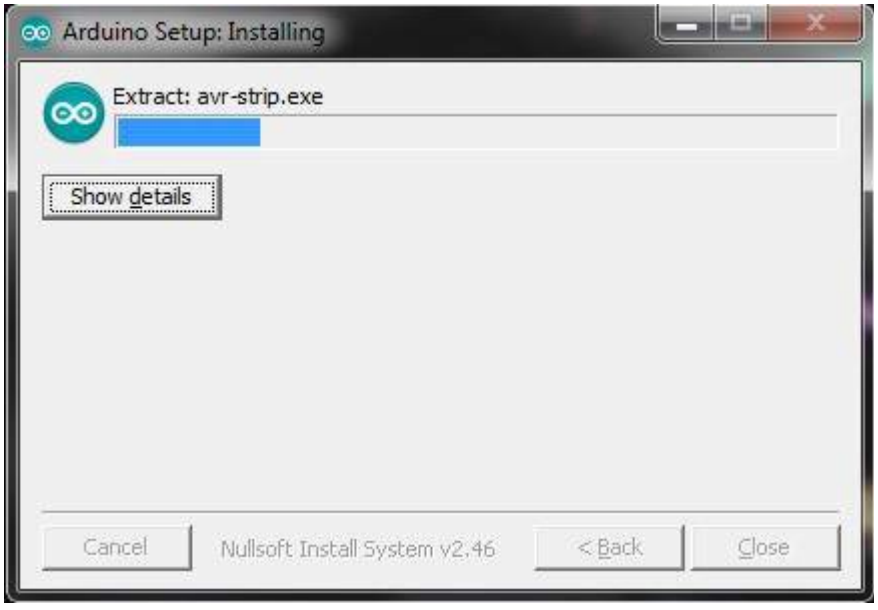


Ilustración 6 Proceso de instalación.

Una vez terminado el proceso, hacemos clic en Close y ya tendremos el IDE instalado en nuestro PC.

ACCEDE AHORA A LOS MEJORES CURSOS

3. Instalación en Linux

Hay dos formas de hacerlo:

a. A.En distribuciones como Ubuntu y derivados (Lubuntu, Linux Mint...) se puede instalar desde los repositorios al menos desde la versión de Ubuntu 14.04 (17 para Linux Mint) estando disponible Arduino-1.0.5. Para instalar abrimos el Gestor de paquetes Synaptic o el Centro de Software y buscamos Arduino

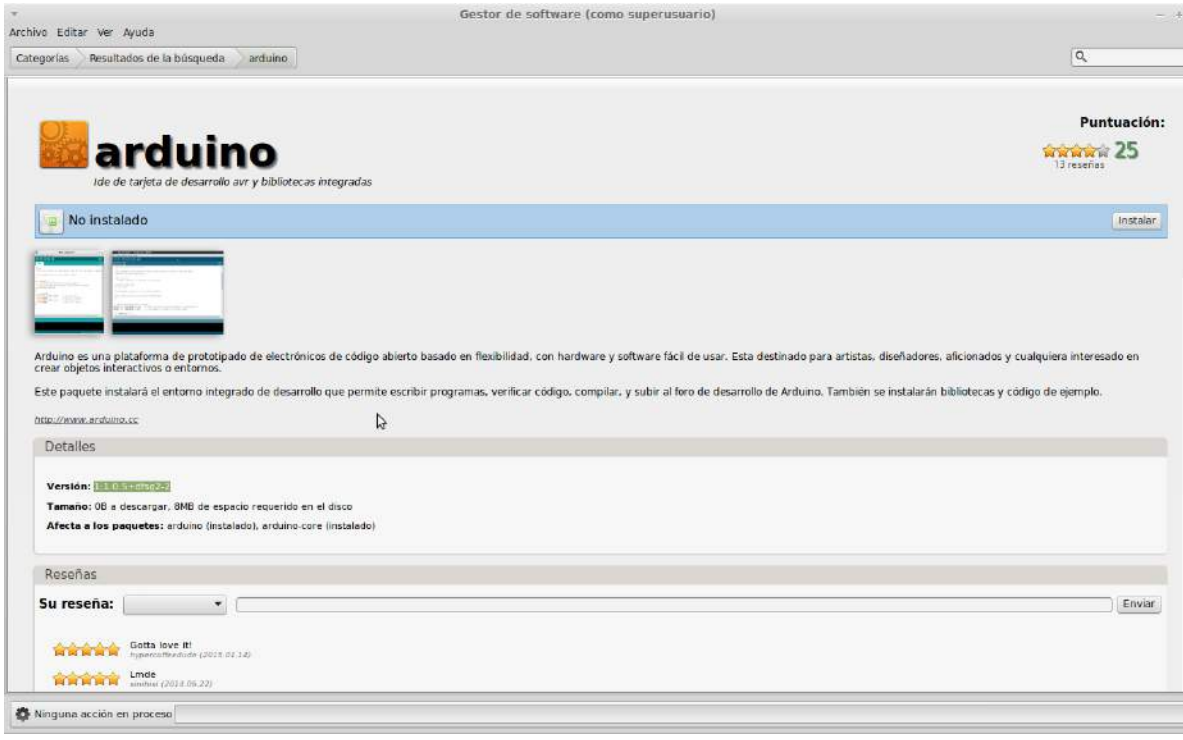


Ilustración 7 Gestor de software de Linux Mint

Pulsamos instalar y empezará el proceso de descarga e instalación.

b. Descargando e instalando el paquete desde la web de Arduino.

ACCEDE AHORA A LOS MEJORES CURSOS

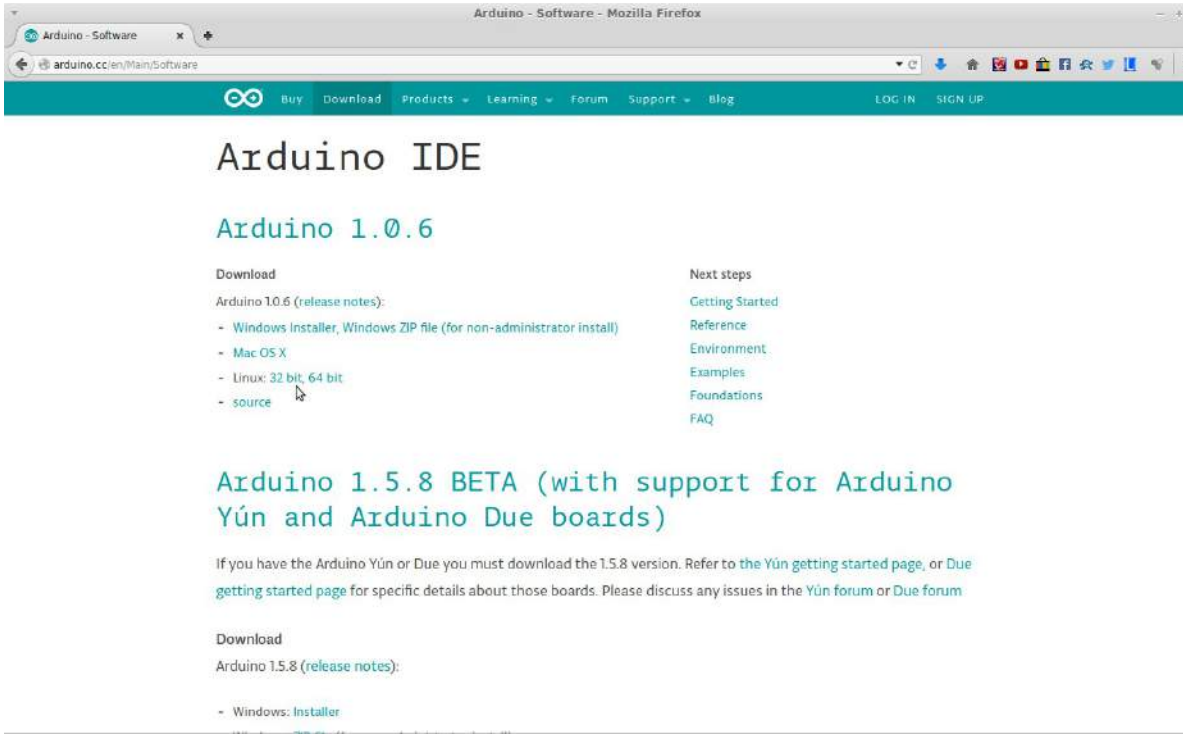


Ilustración 8 Apartado de descargas de Arduino

Descargamos la versión 32 o 64 bits dependiendo del sistema operativo. Después abrimos terminal y nos vamos a la carpeta donde lo hemos descargado. En nuestro caso lo descargamos en el Escritorio así que usamos el comando "cd Escritorio".

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

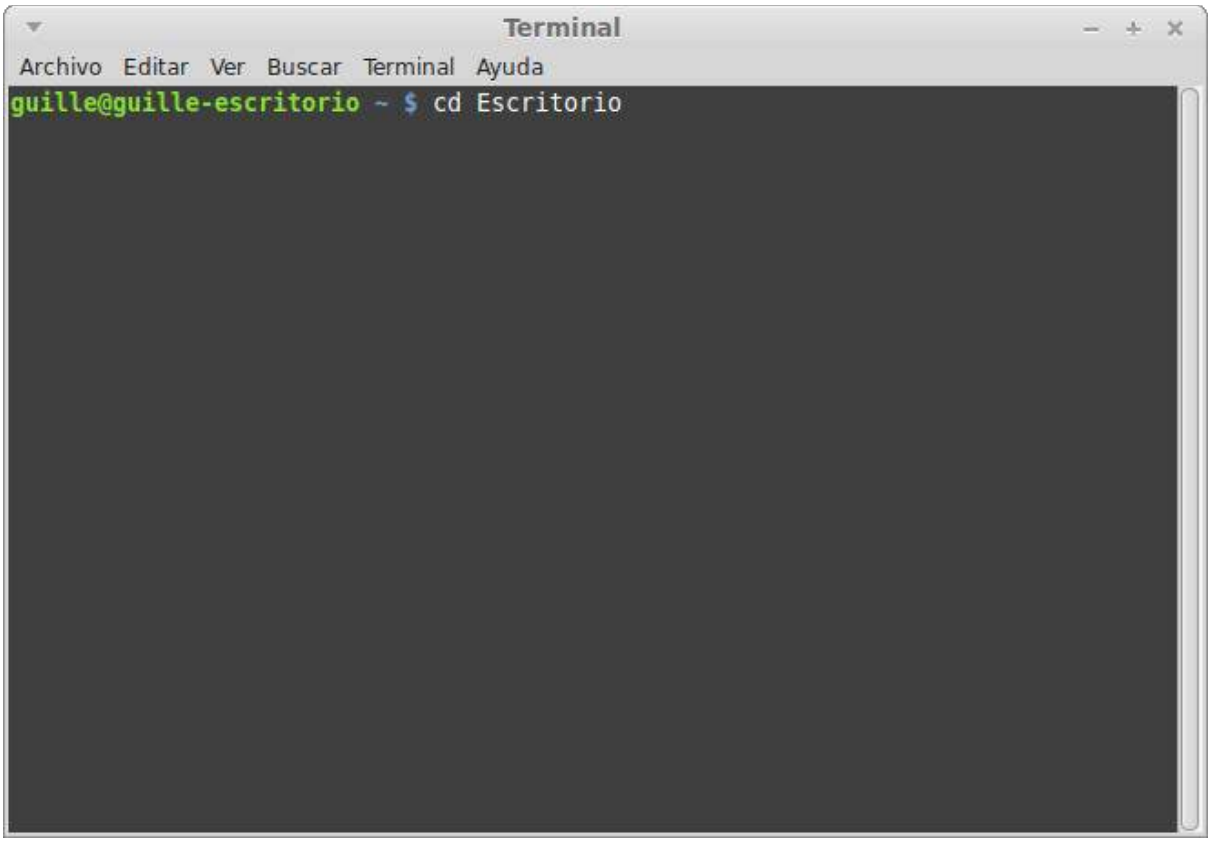


Ilustración 9 Cambiando el directorio de trabajo

Una vez que estemos en la carpeta donde lo hemos descargado, escribimos el comando "tar -zxvf arduino-1.0.6-linux32.tgz". (En general es "tar -zxvf nombrearchivo.tgz")

ACCEDE AHORA A LOS MEJORES CURSOS

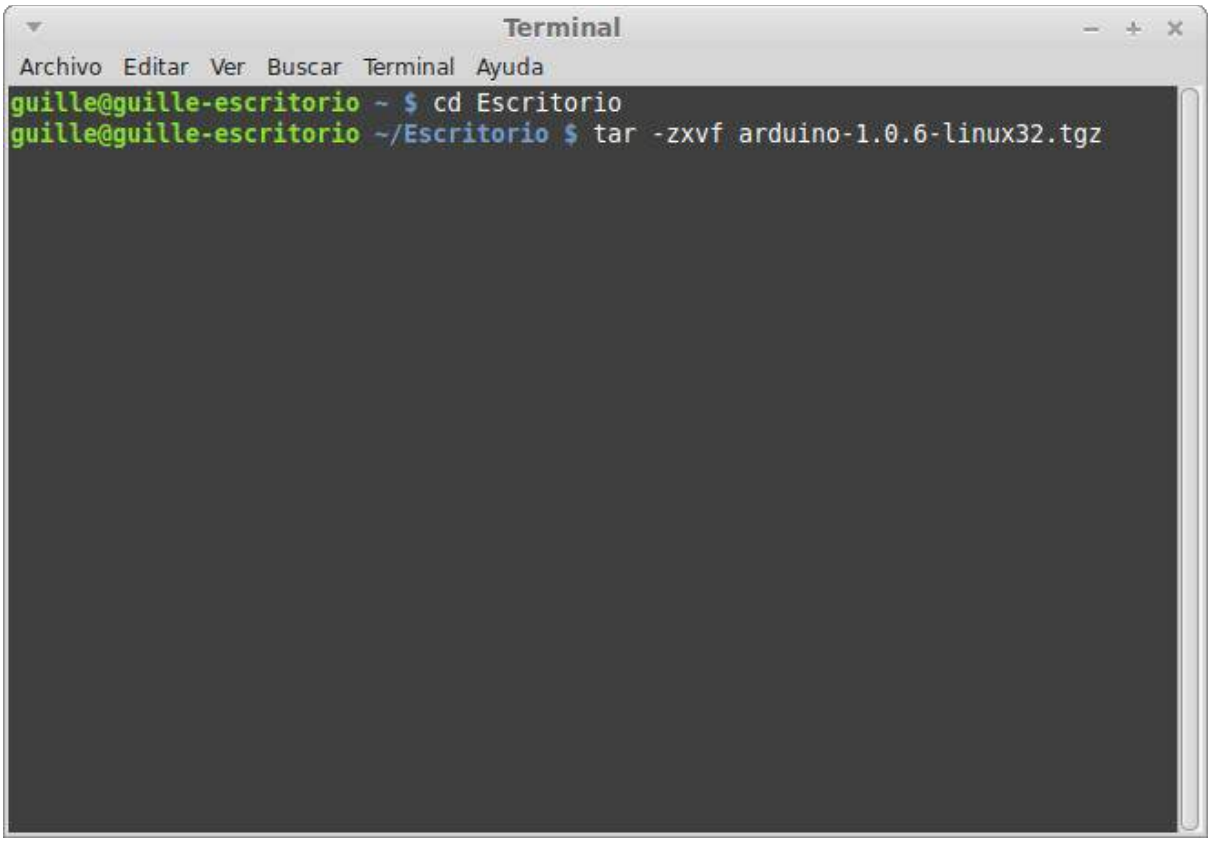


Ilustración 10 Cambiando el directorio de trabajo 2

Después de pulsar intro, empieza la instalación que dura pocos segundos. Al finalizar podemos comprobar que se ha creado un directorio llamado arduino1.0.6 (o versión descargada). Entramos y tenemos un ejecutable llamado "arduino"

ACCEDE AHORA A LOS MEJORES CURSOS

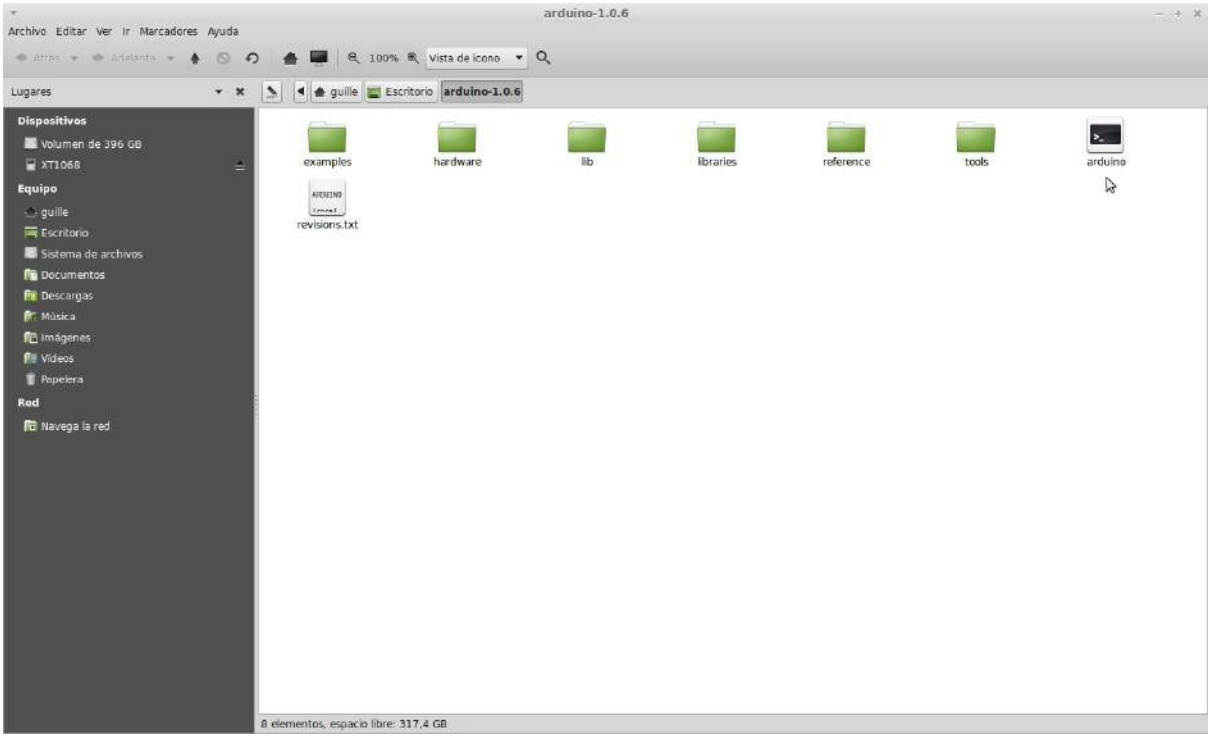


Ilustración 11 Captura explorador de archivos

Hacemos doble clic en ese archivo y, en la ventana que se abre, pulsamos "Ejecutar".

Entorno de programación y configuración

Al abrir el programa veremos cómo nos aparece la consola principal del IDE Arduino en la cual podemos ver las siguientes zonas

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

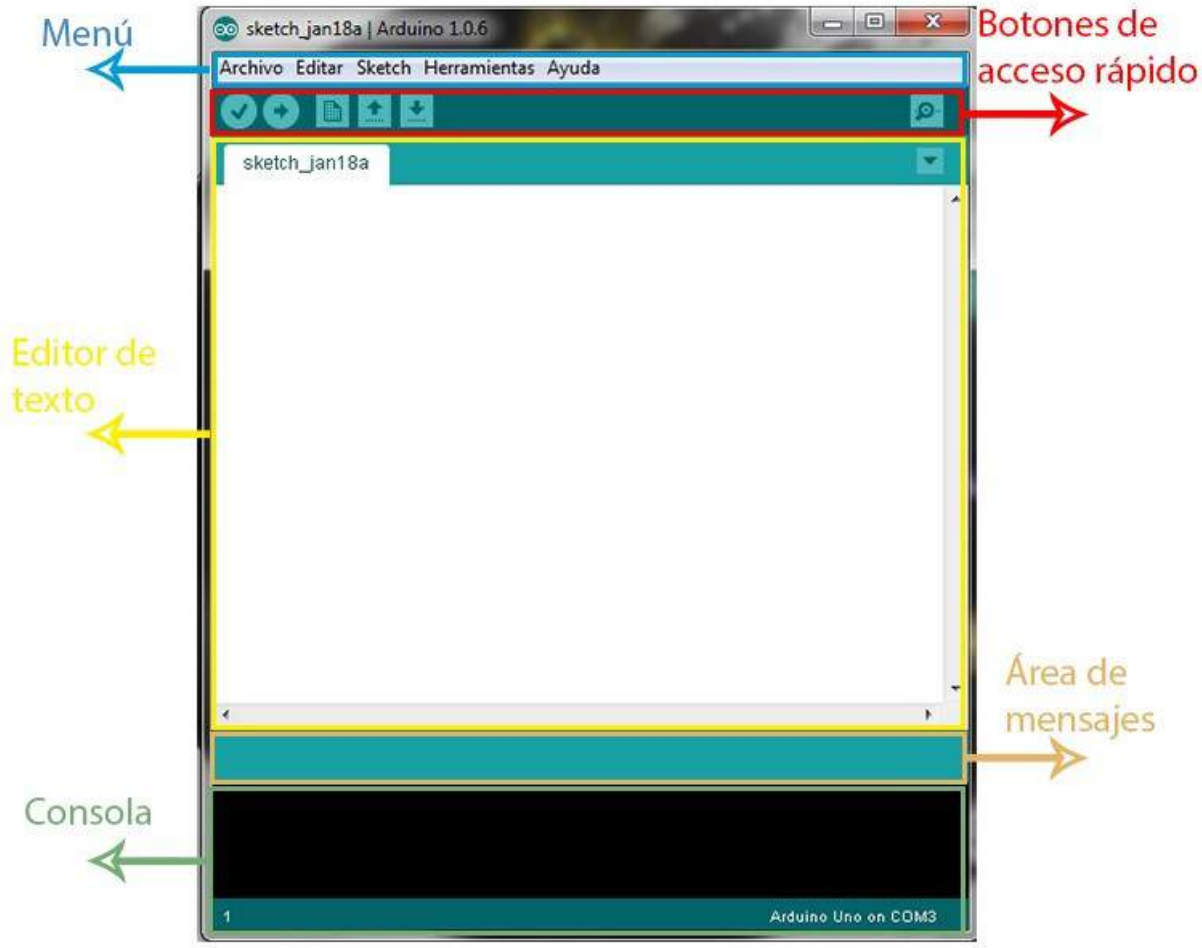


Ilustración 12 Zonas del IDE Arduino

En la parte de menú tenemos una zona para acceder a funciones como carga de archivos, edición del texto del código, carga de librerías y ejemplos, configuración, herramientas…etc.

En los botones de acceso rápido tenemos los siguientes iconos:

-  Verifica si tu programa está bien escrito y puede funcionar.

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

-  Carga el programa a la placa de Arduino tras compilarlo.
-  Crea un programa nuevo.
-  Abre un programa.
-  Guarda el programa en el disco duro del ordenador.
-  (En la parte derecha de la barra de herramientas se encuentra el Monitor Serial) abre una ventana de comunicación con la placa Arduino en la que podemos ver las respuestas que nuestro Arduino nos está dando, siempre que tengamos el USB conectado.

En el cuadro del editor de texto escribiremos el código del programa que queramos que Arduino ejecute.

Finalmente, en el área de mensajes y la consola Arduino nos irá dando información sobre si la consola está compilando, cargando…y sobre los fallos o errores que seproduzcan tanto en el código como en el propio IDE.

El siguiente paso que realizaremos será configurar nuestro IDE para que se comunique con nuestra placa Arduino. Para ello conectaremos nuestro Arduino mediante el cable USB al PC y después de que el sistema operativo haya reconocido e instalado la tarjeta automáticamente, nos dirigimos a la zona de menú, pulsamos en Herramientas y después en Tarjeta. Ahí seleccionamos el modelo de tarjeta Arduino que tengamos, en nuestro caso "Arduino Uno".

ACCEDE AHORA A LOS MEJORES CURSOS

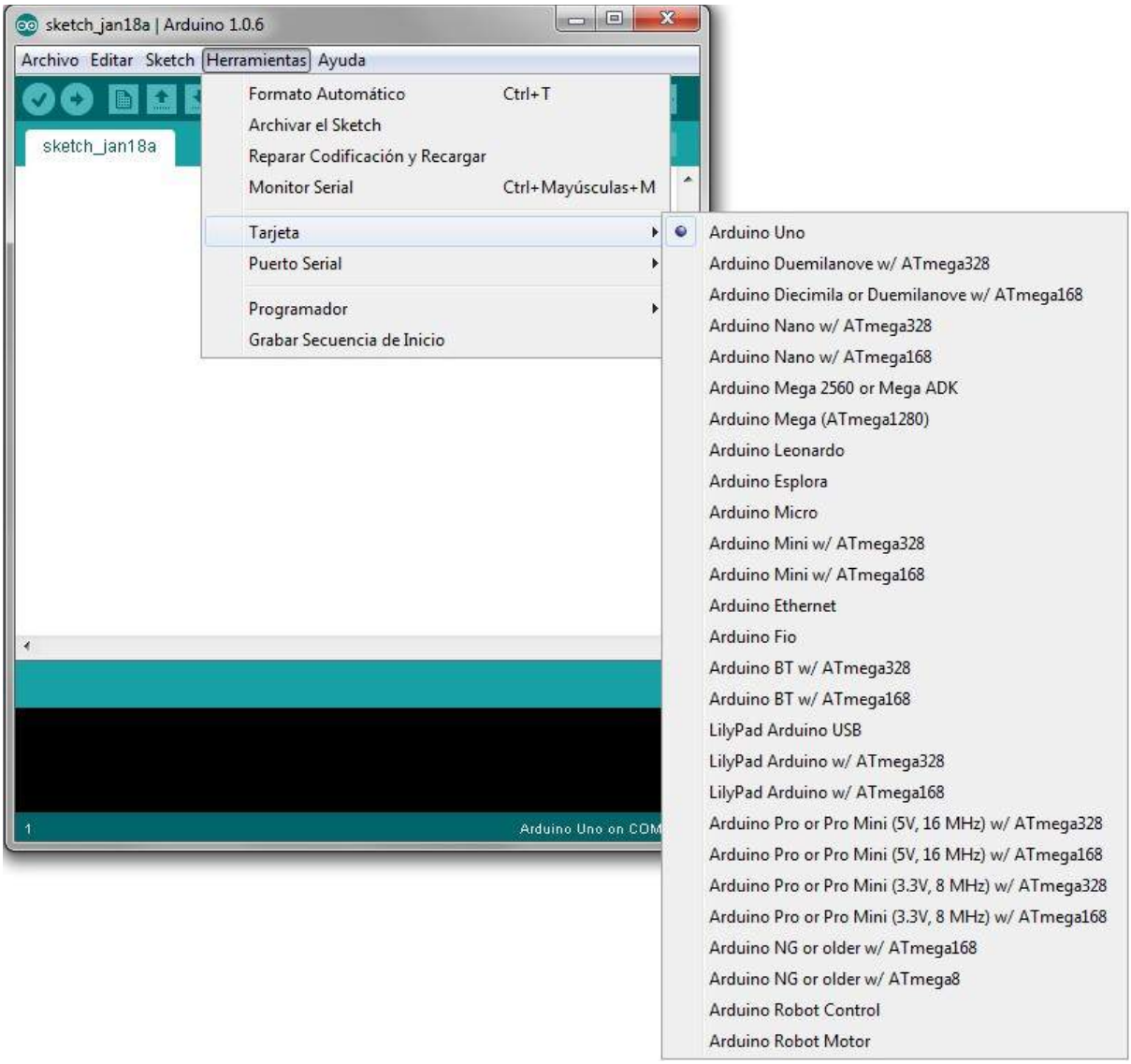


Ilustración 13 Selección de Tarjeta

Después vamos a la opción Puerto Serial y elegimos el COM en el que tenemos conectado nuestro Arduino.

ACCEDE AHORA A LOS MEJORES CURSOS

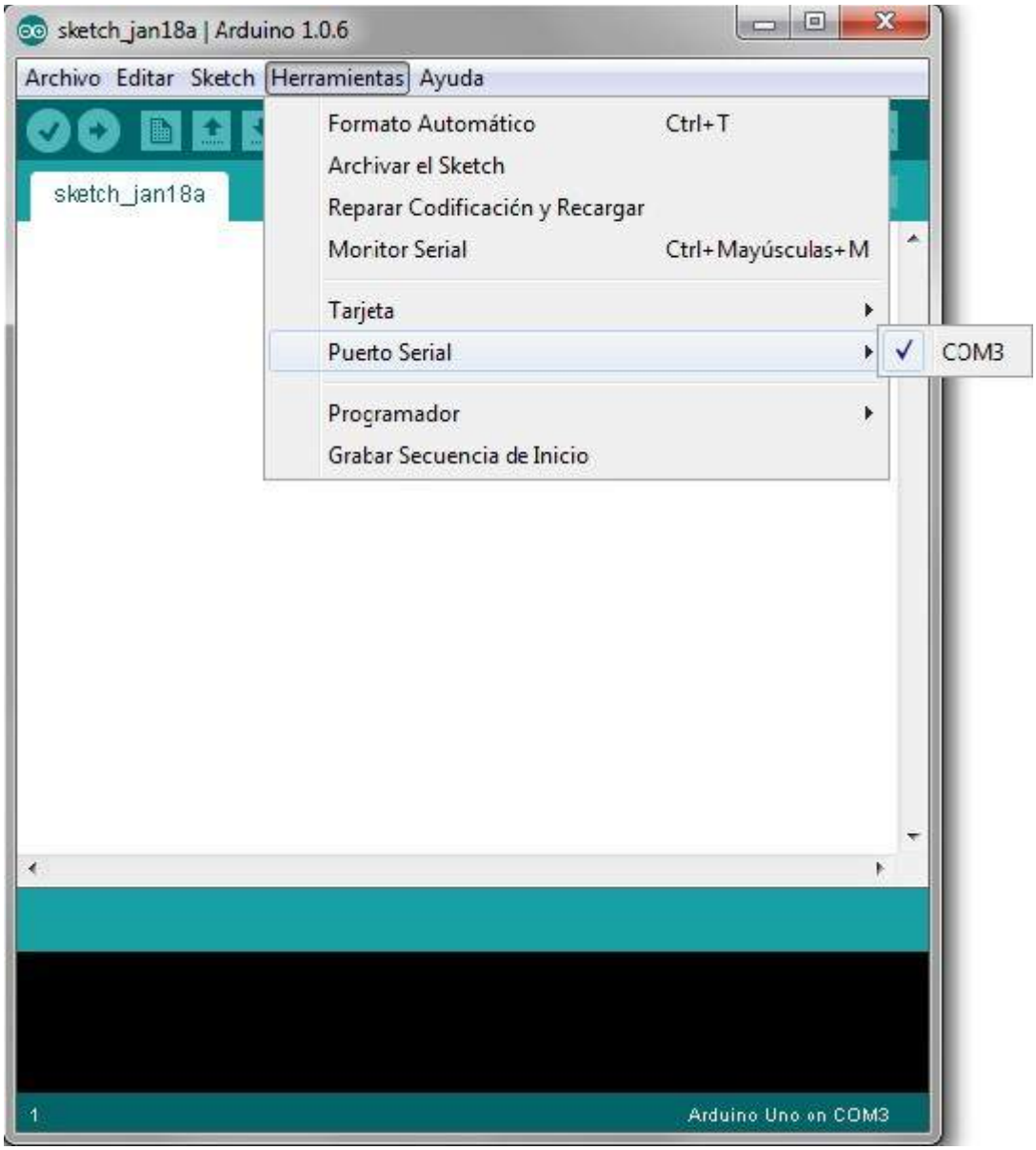


Ilustración 14 Selección de puerto COM

Si nos aparecieran varios COM activos, porque estemos usando otros dispositivos serial o por otro motivo, para saber cuál de ellos es el que se comunica con nuestra placa, solo tenemos que irnos a Panel de control/Hardware/Administrador de dispositivos.Miramos la pestaña (Puertos COM y LPT) y ahí nos aparecerá nuestro Arduino y el COM en el que está conectado. Con esto, ya podemos empezar a programar nuestro Arduino.

ACCEDE AHORA A LOS MEJORES CURSOS

Programa de testeo: "HELLO WORLD"

Para finalizar, probaremos que todo está correctamente instalado y configurado ejecutando nuestro primer programa, el más simple, el típico "HELLO WORLD" ("HOLA MUNDO"). Para ello solo tenéis que pegar el siguiente código en la zona del editor de texto del IDE Arduino:

Conectamos la placa Arduino al PC usando el cable USB y hacemos clic en el botón de cargar  . Si habéis seguido todo correctamente, podréis ver cómo el led de vuestro Arduino parpadea y si clicáis en el botón del monitor serial  veréis como Arduino os está escribiendo "HOLA MUNDO".

Espero que os haya resultado fácil. En el próximo post veremos la estructura de un programa, las sentencias más comunes y realizareis vuestro primer ejemplo práctico.

Ejemplo semáforo

ACCEDE AHORA A LOS MEJORES CURSOS

En este nuevo post daremos unas pautas sobre cómo debe estructurarse un programa en Arduino, también veremos cómo son las sentencias básicas usadas con esta IDE, cómo cargar nuestro programa a la placa y para finalizar, realizaremos un ejemplo con el que encenderemos leds con Arduino montando nuestro propio semáforo.

Estructura básica de los códigos

Un código Arduino es una serie de comandos de programación que le dirán a nuestro microcontrolador como configurarse al iniciarse y qué acciones tiene que realizar mientras esté en funcionamiento. Estos comandos utilizados en Arduino son sentencias muy fáciles e intuitivas.

El bloque de código debe tener 2 partes, o funciones principales, que siempre debemos incluir.

ACCEDE AHORA A LOS MEJORES CURSOS

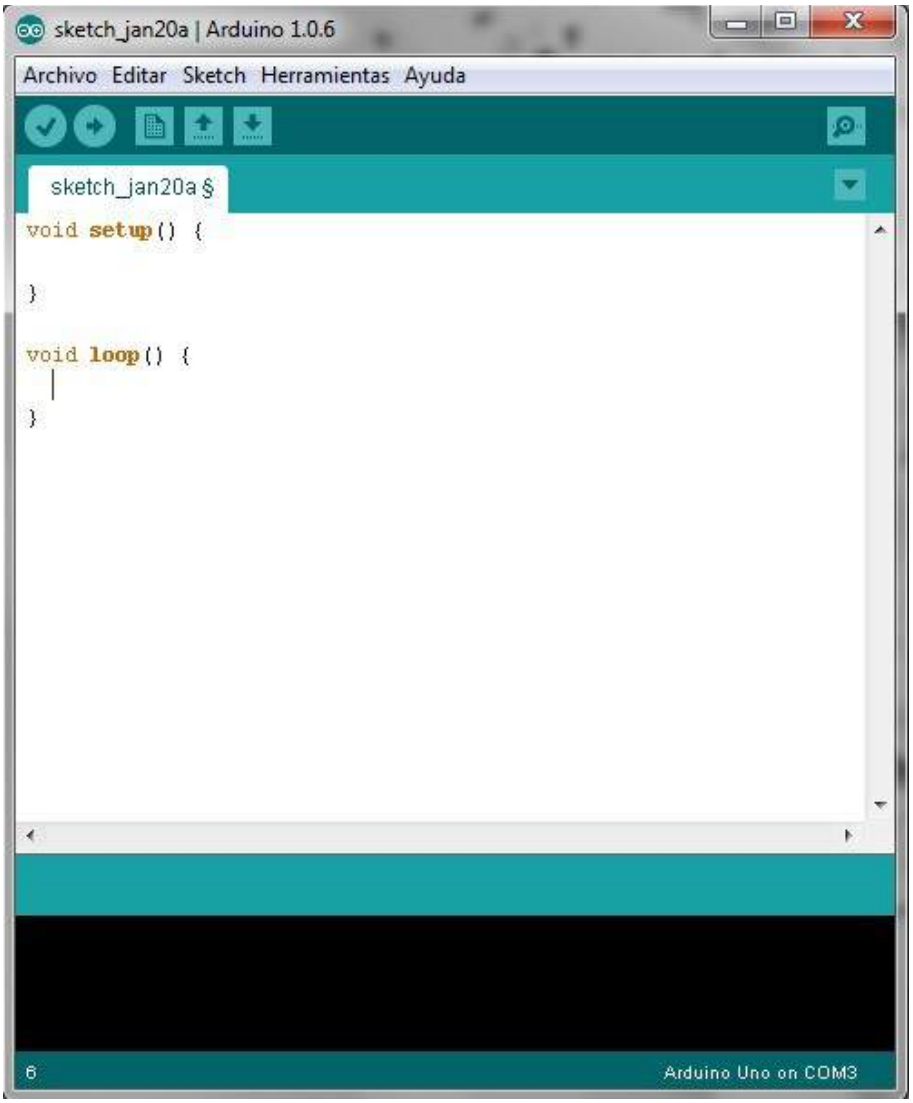


Ilustración 1 Partes básicas de un código

Primero void setup() . Dentro de esta función principal escribiremos las sentencias de configuración que necesitaremos para que Arduino trabaje correctamente. Éstas se ejecutarán al iniciar Arduino y una única vez. Con ellas, por ejemplo, asignaremos la característica de entrada/salida a los pines, el modo de comunicación serial, activaremos los sensores que vayamos a necesitar, escribiremos órdenes de inicio del programa…etc. Algunas de estas sentencias pueden ser: pinMode(,), Serial.begin(), sensors.begin()…

ACCEDE AHORA A LOS MEJORES CURSOS

La segunda función principal es void loop() . Ésta debe ir siempre después de void setup(). En ella escribiremos todas las sentencias, bucles y llamadas a funciones que necesitemos que nuestro Arduino repita constantemente. Se escribirán en orden de ejecución. Ejemplo de éstas pueden ser digitalWrite(,), Serial.print(“ “), if( )…

Existen otras partes del código que no son obligatorias pero que podemos necesitar, según el código lo requiera, para organizarlo todo.

Estas zonas pueden ser un espacio para el título, el autor, librerías, definiciones (variables que vayamos a usar en el programa), una zona para funciones a las que el programa irá llamando…Cada uno es libre de organizarlo como quiera, pero cuanto más ordenado esté todo, más difícil será perdernos cuando los códigos se compliquen.

Abajo podemos ver un ejemplo simple de cómo debe estructurarse un código.

ACCEDE AHORA A LOS MEJORES CURSOS

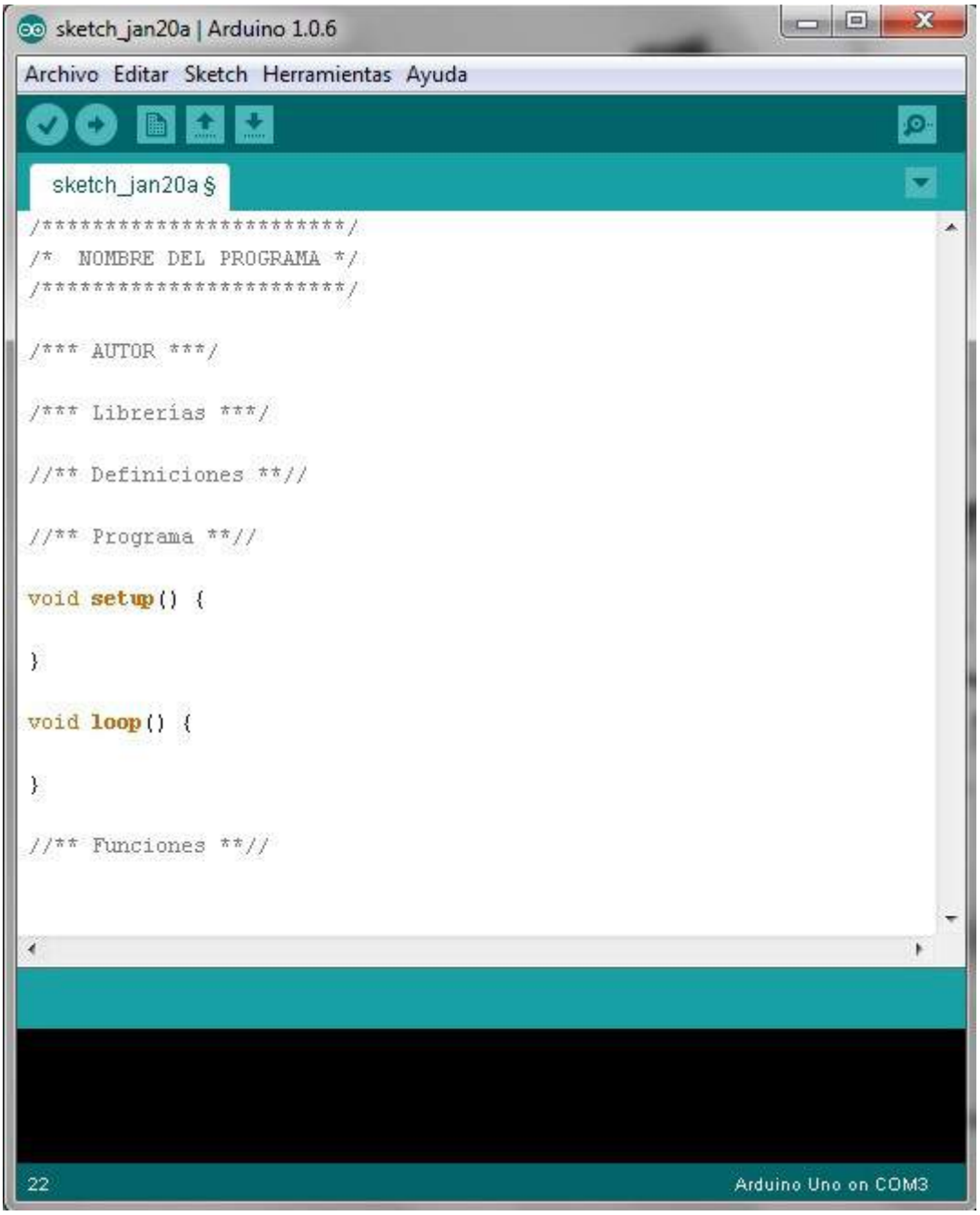


Ilustración 2 Como debe estructurarse un código

ACCEDE AHORA A LOS MEJORES CURSOS

Sentencias básicas, funciones y librerías

Para poder realizar un código Arduino debemos tener las herramientas con las que decirle a nuestro microcontrolador que es lo que debe hacer en cada momento. Estas herramientas son las sentencias y funciones.

Arduino usa una serie de sentencias y comandos básicos muy sencillitos pero a la vez muy potentes. Combinándolos crearemos programas para decirle a nuestra placa que haga prácticamente cualquier cosa que queramos.

Las sentencias se dividen por su manera de funcionar dentro del programa, pudiendo dividirlas en 3 grades grupos:

- Estructurales: Nos dan la forma computacional del código y las operaciones lógicas a ejecutar. Con éstos son con los que le marcamos al programa qué camino debe seguir el código al ejecutarse y qué operaciones debe hacer. Algún ejemplo de las más usadas son if, for, while, +, -, =, *, /, ==, >, < …
- Variables: Con éstas definiremos qué tipo y qué características tendrán los valores que vamos a usar en el código. Pueden ser números, caracteres o estados. Podemos encontrarnos aquí las sentencias int, float, HIGH, LOW, char, string, true, false…
- Funciones: Nos permiten realizar una serie de operaciones concretas y volver a la zona del código en la que se ejecutó. Existen funciones que están propiamente definidas ya por Arduino como tales y también podemos crearnos nuestras propias funciones para que sean llamadas en las zonas de setup o loop cuando lo necesitemos. Son muy útiles cuando necesitamos realizar un grupo de acciones varias veces en distintas zonas del código. También es una buena forma de ahorrar líneas y de organizarse. Ejemplo de funciones tenemos pinMode, digitalWrite, delay, max, sin, cos, analogWrite…

ACCEDE AHORA A LOS MEJORES CURSOS

Cómo explicar qué acción concreta realiza cada sentencia o función sería muy extenso, os dejamos este enlace para que podáis consultarlos. Nosotros iremos comentando todos los que vayamos usando en nuestros ejemplos.

Otra cosa importante son las librerías. Son una serie de sentencias y funciones específicas de cada elemento o shield que conectemos a Arduino, que no están dentro de las propias de la IDE, y que realizarán acciones específicas de cada uno facilitándonos mucho la programación.

Por último nombraremos un par de pequeños detalles necesarios para poder programar nuestro código. Para que cada sentencia o función se ejecute debe llevar al final de cada orden el signo de puntuación punto y coma " ; " y al utilizar la doble barra " // " le decimos al código que lo que se escriba a la derecha es un comentario y que no debe ejecutarlo (Arduino lo coloreará en gris). Esto veréis que es muy útil para explicar que estamos haciendo en cada momento.

Compilación y carga.

Una vez que hayamos escrito nuestro código en la IDE Arduino procederemos a verificar que está correctamente escrito. Para ello pulsamos el botón  arriba a la izquierda. Arduino leerá el texto y nos dirá si es correcto o no. Para ello lo que hará será compilarlo, que significa traducir el código del lenguaje de programación a un lenguaje que entienda el microcontrolador directamente.

En el caso de contener un error nos aparecerá en la consola una explicación orientativa del error.

ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 3 Verificando el código

ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 4 Código correcto

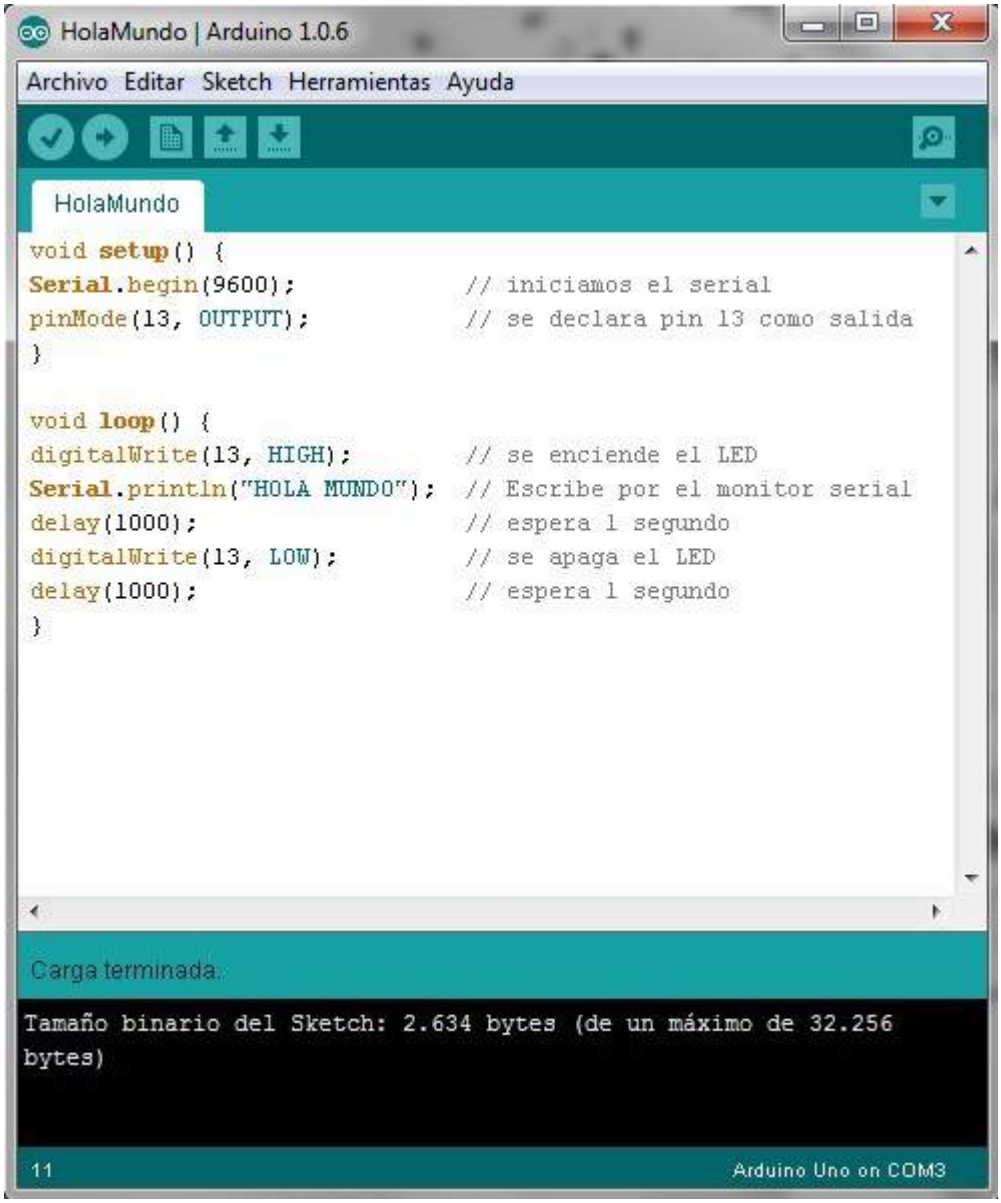
ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 5 Error en el código. Falta un ";"

ACCEDE AHORA A LOS MEJORES CURSOS

Si la compilación ha sido correcta, ya podremos cargar nuestro programa al Arduino.

Para ello, con la placa conectada por USB, pulsamos el botón de carga y esperamos a que termine.



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 6 Carga de programa terminada

ACCEDE AHORA A LOS MEJORES CURSOS

Ejemplo: Semáforo

Como primer proyecto, realizaremos un semáforo con unos Leds. Es ideal para iniciarnos en la programación y el montaje de circuitos porque, como veréis, es muy fácil.

Empezaremos nuestro primer proyecto describiendo los elementos que vamos a necesitar:

- 1 x Arduino UNO R3
- 1 x Protoboard
- 1 x Led rojo 3mm
- 1 x Led amarillo 3mm
- 1 x Led verde 3mm
- 3 x resistencias de 220Ω.
- Cables para conectar todo.

Una vez que tenemos todo, hacemos nuestro montaje siguiendo el siguiente esquema.

ACCEDE AHORA A LOS MEJORES CURSOS

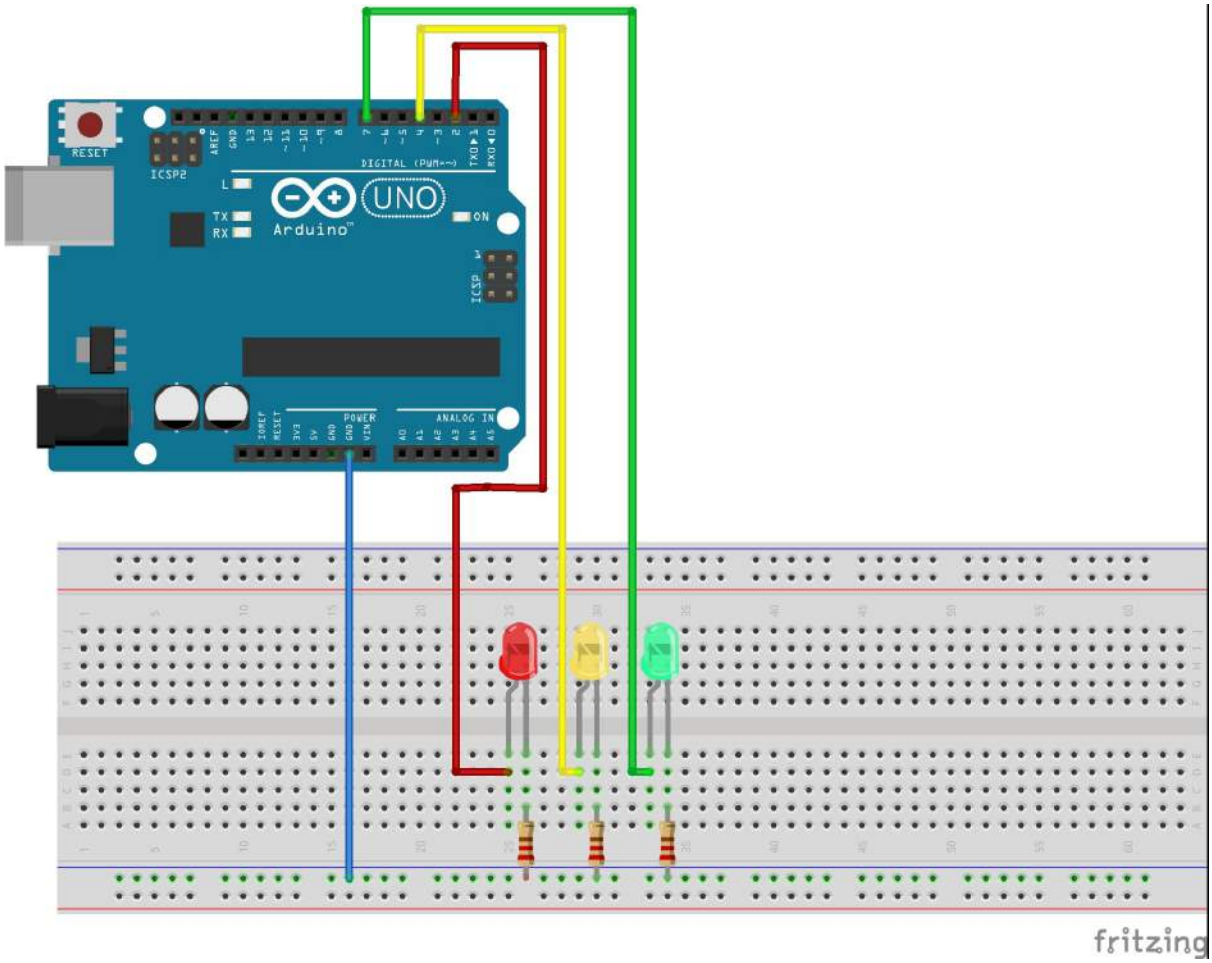


Ilustración 7 Esquema de montaje del semáforo

Utilizaremos los pines digitales 2 (rojo), 4 (amarillo) y 7 (verde). Al conectar los Leds debéis tener en cuenta que tienen polaridad, por lo que tenéis que colocarlos bien para que funcionen. En los Leds la patilla corta, o el lado que está achatado, es el negativo e irá conectado a tierra (GND en la placa) a través de una resistencia. La patilla larga, o lado redondeado, es el positivo y se conectará al pin del Arduino correspondiente.

ACCEDE AHORA A LOS MEJORES CURSOS

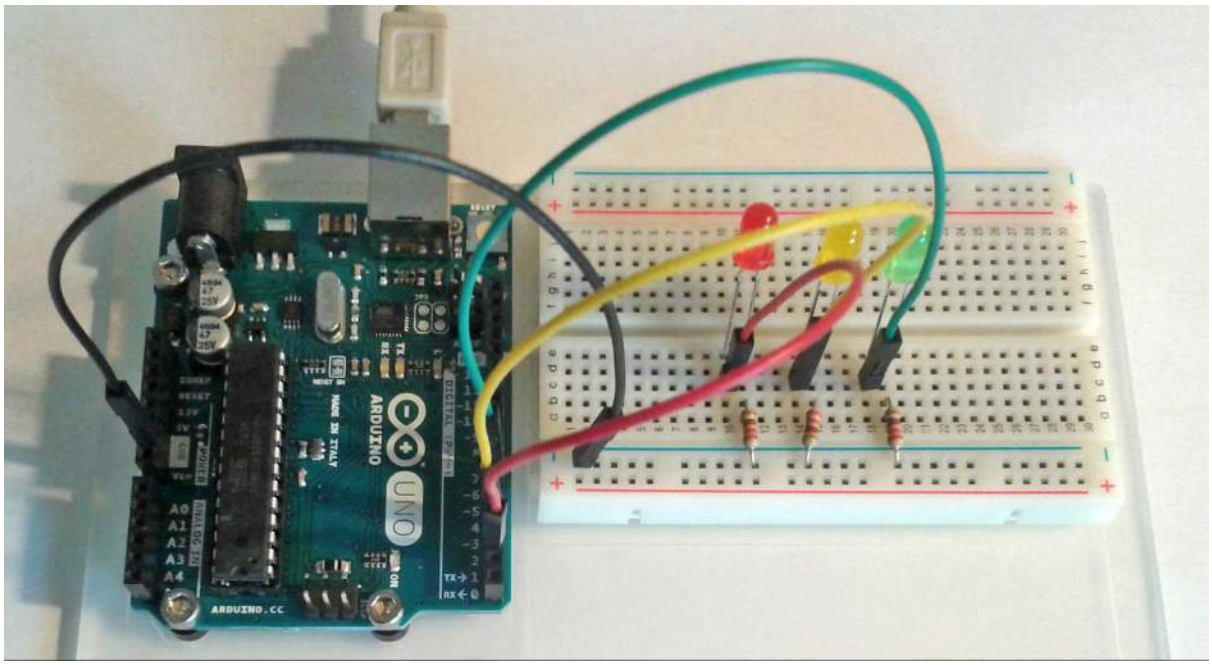


Ilustración 8 Protoboard del proyecto semáforo

Una vez montado, abriremos nuestro IDE Arduino y escribiremos el programa (sketch).

```
/*************/
/*  SEMAFORO */
/*************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//
int rojo=2;      //definimos el valor del pin para el led rojo
int amarillo=4;  //definimos el valor del pin para el led amarillo
int verde=7;     //definimos el valor del pin para el led verde

//** Programa **//

void setup() {
 pinMode(verde,OUTPUT);   //declaramos el pin verde como salida
 pinMode(amarillo,OUTPUT);//declaramos el pin amarillo como salida
 pinMode(rojo,OUTPUT);    //declaramos el pin rojo como salida
}
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
void loop() {
digitalWrite(verde,HIGH); //encendemos el led rojo
delay(2000);             //esperamos 2 segundos
digitalWrite(verde,LOW);  //apagamos el led rojo
delay(500);              //esperamos medio segundo

digitalWrite(amarillo,HIGH); //encendemos el led amarillo
delay(2000);                 //esperamos 2 segundos
digitalWrite(amarillo,LOW);  //apagamos el led amarillo
delay(500);                  //esperamos medio segundo

digitalWrite(rojo,HIGH); //encendemos el led verde
delay(2000);              //esperamos 2 segundos
digitalWrite(rojo,LOW);  //apagamos el led verde
delay(500);               //esperamos medio segundo
}
```

Con la sentencia int estamos declarando una variable numérica entera, para poderla usar después en nuestro código.

El comando delay hace que el programa pare un tiempo determinado. Éste lo definiremos, en milisegundos, dentro de los paréntesis.

Las funciones pinMode y digitalWrite se explicarán en el siguiente post, salidas, con detalle.

Salidas

Pines de salida

Arduino utiliza sus pines de salida para enviar señales eléctricas. Éstas señales pueden utilizarse para alimentar otros dispositivos (led's, zumbadores...) o para comunicarse enviando una serie de pulsos que el receptor debe entender.

Las salidas se dividen en dos niveles o estados: HIGH a 5 V y LOW a 0 V. Por defecto, y sin ningún programa cargado en nuestra tarjeta controladora, las salidas

ACCEDE AHORA A LOS MEJORES CURSOS

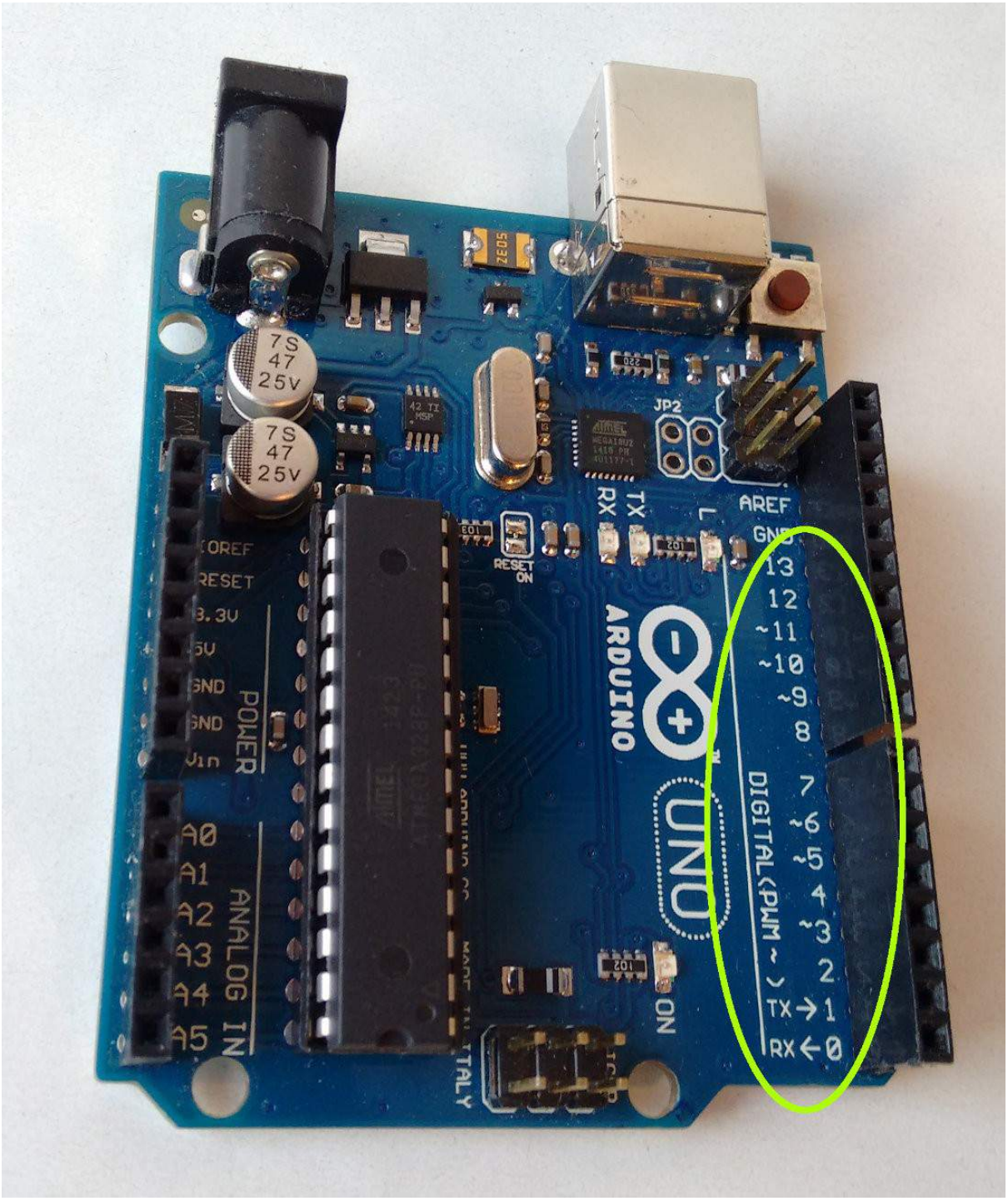
están a nivel LOW. Para valores intermedios veremos los pines con función PWM en el punto 2.

Para valores constantes de 5V, 3.3V y 0V (GND) podemos utilizar los pines correspondientes del apartado “POWER” de nuestro Arduino.

Configuración y uso de las salidas digitales

Los pines que podemos configurar para que cambien el nivel de salida según nuestro programa son los digitales, es decir, los 1 a 13. En realidad existen otros pines digitales pero no serán tratados en este tutorial.

ACCEDE AHORA A LOS MEJORES CURSOS



Por defecto estos pines están configurados como entradas así que lo primero que tenemos que hacer para utilizar un pin como salida es configurarlo como tal. Para ello escribimos esta sentencia dentro del apartado setup del programa.

ACCEDE AHORA A LOS MEJORES CURSOS

```
pinMode(pin,OUTPUT);
```

Donde sustituiremos "pin" por el número del pin que queremos configurar como salida. También podemos almacenar el número de ese pin en una variable, por ejemplo el pin 10 en la variable llamada "pinsalida" y escribirla dentro de la función pinMode.

```
int pinsalida = 10;                 // almacena el número 10 en la variable "pinsalida"

pinMode(pinsalida,OUTPUT);    // configura el pin número "pinsalida" (10) como salida
```

Ahora ya lo tenemos configurado como salida pero aún sigue a 0 V, es decir, a nivel LOW. Cuando en un determinado momento del programa queramos que ese pin se ponga a nivel HIGH escribiremos esta sentencia

```
digitalWrite(pinsalida,HIGH);
```

Ahora tendremos 5 V permanentemente en ese pin a no ser que en el programa insertemos

```
digitalWrite(pinsalida,LOW);
```

en cuyo caso se volverá al valor de 0 V. Podemos poner los pines a HIGH o LOW tantas veces como queramos.

IMPORTANTE: hay que tener en cuenta que estos pines tienen una intensidad limitada a 40 mA, por tanto no será suficiente para alimentar a algunos relés, motores, bombillas y todo aquello que necesite mayor corriente. En otros post veremos cómo solucionamos este "problema".

ACCEDE AHORA A LOS MEJORES CURSOS

PWM

La función PWD nos permite obtener valores intermedios de salidas entre 0 y 5V escalados en en 255 niveles, siendo 0 el más bajo (0 V) y 255 el más alto (5 V).

Lo que en realidad sucede es que el microcontrolador alterna rapidísimamente estados HIGH y LOW un tiempo proporcional al valor requerido. Así, con 127 estaría el 50% del tiempo a HIGH y el otro 50% restante a LOW. Nosotros no lo notamos porque todo ello sucede a una frecuencia superior a la que captan nuestros ojos.

Los pines que permiten esta función están señalados con el símbolo ~ . En la placa Uno son los pines 3, 5, 6, 9, 10 y 11.

Si como vimos en en anterior apartado ya tenemos nuestro pin configurado como salida, para obtener un nivel de salida de 75% escribiremos en nuestro programa

```
analogWrite(pinsalida,191);
```

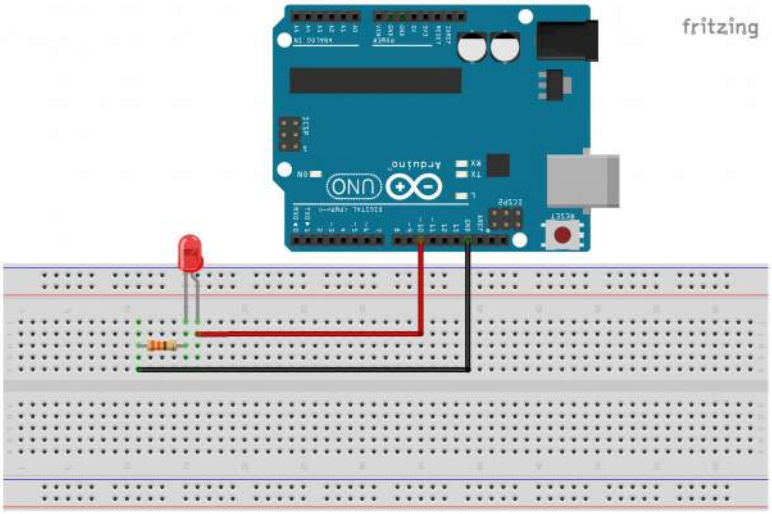
Ya que 191 es el 75% de 255.

Ejemplo función PWM con LED

Este ejemplo consta de dos partes. Primero un led se encenderá durante dos segundos. Luego variará su intensidad entre los valores máximo y mínimo gradualmente. Para conseguirlo utilizaremos uno de los pines PWM.

El montaje es de lo más sencillo. Insertamos el led en la protoboard. Para no dañar el led ponemos en serie una resistencia de unos 220-470 Ohmios. Con valores menores se corre el riesgo de dañarlo y valores mayores atenúan demasiado el brillo.

ACCEDE AHORA A LOS MEJORES CURSOS



Seguidamente unimos mediante cables el pin 10 con la fila del ánodo (pata larga) y el GND con la fila del extremo libre de la resistencia. Si el led tiene las patas iguales, una forma de distinguir el cátodo del ánodo es fijándose en el borde del led que no es completamente circular. Pues la parte biselada corresponde al cátodo.
Ahora vamos con el programa. En la primera parte declaramos las variables.

```
int pinsalida = 10;          // variable para guardar el número del pin
int velocidad = 100;         // variable para cambiar la velocidad de encendido/apagado
int pwm;                     // variable que almacenará el valor entre 0 y 255
```

Cambiando el valor de la variable velocidad haremos que el led tarde más o menos tiempo en completar los ciclos de cambio de intensidad.
En la función setup hacemos que el led se encienda 2 segundos y luego se apague.

```
void setup()
{
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
        pinMode(pinsalida,OUTPUT);       // configura "pinsalida" como salida
        digitalWrite(pinsalida,HIGH);     // pone el "pinsalida" a nivel alto
        delay(2000);                      // espera 2000 milisegundos (2 segundos)
        pinMode(pinsalida,LOW);           // pon "pinsalida a nivel bajo
        delay(1000);                       // espera 1 segundo
}
```

En la función loop hacemos uso del PWM. Escribimos un bucle for que dará a la variable pwm un valor inicial de 0 y lo irá incrementando en una unidad hasta que alcance el valor máximo de 255. A continuación ponemos otro bucle for que irá decrementando el valor desde el máximo hasta el mínimo.

```
void loop()
{
   for(pwm=0;pwm<256;pwm++)          // desde pwm valiendo 0; hasta que valga 255; incrementa pwm
   {
      analogWrite(pinsalida,pwm);          // pone el "pinsalida" a el valor que almacena la variable "pwm"
      delay(1000/velocidad);               // espera (1000/velocidad) milisegundos
   }
   for(pwm=255;pwm>-1;pwm--)          // este bucle hace lo mismo pero decrementando pwm
   {
      analogWrite(pinsalida,pwm);
      delay(1000/velocidad);
   }
}
```

ACCEDE AHORA A LOS MEJORES CURSOS

El programa entero quedaría así.

```
/**********************/
/*          ejemplo led pwm     */
/**********************/

int pinsalida = 10;          // variable para guardar el número del pin
int velocidad = 100;         // variable para cambiar la velocidad de encendido/apagado
int pwm;                     // variable que almacenará el valor entre 0 y 255

void setup()
{
          pinMode(pinsalida,OUTPUT);     // configura "pinsalida" como salida
          digitalWrite(pinsalida,HIGH);       // pone el "pinsalida" a nivel alto
          delay(2000);                  // espera 2000 milisegundos (2 segundos)
          pinMode(pinsalida,LOW);          // pon "pinsalida a nivel bajo
          delay(1000);                   // espera 1 segundo
}

void loop()
{
   for(pwm=0;pwm<256;pwm++)           // desde pwm valiendo 0; hasta que valga 255; incrementa
pwm
   {
      analogWrite(pinsalida,pwm);           // pone el "pinsalida" a el valor que almacena la variable
"pwm"
      delay(1000/velocidad);                // espera (1000/velocidad) milisegundos
   }
   for(pwm=255;pwm>-1;pwm--)           // este bucle hace lo mismo pero decrementando pwm
   {
      analogWrite(pinsalida,pwm);
      delay(1000/velocidad);
   }
}
```

ACCEDE AHORA A LOS MEJORES CURSOS

Entradas Analógicas y Digitales

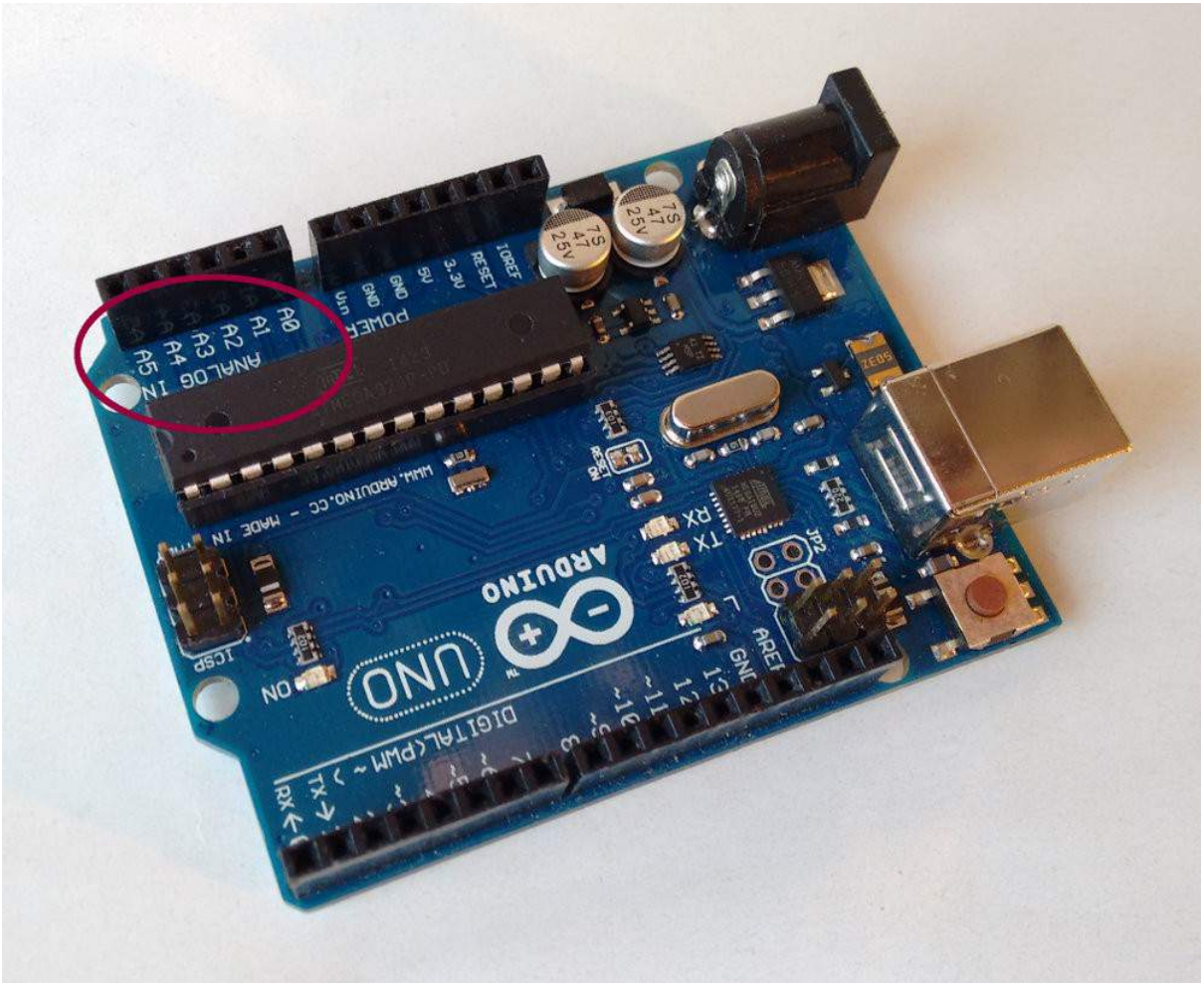
Descripción de las entradas

Nuestro Arduino no sólo puede enviar señales sino que también puede recibirlas con dos propósitos principales como son leer datos de sensores y recibir mensajes de otros dispositivos (shield, otro Arduino, PC, etc.). Las entradas las clasificaremos en analógicas y digitales.

Entradas analógicas

Las entradas analógicas del modelo Uno son las correspondientes a los pines de A0 a A5. Se caracterizan por leer valores de tensión de 0 a 5 Voltios con una resolución de 1024 (10 bits). Si dividimos 5 entre 1024 tenemos que es capaz de detectar variaciones en el nivel de la señal de entrada de casi 5 mV.

ACCEDE AHORA A LOS MEJORES CURSOS



Para hacer la lectura de uno de estos pines escribiremos en nuestro código

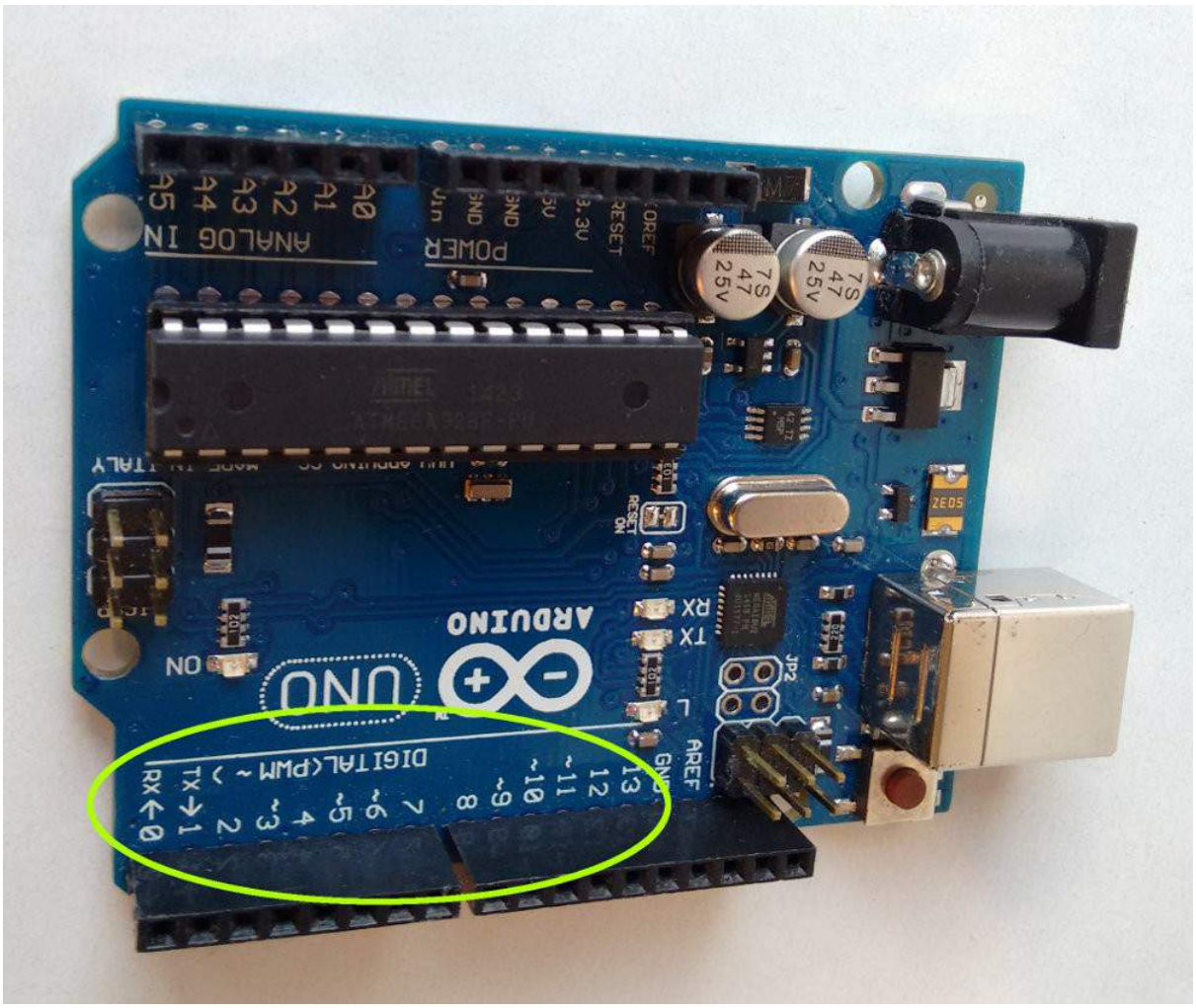
```
lectura = analogRead(pinentrada);
```

“lectura” lo sustituimos por el nombre de la variable donde queramos almacenar el valor leído y en “pinentrada” tendremos que poner el número del pin analógico que hemos elegido (0,1,...5) o el nombre de la variable que almacena dicho número.
Esta función nos devolverá un valor que va de 0 a 1023 en proporción al nivel de la señal de entrada. Para una entrada nula obtendremos el valor 0, para una entrada de 2.5 Voltios 511 (la mitad de 1023) y para 5 Voltios 1023.

ACCEDE AHORA A LOS MEJORES CURSOS

Entradas digitales

Las entradas digitales son las mismas que las salidas digitales, es decir, los pines que van del 1 al 13. Se diferencian de las analógicas porque éstas son capaces de “entender” sólo dos niveles de señal, LOW o valores cercanos a 0 V y HIGH o valores cercanos a 5 V. Puede parecer una desventaja pero en realidad puede ser todo lo contrario. Y no sólo porque a veces únicamente necesitamos saber dos estados (interruptor, pulsador, sensor de presencia, final de carrera....) sino porque así es capaz de leer señales de pulsos digitales. Esto significa que puede comunicarse .



ACCEDE AHORA A LOS MEJORES CURSOS

Por poner un ejemplo, un sensor analógico de temperatura como es el LM35 incrementaría el nivel de la tensión que llega a la placa de forma proporcional a la temperatura. Sin embargo, uno digital como el ds18b20 lo que haría es cambiar la sucesión de pulsos y por tanto el mensaje que contiene el valor de la temperatura.

Aunque los pines digitales por defecto vienen configurados como entrada, si queremos hacerlo manualmente escribimos en nuestro código

```
pinMode(pinentrada,INPUT);
```

Para almacenar los dos valores posibles LOW o HIGH en una variable llamada "lectura" escribimos

```
lectura = digitalRead(pinentrada);
```

Medición de temperatura. Sensor M35

Este sensor de temperatura lo podéis encontrar por poco más de 1€ . Es un sensor lineal, tiene una precisión de 0.5 ºC y una sensibilidad de 10 mV/ºC. Podéis encontrar más información en su datasheet .

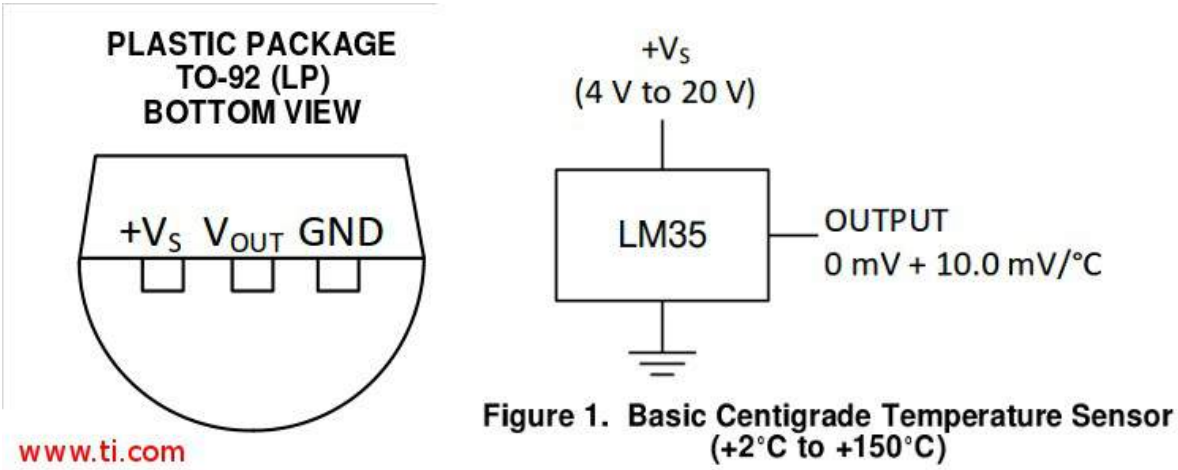
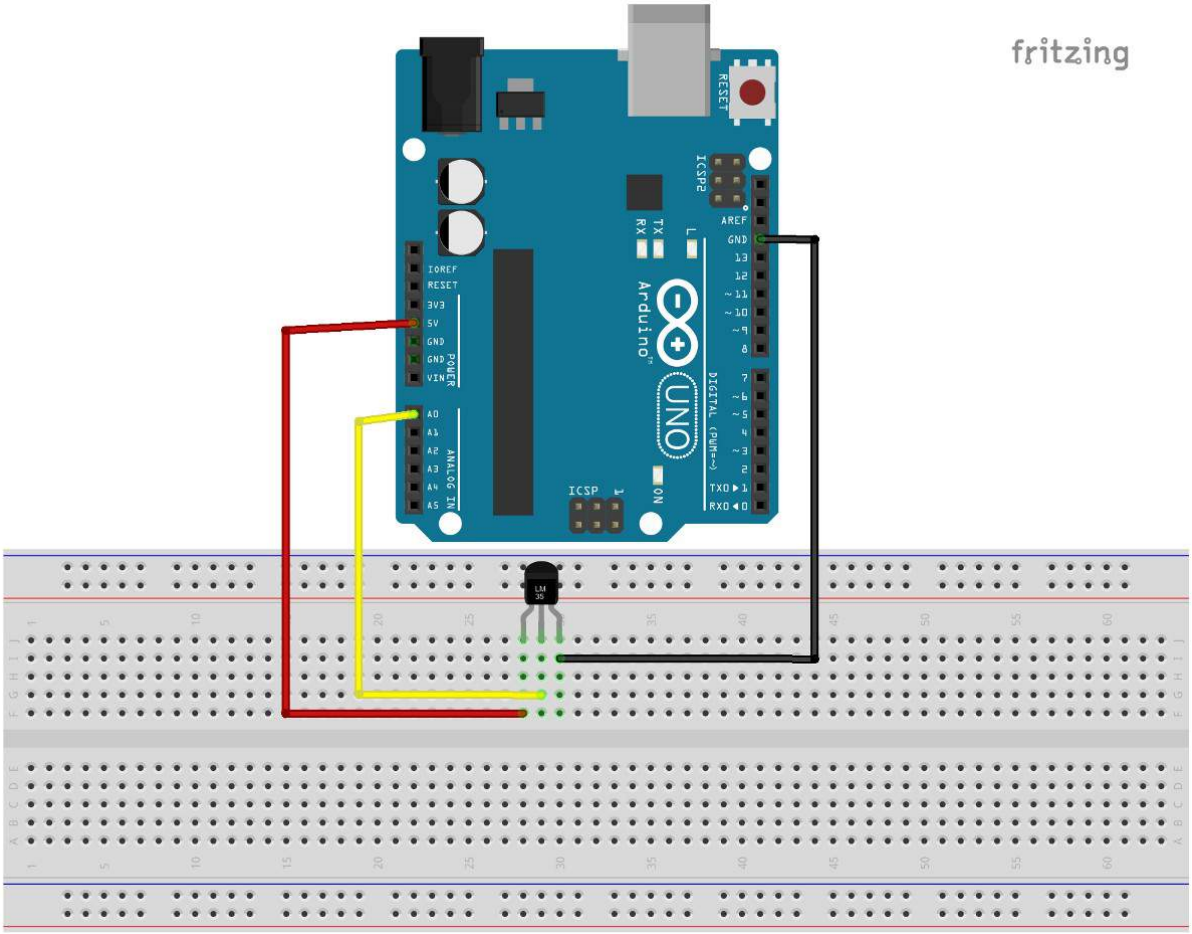


Figure 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

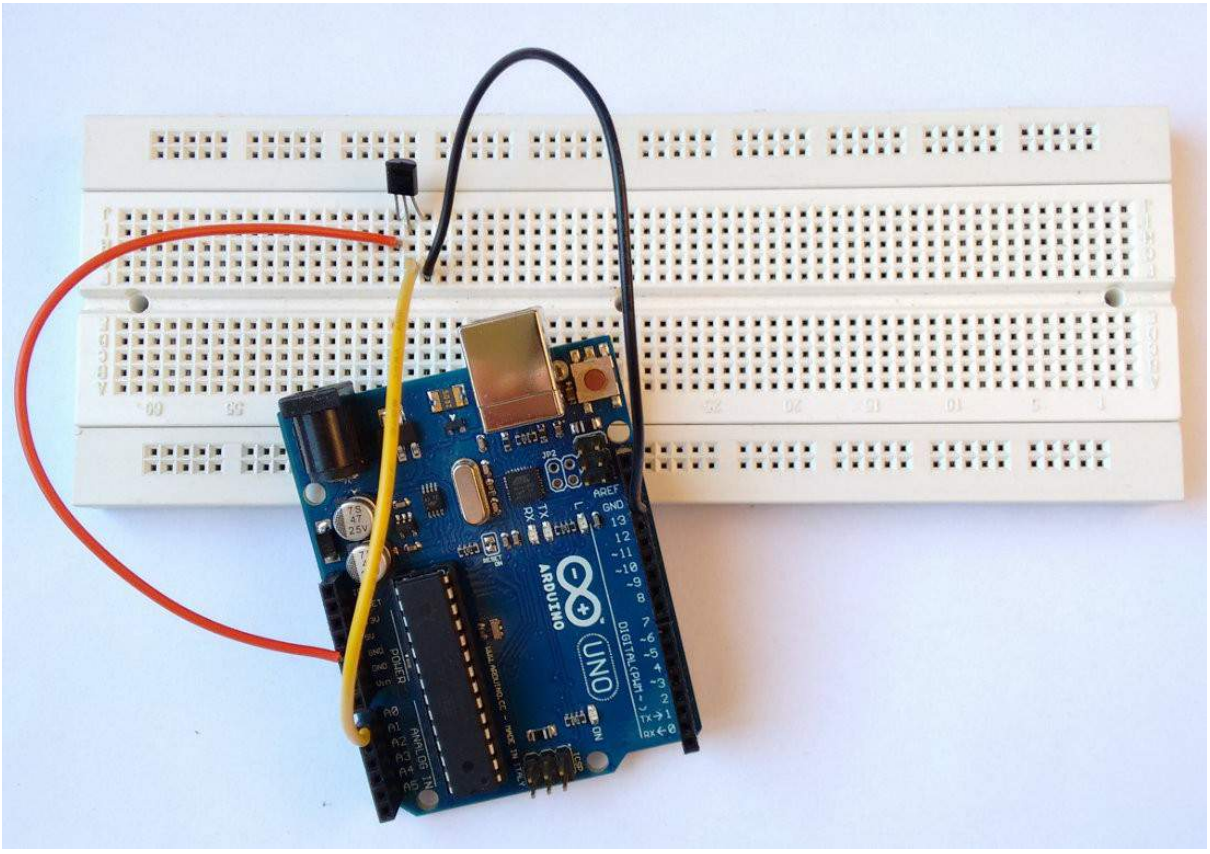
ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

Vamos a hacer un pequeño proyecto para medir la temperatura ambiente y visualizarla en nuestro PC. Su montaje no lleva más de un par de minutos y podemos verlo claramente en el esquema. La pata +Vs va al pin 5V, la pata Vout al pin que utilicemos para la lectura, en este caso el 0, y la restante GND al pin GND. Tened en cuenta que el dibujo del LM35 está hecho con vista desde las patas . CUIDADO con colocarlo al revés porque si lo hacéis mal el sensor se calentará rápidamente y a parte de estropearlo os quemaréis al tocarlo.



ACCEDE AHORA A LOS MEJORES CURSOS



El programa para este ejemplo es también muy simple y corto. Como siempre empezamos con la declaración de las variables. Esta vez, para almacenar la temperatura no utilizaremos una variable del tipo int sino float para poder almacenar decimales.

```
float temperatura;          // aquí almacenaremos el valor leído
int pinentrada = 0;         // utilizaremos el pin A
```

ACCEDE AHORA A LOS MEJORES CURSOS

Lo siguiente es abrir el puerto serial en la función setup . Ésto se hace para poder comunicarnos desde nuestro ordenador con el controlador y así poder mostrar los datos por pantalla.

```
void setup()
{
        Serial.begin(9600);        // abrimos el puerto reial a 9600 bps
}
```

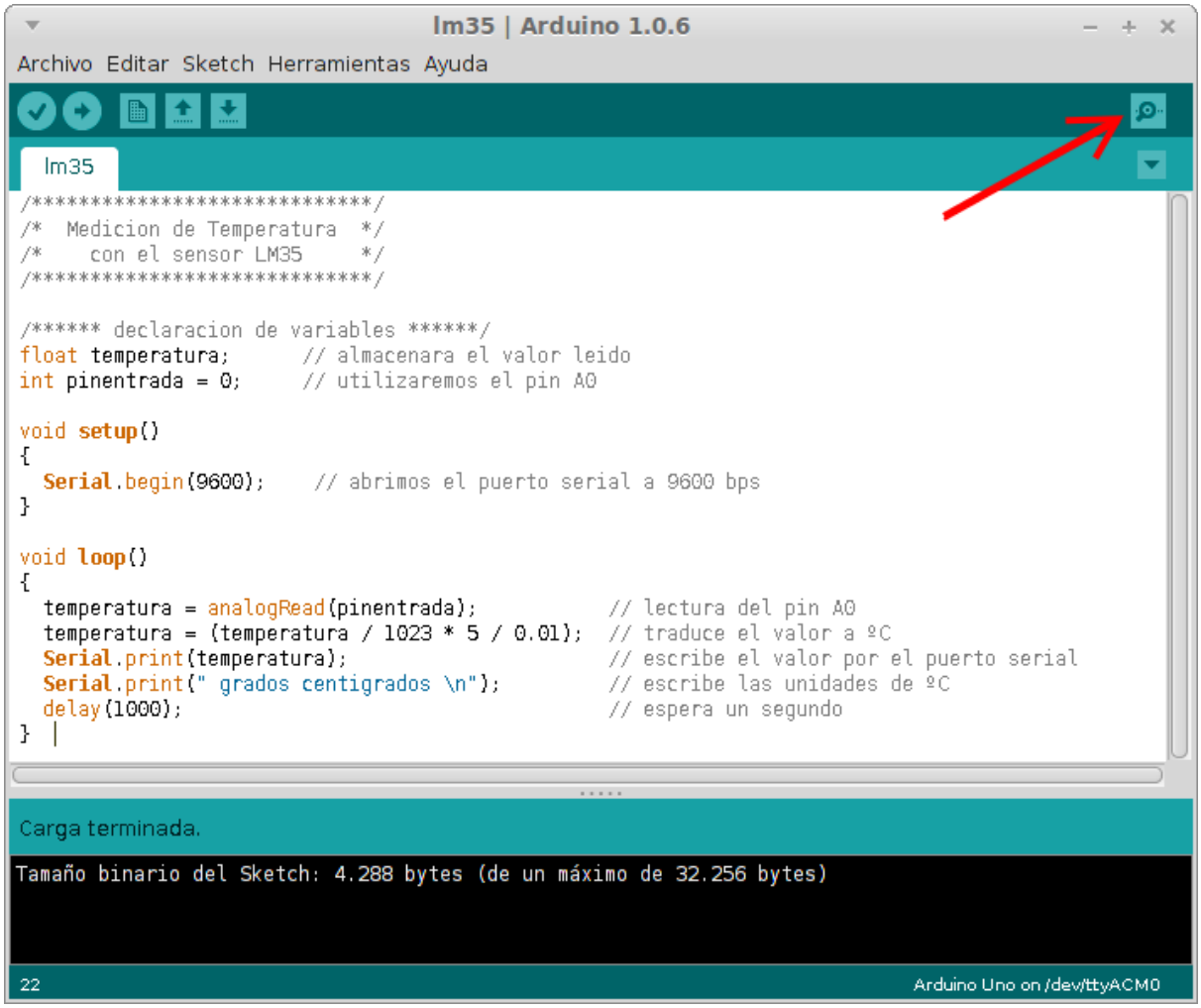
$$T(^{o}C) = \frac{Valor\ leído}{1023} \frac{5\ Voltios}{0{,}01\ Voltios/^{o}C}$$

En el código quedaría así

```
void loop()
{
   temperatura = analogRead(pinentrada);                    // lectura del pin A0
   temperatura = (temperatura / 1023 * 5 / 0.01);          // "traduce" el valor leído a grados
   Serial.print(temperatura);                              // escribe el valor de la temperatura por el
puerto serial
   Serial.print(" grados centigrados \n");                  // escribe las unidades
   delay(1000);                                            // espera 1 segundo
}
```

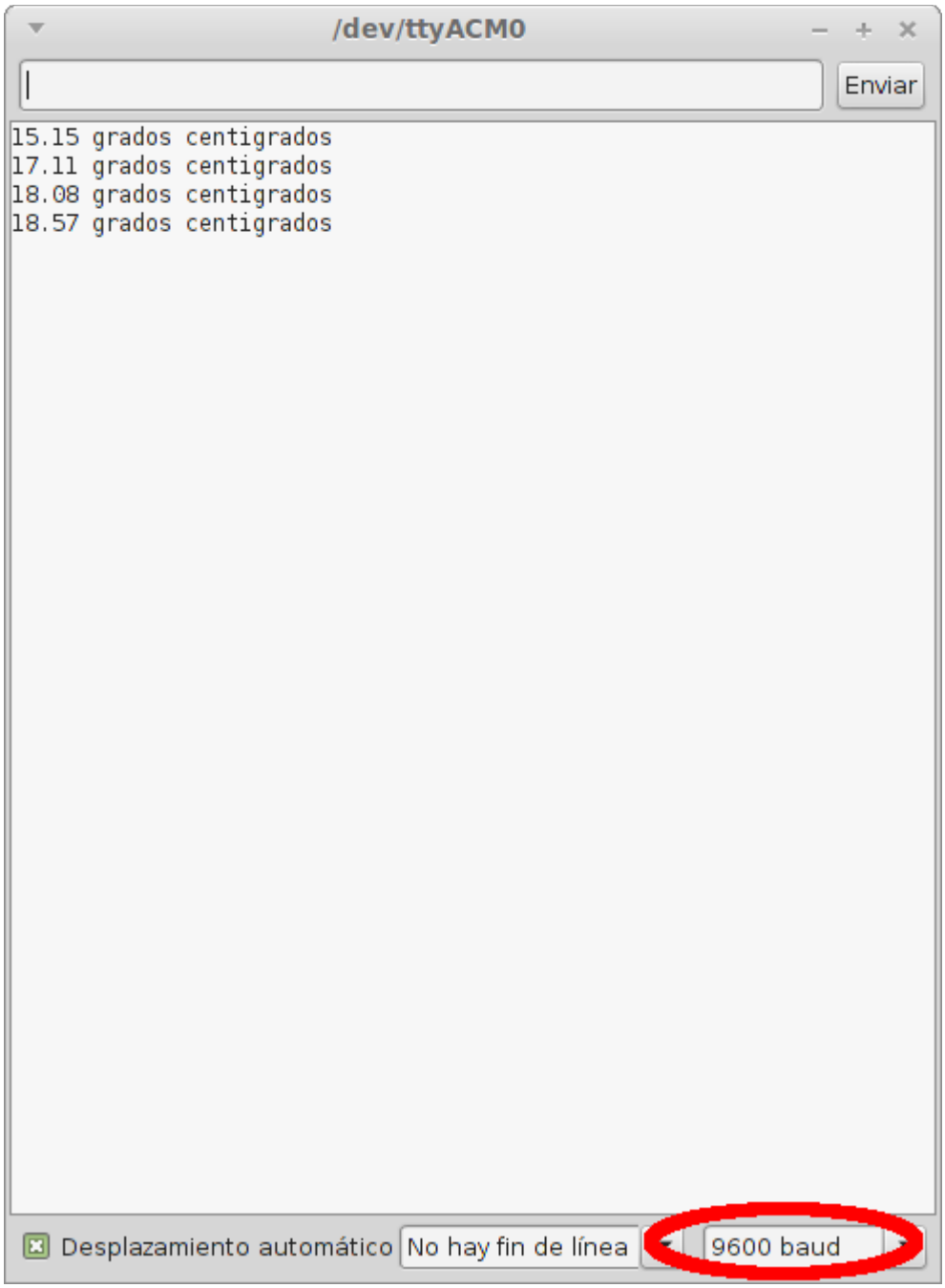
ACCEDE AHORA A LOS MEJORES CURSOS

Cargamos el código en la placa y sólo tenemos que pinchar en el icono del monitor serial



Si nos muestras caracteres extraños es que la configuración del monitor serial está comunicándose con una tasa bps distinta a los 9600 con los que hemos configurado nuestro puerto serial. Para corregirlo sólo tenemos que seleccionar el valor adecuado de la pestaña desplegable como se puede ver en la siguiente imagen.

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

Os dejo el programa entero para que sólo tengáis que copiar y pegar a vuestro IDE.

```
/*****************************/
/*  Medicion de Temperatura  */
/*    con el sensor LM35     */
/*****************************/

/****** declaracion de variables ******/
float temperatura;        // almacenara el valor leido
int pinentrada = 0;       // utilizaremos el pin A0

void setup()
{
   Serial.begin(9600); // abrimos el puerto serial a 9600 bps
}

void loop()
{
   temperatura = analogRead(pinentrada);              // lectura del pin A0
   temperatura = (temperatura / 1023 * 5 / 0.01); // traduce el valor a ºC
   Serial.print(temperatura);                          // escribe el valor por el puerto serial
   Serial.print(" grados centigrados \n");          // escribe las unidades de ºC
   delay(1000);                                        // espera un segundo
}
```

Entradas (2): Botones

En este post, haremos una extensión del anterior. Nos detendremos en uno de los elementos más básicos en el control de Arduino, pero de los más usados y prácticos, los botones. Éstos son un tipo de elemento externo que nos permite controlar físicamente acciones a realizar por nuestro sistema a través de dar continuidad o no a la señal de entrada en la que están instalados. Como podréis

ACCEDE AHORA A LOS MEJORES CURSOS

imaginar son parte fundamental de casi cualquier proyecto y son muy fáciles de usar y programar, aunque hay que tener en cuenta un par de detalles a la hora de usarlos. Para comprender bien cómo funcionan en los distintos casos y como solucionar algunos problemillas propios de los botones, realizaremos 3 ejemplos para verlos detalladamente y en los que usaremos el mismo circuito/esquema y solo iremos variando su programación.

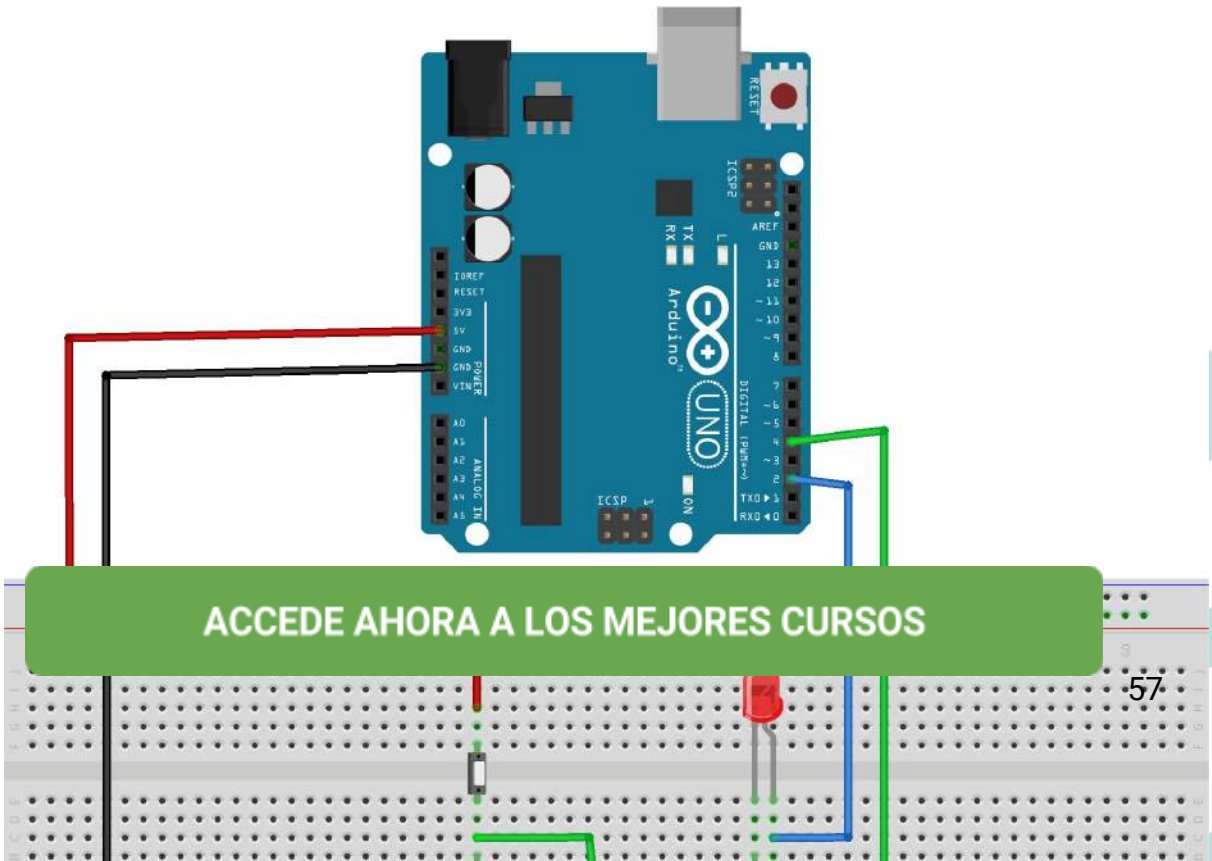
Esquema común

Para realizar todos los ejemplos necesitaremos los siguientes elementos:

- 1 x Arduino UNO R3
- 1 X Protoboard
- 1 x led (usaremos rojo pero vale cualquier color)
- 1 x Botón
- 1x Resistencia de 220Ω (puede valer de 330Ω)
- 1 x Resistencia de 10kΩ (puede valer de 1 ó 5 kΩ)
- Cables para conectar todo

Con todo comenzaremos a montar nuestro circuito como se describe en el siguiente esquema.

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

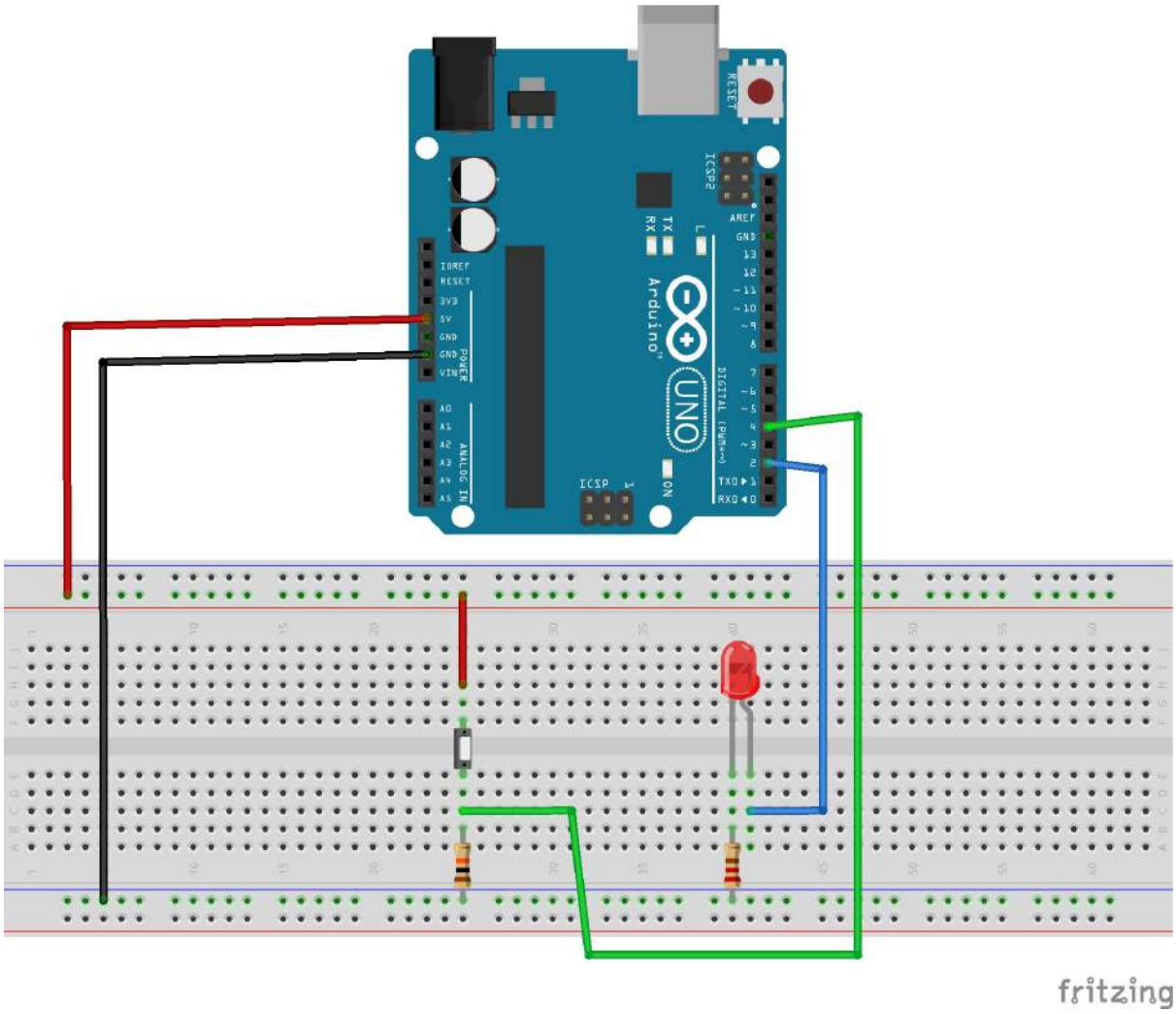
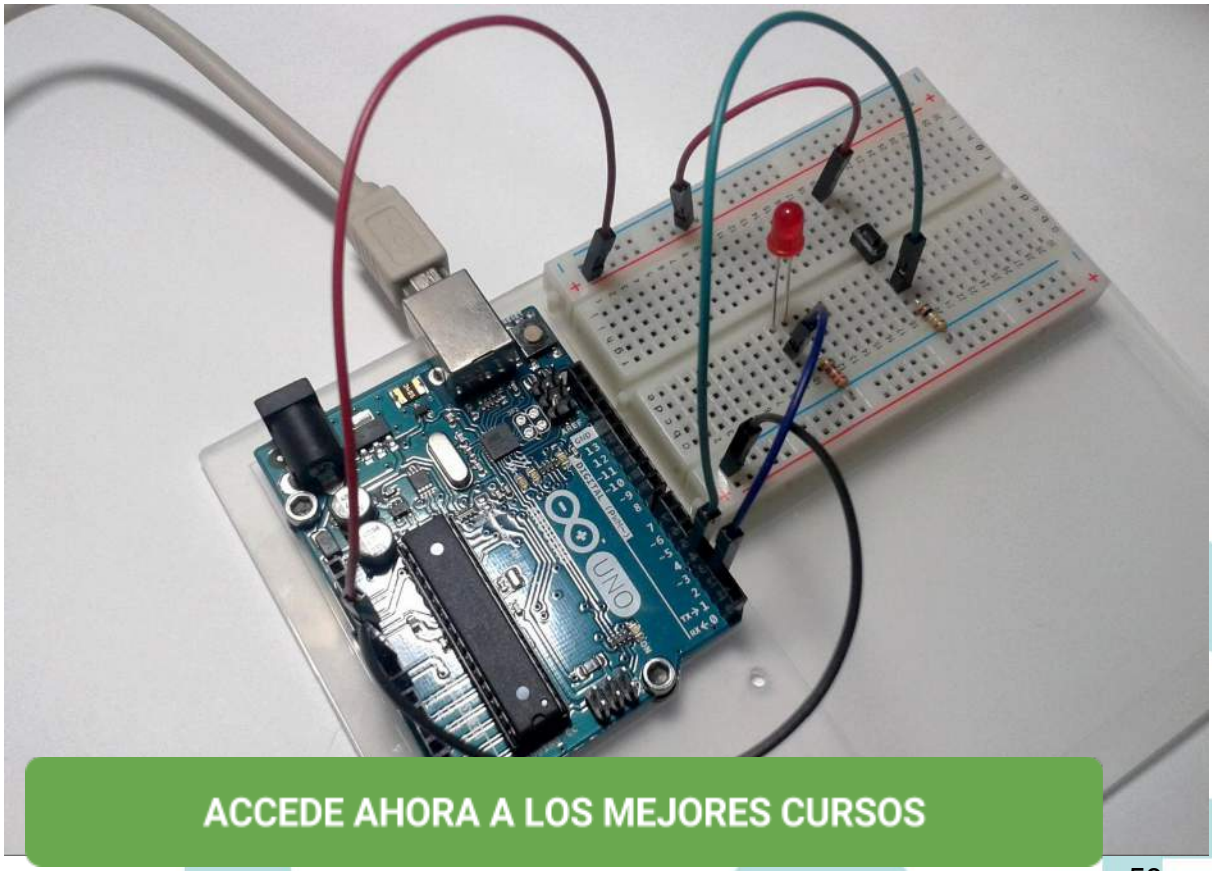


Ilustración 1 Esquema de montaje del interruptor.

Usaremos el pin 2 para encender el led y el 4 para saber el estado del botón. Al montarlo debemos tener en cuenta un par de detalles. Primero, que conectemos correctamente la polaridad del led, siguiendo las indicaciones que os dimos en el ejemplo del semáforo. Segundo, que usemos laresistencia de 10kΩ para conectar el botón a tierra. Esto es muy importante, ya que con eso protegemos a nuestra placa de crear un cortocircuito a tierra que podría deteriorarla. La resistencia de 220Ω, como en ejemplos anteriores, la usaremos para conectar el led a tierra.

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

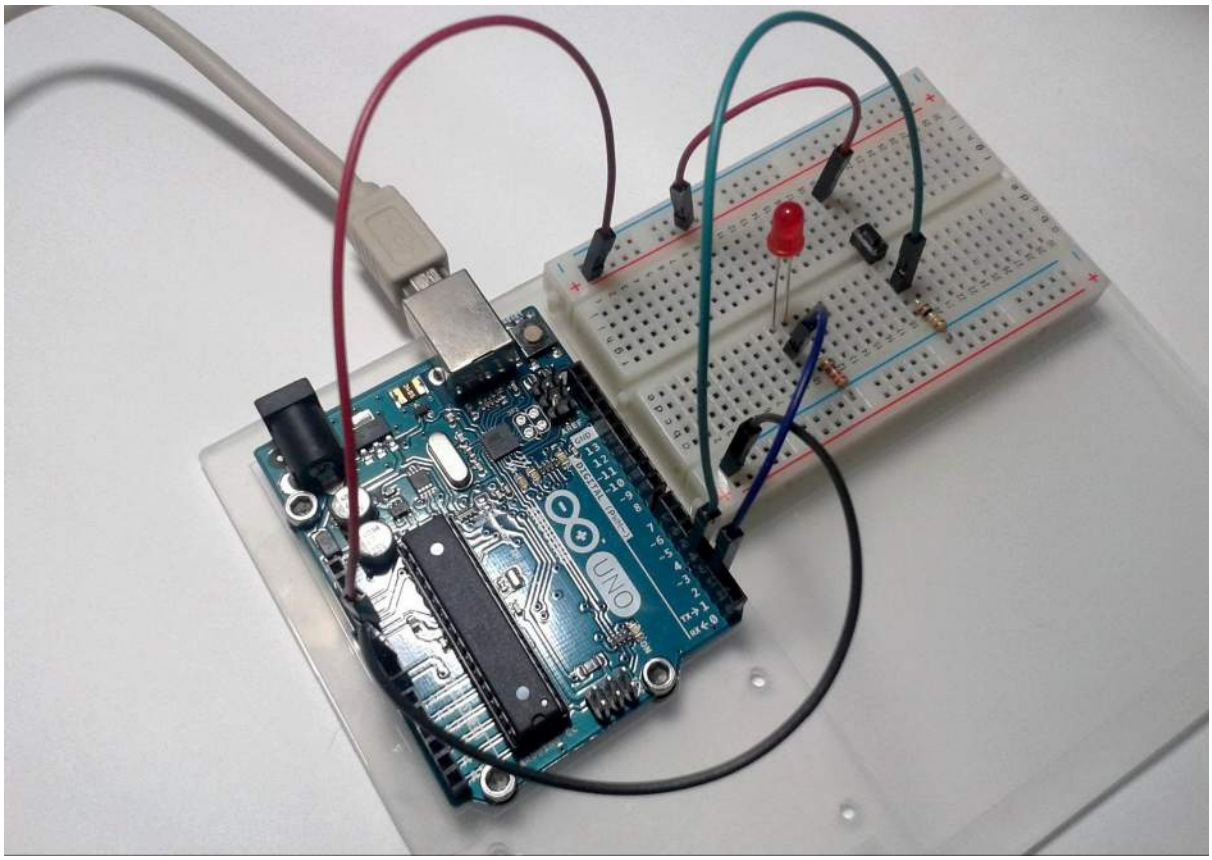


Ilustración 2 Montaje real para ejemplos de botones e interruptores

Pulsador

Empezaremos con el botón más sencillo de programar y usar que tenemos, el pulsador. Éste nos permite realizar una acción mientras mantengamos pulsado nuestro botón, por ejemplo que un led se quede encendido mientras estemos pulsando.

Para nuestro programa declararemos una variable llamada “pulsador” que usaremos para saber si nuestro botón está pulsado o no.

```
int pulsador=0;
```

ACCEDE AHORA A LOS MEJORES CURSOS

Dentro del setup, configuraremos el pin digital 2 como salida para poder dar la orden de encender o apagar.

```
pinMode(2, OUTPUT);
```

Para que podamos saber en qué estado se encuentra nuestro botón configuraremos el pin digital 4 como entrada.

```
pinMode(4, INPUT);
```

Finalmente nuestro código loop lo que hará será, primero leer del pin 4 en qué estado está el botón mediante la sentencia digitalRead. Este valor lo almacenaremos en la variable "pulsador" que declaramos anteriormente.

```
pulsador = digitalRead(4);
```

Una vez que sepa cómo se encuentra el botón, mediante una función "if", si el pulsador está HIGH (pulsado) encenderá el led y si el pulsador está en LOW (sin pulsar) lo apagará.

```
if(pulsador==HIGH) {
      digitalWrite(2, HIGH);
  }
  else{
      digitalWrite(2, LOW);
```

El código completo quedaría así:

```
/************************ */
/* Encender LED con Botón */
/*          Pulsador      */
/************************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
int pulsador=0;              //almacena el estado del botón

//** Programa **//
void setup() {
 pinMode(2, OUTPUT);       //declaramos el pin 2 como salida
 pinMode(4, INPUT);        //declaramos el pin 4 como entrada
}

void loop() {
 pulsador = digitalRead(4);   //lee el estado del botón
 if(pulsador==HIGH) {         //si el estado es pulsado
   digitalWrite(2, HIGH);    //se enciende el led
 }
 else{                                 //si el estado es no pulsado
   digitalWrite(2, LOW);             //se apaga el led
 }
}
```

Interruptor

Con este tipo, podremos usar un botón de la misma manera que si usáramos un interruptor de los que conocemos comúnmente. Cuando pulsemos, se realizará la acción programada (encender el led) y se mantendrá en este estado hasta que volvamos a pulsar nuevamente (se apagará el led).

Para obtener este tipo de botón solo haremos una pequeña modificación del código anterior. Introduciremos una nueva variable llamada “estado” que almacene el estado en el que se dejó el led tras la última pulsación. De esta forma cuando pulsemos, Arduino se acordará si el led estaba encendido o apagado.

```
int estado=0;
```

Usaremos esta “memoria” de Arduino para alternar entre encendido y apagado cada vez que pulsemos. Esto lo haremos modificando el código en la zona de void loop().

ACCEDE AHORA A LOS MEJORES CURSOS

Primero introducimos un “if” que hará que cuando pulsemos el botón alterne el valor de la variable estado entre 0 y 1.

```
if(pulsador==HIGH){
     estado=1-estado;
}
```

La expresión “estado=1-estado” lo que hace es que si el estado era apagado, igual a 0, al pulsar almacenará en la variable el valor 1-0 =1 y si el estado era encendido, igual a 1, al pulsar almacenará 1-1=0.

Después, en vez de comprobar cómo está la variable “pulsador”, como hacíamos antes, lo que comprobamos es el valor que tiene la variable “estado”. Si es 1 encenderá el led y si es 0 lo apagará.

```
if(estado==1) {
     digitalWrite(2, HIGH);
}
else{
     digitalWrite(2, LOW);
```

Os dejamos el código completo para que solo tengáis que copiar y pegar en el IDE.

```
/************************ */
/* Encender LED con Botón */
/*         Interruptor 1        */
/************************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
int pulsador=0;              //almacena el estado del botón
int estado=0;                //0=led apagado, 1=led encendido

//** Programa **//

void setup() {
 pinMode(2, OUTPUT);       //declaramos el pin 2 como salida
 pinMode(4, INPUT);        //declaramos el pin 4 como entrada
}

void loop() {
 pulsador = digitalRead(4); //lee si el botón está pulsado

 if(pulsador==HIGH){       //si el boton es pulsado
   estado=1-estado;
 }

 if(estado==1) {           //si el estado es 1
   digitalWrite(2, HIGH);   //se enciende el led
 }
 else{                      //si el estado es 0
   digitalWrite(2, LOW);   //se apaga el led
 }
}
```

Interruptor con corrección de rebote

Al realizar el anterior ejemplo de interruptor, podréis daros cuenta que hay algunas veces que nuestro botón falla y no hace correctamente su función. Es imposible predecir su comportamiento. A este efecto se le denomina rebote y es debido a varios motivos. Aquí vamos a dar un par de soluciones rápidas y efectivas que nos pueden ayudar en la mayoría de casos, pero en el módulo de nivel intermedio explicaremos una solución al problema del “debounce” más elaborada y fiable.

ACCEDE AHORA A LOS MEJORES CURSOS

Uno de los motivos de este efecto, es que Arduino repite nuestro loop de instrucciones miles de veces cada segundo. Esto provoca que cuando pulsamos el botón una sola vez, Arduino leerá cientos de veces seguidas que hemos pulsado e interpretará que hemos pulsado todas esas veces. Por eso, la acción de encender y apagar se repetirá muy rápidamente, sin que lo apreciemos, hasta dejar el led en la última posición leída.

Solucionar esto es sencillo. Solo tenemos que introducir una nueva variable "pulsadorAnt" donde almacenemos en qué estado anterior se encontraba el botón, pulsado o no, antes de nuestra pulsación.

```
int pulsadorAnt=0;
```

Seguidamente, en nuestro loop, modificaremos el primer "if" que teníamos introduciendo una segunda condición. Diremos que para que Arduino cambie la posición del interruptor, no solo debe leer que el botón está pulsado, "pulsador==HIGH", sino que también debe cumplirse que justo en el instante antes no estuviese pulsado, "pulsadorAnt==LOW". De esta forma Arduino solo leerá nuestra pulsación una sola vez cada vez que pulsemos.

```
if((pulsador==HIGH)&&(pulsadorAnt==LOW)){
      estado=1-estado;
  }
```

Para finalizar, justo después de este "if", actualizaremos el valor de la variable "pulsadorAnt" con el nuevo valor de "pulsador" mediante la siguiente línea.

```
pulsadorAnt=pulsador;
```

Si cargásemos ya este nuevo código a la placa, veríamos que nuestro problema se ha solucionado casi del todo, pero que aún, algunas veces, falla su funcionamiento. Esto es debido a la propia construcción del botón.

Un botón, internamente, no es más que dos láminas metálicas que se unen o separan por la acción de un resorte. En el momento de la unión, o separación, de las

ACCEDE AHORA A LOS MEJORES CURSOS

láminas, el resorte provoca una serie de rebotes entre las láminas que Arduino es capaz de detectar.

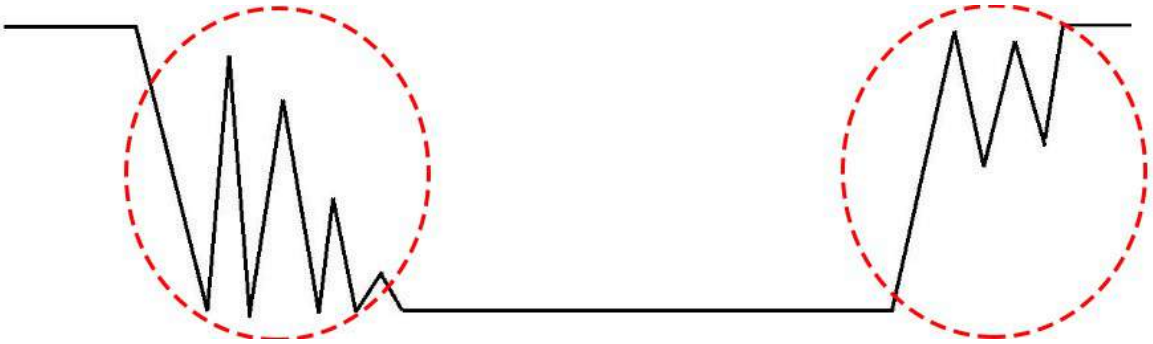


Ilustración 3 Gráfico del efecto rebote en el apagado y encendido de un botón.

Para evitar este problema, una solución sencilla es meter una pequeñísima pausa al programa justo después de que Arduino detecte nuestra pulsación. De esta forma, cuando el código retome su funcionamiento, los rebotes habrán terminado y no serán detectados. En nuestro caso introduciremos una pausa de 40 milisegundos, aunque podéis probar a variar este valor a vuestro gusto para afinar el resultado.

```
delay(40);
```

Podréis observar que aún falla alguna vez pero que su funcionamiento es bastante más preciso que antes.

ACCEDE AHORA A LOS MEJORES CURSOS

Esto es todo. Esperamos que se haya entendido bien y que no quede muy lioso. Al seguir los pasos veréis que es muy fácil y divertido. Os dejamos el código completo.

```
/*************************/
/* Encender LED con Botón */
/* Interruptor sin rebote */
/*************************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//

int pulsador=0;           //almacena el estado del botón
int estado=0;             //0=led apagado, 1=led encendido
int pulsadorAnt=0;        //almacena el estado anterior del boton

//** Programa **//

void setup() {
  pinMode(2, OUTPUT);     //declaramos el pin 2 como salida
  pinMode(4, INPUT);      //declaramos el pin 4 como entrada
}

void loop() {
  pulsador = digitalRead(4); //lee si el botón está pulsado

  if((pulsador==HIGH)&&(pulsadorAnt==LOW)){  //si el boton es pulsado y antes no
lo estaba
       estado=1-estado;
       delay(40);                //pausa de 40 ms
  }
  pulsadorAnt=pulsador;    //actualiza el nuevo estado del boton

  if(estado==1) {          //si el estado es 1
       digitalWrite(2, HIGH);   //se enciende el led
  }
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
  else{                   //si el estado es 0
       digitalWrite(2, LOW);      //se apaga el led
  }
}
```

Comunicación serie

¿Qué es la comunicación serie?

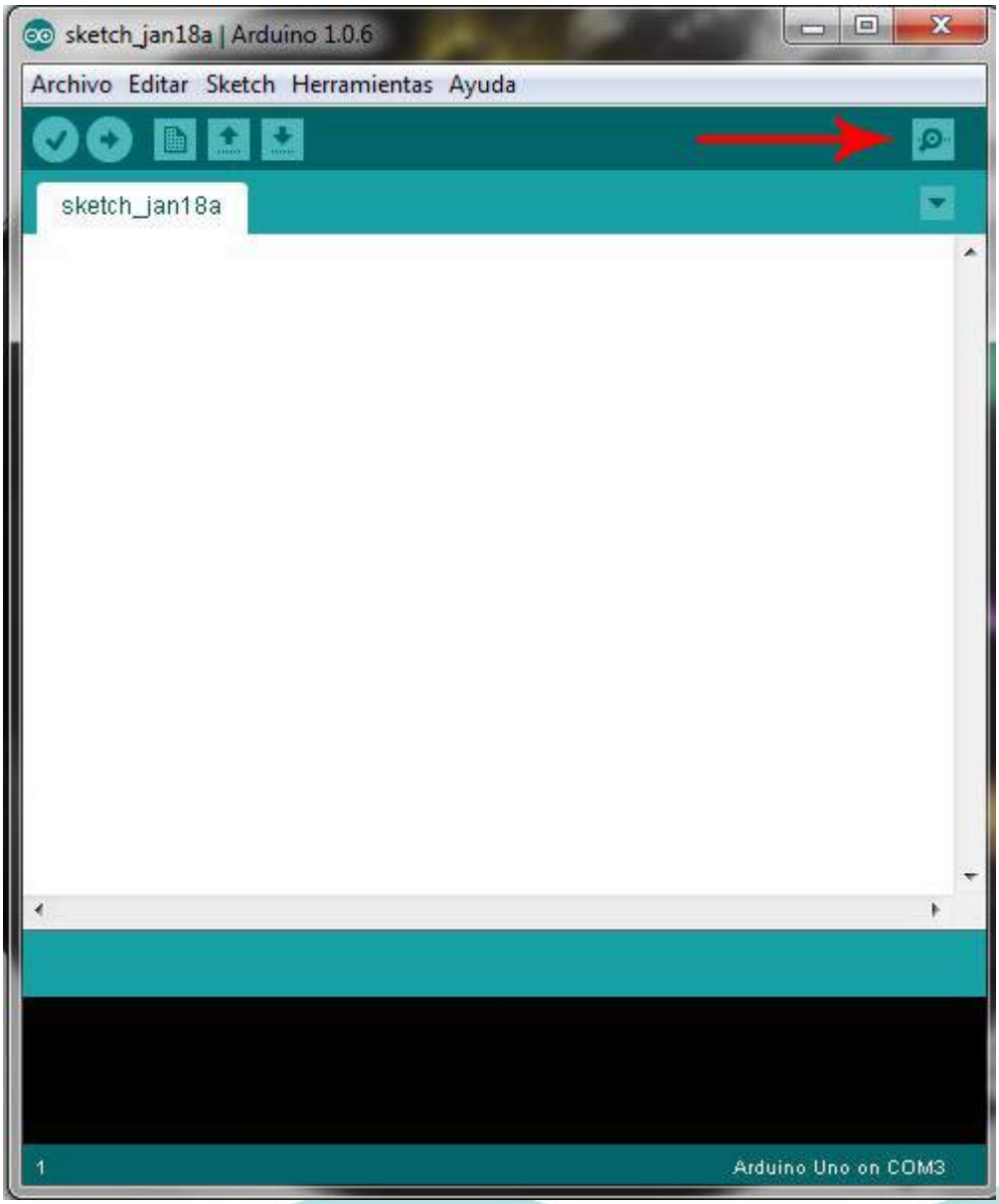
Antes de comenzar, aclararemos que no nos adentraremos mucho en la definición teórica de la comunicación serie ni en sus especificaciones técnicas. Existe en internet una gran cantidad de información muy detallada al respecto y para nuestro cometido, que es aprender a manejar Arduino y poder usarlo para nuestros propios proyectos, no nos interesa complicarnos. Nuestra intención es hacer accesible, fácil y divertido el mundo Arduino.

Una vez comentado esto, definiremos la comunicación serie como una "interfaz de comunicación de datos digitales que nos permite establecer transferencia de información entre varios dispositivos". Esto nos va a permitir que podamos interactuar con nuestro Arduino, recibiendo información y enviándosela nosotros cuando lo necesitemos. Otra cosa que nos será muy útil de esta comunicación, es que podremos darle las órdenes, previamente programadas, que queramos.

¿Dónde tenemos que dirigirnos para comunicarnos en serie con Arduino?

ACCEDE AHORA A LOS MEJORES CURSOS

Para poder establecer esta comunicación usaremos el monitor serial que tenemos en el IDE. Lo encontraremos pulsando el botón  , con el mismo nombre, que tenemos arriba a la derecha.



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 1 Botón de Monitor Serial

Una vez que lo pulsemos, se abrirá el monitor en el que podremos recibir información de Arduino y escribirle nosotros, tanto información que nos solicite como órdenes.

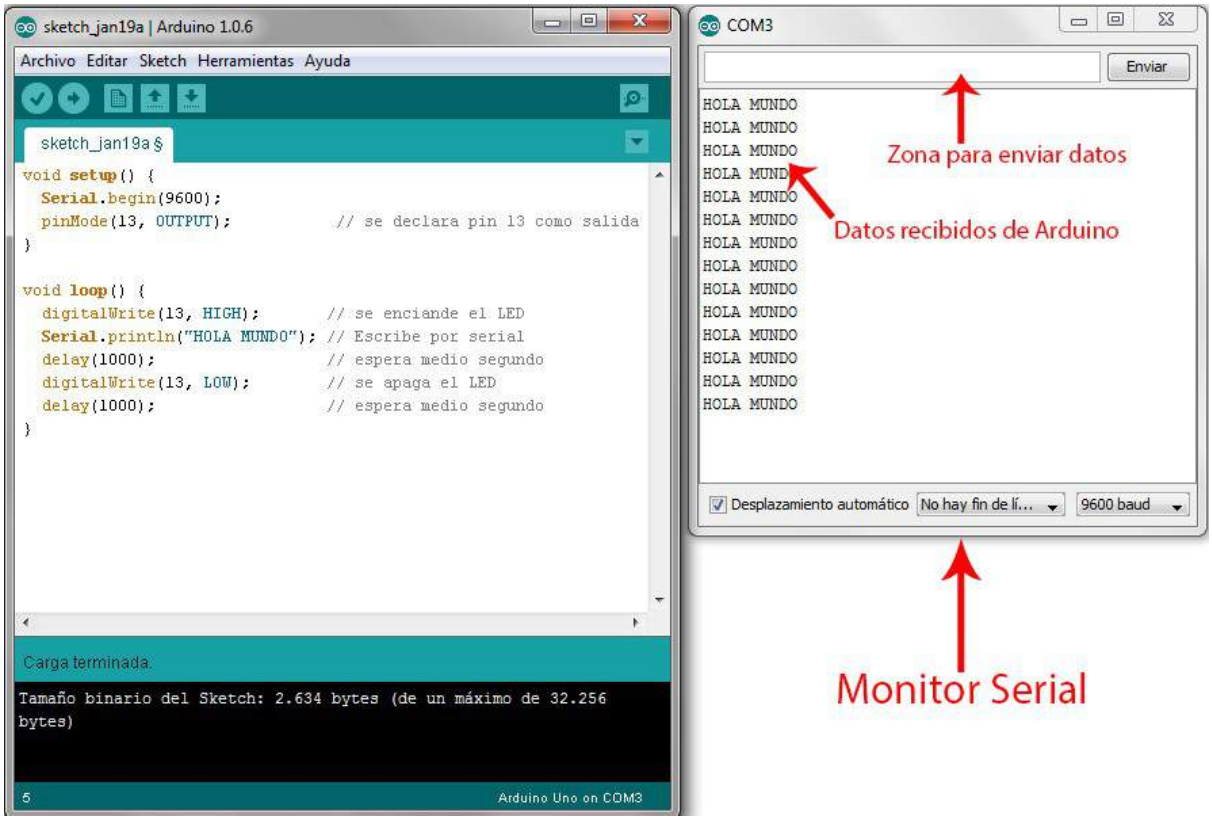


Ilustración 2 Monitor Serial

Conexiones serie en Arduino

En la mayoría de Arduinos podemos encontrar 2 tipos de conexión serie para las comunicaciones. Primero, tendremos los pines de transmisión serie, que en Arduino UNO son los pines 0 (RX) para recibir datos y 1 (TX) para transmitirlos. Con ellos

ACCEDE AHORA A LOS MEJORES CURSOS

podremos, por ejemplo, conectar 2 Arduinos entre sí, conectándolos de manera cruzada, para que trabajen en paralelo o conectar un Arduino con otro dispositivo que también se comunique con este protocolo. Por razones obvias no podremos usar estos pines como salidas digitales mientras los usemos como conexión serial. En segundo lugar, tenemos los puertos USB (Universal Serial Bus) propios de las placas Arduino. Estos puertos, al trabajar con protocolo serial, los podemos usar para conectarnos al PC directamente y facilitarnos así el conexionado de Arduino con éste.

Ejemplo: Ordenar a Arduino que encienda un Led

Para entender mejor todo lo explicado, vamos a realizar un ejemplito muy sencillo pero que nos sirve para darnos cuenta de lo útil y potente que es esta característica de Arduino.

Comenzaremos realizando una simplificación del esquema de montaje que utilizamos en el post 3 para el semáforo, utilizando los mismos elementos pero para un solo Led.

ACCEDE AHORA A LOS MEJORES CURSOS

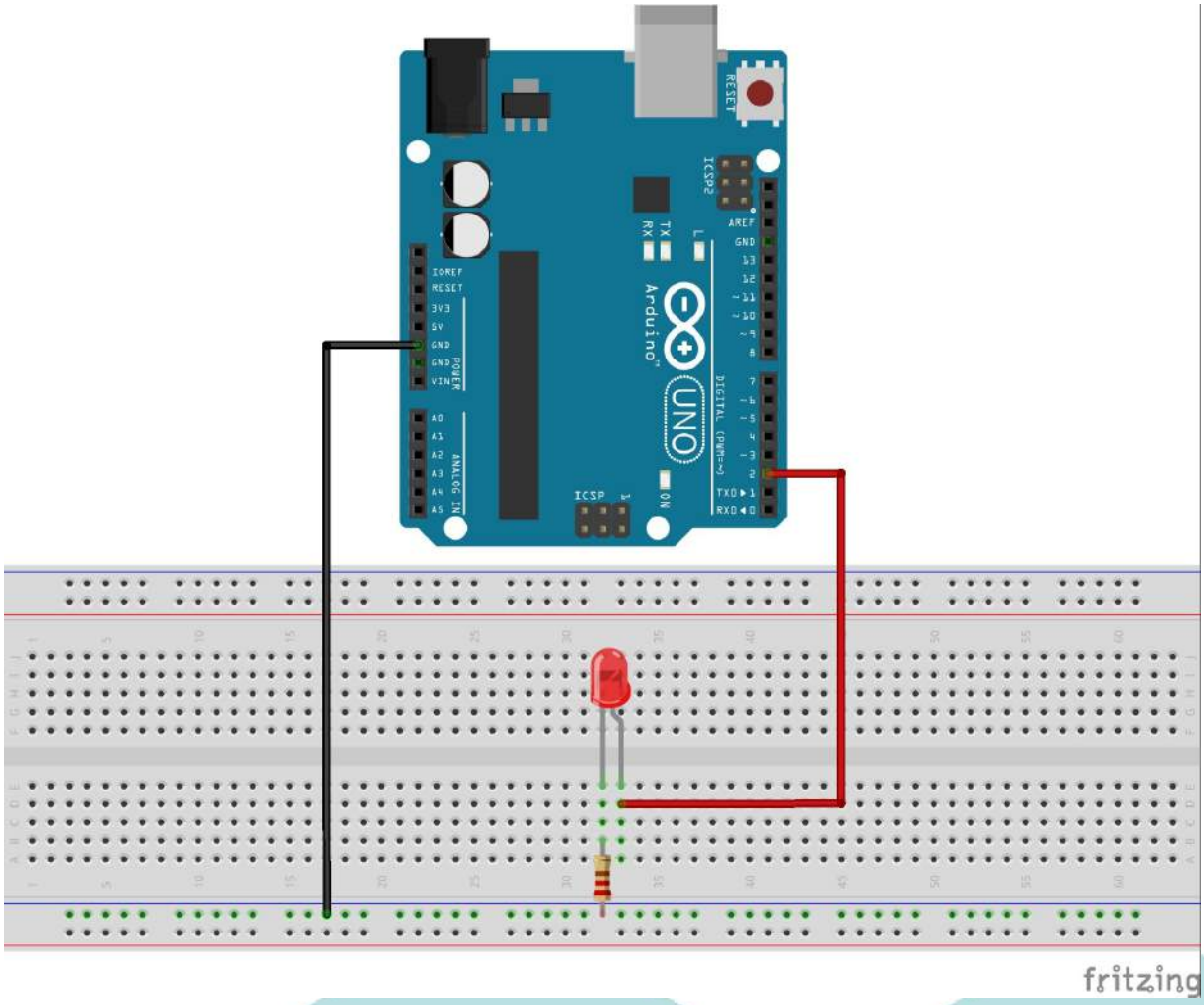


Ilustración 3 Montaje para encender Led

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

Una vez hemos montado el circuito, nos dirigimos al IDE y escribimos el siguiente código, lo enviamos a nuestro Arduino y una vez terminado el proceso haremos click en el botón del monitor serial.

```
/*********************************/
/*  Ordenar Parpadeos de un Led */
/*********************************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//

int parpadeos;
int led = 2;

//** Programa **//

void setup(){
  Serial.begin(9600);     // Inicia la comunicación serial
  pinMode(led, OUTPUT);
}

void loop(){
  if (Serial.available()>0){           //   Comprueba   si   el   serial   está
disponible
      parpadeos =  Serial.read()-48;         // leemos el número de parpadeos
      if (parpadeos >= 1 && parpadeos <= 9)   // Si el valor introducido está
entre 1 y 9
      {
      Serial.print("Se van a dar ");
      Serial.print(parpadeos);
      Serial.println(" parpadeos");
      delay(1500);
      for(int i=0;i<parpadeos;i++){          // Realiza los parpadeos
      digitalWrite(led, HIGH);
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
        delay(100);
        digitalWrite(led, LOW);
        delay(200);
        }
        }
        else {                                  //   Si   introducimos   un   valor
erroneo
        Serial.println("El dato introducido es incorrecto");
        }
   }
}
```

Este programa lo que hace es que nuestro Led parpadee el número de veces que le digamos. Para ello introduciremos valores, entre 1 y 9, por el monitor serial y pulsamos enter.

Para realizar el código, en primer lugar, hemos declarado las variables que vamos a usar, como en anteriores ejemplos. En la zona de setup hemos iniciado la comunicación serial con el comando “Serial.begin(9600)” para que Arduino sepa que nos comunicaremos con él a la velocidad de 9600 bits/seg (la velocidad es variable siempre que se establezca la misma en el código y en el receptor de la señal). Esto es necesario ponerlo siempre que vayamos a usar el monitor serial, ya que sin esta línea Arduino no se comunicará por serial. También hemos configurado nuestro pin del Led como salida. Usaremos el pin 2.

En el cuerpo de la función loop, comenzamos con la función “if (Serial.available()>0)” que comprueba si hemos enviado datos por serial para que, en nuestro caso, solo nos devuelva datos cuando los pedimos. Una vez enviado el número, con la expresión “parpadeos = Serial.read()-48” almacenaremos el valor que hemos enviado en la variable “parpadeos”. Como habréis observado, al valor que lee del serial le restamos 48. Esto es debido a que Arduino lee los caracteres en código ASCII, el cual podéis ver aquí, y en este código el número cero equivale al valor 48. De esta forma transformamos los números de ASCII a decimales.

ACCEDE AHORA A LOS MEJORES CURSOS

En OpenWebinars.net tenemos todos estos cursos a tu entera disposición

Linux LPIC-1
Examen 101

Linux LPIC-1
Examen 102

NodeJS,
ExpressJS y
MongoDB

AngularJS y
TypeScript

Apps móviles con
PhoneGap

Servidores VoIP
con Asterisk

Desarrollo
Frontend
Profesional

Virtualización de
Servidores con
Promox

Apps Móviles con
Titanium Alloy

Desarrollo
Backend con
Django

ACCEDE AHORA A LOS MEJORES CURSOS

Continuaremos con “if (parpadeos >= 1 && parpadeos <= 9)” imponiendo que el programa solo funcione si introducimos un número mayor o igual 1 y menor o igual 9, en caso contrario nos devolverá el mensaje “El dato introducido es incorrecto”.
Mediante la función “Serial.print()” Arduino nos imprimirá en el monitor serial lo que pongamos dentro del paréntesis. Ya sea una frase, que habremos de introducir entre comillasSerial.print(“frase”), o una variable, para lo que pondremos directamente su nombre sin comillasSerial.print(variable). Escrito así nos irá imprimiendo todo en la misma línea, pero si escribimos “Serial.println()” lo que haremos será que, tras esa impresión, la siguiente se empezará a escribir en la línea de abajo. Usaremos esta función para que Arduino se comunique con nosotros.
Finalmente definiremos un bucle for. En nuestro caso “for (int i=0;i<parpadeos;i++)”, para que iniciando el valor de “i” en 0, si “i” es menor que el valor de parpadeos introducido, encenderá y apagará el Led una vez e incrementará el valor de “i” en 1 cada vez que pase por el bucle. De esta forma conseguimos que el Led parpadee las veces que le hemos ordenado.
Divertido ¿verdad?.

5.- Ejemplo: Reconocimiento de valores RGB por puerto serie

En el siguiente ejemplo, utilizaremos la comunicación serial para que Arduino nos dé información, que necesitamos, de nuestro circuito. Le pediremos que nos dé el valor que tienen 3 potenciómetros conectados a un Led RGB para saber el color exacto que está dando. Éste es un ejemplo bastante usado en distintos cursos para enseñar cómo funciona la comunicación serial en Arduino, pero nos parece útil y divertido realizarlo con vosotros.

ACCEDE AHORA A LOS MEJORES CURSOS

Con este circuito, controlaremos qué cantidad de rojo, verde y azul dará nuestro Led y además, gracias a la comunicación serial, Arduino nos dirá que color exacto está dando y podremos usar estos datos con otros programas o decirle nosotros al sistema que dé un color exacto que queremos.

Antes de comenzar, explicaremos que es un Led RGB. Es un Led que dentro lleva 3 colores independientes que mezclan su intensidad para proporcionar casi toda la gama de colores. Existen de dos tipos, de cátodo común y de ánodo común. Esto solo quiere decir que los 3 Leds internos que lleva el RGB comparten una de las conexiones, ya sea el cátodo que se conecta a tierra o el ánodo que se conecta a la alimentación. Es fácil identificar cual es la conexión compartida, ya que estos Leds tienen 4 filamentos y el más largo es el común. Los otros 3 filamentos son los que controlan cada color. Da igual el tipo que usemos, siempre que los conectemos correctamente.

ACCEDE AHORA A LOS MEJORES CURSOS

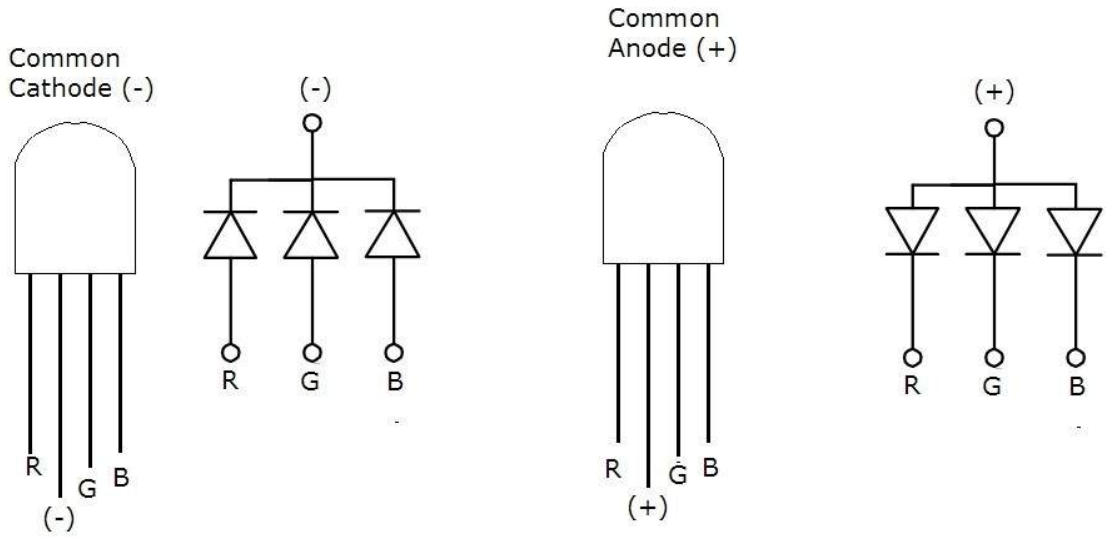


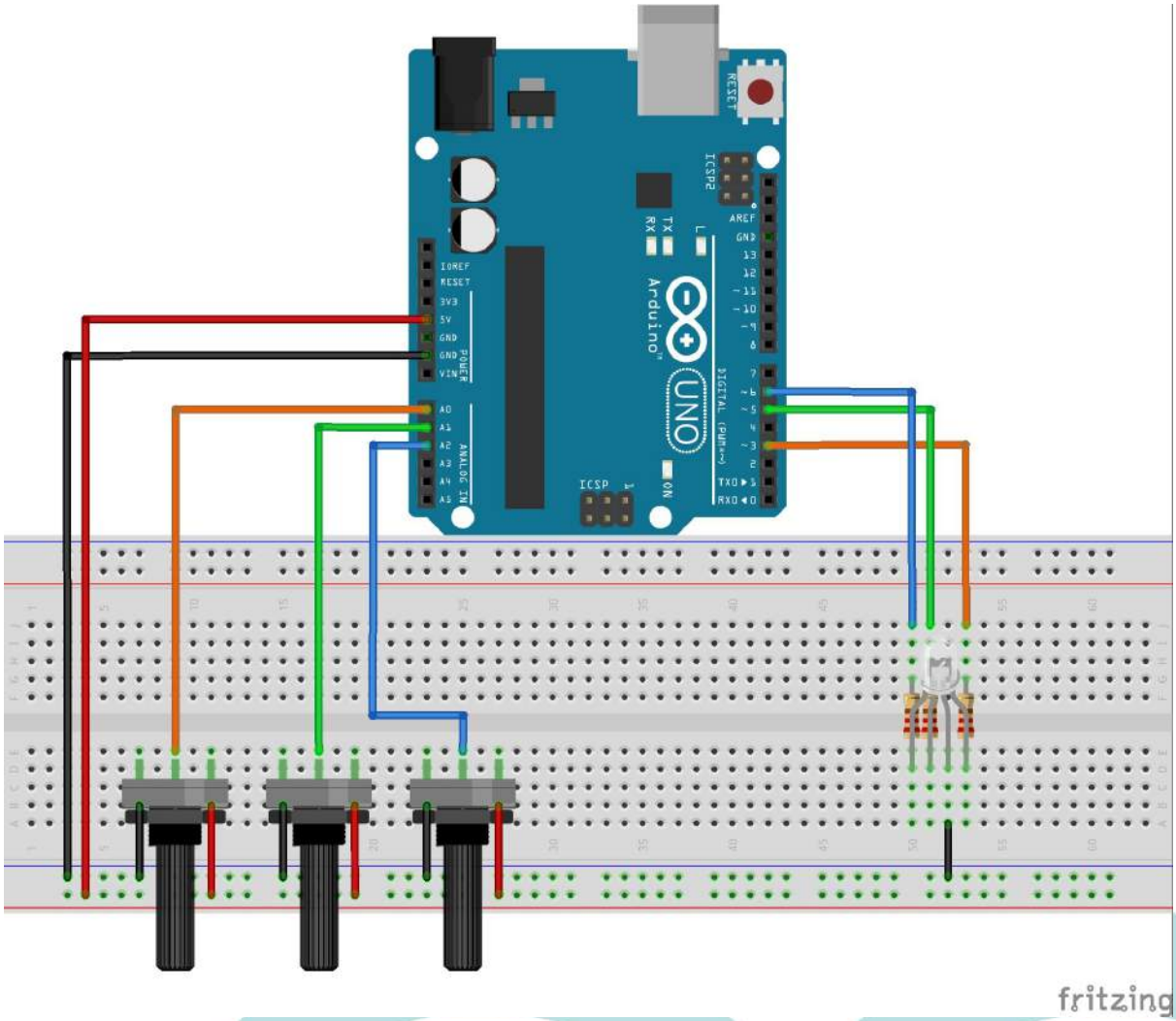
Ilustración 4 Conexiones de Led RGB (arduino utfsm)

Para realizar este ejemplo necesitaremos los siguientes elementos:

- 1 x Arduino UNO R3
- 1 X Protoboard
- 1 x led RGB (de cátodo común)
- 3 x Resistencia de 220Ω (puede valer de 330Ω)
- 3 x Potenciómetros lineales de 10kΩ
- Cables para conectar todo

El circuito que debemos montar es el siguiente.

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 5 Montaje de control de Led RGB

Hay que resaltar, que conectaremos cada ánodo a pines digitales PWM del Arduino para controlar la intensidad que enviamos a cada uno. Nosotros usaremos los pines digitales 3, 5 y 6, y los analógicos 0, 1, y 2.

Finalmente nos dirigiremos al IDE, escribiremos el siguiente código y lo cargaremos a nuestro Arduino. Una vez cargado pulsaremos en el botón de monitor serial.

```
/*********************************/
/*  Control de Led RGB con Serial */
/*********************************/

/*** Fernando Martinez Mendoza ***/

//** Definiciones **//

const int ledRojo=3;      // definimos los pines digitales para el Led RGB
const int ledVerde=5;
const int ledAzul=6;
const int potRojo=0;      //   definimos   los   pines   analogicos   para   los
potenciometros
const int potVerde=1;
const int potAzul=2;
int rojo;                 //  definimos  variables  para  los  valores  de  cada
potenciometro
int verde;
int azul;

//** Programa **//

void setup(){
  Serial.begin(9600);     // inicia la comunicación serial
  pinMode(ledRojo, OUTPUT);  // configuramos los pines digitales como salida
  pinMode(ledVerde, OUTPUT);
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
  pinMode(ledAzul, OUTPUT);
}

void loop(){
  rojo=analogRead(potRojo) / 4;  // leemos el valor de cada potenciómetro
  verde=analogRead(potVerde) / 4;  // y lo almacenamos
  azul=analogRead(potAzul) / 4;

  analogWrite(ledRojo,rojo);
  analogWrite(ledVerde,verde);
  analogWrite(ledAzul,azul);

  Serial.print("ROJO: ");
  Serial.print(rojo);      // si fuese ánodo común sería 255-rojo
  Serial.print(" / ");
  Serial.print("VERDE: ");
  Serial.print(verde);     // si fuese ánodo común sería 255-verde
  Serial.print(" / ");
  Serial.print("AZUL: ");
  Serial.println(azul);    // si fuese ánodo común sería 255-azul

  delay(300);
}
```

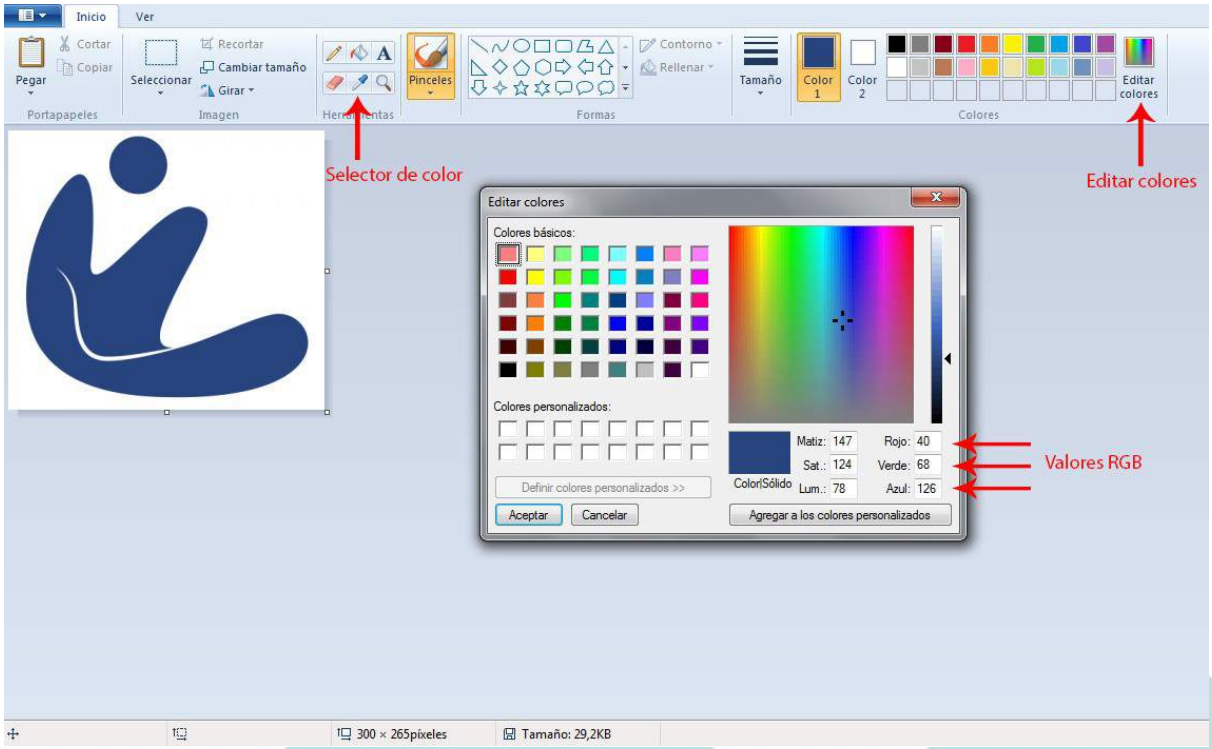
Si prestamos atención al código, vemos que no tenemos funciones que no conozcamos. Solo destacar que, como la función analogRead lee valores entre 0-1023 y analogWrite los escribe entre 0-255, para transformar los valores de uno a otro dividiremos el valor leído por analogReadentre 4 para obtener una relación aproximada y suficientemente buena. El resto del programa, con lo aprendido en los post anteriores, se entiende perfectamente.

Una vez que tenemos todo listo, podemos ver cómo, mientras movemos los potenciómetros, el Led RGB va creando distintos colores y en el monitor serial nos aparecen los valores de intensidad que se están utilizando para conseguirlos. Con

ACCEDE AHORA A LOS MEJORES CURSOS

estos valores podemos dirigirnos a cualquier programa de edición gráfica y utilizar ese color exacto, o viceversa.

Probaremos a conseguir el color del logo de OpenWebinars.net. Primero, abrimos el logo con Paint y haremos clic en la herramienta “Selector de color”. Picaremos en el color del logo y después pulsaremos en “Editar colores”. En la ventana emergente ya podremos ver los valores RGB de nuestro color.



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 6 Selección de valores RGB de una imagen usando Paint

Ahora, si metemos esos valores en Arduino utilizando los potenciómetros, nuestro Led RGB brillará con el color que buscábamos. Podemos jugar con los colores y usarlo como más nos divierta.

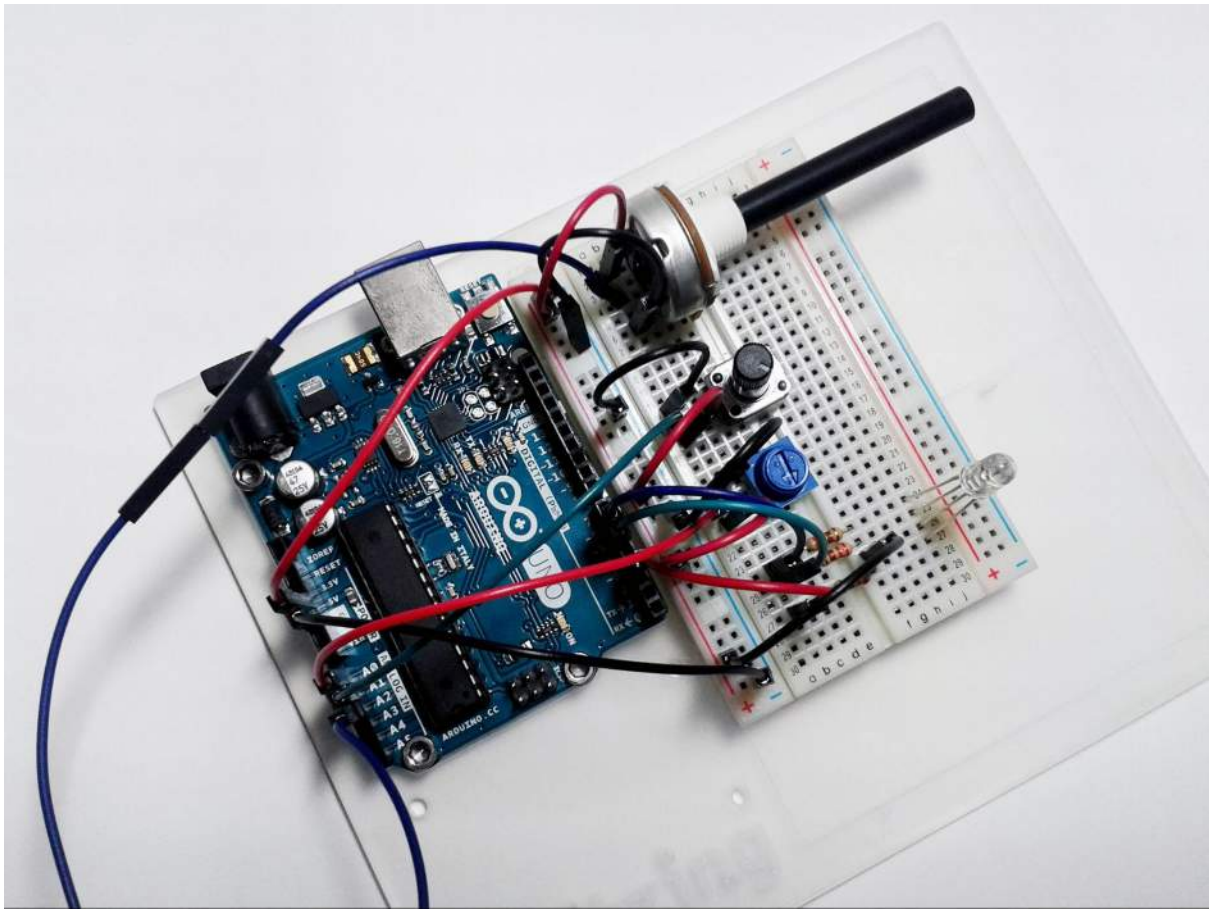


Ilustración 7 Montaje de control Led RGB

Sonidos con Arduino

Función tone

Con Arduino también podemos crear sonidos gracias a la función tone cuya sintaxis es la siguiente

ACCEDE AHORA A LOS MEJORES CURSOS

```
tone(pinsalida,frecuencia);
```

Esta función lo que hace es intercambiar valores HIGH/LOW a la frecuencia deseada en el pin seleccionado hasta que la volvemos a llamar con otra frecuencia o le ordenamos que pare con esta función

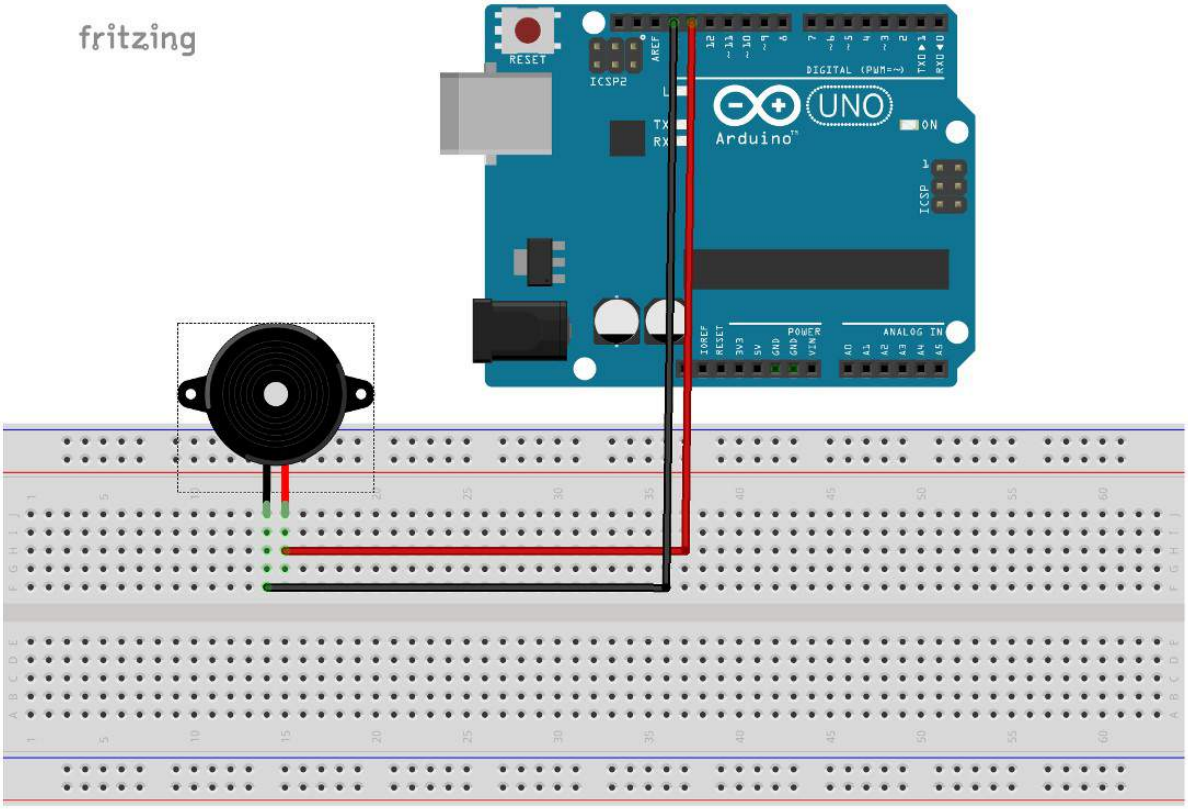
```
noTone(pinsalida);
```

Tenemos que tener en cuenta que sólo es capaz de generar un único tono a la vez. Si llamamos a la función tone para que genere sonidos en otro pin sin haber detenido el que está sonando no ocurrirá nada.

Zumbador

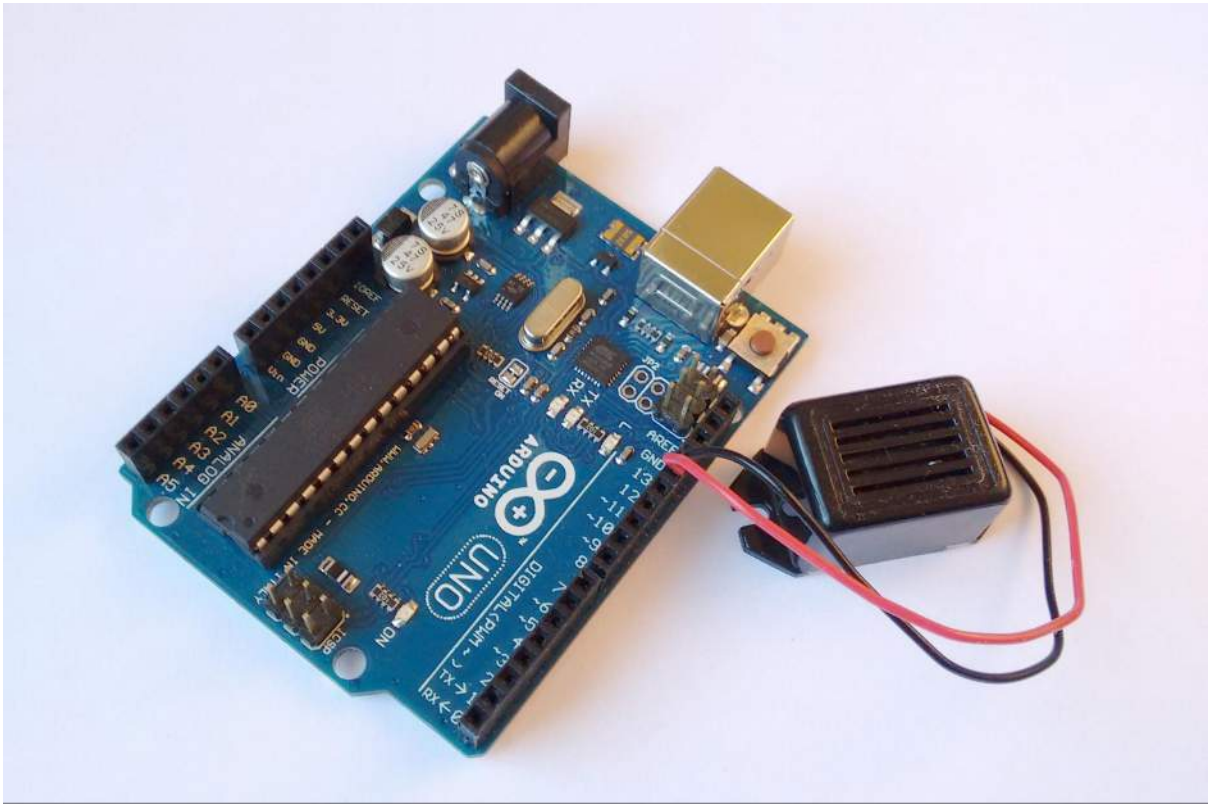
Un zumbador es un elemento parecido a un altavoz pero sólo emite zumbidos (típico sonido que emiten los electrodomésticos). Su precio suele rondar los 2€.
Para hacerlo sonar utilizaremos los pines 13 y GND. El montaje no merece más explicación que la de la imagen

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars



Ya sólo falta crear el código y cargarlo. En la función loop hacemos la llamada a tone para que suene y noTone para detener el sonido. Cambiando los valores de la función delay haremos que suene durante más o menos tiempo.

```
int pinzumbador = 13;    // pin del zumbador
int frecuencia = 220;      // frecuencia correspondiente a la nota La

void setup()
{
}

void loop()
{
   tone(pinzumbador,frecuencia);    // inicia el zumbido
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
    delay(2000);
    noTone(pinzumbador);          // lo detiene a los dos segundos
    delay(1000);
}
```

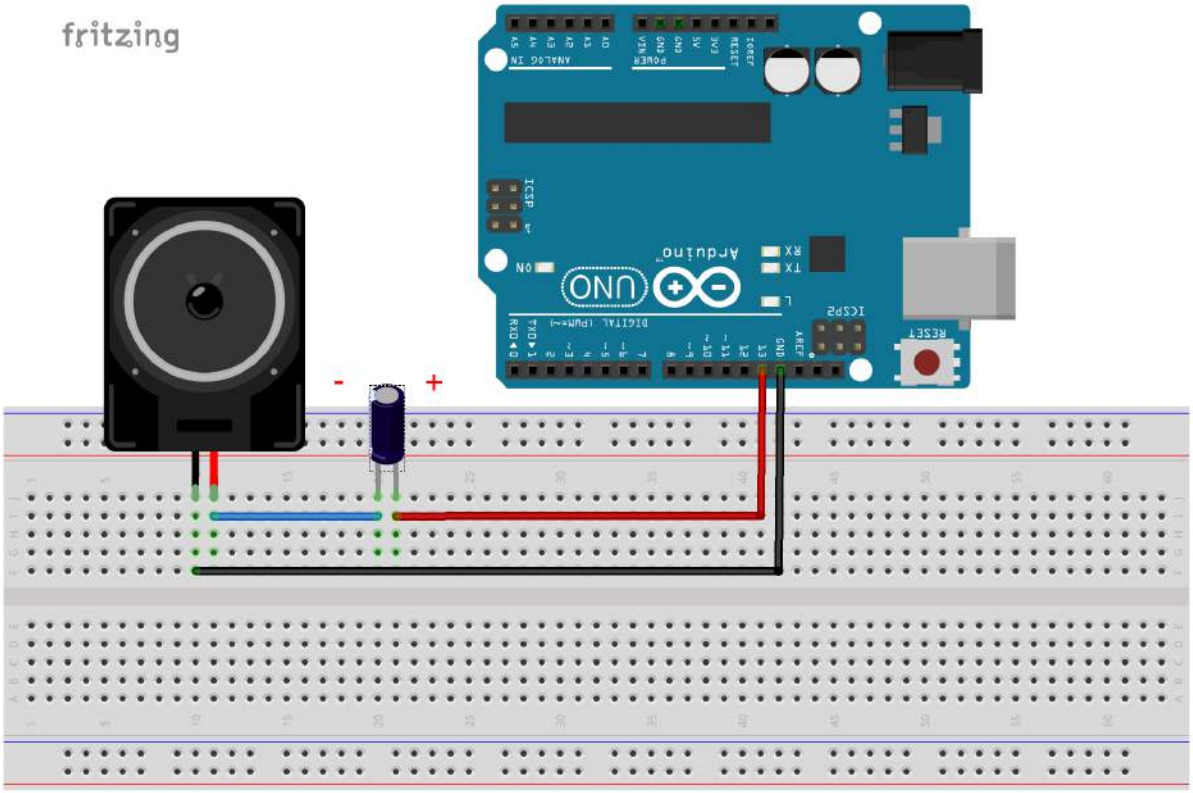
Podemos observar dos cosas:

- Al cargar el programa suena brevemente el zumbador. Esto se debe a que utilizamos el pin 13 y éste se pone a HIGH brevemente unas veces cuando se inicia la placa.
- Si variamos la frecuencia a penas varía el tono o no suena. La causa de ésto es que un zumbador tiene poca capacidad para reproducir sonidos fielmente. La frecuencia influye tan poco que incluso sonaría conectándolo entre los pines 5V y GND. Para frecuencias demasiado altas el zumbador no responde.

Música

En este punto utilizaremos un pequeño altavoz. Podéis aprovechar uno que tengáis en algún aparato que no utilicéis.
El mismo montaje servirá para los dos ejemplos siguientes. El primero consistirá en reproducir una escala de 220 a 440 HZ con todos sus semitonos. En el segundo reproduciremos música.

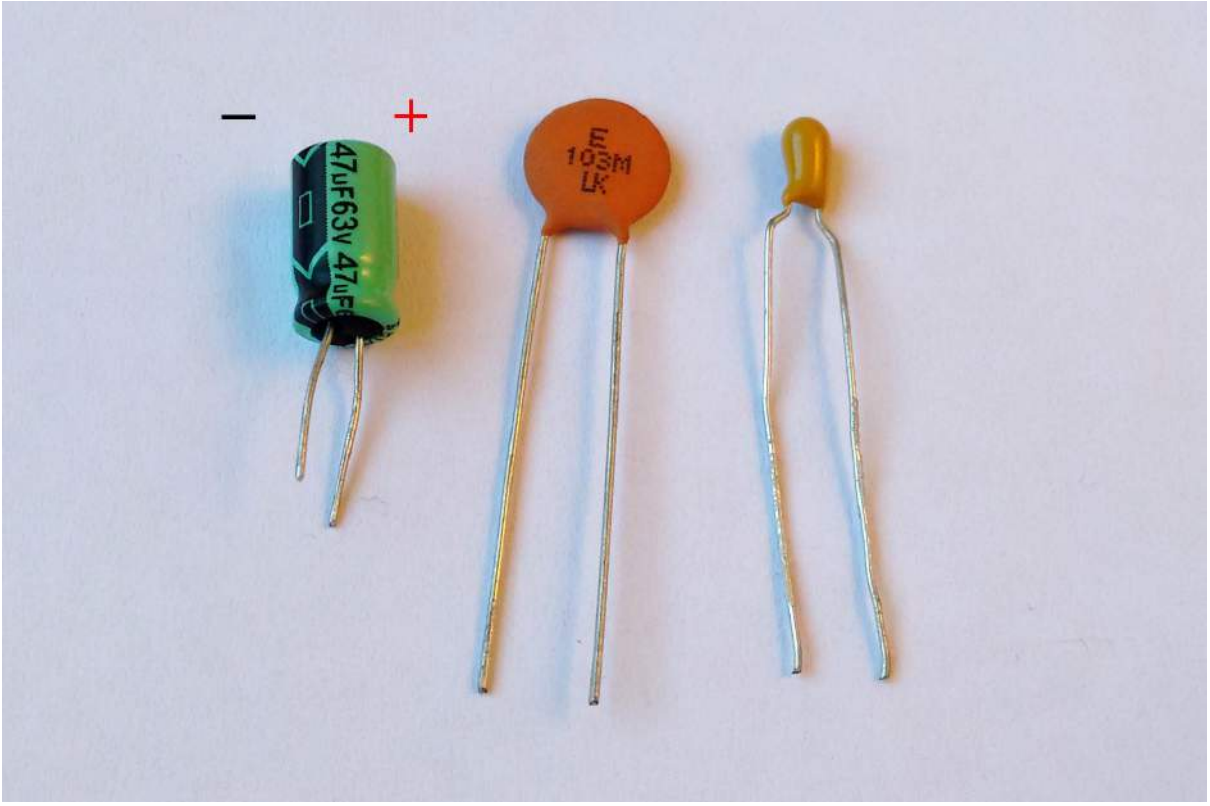
ACCEDE AHORA A LOS MEJORES CURSOS



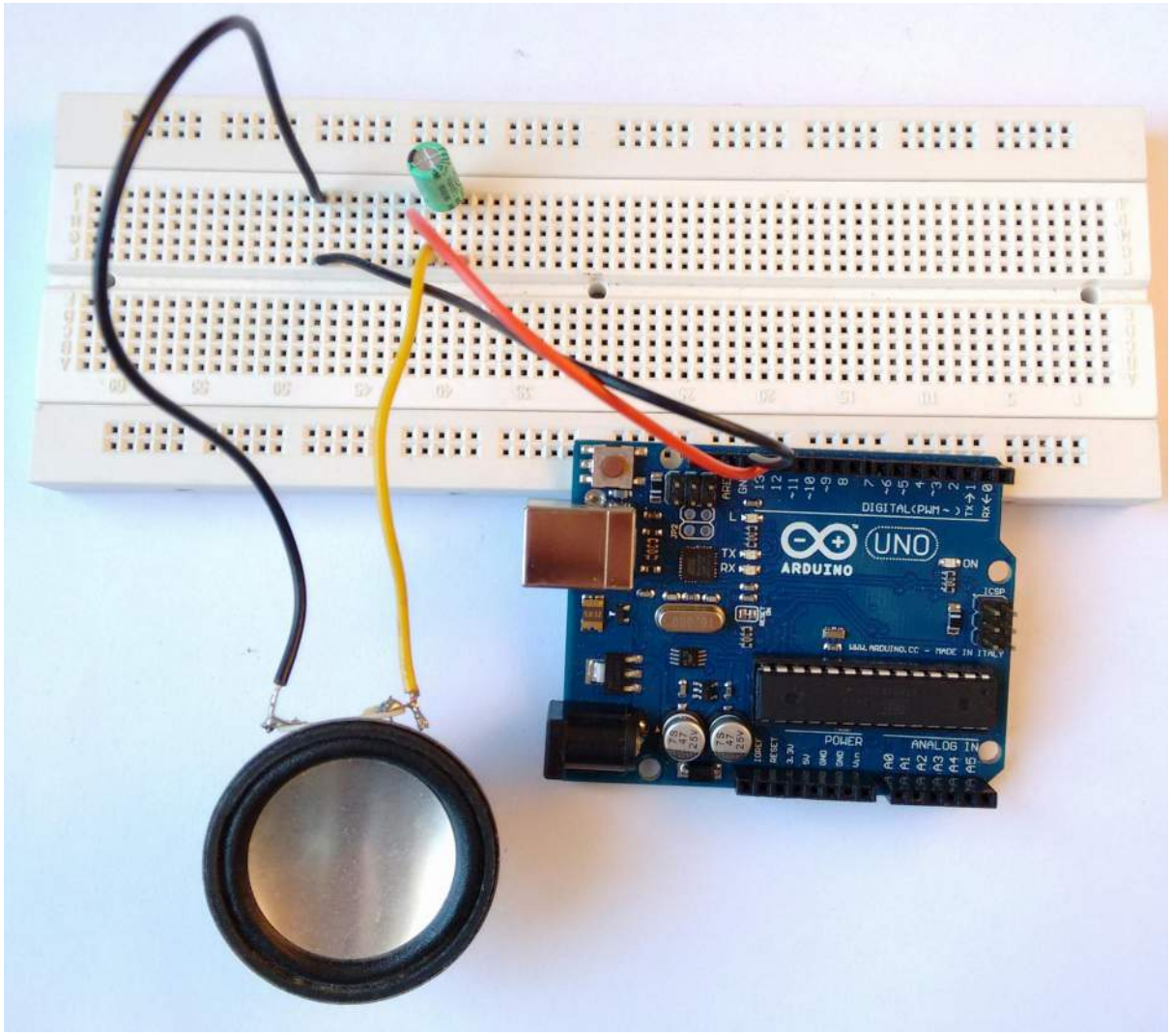
Si os fijáis en el esquema he puesto un condensador (~0.20€) en serie con el altavoz. Ésto es para eliminar cualquier componente de continua. Tened en cuenta que si elegís valores muy pequeños puede recortar las frecuencias más bajas. Con valores de 47 uF o mayores va bien. IMPORTANTE:Si usáis condensadores electrolíticos (los cilíndricos) tenéis que tener en cuenta que tienen polaridad y si lo colocáis al revés pueden explotar. Fijaros que tienen una franja con signos “-” en la pata negativa.

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars



ACCEDE AHORA A LOS MEJORES CURSOS



Vamos con el primer ejemplo. Para no almacenar todas las frecuencias y tener que escribir la función tone en nuestro código cada vez que queramos una nueva nota, haremos un pequeño truco. Almacenaremos sólo el valor de la frecuencia inicial, y las sucesivas notas tendrán la frecuencia de la anterior multiplicada por 1,059. De este modo escribiremos una sola vez la función para hacerlo sonar y un bucle for será el encargado de ir incrementando el valor de la frecuencia.

ACCEDE AHORA A LOS MEJORES CURSOS

```
/***********************/
/* recorrido de octava */
/***********************/

int pinaltavoz = 13;
int frecuencia=220;    // frecuencia correspondiente a la nota La
int contador;          // variable para el contador
float m=1.059;         // constante para multiplicar frecuencias

void setup()
{
}

void loop()
{
    for(contador=0,frecuencia=220;contador<12;contador++)
    {
        frecuencia=frecuencia*m;     // actualiza la frecuencia
        tone(pinaltavoz,frecuencia); // emite el tono
        delay(1500);                 // lo mantiene 1.5 segundos
        noTone(pinaltavoz);          // para el tono
        delay(500);                  // espera medio segundo
    }
}
```

Ahora pasamos a lo bueno: reproducir música, pero de la que recuerda a los videojuegos con música de 8 bits.

En este código utilizaremos una función auxiliar que llamaremos nota con la siguiente estructura

```
nota(frecuencia,duración);
```

ACCEDE AHORA A LOS MEJORES CURSOS

La ventaja es que nos ahorrará escribir decenas de veces la función delay para para indicar el tiempo que debe durar la nota. Cada vez que llamemos a esa función se ejecutará esta parte del código

```
void nota(int frec, int t)
{
   tone(spk,frec);      // suena la nota frec recibida
   delay(t);            // para después de un tiempo t
}
```

Os dejo varios fragmentos de canciones. Podéis buscar partituras de canciones y pasar las notas a frecuencia y tiempo.

```
/************************ */
/*    popurri para Arduino    */
/************************/

/*************** Antonio Guillermo Pérez Coronilla ***************/

/* declaración de variables */
int spk=13;                              // altavoz a GND y pin 13
int c[5]={131,262,523,1046,2093};        // frecuencias 4 octavas de Do
int cs[5]={139,277,554,1108,2217};       // Do#
int d[5]={147,294,587,1175,2349};        // Re
int ds[5]={156,311,622,1244,2489};       // Re#
int e[5]={165,330,659,1319,2637};        // Mi
int f[5]={175,349,698,1397,2794};        // Fa
int fs[5]={185,370,740,1480,2960};       // Fa#
int g[5]={196,392,784,1568,3136};        // Sol
int gs[5]={208,415,831,1661,3322};       // Sol#
int a[5]={220,440,880,1760,3520};        // La
int as[5]={233,466,932,1866,3729};       // La#
int b[5]={247,494,988,1976,3951};        // Si
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
void nota(int a, int b);              // declaracion de la funcion auxiliar. Recibe dos numeros
enteros.

void setup()
{
/************************************ */
/*              HARRY POTTER              */
/************************************/
nota(b[2], 500);
nota(e[3],1000);
nota(g[3], 250);
nota(fs[3],250);
nota(e[3],1000);
nota(b[3],500);
nota(a[3],1250);
nota(fs[3],1000);
nota(b[2], 500);
nota(e[3],1000);
nota(g[3],250);
nota(fs[3],250);
nota(d[3],1000);
nota(e[3],500 );
nota(b[2],1000 );
noTone(spk); delay(1000);
nota(b[2], 500);
nota(e[3],1000);
nota(g[3], 250);
nota(fs[3],250);
nota(e[3],1000);
nota(b[3],500);
nota(d[4],1000);
nota(cs[4],500);
nota(c[4],1000);
nota(a[3],500);
nota(c[4],1000);
nota(b[3],250);
nota(as[3],250);
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
nota(b[2],1000);
nota(g[3],500);
nota(e[3],1000);
noTone(spk);
delay(2000);

/******************/
/*     STAR WARS      */
/******************/
/**** tema principal ****/
nota(d[1],150);noTone(spk);delay(50);
nota(d[1],150);noTone(spk);delay(50);
nota(d[1],150);noTone(spk);delay(50);
nota(g[1],900);noTone(spk);delay(150);
nota(d[2],900);noTone(spk);delay(50);
nota(c[2],150);noTone(spk);delay(50);
nota(b[1],150);noTone(spk);delay(50);
nota(a[1],150);noTone(spk);delay(50);
nota(g[2],900);noTone(spk);delay(150);
nota(d[2],900);noTone(spk);delay(100);
nota(c[2],150);noTone(spk);delay(50);
nota(b[1],150);noTone(spk);delay(50);
nota(a[1],150);noTone(spk);delay(50);
nota(g[2],900);noTone(spk);delay(150);
nota(d[2],900);noTone(spk);delay(100);
nota(c[2],150);noTone(spk);delay(50);
nota(b[1],150);noTone(spk);delay(50);
nota(c[2],150);noTone(spk);delay(50);
nota(a[1],1200);noTone(spk);delay(2000);

/**** marcha del imperio   ****/
nota(g[2],500);noTone(spk);delay(100);
nota(g[2],500);noTone(spk);delay(100);
nota(g[2],500);noTone(spk);delay(100);
nota(ds[2],500);noTone(spk);delay(1);
nota(as[2],125);noTone(spk);delay(25);
nota(g[2],500);noTone(spk);delay(100);
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
nota(ds[2],500);noTone(spk);delay(1);
nota(as[2],125);noTone(spk);delay(25);
nota(g[2],500);
noTone(spk);delay(2000);

/**********************/
/*     entre dos aguas     */
/**********************/
nota(a[1],400);noTone(spk);delay(400);
nota(e[1],400);noTone(spk);delay(400);
nota(a[1],400);noTone(spk);delay(200);
nota(e[1],200);noTone(spk);delay(200);
nota(a[1],200);noTone(spk);delay(200);
nota(as[1],100);noTone(spk);delay(100);
nota(b[1],400);noTone(spk);delay(400);
nota(fs[1],400);noTone(spk);delay(400);
nota(b[1],400);noTone(spk);delay(200);
nota(fs[1],200);noTone(spk);delay(200);
nota(b[1],200);noTone(spk);delay(200);
nota(as[1],100);noTone(spk);delay(100);
nota(a[1],400);noTone(spk);delay(400);
nota(e[1],400);noTone(spk);delay(400);
nota(a[1],400);noTone(spk);delay(400);
}

void nota(int frec, int t)
{
   tone(spk,frec);      // suena la nota frec recibida
   delay(t);            // para despues de un tiempo t
}

void loop()
{
}
```

ACCEDE AHORA A LOS MEJORES CURSOS

¿Qué es el sistema OneWire?

En este post, realizaremos un ejemplo de medición de temperatura con el sensor ds18b20. Este sensor, aparte de tener una buena precisión y una versión sumergible, funciona utilizando un sistema de conexionado muy particular. Este sistema se denomina OneWire. Con él conseguimos enviar y recibir datos por un único cable. Esto tiene sus ventajas e inconvenientes, pero no deja de ser una opción muy a tener en cuenta cuando vamos justos de conexiones de control para nuestro proyecto, ya que nos ahorramos un cable en cada sensor.

En otro ejemplo, ya realizamos mediciones de temperatura con un sensor. Lo interesante de este ejemplo no es el medir la temperatura, si no aprender el funcionamiento del sistema OneWire, para poder usarlo en cualquier sensor que lo utilice, y la instalación e importado de librerías de terceros.

Existen muchos detractores de este sistema, que ponen de manifiesto que el usarlo exige unos códigos algo complicados…y que el conexionado no es demasiado intuitivo… La verdad es que ni son tan complicados los códigos, ya que las librerías propias y los ejemplos que traen estos sensores son una buena herramienta, ni es tan complicada su conexión, solo hay que seguir el esquema que os daremos posteriormente. Como veréis en internet la mayoría de errores y problemas de las personas que los usan radican en un mal conexionado.

¿Cómo usar las librerías y los ejemplos?

Esta vez, usaremos unas librerías en nuestro proyecto para controlar el sensor térmico con el sistema OneWire que os hemos comentado. En nuestro post 3, os definimos qué eran las librerías y ahora os explicaremos como usarlas. Arduino tiene una serie de librerías genéricas que podemos usar simplemente escribiendo al

ACCEDE AHORA A LOS MEJORES CURSOS

principio de nuestro código la sentencia #include <librería.h>repitiendo con todas las librerías que necesitemos.

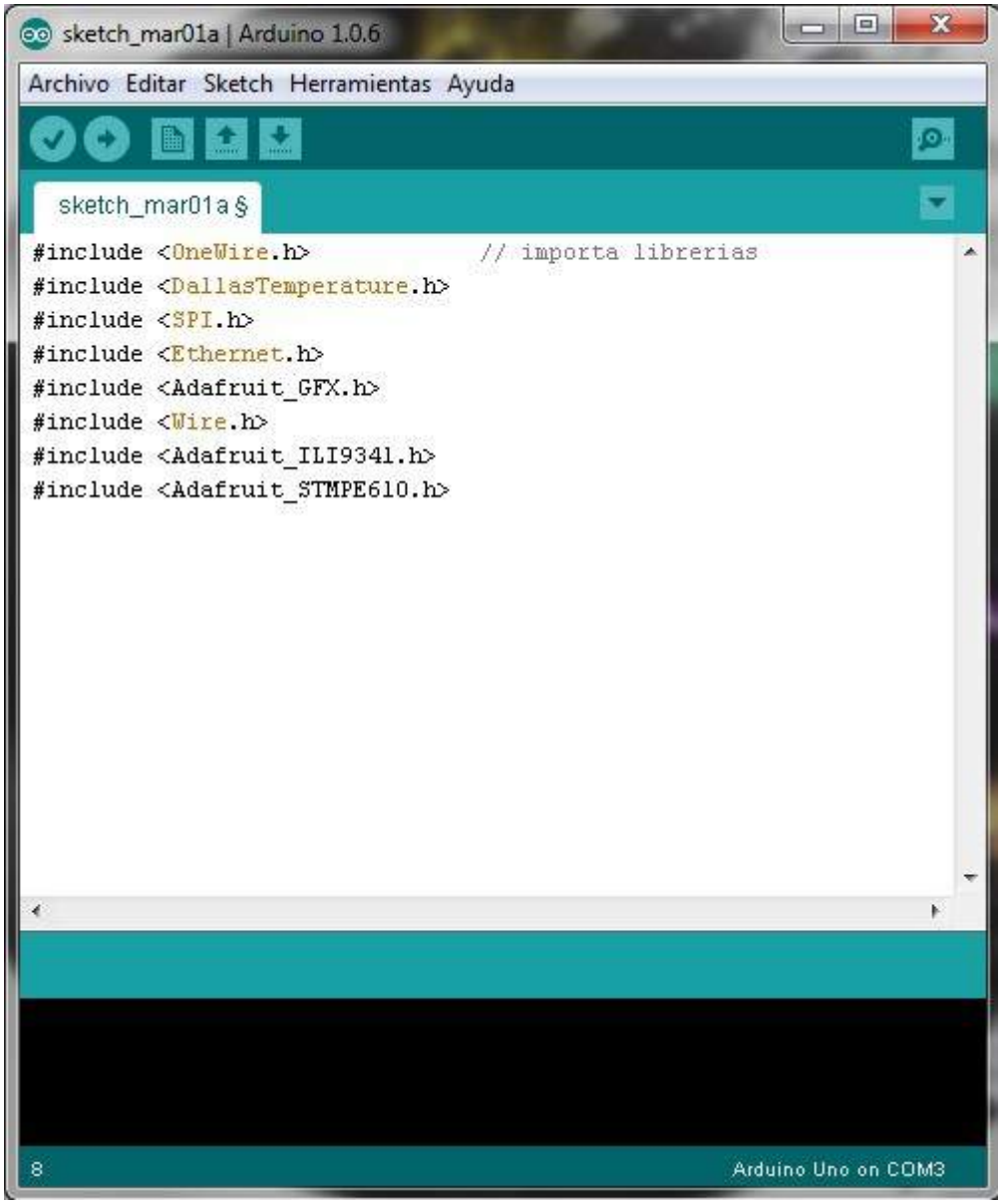


Ilustración 1 Importación de librerías en IDE Arduino

ACCEDE AHORA A LOS MEJORES CURSOS

En este ejemplo vamos a usar las librerías propias del sensor en cuestión, en nuestro caso usaremos las librerías DallasTemperature y OneWire. Para instalar las librerías propias, o de terceros, una vez descargadas existen 2 métodos:

1. Método normal: Dentro del IDE, nos dirigimos a Sketch->Importar librería->Ad library…. Se nos abrirá una ventana de exploración en la que debemos buscar el archivo rar/zip que hemos descargado. Lo seleccionamos y ya la tendremos instalada.

ACCEDE AHORA A LOS MEJORES CURSOS

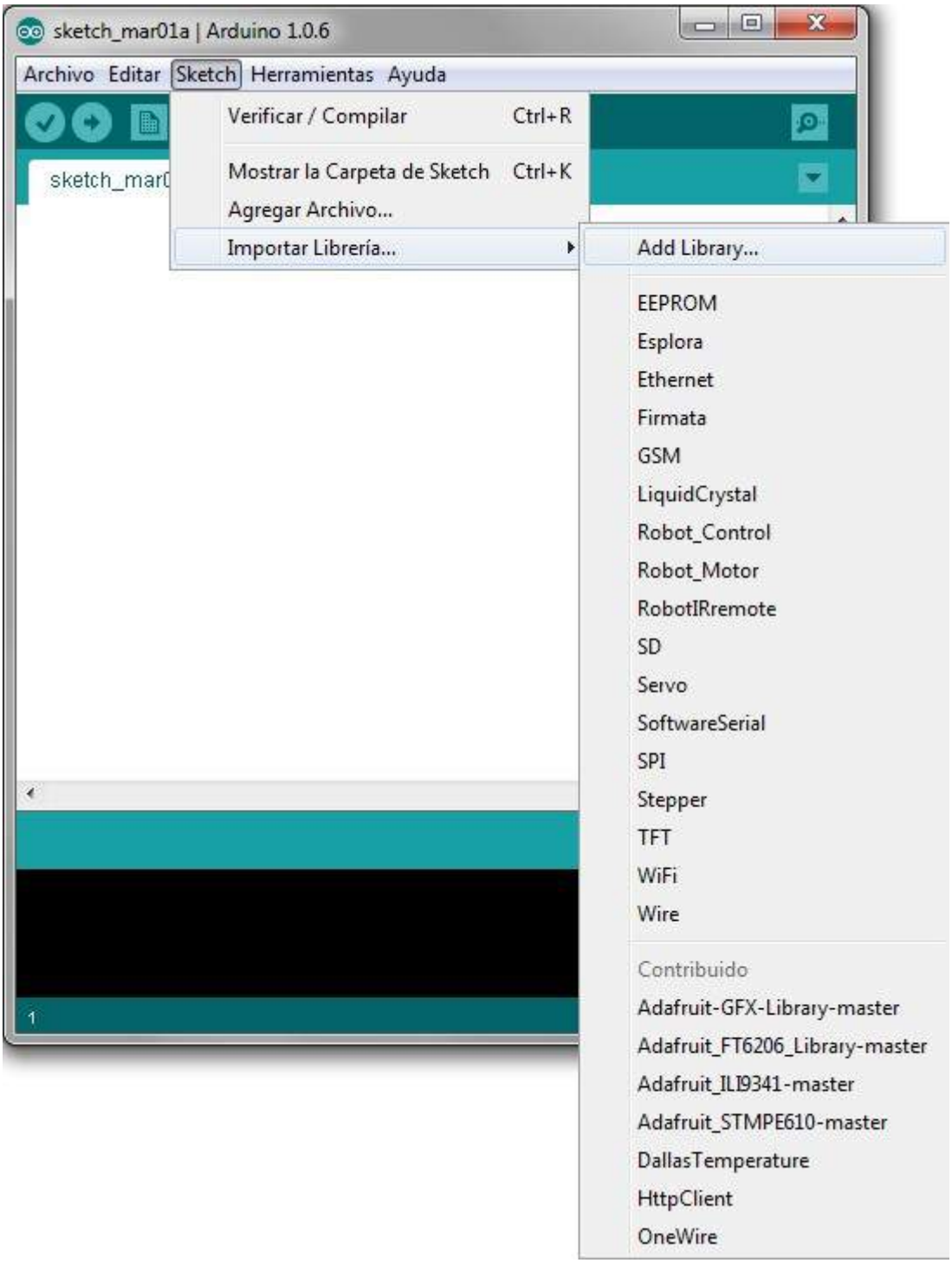


Ilustración 2 Instalación de librerías en IDE Arduino

ACCEDE AHORA A LOS MEJORES CURSOS

2. Método manual: Si el método anterior no funciona, utilizaremos la instalación manual. Para proceder, descomprimiremos el rar/zip descargado y copiamos la carpeta de la librería en la ruta Documentos\Arduino\libraries . Quedando, por ejemplo, de la siguiente manera.

ACCEDE AHORA A LOS MEJORES CURSOS

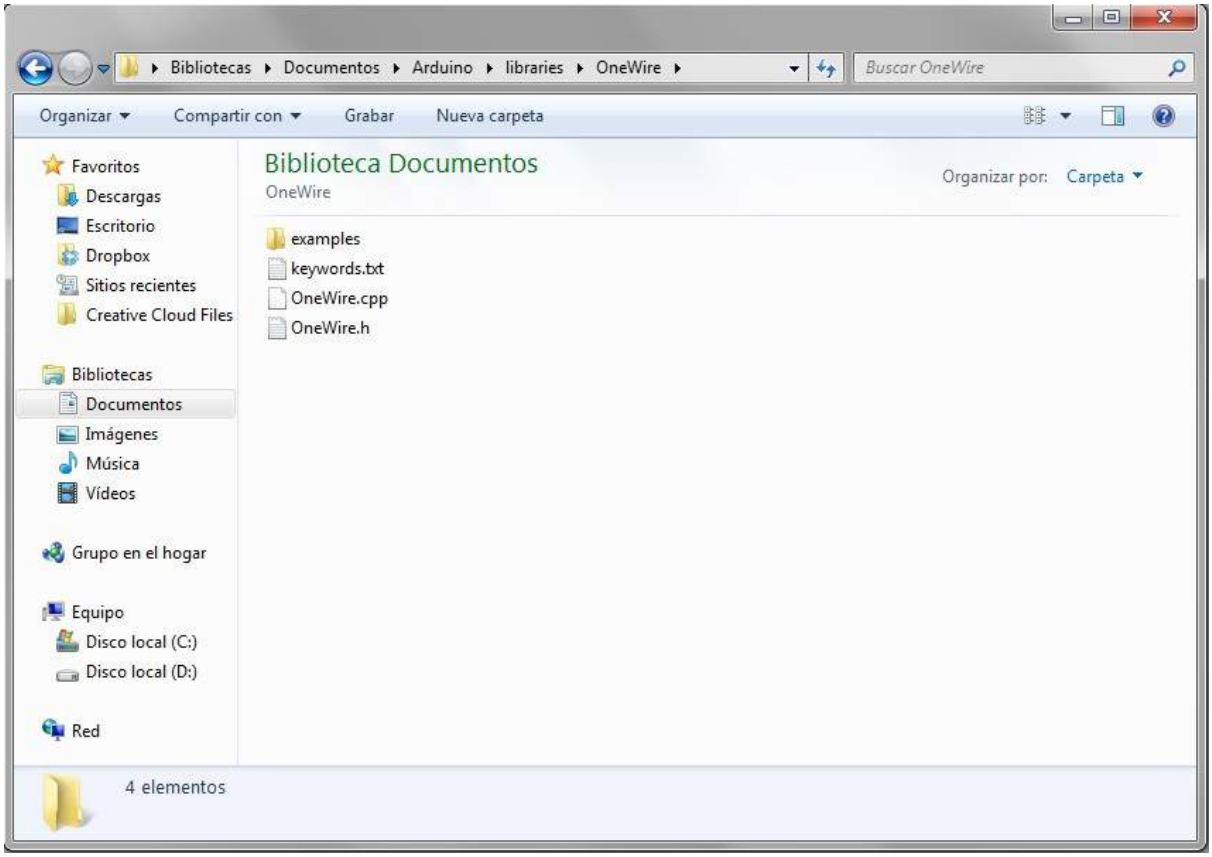


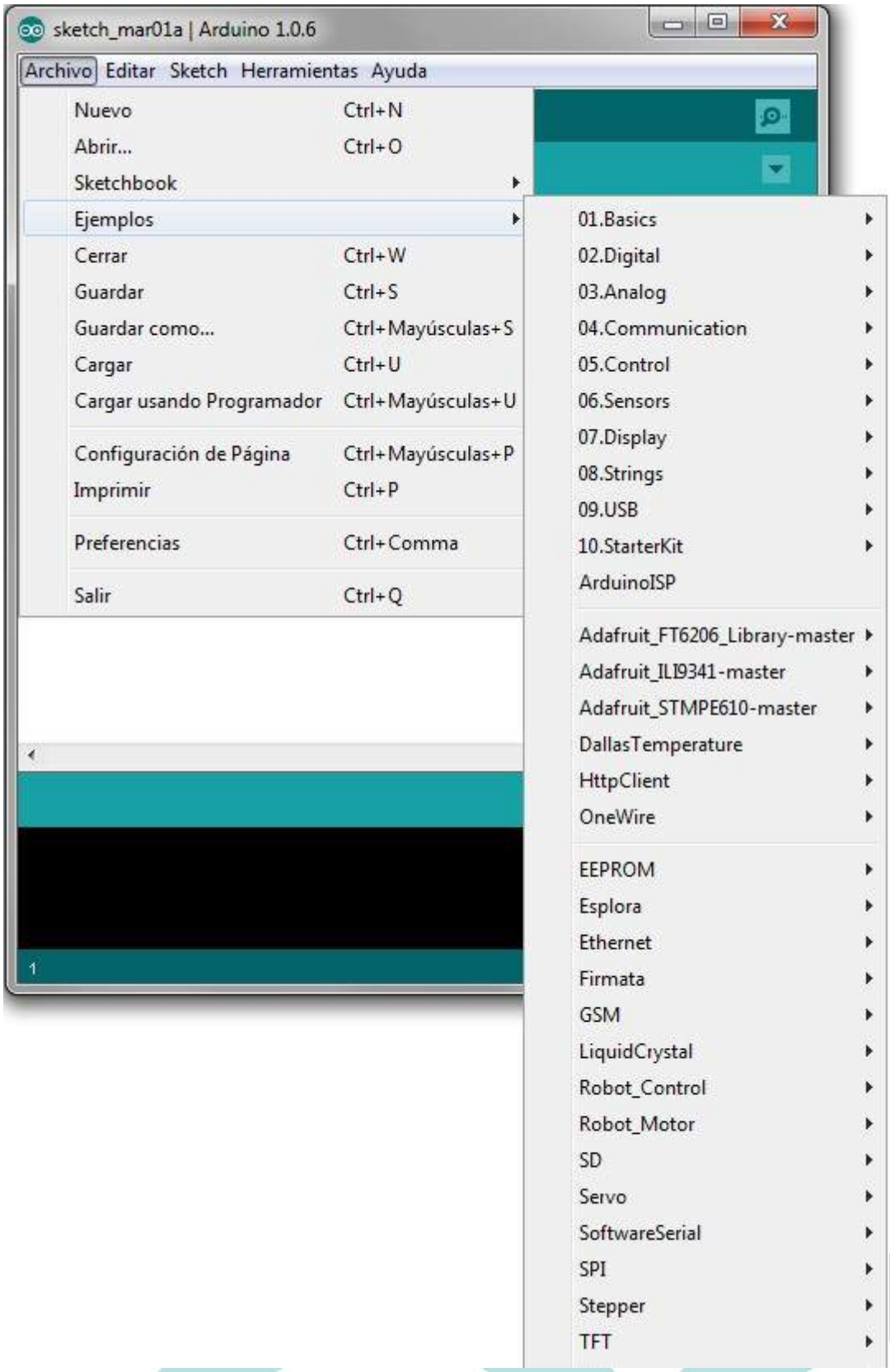
Ilustración 3 Ejemplo de carpeta de librería instalada

Una vez instaladas las librerías que vayamos a usar, podremos usar los ejemplos que traen cada una para probar sus accesorios. Es interesante echarles un ojo antes de ponernos a programar como locos, porque de estos ejemplos podemos

ACCEDE AHORA A LOS MEJORES CURSOS

aprovechar partes de código o ver cómo usa las funciones… y así ahorrarnos mucho trabajo. Para abrir estos ejemplos solo tendremos que dirigirnos dentro del IDE a Archivo->Ejemplos y ahí veremos todas las librerías disponibles y dentro de cada una, sus ejemplos correspondientes.

ACCEDE AHORA A LOS MEJORES CURSOS



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 4 Cargar ejemplos

Ejemplo: Medición de temperatura con sensor ds18b20

Para empezar con este ejemplo detallaremos el material necesario.

- 1 x Arduino UNO R3
- 1 x Protoboard
- 1 x Sensor de Temperatura ds18b20 (versión sumergible)
- 1 x Resistencia de 4,7kΩ
- Cables para conectar todo

En esta ocasión, es muy importante que prestéis atención al montaje del circuito. No es para nada difícil, pero si no está correcto el sensor dará medidas erróneas o incluso puede estropearse. El esquema es el siguiente.

ACCEDE AHORA A LOS MEJORES CURSOS

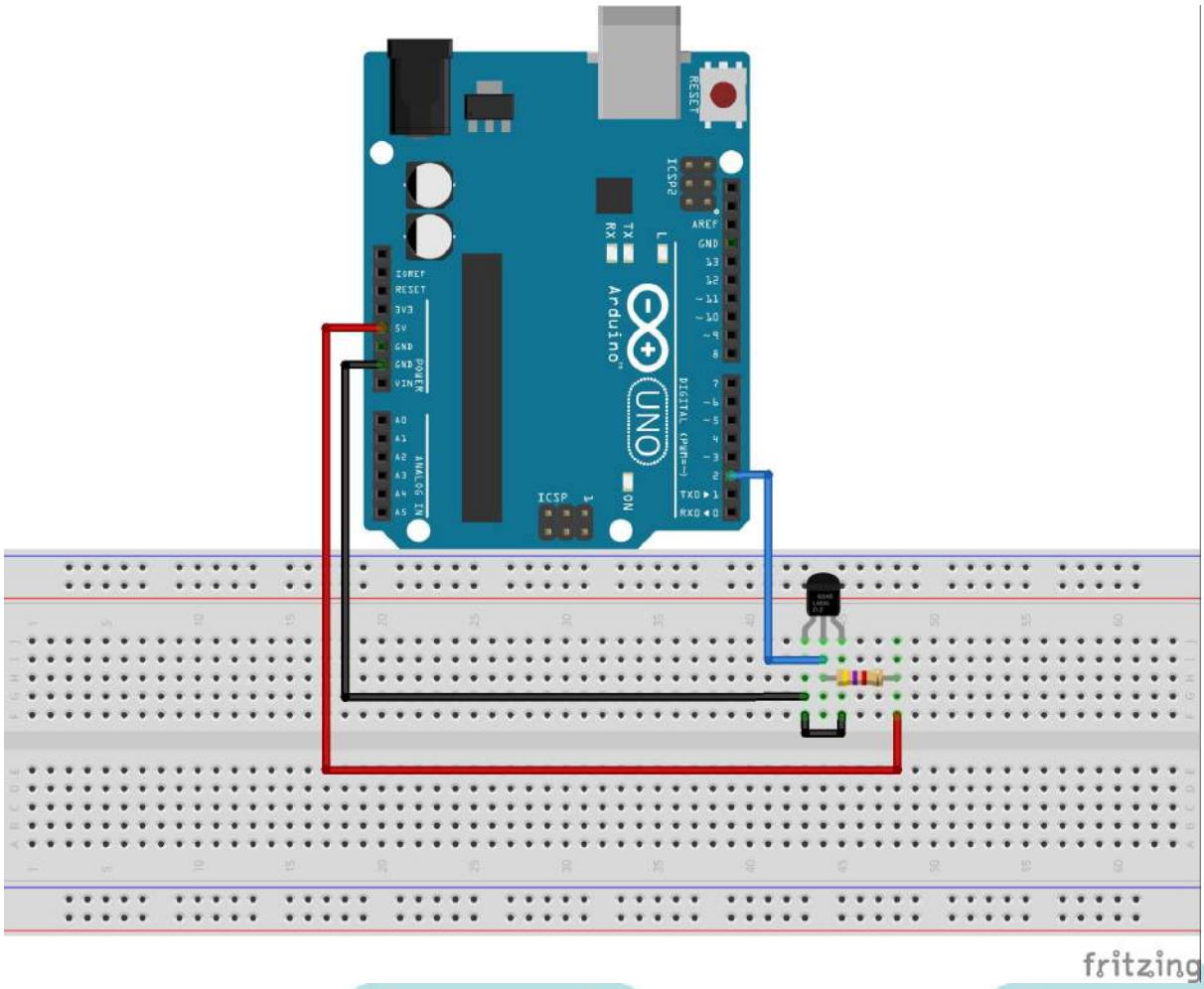


Ilustración 5 Montaje para el sensor ds18b20

ACCEDE AHORA A LOS MEJORES CURSOS

Como observamos en el circuito, alimentaremos a través del pin DATA, por medio de la resistencia de 4,7kΩ, y los pines VCC y GND van conectados entre sí y a tierra. Aquí en el datasheet del sensor podéis ver cómo va el patillaje, o bien si usáis la versión sumergible, como nosotros, la correspondencia de los 3 cables sería: [Rojo]=VCC, [Azul o Negro] = GND y [Amarillo o Blanco] = DATA.
Una vez conectado todo correctamente, nos dirigiremos al IDE y escribiremos el siguiente código.

```
/*****************************/
/*  Medir Temperatura ds18b20 */
/*****************************/

/*** Librerías ***/

#include <OneWire.h>                    //Se importan las librerías
#include <DallasTemperature.h>

//** Definiciones **//

#define Pin 2                           //Se declara el pin donde se conectará
la DATA

OneWire ourWire(Pin);                   //Se establece el pin declarado como bus
para la comunicación OneWire

DallasTemperature sensors(&ourWire); //Se llama a la librería DallasTemperature

//** Programa **//

void setup() {
delay(1000);
Serial.begin(9600);
sensors.begin();                        //Se inician los sensores
```

ACCEDE AHORA A LOS MEJORES CURSOS

```
}

void loop() {
sensors.requestTemperatures();         //Prepara el sensor para la lectura

Serial.print(sensors.getTempCByIndex(0)); //Se lee e imprime la temperatura en
grados Centigrados
Serial.println(" Grados Centigrados");

delay(1000);                           //Se provoca una parada de 1 segundo
antes de la próxima lectura

}
```

(Panama hiteck)

En el programa, simplemente comenzaremos importando las librerías, OneWire y DallasTemperature, de la forma que comentábamos más arriba. Continuaremos definiendo el pin que utilizaremos como bus de datos. Para ello usaremos la sentencia #define, la cual nos permite crear un valor constante pero con la ventaja de no consumir la RAM que usaría, al compilar, la creación de una variable. Hay que tener la precaución de escribir correctamente las definiciones por que pueden provocar el lastrado de errores a lo largo de todo el código si no se hace bien. En nuestro caso asignaremos el pin 2 para el bus.

Con la línea OneWire ourWire(Pin), le decimos al sensor a través de que pin se debe comunicar con Arduino y con DallasTemperature sensors(&ourWire) llamaremos a la librería de Dallas para que interprete los datos enviados y recibidos por nuestro bus.

Seguidamente, en nuestra función de setup, iniciaremos la comunicación serial a 9600 bits/seg y con la sentencia sensors.begin() activaremos los sensores que estemos usando.

Nuestro código en bucle, lo que hará será primero preparar al sensor ds18b20 para medir, usando la función sensors.requestTemperatures() y con la

ACCEDE AHORA A LOS MEJORES CURSOS

líneaSerial.print(sensors.getTempCByIndex(0)) tomará la medida y la imprimirá, por el monitor serial y en grados centígrados, en una única sentencia. Finalmente añadiremos la impresión del texto “Grados Centígrados” y haremos una pausa de 1 segundo para que el sensor no esté realizando miles de medidas constantemente.

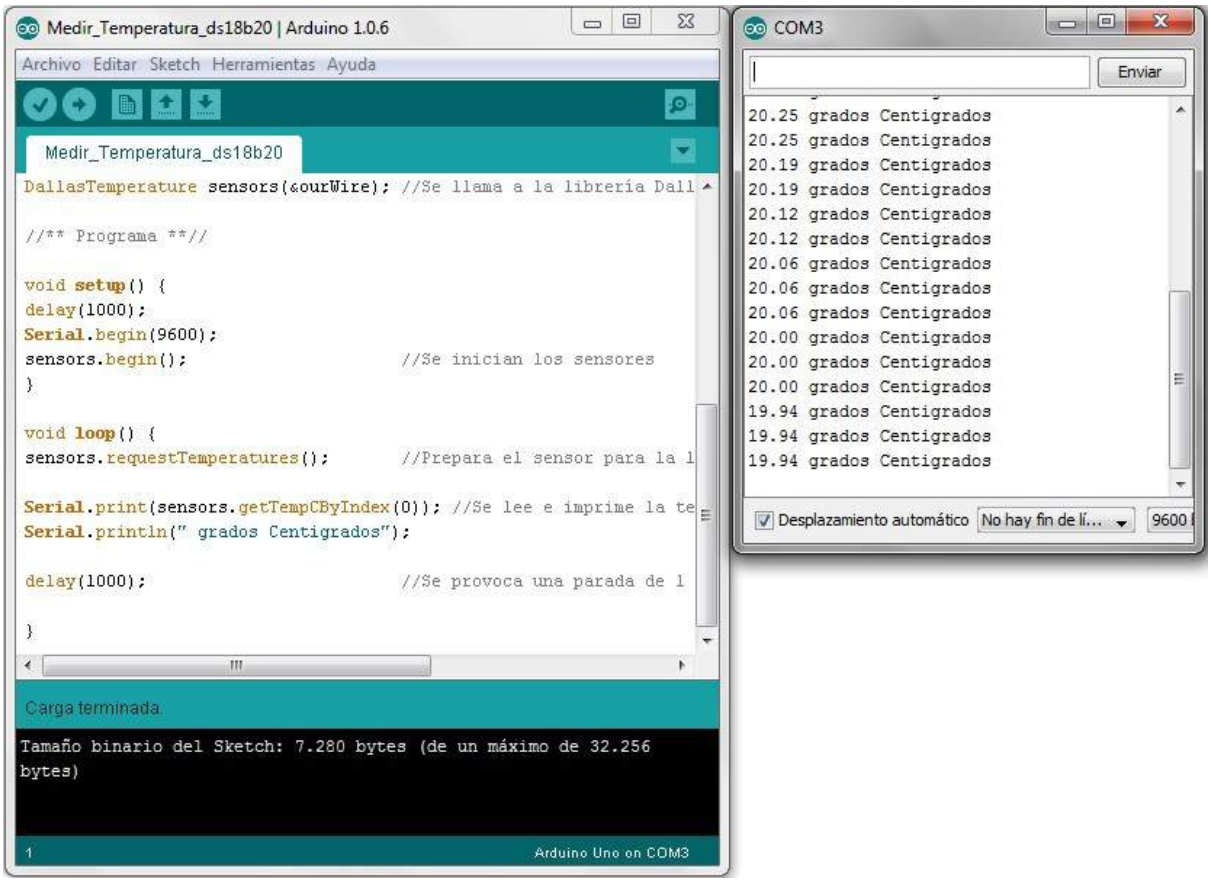
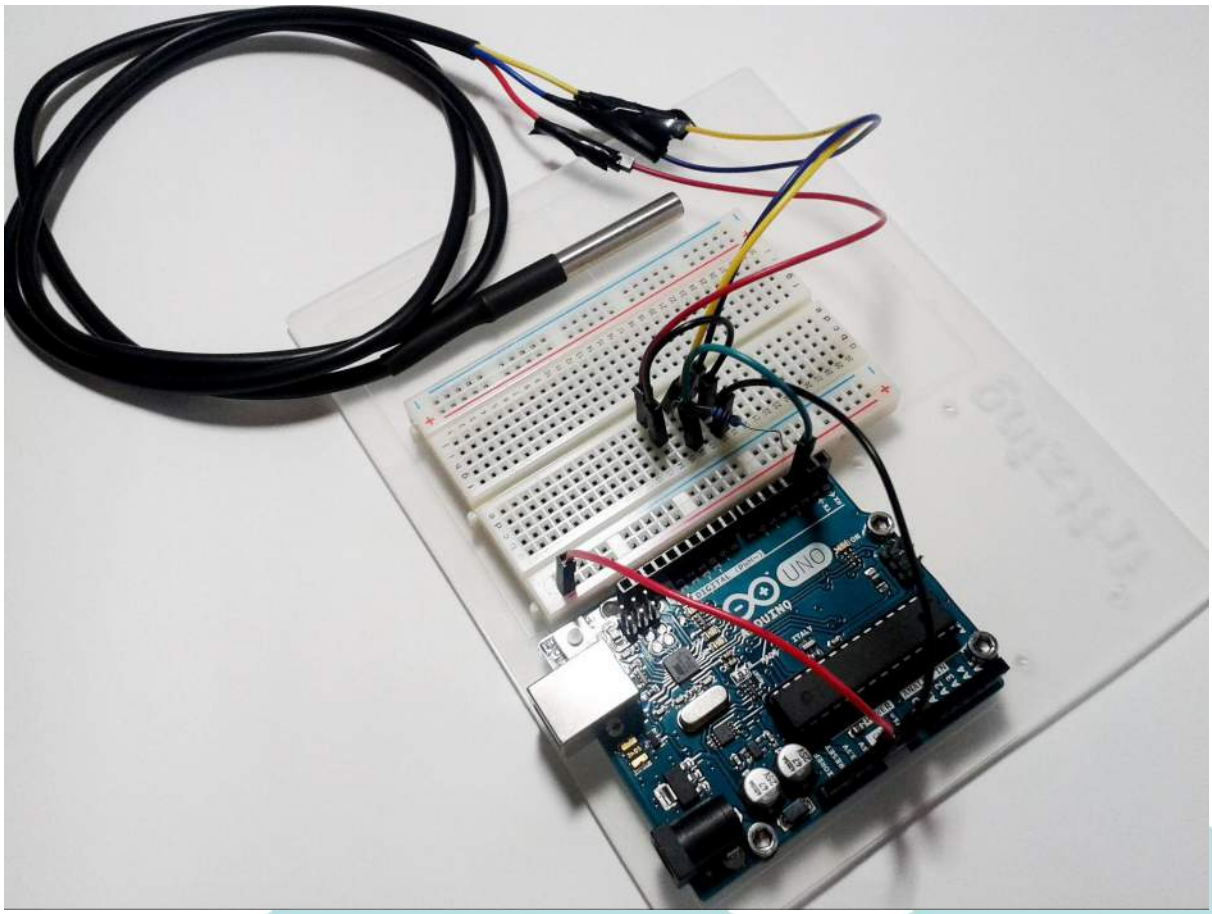


Ilustración 6 Visualización de la temperatura en el monitor serial

Como habéis podido comprobar, es muy sencillo y práctico utilizar este sistema. Solo hay que familiarizarse con las funciones específicas de la librería de cada sensor para sacarles el máximo provecho. Os recomendamos que leáis que funciones específicas trae cada librería y juguéis un poco con ellas antes de poneros

ACCEDE AHORA A LOS MEJORES CURSOS

a programar vuestro proyecto, seguro que encontraréis algunas funciones o ejemplos propios que os simplifican el código bastante.



ACCEDE AHORA A LOS MEJORES CURSOS

Ilustración 7 Montaje para el sensor ds18b20

Pantalla LCD

En este último post del primer tutorial veremos cómo mostrar a través de una pantalla LCD la temperatura medida con el sensor LM35 que ya aprendimos a utilizar en el post 5. ¿una pantalla LCD? Sí, pero no la confundamos con la pantalla de nuestra televisión o la de nuestro smartphone, esas son pantallas LCD-TFT,LCD-SPI... nosotros usaremos una LCD simple y a un sólo color del tipo de las que podemos encontrar en calculadoras y algunos electrodomésticos.

Material necesario

Suponiendo que ya tenemos nuestra placa Arduino Uno, nuestra protoboard y un juego de cables, tendremos que hacernos con el siguiente material:

- Pantalla LCD. En este caso he elegido una de 16 columnas y 2 líneas (16x2) por ser probablemente la más común. En tiendas online las he visto por unos 4 € envío incluido.
- Sensor LM35. Si seguiste el post 5 probablemente ya lo tendrás. Si no tampoco te preocupes porque no valdrá más de 2€.
- Potenciómetro 10k. Lo necesitamos para ajustar el contraste de a pantalla. Sin él sería ilegible. Cuesta pocos céntimos.

Montaje

Viendo la foto puede parecer complicado por la cantidad de cables que tiene, pero en realidad tener más conexiones no lo hace más difícil. Si hacemos las conexiones paso a paso nos funcionará a la primera. La que vamos a ver no es la única forma de conectar la pantalla, pero así podemos utilizar los códigos de ejemplo de la IDE para

ACCEDE AHORA A LOS MEJORES CURSOS

En OpenWebinars.net tenemos todos estos cursos a tu entera disposición

Linux LPIC-1
Examen 101

Linux LPIC-1
Examen 102

NodeJS,
ExpressJS y
MongoDB

AngularJS y
TypeScript

Apps móviles con
PhoneGap

Servidores VoIP
con Asterisk

Desarrollo
Frontend
Profesional

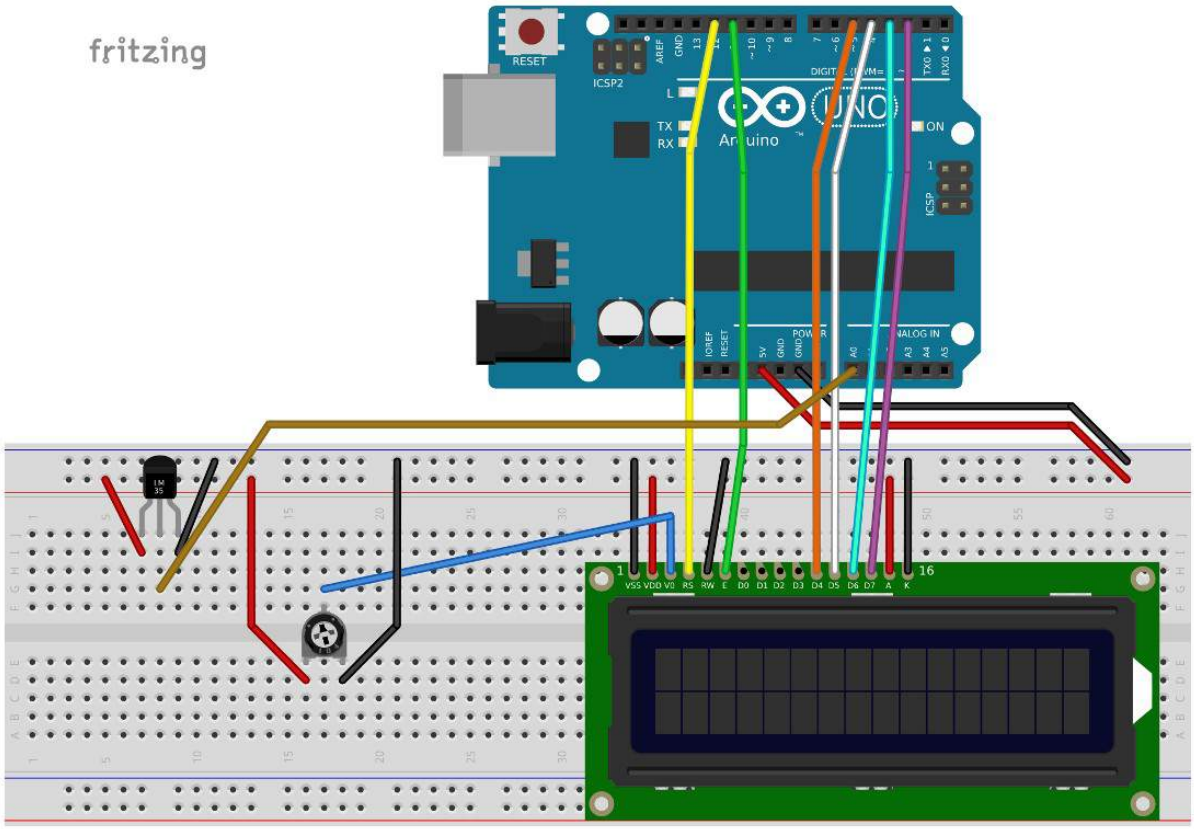
Virtualización de
Servidores con
Promox

Apps Móviles con
Titanium Alloy

Desarrollo
Backend con
Django

ACCEDE AHORA A LOS MEJORES CURSOS

LCD sin cambiar ninguna línea. También tenéis que tener en cuenta que las pantallas las suelen vender con el agujero para soldarle un pin macho o un cable. Yo le he soldado una tira de pines macho para poder insertarla en la protoboard pero si no queréis soldar podéis meter los cables en los agujero con cuidado de que las puntas no se toquen entre sí. Ahora vamos a ver de izquierda a derecha como se conecta cada pin de la pantalla. Como los pines 5V y GND los vamos a utilizar en más de una conexión, los conectamos a las filas largas de nuestra protoboard y cuando un pin de la pantalla se corresponda con 5V o GND lo conectaremos con su fila correspondiente.

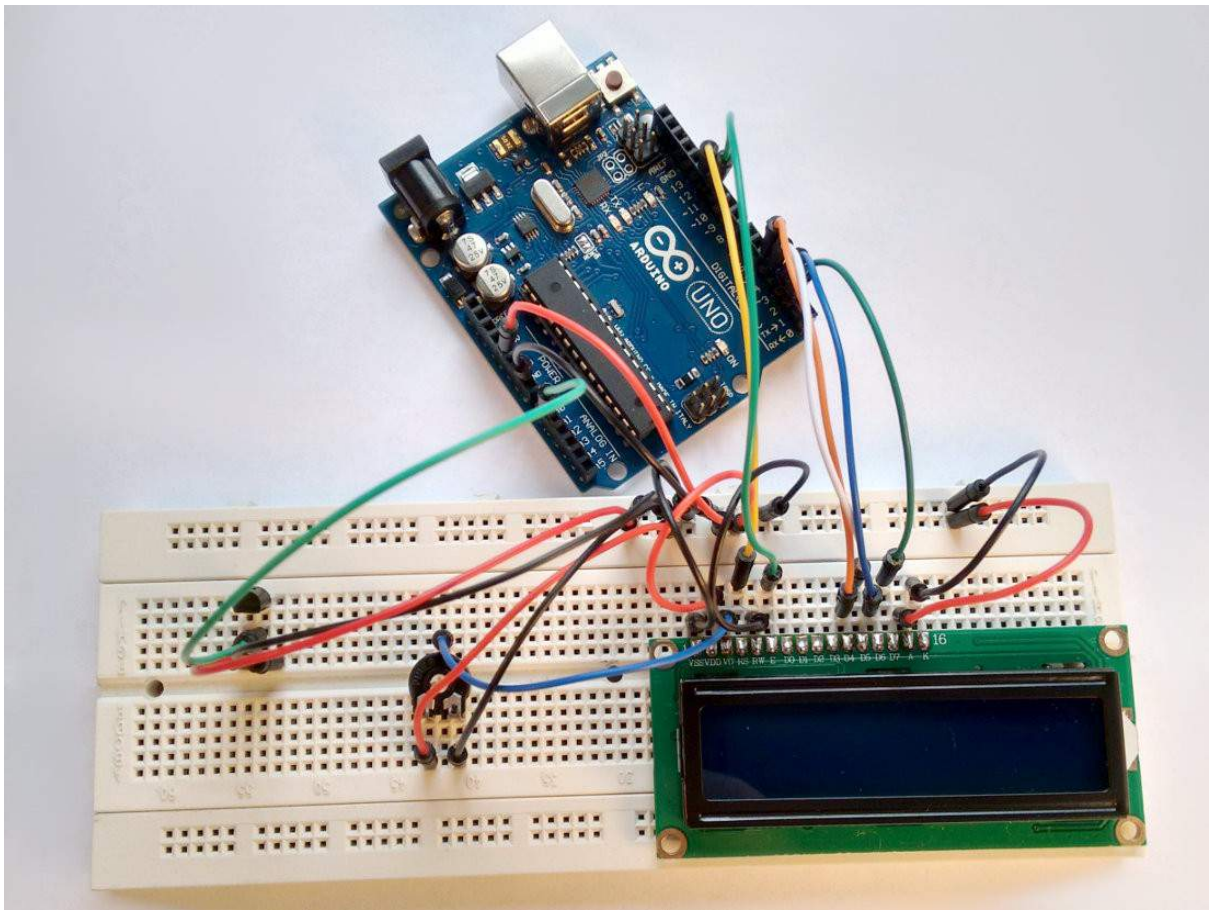


- VSS a la fila GND
- VDD a la fila 5 V
- V0 a la pata central del potenciómetro

ACCEDE AHORA A LOS MEJORES CURSOS

- RS al pin 12
- RW a la fila GND
- E al pin 11
- D0 a D3 sin conexión
- D4 al pin 5
- D5 al pin 4
- D6 al pin 3
- D7 al pin 2
- A a la fila 5 V
- K a la fila GND
- Del potenciómetro conectamos una de las patas libres a GND y otra a 5 V
- Pata VS del sensor a la fila 5 V
- Pata central del sensor al pin A0
- Pata GND del sensor a la fila GND
- Si la pantalla no tiene iluminación prescindiremos de los pines A y K

ACCEDE AHORA A LOS MEJORES CURSOS



No debemos olvidar que la primera vez que se conecta, tras alimentarla debemos ajustar el potenciómetro hasta encontrar el contraste óptimo. Antes de seguir podemos verificar que funciona perfectamente cargando uno de los códigos de ejemplo de la IDE como el que encontramos en Archivo → Ejemplos → LiquidCrystal → HelloWorld.

Código

Sólo vamos a tratar en profundidad la parte del código que tiene que ver con el LCD y no con el sensor ya que está visto anteriormente. Al final del post está el código

ACCEDE AHORA A LOS MEJORES CURSOS

completo. En primer lugar teneos que añadir la librería para LCD en Esketch → Importar librería → LiquidCrystal o escribiendo

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

Como la configuración de algunos pines puede variar tenemos que indicarle a la librería cuáles hemos utilizado.

```
LiquidCrysta lcd(12,11,5,4,3,2);
```

Ya podemos pasar a la función setup. Ahora iniciamos el LCD.

```
lcd.begin(6,2);
```

El 16 y el 2 se refiere al número de columnas y filas de nuestra pantalla. Lo siguiente es pasar a la función loop e indicar en qué posición de la pantalla queremos el cursor. De manera genérica, el comando es

```
lcd.setCursor(columna, fila);
```

Pero ojo, empieza a contar desde cero, es decir, si queremos el cursor en la primera columna de la primera fila tendremos que escribir

```
let.setCursor(0,0)
```

Una vez situado el cursor escribimos por pantalla la cadena de caracteres “Temperatura:”. Las cadenas de caracteres van entre comillas.

```
lcd.print(“Temperatura:”);
```

Suponiendo que ya hemos leído la temperatura y la tenemos almacenada en una variable llamadatemp falta mostrarla por pantalla. Para que quepa la escribimos en la primera columna de la segunda fila

```
lcd.setCursor(0, 1);

lcd.print(temp);
```

Y con esto escribirá el número que almacena la variable temp. Para que nuestro proyecto quede más estético escribiremos la cadena de caracteres “C” en la octava columna de la segunda fila (el símbolo “º” no se muestra correctamente).

```
lcd.setCursor(7, 1);

lcd.print(“C”);
```

ACCEDE AHORA A LOS MEJORES CURSOS

OpenWebinars

Aunque es este código no vamos a tener problemas al sobrescribir, tenemos que tener en cuenta que sólo se borran las posiciones en las que se escriben, el resto sigue manteniendo los caracteres que mostraban anteriormente. Para limpiar la pantalla utilizaremos la opción clear que borrará toda la pantalla.

```
lcd.clear();
```

Por último, introducimos un delay al gusto para evitar que refresque tan rápido que no nos de tiempo a leer los datos antes de que cambien.

ACCEDE AHORA A LOS MEJORES CURSOS

En OpenWebinars.net tenemos todos estos cursos a tu entera disposición

Linux LPIC-1
Examen 101

Linux LPIC-1
Examen 102

NodeJS,
ExpressJS y
MongoDB

AngularJS y
TypeScript

Apps móviles con
PhoneGap

Servidores VoIP
con Asterisk

Desarrollo
Frontend
Profesional

Virtualización de
Servidores con
Promox

Apps Móviles con
Titanium Alloy

Desarrollo
Backend con
Django

ACCEDE AHORA A LOS MEJORES CURSOS